Artificial Physical Chemistry: Analysis and Design of Networking and Communication Systems

Inauguraldissertation

zur
Erlangung der Würde eines Doktors der Philosophie
vorgelegt der
Philosophisch-Naturwissenschaftlichen Fakultät
der Universität Basel (Schweiz)
und der
Universität Pisa (Italien)

von
Massimo Monti
aus Italien

Basel, 2014

Genehmigt von der Philosophisch-Naturwissenschaftlichen Fakultät

Auf Antrag von

Prof. Dr. Christian F. Tschudin,
Universität Basel, Schweiz
(Fakultätsverantwortlicher)

Prof. Dr. Marco Luise,
Università di Pisa, Italia
(Dissertationsleiter)

Prof. Dr. Edouard Bugnion,
École Polytechnique Fédérale De Lausanne (EPFL), Suisse
(Korreferent)

Prof. Dr. Mérouane Debbah,
Grande École Supèrieure d'Électricité (SUPÉLEC), France
(Korreferent)

Basel, den 16. 09. 2014

Prof. Dr. Jörg Schibler
(Dekan)

To my wife, Elena

Abstract

In this thesis, we use concepts, principles, and theoretical results from Physical Chemistry to engineer communication and networking systems. We focus on system dynamics and exploit laws from chemical kinetics in order to govern the dynamics of a communication network. This is achieved by orchestrating the interactions among network nodes by means of "artificial chemistries". We provide a new perspective on traditional issues concerning the design, the formal analysis and the deployment of distributed algorithms and communication protocols, ultimately leading to programmable network dynamics with provable properties.

Specifically, (*i*) we introduce a class of chemistry-inspired flow controllers that can easily be customized to accommodate many requirements of network (resource) management, such as distributed coordination of flow aggregates, capacity allocation, access regulations and service differentiation among user flows or flow bundles. (*ii*) We show the benefit of the chemical approach in designing solutions to the "distributed consensus problem" for wireless sensor networks. After having designed and analyzed the required interaction rules "on paper", we use the derived communication protocol in a hardware testbed. Salient features of this minimalistic setup are mainly three. Nodes achieve consensus based only on asynchronously emitted RF pulses. No media access control is used. The protocol works in an embodied fashion by exploiting subtle timing differences and without recurring to symbolic information. (*iii*) Finally, in order to demonstrate the use of Chemistry-driven mechanisms also for high-performance tasks (e.g., high-rate packet-pacing), we describe the implementation of artificial chemistries on an FPGA-based hardware. At the same time, we provide an abstraction for designing runtime-programmable hardware controllers.

Keywords: Artificial Chemistry, Emergent Control, Consensus, Convergence and Sensitivity Analysis, Algorithm Dynamics, Hardware Implementation.

Funding: This doctoral work has been supported by the Swiss National Science Foundation (SNSF) under the grant #132525.

Co-tutelle: This doctoral work was regulated by the co-supervision agreement between the University of Basel (Switzerland) and University of Pisa (Italy).

Acknowledgments

This work would not have been possible without the support of many people. First, I would like to thank my advisers Prof. *Christian Tschudin* and Prof. *Marco Luise* for their respectful support, and the possibility they gave me to conduct my own research in complete freedom.
A sincere gratitude goes to Dr. *Manolis Sifalakis* and Dr. *Thomas Meyer* that co-supervised this project in its different stages. Discussions as well as collaborations with them have always revealed productive, and brought to works that later the research community has recognized as valid and promising.
I also thank all my colleagues at the Computer Network group, University of Basel, for their advice and suggestions during these years, in particular Dr. *Pierre Imai*.
A part of this thesis has been developed during my stay at the University of Pisa, Dept. Information Engineering. I am very grateful to all my colleagues at the DSP Communication Lab, particularly, to Dr. *Giacomo Bacci*, Dr. *Andrea Emmanuele* and Dr. *Mario di Dio* for their support, and to Dr. *Luca Sanguinetti*, who considerably helped me in part of this work.
A special thanks is reserved to my wife, *Elena Puri*, who has always supported my work and encouraged me in good and bad days. I thank her for the valid help she gave me in writing and revising all publications, including this thesis, and the help in everyday life in Basel.
Last but not least, I would like to thank my mother *Pierangela* for her confidence in me, and my father *Mario* for the continuous consultancy he gave me in the context of hardware design and implementation.

Basel, 2014 Massimo Monti

Contents

List of Publications

International Journals

- Massimo Monti and Manolis Sifalakis and Christian F. Tschudin and Marco Luise, "*Towards Programmable Network Dynamics: A Chemistry-inspired Abstraction for Hardware Design*," IEEE/ACM Transactions on Networking, submitted for review, May 2014.
- Massimo Monti and Luca Sanguinetti and Christian F. Tschudin and Marco Luise, "*A Chemistry-Inspired Framework for Achieving Consensus in Wireless Sensor Networks*," IEEE Sensors Journal, vol. 14, no. 2, pp. 371–382, Feb 2014.
- Massimo Monti and Pierre Imai and Christian F. Tschudin, "*Designing Run-time Environments to have Predefined Global Dynamics*," International Journal of Computer Networks and Communications (IJCNC), vol. 5, no. 3, pp. 1–16, May 2013
- Massimo Monti and Thomas Meyer and Christian F. Tschudin and Marco Luise, "*Stability and Sensitivity Analysis of Traffic-Shaping Algorithms inspired by Chemical Engineering*," IEEE Journal on Selected Areas of Communications (JSAC) Special Issue on Network Science, vol. 31, no. 6, pp. 1–11, Jun 2013

International Conferences and Workshops

- Manolis Sifalakis, Stefan Braun, Massimo Monti and Christian F.Tschudin, "*An Empirical Study of Receiver-based AIMD Flow-Control Strategies for CCN*," In Proc. of IEEE International Conference on Computer Communications and Networks (ICCCN) (NACSD-track), pp. 1–8, Nassau, Bahamas, Jul 2013.
- Massimo Monti and Luca Sanguinetti and Christian F. Tschudin and Marco Luise", "*Chemistry-Inspired Algorithm for Emergent Distributed Consensus in WSNs*," in Proc. of the IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS), pp. 423–429, Cambridge (MA), USA, May 2013.

- Massimo Monti and Manolis Sifalakis and Thomas Meyer and Christian F. Tschudin and Marco Luise, "*A Chemical-Inspired Approach to Design Distributed Rate Controllers for Packet Networks*," in Proc. of the IFIP/IEEE-IM International Workshop on Distributed Autonomous Network Management Systems (DANMS), pp. 1358–1364, Ghent, Belgium, May 2013.
(*Best Paper Award*)

- Stefan Braun, Massimo Monti, Manolis Sifalakis, Christian F. Tschudin "*CCN and TCP co-existence in the Future Internet: Should CCN be compatible to TCP?*," in Proc. of the IFIP/IEEE-IM International Workshop on Management of the future Internet (ManFI), pp. 1109–1115, Ghent, Belgium, May 2013.

- Massimo Monti and Thomas Meyer and Christian F. Tschudin and Marco Luise, "*Signal Processing Applied to Chemically Inspired Communication Protocols*," In proc. of IEEE International Conference On Communications (ICC) (CQR-track), pp. 1144–1148, Ottawa, Canada, Jun 2012.

- Igor Talzi, Massimo Monti, Thomas Meyer and Christian F. Tschudin, "*Force-Based Navigation in Wireless Sensornets*," in Proc. of the IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS), pp. 1–6, Barcelona, Spain, Jun 2011.

Others

- Stefan Braun and Massimo Monti and Manolis Sifalakis and Christian F. Tschudin, "*CCN/TCP co-existence in the future Internet: E2E vs. Receiver-based flow control*," Poster at the IEEE INFOCOM Workshop on Emerging Design Choices in Name-Oriented Networking (NOMEN), Turin, Italy, Apr 2013.

- Christian F. Tschudin and Manolis Sifalakis and Pierre Imai and Massimo Monti and Thomas Meyer, "*CCN-lite - A minimal code base for CCNx*," Poster at the CCNxCon12 community meeting, Nice, France, Sep 2012.

- Manolis Sifalakis and Massimo Monti and Christian F. Tschudin, "*Embedding Cloud-Centric-Networking in CCN*," Presentation at the CCNxCon12 community meeting, Nice, France, Sep 2012.

- Stefan Braun and Manolis Sifalakis and Massimo Monti and Christian F. Tschudin, "*Steaming up CCN against TCP*," Presentation at the CCNxCon12 community meeting, Nice, France, Sep 2012.

- Massimo Monti and Thomas Meyer and Christian F. Tschudin and Marco Luise, "*Simulating and Exploiting Physical Forces in Computer Networking*," Poster for the Student Competition at the International Workshop on Self-Organizing Systems, Karlsruhe, Germany, Feb 2011.

List of Figures

List of Tables

List of Algorithms

List of Acronyms

2MA	two moment approximation
ACK	acknowledgment
AIMD	additive increase multiplicative decrease
ANN	artificial neural network
AQM	active queue management
ARP	address resolution protocol
ASIC	application-specific integrated circuit
AWGN	additive white Gaussian noise
BDP	bandwidth-delay product
BIBO	bounded-input bounded-output
BIC	binary increase congestion
BpMol	bytes per molecule
BR	broadcast gossip algorithm
CDF	cumulative distribution function
CLE	chemical Langevin equation
CME	chemical master equation
CDN	content delivery networks
CH	chemical consensus algorithm
CLB	configurable logic blocks
CoDel	controlled delay
CW	continuous wave
CSP	content service provider
CPLD	complex programmable logic devices
CPU	central processing unit
DCCP	datagram congestion control protocol
DCTCP	data center TCP
CSMA	carrier sense multiple access
DRAM	dynamic RAM
DRED	dynamic RED
DRR	deficit round robin
DSACK	duplicate selective ACK

DSL	digital subscriber line
DTRC	distributed traffic rate controller
ECN	explicit congestion notification
EM	electromagnetic
EMC	EM compatibility
EMI	EM interference
FCFS	first come first served
FF	flip-flop
FPGA	field-programmable gate array
FPS	flow proportional sharing
FQ	fair queueing
GPS	generalized processor sharing
GRED	gentle RED
HDL	hardware description language
HS-TCP	high speed TCP
HULL	high-bandwidth ultra-low latency architecture
IEEE	institute of electrical and electronics engineers
i.i.d.	independent and identically distributed
IP	internet protocol suite
IP	intellectual property cores
ISE	integrated software environment
ISM	industrial, scientific and medical radio bands
LoMA	law of mass action
LNA	linear noise approximation
LSB	least significant bit
LTI	linear time-invariant
LUT	look-up table
MAC	media access control
MCA	metabolic control analysis
MSB	most significant bit
MSE	mean squared error
NIC	network interface controller
NMSE	normalized mean squared error
PCI	peripheral component interconnect
PDF	probability density function
PIE	proportional integral controller enhanced
OSI	open systems interconnection
ODE	ordinary differential equation
QoS	quality of service
R&D	research and development

RAM	random-access memory
RCP	rate control protocol
RED	random early detection
RN	randomized gossip algorithm
ROM	read-only memory
RR	round robin
RR-TCP	reordering-robust TCP
RTL	register transfer level
RTO	retransmission timeout
RTT	round trip time
SACK	selective ACK
SCTP	the stream control transmission protocol
SFED	selective fair early detection
SFQ	stochastic fairness queueing
SHRED	short-lived flow friendly RED
SNR	signal-to-noise ratio
SRAM	static RAM
SRED	stabilized RED
TCP	transmission control protocol
TB	token bucket
TF	transfer function
TM	traffic management
UART	universal asynchronous receiver/transmitter
UDP	user datagram protocol
USB	universal serial bus
UWB	ultra-wide band
VHDL	VHSIC HDL
VHSIC	Very High Speed Integrated Circuit
WFQ	weighted FQ
WLAN	wireless local area network
WSN	wireless sensor network
WRR	weighted round robin

List of Symbols and Operators

Chemistry-related symbols

artificial chemistry	$\mathcal{AC}$
distributed artificial chemistry	$\mathcal{DAC}$
reaction algorithm	$\mathcal{A}$
reaction set	$\mathcal{R}$
species set	$\mathcal{S}$
local species set at node i	$\mathcal{S}_i$
species set at node i comprehensive of species at neighbors	$\mathcal{M}_i$
reaction coefficient (rate constant) of reaction r	k_r
reaction coefficient (rate constant) vector	$\mathbf{k}$
molecular concentration of species s	c_{s}
steady state of species s	c_s^*
molecular concentration vector	$\mathbf{c}$
propensity of reaction r	a_r
reaction propensity vector	$\mathbf{a}$
rate of reaction r	v_r
reaction rate vector	$\mathbf{v}$
stoichiometric matrix	$\mathbf{\Xi}$
stoichiometric matrix elements	$\Xi_{s,r}$
stoichiometric reactant coefficient of species s and reaction r	$\alpha_{s,r}$
stoichiometric product coefficient of s and reaction r	$\beta_{s,r}$
order of reaction r	$\mathcal{O}(r)$
chemical system's state (vector of the amount of molecules)	$\mathbf{n}$
stochastic process related to the chemical system's state	$\mathbf{N}(\mathrm{t})$
chemical system's state change vector	$\mathbf{x}$
deficiency value of a reaction network	$\wp$
set of chemical complexes	$\mathcal{C}$
linkage class	ℓ
volume of the vessel	Ω

Communication-networking symbols

communication network graph	$\mathcal{G}$
set of communication network edges	$\mathcal{E}$
set of communication network vertexes	$\mathcal{V}$
ith communication network vertex (network node)	ν_i
communication network adjacency matrix	$\mathbf{\Delta}$
(average) arrival or generation rate	λ
sending rate	v_{out}
rate limit	v_{max}
expected waiting time (delay)	d
capacity of a media access system	R
average scheduling process of MAC algorithms	g
capacity-normalized average scheduling process	G
throughput of MAC algorithms	μ

Transient (sensitivity) analysis symbols

Sensitivity function	$\varepsilon(t)$
Dirac's delta function	$\delta(t)$
transfer function (TF) matrix in Laplace domain	$\mathbf{H}(s)$
transfer function (TF) in Laplace domain	$H(s)$
Laplace transform of function $f(t)$	$L\{f(t)\}$
inverse Laplace transform of function $F(s)$	$L^{-1}\{F(s)\}$
state matrix of an LTI system	$\mathbf{A}$
input matrix of an LTI system	$\mathbf{B}$
output matrix of an LTI system	$\mathbf{C}$
feedforward matrix of an LTI system	$\mathbf{D}$
identity matrix	$\mathbf{I}$
Jacobian matrix	$\mathbf{J}$
Laplacian matrix	$\mathbf{L}$
State vector	$\mathbf{c}$
Parameter vector	$\mathbf{p}$
Input vector	$\mathbf{u}$
Perturbation vector	$\mathbf{x}$
Output vector	$\mathbf{y}$

Set operator

subset	$\subseteq$
strict subset	$\subset$
superset	$\supseteq$
strict superset	$\supset$
union	$\cup$
element of	$\in$
exists	$\exists$
for all	$\forall$
size of set $\mathcal{A}$	$\lvert\mathcal{A}\rvert$

Consensus-related symbols

set of neighboring nodes of node ν_i	$\mathcal{N}_i$
measured quantity at node ν_i	z_i
z_i-difference between consecutive sampling	$\delta_{z,i}$
design parameter related to the estimation of neighbors ν_i	α
design parameter related to the self-stabilization mechanism	β
local sensor's state	S

Other symbols and operators

uniform distribution	$\mathcal{U}[a, b]$
normal distribution	$\mathcal{N}[\eta, \sigma]$
exponential distribution	$\exp[\eta]$
mean of a random variable v	η_v
standard deviation of a random variable v	σ_v
expected value of random variable x	$\mathbb{E}\{x\}$
probability	P
rank of matrix $\mathbf{X}$	$\text{rank}(\mathbf{X})$
minimum between two values x and y	$\min(x, y)$
proportional to	$\propto$
relation	$\mid$
number of	$\#$

Chapter 1

Introduction

"Science is what you know. Philosophy is what you don't know."

Bertrand Russell

THIS THESIS promotes the adoption of an unconventional approach for engineering communication and networking systems, which we believe can contribute to opening new directions in the research on this wide topic. We deviate from traditional techniques to venture on a tour into Physical Chemistry in search of breakthroughs on the design and the analysis of communication and networking systems.

1.1 Context and motivations

Predicting and controlling the dynamics of communication and networking systems are difficult but important challenges in the engineering process. While in the early days of communication networks the researchers' attention was on merely functional aspects, in the last three decades, the focus has shifted towards mastering system dynamics and designing stable solutions. The design process has thus been enriched by theoretical studies that care about predicting possible system's behaviors (trajectories) induced by the adopted algorithms and protocols.
In the past, researchers and engineers often mastered a problem by decomposition – they broke a problem and its solution into parts, and analyzed and implemented them by further iterations of decomposition. They cared about the algorithmic functionality but rarely about analyzability. Such a modular (layered) controllable environment was easy to maintain but difficult to optimize (cross-layer optimization). Also, it made interactions among the entities that constitute the system, or between the system and the environment, hardly predictable and controllable. As a result, algorithms and protocols, which worked well under nominal conditions, produced instead behaviors not fully

understood and exhibited shortcomings in evolving scenarios and unexpected conditions.

Today, the design of communication and networking protocols follows different approaches that are, more or less, empirical, rigorous/formal, or mimicking natural phenomena.

Empirical design

Many protocols currently in use represent mere adaptations of former solutions to newly faced problems, e.g. the progression of congestion-avoidance solutions in the Internet. Researchers and engineers have often patched and extended existing algorithms according to empirically-gained insights and without a theoretical ground truth. With the rapid evolution of communication environments and infrastructures, these solutions often deal with conditions that are different from those they have been engineered for. As a result, they turn out to barely work and exhibit incomprehensible behavior.

Analysis-based design

Engineers have proposed to use analytical tools such as control theory to understand critical aspects of algorithms and calibrate/configure them accordingly. The use of control theory has been further advocated in early stages of the design process to plan corrective actions that keep the system stable. A central challenge in this approach is deriving a (fluid-)model that approximates sufficiently well the algorithms dynamics and then reducing it to an analytically tractable system. Another problem not easy to be tackled with control-theoretical tools is the need for autonomic (self-regulating) systems – systems must be able to automate their management and continuously self-optimize their behavior in order to accommodate to the continuous evolution of the operating environment. This is a feature that seems conflicting with the need for dynamics controllability.

Nature-inspired design

Looking for self-* properties of systems, most recently, researchers resort to bottom-up (nature-inspired) design approaches. Nature-inspired algorithms are defined by simple (microscopic) interaction rules from which the behavior and functionality of the algorithm itself (naturally) emerges at the macroscopic level. Most of the time, these (distributed) algorithms tend to exhibit, and induce in the systems where they are implemented, stable dynamics that are easily predictable (at least in average terms); nature-inspired systems tend to be more robust against perturbations and to changes of nominal conditions. Aspiring to develop systems as Mother Nature does, proposed communication solutions use specific models describing processes and mechanisms in living organisms. To this end, researchers have to properly engineer the (microscopic) implementation details that correctly embody the mimicked model, and that let (macroscopic) functional requirements emerge from interactions within the system and between the system and the environment. This process turns out to be not always simple but mostly, it is not automated and requires different ploys from case to case.

Aim

What is missing, and seems to us important for developing future communication networks, is an intuitive design framework that offers analytical tools to understand, predict, and control network dynamics. We aim at an alternative to the classical finite-state-machine modeling of reactive systems that should

ease and make almost automated the design of stable interaction-systems; systems where functionalities are not distinct from the system itself but rather embodied in it and its environment. Communications should be organized in a way to autonomously adapt and evolve according to the evolution of the environment and working conditions.

1.2 Our approach

In this thesis, we propose to approach the design and the analysis of algorithms and protocols for communications and networking by exploiting well-known principles and laws of Chemistry. Specifically, we model interactions among nodes in a communication network (e.g., computers, routers, single queues in a packet-switched network such as the Internet, but also sensors in a wireless network) as chemical reactions, and we metaphorically associate information (e.g., data packets, radio frequency pulses) to chemical molecules. Practically, we implement in each node of the network a so-called *artificial chemistry* [71], a man-made system that "behaves similarly" to a chemical system and has some constituent features in common with actual chemical reaction networks. The implemented artificial chemistry constitutes an underlying dynamical system that governs the behavior of the network's node – the artificial chemistry shapes the node's response to external events (e.g., packet arrival, radio frequency pulse reception, packet loss detection), orchestrates internal computations (e.g., computation of the next packet dequeueing time, reduction of the transmission rate), and triggers its actions (e.g., packet drop, transmission of radio frequency pulses, packet dequeueing). Chemical reactions occur both within the network node (this is how the state of the node evolves in response to external events and internal updates), and among spatially distributed entities, in the form of information exchanges. From this perspective, Artificial Chemistry can be used to model and perform *distributed computations* and *processing* tasks.

Use of chemical laws and principles

With the Chemistry-inspired approach, the design of algorithms and protocols for communication networks becomes "drawing" an opportune reaction network. Once the reaction network that performs the desired task is designed, it can be compiled to a program that "executes" chemical reactions on-line. At this micro-level, each event in the communication network contributes to a state change of the chemical reaction network (the dynamical system governing the behavior of the network's node). The state of the node evolves according to the designed reaction network and to well-defined timings, such that the expected system trajectory emerges at the macro-level.

We focus on the dynamical aspect and thus refer specifically to the branch of Chemistry that studies dynamics of chemical reaction systems, i.e. *Physical Chemistry*. We import into communication network engineering laws and principles that define chemical kinetics. This allows enforcing a strict relationship between the performed functionality and the exhibited dynamics of Chemistry-inspired mechanisms.

Focus on network dynamics

Benefits to networking and communication systems...

..analysis

..design

Both *analysis* and *design* of algorithms/protocols for communication networks can benefit from the Chemistry-inspired approach – researchers can take advantage of the chemical theory to study traditional problems (e.g., consensus problem in wireless sensor networks [180]) or solve current open issues (e.g., emergent control to achieve a collective goal [135]). Two major improvements characterize the analysis process: (*i*) In the Chemistry-inspired engineering framework, the fluid model of the system can be generated automatically from the execution model (automaton) of the algorithm/protocol. (*ii*) The underlying chemical metaphor enables the direct use of new (in the field of communications and networking) analytical tools (e.g., steady-state analysis, sensitivity analysis, Chemical-Organization theorems) to study equilibrium points, the dynamical behavior, the sensitivity, and the stability of the developed system. Two major improvements also characterize the design process: (*i*) Designers can easily import chemical patterns, which are known to embody vital regulatory mechanisms in living complex systems, to perform traditional tasks in a stable, robust, and self-regulatory way (e.g., adopting the chemical Enzymatic model to control the traffic in the Internet). (*ii*) Thanks to the applied chemical laws, Chemistry-inspired systems tend to exhibit smooth transitions and leverage stable attractors. That is, an intrinsic feature of Chemistry-inspired systems is the propensity to achieve equilibrium states and be robust against perturbations.

1.3 Contributions of the thesis

We identify the dynamical aspect as the main benefit of adopting the chemical metaphor in communication and networking engineering, and focus on predicting and controlling dynamics of communication networks by applying rules and principles of Physical Chemistry. This approach is shown to improve the analysis and the design of communication and networking protocols and to originate deployable solutions.
The specific contributions of this thesis are as follows:

- We broaden part of the ideas published in [163]. We do not use the chemical metaphor only to *model* some communication aspects of *computer networks*. Rather, we design and *implement* mechanisms for *communication* and *networking* systems, which can be actually deployed in today's infrastructures and do not rely on special (chemical) packets nor require the hypothetical replacement of existing infrastructures' layers and elements.

- We extend Artificial Chemistry by relaxing the requirements deriving from the will to reproduce faithfully phenomena observed in real Chemistry. This enhances both functionality and analyzability of the designed mechanism, in the communication and networking context.

- We show that applying laws and principles of (artificial) Physical Chemistry to the design of communication algorithms is not a mere intellectual exercise – we implement Chemistry-inspired algorithms that improve state-of-the-art solutions in terms of stability and predictability.

- We derive a class of rate controllers that can be easily parameterized, extended, and customized, for various purposes and in different operational environments.

- We analyze and implement a Chemistry-inspired solution for distributed computing in wireless sensor networks.

- We give evidence that Chemistry-inspired mechanisms can actually be deployed nowadays – we provide promising results from experiments where artificial chemistries are implemented at kernel and user space of computers, on embedded microprocessors, and on hardware.

- We develop a high-level abstraction for designing hardware control modules, on Field-Programmable Gate Arrays (FPGA) technology.

1.4 Thesis outline

The thesis is structured as follows: We first introduce concepts, formal definitions, laws, and analytical tools related to Artificial Chemistry, and start explaining the chemical metaphor for distributed computations and information exchanges (Chapter 2). We then take advantage of the introduced theory to analyze and design traffic shaping solutions for packet-switched networks, and specifically derive a class of distributed traffic rate controllers (Chapter 3). We adopt the Chemistry-inspired engineering approach to define a set of interaction rules that allow achieving distributively consensus in wireless sensor networks, and derive from them a simple communication protocol that lets nodes exchange their data in an asynchronous admission-free manner, with no need for symbolically encoded information and for packet exchanges (Chapter 4). We show how to guarantee high-speed performances of Chemistry-inspired algorithms and build them in hardware, and at the same time, we introduce a user-friendly high-level abstraction for the design of hardware controllers (Chapter 5). We conclude by summarizing the main features of Chemistry-inspired systems and our contributions in the context of communications and networking, and discussing possible future research directions (Chapter 6).
All chapters are both theoretical and practical, with aspects concerning the analysis and the background theory as well as details related to implementations and experimental evaluations. After all, our ultimate belief is that *modeling*, *analysis*, *design*, and *implementation* of communication and networking solutions should be one and the same thing.

This thesis has a "self-similar" structure – it has approximatively the same structure of its chapters (i.e., introduction, main body, conclusion), and chapters' sections are in line with this writing scheme (although not explicitly structured in this way). Each chapter treats different topics and different contexts. For this reason, all chapters start with a generic overview on the covered topic, which includes a thorough discussion of related works and on our specific contributions. All chapters end with a discussion of developed insights/open issues and a short summary that recalls the milestones of the treated argument. The Introduction and the Conclusion (chapters 1 and 6) are exceptions to these general rules.

Chapter 2

Artificial Chemistry for Networking and Communications

"Choosing which aspects of the system should be modeled and which aspects can be ignored is an art rather than science."

Srinivasan Keshav

This chapter gives the reader all tools to understand the work presented in this thesis. We provide an introduction to Artificial Chemistry and start clarifying how one can look at communications and distributed computations from a chemical perspective. We identify the dynamical aspect as the main benefit of importing the chemical theory and using it in communication engineering. For this reason, we focus on analytical tools and theories concerning chemical kinetics, likely unfamiliar to most computer scientists and communication engineers. For the sake of clarity, we support the introduced concepts with toy examples, then generalized and elaborated in the next chapters to support interesting applications in computer networks and communication systems.
This chapter is structured as follows: We first report historical notes in the field of Artificial Chemistry and specify the space and scope of our contribution in Section 2.1. We introduce the basics of Artificial Chemistry in Section 2.2 and give formal definitions. We (*i*) introduce a fundamental law (law of mass action) that describes the emergent macro-behavior of chemical systems, (*ii*) clarify how to study chemical kinetics on the micro-level, and (*iii*) illustrate an efficient way to reproduce chemical dynamics on traditional computers. Then, we present in Section 2.3 all important analytical tools that we will use in the next chapters to study the equilibria of our Chemistry-inspired algorithms, their dynamical behavior, and if and how they converge. In Section 2.4, we show how to parametrize and extend Artificial Chemistry to realize Chemistry-inspired systems for communication and networking, abstracting away from requirements of actual Physics. Finally in Section 2.5, we point at some issues that possibly arise during the engineering phase, and we hint at general design aspects to overcome these issues.

2.1 Introduction

Artificial Chemistry ($\mathcal{AC}$) represents a tool for modeling different types of natural systems and investigating their (possibly complex) dynamics. In general, the $\mathcal{AC}$-approach is based on a chemical metaphor, with molecules that collide and react according to certain rules (reactions). As we see later in detail, $\mathcal{AC}$ is used to study the origin, the evolution, and the maintenance of organizations, as well as a method to find solutions to optimization problems. Another interesting application of $\mathcal{AC}$ is to process information and perform computations by taking advantage of the computational properties of chemical systems. This can be done by using real molecules and chemical reaction systems, or rather can be done on traditional computers and machines by using $\mathcal{AC}$ as a design paradigm for new software and hardware architectures.
In the following, we give some historical notes on $\mathcal{AC}$s and review a few major works in this context. We focus mainly on the works that propose $\mathcal{AC}$s as information processing systems, and that deal with Chemistry as a mere design paradigm for communication and networking solutions.

2.1.1 Related works

Research in the context of $\mathcal{AC}$ has been, and still is, characterized by a broad application spectrum. Mainly, it has been applied to modeling, information processing, and optimization. For a compact but complete review on the history and works in the context of $\mathcal{AC}$s, refer to [71].
$\mathcal{AC}$s have been used as a *modeling* system in different domains, such as biological, evolutionary, social, and parallel-processing systems. Researchers have mainly aimed at understanding the "logic" of certain natural phenomena, such as examining the abstract problem of the emergence and organization in *life*. For example, in the fifties, Turing introduced an abstract chemical system[2.1] to show how spatial patterns can be generated by simple chemical processes (e.g., he explained how "destabilization through diffusion" mechanism motivates "symmetry breaking and pattern formation" phenomena) [233]. Another famous example is the Algorithmic Chemistry "AlChemy"– refer to [90] and [91] for further details – that showed how self-organizing sets (organizations[2.2]) naturally arise in a system of interacting elements.
Artificial chemical systems can also be seen as a generalization of many evolutionary algorithms. The reason lies in the ability of $\mathcal{AC}$s to create evolutionary behavior (or self-evolution [69]). For example in [134], Koza introduced self-replicating computer programs that were self-improving and evolving. The in-

[2.1] Touring's model was made of a few chemical species interacting according to a couple of reaction rules, and had the structure of a conventional chemical reaction model.

[2.2] An organization is a set of elements such that each element can be generated by the conjunct action of the others, and such that given any two elements in the system, their interaction will always generate only elements of the set.

troduced environment can be regarded as an $\mathcal{AC}$ where molecules are computer programs and reactions take place while programs are executed. By exploiting the self-evolutionary feature, researchers have proposed $\mathcal{AC}$s to find solutions to "difficult", mostly combinatorial problems. As a consequence, many $\mathcal{AC}$-based search and optimization algorithms have been introduced, such as the catalytic search to solve simple problems and find approximations for complex ones [258].

$\mathcal{AC}$s as...

...information processing systems

...distributed, adaptive computing systems

Most chemical processes in nature can be interpreted as processing information and performing computations. Every living entity can formally be seen as an information processing system where the basic mechanism of information processing is Chemistry – i.e., an information processing system generates an output (in general, a certain behavior) as a result of processing the input (the current internal state and the environmental inflow). Many works have exploited $\mathcal{AC}$s both for "*real* chemical computing", by the use of real molecules, and for "*artificial* chemical computing", where the chemical metaphor is applied as a design paradigm for new hardware and software architectures. For example, artificial chemical systems have been extensively used for control tasks in robotics – e.g., hormone systems have been used in [40] to achieve asynchronous information flow and coherent behavior in a distributed parallel control architecture for a humanoid robot torso; excitable lattices in [3] and simplified enzyme-substrate kinetics in [268] were used to control real mobile robots. Whereas in computer architectures, a relevant example of applying a chemical paradigm to perform distributed computations dates back to the eighties when Banâtre *et al.* in [27] envisioned a highly abstract coordination model that aimed at autonomous, distributed, and dynamically adapting programming workflows. Their abstract rewriting model referred to the following chemical metaphor: A chemical reaction model describes computation in terms of chemical reactions; data (γ-abstractions) is represented by molecules (γ-expressions) that remain in the multiset at the end of the program; a set of "reaction" rules is given to combine elements in the multiset and produce new elements. The execution of a γ-program can be seen as the evolution of a solution of molecules, which react until the solution becomes inert. The main feature of γ-programs is that they involve implicit parallelism – data is processed in parallel and thus computation should be easily performed on one as well as on multiple processors. However, the gap between a γ-program and any real computer on which it is assumed to run is large and thus, the design of reasonably efficient implementations of the model is a challenging task.

γ-calculus

Since the introduction of γ-calculus, later extended in [26], many other chemical formalisms have been proposed. The Cham [34] investigated more theoretical aspects and added new features to the chemical programming paradigm by considering solutions of multisets and allowing for the definition of membrane-like encapsulation of sub-solutions. Higher-order multiset rewriting [141] enabled any configuration, made of multisets and a set of rewriting rules, to handle other configurations through their program. The hmm-calculus [57] attempted to generalize the γ-model by introducing "one-shot and first-class

citizen" reactions. Finally, the P-system [193] interpreted as computations the processes happening in the compartmental structure of a living cell and based on this perspective provided an alternative and complementary approach to classical methods such as the ones that are based upon Ordinary Differential Equations (ODEs). All these chemical formalisms were built on the same basic paradigm but had different properties and different expressive powers as studied and summarized in [25].

A more recent proposal, which indirectly has inspired also the work presented in this thesis, is the work by Tschudin [230]; he proposed a Chemistry-inspired rewriting system that, by combining the γ-formalism and the tag-system introduced by Post in [197], is able to efficiently process headers of data packets. Originally, the Fraglet-programming language was defined as "a molecular Biology-inspired execution model for computer communications", as Fraglets (small elements that represent fragments of a distributed computation, either code or data) irreversibly interact with other Fraglets according to a few simple rules (string substitution patterns), similarly to Chemistry where molecules react with each other according to reaction rules.

The chemical metaphor has been strengthened by Meyer *et al.* in [166], where the Fraglet system has been extended by introducing the chemical *Law of Mass Action* (LoMA) as a scheduling policy for part of the string rewriting rules. This has brought chemical kinetics into protocol programming languages and, de-facto represents a big step toward a generic approach to characterize the dynamics of computer networks with chemical laws and principles. Authors of [166] made explicit how Fraglet tag-matching systems can be mapped to an $\mathcal{AC}$ and can be used to design network protocols. They made use of the LoMA-extended Fraglet system to develop a networking protocol for load balancing, where the desired result emerges from the combination of distributed reactions, thanks to the applied laws and kinetics principles. Further, they exploited chemical kinetics as a tool to create computer network functionalities, which rely on *rate-encoded* information rather than the symbolic information, traditionally used in networking protocols.

Chemical Networking Protocols...

...kinetics into protocol programming languages

...rate-encoded information

2.1.2 Space and scope of our contribution

Our work is in line with those approaches that exploit $\mathcal{AC}$s in the context of information processing, and with those solutions that import laws, principles, and theories form Chemistry in order to achieve distributed adaptive computing systems.

For example, similarly to what Banâtre *et al.* proposed in [27] or to the P-system [193] and other chemical formalisms such as [34, 57, 141], we adopt the chemical metaphor to solve parallel concurrent computations. And similarly to [28, 64, 230], we take advantage of the chemical metaphor in order to design algorithms for communication- and networking-related applications. We focus on the *dynamics* and thus refer to specific principles and laws of Physical Chemistry. Such as in [157, 163], we employ the chemical metaphor to

build information processing systems where computation emerges out of "an orchestrated interplay of many decentralized relatively simple components". Specifically, this chapter extends and broadens the scope of some ideas that Meyer *et al.* proposed [163,165,166]: the use of the chemical metaphor to model self-healing protocols for computer networks, whose dynamics are inspired by chemical reaction networks – Chemical Networking Protocols (CNPs). Following the CNP approach, we let algorithms "be driven" by underlying reaction models. The direct coupling between quantity and rate, which is enforced by the reaction law (i.e., law of mass action), permits to automatically build a deterministic ODE model that describes the algorithm's dynamics. By allowing deviations from what happens in real chemistry and by sticking to the only constraint of implementing a mass-action scheduler, we enhance controllability and analyzability of networking and communication systems. We make use of the chemical metaphor to enable working with *rate-encoded* information rather than, as traditionally, with explicit numeric values. We recognize the dynamical aspect as the main benefit of importing Chemistry into information processing and communication engineering, and thus focus on the latter aspect only. We do not treat active (code-oriented) networking, and self-organizational aspects of "chemical software code".

Focus on dynamics

In general, in the context of protocol design, the idea of mimicking phenomena and mechanisms observed in nature is not new. The literature reports many trials (successful or not) that exploit biological models (e.g., swarm-based systems, firstly introduced in [212]), physical fields (e.g., EM, temperature, and other generic artificial force fields such as potentials in [31]) and bio-chemical models (e.g., reaction diffusion for epidemic routing in [131]). However, the majority of research takes care of the spatial distribution/diffusion of information (e.g., routing) and often overlooks the dynamics. Additionally, solutions usually rely on importing specific Nature-inspired models to fulfill specific tasks. This profoundly differs from our new paradigm for the design and analysis of algorithms which (*i*) focuses on achieving stable and predictable dynamics, without directly caring about the spatial-aspect of the distribution of contents (information), (*ii*) adopts a whole theory (chemical theory), and (*iii*) is not confined to a specific application but rather has a wide utilization spectrum.

2.2 Basics of Artificial Chemistry

$\mathcal{AC}$ can be seen as a subfield of Artificial Life research that, by abstracting from natural molecular processes, tries to investigate the behavior of complex systems. In these systems, global properties emerge naturally from the local interactions between their subsystems. In this section, we start to make clear how to take advantage of this peculiarity in networking and communication systems.

After introducing colloquially the chemical metaphor in Section 2.2.1, we give its formal definition in Section 2.2.2. In Section 2.2.3, we introduce the law of

mass action which describes macroscopically how reactions occur in $\mathcal{AC}$s and which represents the rule we, indirectly or directly, apply to schedule events in networking and communication systems. In Section 2.2.4, we look at the micro kinetics of actual chemical systems in order to grasp the reason behind implementation choices of existing $\mathcal{AC}$s. Finally, we show in Section 2.2.5 how chemical kinetics can be reproduced on traditional CPUs.

2.2.1 The chemical metaphor for networking

Microscopic level

Macroscopic level

Generally, we can look at networking protocols and the way how messages are exchanged from both a microscopic and a macroscopic level. At the microscopic level, a protocol implementation cares about the individual details: processing of data, the actions triggered when information (e.g., a packet) is received, the scheduling of consequent actions, and so forth. Protocols are traditionally implemented as state-machines where an asynchronous event (e.g., a packet reception) triggers an action and a state transition. Instead at the macroscopic level, individual events are not important, neither observable. Rather, the macroscopic view deals with flows (e.g. packet streams), and focuses on the large-scale dynamics of protocols.

A direct micro-macro mapping

Surprisingly, we can refer to Chemistry and the related chemical laws and principles to obtain an alternative to design, execute, and analyze algorithms and protocols for networking. Thanks to the Chemistry-inspired framework, the designer does not have to care anymore about the microscopic details of the implementation in order to satisfy requirements at the macroscopic level. According to the chemical metaphor, we can compare information/messages to molecules and let them "react" with each other akin to molecular reactions. With this novel framework, the designer can program networking and communication algorithms at the macro-level by "drawing" reaction networks. At the same time, the chemical approach provides an executable model for these reaction networks at the micro level, without the need for "programming" the emergent algorithm. Once the reaction network that controls the flow is designed, it can be compiled to a program that "executes" chemical reactions on-line and one-by-one. At this micro level, there is a direct mapping between events in the communication network and those in the chemical reaction network, such that the expected flow trajectory emerges at the macro level.

$\mathcal{AC}$s to...

...solve concurrent computations

...perform simultaneous communications

This approach can be used to solve concurrent computations and to perform simultaneous communications in a natural way [68]. Practically, we let a Chemistry-inspired dynamical system govern the behavior of each single entity of the communication system: We associate events in a communication network (e.g., packet arrivals, packet generations, reception of simple RF-pulse and so forth) to the production or destruction of virtual molecules in a chemical dynamical system, which represents a sort of control plane of the protocol's dynamics (i.e., events in the chemical dynamical system, such as the execution of a specific reaction or the generation of a specific molecule, correspond to specific actions, such as packet transmission, packet dropping, and so forth).

Molecules react with other molecules and the obtained web of reactions can be used to perform computation. When several network nodes (e.g. hosts in a computer network, or sensors in a wireless sensor network) interact with each other and behave according to the chemical dynamical system, the overall system can be seen as a unique system (reaction network) that performs a *distributed* computation.

2.2.2 Formal definition

Formally, we can characterize each node (entity) of the communication system according to an Artificial Chemistry. An Artificial Chemistry is univocally defined as a triple

Artificial Chemistry

$$\mathcal{AC} \doteq (\mathcal{S}, \mathcal{R}, \mathcal{A}) \tag{2.1}$$

where $\mathcal{S} = [s_1, s_2, \ldots, s_{|\mathcal{S}|}]$ is the set of molecular species that may appear in a certain Chemistry, $\mathcal{R} = [r_1, r_2, \ldots, r_{|\mathcal{R}|}]$ is the set of reactions that expresses which reactant molecules can collide and which product molecules are generated during this process, and $\mathcal{A}$ is the algorithm that defines which and when reactions occur [71]. In this section, we are going to comment on the topology of a reaction network (i.e., we explain the meaning of $\mathcal{S}$ and $\mathcal{R}$ terms). We leave to the next sections the discussion of its dynamics (i.e., we explain the third element $\mathcal{A}$ not sooner than Section 2.2.5).
A reaction rule $r \in \mathcal{R}$ operates according to a given equation whose general form is as follows:

Reaction equation

$$r: \quad \sum_{s \in \mathcal{S}} \alpha_{r,s} s \xrightarrow{k_r} \sum_{s \in \mathcal{S}} \beta_{r,s} s \tag{2.2}$$

where k_r is a constant parameter (known as reaction coefficient) that contributes to regulating the average rate at which reaction r occurs, $\alpha_{r,s}$ is the number of molecules of species s consumed by reaction r (known as stoichiometric reactant coefficient), and $\beta_{r,s}$ is the number of molecules of species s produced by the r-reaction (known as the stoichiometric product coefficient). Basically, the above equation states that the r-reaction replaces an amount $\alpha_{r,s}$ of reactant molecules s (left-hand side) to produce an amount $\beta_{r,s}$ of product molecules s (right-hand side), with an average rate controlled by coefficient k_r. The number of reactant species $\mathcal{O}(r)$ is called the order of the reaction.
For example, a simple unimolecular (order one) reaction (illustrated in Figure 2.1(a))

$$r: \qquad 1 \cdot \mathrm{S} \xrightarrow{1} \emptyset$$

describes the process that consumes one molecule of species S each time reaction r occurs. We neglect for the moment the role of the reaction coefficient, set to one in the example. Of course, the reaction can happen only when at least one S-molecule is present in the reactant vessel.[2.3] When this condition is

[2.3] We call "reaction vessel" an instance of an $\mathcal{AC}$ as defined in (2.1). A vessel thus contains a multiset of molecules; each molecule in it is an instance of one of the molecular species $s \in \mathcal{S}$.

(a) Chemical model (b) Corresponding queuing network

Figure 2.1: *A unimolecular reaction "extracts" a molecule of chemical species S like a server extracts a packet from a traditional queue.*

satisfied the reaction is said to be "active", otherwise it is called "inert". To call back concepts of communication system engineering (refer to Figure 2.1(b)), we can see the molecular species S as a traditional queue that stores packets. In this context, the execution of the simple reaction $r : \mathrm{S} \rightarrow \emptyset$ corresponds to the action of dequeueing a packet.

As a next step, we introduce the concept of Distributed Artificial Chemistry ($\mathcal{DAC}$) by extending the formal definition of an $\mathcal{AC}$ to a communication network whose interaction topology is represented by a directed graph ("digraph")

Interaction graph of communication networks

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}). \tag{2.3}$$

The set $\mathcal{V} = \{\nu_1, \nu_2, \ldots, \nu_{|\mathcal{V}|}\}$ contains all graph vertexes (i.e., computers in a computer network, sensors in a wireless sensor network, and so forth) whereas the set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ contains the edges, with the convention that $(\nu_i, \nu_j) \in \mathcal{E}$ if and only if there exists a connection from node ν_i to node ν_j (i.e., the information flows from ν_i to ν_j). The structure of a digraph can be described by the $|\mathcal{V}| \times |\mathcal{V}|$ adjacency matrix $\mathbf{\Delta}$ whose generic entry $[\mathbf{\Delta}]_{i,j}$ is equal to 1 if $(\nu_i, \nu_j) \in \mathcal{E}$ and 0 otherwise.

In a $\mathcal{DAC}$, molecules can be exchanged over the network links by executing reaction rules that generate remote actions. In particular, at each node $\nu_i \in \mathcal{V}$, a reaction algorithm $\mathcal{A}$ (the same for all nodes) updates a local multiset of molecules according to a set of local reaction rules. That is, each node ν_i defines a local $\mathcal{DAC}$ as the triplet

Distributed Artificial Chemistry

$$\mathcal{DAC} \doteq (\mathcal{M}_i, \mathcal{R}_i, \mathcal{A}) \tag{2.4}$$

in which $\mathcal{M}_i = \mathcal{S}_i \cup \mathcal{S}_i^{(j)}$. The set $\mathcal{S}_i$ defines the species of all molecules that can possibly be found in the local multiset, whereas $\mathcal{S}_i^{(j)} \subseteq \cup_{j \in \mathcal{N}_i} \mathcal{S}_j$ is the set of species that can possibly be found in its neighbors. Each node also defines its own set of reaction rules $\mathcal{R}_i$ where a reaction $r_i \in \mathcal{R}_i$ is specified as follows:

Interactions among spatially distributed nodes

$$r_i : \quad \sum_{s \in \mathcal{S}_i} a_{r_i,s} s \overset{k_{r_i}}{\rightarrow} \sum_{s \in \mathcal{M}_i} b_{r_i,s} s.$$

Note that all reactants are local species only whereas products may also be species located in the neighboring nodes. This is how transmission or exchange

(a) Chemical model (b) Corresponding queuing network

Figure 2.2: *A molecule of chemical species S_1, located at node ν_1, is transformed into a molecule of chemical species S_2 located at node ν_2. In terms of computer networks, a packet is extracted from queue S_1, sent over the traditional communication infrastructure, and enqueued into queue S_2 at the recipient.*

of information is modeled in a chemical way: by allowing a reaction to create product molecules in nearby nodes.[2.4]

As an example of the chemical metaphor in the context of distributed communication systems, we can consider a simple unimolecular reaction (illustrated in Figure 2.2(a))

$$r: \qquad 1 \cdot \mathrm{S}_1 \xrightarrow{1} 1 \cdot \mathrm{S}_2$$

that affects species located in two different, spatially separated vessels: We can see such a reaction as the processes of dequeuing a packet from the queue located in node ν_1, sending the packet to node ν_2, which correctly receives the packet and enqueues it in its own queue. In this example, node ν_1 is characterized by $\mathcal{DAC}_1$ defined according to (2.4) where $\mathcal{M}_1 = \mathcal{S}_1 \cup \mathcal{S}_1^{(2)}$, $\mathcal{S}_1 = \{\mathrm{S}_1\}$, $\mathcal{S}_1^{(2)} = \{\mathrm{S}_2\}$ and $\mathcal{R}_1 = \{r\}$. (Refer to Figure 2.2 for a graphical representation of this example.)

In the previous example the structures of communication and chemical networks coincide: two entities of the communication systems (i.e., two queues in two network nodes) correspond to two molecular species (i.e., S_1 and S_2); a link that guarantees that packets travel from node ν_1 to node ν_2 corresponds to the reaction r. However this does not hold true in general, and most of the time. For example, consider the previous chemical reaction with an additional species C that controls the extraction of molecules from species S_1 in node ν_1 according to the following, slightly different reaction (illustrated in Figure 2.3(a)):[2.5]

$$r': \qquad 1 \cdot \mathrm{S}_1 + 1 \cdot \mathrm{C} \xrightarrow{1} 1 \cdot \mathrm{S}_2.$$

[2.4] Communications among spatially distributed nodes takes place by means of traditional network infrastructure (e.g. Internet Protocol (IP), Ethernet medium, wireless medium).

[2.5] The control mechanism stems from the inertness condition concerning reaction r': at least one molecule C must be present to authorize the consumption of a molecule S_1. The system can be seen as a queue C that contains virtual tokens and controls the dequeuing of payload packets stored in S_1. This control mechanism will be thoroughly investigated in the next chapter.

(a) Chemical model (b) Corresponding queuing network

Figure 2.3: *One molecule of chemical species S_1 and one of species C, located at node ν_1, are transformed into a molecule of chemical species S_2 located at node ν_2. Similarly, a special (controlled) server dequeues a packet from queue S_1 and sends it to the recipient ν_2. Structures of chemical and communication networks differ.*

In this case, the communication network (refer to Figure 2.3(b)) has the same structure whereas the chemical reaction network has an additional vertex (i.e., species C).

The structure of the chemical reaction network is described by the stoichiometric matrix $\boldsymbol{\Xi}$:

Stoichiometric matrix...

$$\boldsymbol{\Xi} = [\Xi_{s,r}]_{|\mathcal{S}|,|\mathcal{R}|} = [\beta_{s,r} - \alpha_{s,r}] \tag{2.5}$$

...the chemical reaction network's structure

whose elements $\Xi_{s,r}$ represent the number of molecules of chemical species s that are transformed by a particular chemical reaction r. In our first example, the chemical network composed by species S_1 and S_2 and reaction r is characterized by a stoichiometric matrix $\boldsymbol{\Xi} = [1\ -1]^T$, having two rows as the number of species present in the network, and one column as the number of reactions characterizing the network. On the contrary, the slightly different chemical network composed by species S_1 S_2 and C and reaction r' is characterized by a stoichiometric matrix $\boldsymbol{\Xi} = [1\ -1\ 1]^T$ which has an additional row that describes the effect of species C. The communication network graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in both examples is constituted by vertex vector $\mathcal{V} = [\nu_1, \nu_2]$ and edge vector $\mathcal{E} = [l_{\nu_1 - \nu_2}]$.

The compact description $\boldsymbol{\Xi}$ of the structure of the chemical network assists chemists and biologists in important tasks such as (*i*) the discovery of pathways that carry out a distinct biological function of the chemical network, (*ii*) the discovery of dead ends and futile cycles (i.e., dependent subsets of reactants) (*iii*) the identification of optimal and suboptimal operating conditions for an organism, and (*iv*) the analysis of network flexibility and robustness. In this thesis, we will also make use of the stoichiometric matrix $\boldsymbol{\Xi}$ and refer to its elements in the analysis and the formal description of proposed Chemistry-inspired solutions for communication and networking systems.

2.2.3 Law of mass action (LoMA)

After having introduced the first two elements that define an $\mathcal{AC}$, we focus now on the dynamics of the chemical system.
The dynamics of chemical reactions (when which reaction occurs) are described on average by the well known Law of Mass Action (LoMA), which essentially states that the average rate of occurrence of a chemical reaction is proportional to its reactant concentrations [241]. That is, the more reactant molecules are present, the more frequent reactions occur. Mathematically, this means that a chemical reaction $r \in \mathcal{R}$ occurs at an average rate $v_r(t)$ proportional to the abundance of involved reactants:

LoMA...

...the more reactant molecules are present, the more frequently reactions occur

$$v_r(t) = k_r \prod_{s \in \mathcal{S}} c_s^{\alpha_{r,s}}(t) \tag{2.6}$$

where k_r is the reaction coefficient mentioned before and $c_s(t)$ denotes the molecular concentration of reactant species s at time t. This quantity is related to the amount (number) of molecules of s-species and to the volume of the reaction vessel times the Avogadros constant. In our artificial chemical setting, and thus in all subsequent discussions, we refer to a simplified notion of concentration, such that the concentration becomes equivalent to the quantity of molecules, except from the fact that the concentration represents a continuous quantity whereas the number of molecules represents a discrete one.

Concentration $\sim$ #molecules

The empirical mass-action principle enables us to describe the time evolution of the average species abundance as a set of $|\mathcal{S}|$ Ordinary Differential Equations (ODEs):[2.6]

Generalized mass-action model

(fluid model)

$$\frac{\partial c_s(t)}{\partial t} = \overbrace{\sum_{r \in \mathcal{R}} \beta_{r,s} v_r(t)}^{\text{inflow}} - \overbrace{\sum_{r \in \mathcal{R}} \alpha_{r,s} v_r(t)}^{\text{outflow}} \quad \forall s \in \mathcal{S} \tag{2.7}$$

or, alternatively, in a compact vector notation

$$\dot{\mathbf{c}} = \boldsymbol{\Xi} \cdot \mathbf{v}(\mathbf{c}(t)). \tag{2.8}$$

Such a mathematical representation of a chemical system represents a deterministic approximation (fluid model) of actual chemical kinetics, which leads to very simple, low-effort solutions and, as we show later in this thesis, is accurate enough for our purposes.

To illustrate the foregoing ideas, consider again the reaction $r : \mathrm{S} \xrightarrow{k} \emptyset$ which simply consumes S-molecules, illustrated in Figure 2.1(a). According to the LoMA in (2.6), the average rate of this reaction is

$$v_r(t) = k c_{\mathrm{S}}(t). \tag{2.9}$$

[2.6] The equation set in (2.7) is also referred to as differential (reaction) rate equations.

We can thus describe this *uni*molecular reaction system with the following (*one*) ODE: $\partial c_{\mathrm{S}}(t)/\partial t = -kc_{\mathrm{S}}$. If we consider species S to be initialized to $c_{\mathrm{S}}(t_0) = c_0$, the ODE has the solution

$$c_{\mathrm{S}}(t) = c_0 e^{-kt} \tag{2.10}$$

that describes the average number of S-molecules decaying exponentially with a speed that is proportional to coefficient k.
If we again regard species S as a packet queue and reaction r as the process of dequeuing its head packet (see the graphical representation in Figure 2.1(b)), the previous result implies that the chemical metaphor suggests to serve the queue in a non-work-conserving manner (by adding an additional service delay, and thus enabling idle phases in the server).[2.7] As we will see soon in this and the next chapter, the non-work-conserving policy improves the predictability of the dynamics of the process itself and thus of the system, and its stability.

The reaction algorithm $\mathcal{A}$ to respect LoMA

The LoMA is respected only if we manage to set properly the time instants when each reaction occurs. As mentioned before, such a scheduling scheme is handled by the reaction algorithm $\mathcal{A}$, which defines how the set $\mathcal{R}$ of rules r is applied to a collection $\mathcal{S}$ of molecules s, and thus it defines the dynamical behavior of a reaction vessel.

2.2.4 Stochastic chemical kinetics

Before introducing the third element characterizing the definition of an $\mathcal{AC}$, the reaction algorithm $\mathcal{A}$, we go through the main findings of Physical Chemistry in the context of chemical kinetics. This allows us to motivate the implementation of the reaction algorithm $\mathcal{A}$ and understand the related insights.
It is convenient to define the "state" of a chemical reacting system as "the current number of molecules of each component species" (we mentioned before that, for us, the molecular concentration represents a continuous quantity that approximates the number of molecules). In reality, the movement of molecules in a well-stirred vessel in thermal equilibrium follows a Brownian motion. This means that when the state of a chemical reacting system is described as a set of concentrations that evolve (in time), we lose information about the positions and momenta of molecules, and hence we cannot predict deterministically the future behavior of the system [99].

Micro-level random trajectories

For example, if we consider again the simple decaying reaction depicted in Figure 2.1(a), we observe random trajectories eventually leading to the same steady state (where no molecules in species S are present). Figure 2.4 shows three possible realizations of the stochastic process that characterizes the time-evolution of the number of molecules S. As we expect, because of the mass-action principle, all three trajectories follow on average the exponential decay described in (2.10).

[2.7] Traditionally, queues are served in a work-conserving manner – serve the queue as fast as possible, avoiding an idle state of the server.

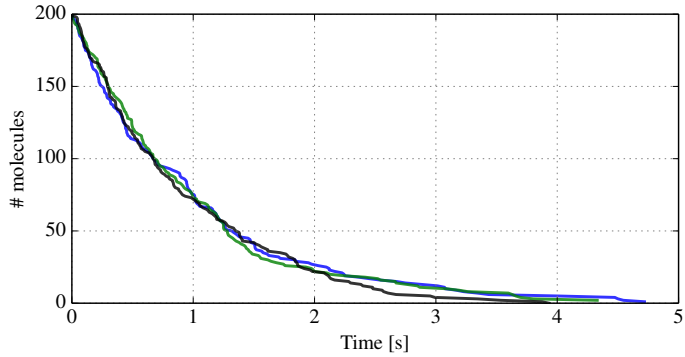


Figure 2.4: *Three possible realizations of the stochastic process that describes the state evolution of a unimolecular $r : S \rightarrow \emptyset$ system, which has initially 200 molecules. Refer to Figure 2.1(a) for a graphical illustration of the concerned chemical model.*

Mc Querry in [160] firstly attempted to mathematically accommodate the stochastic nature of chemical reacting systems and proposed what today is known as the *chemical master equation*.

2.2.4.1 Chemical Master Equation (CME)

The Chemical Master Equation (CME) describes, in terms of probabilities, the time-evolution of a chemical system's state, i.e., the possible compositions of the reaction vessel. Its *a priori* validity and physical fidelity was proved by Gillespie in [99]. In the following, we elaborate on the derivation of the CME provided in [99], by using insights published in [247]. For convenience, we refer to the state of a chemical reacting system as the number of molecules of each species that evolves in time. We ignore the positions and velocities of the individual molecules (we assume the system being well-stirred[2.8]) and rigorously, we accept some randomness in the process that describes the behavior of the system.
From this point of view, molecular populations in a chemically reacting system are integer variables that evolve stochastically, according to a Markov process. The changes in the species populations are a consequence of the chemical reactions. From this perspective, each reaction r is characterized mathematically by two quantities. The first is its state-change vector $\mathbf{x}_r = [x_{1,r}, \ldots, x_{|\mathcal{S}|,r}]$, where $x_{s,r}$ is the change in the population of species s caused by a reaction r, such that if the system is in state $\mathbf{n}$ and a reaction r occurs, the system immediately jumps to state $\mathbf{n} + \mathbf{x}_r$. The other characterizing quantity for r is its *propensity* function $a_r(t)$, which reflects the probability, given a certain

[2.8] In a well-stirred vessel, the probability that a molecule will participate in a reaction does not depend on its position in the vessel.

state $\mathbf{N}(t) = \mathbf{n}$, that one r reaction will occur in the next infinitesimal time interval $[t, t + dt)$. This definition can be seen as "the fundamental premise of stochastic chemical kinetics" because "everything else in the theory follows from it via the laws of probability" [96].

Propensity...

The propensity can be seen as the product of the so-called "stochastic rate constant" c_r and a term h_r that reflects the number of distinct combinations of reactant molecules of reaction r [247]:

...probability that a certain reaction occurs in the next infinitesimal time interval

$$a_r(\mathbf{N}(t)) = c_r \cdot h_r(\mathbf{N}(t)). \tag{2.11}$$

The term h_r changes over time according to the binomial coefficient

$$h_r(\mathbf{N}(t)) = \prod_{s \in \mathcal{S}} \begin{pmatrix} N_s(t) \\ \alpha_{s,r} \end{pmatrix}.$$

For example, in the unimolecular reaction $r : \mathrm{S} \to \emptyset$, we would have $h_r = N_\mathrm{S}$ possible combinations. In a higher-order reaction $r : \mathrm{A} + 2\mathrm{B} + 3\mathrm{C} \to \ldots$, we would have

$$h_r = \begin{pmatrix} N_\mathrm{A} \\ 1 \end{pmatrix} \begin{pmatrix} N_\mathrm{B} \\ 2 \end{pmatrix} \begin{pmatrix} N_\mathrm{C} \\ 3 \end{pmatrix} = N_\mathrm{A} \cdot \frac{N_\mathrm{B}(N_\mathrm{B} - 1)}{2} \cdot \frac{N_\mathrm{C}(N_\mathrm{C} - 1)(N_\mathrm{C} - 2)}{2 \cdot 3}$$

possible combinations of reactants (we omit the time-dependency). The stochastic rate constant c_r depends on physical properties of the reactant molecules and the term $c_r dt$ represents the probability that a particular selected combination of reactant molecules of a reaction r at time t will react in the next infinitesimal time interval dt. Thus, the definition in (2.11) agrees with the "fundamental premise". Indeed, if we multiply the probability $c_r dt$, which applies to a particular selected combination of reactant molecules of a reaction r, by the total number of distinct combinations of reactant molecules at time t, we obtain the probability that a reaction r will occur in the next infinitesimal time interval $[t, t + dt)$, because reactions occur in a maximum parallel and independent manner.

A physical derivation for c_r is in general not possible and scientists often rely on other arguments. We avoid to delve into details regarding the derivation of this quantity; we just note that Wolkenhauer *et al.* have demonstrated in [247] that c_r exhibits a strong relationship with the reaction rate coefficient k_r – the constant that appears in the empirical LoMA and the related rate equation (ODE) models. Wolkenhauer *et al.* offer us an alternative definition of the propensity (rigorously correct only if a high number of molecules is present in the vessel):

Propensity ∼ LoMA

$$\langle a_r \rangle = k'_r \prod_{s \in \mathcal{S}} \langle N_s \rangle^{\alpha_{s,r}} \tag{2.12}$$

where the symbolism $\langle * \rangle$ identifies the expectation operator and k'_r is a term that depends on the Avogadro number and the considered volume, on the reaction rate constant k_r, and on the reaction molecularity $K_r = \sum_{s \in \mathcal{S}} \alpha_{s,r}$.

For example, by adopting such a simplification to derive the propensity of the unimolecular reaction $r : \mathrm{S} \to \emptyset$ we get

$$\langle a_r \rangle = k'_r \langle N_\mathrm{S} \rangle. \tag{2.13}$$

We can easily observe the similarity between the above expression and the mean reaction rate expressed in (2.9).
The stochastic nature of the given propensity definition implies that we cannot exactly predict $\mathbf{N}(t)$. However we can infer the probability to be in a certain state $\mathbf{N}(t) = \mathbf{n}$ at time t given the fact that initially at time t_0 the system was in the state $\mathbf{N}(t_0) = \mathbf{n}_0$

$$P(\mathbf{n}, t|\mathbf{n}_0, t_0) \doteq \text{ Prob}\Big\{\mathbf{N}(t) = \mathbf{n}, \text{ given } \mathbf{N}(t_0) = \mathbf{n}_0\Big\}$$

and derive a time-evolution equation for this probability by applying the laws of probability to the "fundamental premise". The result of this is the CME [96]:

Chemical Master Equation (CME)...

$$\frac{\partial P(\mathbf{n}, t|\mathbf{n}_0, t_0)}{\partial t} = \sum_{r=1}^{|\mathcal{R}|} \Big(\overbrace{a_r(\mathbf{n} - \mathbf{x}_r) P(\mathbf{n} - \mathbf{x}_r, t|\mathbf{n}_0, t_0)}^{\text{entering state } \mathbf{n}} - \overbrace{a_r(n) P(\mathbf{n}, t|\mathbf{n}_0, t_0)}^{\text{leaving state } \mathbf{n}} \Big). \tag{2.14}$$

...probability that each species has a molecular population at a future time

The first term on the right-hand side describes the change to state $\mathbf{n}$, while the second term describes the changes away from state $\mathbf{n}$.[2.9]
Again considering the simple unimolecular reaction $\mathrm{S} \to \emptyset$, when species S is initialized to n_0, the CME gives

$$\frac{\partial P(n, t|n_0, 0)}{\partial t} = a_r(n+1) P(n+1, t|n_0, 0) - a_r(n) P(n, t|n_0, 0).$$

As $P(n_0 + 1, t|n_0, 0) \equiv 0$, this equation can be solved exactly by successively putting $n = n_0, n_0 - 1, \ldots, 0$ and results in the following probability density function:

$$P(n, t|n_0, 0) = \frac{n_0!}{n!(n_0 - n)!} e^{-k' n t} (1 - e^{-k' t})^{n_0 - n} \quad (n = 0, \ldots, n_0). \tag{2.15}$$

The above probability density function describes a binomial random variable, which has $\eta_{N_\mathrm{S}}(t) = n_0 e^{-k' t}$ as a first moment and $\sigma_{N_\mathrm{S}}(t) = \sqrt{n_0 e^{-k' t}(1 - e^{-k' t})}$ as a second moment. The expression of the expected number of S-molecules is very similar to the generalized mass-action model in (2.10).
The CME represents a formal analytical tool to fully describe the dynamical behavior of chemical systems, taking into account both stochasticity and discreteness aspects. However, the CME is actually a set of coupled ODEs, with one equation for every state, i.e. every possible combination of reactant

[2.9] The multiplication of propensity and probability makes sense if we consider the time-derivative on the left; propensity times dt gives a probability.

molecules. This implies that the CME can be solved analytically for a few simple cases only, and numerical solutions are prohibitively difficult in other cases. That is, for first-order reaction systems, each species adds one dimension to the problem leading to an exponential growth of the computational complexity. For instance, if we would analyze a system of 3 species, having each between 1 and 100 molecules, the CME will contain 100^3 coupled ODEs. Actually, one ODE per state. For higher-order reaction systems, the solution of CME is not even possible to achieve [122].

2.2.5 Reaction algorithm $\mathcal{A}$

From the same set of theorems used to formally validate the CME, Gillespie further derived a stochastic simulation algorithm [98]. The algorithm, being logically equivalent to the CME, simulates *correctly* molecular collisions and provides a sample trajectory of the random process characterizing the state evolution of the chemical system. Such an algorithm constitutes the third element $\mathcal{A}$ of the $\mathcal{AC}$-definition.

The idea behind the exact stochastic simulation is to try generating a single sample of the state evolution of a chemical process in the stochastic framework.

Reaction algorithm $\mathcal{A}$...

...to produce a realization of the process described by the CME

Indeed, a big disadvantage of the (intractable) CME approach is that it tries to write a system of equations and solve it simultaneously for the probability of all possible trajectories. Exact stochastic simulation methods generate a sample of the system's state evolution by picking reactions and times according to the "correct" probability distributions. The term "correct" refers to a procedure which is able to generate a certain trajectory with exactly the same probability as that obtained with the CME approach. As a result, one can write an algorithm that generates "correct trajectories" even when it is not possible to write the related CME explicitly [94]. For example, the Direct Method proposed by Gillespie in [98] calculates explicitly *which* reaction r occurs next and *when* it occurs; the reaction r_i is chosen according to $P(r=i) = a_i / \sum_{r\in\mathcal{R}} a_r$, and the time τ is chosen according to $P(\tau)d\tau = \sum_{r\in\mathcal{R}} a_r e^{-\tau \sum_{r\in\mathcal{R}} a_r} d\tau$.

Besides being "correct", the algorithm $\mathcal{A}$ must be efficient. The literature reports many Monte Carlo procedures for numerically generating time trajectories of the molecular populations in exact accordance with the CME (refer to [96] for an overview). In the following, we review the Next Reaction Method as proposed by Gibson and Bruck in [94]. This stochastic simulation approach, which is rigorously equivalent to the CME approach, is shown to be efficient, its computational complexity is proportional to the logarithm of the number of reactions, and it requires a single random number per simulation event. This approach reduces the computational effort of the Direct Method and of its improved version, First Reaction Method, proposed by Gillespie in [97].

The Next Reaction Method $\mathcal{A}$

The Next Reaction Method computes the next reaction time t_r of each reaction rule r relying on the propensity a_r and stores it in an *"indexed priority queue"*. This queue consists of a tree structure of ordered pairs of the form (i, t_i), where i is the identifier of a certain reaction and t_i is the putative time when

that reaction should occur. Element pairs are stored in the structure according to the value of the corresponding reaction time: the first stored element has the next reaction time.
The derivation of the next reaction time follows from the definitions we gave in Section 2.2.4: The propensity has been described as a function $a_r(\mathbf{n})$ such that the product $a_r(\mathbf{n})dt$ represents the probability that a reaction $r \in \mathcal{R}$ will occur in the next infinitesimal time interval $[t, t+dt)$, given a certain state $\mathbf{N}(t) = \mathbf{n}$. A consequence of this is the form of the next-reaction density function, which is defined as the probability $p(\tau, r|\mathbf{n}, t)d\tau$ that, given a certain state $\mathbf{N}(t) = \mathbf{n}$, the next reaction will occur in the infinitesimal time interval $[t+\tau, t+\tau+dt)$, and will be reaction r. After a series of reasonings (see [95,97]), the quantity $p(\tau, r|\mathbf{n}, t)$ is found to be

$$p(\tau, r|\mathbf{n}, t) = a_r(\mathbf{n})\exp\left(\sum_{i\in\mathcal{R}} a_i(\mathbf{n})\tau\right) \quad (0 \leq \tau < \infty;\ r \in \mathcal{R}).$$

This formula provides the basis for stochastic simulation algorithms to generate the (random) putative next reaction time t_r (also referred to as inter-reaction time):

$$t_r = \exp\left(\frac{1}{a_r(\mathbf{n})}\right) \quad (r \in \mathcal{R})$$

where the propensity a_r, calculated according to (2.11), directly stems from physical properties of actual reaction systems. That is, the next time of each reaction is a random variable extracted from the exponential distribution with mean equal to the reciprocal of the propensity.
Details on how to implement the Next Reaction Method are reported in Algorithm 1. Here, we limit to observe that the Next Reaction Method, besides being an efficient, statistically exact procedure for numerically generating time-trajectories of the molecular populations, represents "a macroscopic mass-action scheduler" [163]. The main aspect is that, like in mass-action models, average reaction rates turn out to be proportional to reactants' quantities.
Note that, by relying on Algorithm 1, one can implement/realize the chemical metaphor shown in Figure 2.1. One can use Algorithm 1 to schedule the service process of a traditional queue by "compiling" the algorithm with the reaction $r : \mathrm{S} \to \emptyset$ and associating the execution of reaction r to the dequeuing process. This would enable predicting the time-evolution of the queue in terms of probabilities by applying the results of the CME in (2.15) – the probability to have a certain queue fill level n_S at a generic time t given the fact that initially the queue was filled with n_S^0 packets. Similar predictions could be done also with traditional analytical tools, such as those derived from queuing theory, when applied to queues that are served according to classic (work-conserving) policies. However, in traditional practice of queue-theoretic modeling, a certain model of the service process is assumed so as to factor out the complexity of the process itself (e.g., the memory-less assumption and thus the assumption of exponentially distributed times and of Poisson processes allow using Markov's

Algorithm 1 Next Reaction Method
Exact and efficient stochastic simulation algorithm as published in [94].

1) Initialize:
   a) set the initial amount of molecules;
   b) calculate the propensity a_r according to (2.11) $\forall r \in \mathcal{R}$;
   c) set a putative reaction-execution time $t_r = \exp\{1/a_r\}$ $\forall r \in \mathcal{R}$;
   d) store values t_r in an indexed priority queue (the first stored element has the next reaction time) and store the related propensity values a_r.
2) Let r_i be the reaction whose putative reaction time, t_i, is least.
3) Let t be t_i.
4) Change the number of molecules to reflect the execution of reaction r_i.
5) Update all reactions, r_j, that depend on the executed reaction r_i:
   a) temporarily store the old value $a_j^{old} = a_j$;
   b) calculate the new propensity a_j, according to (2.11);
   c) if $r_j \neq r_i$, scale the reaction execution time by setting it equal to $t_j = (a_j^{old}/a_j)(t_j - t) + t$;
   else if $r_j = r_i$, set the reaction execution time $t_j = \exp\{1/a_j\} + t$;
   d) store the calculated reaction execution time t_j in the indexed priority queue and store the related propensity value a_j.
6) Go to Step 2.

Note:

(i) The variable t reflects the simulation time.

theory). By contrast, applying Algorithm 1 as a regulator of the queue's service process makes the modeling of the micro-level patterns of such a process exact. (This aspect will be investigated in detail in the next chapter.)

2.3 Analytical tools

We have already started glimpsing some benefits derived from the import of chemical concepts into communication systems. One main advantage is definitively the predictability of the system's dynamics. Additionally, we can directly import many tools from Chemistry and Biology into the analysis of communication and networking systems. In doing this, we avoid complex analyses and study the system from a high-level perspective.
Generally, we want to get insights on the system's dynamics in order to optimize the setting of its key parameters, to know its behavior in normal conditions, to understand its trajectories consequently to unpredicted situations, and to avoid uncontrolled responses to internal and external perturbations. Challenges in this endeavor are commonly represented by both the size and the complexity of the concerned networked system (e.g., consider the complex and large-scale Internet), and the complexity of the detailed analysis at the micro-level. For example, we cannot think of studying the dynamics of a communication network without abstracting away from specific node interactions and single events, such as actions on single packets. However, it is often a complex task to find abstract models that approximate closely enough the real behavior, and definitively, the modeling cannot be generalized and made automatic.
In this section, we demonstrate how the chemical approach assists us in this context. We explain in details all tools that the chemical metaphor may offer to those scientists and engineers that make use of the chemical approach to design and analyze their systems. This section should thus give the reader the expertise to follow the analysis related of chemical rate controllers (Chapter 3), and chemical consensus algorithms (Chapter 4).

2.3.1 Deterministic approximation

As we have shown in the previous section, an artificial chemical system can be implemented and run in an exact and efficient way by means of a stochastic algorithm $\mathcal{A}$, such as the Next Reaction Method (Algorithm 1). The behavior of such a system represents a stochastic process that can be modeled as a continuous-space Markov jump process, discrete in time. By means of the CME, we can describe at the microscopic level the stochastic evolution in time of molecules in the system (system's state-variables). However, such an approach suffers from the known curse of dimensionality: Each species adds one dimension to the problem, leading to a combinatorial growth of the computational complexity.
Due to the mathematical complexity of finding solutions to the CME, the

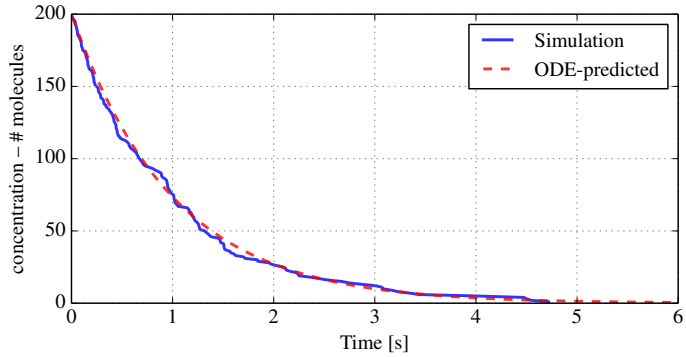


Figure 2.5: *Differential reaction-rate equation approximation (ODE-model)* vs. *simulated system trajectory. Blue-continuous line: actual realization of the stochastic process describing the state evolution of a unimolecular $r : S \rightarrow \emptyset$ system, which had initially 200 molecules. Red-dashed line: average trajectory of S-species concentration over time. Refer to Figure 2.1(a) for a graphical illustration of the concerned chemical model.*

common deterministic model of differential reaction-rate equations (i.e., the set of ODEs in (2.7)) is often exploited. From this description, when the parameters of the system are known and for specific initial conditions, the average dynamics of the system can be evaluated without taking stochasticity into account. This model approximates the exact dynamical behavior of actual chemical systems, and extremely reduces the computational work derived from the analysis: the dimension of the problem grows polynomially, instead of exponentially as for the CME method.

Differential reaction-rate equations...

As far as Chemistry-inspired algorithms are concerned, the set of differential reaction-rate equations in (2.7) represents the *fluid-model* of the system, a macro-level description of its large-scale dynamical behavior. This fluid-model directly derives from the system itself, i.e., from the underlying chemical reaction network. This means that the fluid model can be generated automatically once the related chemical reactions have been defined.

...an automatically-extracted fluid-model

Figure 2.5 compares an actual trajectory of the unimolecular $r : \mathrm{S} \rightarrow \emptyset$ system represented in Figure 2.1(a) and the average, approximated trajectory, calculated by applying the mass-action principle (LoMA). Respectively, the blue-continuous line is a realization of the stochastic process, and the red-dashed line is the solution (2.10) to the differential reaction-rate equation that describes the time-evolution of the average concentration of species S, when initially set to 200 molecules.

...the first step in the analysis

Throughout this thesis, the first step of the analysis will always be the derivation/extraction of the deterministic, macro-level description in terms of differential reaction-rate equations in (2.7). As we show next, this fluid-model description is most of the time accurate enough (we discuss in Section 2.4

and Section 2.5 how to improve this fidelity) and allows studying steady-state points, stability conditions, and system's sensitivity to internal as well as external perturbations.

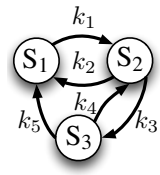
2.3.2 Steady-state analysis

A steady state captures the condition in which all state variables (i.e., the amount of all molecules) are constant in spite of ongoing reactions that strive to change them. Steady states are states in which the system spends most of the time, and to which it returns after being perturbed. We can easily estimate the steady state(s) of chemical systems by means of the so called steady-state approximation (also referred to as stationary-state approximation): First, we have to describe the system's average trajectory by exploiting the mass-action principle in (2.6) and thus, by writing the set of $|\mathcal{S}|$ ODEs in (2.7) that describe the time evolution of the average species abundance. Then, we can simply impose the steady-state condition, thus set the derivatives to zero (i.e., left-hand side of (2.7) set to zero), and study the solution c_s^* such that

$$0 = \sum_{r\in\mathcal{R}} \beta_{r,s} \overbrace{\left(k_r \prod_{s\in\mathcal{S}} c_s^{*\alpha_{r,s}} \right)}^{v_r} - \sum_{r\in\mathcal{R}} \alpha_{r,s} \overbrace{\left(k_r \prod_{s\in\mathcal{S}} c_s^{*\alpha_{r,s}} \right)}^{v_r} \quad \forall s \in \mathcal{S}. \tag{2.16}$$

When solving such a problem, we must keep in mind all other constraints that may characterize the system such as the constraint derived from the mass-conservation principle valid for closed system.
For example, we can study the steady state of a non-trivial chemical system composed by three species $[\mathrm{S}_1, \mathrm{S}_2, \mathrm{S}_3]$ that react according to the following reactions:

$$\begin{aligned} r_1 &: \ \mathrm{S}_1 \xrightarrow{k_1} \mathrm{S}_2 & \qquad r_4 &: \ \mathrm{S}_3 \xrightarrow{k_4} \mathrm{S}_2 \\ r_2 &: \ \mathrm{S}_2 \xrightarrow{k_2} \mathrm{S}_1 & \qquad r_5 &: \ \mathrm{S}_3 \xrightarrow{k_5} \mathrm{S}_1 \\ r_3 &: \ \mathrm{S}_2 \xrightarrow{k_3} \mathrm{S}_3. \end{aligned} \tag{2.17}$$



The system is initially in the state $\mathbf{c}(t=0) = [c_1^0,\ c_2^0,\ c_3^0]$. We solve reaction rate equations when the steady-state condition is imposed, namely

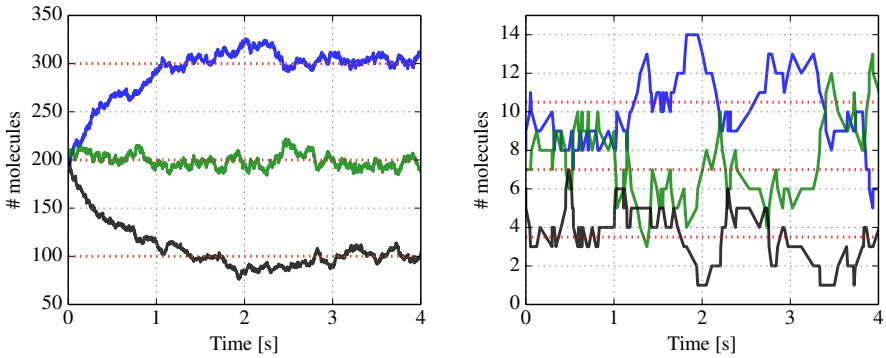
$$\begin{aligned} 0 \equiv \ \dot{c}_1 &= k_2 c_2 + k_3 c_3 - k_1 c_1 \\ 0 \equiv \ \dot{c}_2 &= k_1 c_1 + k_4 c_3 - (k_2 + k_3) c_2 \\ 0 \equiv \ \dot{c}_3 &= k_3 c_2 - (k_4 + k_5) c_3. \end{aligned}$$

Because the system is closed, i.e. there is no external inflow or outflow, we have the additional constraint that imposes a constant total number of molecules:

$$c_1(t) + c_2(t) + c_3(t) = const. = c_1^0 + c_2^0 + c_3^0 \quad \forall\, t.$$

Only by knowing the initial condition, we thus obtain the steady-state vector

$$\mathbf{c}^* = \begin{bmatrix} c_1^* \\ c_2^* \\ c_3^* \end{bmatrix} = \frac{c_1^0 + c_2^0 + c_3^0}{k_1 k_3 + (k_1 + k_2) k_5 + (k_1 + k_2) k_4 + k_3^2} \cdot \begin{bmatrix} k_2 k_5 + k_2 k_4 + k_3^2 \\ k_1 k_5 + k_1 k_4 \\ k_1 k_3 \end{bmatrix}.$$



(a) Initial condition $c_i^0 = 200$

(b) Initial condition $c_i^0 = 7$

Figure 2.6: *Steady-state approximation is not always accurate. Both charts show the number of molecules over time of the three species S_1 (blue line), S_2 (green line), and S_3 (black line) that reacted according to reaction rules (2.17) with $k_i = 1 \forall i$. Figure 2.6(a) shows a state-variables realization when the process was initialized to $c_1^0 = c_2^0 = c_3^0 = 200$, and Figure 2.6(b) when it was initialized to $c_1^0 = c_2^0 = c_3^0 = 7$. Steady-state values are plotted in red dotted line.*

For example, by setting all reaction coefficients to one and assuming an initial condition where $c_1^0 = c_2^0 = c_3^0 = 200$, we would have $c_1^* = 300$, $c_2^* = 200$, $c_3^* = 100$ at the steady state. Figure 2.6(a) shows the number of molecules over time of the three species S_1 (blue line), S_2 (green line), and S_3 (black line) when initialized as mentioned, and plots the steady-state values (red dotted lines).
By working with reaction rate equations and steady-state approximations, we implicitly refer to concentrations and hence, we deal with an approximation of the actual integer state variables. Again referring to the previous example (reaction system described in (2.17)) and considering $c_1^0 = c_2^0 = c_3^0 = 7$ as initial condition, we would predict a steady state of $c_1^* = 10.5$, $c_2^* = 7$, $c_3^* = 3.5$. Of course, when we implement a Chemistry-inspired system and we actually run it by means of a reaction algorithm $\mathcal{A}$ such as the Algorithm 1, we can have only an integer number of molecules for each species. In such a case, we would have the species in the chemical system continuously updated by ± 1 molecule, matching on average the approximated value $\mathbf{c}^*$.

Possible inaccuracy of the steady-state approximation

The deviations observed between the system's actual state and its steady-state approximation are significant when we deal with low concentrations (low number of molecules). For example, Figure 2.6(b) shows this phenomenon occurring in the system described in (2.17) when the species are initialized to seven molecules. The reason is twofold: (*i*) For a high number of molecules the approximation of discrete quantities by continuous ones is appreciable. (*ii*) The more reactant molecules there are, the lower the randomness affecting the

system's trajectory is. This is a consequence of having exponentially distributed reaction intervals (time between two consecutive reaction executions), with mean-parameter proportional to the amount of involved reactant molecules.[2.10] This phenomenon and the related consequences in the design and analysis of Chemistry-inspired communication systems are thoroughly investigated in Section 2.4.

Differently from the chemical equilibrium where state variables do not change due to a zero net reaction rate (i.e., reactants transform into products at the same rate as products transform into reactants), the steady-state condition can be reached also when no reactions occur at all. For example if we consider the system composed by species S_1 and S_2 and the single unimolecular reaction $r : S_1 \rightarrow S_2$, we have the simple steady-state solution $c_1^* = 0,\ c_2^* = c_1^0 + c_2^0$, given the initial condition $\mathbf{c}(t = 0) = [c_1^0,\ c_2^0]$. This state does not represent a chemical equilibrium as, in this case, the reaction system becomes inert.

2.3.3 Transient analysis

In simple systems, state variables decrease or increase gradually until they reach their steady-state value. In more complex systems, state variables might fluctuate around the theoretical steady state, either forever (a limit cycle) or gradually coming closer and closer. By looking at steady states only, we can just imagine how the system behaves under "normal" conditions. However, we (*i*) completely miss insights about the *transient* phase of the system, (*ii*) ignore how the system responds to *perturbations*, and (*iii*) cannot estimate *how long* it takes for the system to reach the predicted steady states.

One approach to predict the dynamical behavior of the system would be writing down the differential reaction-rate equations in (2.7) and directly solving them to find out how the concentrations evolve in time on average. However, this direct approach, although simple in the case of trivial systems, may be impracticable for complex ones. Furthermore, the sole study in the time domain may turn out not sufficient to capture all details of the system's behavior.

Methods from different domains assist us in analyzing the system's transient behavior in order to reveal the *semantics* and the *sensitivity* of its *key parameters*: We first describe the fluid model in terms of sensitivity to perturbations, we then reformulate it in control-theoretic terms, and finally, we characterize it in the frequency domain.

2.3.3.1 Sensitivity analysis

A powerful tool to analyze chemical systems is the Metabolic Control Analysis (MCA) [119, 203]. In this (and next) section(s), we introduce the MCA with

[2.10] According to the Next Reaction Method by Gibson and Bruck [94] (see Algorithm 1), the exact and efficient reproduction of chemical kinetics are obtained modeling the time as exponentially distributed.

formal definitions and intuitive examples that again refer to the metaphor with a simple queueing system, further extended and analyzed in the next chapter. Imported from Systems Biology, the MCA is based upon the macro-level fluid-model description in (2.7) and focuses on changes of the reaction network's trajectory due to perturbations of the system's parameters. The system's sensitivity can be formally quantified through the following function

Sensitivity function...

$$\varepsilon(t) \doteq \partial \mathbf{c}(t)/\partial \mathbf{p}(t),$$

...derivative of states $\mathbf{c}$ with respect to perturbations $\mathbf{p}$

where $\mathbf{c}$ and $\mathbf{p}$ are state and parameter vectors. As usual, we refer to the state of a chemical reaction network as the molecular concentrations. The elements of $\mathbf{p}(t)$ can be formed by any variable whose influence we want to investigate. For the sake of clarity, we consider a simple LoMA-served queue as the one depicted in Figure 2.7, whose dequeueing process is implemented by means of a stochastic simulation algorithm such as Algorithm 1. The related chemical model represented in Figure 2.7(a) consists of the usual unimolecular reaction $r : \mathrm{S} \xrightarrow{k} \emptyset$, whose rate is controlled by the coefficient k (we consider $k \sim k'$) and species S, which is fed at rate λ (we may use the symbolism $\emptyset \xrightarrow{\lambda} \mathrm{S}$ to express the enqueueing process in chemical terms). In MCA terms, the fill level of the queue constitutes the single element of vector $\mathbf{c}(t)$. Differently, we can define the reaction coefficient k, the queue's fill level c, the enqueuing rate λ, and the dequeueing rate v_{out} as elements of the parameter vector $\mathbf{p}(t)$, in order to investigate the influence of these parameters on the system.
Following the chain rule for differentiation, one can show that the sensitivity function ε satisfies the following differential equation [201]:

$$\dot{\varepsilon}(t) = \boldsymbol{\Xi} \cdot \frac{\partial \mathbf{v}}{\partial \mathbf{c}} \cdot \varepsilon(t) + \boldsymbol{\Xi} \cdot \frac{\partial \mathbf{v}}{\partial \mathbf{p}}, \tag{2.18}$$

which depends on the stoichiometric matrix $\boldsymbol{\Xi}$ (i.e. the description of the chemical network's topology) and on the reaction rate vector $\mathbf{v}$ (whose elements v_r are derived directly from the mass-action principle (2.6)). Terms $\partial \mathbf{v}/\partial \mathbf{c}$ and $\partial \mathbf{v}/\partial \mathbf{p}$ (so called elasticity coefficients) give a measure of how strongly each single reaction in isolation (e.g., a queue-interaction) is affected by infinitesimal perturbations of either system's states (e.g., queue fill levels) or parameters (e.g., enqueueing rates).

...a linearization of the fluid model, around steady states

We can also regard (2.18) as the linearization of the following set of ODEs

$$\dot{\mathbf{c}}(t) = \boldsymbol{\Xi} \cdot \mathbf{v}(\mathbf{c}(t), \mathbf{p}(t))$$

and thus have an alternative representation of the fluid model in (2.8). Since we are interested in perturbations around the nominal behavior, we first find steady state $\mathbf{c}^*$, $\mathbf{p}^*$ by solving (2.8) for $\dot{\mathbf{c}}(t) \equiv \mathbf{0}$. Then we define an input vector

Input vector

$$\mathbf{u} \doteq \mathbf{p} - \mathbf{p}^*$$

that denotes perturbations around nominal parameter values, and we define a

(a) Chemical reaction model (b) Corresponding queuing network

Figure 2.7: *A LoMA-served queue.*

vector

$$\mathbf{x} \doteq \mathbf{c}(\mathbf{p} - \mathbf{p}^*) - \mathbf{c}(\mathbf{p}^*)$$

that describes the resulting state fluctuations around the attractor. (For the sake of clarity, we omitted the time-dependency in the vector definitions.) Finally we define the system's sensitivity at steady state as the system's response around the fixed point $(\mathbf{c}^*, \mathbf{p}^*)$ for small perturbations of the input signal [119]:

$$\dot{\mathbf{x}}(t) = \boldsymbol{\Xi} \cdot \left.\frac{\partial \mathbf{v}}{\partial \mathbf{c}}\right|_{(\mathbf{c}^*,\mathbf{p}^*)} \cdot \mathbf{x}(t) + \boldsymbol{\Xi} \cdot \left.\frac{\partial \mathbf{v}}{\partial \mathbf{p}}\right|_{(\mathbf{c}^*,\mathbf{p}^*)} \cdot \mathbf{u}(t). \tag{2.19}$$

System sensitivity at steady state

Let us for example study the sensitivity of the simple queuing network in Figure 2.7, whose steady-state value is $c_S^* = \lambda/k$. By defining the enqueueing rate as sole parameter, $\mathbf{p} = [\lambda]$, we obtain the following description of queue-level variations:

$$\dot{\mathbf{x}}(t) = [-1\ 1] \cdot \begin{bmatrix} 0 \\ k \end{bmatrix} \mathbf{x}(t) + [-1\ 1] \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \mathbf{u}(t) = k\mathbf{x} - \mathbf{u}.$$

We observe that, as a consequence of the LoMA-service policy, the dequeueing rate is sensible to perturbations/variations of the rate at which packets are enqueued. The way how the system reacts depends on, and can be controlled by, the coefficient k.

We conclude this section by summarizing the main features of MCA: (*i*) MCA can be carried out in a linear regime, in which only small perturbations are addressed; (*ii*) the analysis is always tractable and often simple, and (*iii*) when the sensitivity is evaluated around the steady state, the system description is reduced to a set of linear ODEs (2.19). For *linear* reaction systems, i.e., consisting of uni-molecular reactions only, (2.19) is independent of steady state $\mathbf{c}^*, \mathbf{p}^*$ and thus, the description of the system's transient behavior is valid in general, also far away from the steady state.

2.3.3.2 Sensitivity analysis in control-theoretic terms

As a next step, we follow [201] and formulate the sensitivity analysis in control-theoretic terms, by expressing the linearized ODEs (2.19) as a Linear Time Invariant (LTI) system:

Linearized fluid model as a Linear Time Invariant (LTI) system

$$\dot{\mathbf{x}}(t) = \mathbf{A} \cdot \mathbf{x}(t) + \mathbf{B} \cdot \mathbf{u}(t) \tag{2.20a}$$
$$\mathbf{y}(t) = \mathbf{C} \cdot \mathbf{x}(t) + \mathbf{D} \cdot \mathbf{u}(t). \tag{2.20b}$$

Although such a state-space representation is common in control theory, in the following, we elaborate more on its components, in the context of Chemistry-inspired algorithms for communications.

The input vector $\mathbf{u}(t)$ combines the signals to be perturbed, such as (*i*) external inputs to which the system responds (e.g., changes of rates that feed species = enqueueing rate changes), (*ii*) system's parameters to be dimensioned (e.g., reaction coefficients to be calibrated = control parameters), and (*iii*) external disturbances (e.g., rate fluctuations of feedback reactions = fluctuations of rates in feedback loops).

The Jacobian state matrix $\mathbf{A}$ defines how a perturbation of concentrations (=fill levels) affects their future changes, whereas the input matrix $\mathbf{B}$ describes the influence of input signal variations on these concentrations. Both matrices are evaluated at the fixed point $(\mathbf{c}^*, \mathbf{p}^*)$:

Jacobian state $\mathbf{A}$ and input $\mathbf{B}$ matrices

$$\mathbf{A} = \boldsymbol{\Psi} \cdot \left.\frac{\partial \mathbf{v}}{\partial \mathbf{c}}\right|_{(\mathbf{c}^*,\mathbf{p}^*)} \qquad \mathbf{B} = \boldsymbol{\Psi} \cdot \left.\frac{\partial \mathbf{v}}{\partial \mathbf{p}}\right|_{(\mathbf{c}^*,\mathbf{p}^*)}. \tag{2.21}$$

Output $\mathbf{C}$ and feedforward $\mathbf{D}$ matrices to study...

The output vector $\mathbf{y}(t)$ represents the result of the sensitivity analysis. We thereby define the output and the feedforward matrices $\mathbf{C}$ and $\mathbf{D}$, respectively, depending on the result we are interested in: To analyze how perturbations of the input induce fluctuations of concentrations, we define

..concentration fluctuations

$$\mathbf{C} = \mathbf{I} \qquad \mathbf{D} = \mathbf{0}. \tag{2.22a}$$

To look at effects of input perturbations on reaction rates, we define

...rate fluctuations

$$\mathbf{C} = \left.\frac{\partial \mathbf{v}}{\partial \mathbf{c}}\right|_{(\mathbf{c}^*,\mathbf{p}^*)} \qquad \mathbf{D} = \left.\frac{\partial \mathbf{v}}{\partial \mathbf{p}}\right|_{(\mathbf{c}^*,\mathbf{p}^*)}. \tag{2.22b}$$

Again referring to the LoMA-served queue depicted in Figure 2.7, we can study the effect of run-time modifications of the reaction coefficients, i.e. by defining $\mathbf{p} = [k]$, or we can look at the consequences of changing the offered load, i.e. by defining $\mathbf{p} = [\lambda]$. We can either define the matrices $\mathbf{C}$ and $\mathbf{D}$ according to (2.22a) to study the effect on the fill levels of the queue S, or alternatively, we may use (2.22b) to study the effect on the dequeueing rate v_{out}. By just defining input and output vectors, we have a powerful and flexible tool at hand, with which we can understand the transient behavior of Chemistry-inspired systems.

For linear systems, the LTI description does not depend on steady-state values – i.e. (2.21) and (2.22) do not depend on $(\mathbf{c}^*, \mathbf{p}^*)$. Thus, the LTI description is valid also far away from the steady-state region.

2.3.3.3 Sensitivity Analysis in the Frequency Domain

Classical sensitivity analysis focuses on the system's asymptotic response to step-changes of parameters [203]. To make more general observations, as suggested in [119], we use the Laplace transform $F(s) \doteq \int_0^\infty e^{-st} f(t)dt$ and transpose the LTI system in (2.20) from the time- to the frequency-domain:

$$s \cdot \mathbf{x}(s) - \mathbf{x}_0 = \mathbf{A} \cdot \mathbf{x}(s) + \mathbf{B} \cdot \mathbf{u}(s) \tag{2.23a}$$

$$\mathbf{y}(s) = \mathbf{C} \cdot \mathbf{x}(s) + \mathbf{D} \cdot \mathbf{u}(s) \tag{2.23b}$$

where $\mathbf{x}(t = 0) = \mathbf{x}_0$. By assuming that the initial conditions refer to the steady state, i.e., that there are no initial perturbations ($\mathbf{x}_0 = \mathbf{0}$), the *Transfer Function* (TF) of the LTI system results in

Transfer Function (TF) matrix: perturbation effects on steady-state outputs

$$\mathbf{H}(s) \doteq \frac{\mathbf{y}(s)}{\mathbf{u}(s)} = \mathbf{C}\left(s \cdot \mathbf{I} - \mathbf{A}\right)^{-1} \mathbf{B} + \mathbf{D}. \tag{2.24}$$

Each element of the TF matrix $\mathbf{H}$ (so-called *response coefficient*) describes how a particular steady-state output $\mathbf{y}$ changes after perturbations of a particular parameter $\mathbf{u}$. Response coefficients are useful to investigate the behavior of the system to external influences. But if we are interested in designing the system's internal functionality, so-called *control coefficients* are often used. They can be derived from (2.24) by setting $\mathbf{B} = \mathbf{I}$.

The frequency-domain view allows understanding the system's response to input signals that are more complex than constant perturbations. Any time-varying signal such as steps, impulses, and periodic or near-periodic signals can be treated in this framework. Furthermore, the TF itself is a good instrument to get an idea of the system's behavior.

The nature of poles (roots of the characteristic equation, namely roots of the denominator of the TF, as well as eigenvalues of the state matrix) reveals system's trajectories under perturbations of initial conditions. That is, the location of these singularities in the s-plane provides qualitative insights about the system's response:[2.11] (*i*) A negative real pole $p_i = -\gamma$, which falls in the left-half of the s-plane on the x-axis, defines an exponentially decaying component $e^{-\gamma t}$ in the system's homogeneous response. The speed of the decay is thus determined by the pole location; poles far from the origin in the left-half plane correspond to components that decay rapidly, whereas poles near the origin correspond to slowly decaying components. (*ii*) A pole at the origin defines a component that is constant in amplitude and defined by the initial conditions. (*iii*) A real pole in the right-half plane corresponds to an exponentially increasing component. (*iv*) A complex conjugate pole pair $\gamma \pm j\omega$ in the left-half of the s-plane combines together to generate a response component that is a decaying sinusoid $Ae^{-\gamma t}sin(\omega t + \Phi)$ where A and Φ are determined by the initial conditions. Thus, the real term γ of poles

TF's poles define the system's response

[2.11] The complex plane on which Laplace transforms are graphed is usually named s-plane.

specifies again the speed of decay whereas the imaginary term ω defines the frequency of oscillation. (v) Finally, an imaginary pole pair $\pm j\omega$, lying on the imaginary axis, generates an oscillatory component with a constant amplitude determined by the initial conditions.

For example, we may be interested in the behavior of the simple queueing system in Figure 2.7 and thus study the sensitivity of the service rate v_{out} with respect to the enqueueing rate λ in the frequency domain. In this way, we obtain the two-element vector

$$\mathbf{H}(s) = [0\ k]^T (s+k)^{-1} (-1) + [1\ 0]^T = \left[1\ \frac{k}{s+k}\right]^T,$$

whose second element describes the relationship between dequeueing (*output*) and enqueueing (*input*) rates: $H_{\text{o-i}}(s) = v_{\text{out}}(s)/\lambda(s) = k/(s+k)$. We observe that the LoMA-served queue behaves as a first-order low-pass filter – it has a TF with one pole (one root in the TF denominator) and no zeros (no roots in the numerator). The cut-off frequency can be calibrated by the reaction coefficient k. That is, the dequeueing rate v_{out} approaches the average enqueueing rate λ, but bursts are filtered out. The lower the reaction coefficient k is, the more the buffer smooths the enqueueing rate, but the more slowly the buffer adapts its dequeueing rate to new values of loads. Figure 2.8 shows the effect of k on the dequeueing rate when the enqueuing rate oscillates at different frequencies around 60 pkt/s. The sole application of the mass-action scheduling, with the "proper" coefficient k, ensures that "fast" variations of the enqueueing rate do not affect the system, i.e., no/reduced burstiness of the dequeueing rate, at the cost of a "slower" adaptation. Note that besides controlling the burstiness of dequeueing rates, we also control the variation of the queue fill level.

Our ultimate motivation of making use of analytical tools is understanding the behavior of Chemistry-inspired systems and fine-tuning them. Usually, in this context, we keep the coefficients as free variables and study the system's response to external perturbations of quantities such as the offered load. By adjusting the coefficients we are able to shape the parametric transfer function such that the system responds quickly, while remaining stable. In a network where coefficients change by design, we may define them as input signals and analyze the system's sensitivity to them.

Studying the effect of non-constant perturbations

In the analysis we have proposed, we refer to the asymptotic behavior of the system (like for standard practices in the study of linear output-input systems) and thanks to the generalized sensitivity analysis in the frequency domain, we gain insights on time-varying perturbations of concerned parameters, not considering constant perturbations only [119]. Tractable time-varying signals are step variations, spikes (impulse inputs), any other periodic or near-periodic signal, and inputs with some regularity in their frequency-content. We recall that the *impulse response* of a system represents the time-domain counterpart of the TF. Besides fully describing the relationship between inputs and outputs of a system in the time domain, the impulse response can express small-scale

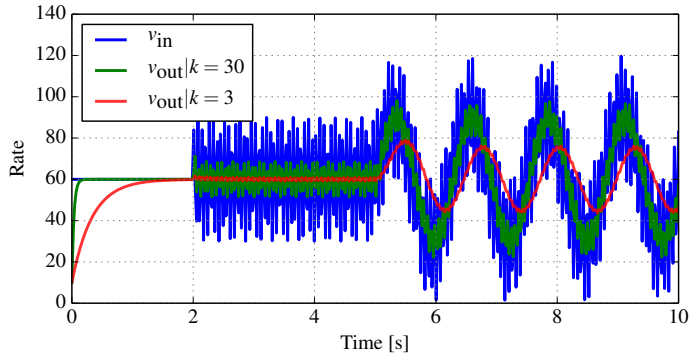


Figure 2.8: *Effect of coefficient k in a LoMA-served queue (refer to Figure 2.7): coefficient k controls how variations of the enqueueing rate affect the dequeueing rate. The Y-axis does not have a measurement unit as it refers generally to rates of the chemical model as well as to rates of the respective queueing system.*

effects of perturbations. By applying the impulse function to the enqueueing rate λ of our simple system in Figure 2.7 and by evaluating the algorithm's response, we quantify the sensitivity of the dequeueing rate v_{out} with respect to the injection of a single packet into the queue: $\mathcal{L}^{-1}\{k/(s+k)\} = ke^{-kt}$ for $t \geq 0$, where $\mathcal{L}^{-1}\{F(s)\}$ is the inverse Laplace transform function.[2.12] That is, the arrival of a packet causes a small dequeueing-rate spike with an amplitude equal to the k-value. Thereafter, the dequeueing rate drops back to its initial value in approximatively five times the value $1/k$. On the contrary, if we are interested in a sudden increase of the enqueueing rate, we have to study the *step response*. The response of the system in the frequency domain can be directly estimated by multiplying the TFs of the system and the input: $F(s) = 1/s \cdot k_1/(s+k_1)$. By anti-transforming the obtained result, we obtain the system's step response in time domain: $\mathcal{L}^{-1}\{1/s \cdot k_1/(s+k_1)\} = 1 - e^{-kt}$ for $t \geq 0$. The dequeueing rate exponentially approximates the enqueueing rate with an error $< 1\%$ in approximatively five times the value $1/k$ (see Figure 2.8, where for $k = 3\,\text{s}^{-1}$, v_{out} matches the value 0.99λ in less than 1.7 s).

2.3.4 Stability analysis

An important feature of a system is its *stability*. In particular, we may be interested in studying the ability of the system to react to a finite, time-

[2.12] The inverse Laplace transform function is defined as

$$\mathcal{L}^{-1}\{F(s)\} \doteq \frac{1}{j2\pi} \lim_{T\to\infty} \int_{\gamma-jT}^{\gamma+jT} e^{-st} F(s)\, dt$$

where γ is a real number so that the contour path of integration is in the region of convergence of $F(s)$.

limited perturbation, and verify that (i) it moves back to its initial state (i.e., asymptotically stable state), (ii) it moves to a new steady-state point which is different from the initial one (i.e., marginally stable state), or rather continues to evolve in time, without reaching a steady state (i.e., instability).

The stability analysis of linear systems is based upon its deterministic approximated description and practically reduces to an inspection of the state matrix of the system or, namely, to a characterization of the frequency-domain description (2.24). We may define a κ^{th}-order linear system "asymptotically stable" when all κ components of its homogeneous response decay to zero as time increases,[2.13] given a finite set of initial conditions.

We know from our previous argumentations that a pole lying in the right half of the s-plane lets the system's response increase without bound from any finite initial condition, and thus indicates the instability of the system. A system is defined to be marginally stable instead, when it has all poles with negative real part and/or poles with zero real value that are simple roots. Finally, an asymptotically stable system has negative real poles only. For example, the LoMA-served queue in Figure 2.7 is characterized by a single pole $p = -k$.

Stability of a LoMA-served queue

Because reaction coefficients are positive per definition, we conclude that the system is (asymptotically) stable. Indeed, if we apply a finite time-limited perturbation to the enqueueing rate, the dequeuing rate will transiently change but at last, once the perturbation ends, will return to the initial state $\lambda = v_{\text{out}}$.

A common approach to study the stability of non-linear reaction systems (i.e., constituted by multi-molecular reactions) is studying the linearized ODE model of the system around its steady state. Such a study is similar to that we described in previous sections – we study the eigenvalues of the Jacobian state matrix in (2.21) or equivalently, the poles of TF matrix in (2.24), evaluated at steady state. Differently from linear systems where these two matrices are independent of steady state $\mathbf{c}^*, \mathbf{p}^*$, for non-linear systems they depend on $\mathbf{c}^*, \mathbf{p}^*$ values. Thus, observations concerning the stability of the system hold true around steady state, and are rigorously correct for small perturbations only.

2.3.5 Deficiency-Zero Theorem

A topological analysis to infer the stability

Chemical reaction systems exhibit a tight relationship between topology and dynamics. This allows focusing on algebraic structures and relationships, which define the "topology" of the reaction network, to obtain general results concerning its stability.

Thanks to the deficiency zero theorem, we can infer qualitative properties of the dynamics from the network structure – the deficiency zero theorem states that a weakly reversible reaction network (following mass-action kinetics) with a zero deficiency value has exactly one fixed point, which is asymptotically stable. This differs from we have done so far when we discussed system's

[2.13] Each pole p_i defines a component $Ce^{-p_i t}$ of the homogeneous response $\sum_\kappa Ce^{-p_i t}$ of the system.

(a) Chemical reaction network (b) Equivalent graph of complexes

Figure 2.9: *A chemical network with 5 complexes and 2 linkage classes defining a weakly reversible system with zero deficiency. According to the Deficiency Zero Theorem the system has exactly one fixed point, which is asymptotically stable.*

dynamics relying on the "analytic" properties of chemical reaction networks, i.e., the properties that relate directly to the rates of change of concentrations (see (2.7) and the following analytical steps). The following treatment is based on findings from Feinberg [79] and Horn [112–115].

To apply the deficiency zero theorem, we first have to rewrite the reaction system in terms of *complexes* and to identify the *linkage classes* that constitute it. Complexes are those multisets of species that appear on the left- and the right-hand side of a reaction. We thus generally define a reaction as an interaction $r : C_r \to C_p$ of complexes C_i, which simply represent a sum of species with integer coefficients. For example, the reaction system described by the following set of reactions

Complexes

$$\begin{aligned} r_1 &: \quad \mathrm{A} \xrightarrow{k_1} \mathrm{B} \\ r_2 &: \quad \mathrm{B} \xrightarrow{k_2} \mathrm{C} + \mathrm{D} \\ r_3 &: \mathrm{C} + \mathrm{D} \xrightarrow{k_3} \mathrm{A} \\ r_4 &: \quad 2\mathrm{A} \xrightarrow{k_4} 2\mathrm{C} \\ r_5 &: \quad 2\mathrm{C} \xrightarrow{k_5} 2\mathrm{A} \end{aligned}$$

and represented in Figure 2.9(a) is constituted by the five complexes A, B, C + D, 2A, and 2C, as depicted in Figure 2.9(b). Linkage classes are the connected components of the directed graph that connects complexes. Again considering the example in Figure 2.9, we identify two linkage classes: subgraph of complexes A,B, and C+D, and subgraph of complexes 2A and 2C, see Figure 2.9(b).

Linkage classes

We further refer to a reaction network as *weakly reversible* if there exists a directed pathway from each member of a linkage class to all other members of the linkage class. In other words, if for every reaction leading from complex C_i to complex C_j, there is a sequence of directed reactions that connect C_j to C_i. Finally, we define *deficiency* $\wp$ of a chemical reaction network as the positive integer

Weakly reversible systems

$$\wp \doteq |\mathcal{C}| - \ell - \mathrm{rank}(\Xi) \tag{2.25}$$

where $\mathcal{C}$ denotes the set of complexes, ℓ is the number of linkage classes (i.e., the number of connected subgraphs in the graph of complexes), and rank($\mathbf{\Xi}$) denotes the rank of the stoichiometric matrix. For example, the rank of the stoichiometric matrix characterizing the network depicted in Figure 2.9 is

$$\text{rank}(\mathbf{\Xi}) = \text{rank} \begin{bmatrix} -1 & 0 & 1 & -2 & 2 \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 2 & -2 \\ 0 & 1 & -1 & 0 & 0 \end{bmatrix} = 3.$$

It follows that the deficiency value is $5 - 2 - 3 = 0$. We recognize the system in Figure 2.9 as weakly reversible (i.e., in the subgraphs of complexes, any vertex is reachable from any other one). Thus, if reactions are scheduled respecting the mass-action principle (i.e., by making use of a stochastic simulation algorithm such as Algorithm 1), we conclude that the system is complex-balanced and exhibits locally-stable dynamics, independently of the value of the reaction rates. Namely, we can infer conclusions on the stability of the equilibrium point of the system from the topological description of the system in terms of complexes and linkage classes.

Three main aspects of this theorem are important: (*i*) Assumptions of the Deficiency Zero Theorem are completely related to the structure of the system whereas the conclusions of the theorem are related to the dynamical properties of the system. (*ii*) The deficiency is dependent only on the complexes and the linkage classes of the system, and not on how the members of the linkage class react with each other. (*iii*) We can see the deficiency as a measure of the linear independence of "necessary" reactions. In a simplified system, if a reaction, which cannot be eliminated, is a linear combination of other reactions in the system, the deficiency will be greater than zero.

Feinberg and Horn introduced the notion of the deficiency of a chemical reaction network as a way to characterize mass-action systems that permit complex balanced equilibrium concentrations. Lately, their work has been expanded and generalized to characterize the equilibrium set of systems for which the underlying network is not weakly reversible and for which the network has a non-zero deficiency value (see [17, 59, 60, 220]). Although, the Deficiency Zero Theorem, as originally proposed, provides stability proofs for a certain class of reaction networks only, it often represents a simple alternative approach that turns out to be enough to prove the stability and the convergent behavior of Chemistry-inspired communication systems. For example in Chapter 4, we will resort to the Deficiency Zero Theorem to validate the stability of the Chemistry-inspired consensus algorithm for WSNs. In this thesis, we do not treat extensions and generalizations of the theorem – e.g., analysis of models that admit multiple equilibria and of persistence and global stability. However, we envision that recent findings in this context (e.g., [17, 59, 60, 220]) may be useful for studying Chemistry-inspired communication system characterized by complex reaction networks.

2.3.6 Randomness in chemical systems

In the past sections, we have avoided the complexity of analyzing the system at the micro-level and have instead used the *deterministic* macro-level description. Consistently with such an approach, in the next chapters, we will focus only on the average deterministic trajectory of chemical systems. Nevertheless, in this last section about analytical tools, we briefly report on possible means to account for the intrinsic stochasticity of the state of chemical reaction networks.

As we highlighted in Section 2.2.5, we may reproduce faithfully a chemical system by making use of a stochastic simulation algorithm such as Algorithm 1. We also pointed out in Section 2.2.4 that we are able study such a chemical system at the micro-level, and treat it as a continuous-time, discrete-space Markov jump process. Indeed, for simple enough systems, we can make use of the CME to describe the evolution in time of the probability to find the system in a certain state $\mathbf{n}$ at time t, given the fact that it was at state $\mathbf{n}_0$ at time t_0. We can thus describe in probabilistic terms (e.g. expected value and standard deviation) the stochastic process that the dynamics of an chemical system represent. On the contrary, the deterministic macro-level description of the system, by means of differential rate equations, describes the average trajectory of the chemical system. Practically, the analysis deals with the macroscopic concentration c_s of species $s \in \mathcal{S}$ which can be expressed as the expected value of the number of molecules n_s, i.e., $c_s(t) \sim \mathbb{E}[n_s(t)]$. And, in fact, it does not take into account the deviations between the average trajectory and an actual realization of the stochastic process.

Approximated, deterministic trajectory + Analytically tractable "noise"

There exist approaches to describe the dynamics of chemical reaction systems at the macro-level still accounting for their randomness: Chemical Langevin Equation (CLE) [95], Linear Noise Approximation (LNA) [237], and Two Moment Approximation (2MA) [235]. In general, the system's dynamics are described as a systemic, deterministic trajectory and a stochastic noise. The first describes the average behavior of the chemical system over all possible realizations of the stochastic process that represents the system's state-evolution. The latter catches the stochastic fluctuations observed in the algorithm's real-behavior around the average trajectory. All approaches approximate the stochastic fluctuations to a *Gaussian* noise, and all approaches take for granted that inter-reaction times are independent, exponentially distributed variables and their accuracy relates to the quantity of molecules.

In the CLE-approach [95], Gillespie derived a hybrid description of the system by means of a set of stochastic differential equations that separates the state change into a deterministic and a noisy part:

$$\dot{c}_s(t) = \overbrace{\sum_{r\in\mathcal{R}} \Xi_{s,r} a_r(\mathbf{c}(t))}^{\text{deterministic}} + \overbrace{\sum_{r\in\mathcal{R}} \Xi_{s,r} \sqrt{a_r\left(\mathbf{c}(t)\right)} \Gamma_r(t)}^{\text{noisy}} \tag{2.26}$$

Chemical Langevin Equation (CLE)

where $\Gamma_r(t)$ are temporally uncorrelated, statistically independent Gaussian

(normal) variables with zero mean and unitary variance, $\mathcal{N}(0,1)$. Hence, the CLE description in (2.26) includes multiplicative Gaussian white noises.
Such a description deals with continuous variables (concentrations $\mathbf{c}$) rather than with integer numbers (number of molecules $\mathbf{n}$). This description is derived from the CME (2.14) and is based on approximations and assumptions: (*i*) Gillespie assumed a high number of molecules such that deviations between the continuous approximation and the actual number of molecules are negligible. (*ii*) Within a small time interval $[t, t+dt)$, the state change has to be small such that the propensity function does not change much (upper limit on dt): $a_r(\mathbf{c}(t)) \sim a_r(\mathbf{c}(t+dt))$. This allows approximating the number of reactions that occur in subsequent "macroscopically infinitesimal time intervals" dt as "handy" random Poisson variables. (*iii*) Within the same macroscopically infinitesimal time intervals dt, each reaction must be executed more than once (lower limit on dt). This allows approximating those Poisson variables with more tractable normal random variables.
The CLE approach describes the stochastic trajectory of the system by means of stochastic differential equations (2.26). The main difficulty in this approach is solving such a set of equations.
A similar method is the LNA-approach proposed by van Kampen (refer to [237] for a complete overview on this topic), which, exploiting a perturbative-like approach, separates the description of the chemical reaction behavior into a deterministic macro-level trajectory, as given by the macroscopic reaction rate equation, and the intrinsic micro-level noise, modeled as an analytically tractable Fokker-Planck equation.
The LNA derivation starts by expressing conditional probabilities in the CME in terms of a new two-term variable:

Linear Noise Approximation (LNA)

$$n_s = \overbrace{\Omega c_s}^{\text{deterministic}} + \overbrace{\sqrt{\Omega}\xi_s}^{\text{fluctuations}}$$

representing the deterministic systematic change plus an additional term that accounts for fluctuations. Then, all terms of the CME are Taylor-expanded near the macroscopic concentration $\mathbf{c}(t)$ using the Ω-expansion method [237]. This method expands the virtual volume Ω of the reaction vessel in powers of $1/\Omega$. The lowest order returns the macroscopic rate equations as described by the mass-action principle and the next order is a linear Fokker-Planck equation characterizing the fluctuations. The Ω-expansion is mostly used in its lowest order form, where the fluctuations are approximated to follow a Gaussian distribution, and the errors in the average value, the second moment and the variance (i.e., the fluctuations) scale at most as Ω^0, Ω^1, and $\Omega^{1/2}$.
LNA approximates the probability density function of the molecular quantities at steady state with a multivariate Gaussian distribution, and focuses on determining the covariances of the molecular quantities around the steady state, dropping second and higher order moments of the stochastic process.
Also the third method we have mentioned, the 2MA, describes macroscopically

Two Moment Approximation (2MA)

the system's behavior by means of a systemic deterministic trajectory and an intrinsic stochastic noise. Differently from CLE and LNA models, in the 2MA model, the noise actually influences the systemic trajectory. Indeed, the 2MA considers the second moment (covariances) of the stochastic process, and thus takes into account the fact that, in second-order reactions (bimolecular reactions, i.e., dependent on two reactant species), the propensity function combines dependent random variables of molecular quantities in a non-linear fashion. Like considering first order moments (LNA approach) leads to an exact description of unimolecular reactions only, taking into account first and second moments provides an exact solution for first- and second-order reactions, and it leads to a more accurate approximation for third- and higher-order reactions. In describing the trajectory of high-order reaction systems, 2MA provides a better accuracy than LNA. This has the cost of a higher complexity when evaluating the covariance matrix describing the fluctuations.

2.4 Parametrizing artificial chemistries for communication and networking engineering

After having introduced the basics of $\mathcal{AC}$ in Section 2.2, we propose here some modifications of its definition, specifically of its dynamics. We aim at enhancing the use of $\mathcal{AC}$ in engineering communication and networking systems – i.e. making the implementation of $\mathcal{AC}$-based systems simpler and widening the range of obtainable (emergent) properties of $\mathcal{AC}$-based systems.

In general, $\mathcal{AC}$s can be characterized according to their level of abstraction. If there is an isomorphism between molecules or reactions of the $\mathcal{AC}$ and actual molecules or reactions in Chemistry, the $\mathcal{AC}$ can be called *analogous*, otherwise it is called *abstract* [71]. Analogous $\mathcal{AC}$s are considered in the fields of computational Chemistry where the goal is to model chemical processes in the computer as closely as possible. Differently, in this thesis, we refer to statistical and qualitative features of reaction laws and chemical principles, and often we will stray from the rigorousness of actual Chemistry, to gain flexibility in protocol design and predictability of the protocol's behavior.

In this section, we first review a way of calculating the interval between consecutive/consequent reactions (inter-reaction time) in order to simplify the implementation of the reaction algorithm $\mathcal{A}$. Then, we argue about benefits and disadvantages of removing the artificial randomness from system's dynamics (otherwise deliberately added to dynamics of $\mathcal{AC}$s to model actual chemical kinetics). We discuss the benefits of relaxing the determinism of algorithms and protocols for communication and networking, while maintaining controllability of the resulting randomness, and we comment specifically on the effect of choosing *normally distributed* reaction *rates* to generate $\mathcal{AC}$s' dynamics.

We do not promote the sole use of a specific distribution for generating reaction intervals nor of deterministic times. Rather, we want to open up the Chemistry-

inspired approach to new possibilities,[2.14] which can benefit the design and the analysis of communication and networking systems. We briefly discuss properties of a few types of inter-reaction times. However, we believe that such a design choice has to be taken depending on the kind of target applications and environments, thus at the design phase.

2.4.1 Simplified inter-reaction times

Until now, we have explained how to implement $\mathcal{AC}$s to exactly simulate chemical systems (Section 2.2.4 and 2.2.5). Here, we simplify the implementation of $\mathcal{AC}$s in order to make their implementation on CPUs and hardware devices an easy task – the propensity value is approximated and its computation is made dependent on the reaction coefficient and the reactants' concentration only.
As we mentioned in Section 2.2.5, we make use of the Next Reaction algorithm by Gibson and Bruck in [94], which improved the computational performance of the stochastic simulation algorithm. This method relies on identifying a putative next reaction r according to the corresponding next reaction time t_r. We know also that the value t_r is a random variable that is generated from an exponential distribution with mean equal to the reciprocal of the propensity:

Next reaction time

$$t_r = \exp\left(\frac{1}{a_r(\mathbf{n})}\right) \quad (r \in \mathcal{R}).$$

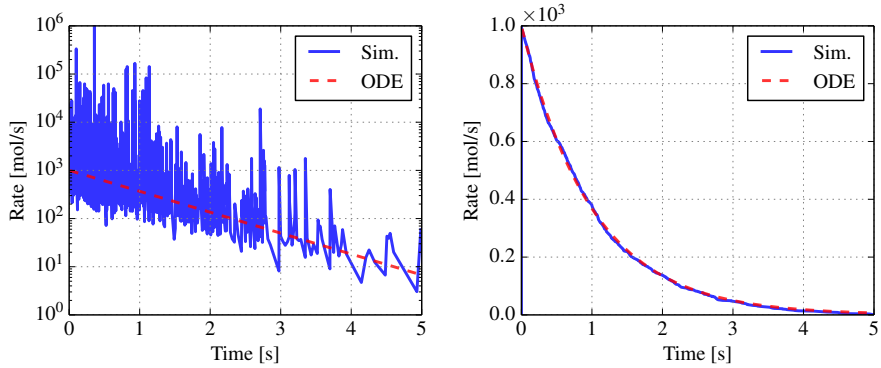
The propensity value is calculated according to (2.11), which directly stems from physical properties of actual reaction systems.
As we aim not at reproducing real chemical reaction systems (*analogous* $\mathcal{AC}$s) but rather at taking advantage of *abstract* $\mathcal{AC}$s to solve computation and communication tasks distributively, we propose to use the variant of the Next Reaction Method that Meyer proposed in [163]. Rather than considering the factorial and stochastic physical terms to calculate the propensity function [247] (as in definition (2.11)), the algorithm makes use of a simplified version of it, calculated as follows:

Simplified propensity function

$$a_r\left(\mathbf{N}(t)\right) = k_r \prod_{s \in \mathcal{S}} N_s^{\alpha_{r,s}} \quad (r \in \mathcal{R}) \tag{2.27}$$

where k_r represents a generic macro coefficient (or reaction rate constant) that directly controls the speed of the concerned reaction r.
We exploit the freedom of programming both the system and the related model. In this way, we simplify the computation of the propensity function. Moreover, we make the relationship between the macro-trajectory (dictated by the LoMA and described by differential rate equations) and the stochastic micro-dynamics (actually exhibited by the chemical reaction system) stronger and more direct.

[2.14] In this chapter, we discuss new distributions of inter-reaction times but still assume the LoMA as the only possible reaction law. We envision the possibility to enhance the flexibility of Chemistry-inspired protocols by means of completely new reaction laws. Appendix A.1 introduces a possible attraction-based law that can simplify the design of some Chemistry-inspired computational mechanisms.



(a) Instantaneous

(b) Average over 0.5s-period

Figure 2.10: *Reaction rate (blue-continuous line) of the unimolecular reaction* $r : S \rightarrow \emptyset$*, when times between two consecutive reactions were drawn from an exponential probability distribution, with the mean-parameter dictated by the LoMA. Instantaneous reaction rate in Figure 2.10(a), and 0.5s-average reaction rate in Figure 2.10(b). The charts detail also the LoMA-predicted rates (red-dashed lines).*

2.4.2 Deterministic inter-reaction times

Adopting deterministic reaction intervals enables us to further simplify the implementation of $\mathcal{AC}$s and, at the same time, to enforce more stable dynamics of the system under control.

We are able to reproduce on computers rigorously correct chemical systems by calculating next reaction times through the exponential probability distribution. However, this method does not always suit the purpose of controlling the dynamics of communication systems. And in fact, drawing inter-reaction times from an exponential distribution implies dealing with highly irregular rates. This is not always desirable. For example, it is known that bursty traffic reduces the performance of computer networks in terms of higher queueing delays, more packet losses, and lower throughput [6, 9, 77].

Issues of exponential reaction times in networking and communications

The exponential distribution may limit the performance of algorithms in different aspects: (*i*) The variance is fixed to the square of the mean value, which is dictated by the applied reaction law (i.e., LoMA). As we know, one of the effects of this is a poor predictability of the dynamics of those systems that work with low concentrations. (*ii*) Statistical measurements such as skewness, excess kurtosis, and mode, suggest that exponentially distributed inter-reaction times often (with very high probability) assume very low values and sporadically (with very low probability) assume high values and on average only, the inter-reaction times match the value that the reaction law (LoMA) predicts.

For example, by using exponentially distributed times to schedule reaction events in the simple unimolecular reaction $r : \mathrm{S} \rightarrow \emptyset$, the instantaneous

rate of reaction r would appear very bursty:[2.15] many, almost instantaneous consecutive reactions and few, very delayed consecutive reactions. This means that, if we associate reaction events to the service process of a traditional packet-queue, the dequeueing and consequent transmissions of packets will be often instantaneous and rarely be characterized by very long delays. The macro-effect and observable behavior is not that different from what we observe when serving the queue with traditional work-conserving policies (serving the queue as fast as possible): the dequeueing rate appears very bursty. Such a phenomenon is captured in Figure 2.10, which shows the rate of the unimolecular reaction $r : \mathrm{S} \rightarrow \emptyset$ when S-species is initialized to 1000 molecules, in the case of exponentially distributed inter-reaction times. Specifically in Figure 2.10(a), the instantaneous reaction rate (associable to the sending rate of a Chemistry-inspired queue) presents very high peaks and only on average, see Figure 2.10(b), it follows the LoMA-predicted value. Note that, in Figure 2.10(a), we have plotted the instantaneous rate on logarithmic scale (y-axis). As imagined, we observe the exponential decay of the reaction rate (linear decaying trend in a logarithmic scale), and high deviations between the actual experienced reaction rate and the theoretically predicted value (indeed, we use the logarithmic scale to deal with this high variability of the instantaneous rate).

A *deterministic* macroscopic mass-action scheduler

As our ultimate goal is not to reproduce dynamics of actual chemical systems on computers, we may implement micro-kinetics that are "less physically correct" but suit better the purpose of controlling the dynamics of communication systems. For example, we may want to control the departure process from a queue via the unimolecular reaction but want the pacing of this process to be constant (i.e., to regularly space out in time the transmission of packets). In generic terms, we desire to have the chance to attain steady trajectories, which do not exhibit oscillations and fast, high-range-amplitude variations, and that follow the theoretically-predicted trajectories as close as possible. That is, we aim to let the reaction algorithm $\mathcal{A}$ be a "precise, deterministic" macroscopic mass-action scheduler.[2.16]

Deterministic reaction algorithm $\mathcal{A}$...

...makes systems' trajectories predictable

To this end, we have to remove the randomness introduced in the next reaction time calculation step and thus, use a deterministic next reaction time – in the initial step 1.c and in the iteratively performed step 5.c in Algorithm 1, we have to schedule the next reaction at the deterministic time given by the mass-action principle (i.e., the time is equal to the reciprocal of the propensity-value rather than being an exponentially distributed random variable that has the propensity as mean-parameter). In this way, the reaction system would follow exactly the LoMA-predicted trajectory (e.g., red-dashed lines in Figure 2.10).[2.17]

[2.15] We refer to the "instantaneous rate" of a reaction as the reciprocal of the period between two consecutive executions of that reaction.

[2.16] We refer to "precision" as the degree to which repeated measurements show the same results under unchanged conditions.

[2.17] The use of deterministic inter-reaction times was also mentioned in [163] as a requirement

...implies a simpler implementation

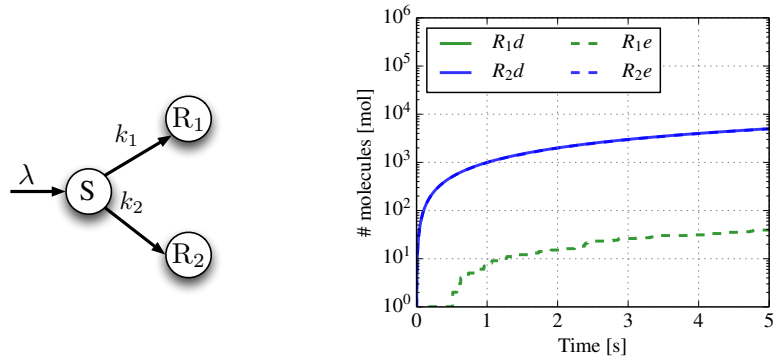
Making the reaction algorithm $\mathcal{A}$ a deterministic mass-action scheduler implies also a simplified implementation. We further reduce the number of required random variables from one to zero for each reaction execution![2.18]
Until now, it seems that using deterministic reaction times to realize a mass-action scheduler is the most precise and efficient method to get trajectories that are close to the LoMA-predicted ones. However, we may experience deviations between predicted and actual trajectories of the system. This is due to the discreteness of the state of the chemical system: In $\mathcal{AC}$s, species change as a consequence of the consumption or the production of one or more molecules (discrete space). However the ODE model describes the changes of the system's state in terms of real quantities (continuous space). This does not represent a problem in the case of sufficiently high concentrations when the deviation between uniform-space prediction and discrete-space measurement are negligible. Instead, in the case of low concentrations, the concerned phenomenon is observable but still does not represent a problem most of the time. However, particularly attention must be paid when reactions "compete" for a shared reactant species whose number of molecules is less than or equal to the number of molecules required to let the concerned reactions be active. To explain the issue we look at the simple example where we have a single species S whose molecules are injected with reaction $r_0 : \emptyset \xrightarrow{\lambda} \mathrm{S}$ and consumed by two reactions $r_1 : \mathrm{S} \xrightarrow{k_1} \mathrm{R}_1$ and $r_2 : \mathrm{S} \xrightarrow{k_2} \mathrm{R}_2$, as depicted in Figure 2.11(a).[2.19,2.20]
According to the ODE describing the dynamics of S-species concentration, at the macroscopic level, we have different outflow rates v_2 and v_3: $c^*_{\mathrm{S}} = \lambda/(k_1+k_2)$ and thus $v_2 = (k_1\lambda)/(k_1+k_2)$ and $v_3 = (k_2\lambda)/(k_1+k_2)$, with the assumption that $k_1 \neq k_2$. Assuming that $c^{(0)}_{\mathrm{S}} = 0$ initially, the egress reactions r_1 and r_2 are inactive. The ingress reaction r_0 is scheduled for time $t = 1/\lambda$ s. At that time, one S-molecule is generated and r_0 is rescheduled for $1/\lambda$ s later. At the same time, both egress reactions become active and are scheduled according to the LoMA. Let us further assume that $\lambda < 1k_2$ and $k_2 > k_1$ and thus, that the r_0-reaction is not fast enough to fill the S-species and the r_2-reaction occurs earlier (with a deterministic inter-reaction time) than the r_1-reaction. At time $1/\lambda + 1/k_2$ s, the only existing S-molecule is consumed by r_2-reaction and both egress reactions become inert again, because there is no molecule left. At each $1/(n \cdot \lambda)$ instant (with $n \in \mathcal{N}$) this cyclic trend starts again and

to enable the correct forwarding of TCP streams by a chemical link layer, working with Fraglets in hypothetical Internet scenarios.

[2.18] The Next Reaction Method by Gibson and Bruck [94] already reduces significantly the number of random variables compared to the Exact Stochastic Simulation Method by Gillespie, i.e. only a single random number per event (reaction execution) instead of $|\mathcal{R}|$ random numbers per iteration (where $|\mathcal{R}|$ is the number of reactions).

[2.19] The symbolism $\emptyset \xrightarrow{v} \mathrm{X}$ signifies the generation of X-molecules at *rate* v, whereas $\mathrm{X} \xrightarrow{k} \emptyset$ signifies the depletion of X-molecules with rate *coefficient* k.

[2.20] In this simple, linear chemical system, R_1 and R_2 species have the sole role to make evidence of the execution of reactions r_1 and r_2, respectively.



(a) Chemical reaction system (b) Dynamics of the number of molecules

Figure 2.11: *Scheduling reactions r_1 and r_2 at deterministic reaction times possibly leads to unexpected dynamics: r_2-reaction prevailed because of its high-valued coefficient k_2 (continuous lines – R_1d and R_2d trajectories). Exponentially distributed random times correctly let r_1 occur sporadically (dashed lines – R_1e and R_2e trajectories). The experiment setup was $\lambda = 10^2$ mol/s, $k_1 = 10^2\ s^{-1}$, $k_2 = 10^4\ s^{-1}$. Lines R_2d and R_2e overlap; line R_1d does not appear because steady at 0.*

thus, an S-molecule is produced with r_0-reaction and then later consumed by the r_2-reaction. In other words, for low concentrations of S-species, by using a deterministic inter-reaction time, the slower reaction never occurs because the faster reaction always (deterministically) fires before the slower one, in this way profiting from all present resources. Figure 2.11(b) illustrates the number of R_1 and R_2 molecules over time in the case of deterministic inter-reaction times (continuous lines) and exponentially distributed inter-reaction times (dashed lines). That is, it shows which reaction occurs and when it occurs. We observe that the R_1-species cannot grow because of reaction r_2 that consumes the only existing reactant molecule S to create R_2-species, making reaction r_1 inert before its execution. Instead, thanks to the randomness derived from using exponentially distributed inter-reaction times, both reactions r_1 and r_2 occur with a frequency that reflect the ratio k_1/k_2 and that depend on the inflow rate λ. For deterministic and exponentially distributed times,[2.21] the number of R_2 molecules reaches respectively 4999 and 4960 (at 5 s). The respective trajectories are almost the same, i.e. lines R_2d and R_2e overlap. On the contrary the number of R_1-molecules is non zero for exponentially distributed times (R_1e-line) whereas it is null for deterministic ones (R_1d-line overlaps the abscissa-axis).

Issues of deterministic reaction times...

...when concurrent reactions that drain a single reactant species have a low number of molecules

[2.21] In Figure 2.11(b), we use the index "*e*" for results obtained with *e*xponentially distributed random reaction intervals, and "*d*" for results obtained with *d*eterministic reaction intervals.

This phenomenon affects models that include (*i*) more than one competing, linear, egress reactions, which have different reaction coefficients, that drain a single species having sufficiently low number of molecules, (*ii*) more than one competing, non-linear, egress reactions that, even if the reaction coefficients are the same, do depend on reactant species having different numbers of molecules and one of these is sufficiently low.

2.4.3 Normally distributed inter-reaction times

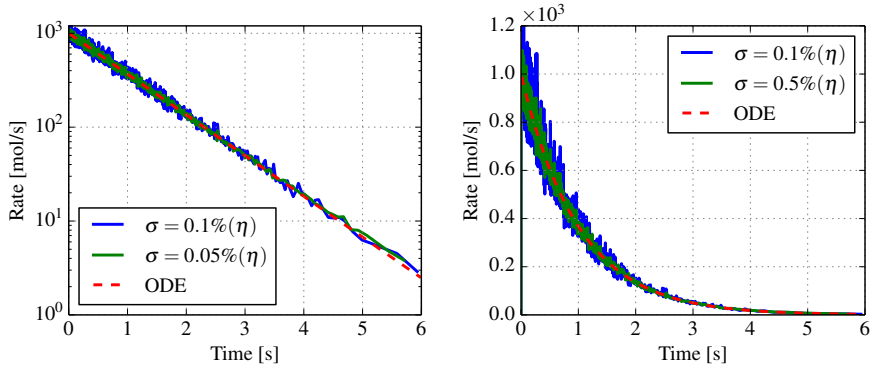
We have seen that deterministic inter-reaction times are suitable to obtain stable dynamics and to simplify the implementation of $\mathcal{AC}$s but, they may lead to the incorrect, permanent inertness of some of the reactions, for certain chemical reaction models and for specific setups. We have also seen that this problem can simply be solved by including a (sufficient) randomness in the next reaction time calculation. This is done automatically when exponential random reaction intervals are used. However, the use of exponential random times has the cost of a less regularity between the scheduled events (e.g., intervals between packet transmissions). In this section, we discuss normally distributed reaction times (i.e., characterized by a Gaussian/normal distribution) to gain controllability of the "randomness level".

In general, referring to networking and communication systems, there exist reasons why designers could desire to relax the determinism of their algorithms and protocols: (*i*) to increase their robustness – e.g. gossip-style protocols, (*ii*) to help to break symmetry – e.g. leader election in anonymous networks, (*iii*) to help resolving access conflicts – e.g. MAC-protocol family, (*iv*) to reduce interferences – e.g. clock dithering for reducing radiated noise. Thus, we propose to not remove randomness from the next reaction time calculation altogether but rather to include it and maintain some sort of controllability of the level of randomness in the algorithms' dynamics.

More than one probability distribution may be candidate to be used to generate reaction times, depending on the underlying physical aspects, the general observations, and the engineering requirements of communication systems. We believe this decision can be postponed to the design phase of each Chemistry-inspired communication system, and generally should concern basic features of random generator functions. Here, we limit to recognize the normal distribution as as good candidate, and briefly comment on its properties. Besides being analytically tractable, the normal distribution guarantees controllability of the level of randomness. That is, through the variance parameter, the designer can control the level of fluctuations affecting the trajectories of the $\mathcal{AC}$. In the example of a packet queue whose departure rate is controlled by the uni-molecular reaction, the use of the Gaussian distribution to generate reaction times enables us to define the average pacing time that has to characterize the packet-transmission process. At the same time, it enables us to enforce a certain randomness in order to prevent side effects such as the synchronization, and to define a tolerance range for the intervals between packet transmissions.

Gaussian reaction times...

...to enable randomness control



(a) Logarithmic scale (b) Linear scale

Figure 2.12: *Instantaneous reaction rate of the unimolecular reaction $r : S \rightarrow \emptyset$, when the interval between two consecutive reactions was drawn from a normal distribution, with a mean dictated by the LoMA $(1/a_r)$ and a deviation of $0.05\%(1/a_r)$ (green line) and $0.1\%(1/a_r)$ (blue line). The charts also show the LoMA-predicted deterministic reaction rate (red-dashed lines).*

Figure 2.12 shows the rate of the simple unimolecular reaction $r : \mathrm{S} \rightarrow \emptyset$ in the case where inter-reaction times are drawn from a Gaussian distribution: The instantaneous transmission rate is smooth and very close to the predicted trajectory. Additionally, the level of the randomness can be trimmed through the standard deviation parameter σ.[2.22] In this simple example we make the deviation be proportional to the LoMA-predicted value, i.e., a percentage (0.05% and 0.1%) of the reciprocal of the simplified propensity.

The fact that normally distributed variables can assume negative values (not reasonable for modeling time) can be solved through the use of its truncated, re-normalized version. Alternatively, designers can choose values of mean and variance such that the probability to have negative times is negligible.

The previously discussed problem concerning concurrent reactions sharing a low-populated reactant may affect the dynamics of chemical systems also in the case of normally distributed inter-reaction times. When the level of randomness (quantified by standard deviation σ) is not sufficient to overcome the rate difference (defined by the reaction coefficients of concerned reactions), the faster reaction prevails on the slower one and fires always (with a very high probability) before the slower one.

[2.22] The stochasticity characterizing the algorithm appears as a fluctuation on the reaction rate around the average value with a controllable variance.

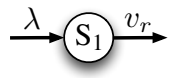
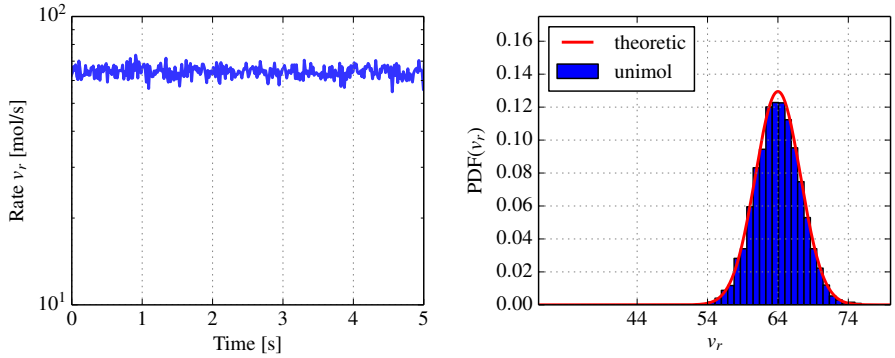


Figure 2.13: *Chemical system made of the unimolecular reaction $r : S \to \emptyset$ and species S, which is fed at rate λ.*



(a) Rate trajectory (b) Rate PDF

Figure 2.14: *Results from simulating the reaction system in Figure 2.13 for $\lambda = 64$ mol/s, by using an artificial additional Gaussian noise with $\sigma = 0.05\%(n_S)$. (a) Reaction rate v_r. (b) Estimated and theoretical PDF.*

2.4.4 Normally distributed reaction rates

Eventually, we comment on $\mathcal{AC}$s with enforced, normally distributed random rates. As we show next, "generating chemical kinetics" by means of inter-reaction *rates* that are *normally distributed* means increasing analyzability and direct controllability of Chemistry-inspired communication systems.

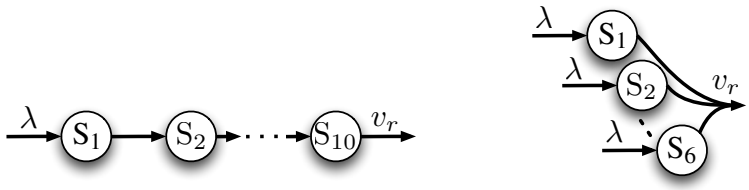
In our attempt to control communication systems by means of $\mathcal{AC}$s, we almost always work with reaction rates (= en- de-queueing rate, RF pulse transmission rate, *etc.*). For this reason, we suggest to enforce rate trajectories that are deterministically dictated by the LoMA according to (2.6), and to add an artificial randomness in terms of an additive Gaussian noise. The mean-parameter of the noise should be null, whereas the variance-parameter (standard deviation) is a free variable, which constitutes a design parameter that affects directly the behavior of the system.

For example, we can set the standard deviation to a constant value such that the level of randomness is always the same, independently of the amount of involved molecules. Alternatively, we can set the standard deviation to a constant percentage of the value dictated by the LoMA. In this way, we obtain a constant relationship between expected value and standard deviation, i.e., a

sort of constant Signal-to-Noise Ratio (SNR). For example in Figure 2.14(a), we report the output rate v_r of the unimolecular reaction system represented in Figure 2.13. In the concerned simulations, the system was constantly fed at rate $\lambda = 64$ mol/s, and inter-reaction rates were extracted from a Gaussian distribution with the mean dictated by the LoMA (i.e., the runtime amount n_S of S-molecules according to (2.27)) and standard deviation set to $0.05\%(n_S)$. In Figure 2.14(b), we show the normalized distribution of the output reaction rate v_r for 6400 samples extracted from simulations – the estimated Probability Density Function (PDF) of v_r. Figure 2.14(b) details also the theoretical $\text{PDF}(x) = 1/(2\pi)e^{-(x-64)^2/(0.05\%(64))^2}$. The estimated PDF matches well the theoretical Gaussian PDF.

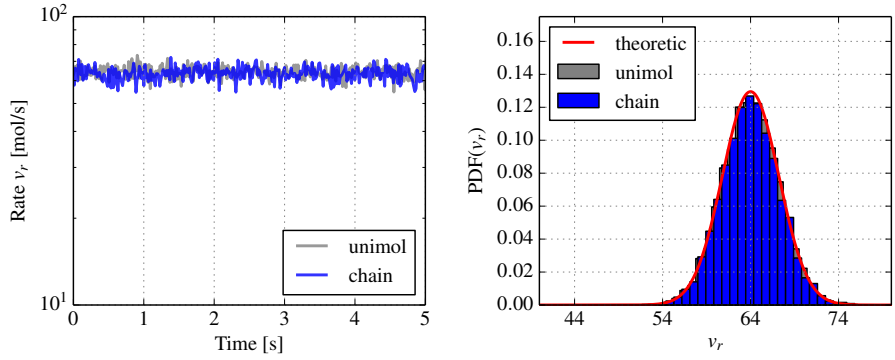
Both analyzability and controllability are not altered for more complex systems, which include a chain (cascade) of simple unimolecular reactions (e.g., the chemical reaction model in Figure 2.15(a)) and non-linear reactions (e.g., the chemical model in Figure 2.15(b)). Figure 2.16 shows the results from simulations for both reaction networks when using Gaussian reaction rates with mean set equal to the propensity value a_r (calculated according to (2.27)), and standard deviation set to $0.05\%(a_r)$. To make results comparable (i.e. same output reaction rate v_r), the chain of ten unimolecular reactions was again fed at constant rate $\lambda = 64$ mol/s, whereas all reactants of the 6-order non-linear reaction were fed at constant rate $\lambda = 2$ mol/s. The instantaneous output reaction rates experienced during the simulations of the reaction-chain (Figure 2.16(a)) and of the non-linear reaction (Figure 2.16(c)) systems are compared with the already-discussed results concerning the simple unimolecular reaction. Similar trajectories are observed. In Figure 2.16(b) and Figure 2.16(d), respectively for the reaction-chain and the non-linear-reaction systems, we compare also the theoretical PDF calculated for mean=64 and standard deviation=3.2, and the one estimated from 6400 rate samples extracted from simulations. We finally compare these estimations with the simulation result of the unimolecular-reaction system. The estimated rate PDF in the case of unimolecular reaction, chain of unimolecular reactions, and non-linear reaction matches well the theoretical Gaussian PDF.

The choice of normally distributed inter-reaction rates is not meant to be a rule. Different design requirements may suggest different distributions for generating chemical kinetics. For example, deterministic inter-reaction times are useful to minimize the complexity of the reaction algorithm or to impose theoretical deterministic dynamics (e.g., for rate control / packet pacing applications). However, the deterministic approach should be used only in those conditions that exclude the inertness problem of concurrent low-reactant reactions. Instead, exponentially distributed reaction times are convenient for enforcing the randomness in the system's dynamics (e.g., for media access control application). This randomness is characterized by a fixed variance and solves always the inertness problem of concurrent low-reactant reactions, without any specific decision from the designer's side. A generic solution that ranges over these two extremes is adding a controllable Gaussian noise to the deterministic

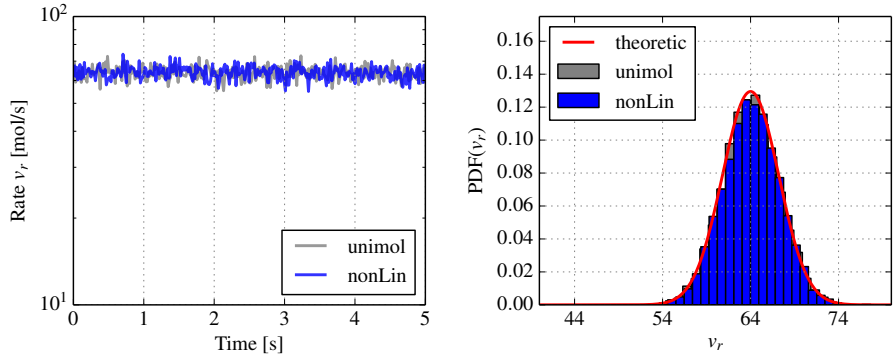


(a) Chain of unimolecular reactions (b) Non-linear reaction

Figure 2.15: *More complex chemical reaction network: (a) chain of ten unimolecular reactions, and (b) a 6-order non-linear reaction. Systems are fed at constant rate λ.*



(a) Chain: Rate (b) Chain: PDF



(c) Non-linear reaction: Rate (d) Non-linear reaction: PDF

Figure 2.16: *Results from simulations of systems shown in Fig.2.15. (a) Output rate v_r, and (b) estimated and theoretical PDF(v_r) of the reaction-chain system in Fig.2.15(a). (c) Output rate v_r, and (d) estimated and theoretical PDF(v_r) of non-linear-reaction system in Fig.2.15(b). The chain of 10 unimolecular reactions is fed at $\lambda = 64$ mol/s, and the reactants of 6-order non-linear reaction are fed at $\lambda = 2$ mol/s. Reaction rates' randomness is enforced Gaussian with $\eta = a_r$ and $\sigma = 0.05\%(a_r)$. The results are compared with those obtained with the unimolecular reaction in Fig.2.13.*

mass-action-defined rate. This approach enables the controllability of the level of randomness, a common practice, for example, in the Address Resolution Protocol (ARP) [195] for randomly-paced address probing, or in clock-dithering technique to stop undesired limit-cycle oscillations of a system, or alleviate the resonance jump phenomenon in sliding-mode control applications [152].[2.23]

2.4.5 Reaction algorithm $\mathcal{A}$ for communication and networking

We finally present the reaction algorithm $\mathcal{A}$ that we have used for the work presented in this thesis (simulations and real-world experiments), refer to Algorithm 2. This algorithm is an extension of Algorithm 1 (of the stochastic simulation algorithm published in [94]) and of the revised version proposed by Meyer in [163], and differs in a few aspects: (*i*) Algorithm 2 takes advantage of the simplified propensity a_r as defined in (2.27) (Algorithm 2, steps 1.c, 4.c, 4.d). In this way, we keep the implementation complexity low but maintain the fidelity of the macro-behavior to the mass-action principle. (*ii*) Steps of Algorithm 2 (steps 1.c and 4.c) concerning the time calculation do not impose the generation of exponential random values but rather include the possibility to have both deterministic and random times, and for the latter case to decide the distribution and, possibly, the level of randomness. This enables a higher design flexibility. (*iii*) Algorithm 2 includes an idle phase in which the system "sleeps" until the next reaction time is actually elapsed (step 2). This enables working in continuous time and keeping track of the real-world time, instead of dealing with a sampled simulation time.

2.5 From science to engineering

In this last section, we discuss the phenomena that arise when importing $\mathcal{AC}$, a tool that natural scientists use to simulate real chemical systems, into the engineering process of communication and networking systems. Specifically, we comment on the implementation of Algorithm 2 on CPUs and hardware devices, on the consequences of its modified step 2, and on how we interface the virtual dynamical system (i.e., the underlying chemical reaction network) to the real environment (i.e., the communication or computer network).

[2.23] Traditionally, dithering is a certified interference-limiting technique in EMC-compliant management of power-supply system. Clock-dithering is used to reduce the radiated noise at any single frequency by spreading the frequency spectrum of the main oscillator. This small variation in the switching frequency decreases the narrow-band emissions by changing the frequency content emitted by the device. In practice, such a technique aims at slightly modulating the oscillator's fundamental frequency. The effect of this is that the peak energy is shared among multiple fundamental frequencies. In this way, while the total EM energy remains the same, the narrowband energy is decreased [61, 78, 169]. (Acronym "EM" stands for ElectroMagnetic, "EMC" for ElectroMagnetic Compatibility.) Today, clock-dithering technique has a wide range of applications in general control systems [152].

Algorithm 2 Reaction algorithm $\mathcal{A}$ for Chemistry-inspired communication and networking systems
A variant of the stochastic simulation algorithm published in [94]. The algorithm uses simplified quantities and gives more flexibility in choosing the randomness that affects the system's trajectory.

1) Initialize:
   a) set the initial amount of molecules;
   b) calculate the propensity a_r according to (2.27) $\forall r \in \mathcal{R}$;
   c) set a putative reaction-execution time $t_r = \mathcal{F}\{1/a_r, (\sigma)\}$ $\forall r \in \mathcal{R}$;
   d) store values t_r in an indexed priority queue (the first stored element has the next reaction time) and store the related values a_r.
2) Wait as long as $t < t_i$, where t_i is the least reaction time among all reactions $\in \mathcal{R}$.
3) Change the number of molecules to reflect the execution of reaction r_i.
4) Update all reactions, r_j, that depend on the executed reaction r_i:
   a) temporarily store the old value $a_j^{old} = a_j$;
   b) calculate the new propensity a_j, according to (2.27);
   c) if $r_j \neq r_i$, scale the reaction execution time by setting it equal to $t_j = (a_j^{old}/a_j)(t_j - t) + t$;
   else if $r_j = r_i$, set the reaction execution time $t_j = \mathcal{F}\{1/a_j, (\sigma)\} + t$;
   d) store the calculated reaction execution time t_j in the indexed priority queue and store the related value a_j.
5) Go to Step 2.

Notes:

(*i*) The variable t reflects the current (virtual chemical) time.

(*ii*) The term $\mathcal{F}\{\eta, (\sigma)\}$ indicates any random generator function (e.g. exponential, Gaussian, *etc.*) with expected value η and, if necessary, characterized by the desired level of randomness (standard deviation) σ. Such a function refers to the deterministic generation of times too (formally, Gaussian generator with $\sigma \to 0$).

(*iii*) Molecular species represent mere counters. Step 3 implies to decrement all counters related to the reactant-species (species appearing in (2.2) on the left-hand side of the arrow) and to increment all counters related to the product-species (species appearing in (2.2) on the right-hand side of the arrow).

We start by observing that we make use of the algorithm $\mathcal{A}$ *to schedule and control events in communication networks*. This differs from what the majority of the works in the $\mathcal{AC}$-context propose – making use of a reaction algorithm $\mathcal{A}$ *to simulate chemical reaction systems in the shortest possible time*. Once a reaction has been scheduled to fire at the next reaction instant $t_r = t + \tau_r$ and no other events that affect the concerned timer happen, we actually have to wait τ_r s until the reaction is executed (i.e., update dependent species and recalculate the next reaction time t_r). We observe further that events in the communication system occur at a physical time whereas events characterizing the dynamics of the underlying chemical reaction system occur at a "virtual time". This conceptual difference is captured by the difference between step 2 and step 3 in Algorithm 1 and step 2 in Algorithm 2.

Physical and virtual time

The mapping (ratio) between virtual and physical time represents another parameter to be defined when designing Chemistry-inspired algorithms and protocols for communications. In theory, the virtual-to-physical time ratio can be any value (but, for consistency, it should be the same in all nodes participating in the considered reaction network). In practice, it must accommodate some requirements imposed by the application-scenario and the implementation-environment. The reaction algorithm $\mathcal{A}$ has to be implemented on CPUs, or more generally on hardware. This means that, if virtual and physical times are coupled by a constant ratio, the reaction intervals in actually-implemented chemical systems exhibit a lower limit. This limit is determined by the specs of the hardware where $\mathcal{A}$ is implemented, i.e., the minimum physical time required by the hardware to perform the concerned computations. We experience deviations between actual and theoretically predicted trajectories as soon as the system works with too big reactant concentrations and reaction coefficients. "Big" means being sufficient to make the reaction interval smaller than the minimum time required to compute the interval itself. Figure 2.17 shows the saturation-like effect of scheduling a simple unimolecular reaction $r : \mathrm{S} \xrightarrow{100} \emptyset$ given the initial condition $c_{\mathrm{S}}(t = 0) = 10^5$. The system was implemented on a simulated hardware that exhibited a minimum computational time of $10\,\mu$s and the virtual-to-physical time ratio was set to one. As we can see, the number of S-molecules initially decays linearly and proportionally to the minimum computational time rather than exponentially; the reaction rate is initially constant and equals the reciprocal of the minimum computational time.

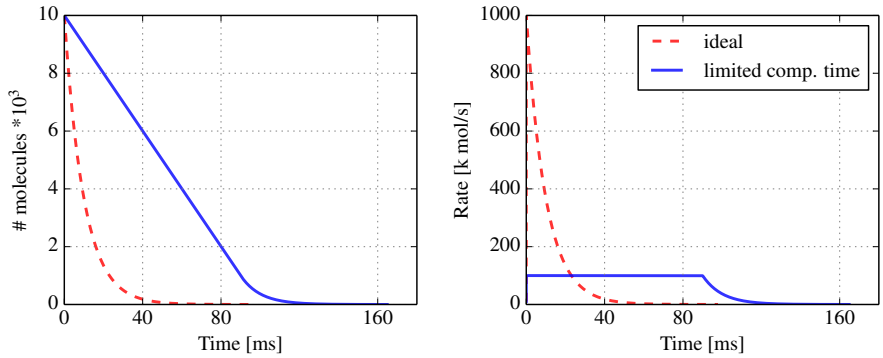
Virtual-to-physical-time ratio...

...a design parameter

Limited (minimum) computational time

Deviations between actual and theoretically predicted trajectories are possibly observable also due the granularity that an actual timer exhibits: We cannot schedule exactly a timer at $1/a_r$ seconds but rather we have to round it to $(1/a_r)_{g_t}$ seconds, where g_t represents the minimum possible time step. Figure 2.18 shows the effect on the number of S-molecules and on the rate in the case the time granularity the hardware allows is $1\,\mu$s. These side-effects must be considered in the design phase, and be avoided by a correct design of aspects concerning the $\mathcal{AC}$-system itself (e.g., reaction rates and upper bounds

Timer granularity



(a) Min. comp. time affects concentration (b) Min. computational time affects the rate

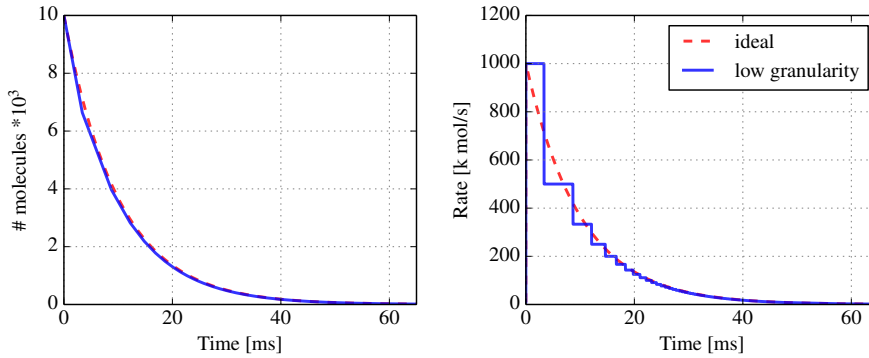
Figure 2.17: *Saturation-like phenomenon affecting the time-evolution of the number of S-molecules and of the reaction rate of the reaction system* $r : S \xrightarrow{100} \emptyset$*, for* $c_S(t = 0) = 10^5$*, when the minimum achievable reaction interval was 10* μs *(minimum hardware computational time).*

to the number of molecules) and of implementation details (e.g. virtual-to-physical time ratio).

Quantity-to-molecule ratio...

...a design parameter

Another phenomenon we may observe when not carefully implementing $\mathcal{AC}$-based communication and networking systems is the discreteness derived from mapping external quantities, discrete or continuous, into an integer number of virtual molecules. To be more concrete, let us refer again to the example where a simple unimolecular reaction is used as a "controller" for the service process of a traditional queue. For fix-sized packets, we can decide to map one packet to a single virtual molecule. However, when applying to actual queues in computer networks (e.g., traffic control of linux kernel queue), we deal with variable sized packets. To this end, we can decide to map each bit to a single virtual molecule. However, for (realistic) rates of Gbits per second, such a strategy is not feasible due to constraints in terms of computational costs (e.g., how fast the hardware has to, and can, process the required chemical computations) or communications (e.g., how many interactions among controller and controlled modules occur). In fact, Gbps-rates would require the hardware to perform the $\mathcal{AC}$-related computations in nano seconds. Instead, to keep the computational cost confined, we map a certain amount of bits into a certain (smaller) amount of molecules (e.g., by mapping 1000 bits into a single molecule, we can deal with Gbps-rates by requiring the hardware to perform computations in micro seconds), at the price of a more noticeable effect of the discretization. That is, how the external "physical" quantity is mapped into a certain amount of molecules represents a parameter to be defined in the design process. This design choice must take into account the available computational resources and



(a) Time-granularity affects concentration

(b) Time-granularity affects the rate

Figure 2.18: *Low-resolution phenomenon affecting the time-evolution of the number of S-molecules and of the reaction rate of the unimolecular reaction system* $r : S \xrightarrow{100} \emptyset$, *for* $c_S(t = 0) = 10^5$. *The time resolution of the hardware was set to* $1\,\mu$s.

the nature of the input quantity itself. Dually, designers have also to define how the $\mathcal{AC}$ interacts with the communication system – e.g., how many bytes the execution of a single reaction authorizes.
Generally, the aforementioned phenomena do not represent a problem, neither in the design nor in the analysis phase. However, their effect might be visible in the system's dynamics. For this reason, designers should take into account the granularity of timers, the computational performance of the used hardware, and the discreteness derived from mapping external quantities (e.g., variable-sized packets, measured temperatures) into a number of virtual molecules. We cannot give here a precise "recipe", which is valid generically for all applications and contexts. We limit to observe that a safe solution must be characterized by design parameters, e.g. virtual-to-physical-time and quantity-to-molecule ratios, such that the discussed phenomena turn out to be negligible. As we will see in the next chapters, the correct calibration of the discussed design parameters is simple and actually represents for designers an "added value" rather than a problem to face (e.g., designers can calibrate these parameters to preserve resources and control the computational cost of running the $\mathcal{AC}$).

2.6 Conclusion

In this chapter, we have introduced and formalized Artificial Chemistry ($\mathcal{AC}$), and have started to understand how $\mathcal{AC}$ can be applied in both the analysis and the design of communication and networking systems.
Traditionally, once the networking algorithm has been designed, the state-of-

the-art method to analyze it is building an abstract fluid model, which focuses on the system's average trajectory. However, choosing which aspects of the system can be ignored for the sake of analyzability and which ones should instead be modeled is one of the most critical tasks in the engineering process. The benefit of the Chemistry-inspired engineering framework is that a fluid model of the executable system can be generated automatically in the form of Ordinary Differential Equations (ODEs). This desirable property stems from the direct coupling between quantity and rate, which is enforced by the reaction law, i.e., the law of mass action.

Once we have the fluid model, we can adopt classical queueing- and control-theory tools *and/or* exploit novel analytical tools borrowed from Chemistry. Relying on the ODE-approximation, we are able to study steady and transient states, stability, convergence, and stochasticity of systems. In this way, we obtain a description of the system that is very close to that used in control theory and engineering. At the same time, we gain also completely new analytical tools such as the Deficiency Zero Theorem, which enables us to infer the dynamical properties of the system from observations on its structure.

$\mathcal{AC}$ can also represent an automated procedure to implement desired requirements and achieve a predefined behavior at the macroscopic level. Indeed, with this framework, we can design a macro behavior by drawing reaction networks; we do not have to care each time about how to implement the required microscopic actions but rather the implementation process is automated: Once the reaction network that specifies the behavior of the $\mathcal{AC}$ (thus the behavior of the linked communication system) is defined, it can be compiled to a program that executes reactions (thus events in the system) online, thanks to the reaction algorithm $\mathcal{A}$. For the sake of controllability, analyzability, and implementation-simplicity in the context of networking and communications, we propose to deviate from reproducing faithfully what happens in real chemistry and to stick only to the constraint of implementing a mass-action scheduler.

In this chapter, we have introduced all the tools we are going to use in the rest of the thesis, where more complex networking and communication mechanisms are treated. For the sake of clarity, we have frequently explained concepts by means of a simple example – the mapping of a traditional packet queue with a (unimolecular) chemical reaction – without taking into account more complex/realistic scenarios. However, all introduced concepts and tools can be generalized to analyze and control arbitrary interaction-based systems (e.g., complex packet-queueing systems).

Chapter 3

Artificial Chemistry to Shape the Traffic in Computer Networks

"Complexity must be grown from simple systems that already work."

Kevin Kelly

In this chapter, we show how to use artificial Physical Chemistry in designing traffic shaping algorithms for computer networks, and demonstrate the advantages of such a design approach in the context of traffic control. We introduce simple principles and chemical motifs that help the design process and the analysis phase of traffic controllers. By using the introduced motifs, we construct and study distributed adaptive mechanisms. At the end of this chapter, we will have presented a family of distributed rate controllers that can be easily customized, and have predictable performance and dynamics.
This chapter is structured as follows: In Section 3.1, we first give an overview about where flow-control and generic traffic-shaping processes are useful in today's computer networks. In Section 3.2, we discuss in detail the use of the chemical mass-action principle in the context of packet-switched networks and queueing systems. In Section 3.3, we introduce a commonly used mechanism, the token-bucket, model it in $\mathcal{AC}$-terms, and finally introduce an analogous chemical motif, the Enzymatic model, with interesting properties stemming from the underlying mass-action principle. For such an Enzymatic traffic rate controller, we provide an in-depth analysis and present results from simulations and real-world experiments. Starting from the Enzymatic model, in Section 3.4, we develop a class of adaptive and distributed traffic shaping algorithms that implement *emergent-control* strategies; we study their stability and sensitivity, and present experimental results in both OMNeT-simulator and real-world environments. Eventually in Section 3.5, we discuss the insights developed throughout this chapter.

3.1 Introduction

Packet-switching

In packet-switched networks, such as the Internet, information is divided into sized packets, which are sent individually through the network and are then reassembled in the original order at the receiver. Packet-switching enables flow multiplexing among senders as well as flow distribution across multiple simultaneous communication routes between each sender and receiver. In this way, communication systems become (more) robust and guarantee an efficient sharing of the available channel capacities. The data flows (data streams, sequences of packets) have therefore a variable rate, delay, and throughput, depending on the network's capacity and the traffic load on the network. Rate, delay, and throughput are directly affected by the experienced buffering and queueing along the traverse router/switches (which are introduced to enable statistical multiplexing of flows, and mitigate effects of rate burstiness).[3.1]

In the context of packet-switching, dynamic capacity-allocation techniques and rate control mechanisms play a crucial role in the resource allocation and the statistical multiplexing of flows. The related literature consists of studies, which vary in (*i*) the application context they target (more or less specific), (*ii*) how they are derived (empirically or formally), and (*iii*) the problems they try to solve at different points of the communication stack and at different locations in the network.

Interestingly, a few key concepts and mechanisms recur and interlace: rate (and congestion) control, packet pacing, flow interaction and multiplexing, rate smoothing and burst absorption. Before revisiting these aspects with the chemical approach, we summarize the main methods and mechanisms that have been introduced until today to enable fair and efficient access to network resources.

3.1.1 Related works

Three network-dynamics-control paradigms (have) contribute(d) significantly to shaping the traffic in packet-switched networks: (*i*) end-to-end transport control, with transport-layer protocols such as Transmission Control Protocol (TCP) and its many versions of congestion-control algorithms, (*ii*) in-network queueing/scheduling with algorithms for Active Queue Management (AQM) and scheduling, and (*iii*) local rate-control mechanisms such as packet pacing and rate control/limiting. To narrow the focus and limit the length of this section, we briefly review only works in the context of local traffic control. However, as in the chapter we will refer often to versions, mechanisms, and features of TCP and AQM, we mention in Appendix A.2 the different means

[3.1] This differs from what happens in the circuit-switching paradigm, the other methodology of implementing communication networks, which sets up a finite number of dedicated connections of constant bit rate and upper-bounded delay between nodes, exclusively for each communication session.

of traffic management and explain basics of TCP's congestion control, AQM, and scheduling algorithms.

From a microscopic perspective, control of network dynamics takes place fundamentally by means of three mechanisms: packet pacing, rate limiting (capping), and rate control. These mechanisms work on a local scale and, although strictly coupled, differ semantically on the action performed: Packet pacing is the act of adding a certain delay between consecutive transmissions. Rate limiting is the act of capping the rate at which transmissions of packets occur. Finally, rate controlling is the act of continuously adapting the rate at which events occur in order to match specific targets (e.g., the respect of a rate limit or the avoidance of congestion). However, the difference between rate limiting and control becomes blurred in the literature when applying control to the collective access rate of a group to a shared resource (e.g., distributed access from multiple entry points to a cloud). As this is our focus, in the rest of the chapter, we also consider these two terms interchangeably.

Packet pacing

Packet *pacing* has been repeatedly proposed to limit traffic burstiness and thereby improve network utilization. The idea is to allow enough time between the transmission of packets, at network edges or end-hosts, so that high-frequency components are removed from the queue arrival process when two or more flows merge, alleviating the need for deep buffers. Pacing in the context of TCP was suggested in [265] for alleviating burstiness due to ACK compression, and then proposed in [188] to avoid falling back to slow start after a packet loss, or when resuming activity after an idle period. A major concern has often been fairness among coexisting paced and non-paced flows [6]. However, Cai *et al.* demonstrated in [45] after exhaustive experimentations that pacing (*i*) can significantly improve the performance of a moderately loaded small-buffer network, (*ii*) brings little negative impact on short-lived flows in a nearly drop-free environment, and (*iii*) is able to maintain the same fairness as achieved on the current Internet, among paced and non-paced flows. In [47], Cai *et al.* have used adaptive pacing to proactively shape the traffic based on the level of single queues, in order to bound delay and to achieve higher throughput than that obtained with pacing at end-systems (e.g., like in [76] instead). A similar approach has also been presented by Alizadeh *et al.* in [9]. Pacing solutions have used different ways to calculate the (intentionally) added delay: In [12, 45–48], a more or less complex non-linear function of the queue fill-level was used. In [6], the delay was calculated according to the ratio of the current round-trip time and the congestion window size (see Appendix A.2 for details on these quantities). And in [219], the pace was based on the combination of the packet arrival curve and the packet deadline curve within the same pacing interval. Pacing solutions have varied also in terms of proposed deployment and technology – e.g., software-based [46] and hardware-based (at the Network-Interface-Controller level) [9]. Finally, pacing solutions have differed in the context or objective for which they have been proposed: Content Delivery Networks (CDNs) [51], traffic control [106], queue management [262], data centers [9], and others.

Interestingly, packet pacing also simplifies the analysis, especially in the case of multiplexed/interacting traffic flows. Intuitively, pacing makes packet flows "more fluid". Beside smoothing packet rates, this makes dynamics models (based on fluid abstractions) more accurate.

Rate control

The dual mechanism to pacing in traffic management is rate control. The simplest application scenario is regulating the access of a single user to a certain resource in order to impose an upper limit on the access rate. Probably, the simplest, most known way of doing this is to use a Token-Bucket policer [120, 234] in order to limit the maximum average rate together with the maxim burst size allowed to pass through (we explain the Token-Bucket mechanism in detail in Section 3.3.1). Alizadeh *et al.* have proposed in [9] a High-bandwidth Ultra-Low Latency (HULL) architecture for Data Centers, where rate capping at less than link capacity, based on a Token-Bucket mechanism, is used to leave "bandwidth headroom" for latency sensitive traffic and gain a significant reduction in latency. In [200], Raghavan *et al.* proposed a distributed rate-limiter for regulating access to cloud resources. In their Flow Proportional Sharing (FPS) approach, distributed collaborating instances of token-bucket rate-allocators exchange statistics and "shift" capacity slices between congestion-aware flows.
Another way to control the traffic is to use a sliding-window-based mechanism in order to limit the number of packets (or bytes) that can be transmitted at a time. This is the kind of mechanism used in TCP's congestion control (see Appendix A.2 for a short review on TCP's control mechanisms). Window-based mechanisms are used in different contexts, for example to distributively limit the access rate to data centers. Abu-Libdeh *et al.* proposed in [2] window-based rate-control policies (AJIL) in order to control the sharing and virtualization of resources within the cloud. They enabled traffic differentiation across multicast channels by means of rate-controlling reactors that enforce slowdown policies when aggregate traffic rates reach a predefined critical threshold.
Rate controllers are used in combination with other measures to attain more controlled or predictable latencies between Virtual Machines (VMs) and their resources (e.g., storage). These proposals aim at coping with the shared nature of the network in today's multi-tenant data-centers, and want to enable current cloud providers to offer network guarantees within capacity allocations (e.g., Amazon EC2 cannot provide performance guarantees of network resources) [16]. The proposals try to counteract the burstiness of network load between VMs and storage in an oversubscribed data center, which can cause VM outages, unpredictable application performance within the cloud, and uncontrolled costs, too. Recent solutions propose programmable rate limiters between pairs of communicating VMs to provide rate guarantees to tenant VMs. For example, Gatekeeper [204] uses rate limiters and guarantees capacity to VMs by implementing a weighted packet scheduler on data senders, and according to the minimum and maximum rates on data receivers. EyeQ [123] proposes an end-to-end rate control mechanism by implementing rate limiters on the data sender and congestion detectors on the data receiver. Oktopus [23] solves the distributed-rate-limiting problem at end-host hypervisors with a

reservation-based strategy in order to allocate capacity to each VM, without explicit bandwidth reservations at datacenter switches. Hadrian [24] uses a weighted-sharing strategy (extended version of RCP [74]) to control intra-tenant traffic and to improve the system efficiency by allowing tenants to occupy unused capacity. Finally, in IOFlow [228] rate control is applied to enforce end-to-end policies (thus dictate performance) of traffic related to an IO request from a VM to a storage back-end.
Rate control is an important means for allocating capacities and providing weighted (statistical) fair sharing. Rate control mechanisms are usually deployed as standalone policers in combination with other measures, or they are intrinsic parts of protocols, like TCP, that try to occupy shared resources. They are much more popular than pacers because operating on average per-flow behavior rather than on individual per-packet behavior. On the other hand, pacers make flows much more "well behaving" and, as we explain further on, in our opinion deserve more attention and deployment opportunities.

3.1.2 Space and scope of our contribution

In this chapter, we present a Chemistry-inspired framework to design and analyze mechanisms to shape traffic in packet-switched networks, and thereby keep their dynamics controlled and predictable. Interestingly, the chemical framework includes by definition all key aspects we have introduced so far: rate control, (adaptive) packet pacing, flow interaction and multiplexing, rate smoothing and burst absorption.

LoMA scheduling...

...to make explicit and programmable packet-flow interactions

We adopt Chemistry-inspired adaptive packet pacing (affected adaptively through the LoMA) in order to avoid *implicit* and *uncontrolled* packet-flow interactions. LoMA-scheduling is a non-work-conserving discipline and, unlike traditional work-conserving scheduling, binds mathematically the queue's fill-level with the packet forwarding process. One one hand, this enables us to describe complex systems without extrinsic modeling. On the other hand, it allows us to design and describe (complex) dynamical systems by means of *explicit* and *predictable* interactions (reactions) among packet flows and queues.

Chemical traffic controllers

We actually employ $\mathcal{AC}$s to shape the Internet traffic. We do not require to transform standard packets into chemical packets (Fraglets [230]), nor to work with actual chemical networking protocols, nor to replace existing layers or elements in today's networking infrastructures. Rather, we advocate $\mathcal{AC}$s as modules to control packet-departure processes, enabling in this way the *controlled coexistence* and the mathematically-tractable interaction of traffic flows, as well as the cooperation/competition of Internet participants. We propose to integrate the chemical execution engine into a traditional traffic-control framework (e.g., Linux kernel `tc` module), where species concentrations are mere counters of packets or bytes and reactions formalize how these counters interact.
By introducing and motivating step-by-step the various design patterns (i.e., chemical pacing policy, chemical rate limiter) and principles (i.e., mass-action,

Customizable distributed rate controllers

mass-conservation, flow-conservation principles), we derive a class of distributed rate controllers that can be easily parametrized, extended, and customized, for various purposes and in different operational environments, for example (*i*) at the client side to enable service differentiation among user's flows (e.g., management of DSL users' traffic), (*ii*) among clients for a distributed coordination between aggregate flows, (*iii*) in an intra-domain network to manage the allocation of capacities to admitted flows (e.g., distributed rate control for a cloud environment), and (*iv*) at the server end to regulate the shared access to a resource (e.g., differentiation of user- from control- traffic in a data center). The distributed chemical rate controller can be used to provide a similar behavior as FPS [200]. However, it is not confined to specific settings and applications and is agnostic to intra-protocol features. It shares also some concepts of building blocks of AJIL [2] although it relies on a different control mechanism: pacing instead of window-based.

In general, Chemistry-inspired controllers for packet-switched networks and, in particular, the distributed chemical rate controller have some ingredients of the recent HULL proposal [9]: They involve queue-length proportional pacing (=LoMA-scheduling), they resort to phantom/virtual queues (=chemical species), and they enforce a rate cap by means of a token-bucket-like scheme. However, what we propose here is a general purpose framework that targets a wide application- and problem-spectrum, not a customized solution for a specific problem. This brings to three key differences: (*i*) Only one simple law defines the relationship between fill-level and rate. By applying LoMA-scheduling, we enforce a linear proportionality among queue fill-level and service rate. We thus have a single parameter, i.e. the reaction coefficient defining the speed of the reaction, rather than having many parameters which define the shape of a non-linear function that relates delays and queue fill-level. (*ii*) We broaden the way how (real and virtual) queues can interact. Although we adopt a very simple law that gathers predictability and stability (LoMA), we enable complex interactions among virtual queues (chemical species) by means of a reaction network, while maintaining predictability and stability of the overall emergent control system. (*iii*) We apply generic principles (mass-action and mass-conservation principles) and use natural schemes (Enzymatic-based systems) to let rate caps *emerge* on traffic.

Other nature-inspired approaches for flow-control have been proposed in the literature: e.g., the Lotka-Volterra competition model has been used in [107] to update the congestion window of TCP and in [18] as a hop-by-hop rate control mechanism for wireless sensor networks. By contrast to them, we do not borrow a specific model and shape the traffic flow by numerically approximating the equations (ODEs) that describe such a model. Rather, we adopt the chemical metaphor and the whole underlying theory (i.e., dynamics and kinetics of reaction networks), which allow us to design, implement, analyze, and execute computational systems.

What we achieve in the end is to combine analysis *and* execution model in one single consistent framework for algorithm design. We design networking

algorithms and analyze their dynamics at the macroscopic flow level, by using continuous-time continuous-value signals. In some respects, our work is conceptually close to what proposed in [130]: we avoid to use queueing network models (cumbersome or impossible) and rely on a deterministic control-theoretic description of the average of an algorithm's trajectory. We use a state-space description of the algorithm and make observations on the algorithm's stability against small perturbations around fixed stable points. Other recent works have used such an approach for TCP-like flow-control schemes (e.g., [260]) and AQM controllers (e.g., [53, 110, 111, 256]). However, these works apply control theory to algorithms that were written and executed in another description/programming language. Our approach goes a step further: there is no need for the derivation of a fluid model that is able to approximate sufficiently well the empirical observed dynamics. Instead, because the execution model (state machine, automaton) *is* the state-space expression of the fluid model, a designer can start with a desired dynamic behavior (fluid model) and end up with the algorithm that computes this model.

Analytical *and* execution model: one and the same thing

The Chemistry-inspired design approach of traffic controllers is very close to the strategy suggested by Kreyssig and Dittrich in [135] to control complex systems. In *emergent control*, the behavior of the controlled system *emerges* at a global scale (macro-level) from a set of (usually simple) local rules that are executed by the different components of the whole system. The control function is not based on global features/parameters of the system and enforced at a single module. Rather, it emerges as asymptotically stable *equilibrium* state of the whole system [135, 231]. In the proposed distributed rate controller, there is no explicit feedback at the macro level (key feature of traditional feedback control). There are instead a micro-level feedback and micro-rules that, thanks to the underlying mass-action kinetics, are mapped to a controlled macro-behavior of the system.

"Emergent controllers"

Emergent control appears already in today's packet-switched networks. For example, we can see TCP's congestion-avoidance algorithm as a manifestation of emergent control – in a congested network where multiple packet flows coexist, efficiency and fairness *emerge* at the macroscopic (flow) level as a consequence of ACK-based decisions at the microscopic (state-machine) level. The evolution history of TCP's algorithms includes all approaches cited in [135] to construct a "macro-micro feed-forward controller" to compute the microscopic parameters from the macroscopic requirements: (*i*) first deducing local rules manually [85] and then proving mathematically the macro behavior [242], (*ii*) first designing the model describing the desired behavior [130] and then extrapolating micro rules that approximate well the model [153], (*iii*) extrapolating the micro-macro relationship by performing experiments [121, 125], (*iv*) mimicking solutions in Nature [107], or (*v*) through evolution and optimization [246].[3.2] We propose what we think is a more *systematic* and *complete* methodology to design, execute, and analyze macro-micro feed-forward control (emergent

[3.2] We have cited a few works just to have examples of mentioned points.

control) algorithms, which incorporates the aforementioned methodologies in one single abstraction.

3.2 Flow interactions in queueing systems

The Internet is a global system of interconnected facilities, to which users access simultaneously ("access-sharing"). Access sharing implies the statistical multiplexing of flows. In the following, we show how the Chemistry-inspired approach (LoMA-scheduling) can assist us in the design and study of interacting queues and thereby, in the design of networking systems.

3.2.1 Law of Mass Action (LoMA) and classic work-conserving packet scheduling

We have already introduced the basics of $\mathcal{AC}$ and explained how it can be used in communications and computer networks. We have discussed principles, laws, and theories by means of the unimolecular reaction and, right from the start, we have established the connection to a traditional packet queue. We now examine in more depth this connection and identify differences between LoMA and traditional scheduling.

Arrival rate, occupancy, and delay...

...the 3 specs of queueing systems

Queueing systems can be studied in more traditional terms by means of queueing theory, which gives some basic understanding of queue-interaction phenomena. Generally, a queuing system can be defined as a network of queues where packets arrive, wait in various queues, receive service at various time-points, and finally exit after some time. In general, we can identify three components for each queue: (*i*) the arrival rate λ, i.e., the long-term number of arrivals per unit time, (*ii*) the occupancy L, i.e., the number of packets in the system (averaged over a long time), (*iii*) the delay d, i.e., the time from packet entry to exit (averaged over many packets).

In queueing theory a model is constructed so that queue lengths (occupancies) and waiting times (delays) can be predicted. The simplest (and most tractable) of queueing models is the M/M/1 model.[3.3] According to the M/M/1 model, the arrival process is a Poisson process with rate λ pkt/s, a reasonable assumption when the accumulated traffic consists of many independent sources. With this assumption, we can take advantage of the memoryless property of exponentially distributed inter-arrival times, and ignore when the last packet arrived and when the last packet was dequeued; we gain a very simple description of the system state, i.e., the amount of packets awaiting in the queue. Perhaps more

[3.3] We use Kendall's notation [128], which is a standard approach used to describe and classify a queueing node. The first term indicates the time between arrivals to the queue (in this case "M" = Markovian, i.e., Poisson arrival process or namely, exponentially distributed inter-arrival times), the second term gives the distribution of service time (in this case "M" = Markovian, i.e., exponentially distributed service time). The last term denotes the number of servers at the node (in this case "1" = one single server).

debatably, the M/M/1 model assumes also that packet transmission (service) times are exponentially distributed with mean $1/\mu$, and therefore assumes their independency. The M/M/1 model further considers a single server which serves packets one at a time from the front of the queue, according to FIFO (First-In-First-Out) discipline. Finally, the queue is considered to have an infinite size such as to exclude the drop of packets.

Already this simple queueing model (one of the simplest, interesting models) leads to a difficult expression of the time-dependent behavior of its state probabilities. The model can be described as a continuous time Markov chain, and its transient behavior can be described as a time-dependent probability mass function, i.e., the probability that the M/M/1 queue is in a particular state at a given time, knowing the queue being initially in a certain state. The solution includes an infinite sum of modified Bessel functions.

Reducing further the accuracy of the model in favor of tractability, we assume sufficiently large timescales such that we can study directly the stationary (steady-state) behavior of the system (much easier to analyze than transients). Assuming that arrivals occur slower than service completions in the queue ($\lambda < \mu$), or in other words if the utilization factor $\rho = \lambda/\mu$ is less than one, then the amount of awaiting packets does not grow without limit, i.e. the system is stable. Applying Markov analysis, in such a condition, the average number of packets in the system turns out to be $L = \rho/(1-\rho)$ and the variance of the number of customers in the system is $\rho/(1-\rho)^2$.[3.4] The average response time (total time a customer spends in the system) does not depend on the scheduling discipline and can be computed using Little's law as $1/(\mu-\lambda)$. The average time spent awaiting in the queue turns out to be $1/(\mu-\lambda)-1/\mu = \rho/(\mu-\lambda)$ and the distribution of response times to be independent of scheduling discipline.

Queueing-theory...

...a *practicable* steady-state study of *tractable* models

By contrast, when applying the LoMA-scheduling, we do not need to model (make simplifying assumptions on) arrival and departure processes in order to preserve the analyzability of the system. Indeed, the LoMA-scheduling imposes specific dynamics to the departure processes, and it makes the modeling and the implementation of the scheduling policy one and the same thing. For example, to get closer to traditional M/M/1 modeling, we can choose to use exponentially distributed inter-reaction times. This choice represents a mere scheduling policy and *not* an assumption to model the departures as a Poisson process. Furthermore, LoMA-scheduling decouples the micro-patterns characterizing arrivals from that of departures (only the average trajectory of the departure process is coupled with the average arrival rate).[3.5] That is, the results concerning the average departure process are valid for any distribution of the arrival process (e.g., Poisson, Gaussian, *etc.*).

Exponential reaction times...

...a design choice, not an assumption to make the analysis tractable!

The direct coupling of fill-levels and rates, dictated by LoMA, leads to an

[3.4] This result holds for any work conserving service regime [102].

[3.5] As we have discussed in the previous chapter, the programmable filtering behavior of LoMA-served queues enables controlling the level with which micro patterns in the arrival process affect the micro-dynamics of the departure process.

Property of Queue	M/M/1	LoMA
Server action	work-conserving	non-work-conserving
Service rate	μ (model)	λ (actual)
Expected fill-level	$L = \lambda/(\mu - \lambda)$	$c_S = \lambda/k$
Expected waiting time d	$1/(\mu - \lambda)$	$1/k$

Table 3.1: *Comparison between M/M/1-modeled queue and LoMA-served queue. Results are valid at steady state only and refer to mean values.*

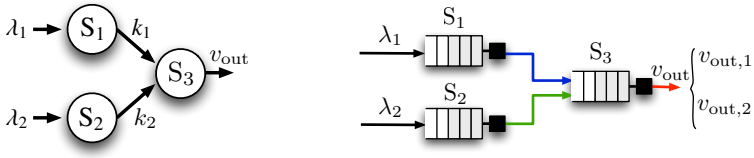
LoMA binds analytical and executable models

accurate mathematical representation in terms of ODEs, which enables the study of both steady state and transient behavior in a methodical way. By summarizing the results of the analysis of a LoMA-served queue in Section 2.3 (see examples on unimolecular reaction system), we provide in Table 3.1 a side-by-side list of main *stationary* properties of the M/M/1-queue and the LoMA-served queue. An interesting aspect of LoMA-scheduling is that, by adding a controlled artificial delay to each packet transmission (higher delay when the fill-level is lower), it makes the expected waiting time constant, independently of arrival and departure processes, and inversely proportional to the design coefficient k. Another interesting feature is the more direct, and generally valid, relationship among the three main components λ, d, and L, which enables an easy dimensioning of network resources (i.e., buffers and CPU performances) for a given target waiting time d.[3.6]

The LoMA-policy enhances the controllability and the predictability of interacting queues. In a traditional work-conserving approach, packet interference often leads to unpredictability of dynamics, burstiness of rates, and thus accumulation of delays, as soon as packet arrivals are not regular or sufficiently spaced apart. To show this, we consider a system of three interacting queues (see Figure 3.1) and evaluate, through simulations (see Figure 3.2), the emerging traffic dynamics when applying (a) work-conserving and (b) LoMA scheduling policies. The Poisson-generated traffics traverse the first two queues S_1 and S_2 and merge at the third queue S_3, where packets interact with irregular patterns. The chemical approach enforces micro-delays for each packet (directly related to the mass action principle), which guarantees regularly paced arrivals, and thus enables us to attain controllability of the system and to describe its average trajectory with ODEs.

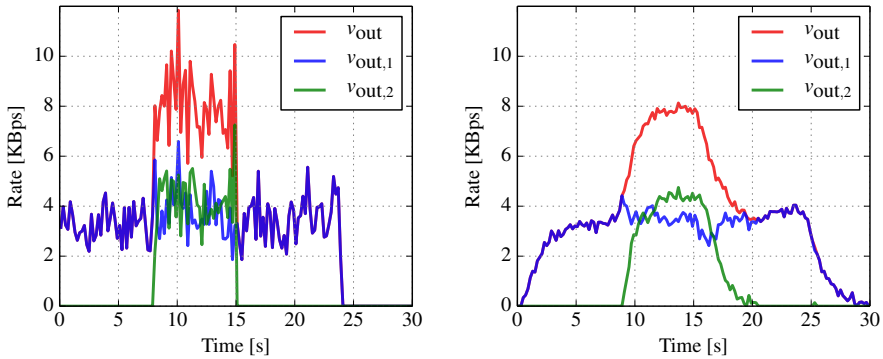
We can easily generalize the analysis and extract main properties of more complex queuing networks by studying them as chains and multiplexing systems made of simple LoMA-served queues (refer to Figure 3.3 for schemes of these systems). In Table 3.2, we report the stationary average features of a chain and a multiplexing of simple LoMA-served queues (see [168] for details about their steady state and transient analysis). For the chain of queues (the scheme

[3.6] We evaluate the expected waiting time d by applying Little's law, which is valid also for LoMA-served queues [163].



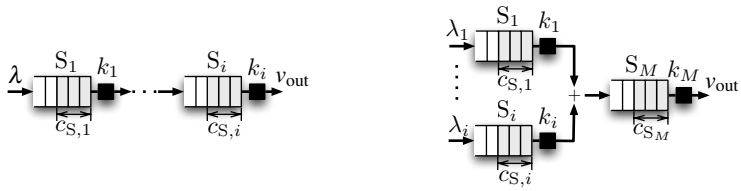
(a) Chemical model (b) Equivalent queuing network

Figure 3.1: *Network of three interconnecting queues. Traffics through the first two queues S_1 and S_2 merge, in this way letting packets interact at the third queue S_3.*



(a) Work-conserving scheduling (b) LoMA-scheduling

Figure 3.2: *Simulation results when the Poisson-generated traffics enqueued at queues S_1 and S_2 had average rate $\lambda_1 = 3770$ Bps and $\lambda_2 = 4200$ Bps (refer to network in Figure 3.1(b)).*



(a) Chain (b) Multiplexing

Figure 3.3: *Generalized interacting LoMA-served queues. (a) Queues on the same path. (b) Load balancing traffic flows over many different links.*

Property of Interacting Queues	Chain	Multiplexing
Service rate	λ	$\sum_i \lambda_i$
Expected fill-level	$\sum_i \lambda/k_i$	$\lambda/k_i + \lambda/k_M \quad \forall\, \mathrm{b}_i$
Expected waiting time	$\sum_i d_i$	$d_i = 1/k_i + 1/k_M \quad \forall\, \mathrm{b}_i$

Table 3.2: *Stationary average features of interacting LoMA-served queues, chain and multiplexing. Refer to Figure 3.3(a) and Figure 3.3(b). (b_i stems for branch i.)*

(a) Chemical reaction model (b) Queuing network

Figure 3.4: *A bimolecular reaction extracts an instance (=packet) from two chemical species (=packet queues), X and Y. At the same time, queue Y is drained by a second server (=reaction). The forwarding rate $v_{out,1}$ is controlled by the abundance c_Y of available forwarding tokens Y. Pictures taken from the original figure in [165].*

in Figure 3.3(a)), we report fill-level and delay for the whole system, made of all queues, where the contribution of each queue S_i is identifiable by the term with subscript i. For the merging of many flows λ_i into a single queue S_M (the "multiplexing" system in Figure 3.3(b)), we report fill-level and delay related to a specific branch b_i, where again contributions of the "branch-queue" S_i and "multiplexing-queue" S_M are identifiable.

From the design point of view, the Chemistry-inspired approach gives the possibility to *design by queues' interactions*. To illustrate this, take the example presented in [165] and shown here in Figure 3.4 (closely related to what already introduced in Section 2.2.2). The related chemical model, shown in Figure 3.4(a), consists of two chemical reactions r_1 and r_2 having two reactants X and Y and two products Z and W. Reaction r_1 is a bimolecular reaction $X + Y \xrightarrow{k_1} Z$ that needs a molecule X and a molecule Y to occur. The competing reaction r_2 simply turns a Y-molecule into a W-molecule, based on the amount of molecules Y only. Again, by analogy between Chemistry and networking, we have two packet queues X and Y served by two servers, see Figure 3.4(b). The upper server is special in that it extracts a packet from two queues at the same time. We can further refer to queue X as the place where actual data-packets with payload are stored, whereas to queue Y as a virtual ("phantom") queue that contains mere control-tokens. This concept, and the related mechanism, are extensively studied in the next sections to design traffic shaping algorithms. Here, we limit to remark that the service process of both queues is non-work-conserving (application of the mass action principle); the service of queue X is further delayed as soon as Y-tokens vanish. Namely, we can control the forwarding rate $v_{out,1}$ with the abundance c_Y of available forwarding tokens Y.

Design by queues' interactions

3.3 Traffic rate control

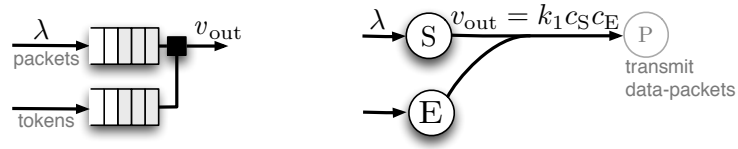
To setup and maintain a shared access facility, such as the Internet, one needs to put in place functions for regulating the relative resource utilization by the admitted flows and for guaranteeing admissibility of all flows that demand access. *Rate-control* is a prominent mechanism for regulating capacity allocations and resource reservation as well as for congestion control and avoidance, DoS mitigation, service differentiation, and traffic shaping. Thus, embedded rate control functions serve as a common means to different ends.
In the following sections, we show how to develop a simple Chemistry-inspired rate controller. We start in Section 3.3.1 by first constructing the well-known token-bucket mechanism with $\mathcal{AC}$-tools. Then in Section 3.3.2, we discuss a Chemistry-inspired scheme to rate control, which exhibits enhanced features. We provide a complete analysis of both steady state and transient behaviors of the enhanced chemical controller in Section 3.3.3. Finally in Section 3.3.4, we report on experiment results on shaping Internet traffic via this chemical rate controller.

3.3.1 Token-Bucket

A *Token Bucket* (TB) or *leaky bucket* scheme (Figure 3.5(a)) has been one of the most common building blocks in the design of more or less sophisticated (and protocol-independent) rate policers and controllers for various purposes, both in reservation-based and reservation-less networks.[3.7]
In the simplest case (rate limiter), a TB scheme is used to regulate the departure process of packets from a queue; tokens essentially authorize the consumption of a respective number of packets or bytes. The bucket is filled with tokens at a constant rate, up to a maximum limit (bucket size) above which excess tokens are tail-dropped. In this arrangement, at any moment in time, the output of packets from the queue cannot exceed a burst size equal to the size of the bucket, and cannot sustain a constant rate higher than the refill rate of tokens. In a more dynamic arrangement (rate controller), the rate limit imposed by the TB scheme may vary in time. This can be achieved by adapting the bucket size and the refill rate of tokens in response to feedback from the network, or in response to the differentiation between departure processes of two or more regulated queues. To analyze the behavior of such a system, without loss of generality one may regard the token bucket as an additional virtual (phantom) queue. The most interesting aspect of such an arrangement is that the departure process depends on the state of both queues: The process blocks if either the packet queue or the token queue is empty.[3.8]

[3.7] The leaky bucket scheme we refer to is the one proposed in [234], or similarly the generic cell rate algorithm in [120], which are both directly comparable to the TB scheme and differ from it in mere implementation details.

[3.8] One can regard the token bucket as yet another queue whose departure process is on



(a) Classical token-bucket (b) Corresponding chemical model

Figure 3.5: *Token-Bucket scheme and simple equivalent chemical reaction network.*

The TB-scheme is very simple, at least from a functional perspective. On the other hand, the introduction of control loops and automations (e.g., non-linearities) complicates substantially the analyzability of the system when assessing its (in-)stability conditions.

An intuitive chemical pattern that captures exactly the typical transient behavior of the classic TB scheme is shown in Figure 3.5(b). This chemical scheme comprises two species S and E, which represent respectively packets with data and tokens, and the bimolecular reaction $\mathrm{S} + \mathrm{E} \xrightarrow{k_1} \mathrm{ES}$. An S-molecule reacts with an E-molecule to produce an ES-molecule. If one of the two reactant species is depleted, the reaction becomes *inert*. This condition essentially captures the dependency between token and packet availability for scheduling transmissions out of a token-bucket regulated queue.

The reaction between the reactant species takes place at an average rate v. If we choose the inter-reaction times to be drawn from an exponential distribution, then this representation models the interaction of a $\cdot/M/1$ queue with the token bucket (subject to token availability). Practically in this model, the reaction merely defines a rule (dependency) that relates the departure process of two (or more) queues. Also, the rate of the reaction describes the probability distribution of this rule being successfully applied.

Traffic bursts with Token Bucket (TB)...

The intuitive chemical pattern captures exactly the typical transient behavior of the classic TB scheme. Both mechanisms suffer from overshooting-problems in the transient phase. After a low-load period, in which tokens accumulate in the phantom queue, bursts can pass through unaltered, temporarily violating the predefined cap on the output rate v_{out}.[3.9] Frequent bursts imply

average dictated by the departure process of one or more other queues. The special difference of the token bucket from a "normal" queue is that its blocking condition is met when it is empty and not when it is full. This is because (unlike a "typical" queue) its instantaneous behavior is dominated by its departure process and not by its arrival process, which has much slower dynamic variations.

[3.9] There exists a variant of the leaky bucket scheme (refer to [225]) that enforces a fixed value on output rate (as soon the bucket is guaranteed to be not completely empty), guaranteeing in this way steady output rates. It consists in simply queueing packets for enough queue space, and sending head packets at a constant predefined rate, independently of the inflow pattern. Due to the latter aspect however, this scheme has efficiency problems and

unpredictable traffic, and thus difficulties in engineering and provisioning the network capacity [6, 9, 77]. Bursts may imply also a temporary congestion of the link, and at the same time the need for large buffers in the network. Indeed, without sufficiently large buffers, some of the packets that arrive in bursts are dropped, leading in this way to a poor efficiency (i.e., the utilization of the link is lower than the predefined limit), poor flow mixing, and often flow lockout. At the same time, large buffers potentially lead to higher latency and jitter, and bufferbloat [173, 178].

...harmful for network performance

3.3.2 Enzymatic reaction model: an elegant alternative

We can take advantage of an elegant motif imported from Chemistry to limit (adaptively control) the traffic rate without encountering burstiness and attaining smooth traffic dynamics. The derived limiting mechanism is thus more suitable to operate in non-steady traffic conditions, such as the common Internet traffic's conditions. The mechanism has also the benefit of being easily analyzable, tunable, and parameterizable.

We can start to design such a mechanism by considering the chemical equivalence of the TB-mechanism, previously shown Figure 3.5(b). As a first step, we have to determine the rate at which the token bucket is filled in the chemical model. If we assume that the concentration of E-molecules is the one that represents the number of tokens in the bucket, we care that this concentration is refreshed at a fixed rate until a maximum size. A simple possibility is to externally inject new E-molecules at a certain rate, and constantly monitor the concentration size such that it does not exceed the set limit (which equals the bucket length). An alternative, elegant design pattern comes directly from Chemistry to capture this situation with a "control loop" instead.[3.10] It is the Enzymatic reaction network.

The kinetics of the Enzymatic reaction model exhibits a maximum reaction rate, i.e. a rate cap.[3.11] This cap emerges progressively and is the result of applying the LoMA-scheduling in a system where the sum of available and used resources remains constant, as free-form enzymes (E) and enzyme-substrate complexes (ES). In the following, E-molecules represent available tokens and ES-molecules used tokens. The simplified Van Slyke-Cullen model of the Enzymatic reaction network, proposed in [238], is shown in Figure 3.6(a)

leads to the full resource utilization also for low-load periods. In the next section, it will be clear why such a simple scheme cannot be utilized for designing an adaptive rate controller.

[3.10] This pattern has also appeared in [163] as "rate limitation motif" for CNPs.

[3.11] Kinetics of Enzymatic reaction model was discovered by Michaelis and Menten in 1913 and published in [167] (see the English translation [226]) and formalized by Briggs and Haldane in [39], 1925.

(a) Enzymatic reaction model (b) Saturation curve

Figure 3.6: *The Enzymatic reaction model can be seen as a token bucket scheme with "feedback-control"; the maximum dequeueing rate v_{max} is defined by the number of enzymes $e_0 = c_E + c_{ES}$, the aggressiveness by the ratio k_2/k_1.*

and consists of the following set of two reactions:[3.12]

$$r_1 : \mathrm{S} + \mathrm{E} \xrightarrow{k_1} \mathrm{ES} \tag{3.1a}$$

$$r_2 : \mathrm{ES} \xrightarrow{k_2} \mathrm{E} + \mathrm{P}. \tag{3.1b}$$

Emergent rate control from...

...mass-action

...mass-conservation

...flow-conservation

The limiting mechanism is based upon the simple *mass-conservation principle*: "The total sum of molecule concentrations along a loop remains constant, if (*i*) the total number of molecules consumed by reactions along the loop is equal to the total number of molecules produced, and if (*ii*) all concentrations along the loop are altered only by reactions involved in this or another loop".
We can describe the Enzymatic system directly by considering, in combination, (*i*) the mass action principle (LoMA), (*ii*) the mass-conservation principle and (*iii*) the fact that, at equilibrium, the total production rate equals the total consumption rate, for each molecular concentration (*flow conservation principle*). Indeed, based on these principles, we can derive the equation that relates the output rate to the control parameters (k coefficients and total number of enzymes/tokens) and to the fill-level of the input queue (c_S concentration):

$$v_{out} = k_2 e_0 \frac{c_S}{c_S + k_2/k_1}. \tag{3.2}$$

$k_2 e_0$ to set the rate cap

k_2/k_1 to define the aggressiveness

According to the mass-conservation law, the total number of enzymes e_0 remains constant over time i.e., enzymes are present in either free form or in substrate complex but never consumed or produced: $c_E + c_{ES} = const. = e_0$. The product $k_2 e_0$ defines the upper rate limit v_{max} that can be achieved, whereas the ratio of coefficients k_2/k_1 specifies how aggressive the rate regulation is. Figure 3.6(b) outlines what formally expressed in (3.2), namely it shows the hyperbolic saturation curve of the output rate over the quantity of S molecules. We observe two limit behaviors: for a low number of packets ($c_S \to 0$), the

[3.12] The Van Slyke-Cullen model differs from the Michaelis-Menten one in the fact that it is a non-reversible mechanism (the substrate-binding reaction is non-reversible).

Enzymatic reaction behaves like the unimolecular reaction; for a high number of enqueued packets ($c_S \to \infty$), the transmission rate asymptotically converges to the maximum rate v_{max}.
Note that the simple Enzymatic motif in Figure 3.6(a) limits the transmission rate based on the fact that the number of tokens in the loop are limited and that each reaction r_i imposes to each packet a small delay $d_i = 1/k_i$, which is inversely proportional to its coefficient k_i. This motif is similar to a token bucket algorithm, but differs in the fact that the tokens are re-used and rotate in a loop. Such a token loop is only feasible thanks to the LoMA-scheduling algorithm that we use to drive the virtual queues. A work-conserving discipline would cause the tokens to loop infinitely fast, and the reaction network would not limit the traffic.
The Enzymatic model can be used as a traffic controller that, in addition to guarantee that the sending rate always respects a certain predefined threshold, decouples the generation (or arrival) rate of packets from their departure (dequeueing/sending) rate, while preserving the efficiency of the system. That is, it filters out fast variations and oscillations that characterize the arrival process and maintains the average arrival-to-departure-rate correspondence, for low-load scenarios.

3.3.3 Analysis of an Enzymatic traffic controller

By applying step by step the analysis methodology we introduced in Section 2.3, we formalize the discussed features of the Enzymatic traffic controller. We report in this section the final description of its steady and transient behavior; for details on analytical steps, refer to Appendix A.3.

3.3.3.1 Enzymatic traffic controller: Fluid model

We can describe the dynamics of the Enzymatic model in Figure 3.6(a) (described by reactions (3.1a) and (3.1b)) by the following set of coupled ODEs:[3.13]

Enzymatic controller: fluid model

$$\dot{c_S} = \lambda - k_1 c_S c_E \tag{3.3a}$$
$$\dot{c_E} = k_2 c_{ES} - k_1 c_S c_E \tag{3.3b}$$
$$\dot{c_{ES}} = k_1 c_S c_E - k_2 c_{ES}. \tag{3.3c}$$

As we know, this ODE set directly stems from the mass-action principle.
From (3.3), we observe that tokens (enzymes in free form E) are consumed with a rate that is proportional to the amount of data packets (c_S) times the amount of available tokens (c_E) (k_1 as a proportionality coefficient). However, data packets are not sent at this rate as the sending process is not related to substrate-binding reaction r_1 in (3.1a). Rather, the actual dequeueing/sending

[3.13] ODEs are coupled because they describe a non-linear system made of non-unimolecular reactions.

process has a rate that is proportional to the amount of ES-molecules (k_2 as a proportionality coefficient) as this process is bound to the execution of the catalytic reaction r_2 in (3.1b), which controls the dequeueing from the additional phantom queue ES and the regeneration of available tokens (free-form enzymes E). As we prove in the following, this is the essential process whereby the Enzymatic traffic controller is able to decouple the arrival rate of packets from their dequeueing/sending rate.

3.3.3.2 Enzymatic traffic controller: Steady state

To study the system at steady-state, we solve (3.3) with respect to species concentrations, when the left-hand side is equal to 0:

Enzymatic controller: steady state

$$c_{\mathrm{S}}{}^{*} = \frac{\lambda}{k_2 e_0 - \lambda}\frac{k_2}{k_1} \tag{3.4a}$$

$$c_{\mathrm{E}}{}^{*} = \frac{k_2 e_0 - \lambda}{k_2} \tag{3.4b}$$

$$c_{\mathrm{ES}}{}^{*} = \frac{\lambda}{k_2} \tag{3.4c}$$

where λ is the average steady-state arrival rate. However, we have to consider the mass-conservation principle in the moiety E-ES, i.e. a closed loop where the sum of the concentrations of involved species is constant over time. By considering the equality $c_{\mathrm{ES}} + c_{\mathrm{E}} = const. = e_0$ and further realizing that molecular concentrations (amount of either packets or tokens) can only be positive quantities, we observe that the validity of (3.4) is limited to the following region:

Rate-limiting for $\lambda > k_2 e_0$

$$\lambda < k_2 e_0. \tag{3.5}$$

The obtained result suggests that, if the generation rate at the data source is sufficiently low, i.e., in an unsaturated regime according to (3.5), the source does not experience any rate-limitation and can transmit with an average rate that equals the average arrival rate: $v_{\mathrm{out}} = k_2 c_{\mathrm{ES}}{}^{*} = \lambda$. On the contrary, when the substrate-binding reaction r_1 in (3.1a) does not drain S-species at a high-enough rate, i.e. (3.5) is not respected (saturated regime), the amount of S molecules increases without limit and the mass of enzymes (tokens) conserved in the moiety lies mainly in ES-species, i.e. $c_{\mathrm{ES}} = e_0, c_{\mathrm{E}} = 0$. The absence of molecules (tokens) E and its automated refill process (parametrized by k_2) control the transmission rate in the saturated regime.
Table 3.3 summarizes the characteristic stationary quantities of the Enzymatic traffic controller. We calculate the expected delay for low-load periods by applying Little's law to the two phantom queues S and ES that packets virtually have to pass through, namely as the sum $c_{\mathrm{S}}{}^{*}/\lambda + c_{\mathrm{ES}}{}^{*}/\lambda$.[3.14] We can further characterize interacting flows out of Enzymatic traffic controllers. Specifically

[3.14] At steady state, for $\lambda < k_2 e_0$, inflow and outflow rates must be equal, according to the flow balance principle. Thus, inflow and outflow of both species are equal to λ on average.

Property	Low Load $\lambda < v_{\max}$	High Load $\lambda > v_{\max}$
Arrival rate, λ	λ	λ
Expected # packets, c_{S}^*	$\frac{k_2}{k_1}\frac{\lambda}{v_{\max}-\lambda}$	max. q. size (theory ∞)
Constant # tokens, e_0	$c_{\mathrm{E}}^* + c_{\mathrm{ES}}^* = const.$	$c_{\mathrm{E}}^* + c_{\mathrm{ES}}^* = const.$
Expected # free tokens, c_{E}^*	$\frac{v_{\max}-\lambda}{k_2}$	0
Expected # busy tokens, c_{ES}^*	$\frac{\lambda}{k_2}$	e_0
Expected sending rate, v_{out}	λ	$v_{\max} = k_2 e_0$
Expected waiting time, d	$\frac{k_2}{k_1}\frac{1}{v_{\max}-\lambda} + \frac{1}{k_2}$	–

Table 3.3: *Stationary average features of the Enzymatic traffic controller. Refer to Figure 3.6(a) for the respective scheme.*

in Table 3.4, we give results for a chain of Enzymatic controllers, i.e., the original flow of packets characterized by average arrival rate λ pass through a cascade of Enzymatic traffic controllers (see Figure 3.7), each defined by parameters $k_{1,i}$, $k_{2,i}$, $e_{0,i}$. Finally, in Table 3.5, we show how these characteristic quantities would vary when Enzymatic controlled flows (each originated from an arrival process with average rate λ_i and regulated according to parameters $k_{1,i}$, $k_{2,i}$, $e_{0,i}$) merge at a downstream Enzymatic-controlled queue. For example, we could refer to such a scenario (illustrated in Figure 3.8) when trying to load balance traffic flows over many different links.

3.3.3.3 Enzymatic traffic controller: Sensitivity and stability

We have already established that, as soon as the average arrival rate exceeds the predefined limit $e_0 k_2$, the dequeueing/sending rate becomes insensitive to variation of the arrival rate, i.e. capped at $e_0 k_2$ value. But how are arrival and sending processes related in "low-load" periods (i.e., before saturation)? By carrying out the sensitivity analysis, we can characterize the effect of variations of arrival rate around its average value λ on the dequeueing/sending rate v_{out}, when condition (3.5) is respected. We achieve the formal description of the system response around its steady state (3.4) in the Laplace domain, in terms of the following transfer function $H(s)$:

$$H(s)|_{(\lambda<k_2 e_0)} = \frac{v_{\mathrm{out}}(s)}{\lambda(s)} = \frac{k_1 k_2 {c_{\mathrm{E}}}^*}{s^2 + s\left(k_2 + k1({c_{\mathrm{S}}}^* + {c_{\mathrm{E}}}^*)\right) + k_1 k_2 {c_{\mathrm{E}}}^*} = \\ = \frac{k_1(k_2 e_0 - \lambda)}{s^2 + s\left(k_2 + k_1\left(\frac{k_2 e_0 - \lambda}{k_2} + \frac{k_2 \lambda}{k1(k_2 e_0 - \lambda)}\right)\right) + k_1(k_2 e_0 - \lambda)}. \tag{3.6}$$

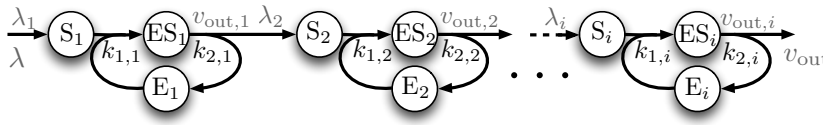


Figure 3.7: *Chain of many Enzymatic-controlled flows.*

Property	Low Load $\lambda < \min_i(v_{\max,i})$	High Load $\lambda > \min_i(v_{\max,i})$
Arrival rate, λ	λ	λ
Local arrival rate, λ_i	λ	$\alpha = \min(\lambda, v_{\max,0}, .., v_{\max,i-1})$
Expected # packets, $c^*_{S,i}$	$\frac{k_{2,i}}{k_{1,i}} \frac{\lambda}{v_{\max,i}-\lambda}$	max.q.size (theory ∞) (note1) $\frac{k_{2,i}}{k_{1,i}} \frac{\alpha}{v_{\max,i}-\alpha}$ (note2)
Expected # free tokens, $c^*_{E,i}$	$\frac{v_{\max,i}-\lambda}{k_{2,i}}$	0 (note1) $\frac{v_{\max,i}-\alpha}{k_{2,i}}$ (note2)
Expected # busy tokens, $c^*_{ES,i}$	$\frac{\lambda}{k_{2,i}}$	$e_{0,i}$ (note1) $\frac{\alpha}{k_{2,i}}$ (note2)
Expected sending rate, v_{out}	λ	$\min_i(v_{\max,i})$
Expected local sending rate, $v_{out,i}$	λ	$\min(\lambda_i, v_{\max,i})$
Expected waiting time, d_i	$\frac{k_{2,i}}{k_{1,i}} \frac{1}{v_{\max,i}-\lambda} + \frac{1}{k_{2,i}}$	– (note1) $\frac{k_{2,i}}{k_{1,i}} \frac{1}{v_{\max,i}-\lambda} + \frac{1}{k_{2,i}}$ (note2)
Expected delay, d	$\sum_i d_i$	–

Table 3.4: *Stationary average features for a chain of Enzymatic traffic controllers (Figure 3.7). Values marked with (note1) refer to the congested module j, i.e. where $\lambda_j > v_{max,j}$. Values marked with (note2) refer to the non-congested module j, i.e. where $\lambda_j < v_{max,j}$.*

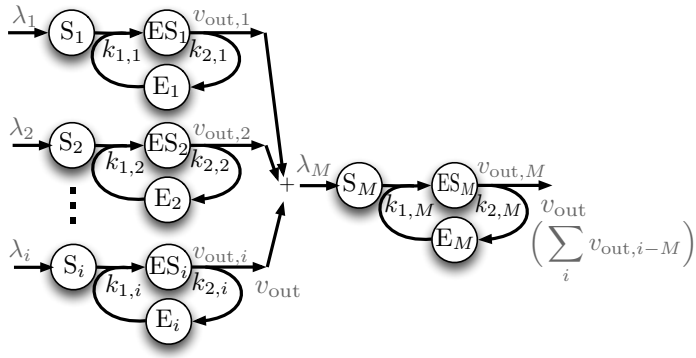


Figure 3.8: *Load balance of many Enzymatic-controlled flows.*

Property	Low Load $\lambda_i =< \min(v_{\max,i}, v_{\max,M})$	High Load $\lambda_i > \min(v_{\max,i}, v_{\max,M})$
Arrival rate per branch, λ_i	λ_i	λ_i
Expected # packets, $c^*_{S,j}$	$\frac{k_{2,j}}{k_{1,j}} \frac{\lambda}{v_{\max,j}-\lambda}$	max.q.size (theory ∞) (note1) $\frac{k_{2,j}}{k_{1,j}} \frac{\lambda_j}{v_{\max,j}-\lambda_j}$ (note2)
Expected # free tokens, $c^*_{E,j}$	$\frac{v_{\max,j}-\lambda}{k_{2,j}}$	0 (note1) $\frac{v_{\max,j}-\lambda_j}{k_{2,j}}$ (note2)
Expected # busy tokens, $c^*_{ES,j}$	$\frac{\lambda}{k_{2,j}}$	$e_{0,j}$ (note1) $\frac{\lambda_j}{k_{2,j}}$ (note2)
Expected sending rate per branch, $v_{out,i-M}$	λ_i	$\min(\lambda_i, v_{\max,i}, v_{\max,M})$
Expected sending rate, v_{out}	$\sum_i \lambda_i$	$\min(\sum_i v_{out,i-M}, v_{\max,M})$
Expected waiting time, d_j	$\frac{k_{2,j}}{k_{1,j}} \frac{1}{v_{\max,j}-\lambda} + \frac{1}{k_{2,j}}$	– (note1) $\frac{k_{2,j}}{k_{1,j}} \frac{1}{v_{\max,j}-\lambda} + \frac{1}{k_{2,j}}$ (note2)
Exp. delay per branch, d_{i-M}	$d_i + d_M$	–

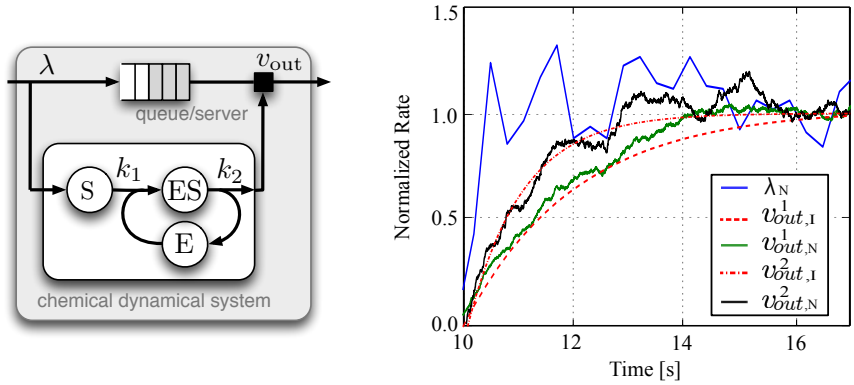
Table 3.5: *Stationary average features when merging Enzymatic-controlled flows (Figure 3.8). Values with subscript j refer to a generic module j in the system; values with subscript i refer to the module i at the beginning of a branch (path). Values marked with (note1) refer to the congested module j, i.e. where $\lambda_j > v_{max,j}$. Values marked with (note2) refer to the non-congested module j, i.e. where $\lambda_j < v_{max,j}$.*

In this seemingly complex result, we can observe several interesting components (valid when the generation rate respects the predefined threshold $e_0 k_2$): (*i*) The rate-controller dampens all sharp variations on the arrival rate in a way that reduces their effect on the sending rate. This is represented by the second-order low-pass filtering-behavior expressed in (3.6). (*ii*) The cut-off frequency of the low pass filter is adjustable through e_0, k_1, and k_2 parameters. (*iii*) The module lets the sending rate match the average arrival rate. This is represented by the fact that the low-pass filter has a unity Continuous Wave (CW) gain. (*iv*) The exact shape of the TF depends on the arrival rate itself, as we observe by the λ-dependency in (3.6). This is due to the non-linearity of the reaction network. (*v*) The adaptation of the system to bounded variations of the arrival rate always produces a limited variation on the transmission rate. Namely, the system can be defined Bounded-Input Bounded-Output (BIBO) stable as the TF always exhibits negative real-part poles.[3.15]

Having the TF, we can characterize the system's behavior in the time domain and study (*i*) how closely the sending rate follows a sudden increase of the arrival rate and (*ii*) the effect of an enqueued packet on the dequeueing/sending rate. In other words, we can quantify the stability (convergence) and sensitivity of the system by means of step and impulse responses, respectively. Through this process, we can reliably select sets of configuration parameters that match our needs.

To validate the presented analytical results, we compare the analytically predicted response of the Enzymatic rate controller and the measurements obtained in simulations (we have implemented an Enzymatic controlled queue, refer to Figure 3.9(a), in OMNeT++ [240] and generated input packets according to a Poisson process in order to reproduce the randomness that a real queue/server system possibly experiences in its input.) In Figure 3.9(b), we plot the rate measurements obtained from two tests and the analytically predicted step responses, for two different parameter pairs (e_0, k_2). We focus on the low-pass filtering effect of the Enzymatic controller when (3.5) is respected. Specifically, the graph illustrates the normalized average over 0.5s-period of the arrival rate (λ_N), the normalized instantaneous sending rate ($v^i_{\text{out},N}$), and the theoretical predicted sending rate ($v^i_{\text{out},I}$) calculated as a step response to the TF in (3.6), for two configurations; $v^1_{\text{out},I}$: $k_1 = 1\,(\text{mol}{\cdot}\text{s})^{-1}$, $k_2 = 0.5\,\text{s}^{-1}$, $e_0 = 2000$ mol and $v^2_{\text{out},I}$: $k_1 = 1\,(\text{mol}{\cdot}\text{s})^{-1}$, $k_2 = 1\,\text{s}^{-1}$, $e_0 = 1000$ mol. As the λ-curve in the graph shows, the step-perturbation on λ was a sudden increase at time $t = 10$ s of the average arrival rate from 250 pkt/s to 500 pkt/s. From this comparison we can observe that the transient analysis predicts the actual behavior of the queue/server system. Figure 3.9(b) attests the decoupling of the arrival process from the sending process. The deviations between measured and theoretical sending rate ($v_{\text{out},N}$ and $v_{\text{out},I}$ curves) are the effect of the stochasticity of the arrival process. The oscillations in the transmission rate

[3.15] Refer to Appendix A.3 for further details.



(a) Host's block diagram

(b) Rates over time

Figure 3.9: *Low-pass filtering effect of the Enzymatic traffic controller and comparison between the transient analysis results and the normalized transmission rate measured in a simulation run: The initial average arrival rate (Poisson process) was* $\lambda = 250pkt/s$ *and the applied perturbations brought it to 500 pkt/s at time* $t = 10s$*.* λ_N*-curve is the normalized 0.5s-averaged arrival rate,* $v^1_{out,N}$*-curve is the normalized instantaneous sending rate for* $k_1 = 1\ (mol{\cdot}s)^{-1}$*,* $k_2 = 0.5\ s^{-1}$*,* $e_0 = 2000\ mol$*,* $v^1_{out,I}$*-curve is the corresponding theoretically predicted sending rate,* $v^2_{out,N}$*-curve is the normalized instantaneous sending rate for* $k_1 = 1\ (mol{\cdot}s)^{-1}$*,* $k_2 = 1\ s^{-1}$*,* $e_0 = 1000\ mol$*, and* $v^2_{out,I}$*-curve is the corresponding theoretically predicted sending rate.*

(and thus also the deviations between real behavior and predicted one) can be smoothed by reducing the cutoff frequency of the implemented low pass filter.

3.3.4 Enzymatic control of traffic in real-world systems

We additionally evaluated the Enzymatic rate controller through real-world experiments by employing the controller directly in the Linux kernel and carrying out tests with TCP and UDP traffic. The objective was to test that (*i*) the Enzymatic controller effectively shapes transport-protocol traffic adhering to the requested dynamics, (*ii*) the calibration of the Enzymatic controller is sufficiently simple, and (*iii*) the Enzymatic controller adheres to the theoretically derived design rules, such that the calibration process can be automated.
We implemented a custom queueing discipline for the Linux kernel (`tc`-framework).[3.16] The chemical reaction network acted as an external manager module that controlled the service process of the egress queue for a certain class of traffic. In order to deal with variable-sized packets, the system worked on byte-basis, i.e.,

[3.16] We explicitly thank T. Meyer for providing us his original source code, which has represented the backbone of our kernel implementation.

(integer) concentrations were mere byte counters. By means of the Enzymatic controller, we shaped iperf-generated UDP and TCP traffic. The experiment involved two computers – one computer generated and sent the packets to the recipient one, which acknowledged the correct reception of a packet in case of TCP-protocol. We used default TCP-configuration as provided by iperf-tool in the (sender) Linux machine (TCP Cubic with window size 22.9 KB), and sent the traffic to a (recipient) Mac OS X 10.6.8 (TCP New Reno with window size of 256 KB). Both computers were connected via wire (Ethernet 1000baseT) to the university network as depicted in Figure 3.10. The Enzymatic controller was deployed at the egress device queue at the (sender) Linux machine.

In the first experiment (see results in Figure 3.11) we controlled UDP steady traffic. We used the configuration set $k_1 = 1\,(\text{mol}{\cdot}\text{s})^{-1}$, $k_2 = 5\,\text{s}^{-1}$, $e_0 = 500$ mol and set the bytes-per-molecule ratio to 1000 BpMol,[3.17] in order to have a predefined rate limit of $v_{\max} = 1$ MBps. We differentiated the traffic features: In a first phase ($t < 15$ s), the rate of the traffic ($\lambda = 2$ MBps) exceeded the predefined value ($v_{\max} = 1$ MBps). In a second phase ($t > 15$ s), the rate of generated traffic was sufficiently low ($\lambda = 0.5$ MBps) to avoid the limiting mechanism. The figure validates empirically (i) the burst-free, limiting effect of the Enzymatic controller in a saturated regime, when the load exceeds the predefined rate limit (similar behavior as the leaky bucket scheme introduced in [225]), and (ii) the responsiveness of the controller in non-saturated regimes (for sufficiently low loads), where the dequeueing/sending rate follows the arrival process, more or less strictly, depending on the filtering level (similar behavior as the traditional TB scheme).

In a second set of experiments (see results in Figure 3.12) we controlled bursty UDP traffic. We generated the bursty traffic by alternating 1.5 s of traffic followed by 0.5 s of silence. Again, first ($t < 15$ s), the rate of this traffic ($\lambda = 2$ MBps) exceeded the predefined value ($v_{\max} = 1$ MBps) and consequently, in this period the Enzymatic controller served to cap the rate. Then ($t > 15$ s), the rate of generated traffic was sufficiently low ($\lambda < 1$ MBps) and consequently, in this period the Enzymatic controller served to decouple patterns of arrival and departure processes. We did two tests for two different settings of the Enzymatic controller, both to cap the rate to the value $v_{\max} = 1$ MBps by using a byte-per-molecule ratio of 1000 BpMol. For the first test reported in Figure 3.12(a), we kept fixed k_1 and set first $k_2 = 1\,\text{s}^{-1}$ and $e_0 = 1000$ mol in order to obtain again $v_{\max} = 1$ MBps. Such a setting was sufficient to smooth the 1.5s-on/0.5s-off traffic – the sending rate did not attain the same level of burstiness that the arrival rate had. For the second test reported in Figure 3.12(b), we enforced a lower filtering effect by increasing coefficient k_2 to 5. To have again 1 MBps rate limit, we thus adjusted e_0 to 200. This

[3.17] The byte-per-molecule [BpMol] ratio specifies the mapping between communication and chemical systems. 1000 BpMol means creating a molecule S each time 1000 bytes arrive, and allowing the dequeueing of 1000 bytes each time reaction r_2 in (3.1b) fires. Note that in order to dequeue, for example, a 2500-byte packet and then a 1500-byte packet, reaction r_2 must occur three times, and then one time only.

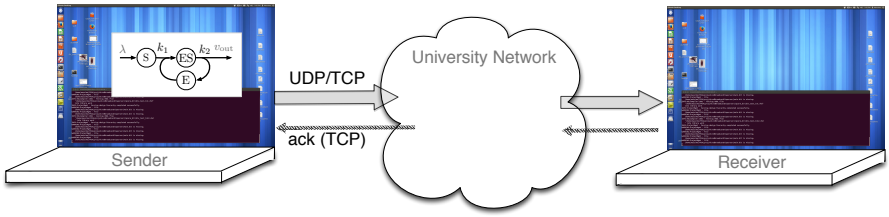


Figure 3.10: *The setup of the real-world experiment included two computers – one computer sent through the university network (attached via Ethernet 1000baseT) `iperf`-generated UDP or TCP traffic to the recipient one, which, in the case of TCP, acknowledged the correctly received packets. The service of the sender's egress queue (`iperf`-traffic only) was controlled by the Enzymatic-`tc` kernel module. We used default TCP-configuration as provided by iperf-tool in the (sender) Linux machine (TCP Cubic with window size 22.9 KB), and sent the traffic to a (recipient) Mac OS X 10.6.8 (TCP New Reno with window size of 256 KB). The Enzymatic controller was deployed at the egress device queue at the (sender) Linux machine.*

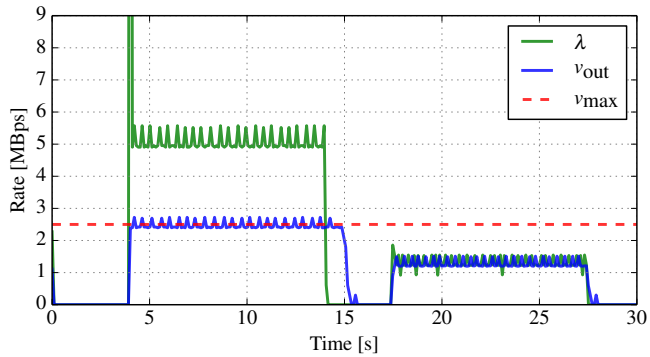
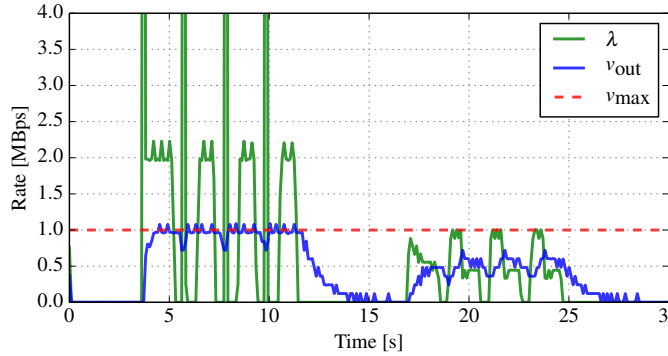
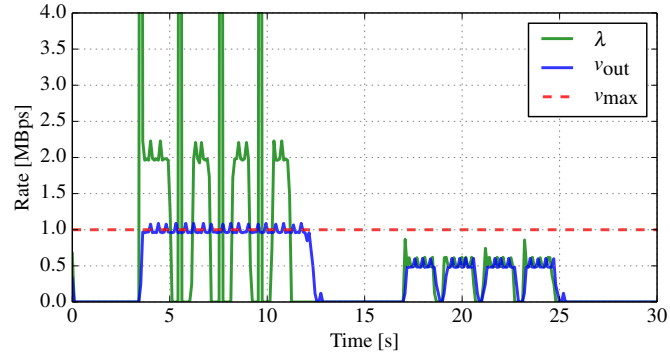


Figure 3.11: *Real-world experiments: Controlling UDP traffic by means of the Enzymatic controller. The iperf-generated UDP traffic had average rate λ so that, in the first phase ($t < 15$ s), $\lambda > v_{max}$, where v_{max} is the predefined rate cap, and in the second phase ($t > 15$ s), $\lambda < v_{max}$. The parametrization of the Enzymatic controller was $k_1 = 1\ (mol{\cdot}s)^{-1}$, $k_2 = 5\ s^{-1}$, $e_0 = 500$ mol, and the byte-per-molecule ratio was 1000 bytes/mol.*



(a) Higher Filtering



(b) Lower Filtering

Figure 3.12: *Real-world experiments: Controlling UDP traffic by means of the Enzymatic controller. The* iperf*-generated traffic was bursty and had average rate* λ *so that, in the first phase* ($t < 15$ *s*), $\lambda > v_{max}$, *where* v_{max} *is the predefined rate cap (1 MBps), and in the second phase* ($t > 15$ *s*), $\lambda < v_{max}$. *The parametrization of the Enzymatic controller related to Figure 3.12(a) was* $k_1 = 1\,(mol{\cdot}s)^{-1}$, $k_2 = 1\,s^{-1}$, $e_0 = 1000$ *mol to impose a higher low-pass filtering effect, and that related to Figure 3.12(b) was* $k_1 = 1\,(mol{\cdot}s)^{-1}$, $k_2 = 5\,s^{-1}$, $e_0 = 200$ *mol to impose a lower low-pass filtering effect. In both cases, we set* 1000 *byte* $\equiv$ *1 molecule.*

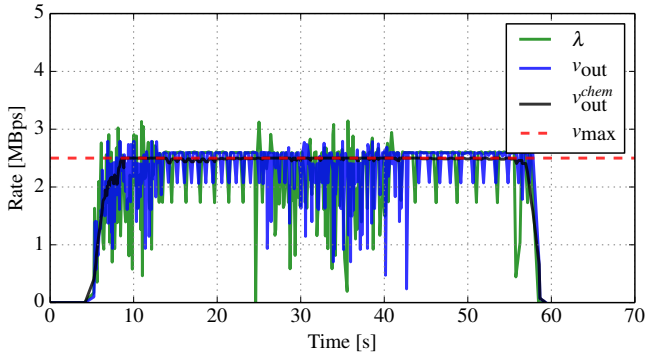


Figure 3.13: *Real-world experiments: Controlling TCP traffic by means of the Enzymatic controller. We had initially ($t < 24$ s) one TCP flow only, and later (at times $t = 24$ s and $t = 36$ s) two additional TCP flows, competing in the controlled environment (iperf-generated traffic). The black curve represents the rate at the output of the chemical system $v_{out}^{chem} = 10v_{r_2}$. The parametrization of the Enzymatic controller was $k_1 = 1\ (mol{\cdot}s)^{-1}$, $k_2 = 5\ s^{-1}$, $e_0 = 500$ mol, and the byte-per-molecule ratio was 1000 bytes/mol.*

setting sped up the controller response so as to allow tracking almost exactly the arrival rate and to let the input bursty pattern pass through.[3.18]

In the last test (see results in Figure 3.13), we controlled TCP traffic. Initially ($t < 24$ s), one TCP flow only was admitted. Afterwards (at times $t = 24$ s and $t = 36$ s), two additional TCP connections followed, competing in the controlled environment. Each time a new TCP connection was admitted, the system needed some time to stabilize to the predefined rate limit. After a few seconds, the flows reached the optimum rate value, efficiently occupying the available resource, no matter how many they were (one, two, three connections). Furthermore, we observed a good mixing of the three flows and a fair sharing of the available resource.[3.19]

In these results, especially those on TCP traffic, we can observe a jitter affecting the instantaneous rates, including the controlled output rate v_{out}. This is a consequence of mapping a molecule to a certain amount of bytes – each time reaction r_2 fires, a certain (fixed) amount of bytes is authorized to be dequeued and sent. For example, by using 1000 BpMol, each time reaction r_2 occurs, 1000-bytes are "authorized" to be dispatched altogether. However, if the amount of authorized bytes is not sufficient to cover the size of the next packet, the transmission of the packet is further postponed until enough bytes will be authorized by subsequent r_2-reaction executions. The effect of dealing

[3.18] Refer to Appendix A.3 for an exact and formal quantification of effects of k-coefficients.

[3.19] When all three flows competed, we observed that, at any moment in time, the instantaneous throughput of each flow was approximatively the same.

with packets (chunks of byte) rather than continuous quantities (or at least, small-sized packets) is appreciable by comparing the rate at the output of the chemical rate (r_2-reaction rate) with the dequeueing rate (the actual rate v_{out} at which packets are sent). In Figure 3.13, we plot the magnified ($\times 1000$) output rate $v_{\text{out}}^{\text{chem}}$ of the chemical system. Differently from the actual dequeueing rate v_{out}, the "authorized-byte rate" $v_{\text{out}}^{\text{chem}}$ turns out to be (*i*) steady, (*ii*) burst-free, and (*iii*) exactly equal to the predefined limit value – i.e., free from the discussed phenomenon. We can dampen such an effect by choosing a smaller bytes-per-molecules ratio. The cost is working with a higher number of molecules, and thus a higher computational effort required to implement the $\mathcal{AC}$ constituting the controller. (This aspect will be revisited in Chapter 5.)
Unlike traditional token-bucket mechanisms, the Enzymatic controller does not have a critical configuration such as the bucket size. The bucket size represents an important parameter to choose when controlling TCP flows with traditional token bucket schemes. This has effect on the performance of TB-controlled flow: generally, bigger bucket sizes lead to better TCP performance, in terms of how close the actual TCP-controlled flow matches the predefined rate value (refer to [236] and references therein for experiments on the impact of TB-schemes on TCP flows). However, large buckets means large data bursts into the network and thus difficulties in provisioning the network capacity, temporary congestion or increased drops/latency. In other words, the bucket size is a trade-off between goodput (i.e., a percentage of the predefined value $v_{\max}$) and accepted burstiness and latency.

3.4 Distributed Traffic Rate Controller (DTRC)

Until now, we have designed and analyzed a chemical rate controller that can enforce its policy independently at each network location. However, we typically have to cope with the far more difficult problem of controlling the aggregate network utilization and guaranteeing admissibility to all flows that demand access to the shared facility from multiple network locations.
We aim at designing and analyzing a class of Distributed Traffic Rate Controllers (DTRCs) that collaborate to enforce a single, aggregate global limit to the network traffic they collectively police. Specifically, the goal of DTRCs is to dynamically and distributively throttle the transmission rate in order to limit the cumulative load on a shared resource, as schematically represented in Figure 3.14. Typical use cases where such a solution is necessary in today's Internet include the traffic of a particular cloud-based service, specific DSL users' traffic, user- and control-traffic in data center management. The key difficulty in distributed rate controlling is to allow individual flows to compete dynamically for capacity, not only with flows traversing the same controller, but with flows traversing other controllers as well.
We opt for a fully distributed solution, where the aggregate rate cap emerges as

Fully distributed solution...

...emerging from LoMA

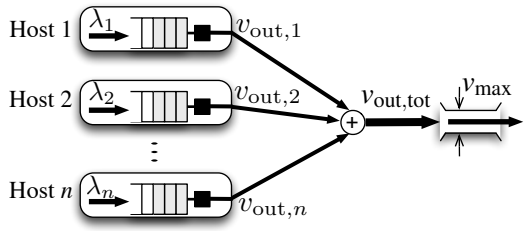


Figure 3.14: *We aim at dynamically and distributively throttling the transmission rate $v_{out,i}$ of each participating host i in order to limit the cumulative load $v_{out,tot}$ on a shared resource, in a fair and efficient manner.*

a result of the collaboration among $\mathcal{AC}$-controllers. Every controller estimates the instantaneous aggregate load by exploiting the direct mapping of packet quantity to rate, derived from the LoMA-scheduling environment. At the same time, every controller allocates capacity in proportion to the local load and the measured available capacity. The result is a system that

- *guarantees flow admissibility* per flow or controlled class of traffic: each host is guaranteed to access the shared resource,
- *is efficient (in capacity)*: if the virtual cumulative load (sum of local load, $\lambda_{\text{tot}} = \sum_i \lambda_i$) is less than the aggregate rate cap (low-traffic: $\lambda_{\text{tot}} < v_{\max}$) then, at each host i, the local rate must be equal to the local load ($v_{\text{out},i} = \lambda_i$); if the cumulative load is equal to or higher than the aggregate rate cap (overload: $\lambda_{\text{tot}} \geq v_{\max}$), then the actual aggregate rate must be equal to the aggregate rate cap ($v_{\text{out,tot}} = v_{\max}$),
- *is fair (in throughput)*: in case of overload ($\lambda_{\text{tot}} \geq v_{\max}$), each host's rate converges to a share $v_{\max,i}$ in response to its local load; the host i that does not overload the network ($\lambda_i < v_{\max,i}$) is not limited ($v_{\text{out},i} = \lambda_i$); the host j that overloads the network ($\lambda_j > v_{\max,j}$) is limited ($v_{\text{out},j} = v_{\max,j}$).

In Section 3.4.1, we introduce and explain the design of a chemical reaction network that implements the desired macro-behavior of the DTRC. In Section 3.4.2, we provide an in-depth analysis of the DTRC, concerning both steady-state and transient behavior, prove its stability, and give guidelines for its calibration. We validate our proposal in Section 3.4.3 and Section 3.4.4, by means of simulations and real world experiments. Finally, we discuss the derived insights in Section 3.4.5.

3.4.1 Design of DTRCs

The complete structure of the DTRC (in Figure 3.15) is an extension of the Enzymatic model we have discussed in the previous sections (refer to Figure 3.6(a) for the exact reaction model), with a series of simple chemical design motifs. In such a control scheme, S-molecules are generated at the rate λ at which packets are enqueued at the controlled queue. The dequeueing and consequent transmission of packets is instead authorized at rate v_{out}, which is controlled by the rest of the reaction network.

In the reaction network, we identify two inter-weaved reaction loops, one of which is encapsulated in (part of) the other. This pattern is highlighted in Figure 3.16(a). We denote the sum of concentrations along the smaller loop as $e_0 = c_{\text{E}} + c_{\text{ET}}$, and the sum of concentrations along the larger loop

Reaction loop ES-ET-E...

...an Enzymatic-like process

$$E_0 = c_{\text{ES}} + c_{\text{E}} + c_{\text{ET}} = const. . \tag{3.7}$$

The sum of concentrations along the larger loop (E_0) remains constant because of the mass-conservation principle. On the contrary, the sum of concentrations along the smaller loop (e_0) changes over time because molecules E are consumed by the two reactions r_1 and r_3 (r_1 is not part of that loop) and produced only by one (r_4). Similarly, molecules ET are consumed by only one reaction (r_4) but produced by two (r_2 and r_3, where again r_2 is not part of that loop). The difference of the two concentrations can be expressed as a function of the ratio v'_3/v'_1 of the reaction rates consuming E-molecules, where $v'_3 = k_3 c_{\text{E}}$ and $v'_1 = k_1 c_{\text{E}}$. The difference of quantities along the two loops $E_0 - e_0 = c_{\text{ES}}$, eventually transformed in F-molecules, determines the number of tokens available for dispatching packets from the input queue S. Thus, by replacing the single conservation loop of Figure 3.6(a) with the extended two-loop pattern of Figure 3.16(a) (and replacing e_0 with $E_0 - e_0$ in (3.2)), the output rate v_{out} becomes dependent on the ratio of outflow rates from E-species. By adjusting this ratio we can adapt the maximum rate limit $v_{\text{max},i} \leq v_{\text{max}}$, where $v_{\text{max}} \propto E_0$ and $v_{\text{max},i} \propto (E_0 - e_0)$.

Adaptive-rate-control motif

What is left to be designed is an extension that takes into account the transmission rates of the other hosts and throttles the own rate accordingly, such that $v_{\text{out,tot}} = \sum_{i=1}^{N} v_{\text{out},i} \leq v_{\text{max}}$. To achieve this, we condition reaction r_3 (which regulates the mass e_0 in the small loop) by the species S (which represents the fill-level of the controlled queue) and the species T which represents the "sensed" transmissions (output process) from other controllers. The relative contributions of S and T species to the r_3-reaction dynamically adjust the local cap for each controller according to the total aggregate cap all controllers are expected to respect. That is, the S-dependency of reaction r_3 enables achieving fairness among flows – i.e., the throttling of the local transmission rate is proportional to the related enqueueing rate. The T-dependency of reaction r_3 enables respecting the total rate cap – i.e., until the estimated aggregate rate is higher than the predefined limit, the local rate is further

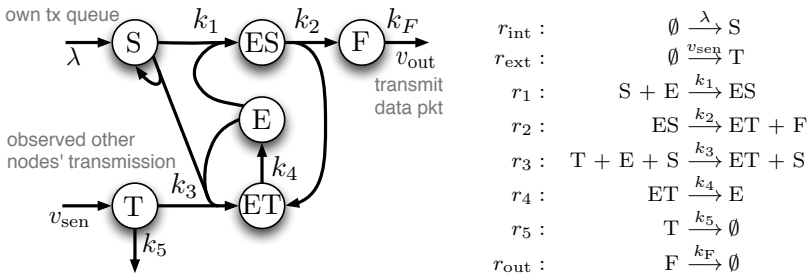
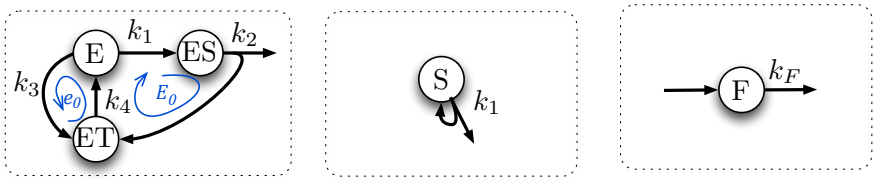


Figure 3.15: *An adaptive Distributed Traffic Rate Controller (DTRC). Complete* $\mathcal{AC} = \{\{S, E, ES, ET, T, F\}, \{r_{int}, r_{ext}, r_1, r_2, r_3, r_4, r_5, r_{out}\}, \mathcal{A}\}$ *implementing the DTRC functionality. The symbolism* $\emptyset \xrightarrow{v} X$ *signifies the generation of X-molecules at rate* v*, whereas* $X \xrightarrow{k} \emptyset$ *signifies the depletion of X-molecules with rate coefficient* k*.*



(a) Adaptive-rate-control (b) Quantity-to-rate (c) 1st-order-LP-filter

Figure 3.16: *Chemical motifs that compose the reaction network implementing the adaptive DTRC.*

throttled.[3.20] More in detail, concentration c_{S} participates in reaction r_3 as described by the pattern shown in Figure 3.16(b), the so-called *quantity-to-rate* pattern [163]. The looping arrow implies that for every molecule consumed from species S, a new S-molecule is produced. Thus reaction r_3 does not affect S-species concentration c_{S} but, at the same time, because of LoMA, its reaction rate is directly proportional to c_{S}. The effect of this pattern is that when the arrival rate λ at S is in flow-balance with rate v_1 of reaction r_1, the input queue is drained fast enough and there is no increasing backlog of S-molecules to "fuel" reaction r_3. The small loop is blocked and the system behaves like a fixed-rate token-bucket. As soon as the enqueuing rate λ increases too much and there are other flows claiming part of the network capacity, there will be an increasing backlog of S-molecules created, which activate reaction r_3 with a rate proportional to the backlog. As there are contributor species to reaction r_3, the rate increase will be further regulated according to the utilization of the

Quantity-to-rate motif

[3.20] How the sensing function is implemented is beyond the scope of this design. Depending on the application environment different solutions may be used – e.g., carrier sensing, multicast, gossip, explicit out-of-band signaling, and others. For example, we used an out-of-band UDP channel in the experimented later discussed in Section 3.4.4.

network by other users through the concentration c_{T}. The result will be an adaptive drop in the maximum allowed rate $v_{\mathrm{max,i}}$.

Filtering motif (1^{st}-order LP-filter)

The last component of the design is the F-species (Figure 3.16(c)), which implements a linear low-pass output filter. When the system is in equilibrium, the instantaneous variation of c_{F} will follow the variations of the influx rate v_2, and because of the LoMA, the average transmission rate will be: $v_{\mathrm{out}} = k_F c_{\mathrm{F}}$. This means that k_{F} determines the cut-off frequency for smoothing (dampening) the oscillations of the transmission rate acceleration. This is important because (*i*) it reduces the burstiness of transmissions and their effects on the sensing function of other rate controllers, (*ii*) in case of long delay networks, it "smoothens" the traffic variations over time so as to prevent fast oscillating responses.

3.4.2 Analysis of DTRCs

Having logically designed the behavior of this chemical flow controller, we have to assert its dynamics mathematically and find out how the reaction coefficients have to be tuned. In the following, we thus apply step by step the analysis methodology introduced in Section 2.3, and formally study the sensitivity/stability and calculate the calibrating parameters. For a detailed progression of analytical steps, refer to Appendix A.4.

3.4.2.1 DTRC: Fluid model

The following set of ODEs (directly extracted from the reaction set in Figure 3.15) connect the reaction rates with the changes of molecular concentrations (virtual queues' fill-levels) over time:

DTRC: fluid model

$$\begin{aligned} \dot{c_{\mathrm{F}}} &= k_2 c_{\mathrm{ES}} - \overbrace{k_F c_{\mathrm{F}}}^{v_{\mathrm{out}}} & \dot{c_{\mathrm{ES}}} &= k_1 c_{\mathrm{S}} c_{\mathrm{E}} - k_2 c_{\mathrm{ES}} \\ \dot{c_{\mathrm{T}}} &= v_{\mathrm{sen}} - k_3 c_{\mathrm{S}} c_{\mathrm{E}} c_{\mathrm{T}} & \dot{c_{\mathrm{ET}}} &= k_3 c_{\mathrm{S}} c_{\mathrm{E}} c_{\mathrm{T}} + k_2 c_{\mathrm{ES}} - k_4 c_{\mathrm{ET}} \\ \dot{c_{\mathrm{S}}} &= \lambda - k_1 c_{\mathrm{S}} c_{\mathrm{E}} & \dot{c_{\mathrm{E}}} &= k_4 c_{\mathrm{ET}} - k_3 c_{\mathrm{S}} c_{\mathrm{E}} c_{\mathrm{T}} - k_1 c_{\mathrm{S}} c_{\mathrm{E}} \end{aligned} \tag{3.8}$$

where the concentrations (queue fill-levels) c_{ET}, c_{ES}, and c_{E} are subject to the mass (token) -conservation constraint (3.7).

Reaction r_5 does not play a role in the principal operation of the rate controller. Rather, its role is to ensure that the system does not end up with stale state (more details at the end of Section 3.4.2.5).[3.21] For this reason and in order to keep the analysis uncluttered, we ignore it for now.

[3.21] It is important that r_5 is calibrated in the same way at each hosts i, i.e. $k_{5,i} = k_5$.

3.4.2.2 DTRC: Steady state

At steady state, the fill-levels of all virtual queues (concentrations) in the network do not change anymore. As usual, we set the derivatives of (3.8) to zero and solve the equations with respect to the fill-level variables c_i to obtain the following steady state:

$$
\begin{aligned}
{c_{\mathrm{F}}}^* &= \frac{\lambda}{k_F} & {c_{\mathrm{ET}}}^* &= \frac{v_{\mathrm{sen}} + \lambda}{k_4} \\
{c_{\mathrm{ES}}}^* &= \frac{\lambda}{k_2} & {c_{\mathrm{S}}}^* &= \frac{\lambda k_2 k_4}{(E_0 k_2 k_4 - \lambda(k_2 + k_4) - v_{\mathrm{sen}} k_2)\, k_1} \\
{c_{\mathrm{T}}}^* &= \frac{v_{\mathrm{sen}}}{\lambda}\frac{k_1}{k_3} & {c_{\mathrm{E}}}^* &= \frac{E_0 k_2 k_4 - (v_{\mathrm{sen}} + \lambda) k_2 - \lambda k_4}{k_2 k_4}.
\end{aligned}
\tag{3.9}
$$

DTRC: steady state (unsaturated regime only)

We intuitively expect a change in the behavior of each local controller when the aggregate transmission rate reaches the predefined rate limit. That is, we expect two regimes: unsaturated and saturated regime.

Unsaturated Regime When the aggregate transmission rate $v_{\mathrm{out,tot}}$ is less than the predefined rate limit $v_{\mathrm{out,max}}$ (in the unsaturated regime), we observe a stable fixed point for all species (virtual queues). Above $v_{\mathrm{out,tot}}$ (in the saturated regime), the concentration of species (fill-level of queue) S is likely to grow unbounded. This formally corresponds to a bifurcation point in the system trajectory. Such a bifurcation point can be extracted from our equations in (3.9) by enforcing (*i*) the mass-conservation expressed in (3.7), and (*ii*) concentrations (fill-levels) c_i to be positive values only. These constraints require the cumulative (offered) load to be below a certain value. Only in this case, the reaction network exhibits a steady state that is valid for all queues:

$$
\lambda < \frac{E_0 k_2 k_4 - v_{\mathrm{sen}} k_2}{k_2 + k_4}. \tag{3.10}
$$

We simplify (3.10) by assuming that $k_2 \gg k_4$ and by setting $k_4 E_0 = v_{\mathrm{max}}$. Thus, in the region

$$
\lambda < v_{\mathrm{max}} - v_{\mathrm{sen}}, \tag{3.11}
$$

Unsaturated regime...

...(average) load and transmission rate match

the transmission rate is not throttled and, at steady state, equals the cumulative (offered) load: $v_{\mathrm{out}} = k_F {c_{\mathrm{F}}}^* = \lambda$. This equality of offered and sent rate is only valid at steady state. The transient analysis will later reveal that in this region, the reaction network acts as a low-pass filter, which means that the transmission rate lags behind the cumulative load.

At the boundary condition where the input rate is $\lambda = 0$, there is no backlog ($c_{\mathrm{S}} = 0$) and consequently the output rate is $v_{\mathrm{out}} = k_{\mathrm{F}} {c_{\mathrm{F}}}^* = 0$, too. In this case the host will not occupy any capacity, and the other participating hosts can benefit from its unused share too.

Saturated Regime As soon as the condition (3.11) is not respected (i.e. the local load becomes higher than the admissible share), the backlog of S-molecules increases, in this way speeding up reaction r_3. The concentration of available E-molecules drops, forcing reaction r_1 to slow down. Unless the sender backs-off, the increasing reaction speed of r_3 will absorb more molecules in e_0 and tend to blocking transmissions.

However, a complete block is unlikely (subject to suitable selection of k_i parameters) because, as the reaction speed of r_3 accelerates, the concentration of T-molecules starts reducing faster, counter-balancing in this way the acceleration rate of r_3.

Saturated regime...

...fill-levels steady at a finite value

This gradual reduction of the acceleration will continue as the number of molecules T approaches the limit value

$$c_{\mathrm{T}}{}^{l} = \frac{k_1(k_2 + k_4)}{k_3 k_2 (k_4 E_0 / v_{\mathrm{sen}} - 1)}. \tag{3.12}$$

Thereafter, reaction r_3 will start decelerating and, as long as the system remains under "pressure" (condition (3.11) not respected), the transmission rate will be limited and will never exceed the maximum rate reinforced by the E_0 loop.

Furthermore, we can express the steady-state concentrations of all other molecule species in terms of input concentrations c_{S} and c_{T}. Of particular interest is the limit concentration $c_{\mathrm{F}}{}^{l}$, which is directly proportional to the controlled output rate:

...stationary transmission rate as $k_F c_{\mathrm{F}}{}^{l}$

$$c_{\mathrm{F}}{}^{l} = \frac{k_1 k_2 k_4 E_0 c_{\mathrm{S}}}{(k_2 k_4 + c_{\mathrm{S}}(k_1(k_2 + k_4) + c_{\mathrm{T}}{}^{l} k_2 k_3)) k_{\mathrm{F}}}. \tag{3.13}$$

We can observe that, if the local data generation rate λ is extremely high and c_{S} theoretically increases without bounds, the rate capping takes place. Indeed, in this limit case the output rate cannot exceed an asymptotic upper bound, which depends on the current average utilization of the resource by all other transmitters:

$$v_{\mathrm{out}}^{l} = k_{\mathrm{F}} c_{\mathrm{F}}{}^{l} \overset{(3.13),\ (3.12)}{=} \frac{k_4 E_0 - v_{\mathrm{sen}}}{1 + k_4/k_2}. \tag{3.14}$$

$k_4 E_0 = v_{\max}$ + $k_2 \gg k_4$...

...*efficiency* guarantee

If we further assume that the local controller is sending alone, i.e., $v_{\mathrm{sen}} = 0$, then from (3.14) we can infer a set of parameters that allow exploiting the (asymptotic) maximum available capacity of the resource: $k_2 \gg k_4$ and $k_4 E_0 = v_{\max}$. This setting guarantees the *efficiency* of the system (and complies with our previous assumption to get the simplified condition (3.11)).

The blocking case of $k_4 E_0 = v_{\mathrm{sen}}$ is theoretically possible only if there are non-cooperative controllers, or if there is an infinite number of cooperative controllers. That is, admissibility of all flows is guaranteed.

In Table 3.6, we summarize the average stationary features of the DTRC. For the amount of queued packets (concentration $c_{\mathrm{S}}{}^{*}$) and the expected waiting time, we have considered the setting $k_2 \gg k_4$ and $k_4 E_0 = v_{\max}$.

Property	Low Load $v_{\text{out}} = \lambda + v_{\text{sen}} < v_{\max}$	High Load $v_{\text{out}} = \lambda + v_{\text{sen}} > v_{\max}$
Arrival rate, λ	λ	λ
Expected # packets, c_{S}^*	$\sim \frac{\lambda k_4/k_1}{v_{\max}-\lambda-v_{\text{sen}}}$	max.q.size (th. ∞)
Constant # tokens, E_0	$c_{\text{E}}^* + c_{\text{ES}}^* + c_{\text{ET}}^* = const.$	$c_{\text{E}}^* + c_{\text{ES}}^* + c_{\text{ET}}^* = c.$
Exp. # free tokens, c_{E}^*	$\frac{E_0 k_2 k_4 - (v_{\text{sen}}+\lambda)k_2 - \lambda k_4}{k_2 k_4}$	0
Exp. # busy token, $c_{\text{ES}}^* + c_{\text{ET}}^*$	$\frac{(v_{\text{sen}}+\lambda)k_2 + \lambda k_4}{k_2 k_4}$	E_0
Exp. sending rate, v_{out}	λ	$\frac{k_4 E_0 - v_{\text{sen}}}{1+k_4/k_2}$
Exp. waiting time, d	$\sim \frac{k_4/k_1}{v_{\max}-\lambda-v_{\text{sen}}} + \frac{1}{k_2} + \frac{1}{k_F}$	–

Table 3.6: *Stationary average features of the DTRC. Refer to Figure 3.15 for the respective scheme.*

3.4.2.3 DTRC: Sensitivity

Through sensitivity analysis, we characterize *how* the transmission rate v_{out} adapts to changes of the local cumulative (offered) load λ and the sensed traffic v_{sen}. As we know, the derived results are valid for small perturbations that do not let the system cross the aforementioned bifurcation point. In the following analysis, we assume that the capacity $v_{\max}$ and all reaction coefficients remain constant.

Unsaturated Regime If the link capacity is not fully exploited, i.e. for a small rate λ, the TF is (refer to Appendix A.4 for details on how to derive the TF)

$$H_{\text{o-i}}(s)|_{(3.11)} = \frac{\lambda}{{c_{\text{S}}}^*} \cdot \frac{k_2 k_{\text{F}} (s + \frac{k_3}{k_1}\lambda)(s + k_4)}{(s + k_{\text{F}}) \cdot \sigma(s)} \tag{3.15a}$$

$$H_{\text{o-e}}(s)|_{(3.11)} = \frac{v_{\text{sen}}\lambda}{{c_{\text{E}}}^* {c_{\text{T}}}^*} \cdot \frac{k_2 k_{\text{F}}\ s}{(s + k_{\text{F}}) \cdot \sigma(s)} \tag{3.15b}$$

DTRC: transient behavior (unsaturated regime)

where

$$\sigma(s) = s^4 + \Big(k_2 + k_4 + \beta + k_3(\lambda + {c_{\text{T}}}^*)\Big)s^3 + \\ + \Big(\beta(k_4 + k_2 + k_3\lambda) + k_2 k_3(\lambda/k_1 + {c_{\text{T}}}^*)\Big)s^2 +$$

$$+\left(\frac{k_2k_4}{k_1}({c_\mathrm{E}}^* + k_3\lambda) + \beta k_2\lambda(k_3+k_4)\right)s+$$
$$+ k_2k_3k_4\lambda{c_\mathrm{E}}^*$$
$$\beta = \frac{(k_2+k_4)^2k_1c_\mathrm{S}^*}{k_2k_4} + \frac{k_2k_4}{c_\mathrm{S}^*}.$$

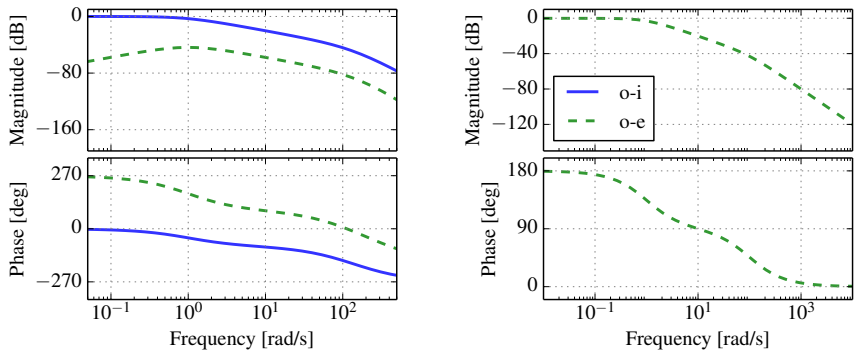
We derive the following interesting facts from this TF: First, it exhibits low-pass characteristics for perturbations of the cumulative (offered) load. With a unity gain for low frequencies, the transmission rate will adapt to the cumulative load on the long run. On the other hand, the algorithm behaves as a band-pass filter for changes of the sensed competing traffic. However, the band-pass filter has a very high attenuation, which means that the controller does not respond significantly to changes of the other hosts' transmission rates in the unsaturated regime.
Second, although the TF shows a dependency on the steady-state values of the inputs, the sensitivity of the algorithm does not change significantly for different inflow rates (as the TF's shape variation is only significant for very low magnitudes of the TF; low TF magnitudes announce an insensitivity to input perturbations anyway).
Third, the effect of the queue F is easy to isolate in the TF: This "LoMA-scheduled virtual queue" simply increases the order of the low-pass filter. Figure 3.17(a) depicts the magnitude and phase of the two TFs for an aggregate rate below the predefined rate limit; the contribution of $k_F = 1\,\mathrm{s}^{-1}$ is clearly visible as a pole at 1 rad/s. The other poles are defined by the coefficients (k_2, k_4) and (k_1, k_3), respectively. Thus, as we could expect, the reaction coefficients not only place the attractor of our algorithm in state space (i.e., define regions of the state-space, which "attract" the trajectories that the system takes through its state-space), but also define the algorithm's transient behavior.

Saturated Regime Beyond the bifurcation point, i.e., if the cumulative load is too large, the rate-limiting mechanism starts to work. For the limit $c_\mathrm{S} \to \infty$ we obtain a simple TF:

DTRC: transient behavior (saturated regime)

$$H_{\text{o-i}}(s)|_{\lambda+v_\text{sen}>v_\text{max}} \cong 0 \tag{3.16a}$$
$$H_{\text{o-e}}(s)|_{\lambda+v_\text{sen}>v_\text{max}} \cong \frac{-k_2k_F}{(k_F+s)(k_2+k_4+s)}. \tag{3.16b}$$

In this regime, a node's transmission rate is insensitive to variations of the local offered load and responds only to changes of sensed traffic v_sen; the local controller adapts its state according to changes of the global state of the entire system. In this regime, the algorithm behaves as a second-order low-pass filter with poles at $-k_f$ and $-(k_2+k_4)$ and negative unity gain at low frequencies (attenuation).



(a) Unsaturated Regime

(b) Saturated Regime

Figure 3.17: *Transfer Function (TF) of the controller in Figure 3.15, parameterized according to Table 3.7.*
(a) $v_{sen} + \lambda < v_{max}$*: Practically, only perturbations on* λ *cause the DTRC's response.*
(b) $v_{sen} + \lambda > v_{max}$*: Only perturbations on* v_{sen} *cause the DTRC's response.*

Figure 3.17(b) shows the magnitude and phase plots of the TF in the saturated regime. The desired effect of the low-pass filter is that sudden variations of the input rate do not pass through to the transmission rate such that the algorithm reacts smoothly to new load distributions.

3.4.2.4 DTRC: Stability

From the structure of the reaction network and its sensitivity we can prove that the DTRC exhibits a single stable attractor.

One basin of attraction

There is only one basin of attraction,[3.22] because the initial conditions (i.e., the queue fill-levels at time $t = 0\,\mathrm{s}$) have no influence on the long-term state of the system. Formally, we can explore the attractors of a chemical reaction network through two theorems: the Deficiency Zero Theorem [79, 115] and the Chemical Organization Theory [70]. Intuitively, we can observe that after a period of inactivity, when removing the external stimuli λ and v_{sen}, the reaction network always resets itself to $c_{\mathrm{S}} = c_{\mathrm{F}} = c_{\mathrm{T}} = c_{\mathrm{ES}} = c_{\mathrm{ET}} = 0$ and $c_{\mathrm{E}} = E_0$. The remaining molecules in species S (=remaining packets or bytes in queue S) are drained by reactions r_1 and r_3. The produced molecules in species ES are moved to species F, from where they are then sent away. All molecules (tokens) in ET sooner or later move to E.
From the TF, we are able to derive a simple proof that the algorithm's attractor is stable. On either side of the bifurcation point, the TF exhibits negative real-part poles, which signifies BIBO stability of the linearized model. That

[3.22] A basin of attraction is the set of all states whose trajectories lead to the same attractor.

is, the adaptation of our algorithm to bounded variations of the offered local load and/or the sensed rate will always produce a limited variation of the output rate v_{out}. The fact that input variations must be bounded is not a problem, because (*i*) if all participating controllers implement the presented reaction network (DTRC-scheme in Figure 3.15), the sensed rate respects the predefined rate limit $v_{\max}$, and (*ii*) even if the offered load increases without bounds, variations of the transmission rate remain bounded. Even though the actual reaction network is non-linear, this stability proof is sufficient and prevents us from needing more complex analytical tools.

3.4.2.5 DTRC: Derived guidelines for the calibration

From the analytical findings we have so far, we are able to identify the key parameters of the DTRC and derive some guidelines on how to chose the reaction coefficients. The coefficients k_2, k_4, and the total number of molecules (tokens) in the Enzymatic loop, E_0, together define the upper rate limit $v_{\max}$. We previously restricted $k_2 \gg k_4$, such that $v_{\max} \simeq k_4 E_0$. The freedom in distributing the target rate to k_4 and E_0 and in choosing k_2 is limited by the fact that coefficients k_2 and k_4 also define one of the low-pass poles and therefore influence the stability and reactiveness of the DTRC, as we discussed before.

The coefficients k_1 and k_3 have no evident effects on the location of the system's attractor, but they affect its transient behavior when the DTRC operates in the unsaturated regime, where $\lambda + v_{\text{sen}} < v_{\max}$. By increasing either k_1 or k_3 with respect to the other coefficient, the response time of the algorithm becomes shorter.

The coefficient k_{F} defines the cutoff frequency of the output-stage low-pass filter of queue F, which smoothes out bursty load changes and reduces fast oscillations of the transmission rate and of the overall emergent behavior. This is the most direct and effective tuning tool from a stability/reactiveness perspective, since k_{F} is independent of all other coefficients. However, choosing k_{F} is not so simple, because we have to find a tradeoff between reaction speed and stability. In general, its reciprocal $1/k_{\text{F}}$ should be higher than the maximum expected path delay of the feedback channel.

As we argued in Section 3.4.2.4, initial conditions (i.e., species concentrations at time $t = 0\,\text{s}$) do not influence the behavior of the system, except of course for the total number of molecules (tokens) we decide to put in the Enzymatic loop, E_0. To guarantee that the system does not remember and respond to stale state after a period of inactivity, we want to avoid infinitely accumulating molecules in c_{T} in a non-active host, which still senses active transmissions in the network. This is done by the draining reaction r_5. Coefficient k_5 defines how fast this process occurs; its calibration does not have to follow any particular rule (apart from the local policy, see Section 3.4.5).

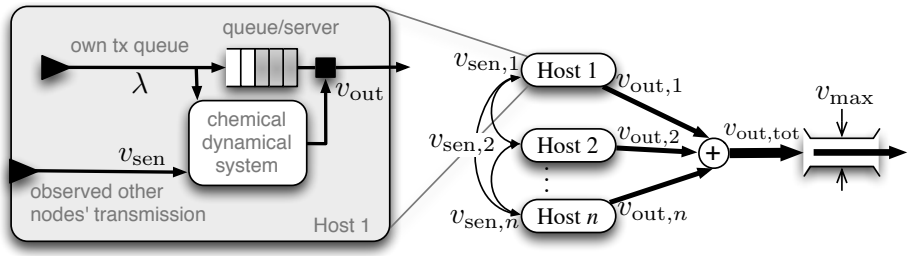


Figure 3.18: *We dynamically and distributively throttled the transmission rate $v_{out,i}$ of each participating host i in order to limit the aggregate load $v_{out,tot}$ on a shared resource with limited capacity v_{max}. This was done by controlling locally the dequeueing/transmission rate $v_{out,i}$ of each host i. The control mechanism worked by regulating the service process of the host's egress queue based on dynamics of the embedded (virtual) chemical system. The chemical system was configured according to the DTRC reaction model (Figure 3.15). Each host i generated and enqueued traffic at average rate λ_i pkt/s, and sensed the load ($v_{sen,i}$) of the shared resource.*

$c_S(0) = 0$ mol	$c_E(0) = E_0$ mol	$k_1 = 1\,(\text{mol}{\cdot}\text{s})^{-1}$	$k_4 = 1\,\text{s}^{-1}$	$E_0 = 10^3$ mol
$c_T(0) = 0$ mol	$c_{ET}(0) = 0$ mol	$k_2 = 100\,\text{s}^{-1}$	$k_5 = 1\,\text{s}^{-1}$	
$c_{ES}(0) = 0$ mol	$c_F(0) = 0$ mol	$k_3 = 1\,(\text{mol}^2{\cdot}\text{s})^{-1}$	$k_F = 1\,\text{s}^{-1}$	

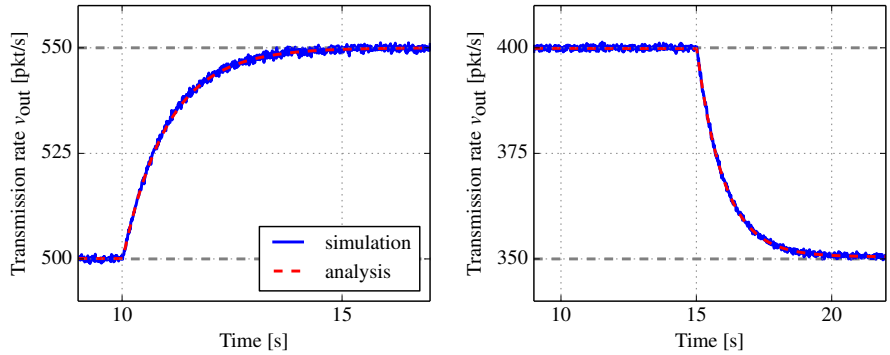
Table 3.7: *Theoretically-derived parametrization of the DTRC for experiments.*

3.4.3 DTRC: Simulations in OMNeT++

We evaluated the chemical rate controller in the OMNeT++ simulation environment. We have deployed a chemical controller at each node's output queue, where operations were on packet-basis (i.e., molecule concentrations were packet counters), refer to Figure 3.18. We have used the system settings shown in Table 3.7 (chosen according to findings of the previous section). The total aggregate rate limit v_{out} was set through the E_0 mass at 1000 pkt/s.
In the first set of experiments, we aimed at demonstrating that the theoretical predictions are in line with measurements during simulations, in ideal and non-ideal conditions. In the second set of experiments, we tested efficiency, fairness, and flow-admissibility-guarantees in a network of eleven hosts. We tested the performance with continuous traffic as well as in more critical conditions when connections were in fits and starts, creating severe burstiness of the traffic.

3.4.3.1 DTRC: Analytical prediction *vs.* simulation measurements

We claimed in the introduction that our chemical approach helps to bridge the gap between microscopic packet interaction during execution on one hand, and macroscopic design and analysis of traffic-shaping algorithms on the other hand.



(a) Unsaturated regime

(b) Saturated regime

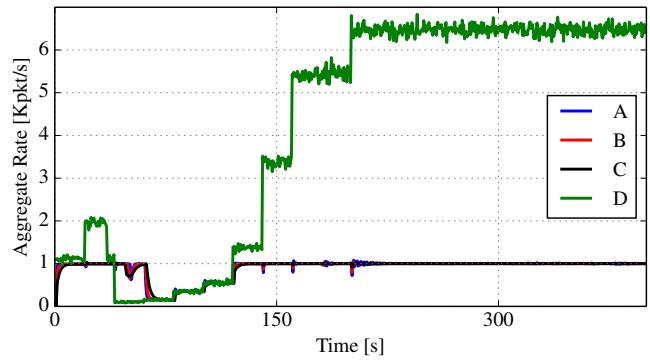
Figure 3.19: *Empirical validation of the step response of a single DTRC for a predefined capacity limit of* $v_{max} = 1000\ pkt/s$.
(a) In $\lambda + v_{sen} < v_{max}$ *region:* $t = 0\ s$: $\lambda = 500\ pkt/s$, $v_{sen} = 300\ pkt/s$; $t = 10\ s$: $\lambda \rightarrow 550\ pkt/s$; $t = 15\ s$: $v_{sen} \rightarrow 350\ pkt/s$.
(b) In $\lambda + v_{sen} > v_{max}$ *region:* $t = 0\ s$: $\lambda = 1400\ pkt/s$, $v_{sen} = 600\ pkt/s$; $t = 15\ s$: $v_{sen} \rightarrow 650\ pkt/s$.

To demonstrate this, we compare the analytically-predicted time response of the DTRC to its empirical behavior in the OMNeT++ simulator. Specifically, we configured a single DTRC-controlled host with the setting in Table 3.7, stimulated its inputs (λ, v_{sen}), and observed the effects at its output v_{out} in two separate tests. We plotted these results against those obtained analytically. Figure 3.19(a) shows the step response of the DTRC when operating in the unsaturated regime: Starting from steady state in the unsaturated regime, we suddenly increased the local load at time $t = 10\,\text{s}$. As a consequence, the output rate smoothly increased to the new attractor in conformance to our analytical prediction. We then let the sensed rate v_{sen} decrease at time $t = 15\,\text{s}$, which showed no visible effect on the output rate, since the total rate of all nodes was still below the total limit. In Figure 3.19(b), we plot the step response when the system operated in the saturated regime, i.e., the generated load was higher than the predefined cap. As predicted, an increase of the sensed total rate v_{sen} at time $t = 15\,\text{s}$ let the reaction network adjust its transmission rate in order to respect the predefined rate limit again.
To validate the usefulness of the analysis and the derived guidelines also for more realistic scenarios, we tested several network topologies and with variable feedback delays. Here we report on a scenario where eleven hosts generated Poisson-generated traffic at different *continuous* rates. All virtual hosts had to share a maximum link capacity of $v_{\text{max}} = 1000\,\text{pkt/s}$, and each host i sensed the utilization of the shared link $v_{\text{sen},i}$ with a variable feedback delay. The out-of-band feedback was implemented by notifying hosts of packet transmissions.

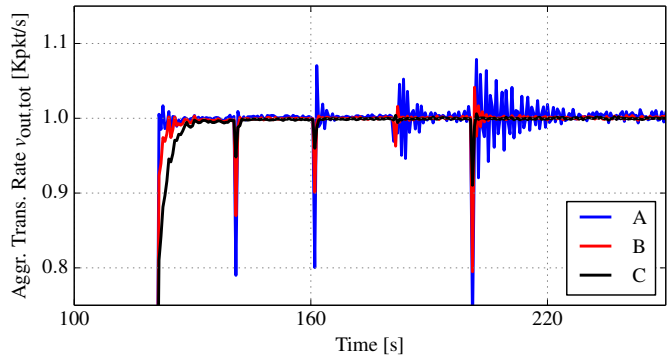
We performed trials for feedback delays ranging from no delay to uniform delay distributions up to $\mathcal{U}(0, 1000)$ ms.
We verified our capability of configuring the total transmission rate by setting the number of tokens E_0 and the reaction coefficient k_4 accordingly ($E_0 k_4 = v_{\max}$). Additionally, we verified that higher values of k_4 reduced the settling time. However for high feedback delays, this caused dampened oscillations in the cumulative load. As expected, we were able to mitigate these fluctuations by reducing k_F at the cost of a slower adaptation speed.
Figure 3.20 shows that the analysis guided us correctly in tuning the algorithm. It plots the emergent aggregate rate v_{out} measured in a scenario with a high feedback delay for three different values of k_F: Reducing k_F to a value lower than the reciprocal feedback delay made oscillations no longer significant.

3.4.3.2 DTRC: fairness, efficiency, and flow-admissibility

Having validated the theory-based calibration of the DTRC, we investigated fairness, efficiency, and flow-admissibility. Here, we report on results obtained in an eleven-host network, affected by variable sensing delays in the range $\mathcal{U}(0, 1000)$ ms, where DTRCs were calibrated according to Table 3.7.
Figure 3.21 shows that the *capacity allocation is efficient*: It depicts the cumulative input and resulting output of the system, i.e., the cumulative offered load $\lambda_{\text{tot}} = \sum_i \lambda_i$ and the total utilization of the capacity $v_{\text{out,tot}} = \sum_i v_{\text{out},i}$. We observe that *on average* the total utilization respected the set limit: $v_{\text{out,tot}} \leq v_{\max}$ (although *instantaneous* rates may assume any value). In an unsaturated regime, i.e., $\lambda_{\text{tot}} < v_{\max}$, the aggregate transmission rate matched the cumulative load, i.e., $v_{\text{out,tot}} = \lambda_{\text{tot}}$. This was true with continuous traffic (Figure 3.21(a)) as well as with bursty traffic (Figure 3.21(b)).
Figure 3.22 shows that the *capacity allocation is fair*: The transmission rate of each node $v_{\text{out},i}$ converged to a fair share over the cumulative offered load. For a certain condition, which was defined by the number N of participating hosts, their average local load λ_i, and the predefined rate limit $v_{\max}$, all hosts converged to the same fair share value v_S, unless their average local load value was lower than the share value ($\lambda_i < v_S$). Namely, $v_{\max,j} = v_S \quad \forall \lambda_j \geq v_S$. For the sake of clarity in Figure 3.22(b), we show in more detail the relations between transmission rate and system load for a few individual nodes in the continuous-traffic tests: Host 3, who was not overloaded (as it generated packets at a very low rate), did not experience rate limitation. Host 4, who generated data slightly above its permissible transmission rate, experienced a slight rate regulation. Host 10, who was significantly overloaded, experienced major rate regulation (its transmission rate was significantly lower than its packet generation rate on admission to the network, $v_{\text{out},10} < 0.2\lambda_{10}$). Those hosts that experienced rate regulation (hosts 4 and 10) converged to the same rate share value $v_S \sim 110$ pkt/s. Based on these results, we can claim that, in a scenario where the transmission rates of competing hosts are regulated by DTRCs, each host *is guaranteed* to have access to the shared resource.

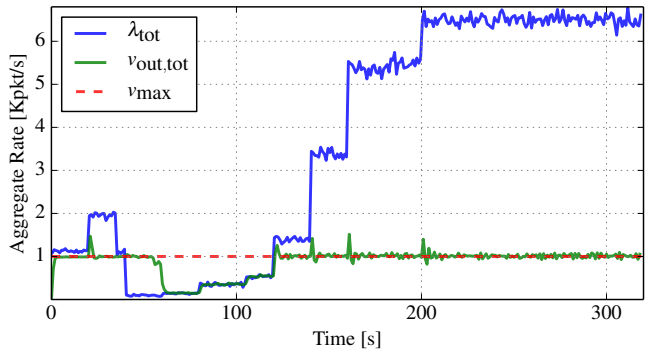


(a) Rate

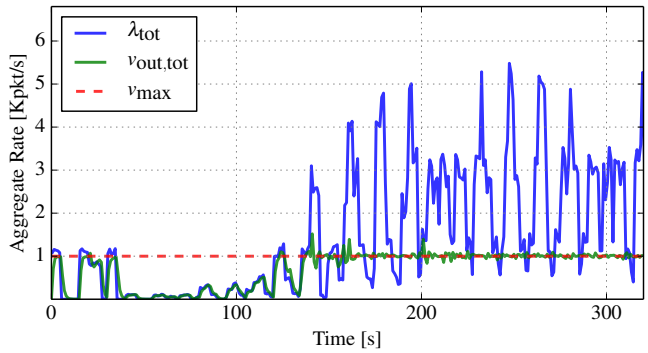


(b) Transm. Rates Details

Figure 3.20: *OMNeT++ simulation of 11 DTRC-controlled hosts, parameterized according to Table 3.7. The rate limit was configured at* $v_{max}=1000\,pkt/s$*, the feedback delay was taken from a uniform distribution* $U(0,1000)\,ms$*.*
A blue line: Aggregate transmission rate ($v_{out,tot}$) *for* $k_F = 2\,s^{-1}$
B red line: $v_{out,tot}$ *for* $k_F = 1\,s^{-1}$
C black line: $v_{out,tot}$ *for* $k_F = 0.5\,s^{-1}$
D green line: Cumulative offered load $\lambda_{tot} = \sum_i \lambda_i$ *(only in chart (a)).*

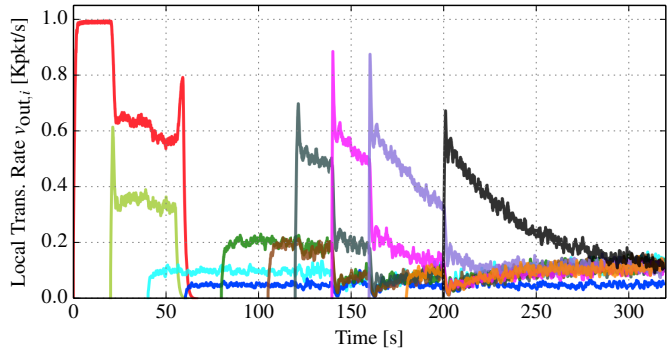


(a) Continuous traffic generation

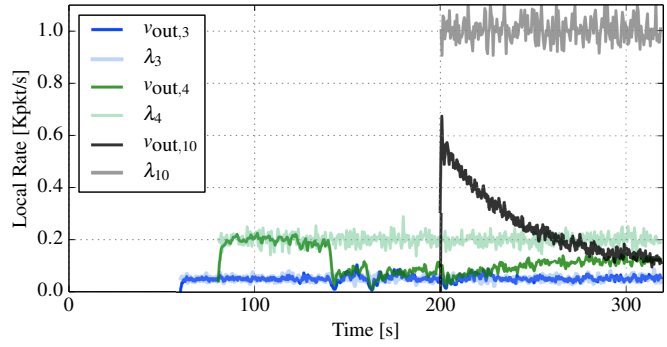


(b) Bursty traffic generation

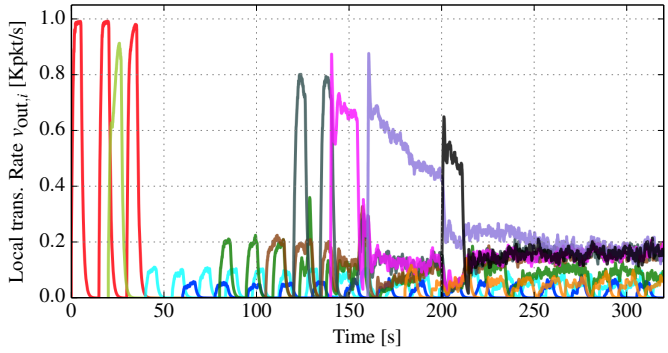
Figure 3.21: *Cumulative offered load λ_{tot} of 11 nodes and total network utilization $v_{out,tot}$. The predefined rate limit was set to $v_{max}=1000\,pkt/s$.*



(a) Continuous traffic



(b) Details on continuous traffic tests



(c) Bursty traffic

Figure 3.22: *Individual transmission rates $v_{out,i}$ of the 11 nodes responding to each other's transmission rates. (a) Steady/continuous traffic. (b) Offered load per node λ_i (in pale colors) and effective transmission rate $v_{out,i}$ (in solid colors) for continuous-traffic tests: Node n_3 starts at $t=60\,s$, n_4 at $t=80\,s$, and n_{10} at $t=200\,s$. (c) Bursty traffic (in fits and starts – 5s-on/10s-off traffic).*

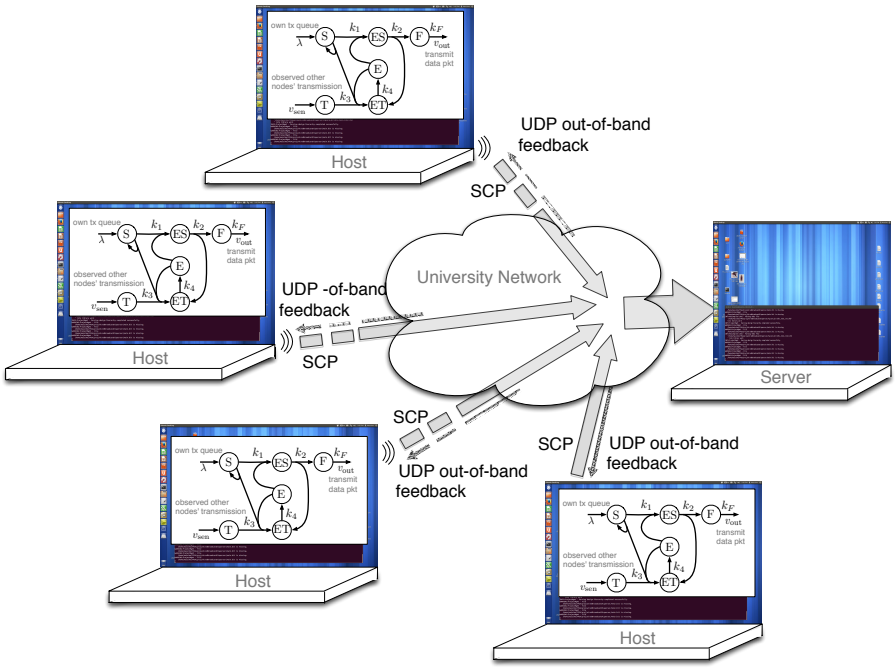


Figure 3.23: *The setup of the real-world experiment included four hosts, one attached to an Ethernet segment (1000baseT) and the other three on the same WLAN (802.11 n). The feedback was provided via out-of-band UDP channel. Each of the four machines generated traffic of the* `tc`*-controlled class by copying a large file to an Internet server via secure copy (*`scp`* over TCP/IP).*

3.4.4 DTRC: Experiments in a real environment

We moved our experiments from simulation to real/native deployment, and integrated the chemical control network in the traffic-control framework of Linux and controlled with it a queue that isolates a certain class of traffic (Figure 3.14). As in the experiments with the Enzymatic controller (Section 3.3.4), we worked with the existing queues in the Linux kernel. Our implementation operated on byte rates to account for packets of different lengths, by mapping each molecule to each kbyte: A packet entering the egress queue of the network device triggered the generation of an S-molecule in the chemical control plane for each kbyte. Similarly, each output molecule authorized the dequeueing of 1 kbyte from the queue.

Our network setup (see Figure 3.23) consisted of four hosts, one attached to an Ethernet segment (1000baseT) and the other three on a Wireless LAN (WLAN 802.11 n). The feedback channel between the hosts was provided by an out-of-band UDP channel. Each of the four nodes generated traffic of the

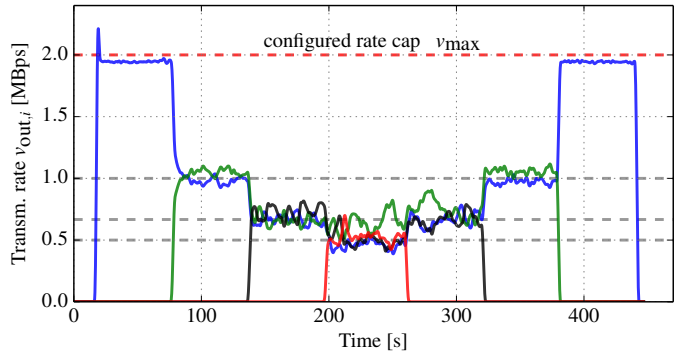


Figure 3.24: *Transmission rates $v_{out,i}$ of four nodes in a real deployment. The chemical rate controller triggered the egress device queue in the Linux traffic control system of the corresponding nodes. Traffic was generated by TCP* `ssh/scp` *sessions. The aggregate rate limit for all rate-controlled traffic was set to $v_{max} = 2$ MBps (via E_0), the filter coefficient to $k_F = 100\,s^{-1}$, and the feedback channel delay was ~ 10 ms.*

`tc`-controlled class by copying a large file to an Internet server via secure copy (`scp` over TCP/IP), starting at different times and for various time durations. We calibrated the DTRC reaction network according to the guidelines of the previous section, summarized in Table 3.7, with two exceptions: (*i*) We set the aggregate rate limit to $v_{\max} = 2$ MBps (via E_0) and (*ii*) we set the low-pass-filter coefficient to $k_{\mathrm{F}} = 100\,\mathrm{s}^{-1}$, because the feedback delay was shorter in this scenario (in the order of ms).

Figure 3.24 illustrates the results that confirm the predicted behavior of the system. Every time a new flow accessed the service, the total capacity was divided equally among the nodes under a rate-fairness regime: 1 flow at 2 MBps, 2 flows at 1 MBps each, 3 flows at 0.65 MBps each, and 4 flows at 0.5 MBps each. This was also validated by transfer rates reported at the end of each `scp` session.

One observation is that as the number of nodes accessing the service increased, the stochasticity did so too (although on average it converged to the per-node share). This came from how the TCP exited from the fast-recovery phase when the rate limit was lowered (i.e., whether it succeeded to stay in congestion avoidance or if it dropped to slow start and then re-entered congestion avoidance).

A second observation is that the allocated aggregate capacity was asymptotically matched but never 100 % utilized, even when there was only one flow admitted (single node). This was related to small variances of the feedback signals (expressed through c_{T}) and of concentration c_{S}, which kept reaction r_3 always active even at a very low speed.

3.4.5 Insights on Distributed Traffic Rate Control

Soft limit on the aggregate access rate

DTRCs collaborate to regulate in a distributed setting a predefined, global rate limit. The reinforcement of this limit is *soft*: the controllers do not synchronize the transmissions (i.e., the *instantaneous* aggregate rate can assume any value). Instead, they cooperate to converge to an *average* aggregate rate that respects the limit. This is much more suitable for the admissibility of flows like TCP-flows, than a hard reinforced cap.

Implicit load estimation (LoMA-based)

Each controller actively monitors and infers its fair share from the current network load (the aggregate transmission rate), without requiring knowledge of the number (or identity) of other transmitters. Moreover, the inference is a continuous process that does not need any explicit or sophisticated discrete calculation: LoMA-scheduling effectively bridges workloads and rates through a simple predictable relationship that underpins the reaction network's operation.

Hard limit on the *local* access rate

The node's *instantaneous* transmission rate never exceeds the predefined threshold. This capability stems from the properties of the Enzymatic pattern. Note that, in contrast to the traditional token-bucket with feedback-control, the basic scheme we used (Enzymatic pattern) has smoother transient characteristics and is free from overshooting-problems. As a result, it is suitable to operate also in bursty-traffic conditions, more common in Internet traffic.

Modular and extendable core design...

The core design we have presented is modular and extendable with chemical reaction components, e.g., for refining the sensing inputs (total utilization, partial utilization), for alternative allocation of rates (equally, proportionally to traffic class, etc.), or for utilizing different feedback channel (the concentration c_T can be associated to ECN marks, out of band signals as in the studied case, or mere carrier sensing in a CSMA medium). This extensibility alongside the parameterizability of the core design can lead to a family of rate controllers suitable for different applications and environments. For example, the DTRC-mechanism can be used directly as a random Media Access Control (MAC) algorithm that guarantees the same maximum throughput of the traditional Pure ALOHA protocol but exhibits better performances in terms of stability. Appendix A.5 treats in detail such an enzymatic MAC protocol and gives an analytical comparison with the ALOHA protocol.

...MAC mechanism

As another example of new facilities via core-design parameterization, we can use DTRCs as a `DiffServ`-like mechanism [223]. By simply regulating a single reaction coefficient (k_5), we can enforce different forwarding regimes for the different traffic classes, while still satisfying the aggregate rate limit, and guaranteeing the efficiency of the system. Indeed, besides avoiding system stale states, the k_5-coefficient controls the speed of the linear reaction draining T-species and thus affects the accounting of others' transmissions. Thus, by differentiating k_5-values among the different DTRCs, we can enable service class weighting.[3.23] We performed further experiments on service-class differentiation, whose results are reported in Appendix A.6.

...service-class differentiation

[3.23] The differentiation of k_5-value has impact also on the speed at which inactive nodes (flows) return to the initial state. However, this effect is not appreciable in the output rates.

3.5 Conclusion

We have started the chapter by advocating the advantages of Law of Mass Action (LoMA) scheduling in the context of packet switching and queueing networks. LoMA-scheduling induces a pace between dequeued packets, proportional to the instantaneous state of the queue. The derived inherent packet-pacing enables a better statistical multiplexing of packet flows in networking environments. LoMA-scheduling also improves the stability and robustness of the system as it leverages stable attractors. Intuitively, LoMA-triggered services lead to smooth transitions as a consequence of the little effect that single packets have on state space. More formally, a LoMA-system is a dynamical system that exhibits smooth and large basins of attraction, and thus perturbations, which move the system a small distance from the original state, have little effect (see [221] for a detailed discussion on the topic).
Using therefore concepts from chemical kinetics, we proposed a novel method to design, implement, and analyze traffic-shaping algorithms. The proposed approach is not yet another tool for modeling existing algorithms. Rather, it provides an engineering framework (*i*) in which traffic-shaping algorithms and their operations can be represented and programmed as reaction networks, (*ii*) which generates an executable model based on interacting queues and LoMA-scheduling, and (*iii*) which provides sound analytical tools to study and tune the algorithm's behavior.
Such a chemical engineering framework bridges the micro-level implementation of the discrete event system and the satisfaction of requirements at the macro-level, hard to overcome with traditional methods. Algorithm design follows intuition at the macroscopic level and complex behavior can be obtained from simple nature-inspired reaction-network motifs. Then, our compiler translates these macroscopic properties to an executable network of virtual queues, which can be attached to and regulate real queues on nodes so as to orchestrate the forwarding of packets.
At the same time, we have a formal mathematical description that allows us to make/verify design choices and to understand/predict the behavior of the algorithm. We can adopt control theory and LTI analysis, which are common-ground in the design of networking algorithms, to analyze the system behavior and leverage the design process. Further, the sensitivity analysis proposed in this section can be applied to systems where the controller is distributed and embedded in the whole network (like in biological systems). In contrast to the traditional analysis of LTI systems where the input and output signals are given, the sensitivity analysis allows us to use any flow, state, or coefficient as input or output, to study its influence on the stability and sensitivity of the overall system.
At the micro-level, we can describe a "Chemistry-inspired queueing network" as well as a traditional queueing network by stochastic processes. At the same time, there is a subtle difference between the two approaches in the *design* of network systems. Using queuing theory, one starts by engineering

the interaction of independent arrival and departure processes of a queue, and observes the resulting effects when queues interact. In the Chemistry-inspired design, the departure process is directly linked to the queue's fill-level and thus, we can directly focus on engineering the interactions (reactions) among queues (species) to build a system that has the behavior (outputs) we want.

Chapter 4

Artificial Chemistry to Solve the Consensus Problem in Wireless Sensor Networks

"We cannot solve our problems with the same thinking we used when we created them."
Albert Einstein

In this chapter, we look at the consensus problem through the eyes of a chemist. This provides an alternative interpretation of the problem itself and new tools that possibly lead to benefits in terms of analysis and/or implementation: We gain in analyzability as we can make use of the underlying chemical theory to study for example convergence, stability, and speed. At the same time, we have at hand an approach to design and implement (with very simple communication techniques) possible solutions that can be directly and successfully used in wireless sensor networks.
This chapter is structured as follows: We first introduce in Section 4.1 the "consensus" problem, by briefly summarizing the most relevant solutions and clarifying our scope and contributions. We formalize the system model and the problem in Section 4.2. In Section 4.3, we start looking at the consensus from a chemical point of view, by first considering pairwise-communication-based systems and then wireless sensor networks with broadcast communications. We study in Section 4.4 the performance of the proposed solution by looking at convergence, stability, and speed. In Section 4.5, we report on simulation results and on comparisons with two known classes of consensus algorithms. We then extend the basic chemical consensus model and illustrate in Section 4.6 how Artificial Chemistry can be beneficial to overcome some practical issues such as measurement and communication errors. We report on the results from real-world experiments with a four-node hardware testbed in Section 4.7. Finally, in Section 4.8, we discuss the insights developed through out this chapter and some open issues.

4.1 Introduction

Consensus problem

The implementation of Wireless Sensor Networks (WSNs) poses several technical challenges. One of primary importance is conjugating the relative unreliability of a single sensor (due to its limited complexity and energy availability) with the high reliability required by certain applications (surveillance, healthcare, factory-automation, in-vehicle sensing and so forth). For this reason, an intense research activity has been devoted to design algorithms whereby clusters of sensors may reach an agreement on certain quantities of interest in a distributed manner, increasing in this way the system reliability.

This problem is known in the literature as the *consensus problem* and has received great attention from many different research communities (in computer science, control and information theory, wireless communications and signal processing). The research area ranges from the design of small, reliable hardware to low-complexity algorithms and energy saving communication protocols. In general, researchers have approached the consensus problem mimicking self-organizing complex systems, such as the human society; gossip algorithms represent a simple and robust approach for distributed data-dissemination and computation in WSNs, by metaphorical reference to the epidemic spreading of gossip in a social network. The general mechanism relies on nodes that iteratively exchange information with their neighbors until the entire network reaches a consensus on the value being computed. *Average computation* represents the simplest and most general application, where sensors deployed in a network, each having an initial value (e.g., a measurement of a temperature), aim to distributively calculate the average of all these values (e.g., average temperature).

In the following, we summarize the most relevant solutions to the consensus problem, and discuss the chemical principles introduced in [163,165] to achieve consensus by means of a chemical gossip-style model. Then, we clarify what our contributions in this context are.

4.1.1 Related works

The consensus problem has received great attention from many different research communities. Good treatments of the results obtained in this field can be found in [22,66,92,180] and references therein. Most of the works are inspired by different mechanisms, such as biological interactions [30, 52], formation control [248], spreading of gossip in social networks [36], synchronization of coupled oscillators [217], and belief propagation [58].

Synchronous *vs.* asynchronous

Generally, the proposed solutions rely on different communication infrastructures: In the literature, we find algorithms that require synchronized [20, 250] or non-synchronized [22, 33, 36, 63, 175] communications, where nodes update their state at the same time instant or on the contrary, they update their state at different time instants, in a randomized fashion. For example, Boyd *et al.* proposed in [36] two time models with comparable performance: In the

synchronous time model, where all nodes communicate simultaneously, time is assumed to be slotted commonly across nodes. In each time slot, each node contacts one of the nodes within its connectivity radius (i.e., neighbors) independently and randomly, and exchanges a state variable to produce a computation update. By contrast, in the *asynchronous* model, only one node communicates at a given time. In this approach, each node has a clock which ticks according to a Poisson process. At every tick, each node contacts one of its neighbors and exchanges a state variable. If each update results in a pairwise average of the two nodes' state values, the operation preserves both the total sum, and hence also the mean, of nodes' state initial values. This algorithm has proven to converge to consensus if the graph is strongly connected.[4.1] However, this scheme is vulnerable to packet collisions and yields a communication complexity in the order of the square of the number of sensor nodes, in random geometric graphs. A common feature to both approaches is that each node contacts only one other node at a time and the communication is sequential and pairwise, and thus bidirectional communications are assumed.

Randomized gossip algorithm

Aysal *et al.* showed in [21, 22] that the convergence speed is extremely improved by exploiting the broadcast nature of the wireless medium. Nodes asynchronously transmit a scalar-valued message, and each time a node receives a message from its neighbors, it performs an update by forming a convex combination of the received value with its own previous value. Then, when it is a given node's turn to broadcast next (as determined by a random timer, according to the asynchronous model [35,232]), the node broadcasts its current value. Such an approach simplifies the implementation aspect too, by removing the need for addressed information exchanges. However, the algorithms provide biased estimations depending on the type and connectivity of the network graph (e.g., for large highly connected graphs); the average is not preserved from iteration to iteration and consequently, for any particular sample path the consensus value is not precisely equal to the average.

Broadcast gossip algorithms

Broadcast and wireless communications inherently imply the issue of interference among simultaneous communications. To this end, works such as [179,218] have considered forms of admission control into consensus and gossip algorithms. Alternatively, Nazer *et al.* in [176] have proposed "computation coding", a new channel coding technique to compute sums reliably over the wireless medium.

To improve the convergence speed and/or to avoid biased estimations, solutions have exploited special features of the underlying communication graph. For example, the algorithm proposed in [100] exploits the natural superposition property of wireless multiple-access channels by letting nodes in a cluster transmit simultaneously their pre-processed initial states to a designated cluster head. Avrachenkov *et al.* in [20] assumed the local knowledge at nodes about the network topology. Their solution improves the convergence speed but

[4.1] A graph is said to be strongly connected if every vertex is reachable from every other vertex.

(i) for each modification of the network topology, nodes need to learn the local structure of the environment where they operate, and (ii) the performance of the local average consensus algorithm is strongly dependent on the type of network and turns out to be satisfying only for clustered networks. Dimakis *et al.* in [67] assumed an underlying infrastructure enabling the nodes to know their geographical location and those of its neighbors. They show that the performance can improve by geographically routing the exchange of information.
Common to most of the proposed solutions, the state variable is symbolically encoded into packets that are exchanged based on preexisting infrastructures that convey data; communication protocols and the low layers of the system have to guarantee a minimum communication service. These intermediate elements often play an important role in the energy budget and also in the definition of the overall system dynamics and stability.

4.1.2 Space and scope of our contribution

Consensus in balanced directed graph (WSNs)

In this chapter, we focus on WSNs characterized by balanced directed graphs (i.e., graphs in which in-degree and out-degree of each node are the same) and propose to look at the consensus problem through the eyes of a chemist. To this end, we first introduce a very simple chemical model that makes clear how key concepts of Distribute Artificial Chemistry ($\mathcal{DAC}$) can assist us in the context of distributed average computation to (i) formalize interactions among distributed nodes in a chemical manner, (ii) model the dynamics of interaction systems/networks in the form of ODEs, and (iii) predict the system's equilibrium points.
As we will see, the use of $\mathcal{DAC}$ leads to a consensus model in the same form of that proposed in [180], on which implementation-oriented works such as [129, 136, 137, 249] rely in order to realize distributed computation systems. In these works, the engineering process of required communication and networking protocols focusses on finding the interaction mechanisms that well-approximate the underlying model. By contrast with the chemical approach, the intuitively derived $\mathcal{DAC}$ represents both an abstract description model and a precise communication technique to let sensors converge on the value being computed. Furthermore, as we have already mentioned in the thesis, the use of chemical theory enables using new analytical tools; we prove the convergence and stability of consensus models by using the Deficiency Zero Theorem.
The first attempt to use $\mathcal{DAC}$s for achieving consensus can be found in [163–165], in the context of packet-oriented communication networks. Meyer modeled the pairwise exchange of information in terms of chemical reactions and offered an implementation model, the "Disperser", that encodes the node's state variable into an amount of molecules present at the node, in a sort of representation-free information encoding. It has been shown via simulations that this model improves the robustness exhibited by other epidemic models such as the one in [127], at the cost of a higher message complexity (i.e., a higher

number of iteration required for converging). Differently from [163–165], we implement and study Chemistry-inspired mechanisms for achieving consensus in WSNs. Specifically, our major contributions are the following:

- We use Chemistry to (*i*) construct a consensus model for WSNs characterized by balanced directed graphs, and (*ii*) prove convergence and stability in the derived system.
- We use simulation results, obtained with the network simulator OMNeT++ [240], to validate the analysis under different network topologies and number of sensor nodes; we also provide comparisons with other traditional gossip-inspired mathematical models.
- We take advantage of the underlying chemical theory to account for practical issues, i.e., we enable self-configuration and self-stabilization.
- We validate the performance of the proposed approach by means of a four-node hardware implementation, which relies on an emergent and simple communication protocol where nodes exchange their data in an asynchronous manner with no need for admission control. To the best of our knowledge, this is the first time that a Chemistry-inspired algorithm for communication is built in a hardware testbed and validated under real-world conditions.

Broadcast-based

Asynchronous

Topology-agnostic

Like in other broadcast-based approaches (e.g., [21,22,44,72,92]), our algorithm exploits the broadcast nature of the wireless medium and lets nodes broadcast their information. The algorithm does not require a synchronism between nodes, hence, it is suitable for WSN applications where synchronization itself can be a challenging task. Our approach differs from approaches that presuppose the local knowledge of the network topology. Furthermore, keeping the main distinctive characteristics of the Disperser, the algorithm we proposed does not encode state information into symbols and does not resort to exchange of packets, rather, it continuously modulates the rate at which nodes emit, for example, Radio Frequency (RF) pulses. This allow us (*i*) to keep small the energy consumption and (*ii*) to refer to a continuous-time analysis of an exact, easy-to-extract fluid-model. From the analysis point of view, we do not adopt analytical techniques traditionally applied in the consensus context, such as Markov chain theory, optimization and percolation, and graph theory (refer to [213] for a brief summary). Rather, we exploit the new analytical tools available thanks to the chemical metaphor.[4.2]

We validate the performance of the chemical interaction mechanisms by means of simulations under different operating conditions and settings. For example, in many scenarios, networks can possess a dynamic topology (these networks are also referred to as "switching networks" [180]), which changes in time due

[4.2] The adjective "new" refers to the consensus-context. We have already introduced these analytical tools in Chapter 2 and used in Chapter 3.

Experiments in...

...switching networks

to failures and creations of links and nodes. This can be related to formation reconfiguration [183], evolution [29], packet-loss [104], asynchronous consensus [108], state-dependence [162], or flocking-based motion [182]. In simulations and real-world scenarios, we have experimented with networks that have a dynamic topology and a varying number of participating sensors.

We make comparisons with existing benchmark solutions that rely on gossip protocols: the randomized gossip algorithm [36], which requires many iterations to converge to the exact average value, and the broadcast gossip algorithm [22], which leads to fast convergence on a value that may differ from the average. We then take advantage of the underlying chemical theory to include, in the dynamical system, mechanisms to account for some practical issues, such as the estimation of the number of neighboring nodes (*self-configuration*) as well as perturbations to the communication and measurement processes (*self-stabilization).*

...a hardware testbed

To validate the proposed solution and to demonstrate that the chemical paradigm is not relegated to pure theory, we also present some real measurements obtained from a four-node hardware testbed. In our simple implementation, the information exchange among the spatially distributed sensors takes place by means of the transmission of a pulse whose rate is proportional to the state of each node. The experimental results are in line with analytical and simulation results, and show that the proposed solution performs reasonably well even under real-world conditions. For the first time, chemically-engineered algorithms lie directly at the physical layer and control the basic hardware transmission technologies of communication systems in a protocol-less environment.

4.2 Problem statement: the average consensus problem

We consider a cluster of $|\mathcal{V}|$ low-mobility sensors, WSNs organized in hierarchical levels, where the lower-level nodes cooperate to achieve local consensus with a reliability greater than the one obtained with a single node, and intermediate nodes convey the information gathered by the lower-level nodes to the control centers. Sensors are connected by wireless links and composed of the following basic components: (*i*) a continuous-time dynamical system (i.e., the local $\mathcal{AC}$) whose state evolves in time according to local measurements and to the states of nearby sensors, (*ii*) a radio transceiver operating in a half-duplex manner that is used to transmit to and receive from nearby sensors, and (*iii*) a transducer that is used to monitor the physical parameter of interest and the related local detector that processes the measurements taken by the sensor. Figure 4.1 summarizes graphically the described system.

Communication network graph

By using the terminology introduced in Chapter 2, we model the interaction topology of the WSN as a directed graph (digraph) $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in which $\mathcal{V} = \{\nu_1, \nu_2, \ldots, \nu_{|\mathcal{V}|}\}$ is the set of all sensors with $|\mathcal{V}|$ being the number of sensors, while $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, with the convention that $\epsilon_{ij} \in \mathcal{E}$ if and only if

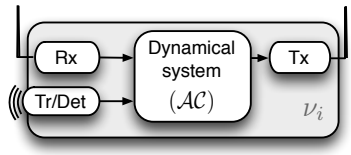


Figure 4.1: *A sensor node ν_i is composed by a continuous-time dynamical system (i.e., $\mathcal{AC}$), a radio transmitter (Tx-module), a radio receiver (Rx-module), and a transducer (Tr-module) and a detector (Det-module).*

there exists an edge from ν_i to ν_j (i.e., the information flows from ν_i to ν_j). We consider network topologies where there is a directed path connecting any two arbitrary nodes of the graph.[4.3] The structure of a digraph can be described by the $|\mathcal{V}| \times |\mathcal{V}|$ adjacency matrix $\mathbf{\Delta}$ whose generic entry $[\mathbf{\Delta}]_{i,j}$ is equal to 1 if $\epsilon_{ij} \in \mathcal{E}$ and 0 otherwise. As mentioned in Section 4.1.2, we concentrate on balanced digraphs for which the number of edges entering and leaving a node is the same for all nodes, i.e.,

$$\sum_{j \neq i} [\mathbf{\Delta}]_{i,j} = \sum_{j \neq i} [\mathbf{\Delta}]_{j,i} \quad \forall \nu_i \in \mathcal{V}. \tag{4.1}$$

For notational convenience, we denote by $\mathcal{N}_i$ the neighbor set within the transmitting and receiving range of sensor ν_i, i.e.,

$$\mathcal{N}_i = \{\nu_j \in \mathcal{V} \mid \epsilon_{ij} \in \mathcal{E}\}. \tag{4.2}$$

Neighbors

The goal of this work is to design, analyze and implement a dynamical system to distributively calculate at each node in the network the average of measured values. By denoting as z_i the discretized measurement of sensor ν_i, we can formalize this goal as follows:

$$z_{\text{avg}} = \frac{1}{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{V}|} z_i. \tag{4.3}$$

Goal

Average computation represents the simplest and most general case of consensus problem. Indeed, it represents a prototype for a variety of distributed tasks. Averaging can easily be adapted to the distributed computation of arbitrary linear projections, and it can even be extended to detection and filtering over networks [49, 208]. Efficient averaging algorithms are therefore of considerable interest.

[4.3] The notion of "strong connectivity" applies to digraphs; for undirected graphs we have simply "connectivity".

4.3 A Chemistry-inspired consensus algorithm

Average computation can *naturally emerge* in a system where mass-action, mass-conservation, and flow-conservation are valid principles. To show this, we first consider in Section 4.3.1 a simple example that helps to understand how more complex chemical reaction networks can be used for distributed computation. We then generalize such a model in Section 4.3.2 to achieve consensus in pairwise-communication systems (the "Disperser"). Finally, we introduce in Section 4.3.3 a general consensus model for WSNs, assuming broadcast communications.

4.3.1 Chemical reversible model

Consider a vessel in which two molecular species S_1 and S_2 are present and interact with each other according to mass-action kinetics and the following simple reaction rule: consuming a molecule S_1 produces a molecule S_2 and vice-versa.[4.4] It can easily be proven that, following the above interaction, such a system reaches an equilibrium in which molecules S_1 and S_2 are present in the same quantities. This is achieved for any initial concentration of S_1 and S_2 and without being explicitly programmed.

Let us formalize the above "chemical reversible" model in terms of $\mathcal{DAC}$ and consider the reaction network whose graphical illustration is given in Figure 4.2(a). The network graph is $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = (\nu_1, \nu_2)$ and $\mathcal{E} = \{\epsilon_{12}, \epsilon_{21}\}$. Each node ν_i defines only a molecular species S_i so that $\mathcal{S}_1 = \{S_1\}$ and $\mathcal{S}_2 = \{S_2\}$. This means that $\mathcal{S}_1^{(2)} = \{S_2\}$ and $\mathcal{S}_2^{(1)} = \{S_1\}$ whereas $\mathcal{M}_1 = \mathcal{M}_2 = \{S_1, S_2\}$. Additionally, each node defines a single reaction rule that consumes one instance of local S-molecules to produce one instance of remote S-molecules. This leads to the following "spatially distributed" reactions:

Chemical reversible model

$$r_1 : \quad S_1 \longrightarrow S_2 \tag{4.4a}$$
$$r_2 : \quad S_2 \longrightarrow S_1. \tag{4.4b}$$

Collecting the above facts together yields $\mathcal{DAC} = \{\{S_1,\ S_2\},\ \{r_1,\ r_2\},\ \mathcal{A}\}$. By applying the mass-action principle (2.6) to the distributed system (4.4), we get that concentrations change over time according to the following set of ODEs:

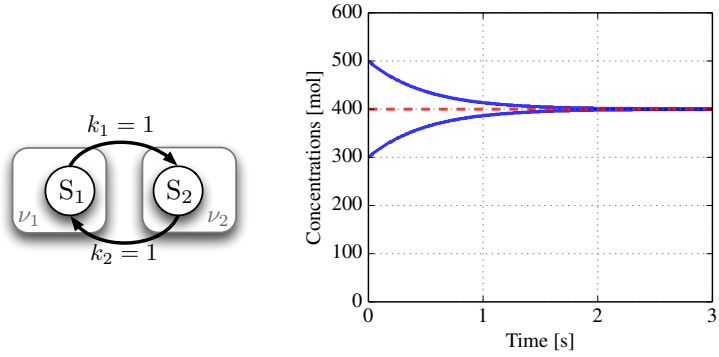
Chemical reversible model: Fluid-model

$$\dot{c}_{S,1}(t) = c_{S,2}(t) - c_{S,1}(t) \tag{4.5a}$$
$$\dot{c}_{S,2}(t) = c_{S,1}(t) - c_{S,2}(t) \tag{4.5b}$$

where we have considered stoichiometric and reaction coefficients of reactions r_1 and r_2 all equal to one.

When reaching equilibrium at time instant $t = t^\star$, the abundance of molecules does not change (i.e., $\dot{c}_{S,1}(t^\star) = \dot{c}_{S,2}(t^\star) = 0$) and the two molecular species are

[4.4] Such a chemical system models the simple "mate-and-spread game" [163].



(a) Reaction system (b) Simulation measurements

Figure 4.2: *Chemical reversible model - The chemical reaction system exhibits an equilibrium point wherein concentrations (blue-continuous line in Figure 4.2(b)) converge to the arithmetic mean (red-dashed line in Figure 4.2(b)) of their initial concentration values ($c_{S,1}(t=0) = 300$ and $c_{S,2}(t=0) = 500$).*

present in the same amount i.e., $c^*_{\mathrm{S},1} = c^*_{\mathrm{S},2}$. Therefore, by denoting the initial amount of molecules of species S at node ν_i as $c_{\mathrm{S},i}(0)$ and studying (4.5) at equilibrium, we obtain

Emergent average computation

$$c^*_{\mathrm{S},1} = c^*_{\mathrm{S},2} = \frac{c_{\mathrm{S},1}(0) + c_{\mathrm{S},2}(0)}{2}. \tag{4.6}$$

This proves that the simple chemical interaction mechanisms, defined by the $\mathcal{DAC}$, enable to balance the number of molecules between the two nodes. Namely, in this simple model, the goal (4.3) of the average consensus is achieved by encoding the node's state into the amount of molecules present in the node, and exchanging molecules among nodes according to mass-action kinetics.
In Figure 4.2(b), we report the time-evolution of molecular concentrations $c_{\mathrm{S},1}(t)$ and $c_{\mathrm{S},2}(t)$ when they are initialized as $c_{\mathrm{S},1}(0) = 500$ and $c_{\mathrm{S},2}(0) = 300$. The results are obtained in the network simulator OMNeT++ by letting two nodes operate according to the artificial chemistries $\mathcal{AC}_1$ and $\mathcal{AC}_2$. As we can observe, the concentrations of the two species converge to the arithmetic mean of the initial values (red-dashed line).[4.5]
The fact that in the chemical reversible model, concentrations of species have the same value at equilibrium, which is equal to the arithmetic mean of the initial amounts of molecules, stems from three principles we already know:

[4.5] In this simulation, as well as in other experiments whose results are reported in this chapter, we use deterministic inter-reaction times. The presented results are still valid, in average terms, for random inter-reaction times such as those drawn from exponential and gaussian distributions. Of course the randomness (noise) affecting the trajectories changes according to the used distribution.

(i) According to the *mass-conservation* principle, the total amount of molecules within the system does not change over time. (ii) According to *mass-action* principle (2.6), the following single unimolecular reaction $r_1 : \mathrm{X} \rightarrow \mathrm{X}$ is characterized by the rate $v_1 = c_{\mathrm{X}}$. This means that such a simple reaction is sufficient to convert the concentration of species X to a reaction rate v_1. In the above pattern, we observe further that, being X both reactant and product species, it is possible to map its concentration to a rate without altering the amount of X-molecules. Although not concerning the chemical reversible model, this design aspect plays an important role in the chemical consensus algorithm for WSNs that we propose and explain at the end of this section. (iii) According to the *flow-conservation* principle, a reaction rate can be converted back to a concentration value. Indeed, if a molecule Y is produced at a certain rate v_1 and consumed by a unimolecular reaction r_2 with unitary rate coefficient ($k_2 = 1$), the average rates of the two reactions must have the same value at equilibrium, and thus according to the LoMA, the draining rate value of r_2-reaction must match the species concentration c_{Y}. Mathematically, we have that if $r_1 : \emptyset \xrightarrow{v_1} \mathrm{Y}$ and $r_2 : \mathrm{Y} \xrightarrow{1} \emptyset$ then, at equilibrium, $v_1^* = v_2^*$. From the LoMA, we have that $v_2^* = c_Y^*$ from which it follows that $c_{\mathrm{Y}}^* = v_1^*$. Although simple, this example is instrumental to understand what mentioned in Section 4.1: modeling network interactions as a $\mathcal{DAC}$ provides (i) the microscopic mechanisms (the reactions and their time intervals of execution through reaction algorithm $\mathcal{A}$) to achieve a specific macroscopic requirement (the average) as well as (ii) the ODEs that are needed to describe the network dynamics and to eventually compute its equilibrium points.

4.3.2 A chemical gossip-style model: The "Disperser"

Rate-based gossip-like model

The chemical reversible model represents the simplest $\mathcal{DAC}$ to achieve consensus. This model has been generalized to achieve average consensus for an arbitrary number of nodes that communicate over a fully-connected graph and through bidirectional links [163–165]. Similar to gossip or epidemic-based protocols such as *Push-Sum* [127], the so-called "Disperser protocol" disseminates information to randomly selected neighbors. Differently from [127], this is achieved by exploiting the rate-based information exchange, which results naturally from a reaction-centric view of communications.

The Disperser basically extends the chemical reversible model to a network with more than two nodes. This is easily achieved by letting each node ν_i contain a molecular species S_i whose concentration $c_{\mathrm{S},i}$ represents the computed average.[4.6] For each link ϵ_{ij} connecting the ν_i-node to the ν_j-node, there is a spatial distributed reaction r_{ij} that consumes an S_i-molecule of the set located at node ν_i, and remotely produces an S_j-molecule of the set

[4.6] The concentration of S_i-species is initially set to the local value $c_{\mathrm{S},i}(t = 0) = c^0_{\mathrm{S},i}$. For example in a sensor network, the initial concentration $c^0_{\mathrm{S},i}$ should match the initially-sensed quantity z_i.

located at the neighbor node ν_j. This can be mathematically formulated in $\mathcal{DAC}$-terms as $\mathcal{DAC}_i = \{\{\mathrm{S}_i, [\mathbf{S}]_j\}, [\mathbf{r}_i]_j, \mathcal{A}\}$ for all values j satisfying the neighboring-condition in (4.2). The set of species $\mathcal{M}_i$ is constituted by the union of $\mathcal{S}_i = \{\mathrm{S}_i\}$ and $\mathcal{S}_i^{(j)} = \{[\mathbf{S}]_j\} = \{\mathrm{S}_j | \, j \in \mathcal{N}_i\}$. Whereas, the $|\mathcal{N}_i|$ reactions are in the following form

Generalized chemical reversible model

$$r_{ij}: \quad \mathrm{S}_i \to \mathrm{S}_j \;\mid\; \nu_j \in \mathcal{N}_i$$

stating that the execution of the single r_{ij}-reaction serves to trigger the transmission of a chemical-packet (i.e., Fraglet) from node ν_i to node ν_j. Thanks to the LoMA, like for the simple two-node network of the chemical reversible model, the network is composed by nodes that send packets (or molecules) to neighbors with a rate proportional to their state (i.e., the local average estimate), according to the quantity-to-rate motif. At the same time, this incoming rate affects/updates the node's state, according to the rate-to-quantity motif. The Disperser model does not exploit the broadcast nature of wireless communications and it basically requires each reaction to be associated with a mono-directional link. This calls for the addressed exchange of packets, which basically gives each node the possibility to discern the transmission to and the reception from its neighbors.

4.3.3 A Chemistry-inspired consensus algorithm for Wireless Sensor Networks (WSNs)

WSNs usually impose communication protocols that are simple in terms of routing and address-managing computation, due to the limited complexity and energy availability of sensors. Moreover, as demonstrated by works such as [20–22,72,92,100,145,176,179,267], the convergence speed of consensus algorithms is extremely improved by exploiting the broadcast nature of the wireless medium. To avoid tasks such as routing and address-managing computation, we need to extend the chemical paradigm illustrated previously (as eventually shown in Figure 4.3).
We start setting $c_{\mathrm{S},i}(0) = z_i$ and defining the following "broadcast" reaction

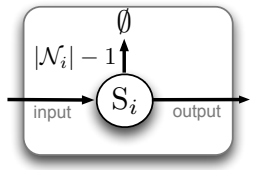
Output broadcast reaction

$$r_{B,i}: \quad \mathrm{S}_i \xrightarrow{1} \sum_{j \in \mathcal{N}_i} \mathrm{S}_j \tag{4.7}$$

from which it follows that the consumption of a single molecule S_i at node ν_i produces one instance of S-molecules at all of its $|\mathcal{N}_i|$ neighbors. This is how broadcast transmission can be modeled in a chemical way. According to the LoMA (2.6), the above reaction occurs at a rate equal to the concentration of the local state, i.e.,

$$v_{B,i}(t) = c_{\mathrm{S},i}(t).$$

Reaction $r_{B,i}$ represents the output of node ν_i that creates molecules in its neighboring nodes $\nu_j \in \mathcal{N}_i$, according to mass-action dynamics. Dually, we



(a) Reaction Network

$$\text{input}\begin{cases} r_{B,j_1}: & S_{j_1} \xrightarrow{1} S_i + \sum\limits_{\substack{\iota\in\mathcal{N}_{j_1}\\ \iota\neq i}} S_\iota \\ r_{B,j_2}: & S_{j_2} \xrightarrow{1} S_i + \sum\limits_{\substack{\iota\in\mathcal{N}_{j_2}\\ \iota\neq i}} S_\iota \\ \vdots & \\ r_{B,j_{|\mathcal{N}_i|}}: & S_{j_{|\mathcal{N}_i|}} \xrightarrow{1} S_i + \sum\limits_{\substack{\iota\in\mathcal{N}_{j_{|\mathcal{N}_i|}}\\ \iota\neq i}} S_\iota \end{cases}$$

$$\text{output}\begin{cases} r_{B,i}: & S_i \xrightarrow{1} \sum\limits_{j\,\in\,\mathcal{N}_i} S_j \end{cases}$$

$$\text{internal}\begin{cases} r_{D,i}: & S_i \xrightarrow{|\mathcal{N}_i|-1} \emptyset. \end{cases}$$

(b) Reaction Equations

Figure 4.3: *Chemical model for broadcast-based consensus. The broadcast reaction $r_{B,i}$ produces a molecule S in all neighbors $\nu_j \in \mathcal{N}_i$ of node ν_i. The $|\mathcal{N}_i|$ reactions $r_{B,j}$ in its neighbors create a single molecule S in node ν_i. The draining reaction $r_{D,i}$ controls the diffusion phenomenon created by broadcast communications.*

have $|\mathcal{N}_i|$ reactions $r_{B,j}$ in its neighbors that create a single molecule S in node ν_i. This represents the input of node ν_i, which is made of reactions of the following form:

Input reaction set

$$r_{B,j}: \quad S_j \xrightarrow{1} \overbrace{S_i + \sum_{\substack{\iota\in\mathcal{N}_j\\ \iota\neq i}} S_\iota}^{\sum_{l\in\mathcal{N}_j} S_l} \quad \text{for } j = 1,\ldots,|\mathcal{N}_i|. \tag{4.8}$$

The rate at which molecules are created at node ν_i is given by the sum of the state values of neighbors $\nu_j \in \mathcal{N}_i$:

Reception rate

$$v_{\mathrm{rec},i}(t) = \sum_{j\,\in\,\mathcal{N}_i} c_{\mathrm{S},j}(t). \tag{4.9}$$

The execution of reaction r_B at each node increases the total number of molecules in the network, thereby violating the mass-conservation principle. To overcome this "diffusion phenomenon", we need to further define a reaction that drains the abundance of S-molecules at each node on the basis of the number of its neighbors. This means locally performing at each node the following "draining" reaction:

Draining reaction

$$r_{D,i}: \quad S_i \xrightarrow{|\mathcal{N}_i|-1} \emptyset, \tag{4.10}$$

whose rate of occurrence is given by

$$v_{D,i}(t) = (|\mathcal{N}_i| - 1)\, c_{\mathrm{S},i}(t)$$

as it follows by applying the LoMA. By collecting the above facts together, the dynamical system of node ν_i is defined as

Chemical broadcast consensus model

$$\mathcal{DAC}_i = \{\mathcal{M}_i, \mathcal{R}_i, \mathcal{A}\} \tag{4.11}$$

with the set $\mathcal{M}_i$ given by the union of $\mathcal{S}_i = \{\mathrm{S}_i\}$ and $\mathcal{S}_i^{(j)} = \{\mathrm{S}_j | \, j \in \mathcal{N}_i\}$, whereas the reaction set is $\mathcal{R}_i = \{[\mathbf{r}_B]_j, r_{B,i}, r_{D,i}\}$. Figure 4.3 shows the graphical illustration of such a $\mathcal{DAC}$ and summarizes the reaction equations.

4.4 Convergence and performance of chemical consensus algorithms

We prove the convergence of the proposed chemical consensus model through traditional tools and through new analytical tools directly derived from the chemical perspective (Section 4.4.1). We also estimate the performance of the chemical consensus model in terms of convergence speed (Section 4.4.2).

4.4.1 Steady state and convergence

Thanks to the chemical metaphor, we directly extract the fluid model of the algorithm from its executable model, and describe the system dynamics in terms of a set of $|\mathcal{N}_i|$ ODEs, having the following form:

Fluid model

$$\dot{c}_{\mathrm{S}_i}(t) = \sum_{j \in \mathcal{N}_i} c_{\mathrm{S}_j}(t) - |\mathcal{N}_i| c_{\mathrm{S}_i}(t), \quad c_{\mathrm{S},i}(0) = z_i. \tag{4.12}$$

We observe that the implementation of the $\mathcal{DAC}_i$ in (4.11) at each sensor node $\nu_i \in \mathcal{V}$ leads to dynamics equivalent to those obtained by deploying the reactions of the generalized reversible model (Disperser) at each node ν_i:

$$\mathrm{S}_i \longrightarrow \mathrm{S}_j \;\; | \;\; \nu_j \in \mathcal{N}_i. \tag{4.13}$$

4.4.1.1 Steady-state and linear-stability analysis approach

Following the approach pursued in [163], the convergence proof comes from the analysis of steady states (when concentrations do not change over time, i.e. $c_i(t) = const.$) and from linear stability analysis.

Convergence proof

Steady state $c^*_{\mathrm{S}_i}$ of a node ν_i (which is found by setting the left hand side of the ODE to zero) is a unique solution as the system is described by a first-order, linear differential equation:

$$c^*_{\mathrm{S},i} = \frac{\sum_{j \in \mathcal{N}_i} c^*_{\mathrm{S},j}}{|\mathcal{N}_i|}. \tag{4.14}$$

We observe that if ν_i knows the actual number $|\mathcal{N}_i|$ of neighbors, the system is closed; the total amount of S-molecules reflects the sum of initial measurements

in all nodes: $\sum_{i\in\mathcal{V}} c_{\mathrm{S},i}(t) = \sum_{i\in\mathcal{V}} c_{\mathrm{S},i}(0) = c_T$. We now assume that molecules are evenly distributed among all nodes at equilibrium and use the ansatz $c^*_{\mathrm{S},i} = c_I \quad (\forall\ \nu_i \in \mathcal{V})$ in (4.14). In this way, we obtain $c^*_{\mathrm{S},i} = (\sum_{j\in\mathcal{N}_i} c_I)/|\mathcal{N}_i| = c_I$, which confirms our hypothesis and validates the ansatz. Finally, by collecting the above facts together, we obtain

$$c_I = c^*_{\mathrm{S},i} = \frac{c_T}{|\mathcal{V}|} = \frac{\sum_{i\in\mathcal{V}} c_{\mathrm{S},i}(0)}{|\mathcal{V}|},$$

which describes exactly the goal (4.3) of our distributed algorithm: each node converges to the average of the initial values.

Stability proof

The equilibrium point is stable as the Jacobian matrix $\mathbf{J}$, which reflects the linearization of the system, evaluated at the equilibrium point, has all negative eigenvalues:

$$\mathbf{J}(\mathbf{c})|_{\mathbf{c}^*} = \left[\frac{\partial \dot{c}_{\mathrm{S},i}}{\partial c_{\mathrm{S},k}}\right]\Bigg|_{\mathbf{c}^*} = -\mathbf{L}(\mathcal{G}), \tag{4.15}$$

where $\mathbf{L}$ is the Laplacian matrix of the graph $\mathcal{G}$, which represents the difference of the diagonal matrix of vertex degree and the adjacency matrix $\mathbf{\Delta}$. As the graph is supposed symmetric, the Laplacian matrix $\mathbf{L}$ has all positive eigenvalues [206].

On the other hand, by recalling that $[\mathbf{\Delta}]_{i,j} = 1$ for any $j \in \mathcal{N}_i$ and observing that

$$|\mathcal{N}_i| = \sum_{j\in\mathcal{N}_i} [\mathbf{\Delta}]_{i,j}, \tag{4.16}$$

we may rewrite (4.12) as $\dot{c}_{\mathrm{S}_i}(t) = \sum_{j\in\mathcal{N}_i} [\mathbf{\Delta}]_{i,j}\,(c_{\mathrm{S},j}(t) - c_{\mathrm{S},i}(t))$ or, equivalently, in matrix form

$$\dot{\mathbf{c}}_{\mathrm{S}}(t) = -\mathbf{L}(\mathcal{G})\mathbf{c}_{\mathrm{S}}(t).$$

The latter is exactly in the same form as the mathematical model proposed in [184], where Olfati and Murray proved the convergence towards the average of the initial measurements as formulated in (4.3), when $\mathcal{G}$ is a strongly connected and balanced digraph. Differently from [184], in which the above fluid model tries to approximate the algorithm dynamics, the set of ODEs in (4.12) is automatically extracted from the reaction network that defines the interactions among sensors. Mimicking sensor interactions in WSNs through $\mathcal{DAC}$s has given us the tools to derive a mathematical model whose convergence to the average is guaranteed in the investigated scenario.

4.4.1.2 Deficiency Zero Theorem

We can prove the convergence and stability of the model by just considering the chemical network topology. To do this, we take advantage of the Deficiency Zero Theorem, already introduced in Section 2.3.5.

The (whole) reaction network arising from (4.13) is weakly reversible: According to the definition given previously in Section 2.3.5, there exists a directed pathway from each member of a linkage class to all other members of the linkage class. To see how this comes about, observe that all species of the reaction network are complexes. Moreover, they do also correspond to the graph vertices of the communication network. This means that in strongly connected graphs for every reaction leading from complex C_i to complex C_j there exists a chain of reactions leading from C_j to C_i. In addition, in the emergent reaction network, the amount of molecules within the system remains constant. From (4.13), it follows that the number of complexes and of network vertices is the same, i.e. $|\mathcal{C}| = |\mathcal{V}|$, and it can easily be proven that the number of linkage classes is $\ell = 1$ since each complex is connected directly or indirectly to any other complex constituting the whole digraph. Moreover, a strongly connected chemical reaction network, where every chemical species appears in precisely one complex, has a stoichiometric matrix with $\text{rank}(\mathbf{U}) = |\mathcal{V}|-1$ (refer to [144] for further details). Collecting the above facts together, we have that the deficiency of the reaction network associated to (4.13) is $\wp = 0$. According to the Deficiency Zero Theorem [80], if the reaction network is weakly reversible and has a null deficiency value then it has a single, asymptotically stable fixed point. As we have shown, by setting $\dot{c}_{\mathrm{S}_i}(t) = 0$ and studying the equilibrium solution, it follows that the fixed point is defined by the right-hand side of (4.3). This is yet another prove that the proposed chemical algorithm converges to the average of initial measurements. To our knowledge, this is the first time that Deficiency Zero Theorem is used for proving the convergence of the Disperser model as well as of consensus algorithms, in general.

4.4.2 Sensitivity analysis and performance

We still do not know how (fast) nodes converge to the average value. To this end, we first study the response of a single sensor node through its sensitivity to internal and external perturbations. Then, we describe the dynamics of the whole network. Finally, we give the closed-form description for two significant network topologies: complete and ring networks.

As a first step, we have to identify the inputs of the sensor and rewrite (4.12) in a notation that is convenient to the sensitivity analysis. To do that, we have to better clarify how the dynamical system ($\mathcal{AC}$) is interfaced with the outside: The amount of S-molecules is changed proportionally to the variation of the measured-quantity. Namely, from the quantity $z_i[nT_s]$, locally measured every T_s seconds, we derive the quantity $\delta_{z,i} := z_i[nT_s] - z_i[(n-1)T_s]$; every T_s seconds, we increase or decrease $c_{\mathrm{S},i}$ by the amount $\delta_{z,i}$. As another input to ν_i, the execution of the $|\mathcal{N}_i|$ remote reactions $r_{B,j} : \mathrm{S}_j \xrightarrow{1} \mathrm{S}_i + \sum_{\iota\in\mathcal{N}_j,\iota\neq i} \mathrm{S}_\iota$ in its neighbors $\nu_j \in \mathcal{N}_i$ produces molecules S_i in node ν_i at a rate we have previously defined in (4.9) as $v_{\mathrm{rec},i}$.

Sensitivity analysis inputs

At this point, if we consider sensors initialized to zero, we can re-describe the

time evolution of S-concentration in node ν_i in the Laplace domain as

$$s \cdot C_{\mathrm{S},i}(s) - 0 = v_{\mathrm{rec},i}(s) - |\mathcal{N}_i| C_{\mathrm{S},i}(s) + \delta_{z,i}(s) \tag{4.17}$$

where we denote as $\delta_{z,i}(s)$ the input signal in the Laplace-domain. From (4.17) we derive the TF of a sensor ν_i that describes its sensitivity to variations of the reception rate $v_{\mathrm{rec},i}$ and to variations of the locally sensed value $\delta_{z,i}$:[4.7]

Node's sensitivity

$$\mathrm{H}_{\mathrm{rec},i}(s) = \mathrm{H}_{\mathrm{sen},i}(s) = \frac{1}{s + |\mathcal{N}_i|}. \tag{4.18}$$

Observe that the sensitivity of node ν_i to a variation of the locally-measured quantity z_i must be studied as an impulse response to (4.18).
The time the system needs to adapt depends on the network topology and its size. In case we would like to study the system response (i.e., sensitivity) to variations of $|\mathcal{N}_i|$ within the convergence period, the analytical method still remains the same: According to the sensitivity analysis introduced in Chapter 2, we would just have to define $|\mathcal{N}_i|$ as analysis input and study the node's sensitivity to this additional input.
Next, we use the above result to characterize the transient behavior of nodes in complete and ring networks (see Figure 4.4). The former provides the best scenario in terms of convergence time whereas the latter provides the worst one [213]. The transient behavior for any networks can be derived still following the same approach.

4.4.2.1 Complete topology

A complete topology describes a network where all nodes can communicate with each other; Figure 4.4(a) shows the topology of such a network. This kind of network represents the best scenario where nodes exponentially adapt their rates, and thus their average estimates, with the shortest convergence time.
The whole system can be described by using (4.18). We first write for each node ν_i its sensitivity to variations (i) of the locally-sensed value z_i, and (ii) of the reception rate $v_{\mathrm{rec},i}$ that reflects the sum of the variation of the data $\delta_{z,j}$ sensed by its neighbors $\nu_j \in \mathcal{V} | \nu_j \neq \nu_i$:

$$C_{\mathrm{S},i}(s) = \mathrm{H}_{\mathrm{sen},i}(s) \cdot \delta_{z,i}(s) + \mathrm{H}_{\mathrm{rec},i}(s) \cdot \sum_{j \in \mathcal{N}_i} \delta_{z,j}(s). \tag{4.19}$$

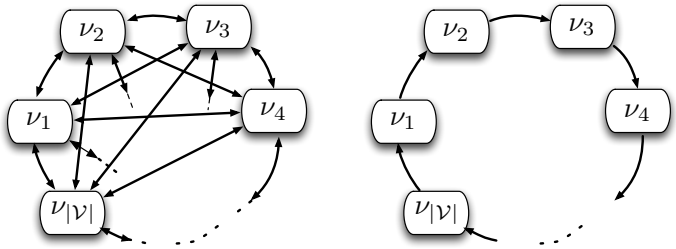
By solving the equation set in (4.19) we get the sensitivity of a node, showing us how a node ν_i exactly reaches consensus in a complete network:

Node's sensitivity in a complete net.

$$C_{\mathrm{S},i}(s) = \frac{1}{s(s + |\mathcal{V}|)} \left((s+1)\delta_{z,i}(s) + \sum_{j \in \mathcal{V}, j \neq i} \delta_{z,j}(s) \right). \tag{4.20}$$

Again, the sensitivity of node ν_i to a variation of the locally-measured quan-

[4.7] Quantities v_{rec} and δz represent inputs. Thus, the two TFs are defined as $\mathrm{H}_{\mathrm{rec},i}(s) = \frac{C_{\mathrm{S},i}(s)}{v_{\mathrm{rec},i}(s)}$ and $\mathrm{H}_{\mathrm{sen},i}(s) = \frac{C_{\mathrm{S},i}(s)}{\delta_{z,i}(s)}$.



(a) Complete network (b) Ring network

Figure 4.4: *Two network topologies representing (a) the best-case scenario and (b) the worst-case scenario.*

tity z_i must be studied as an impulse response to $H(s) = \frac{s+1}{s(s+|\mathcal{V}|)}$, whereas the sensitivity of node ν_i to a variation of quantity z_k measured at a generic node $\nu_k \in \mathcal{V} \mid \nu_k \neq \nu_i$ must be studied as an impulse response to $H(s) = \frac{1}{s(s+|\mathcal{V}|)}$.
Two things are remarkable: (*i*) No matter which neighbor experiences a variation of the sensed data, the sensitivity of the node is the same (same convergence time too). (*ii*) The difference between the new sensed value and the current estimated average does not influence the time response of the sensor (as a direct consequence of the LoMA, the time the sensor needs to reach consensus is independent of the magnitude of sensed data variation).

4.4.2.2 Ring topology

A ring network is constituted by nodes that can transmit to and receive from one neighbor only; Figure 4.4(b) shows the topology of such a network. This kind of network represents the worst scenario where nodes are poorly connected and thus, the required convergence time becomes very high as soon as the number of participating nodes increases.
To describe the system, again, we first write for each node ν_i its sensitivity to variations of the locally-sensed value z_i, and of the reception rate $v_{\text{rec},i}$, which reflects the variation of the data sensed by its neighbor:

$$C_{\text{S},i}(s) = \text{H}_{\text{sen},i}(s) \cdot \delta_{z,i}(s) + \text{H}_{\text{rec},i}(s) \cdot \delta_{z,(i-1)}(s). \tag{4.21}$$

We then solve the set of $|\mathcal{V}|$ equations generalized in (4.21) and formalize the sensitivity of a single node ν_i to variations of the locally sensed data z_i and of the data z_j sensed by all the other nodes $\nu_j = 1, \ldots, |\mathcal{V}| - 1, \neq i$:

$$C_{\text{S},i}(s) = \frac{(s+1)^{(|\mathcal{V}|-1)}}{(s+1)^{|\mathcal{V}|} - 1} \left(\delta_{z,i}(s) + \sum_{j \in \mathcal{V}, j \neq i} \frac{1}{(s+1)^{(|\mathcal{V}|-|i-j|)}} \delta_{z,j}(s) \right). \tag{4.22}$$

Node's sensitivity in a ring network

As expected, (4.22) indicates that the node's response depends on how far, in terms of hops to neighbors, the variation of sensed data is. Differently from a complete network where the distance is always one hop, in a ring network the distance may range from 1 to $|\mathcal{V}|-1$ hops. In addition, (4.22) reveals that the sensitivity of each sensor in a ring network highly depends on the total number of sensors. Local estimates of the average follow an under-dampened trajectory which results in (*i*) a fast exponential adaptation to a new sensed value to which is added (*ii*) a dampened oscillation around the actual average whose amplitude reduces to a negligible value in a finite time.
From the analytical description of the sensors' response/sensitivity in ring and complete networks, we can extract the convergence time characterizing these boundary cases. We use (4.20) and (4.22) to validate the convergence time experienced in simulations (later reported in Table 4.1).

Generalized sensitivity

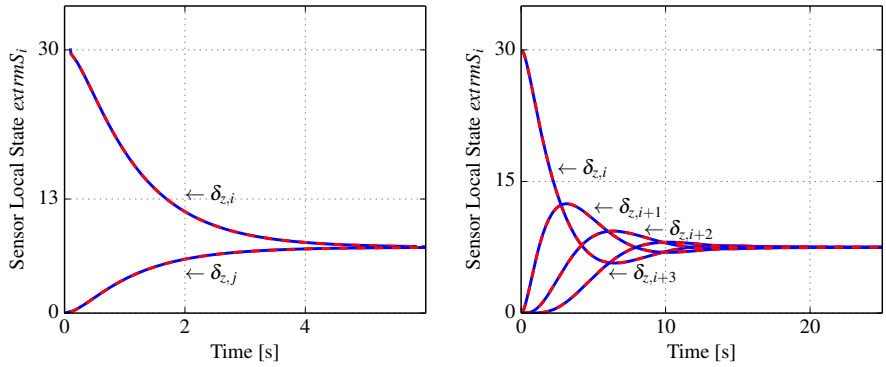
We can generalize the above findings concerning the transient response of sensors and define $\mathbf{H}(s)$ as the transfer function in the Laplace domain given by $\mathbf{H}(s) = (s\mathbf{I}_{\mathcal{V}} + \mathbf{L}(\mathcal{G}))^{-1}$ with $\mathbf{I}_{\mathcal{V}}$ being the identity matrix of order $|\mathcal{V}|$. The graph-related matrix $\mathbf{L}(\mathcal{G})$ (Laplacian of the graph, already introduced in (4.15)) plays a crucial role in the performance of consensus algorithms. Olfati-Saber and Murray proved in [184] that the second smallest eigenvalue λ_2 of graph Laplacians, called "algebraic connectivity" and also referred to as "Fiedler eigenvalue", quantifies the speed of convergence of consensus algorithms. For dense graphs, where λ_2 is relatively large, the consensus problem is solved faster than for sparse graphs, where λ_2 is instead relatively small. (We investigate this aspect through simulations in Section 4.5.2.)

4.5 Simulating in OMNeT++ the chemical consensus algorithm

We validated the derived chemical consensus model and related findings via simulations. We compare in Section 4.5.1 the trajectory experienced in simulations in OMNeT-environment and that analytically predicted with findings of the previous section. We report results obtained by simulating the consensus model in Figure 4.3 under different operating conditions, such as different network topologies, number $|\mathcal{V}|$ of sensors (Section 4.5.2), and time-varying measurements (Section 4.5.3). Finally, we compare in Section 4.5.4 the chemical model with two known classes of consensus-algorithms: randomized [36] and broadcast [22] gossip algorithms.

4.5.1 Simulation measurements *vs.* analytical predictions

One of the benefits of the chemical approach is the simplicity in deriving a correct fluid model that describes the macro-behavior of the node and consequently of the whole networked system. To demonstrate this, we have compared the analytically-predicted time response with its execution behavior in the



(a) Complete network (b) Ring network

Figure 4.5: *Comparison between simulation measurements (blue-continuous line) and analytical predictions (red-dashed line) for 4-node networks.*

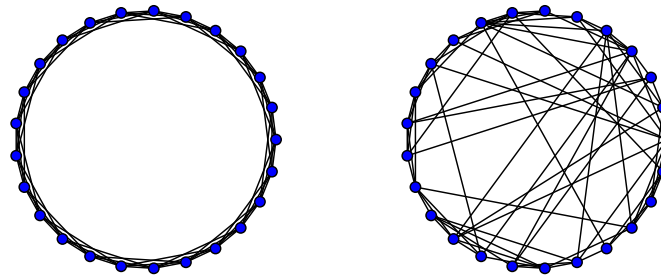
OMNeT++ simulator (we are still in an ideal scenario without communication delays and interferences). Specifically, we applied a variation on the value sensed by one of the sensors in a four-node network (for both complete and ring networks) and we sampled every 100 ms the state of each sensor. We then plot these results together with the impulse response to δ_z-input, calculated using (4.20) and (4.22) for complete and ring networks, respectively.

Figure 4.5(a) shows the response in a complete network: Starting from steady state, we suddenly changed the locally sensed quantity of one of the four sensors. As a consequence, all sensors smoothly adapted their estimates to the new attractor, exactly conforming to our analytical prediction (red-dashed line) – the "perturbed" sensor node exponentially decreased its state, whereas the other three non-perturbed sensors exponentially increased their state to reach the new average value.

Figure 4.5(b) shows the response in a ring network: This time, as a consequence of the stimuli of the locally sensed quantity of one of the four sensors, all sensors adapted their state following an "under-dampened trend". Also in this case, our transient analysis exactly describes the behavior of the whole network – an under-dampened trajectory made of a fast exponential adaptation to the new sensed value to which is added a dampened oscillation around the actual average.

4.5.2 Simulations for different topologies and sizes of the communication network

We tested the convergence of the sensor's state towards the arithmetic mean for different network topologies. In the first set of experiments, we studied

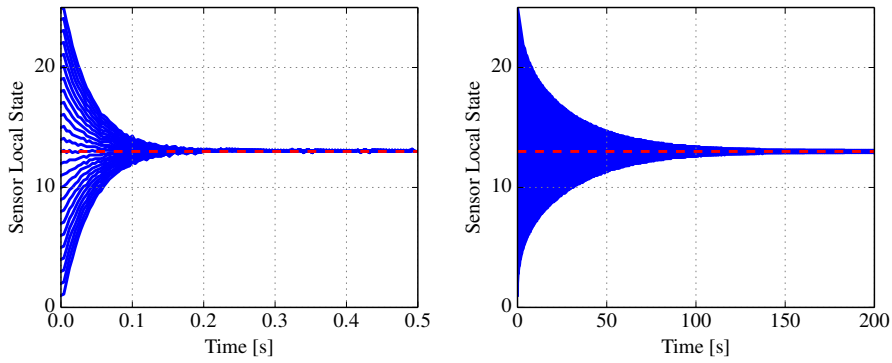


(a) Regular-lattice network (b) Small-world network

Figure 4.6: *Two network topologies having the same number of nodes ($|\mathcal{V}| = 25$) and links ($|\mathcal{E}| = 75$) but different algebraic connectivity, (a) $\lambda_2 \sim 0.85$ and (b) $\lambda_2 \sim 2$.*

relatively small networks made of $|\mathcal{V}| = 25$ nodes. Each node had a different initial state, equal to $z_i = i$ for $i = 1, 2, \ldots, |\mathcal{V}|$.
We studied a complete (Figure 4.4(a)) and a ring (Figure 4.4(b)) network, which we know represent the best and worst scenarios in terms of convergence speed; results are reported in Figure 4.7(a) and Figure 4.7(b), respectively. We also studied a regular lattice network with interconnections to $K = 3$ nearest neighbors (Figure 4.6(a)), and a small-world network with $3|\mathcal{V}| = 75$ links (Figure 4.6(b)). These two topologies are very interesting in terms of convergence speed and network connectivity.
Indeed, Olfati-Saber demonstrated in [181] that it is possible to greatly increase the algebraic connectivity in regular complex networks, without adding new links or nodes, rather by using the random rewiring scheme proposed by Watts and Strogatz. In [244], Watts and Strogatz showed the small-world phenomenon (popularly known as six degrees of separation) by starting with regular ring lattice networks with $|\mathcal{V}|$ vertices and K edges per vertex (equal for all vertices $\nu_i \in \mathcal{V}$) and then rewiring each edge with probability p. Olfati-Saber showed that the consensus problem can be solved more quickly on certain "small-world networks".[4.8] The results of the performance in the regular lattice and the small-world network are reported in Figure 4.7(c) and Figure 4.7(d), respectively. The results in Figure 4.7 are in line with those in [180]: compared to the other networks, small-world networks exhibit shorter convergence times,

[4.8] In [181], Olfati-Saber also stated that a network with relatively high algebraic connectivity is necessarily robust against both node-failures and edge-failures. However, in [43], Byrne *et al.* showed that this holds true only for systems where nodes are redundant. On the contrary, in systems where nodes can be indispensable, a higher algebraic connectivity does not necessarily mean more robustness. Instead, concepts like node- and edge-connectivity are important parameters for assessing the robustness.



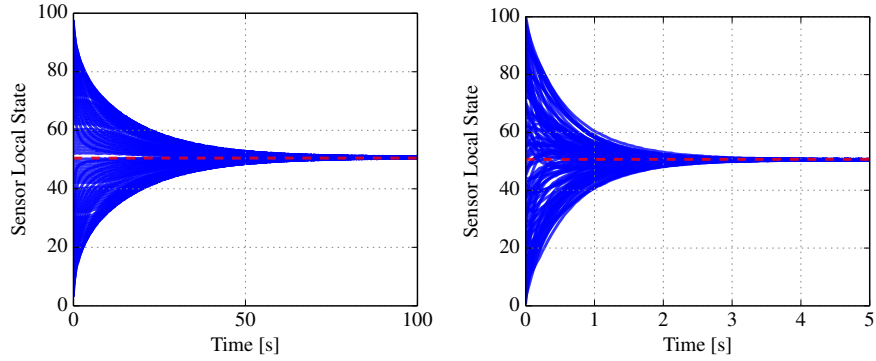
(a) Complete network

(b) Ring network

(c) Regular lattice $K = 3$

(d) Small-world network

Figure 4.7: *Sensors' state evolution obtained when $|\mathcal{V}| = 25$ sensors were connected through (a) a complete network, (b) a ring network, (c) a regular lattice network topology with interconnections to $K = 3$ nearest neighbors, and (d) a small-world network topology with $3|\mathcal{V}|$ links (see [180] and references therein for more details on such networks). The initial state was set to $z_i = i$ for $i = 1, 2, \ldots, |\mathcal{V}|$.*



(a) Regular lattice $K = 3$ (b) Small-world network

Figure 4.8: *Sensors' state evolution obtained when* $|\mathcal{V}| = 100$ *sensors were connected through (a) a regular lattice network topology with interconnections to* $K = 3$ *nearest neighbors and (b) a small-world network topology with* $3|\mathcal{V}|$ *links. The initial state was set to* $z_i = i$ *for* $i = 1, 2, \ldots, |\mathcal{V}|$.

while still keeping the number of links reasonably low. This occurs thanks to the high algebraic connectivity of the associated digraph $\mathcal{G}$. The second smallest eigenvalue λ_2 for the used complete, ring, regular lattice, and small-world networks was 25, 0.0314, 0.8523, and 2.0269, respectively.

Same dynamics as the traditional model

To ease the comparison of the chemical algorithm's dynamics with those of the traditional theoretical model in [180], we also report in Figure 4.8 how $|\mathcal{V}| = 100$ sensors converge to the average value, when they are connected through a regular lattice and a small-world network (similar wiring procedure as previously described). These charts are directly comparable with the ones in Figure 4 in the paper [180], and the results fully comply with those obtained through numerical integration of the traditional ODE-model. Figure 4.8 demonstrates the convergence in the same operating conditions of those used to obtain the chart in Figure 4.7 for a network with $|\mathcal{V}| = 100$ nodes. As before, the convergence time of a regular lattice network is shown to be significantly larger than that of a small-world.

Differently from works such as the one in [180], we do not refer to "simulations" as the numerical integration/solution of ODEs in (4.12). Rather, in the here-presented simulations, we let the dynamical system of each node ν_i operate according to the $\mathcal{DAC}$ defined in (4.11) and exchange messages (molecules) accordingly.

In order to get a wider view on the effects that the size and the topology of the network have on the speed of the chemical consensus algorithm, we did many further experiments. Here, we limit to report in Table 4.1 on the convergence times required to reach a normalized mean squared error equal to 0.01 for some noteworthy tests. Differently from previous experiments, the initial state

Network with $\lvert\mathcal{V}\rvert =$	25	50	100	250	1000
Ring	80 s	230 s	450 s	900 s	1250 s
Regular lattice, $K = 3$	4 s	12 s	35 s	90 s	180 s
Small-world, $3M$ links	1 s	1 s	1 s	1 s	1 s
Complete	0.26 s	0.09 s	0.04 s	0.012 s	0.005 s

Table 4.1: *Convergence times for achieving a normalized mean squared error less than* 0.01 *in different network topologies. The initial state was set to* $z_1 = 60$ *and* $z_i = 30$ *with* $i = 2, 3, \ldots, |\mathcal{V}|$.

of nodes was set to $z_1 = 60$ and $z_i = 30$ with $i = 2, 3, \ldots, |\mathcal{V}|$. From the results reported in Table 4.1, we can see that the convergence time of a complete network decreases substantially as $|\mathcal{V}|$ becomes larger due to the exponential increase of the number of connected links. The opposite happens for a ring network as, in this case, the information is exchanged in a serial manner and thus, the time required to exchange information among all nodes highly grows with increasing number of nodes. On the other hand, the convergence time remains constant for small-world networks.

Scalability for different network topologies

4.5.3 Simulations for time-varying measurements

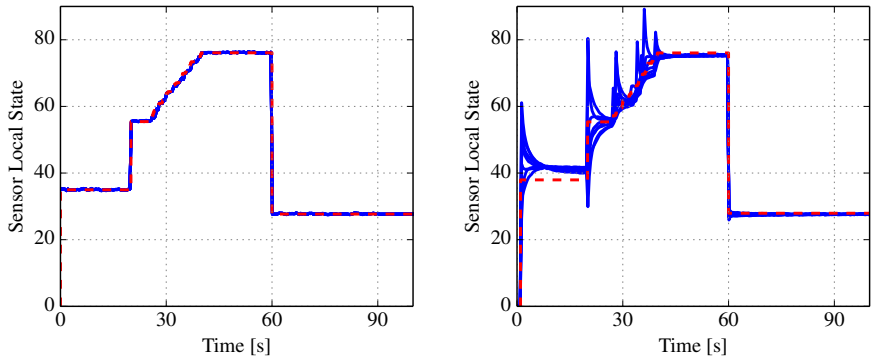
We experimented in a more dynamical scenario, where nodes measure time-varying quantities. We report here on the ability of the chemical consensus model to track quantities that vary also with shorter times than the period required for the convergence. We focus on complete and ring networks only (further experiment results are reported in Section A.7).

Figure 4.9 illustrates sensors' state evolution when the network was composed of $|\mathcal{V}| = 30$ nodes and was characterized either by a complete or a ring topology. To validate the convergence of the algorithm in presence of measurement changes, the quantities z_i were randomly generated (at certain time instants) according to a uniform distribution with given intervals: at time $t = 0$, 20, and 60 (in seconds) the sensor measured independent random variables chosen uniformly within the intervals $[5, 75]$, $[50, 60]$ and $[25, 35]$, respectively; during the time interval between $[25, 40]$ seconds, each sensor independently chose a time instant to change its measurement, which was a uniform random variable within the interval $[75, 77]$ (z_i-dynamics are summarized in Table 4.2).

From the experimental results in Figure 4.9, it follows that the convergence to each new value of the arithmetic mean is guaranteed for both network topologies, although this holds true only if the measured quantities vary sufficiently slow compared to the convergence time. Due to the different algebraic connectivity, the convergence is achieved almost instantaneously in the complete network, whereas a longer time interval is required in the ring network.

Time instant t [seconds]	0	20	$\mathcal{U}[25, 40]$	60
Value of z_i	$\mathcal{U}[5, 75]$	$\mathcal{U}[50, 60]$	$\mathcal{U}[75, 77]$	$\mathcal{U}[25, 35]$

Table 4.2: *Local data z_i detected by node ν_i at time t during the experiment whose results are plotted in Figure 4.9.*



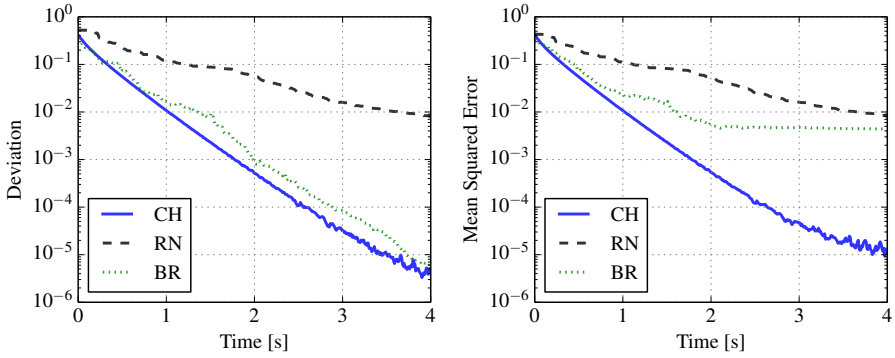
(a) Complete network (b) Ring network

Figure 4.9: *Sensors' state evolution when $|\mathcal{V}| = 30$ sensors were connected through (a) a complete and (b) a ring network topology. Sensors' measurements varied as described in Table 4.2. The analytical mean is reported as a red-dashed line.*

4.5.4 Comparisons with randomized and broadcast gossip

We made comparisons with the two following consensus gossip-based algorithms: the randomized (RN) solution proposed in [36] and the broadcast (BR) one illustrated in [22]. Both algorithms were simulated according to the asynchronous model described in [36] and [22], whereby each node is assumed to have a clock that ticks independently according to a Poisson process's rate γ. This corresponds to a single global clock whose ticking times form a Poisson process of rate $|\mathcal{V}| \cdot \gamma$ [22]. In all subsequent simulations, we set an average of $\gamma = 2$ ticks per second in each node. When RN is used, at each tick, node ν_i randomly interacts with a single nearby sensor. On the other hand, when BR is applied, node ν_i wirelessly broadcasts its current state value.

Comparisons are made in terms of the normalized deviation of sensors' states from their average and in terms of the mean squared error with respect to the average of their initial states (measured quantities z_i). For this purpose, we considered the same operating conditions of those used to obtain the chart in Figure 4.8(b): a small world network with $|\mathcal{V}| = 100$ nodes and $3|\mathcal{V}| = 300$ links in which the initial states were set to $z_i(0) = i$ for $i = 1, 2, \ldots, |\mathcal{V}|$. This means that, on average in the network, $2 \cdot |\mathcal{V}| \cdot \gamma = 400$ packets per second



(a) Deviation

(b) Mean squared error

Figure 4.10: *Performance comparison among the chemical (CH) consensus algorithm, the broadcast (BR) and randomized (RN) gossip algorithms. Initial state and network setup were like those related to Figure 4.8(b).*

are exchanged when RN is used, and $|\mathcal{V}| \cdot \gamma = 200$ packets per second are transmitted when BR is used.
The results of Figure 4.10(a) show that the convergence time of the proposed chemical (CH) algorithm is similar to that experienced with the BR algorithm. As seen, both largely outperform RN, which does not take advantage of the broadcast nature of the wireless medium. On the other hand, Figure 4.10(b) shows that the estimation accuracy of the CH algorithm is higher than that of the BR algorithm. This difference is due to the bias term that the BR algorithm introduces in the average estimation. Refer to [214] for further detail and for a specific solution to improve the estimation accuracy based on the knowledge of the number of neighbors. The higher accuracy of the solution proposed in [214] is achieved at the price of a higher convergence time and a more complex communication model that requires the transmission and the processing of a "companion" variable, in addition to the node's state.
In summary, the results of Figure 4.10 show that the proposed CH algorithm allows one to achieve a good trade-off between convergence time and estimation accuracy.

4.6 A chemical approach to practical issues

The basic chemical consensus model can be extended to overcome some practical issues. We derive a consensus algorithm that exhibits self-configuration (Section 4.6.1) and self-stabilization (Section 4.6.2) properties, and then again validate the derived algorithm and compare it with existing solutions in Section 4.6.3.

4.6.1 Self-configuration: Estimating the neighborhood size

The execution of the draining reaction $r_{D,i}$ in (4.10) requires the knowledge about the number of neighbors $|\mathcal{N}_i|$. However, this knowledge is hardly available at each node, especially in those applications in which nodes appear and disappear over time (switching networks). To address this issue, we start rewriting (4.7) and (4.10) as follows:

$$r_{B',i}: \quad \mathrm{S}_i \xrightarrow{1} \sum_{j \in \mathcal{N}_i} \mathrm{S}_j + \mathrm{S}_i \tag{4.23}$$

$$r_{D',i}: \quad \mathrm{S}_i \xrightarrow{|\mathcal{N}_i|} \emptyset. \tag{4.24}$$

The above operation has no effect on the dynamics of the $\mathcal{DAC}_i$ (equal set of ODEs as in (4.12)) and it is only used to make the system depend on $|\mathcal{N}_i|$ rather than $|\mathcal{N}_i| - 1$.[4.9] To proceed further, we let each node ν_i define two molecular species X_i and Y_i, with X_i characterized by a constant concentration equal to α, i.e., $c_{\mathrm{X}_i}(t) \equiv \alpha$. Then, we define the following reactions:

Estimation of the runtime number of neighbors

$$r_{X,i}: \quad \mathrm{X}_i \xrightarrow{1} \sum_{j \in \mathcal{N}_i} \mathrm{Y}_j + \mathrm{X}_i \tag{4.25a}$$

$$r_{Y,i}: \quad \mathrm{Y}_i \xrightarrow{1} \emptyset. \tag{4.25b}$$

Applying the mass-action principle according to (2.7) and recalling that $c_{\mathrm{X}_i}(t) = \alpha$ yields

$$\dot{c}_{\mathrm{Y}_i}(t) = \alpha|\mathcal{N}_i| - c_{\mathrm{Y}_i}(t)$$

from which it follows that at the equilibrium (i.e., $\dot{c}^*_{\mathrm{Y}_i} = 0$) the abundance of Y-molecules at each node is $\alpha|\mathcal{N}_i|$:

$$c^*_{\mathrm{Y}_i} = \alpha|\mathcal{N}_i|. \tag{4.26}$$

Then, replacing the draining reaction $r_{D',i}$ with

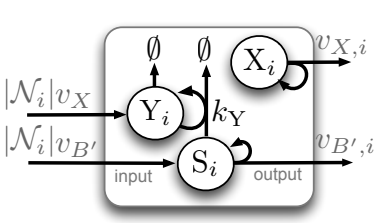
$$r_{D'',i}: \quad \mathrm{S}_i + \mathrm{Y}_i \xrightarrow{1/\alpha} \mathrm{Y}_i \tag{4.27}$$

yields the following ODEs

$$\dot{c}_{\mathrm{S}_i}(t) = \sum_{j \in \mathcal{N}_i} c_{\mathrm{S}_j}(t) - \frac{1}{\alpha} c_{\mathrm{Y}_i}(t) c_{\mathrm{S},i}(t), \quad c_{\mathrm{S},i}(0) = z_i. \tag{4.28}$$

By assuming that the convergence time of the subsystem defined by (4.25a) and (4.25b) is smaller than that concerning by (4.23) and (4.27) and thus by

[4.9] This allows overcoming some implementation issues. First, computing the difference $|\mathcal{N}_i| - 1$ would require an additional set of reactions (see for example the motif proposed in [163] and reported in Appendix A.1.1). Second, using (4.23) and (4.24) allows maintaining the sensor state also in a single-node network.



(a) Reaction Network

$$
\begin{aligned}
r_{B',j}: &\quad \mathrm{S}_j \xrightarrow{1} \mathrm{S}_j + \mathrm{S}_i + \sum_{\substack{\iota\in\mathcal{N}_j\\ \iota\neq i}} \mathrm{S}_\iota \quad (j=1,..,|\mathcal{N}_i|)\\
r_{B',i}: &\quad \mathrm{S}_i \xrightarrow{1} \mathrm{S}_i + \textstyle\sum_{j\in\mathcal{N}_i} \mathrm{S}_j\\
r_{D'',i}: &\quad \mathrm{S}_i + \mathrm{Y}_i \xrightarrow{k_Y} \mathrm{Y}_i\\
r_{X,i}: &\quad \mathrm{X}_i \xrightarrow{1} \mathrm{X}_i + \textstyle\sum_{j\in\mathcal{N}_i} \mathrm{Y}_j :\\
r_{Y,i}: &\quad \mathrm{Y}_i \xrightarrow{1} \emptyset\\
r_{X,j}: &\quad \mathrm{X}_j \xrightarrow{1} \mathrm{X}_j + \mathrm{X}_i + \sum_{\substack{\iota\in\mathcal{N}_j\\ \iota\neq i}} \mathrm{X}_\iota \quad (j=1,..,|\mathcal{N}_i|)
\end{aligned}
$$

(b) Reaction Equations

Figure 4.11: *Chemical model for estimating the runtime number of neighbors. Species X is initialized to a value α, the same for all nodes $\nu \in \mathcal{V}$. Coefficient of reaction r_Y is set to $k_Y = 1/\alpha$.*

substituting (4.26) in (4.28), we get the following result

$$\dot{c}_{\mathrm{S}_i}(t) = \sum_{j\in\mathcal{N}_i} c_{\mathrm{S}_j}(t) - |\mathcal{N}_i| c_{\mathrm{S},i}(t), \quad c_{\mathrm{S},i}(0) = z_i. \tag{4.29}$$

Equation (4.29) is in the same form as (4.12) but has been obtained without knowing $|\mathcal{N}_i|$, simply by using the reactions (4.25) and modifying the draining reaction to (4.27). Figure 4.11 summarizes the extended $\mathcal{DAC}$ which guarantees each node to converge to the average consensus value when estimating the runtime number of neighbors. The above results hold true only if (4.25a) and (4.25b) reach the equilibrium before (4.23) and (4.27). This is reasonable for low-mobility applications but can be achieved in general by properly setting the design parameter α, which dictates the rate of execution of (4.25a). The higher the α-coefficient is, the faster the convergence is.
In those applications where $|\mathcal{N}_i|$ remains constant for a long time interval, its value can easily be estimated through the $\mathcal{DAC}_i$ defined in (4.11) (with no need for additional reactions): During an initialization phase, the sensors' state should be forced to a constant pre-defined value α (i.e., $c_{\mathrm{S},i}(t) \equiv \alpha$). The execution of $r_{B,i}$ would induce the production of S-molecules in nearby sensors with an average rate $|\mathcal{N}_i|$ times bigger than the pre-defined value α (this easily follows recalling the broadcast nature of reaction $r_{B,i}$). Therefore, an estimate of the number of neighbors $|\mathcal{N}_i|$ could easily be obtained by comparing the measured reception rate and the predefined one.

4.6.2 Self-stabilization: Recovering from erroneous states

Self-stabilization, introduced by Dijkstra in his seminal paper [65], is a concept of fault-tolerance in distributed computing. Fault-tolerance aims to guarantee that the system always remains in a correct state, under certain state transitions. However, this cannot always be achieved, such as for example when the

system starts from an incorrect state or is corrupted by errors. On the contrary, self-stabilization relaxes fault-tolerance, and identifies all systems that end up in a correct state, independently of their initialization, after a finite time or number of iterations. Namely, self-stabilization is the ability to cope with faults that were not foreseen in the design of the algorithm.

In a closed system, such as in the Disperser, molecules do not leave the reaction vessel; there is no "dilution flow"; reaction rules must be balanced. Furthermore, in a closed system, a stable solution is a set of molecules, which do not react further. One could believe that such a system is more stable than an open system where instead, mass and energy can be exchanged with its environment. Additionally, in a closed system there are no concerns that information gets lost via dilution flows and on how to regenerate molecules. On the contrary, with an open system approach, we arrive at more robust and flexible organic systems, where a stable solution is a self-regenerating set of molecules [68]. Open reaction systems are especially suitable in unreliable and highly dynamic scenarios – in environments that are under constant change (nodes are added and removed at runtime). For example, if a node in a WSN is switched off, the molecules residing in it vanish too, causing in this way the so-called non-selective dilution flow. Thus, robust/stable structures must consist of molecules that constantly reproduce themselves as a whole; according to the theory of chemical organization [70], they must encompass a self-maintaining set of molecules. By introducing a selective dilution flow, we can move gradually from an open to a closed system and can capture aspects from both.

In our context, a WSN must be robust against possible perturbations, such as measurement errors or sensors leaving (entering) the network in advance (at a later stage). To this end, we have to further extend the $\mathcal{DAC}$ in (4.11). We let each node define a molecular species Z_i whose concentration is maintained constant and equal to the local measurement value, i.e., $c_{\mathrm{Z}_i}(t) = z_i$. Then, we introduce the two following reactions:

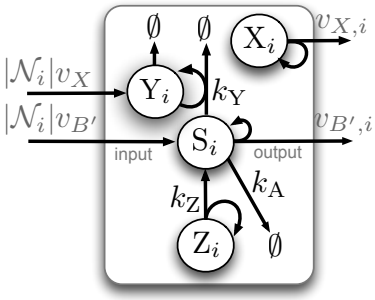
Self-stabilization

$$r_{Z,i}: \quad \mathrm{Z}_i \xrightarrow{\beta} \mathrm{S}_i + \mathrm{Z}_i \tag{4.30a}$$

$$r_{A,i}: \quad \mathrm{S}_i \xrightarrow{\beta} \emptyset \tag{4.30b}$$

with β being a design parameter. The execution of $r_{Z,i}$ continuously feeds S_i-species at a rate β-proportional to the local measurement z_i. At the same time, S_i-species is drained with the same proportionality coefficient β through reaction $r_{A,i}$. As a result, the sensors' state (the concentration of molecules S_i) is continuously refreshed and if an error occurs, after a transient time, the sensor's state goes back to the correct value. The higher the β-coefficient is, the faster the recovering is. However, as shown next by means of simulation results, this is achieved at the price of a reduced estimation accuracy. Figure 4.12 summarizes the extended $\mathcal{DAC}$ which guarantees each node to converge to the average consensus value, even in the presence of transient communication and measurements errors.

The effect of the control parameter β can be formalized following the same



(a) Reaction Network

$$
\begin{aligned}
r_{B',j}: &\quad \mathrm{S}_j \xrightarrow{1} \mathrm{S}_j + \mathrm{S}_i + \sum_{\substack{\iota \in \mathcal{N}_j \\ \iota \neq i}} \mathrm{S}_\iota \quad (j = 1, .., |\mathcal{N}_i|) \\
r_{B',i}: &\quad \mathrm{S}_i \xrightarrow{1} \mathrm{S}_i + \textstyle\sum_{j \in \mathcal{N}_i} \mathrm{S}_j \\
r_{D'',i}: &\quad \mathrm{S}_i + \mathrm{Y}_i \xrightarrow{k_Y} \mathrm{Y}_i \\
r_{X,i}: &\quad \mathrm{X}_i \xrightarrow{1} \mathrm{X}_i + \textstyle\sum_{j \in \mathcal{N}_i} \mathrm{Y}_j \\
r_{Y,i}: &\quad \mathrm{Y}_i \xrightarrow{1} \emptyset \\
r_{X,j}: &\quad \mathrm{X}_j \xrightarrow{1} \mathrm{X}_j + \mathrm{X}_i + \sum_{\substack{\iota \in \mathcal{N}_j \\ \iota \neq i}} \mathrm{X}_\iota \ (j = 1, .., |\mathcal{N}_i|) \\
r_{Z,i}: &\quad \mathrm{Z}_i \xrightarrow{k_Z} \mathrm{S}_i + \mathrm{Z}_i \\
r_{A,i}: &\quad \mathrm{S}_i \xrightarrow{k_A} \emptyset
\end{aligned}
$$

(b) Reaction Equations

Figure 4.12: *Self-stable chemical model which allows nodes to recover from erroneous states. Species Z is maintained equal to the local measurement value, i.e., $c_{Z_i}(t) = z_i$. The coefficient of reaction r_Y is set to $k_Y = 1/\alpha$. Coefficients of reactions r_Z and r_A are set to $k_Z = k_A = \beta$.*

steps shown in the sensitivity analysis in Section 4.4. The dynamics of the introduced $\mathcal{DAC}$ are described by the following ODE:

$$\dot{c}_{\mathrm{S}_i}(t) = \sum_{j \in \mathcal{N}_i} c_{\mathrm{S}_j}(t) - |\mathcal{N}_i| c_{\mathrm{S}_i}(t) + \beta \left(c_{\mathrm{Z},i} - c_{\mathrm{S}_i}(t) \right) \tag{4.31}$$

whose matrix form is given by

$$\dot{\mathbf{c}}_{\mathrm{S}}(t) = -\mathbf{L}\mathbf{c}_{\mathrm{S}}(t) + \beta \left(\mathbf{c}_{\mathrm{Z}} - \mathbf{c}_{\mathrm{S}}(t) \right). \tag{4.32}$$

Again, by rewriting (4.32) after taking the Laplace transform of both sides at a certain reference time t', we get

$$\mathbf{C}_{\mathrm{S}}(s) = \mathbf{H}(s)\big(\mathbf{c}_{\mathrm{S}}(t') + \beta \mathbf{c}_{\mathrm{Z}}\big) \tag{4.33}$$

where $\mathbf{H}(s)$ is the Laplacian transfer function given by

$$\mathbf{H}(s) = \left(s\mathbf{I}_{\mathcal{V}} + \mathbf{L} + \beta \mathbf{I}_{\mathcal{V}} \right)^{-1}, \tag{4.34}$$

with $\mathbf{I}_{\mathcal{V}}$ being the identity matrix of order $|\mathcal{V}|$. One can use $\mathbf{H}(s)$ to analytically evaluate how the β-coefficient must be chosen: a trade-off between convergence time and estimation accuracy – each node tends to converge *slower or faster* to a value that is *less or more* weighted by the local measurement.

We can explicitly characterize the behavior of the sensor ν_i to variations $\delta_{z,i}$ of the locally measured value and to variations $\delta_{z,\iota}$ on the value measured from

the other sensor nodes $\nu_\iota \in \mathcal{V}$ that constitute the network:

Higher β-values...

...stronger robustness + faster recovery

$$H_{\text{rec}}(s) = \frac{1}{s+\beta+|\mathcal{N}_i|} + \frac{\beta}{s(s+\beta+|\mathcal{N}_i|)} \tag{4.35a}$$

$$H_{\text{sen}}(s) = \frac{1}{s+\beta+|\mathcal{N}_i|}. \tag{4.35b}$$

From the above functions, we can see that the β-coefficient effectively regulates the effect of the correcting action: higher β-values announce a stronger robustness to errors and a faster recovery from erroneous state. Again, the number of neighbors $|\mathcal{N}_i|$ mainly affects how fast the node adapts to new sensed values. We can look once more at the two boundary scenarios. In a complete network, the Laplace transform of each node ν_i with respect to variations of the locally-sensed value $\delta_{z,i}$, and of the values $\delta_{z,j}$ sensed by neighboring nodes is

Node's sensitivity in complete net.

$$C_{\text{S},i}(s) = \frac{H_{\text{sen},i}(s)}{1+H_{\text{rec},i}(s)}\left(\frac{1-(|\mathcal{N}_i|-1)H_{\text{rec},i}(s)}{1-(|\mathcal{V}|-1)H_{\text{rec},i}(s)}\delta_{z,i}(s) + \right.$$
$$\left. + \frac{H_{\text{rec},i}(s)}{1-(|\mathcal{V}|-1)H_{\text{rec},i}(s)} \sum_{j=0,j\neq i}^{|\mathcal{V}|-1} \delta_{z,j}(s)\right). \tag{4.36}$$

For a ring network instead we have

Node's sensitivity in ring networks

$$C_{\text{S},i}(s) = \frac{H_{\text{sen},i}(s)}{1-H_{\text{rec},i}(s)^{|\mathcal{V}|}}\left(\delta_{z,i} + \sum_{j=0,j\neq i}^{|\mathcal{V}|-1} H_{\text{rec},i}(s)^{(|\mathcal{V}|-|j-i|)}\delta_{z,j}\right). \tag{4.37}$$

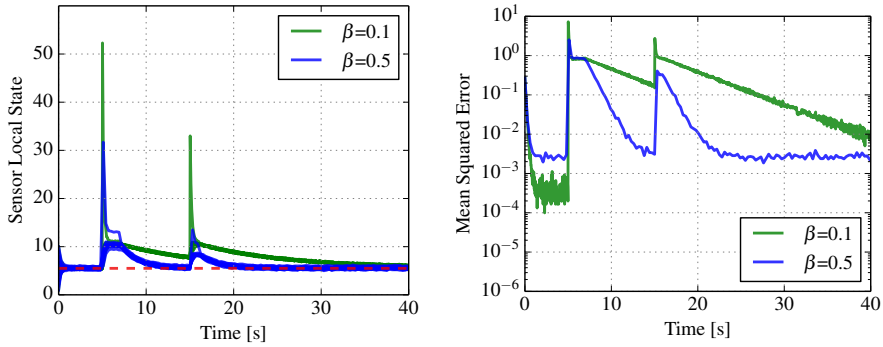
In the above analysis, we have again assumed that the average number of neighbors $|\mathcal{N}_i|$ does not change significantly over the time required by the algorithm to converge. In case we would like to study the system response (i.e., sensitivity) to variations of $|\mathcal{N}_i|$ ($= c_{\text{Y},i}$) within the convergence period, we would have to define also $c_{\text{Y},i}$ as analysis input and study the node's response to this additional input.

The $\mathcal{DAC}$ we have derived in this section has been used for experiments in simulations and real-world scenarios where conditions were not ideal (results reported in the following sections). The analytical results presented here were used to calibrate the $\mathcal{DAC}$ in such scenarios and are used in the following to compare the analytically predicted response and the actual measured trajectory.

4.6.3 Simulations and comparisons in non-ideal scenarios

We implemented the extend $\mathcal{DAC}$ in Figure 4.12 (illustrated in the previous section) and tested it in perturbed, dynamic scenarios. The dynamical system of each node was simply modified such as to operate according to the following formal definition:

$$\mathcal{DAC}'_i = \{\mathcal{M}'_i, \mathcal{R}'_i, \mathcal{A}\} \tag{4.38}$$



(a) Sensors' state evolution

(b) Mean squared error

Figure 4.13: *The effect of β-parameter on the recovery from perturbations in a small world network with $|\mathcal{V}| = 10$ nodes, $3|\mathcal{V}| = 30$ links. The initial state was $z_i = i$ for $i = 1, 2, \ldots, |\mathcal{V}|$. At $t = 5$ s, node ν_1 introduced an error of 50% for 2 seconds. At $t = 15$ s, node ν_4 introduced an error of 30% for 1 second. As usual, the analytical mean is reported as a red-dashed line.*

Final $\mathcal{DAC}$ for broadcast consensus

with $\mathcal{M}_i' = \mathcal{S}_i' \cup \mathcal{S}_i^{(j)'}$, $\mathcal{S}_i' = \{\mathrm{S}_i, \mathrm{X}_i, \mathrm{Y}_i, \mathrm{Z}_i\}$, $\mathcal{S}_i^{(j)'} = \{\mathrm{S}_j, \mathrm{Y}_j | \, j \in \mathcal{N}_i\}$ and

$$\mathcal{R}_i' = \{[\mathbf{r}_B']_j, r_{B',i}, r_{D'',i}, r_{X,i}, r_{Y,i}, [\mathbf{r}_X]_j, r_{Z,i}, r_{A,i}\}. \tag{4.39}$$

We first assess the impact of control parameter β. As argued before, the dimensioning of β represents a trade-off between robustness to perturbations and estimation accuracy. Figure 4.13 illustrates sensors' state evolution and mean squared error of CH algorithm when β was either 0.1 or 0.5 (we recall that $k_Z = k_A = \beta$ s^{-1}). The investigated setting was a small world network with $|\mathcal{V}| = 10$ nodes, $3|\mathcal{V}| = 30$ links and initial values equal to $z_i = i$ for $i = 1, 2, \ldots, |\mathcal{V}|$. The number of sensors was fixed to 10 since errors are expected to have a high influence in networks with a relatively small number of nodes. In this experiment, certain nodes were forced to exhibit transient problems in sensing or transmitting, and thus to introduce some perturbations in the chemical network: at time instant $t = 5$ s, node ν_1 introduced a measurement error of 50% for 2 seconds; at $t = 15$ s, node ν_4 introduced an error of 30% for 1 second. The results in Figure 4.13 show that a higher value of β allowed a faster recovery from perturbations at the expense of a lower estimation accuracy. As shown in Figure 4.13(b), the mean squared error during the first five seconds was less than 10^{-3} for $\beta = 0.1$ whereas it was higher than 10^{-3} for $\beta = 0.5$. Further simulations showed that $\beta = 0.1$ allows achieving a good tradeoff between the two conflicting requirements. For this reason, we set $\beta = 0.1$ in all simulations subsequently discussed. Nevertheless, different values of parameter β may be required in scenarios exhibiting different features (e.g., affected by

perturbations of different intensity) or for different application constraints (e.g., different constraints in terms of estimation accuracy).

Improved error resiliency

Figure 4.14 illustrates performance comparisons among CH, BR, and RN algorithms, in the same perturbed scenario as before. The CH algorithm resulted to be resilient to measurement errors (perturbations) while RN and BR did not guarantee the achievement of average consensus. A simple (but inefficient) solution to make RN and BR recover from errors would be to include a mechanism that automatically switches off all sensors whenever a perturbation occurs, and lets them run again with the new measurements.

Improved accuracy in switching networks

Figure 4.15 illustrates the performance of CH, RN, and BR in the operating conditions similar to those of previous tests (results in Figure 4.13 and Figure 4.14). This time, node ν_1 suddenly disappeared at time $t = 5$ s whereas node ν_{10} only appears at $t = 20$ s. As we can see, the correction mechanism introduced in CH lets nodes track the variations induced in the average value by the intermitting communications, while RN and BR algorithms fail.

In this section, we have not discussed all experiments done. We have tested the performance of the chemical consensus model in other network topologies, which were taken into account by few publications in the consensus/gossip context. We experimented in a *clustered* network, where sensors formed groups of star-connected nodes. The central node of each cluster, assumed to have higher performance in terms of reception and transmission, was able to communicate with the next cluster-head node. Further, we considered *inline* networks, and positioned nodes on a virtual line; middle nodes communicated with the two neighbors whereas edge-nodes communicated with only one neighbor. And, we generated *random* topologies by placing sensors at a random uniform distance between each other and allowing the communication between sensors within a certain distance. The result was that, in all topologies, nodes converged to the average value and recovered from erroneous states, with a speed related to the network connectivity.

Clustered networks

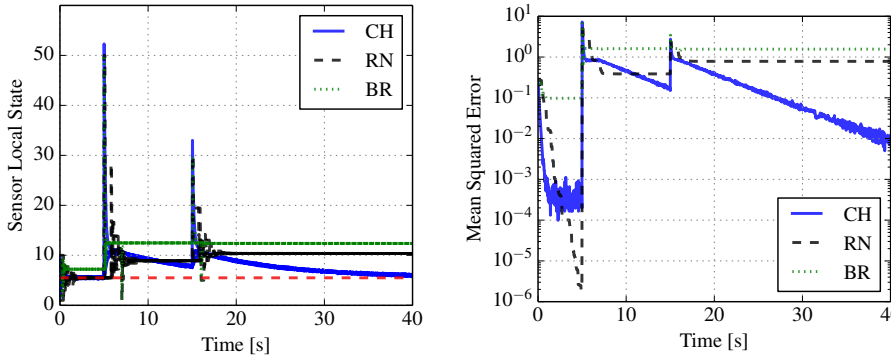
Inline networks

Random networks

Delayed networks

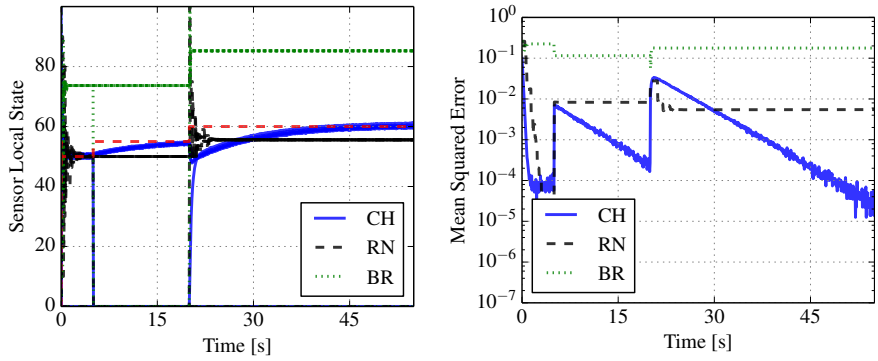
We tested also communications affected by "high" delays, i.e. $\mathcal{U}$[1,100] ms.[4.10] In a consensus mechanism relying on the $\mathcal{DAC}_i$ defined in (4.11), the effect of (highly) delayed links is the consequent underestimation of the average value by nodes. Indeed, as soon as the delay is higher than the reciprocal of the rate at which nodes transmit, a certain amount of molecules is "captured within" the link and do not appear in any node. As a consequence, when the delay is much higher than such a threshold, an important underestimation occurs. We found that the β-controlled correcting mechanism, which we have introduced to guarantee self-stabilization, also helps to correct the underestimation derived from delayed links.

[4.10] A delay $\mathcal{U}[0, 100]$ ms is even excessive if we consider that nodes do not need to exchange formatted packets. Instead, the information is encoded in the rate at which, for example, RF pulses are transmitted (broadcasted to all neighboring nodes). The experienced delay is thus directly linked to the radio propagation delay.



(a) Nodes' state evolution (b) Mean squared error

Figure 4.14: *Performance comparison among the chemical consensus (CH) algorithm and the broadcast (BR) and randomized (RN) gossip algorithms in presence of perturbations. At $t = 5$ s, node ν_1 introduced an error of 50% for 2 seconds. At $t = 15$ s, node ν_4 introduced an error of 30% for 1 second. Coefficient β was set to 0.1, and initial state and network setup were like those related to Figure 4.13. The analytical mean is reported as a red-dashed line.*



(a) Nodes' state evolution (b) Mean squared error

Figure 4.15: *Performance comparison among the chemical consensus (CH) algorithm and the broadcast (BR) and randomized (RN) gossip algorithms in switching networks – at $t = 5$, node ν_1 disappeared whereas node ν_{10} switched on at $t = 20$. The initial state was $z_i = 10i$ for $i = 1, 2, \ldots, |\mathcal{V}|$. Coefficient β was set to 0.1 and the network setup was like that related to Figure 4.13. The analytical mean is reported as a red-dashed line.*

4.7 Testing the chemical consensus algorithm under real conditions

Chemically-engineered algorithms can lie directly at the physical layer and can be successfully used to control the basic hardware transmission technologies. We describe in Section 4.7.1 how we have implemented the $\mathcal{DAC}$ (4.38) on embedded microprocessors, and we report in Section 4.7.2 on results of real-world experiments of a four-node testbed (photograph in Figure 4.16), in which nodes exchanged data in an asynchronous manner with no need for packets and admission control.

4.7.1 Artificial chemistries on a hardware testbed

$\mathcal{DAC}$ implementation

The dynamical system, operating according to the $\mathcal{DAC}$ in Figure 4.12, has been implemented into the embedded processor TI-MSP430F5438A, which served also for data acquirement and conversion. The radio interface (transceiver) has been developed with a simple low-cost circuit by using the TXM-433-LR integrated chip for transmitting to and the RXM-433-LR integrated chip for receiving from nearby nodes. To limit the complexity of each node, we let the sensor interactions occur in a simple manner. Specifically, a pulse $g_\tau(t)$ of duration τ was sent over the channel whenever the remote reaction $r_{B',i}$ in (4.39) was executed (in the dynamical system). On the other hand, the production of an S-molecule was induced whenever a pulse $g_\tau(t)$ was received from nearby sensors. According to the LoMA in (2.6), the average rate of occurrence of reaction $r_{B',i}$ turned out to be proportional to the concentration of S-molecules (sensor's state) at node ν_i. This means that the spatially distributed nodes interacted by means of pulses whose transmission rate encoded the nodes' state. Following the same line of reasoning, we let a pulse $g_{\tilde{\tau}}(t)$ of duration $\tilde{\tau} \neq \tau$ be sent whenever the remote reaction $r_{X,i}$ was executed, and a Y-molecule be produced whenever $g_{\tilde{\tau}}(t)$ was received.

To ease understanding, consider a simple network only composed by $|\mathcal{V}| = 2$ nodes. Without loss of generality, we concentrate on the first node and explain how sensor interactions occur. As a consequence of reaction $r_{1,B'}$ in (4.39), the node starts transmitting pulses $g_\tau(t)$ at a rate equal to the initial concentration of its local species (value equal to the measured quantity, i.e., $c_{\mathrm{S}_1}(0) = z_1$). At the same time, $c_{\mathrm{S}_1}(t)$ is continuously modified on the basis of the number of $g_\tau(t)$-pulses that are received from node ν_2 (as a consequence of reaction $r_{B,2}$). This happens at a rate equal to $c_{\mathrm{S}_2}(t)$. Internally, the current state $c_{\mathrm{S}_1}(t)$ is also continuously decreased at a rate $c_{\mathrm{S}_1}(t)$, according to reaction $r_{1,D''}$ in (4.27), and modified proportionally to the β-parameter, according to the reactions in (4.30). To continuously estimate the number of neighbors, node ν_1 has also to transmit $g_{\tilde{\tau}}(t)$-pulses at a constant rate α and to increase $c_{Y_1}(t)$ whenever a pulse $g_{\tilde{\tau}}(t)$ is received from node ν_2. All these interactions are managed at run time by the simple execution of $\mathcal{DAC}$ in Figure 4.12, according to chemical

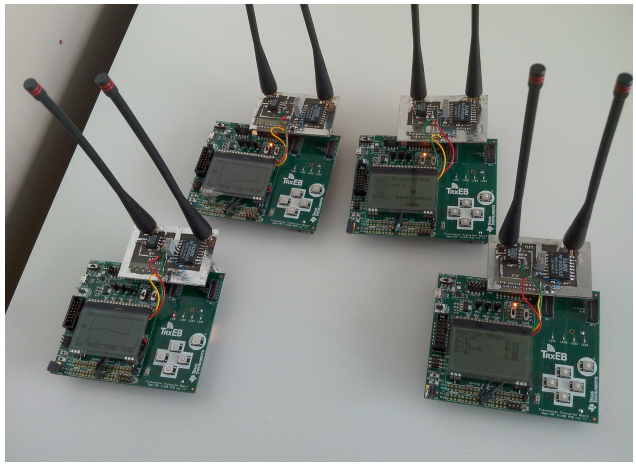


Figure 4.16: *Experiment testbed consisting of four sensors where the dynamical system ($\mathcal{AC}$) has been implemented in the embedded processor TI-MSP430F5438A, and the radio transceiver through transmitter TXM-433-LR, receiver RXM-433-LR, and comparator MAX-921. Sensors have been implemented on TI-CC1120 evaluation board, having also an LCD display for a practical setup and tracking of run-time computation (RF-module provided with TI-CC1120 has been replaced by the self-developed RF transceiver).*

kinetics (running the reaction Algorithm 2).

Quantity-to-molecule ratio to control...

...number of transmissions

...accuracy

...representable range

The pulse rate depends on the sensor's state. However, this does not mean that if $z_i = 10^9$ then 10^9 pulses $g_\tau(t)$ per second must be transmitted by node ν_i. Instead, it is important to decide how the values are encoded or, in other words, how a certain amount of molecules has to be interpreted. For example, assume that the system has to measure the average temperature in a sensor network. Then, in order to limit the pulse rate, one has to properly associate the right quantity (e.g., degrees celsius, Kelvin, Fahrenheit) to a single molecule instance. This enables controlling the maximum transmission rate at the price of a reduced accuracy of the computation (accuracy in the average estimation).

We have used the free Industrial Scientific Medical (ISM) radio band $433.05 \div 434.79$ MHz. Pulses $g_\tau(t)$ and $g_{\tilde{\tau}}(t)$ had durations of $\tau = 100$ μs and $\tilde{\tau} = 200$ μs, respectively. To obtain better performance in terms of on/off transition time of the transceiver, we have not used directly the output signal of RXM-433-LR but rather its RSSI-signal as input to the MAX-921 comparator, and extracted from there the demodulated pulse. Transmitter and receiver have been equipped with 1/4-wave antennas. In such a configuration, the whole sensor was characterized roughly by a total consumption of ~ 36 mW distributed in ~ 13 mW consumed by the embedded processor, ~ 17 mW by the receiver, and ~ 6 mW by the transmitter. The receiver's typical sensitivity

was −120 dBm and its dynamic range 80 dB, whereas the transmitter's typical output power was 3 dBm.
WSNs require simple, low-energy solutions since sensors undergo power-supply constraints and should be low-cost and robust. Motivated by the above reasons, we have adopted a time-division solution where sensors used the same frequency carrier (433.92 MHz) and "shared" a common period $T_{\max} = 10$ ms: we have encoded the transmission of each S-molecule into a 100 μs-long pulse ($g_\tau(t)$ with $\tau = 100$ μs) every time the broadcast reaction r_B fired. Additionally, each sensor had a local 10 ms-clock and for each time-tick, it transmitted a 200 μs-long pulse ($g_{\tilde{\tau}}(t)$ with $\tilde{\tau} = 200$ μs) if the remote reaction had not fired in the last 10 ms. From the reception point of view, each sensor produced an S-molecule for each received pulse $g_\tau(t)$ but produced also a Y-molecule for each received long $g_{\tilde{\tau}}(t)$ or short $g_\tau(t)$ pulse. This allowed saving part of the transmission power. However, the side effect was a fixed maximum rate at which sensors could transmit, i.e. $1/T_{\max}$, thus a fixed ratio "range-of-representable-quantity to estimation-accuracy".
A possible drawback of the above implementation is that no countermeasures are taken against interferences that might arise in WSNs, when the signals transmitted by multiple nodes collide at a given receiving node. Although a judicious design of the system parameters (maximum value of concentrations, duration of the pulse and so forth) could reduce the occurrence of collisions, more advanced multiple access protocols are required to effectively counteract the above issue, thus increasing the complexity of each sensor. For this reason, we have decided not to take countermeasures against interferences. This choice has also been motivated by the observation that chemical systems usually exhibit strong robustness to perturbations thanks to the mass-action kinetics governing their interaction mechanisms (refer to [215] for a recent work in the context of sensitivity and robustness of chemical reaction systems). The robustness is a direct consequence of the fact that the information exchange is encoded into a rate rather than in one or few information packets, and thus any corruption of one or few of these transmissions does not significantly affect the system. Another reason for the inherent robustness is that mass-action kinetics often induces low-pass filtering behaviors and transfer functions exhibiting negative real-part poles. Such a robustness has indeed been confirmed in our experiments, although only four-node WSNs were tested.

4.7.2 Experiments on a four-node testbed

We experimented on our four-node hardware testbed (see Figure 4.16), operating according to the $\mathcal{DAC}$ defined in Figure 4.12. To our knowledge, this was the first time that a Chemistry-inspired algorithm was built in a hardware testbed and validated under real-world conditions.
In the experiment, we directly controlled the quantities $\mathbf{z}$ characterizing the sensors. In Table 4.3, we report how the value z_i in each sensor ν_i changed at time t_j. The experiment was divided into five phases: Initially ($t = 0$ s), when

all sensors sensed 0, ν_1 suddenly changed the local quantity z_1 from 0 to 50. Similarly, at time $t \sim 9.8$ s, ν_2 suddenly changed z_2 from 0 to 50, whereas at time $t \sim 19.9$ s, ν_3 suddenly changed z_3 from 0 to 20. At $t \sim 57$ s, ν_2 started decreasing z_2, reaching $z_2 = 0$ at $t \sim 57$ s. Finally, ν_2 left the network at $t \sim 63$ s. The nodes moved in an area of 50 m^2, in an indoor environment.
Figure 4.17(a) illustrates the sensors' state evolution of each node ν_i (the amount of molecules S_i sampled 100 times per second) under the aforementioned operating conditions. Sensor nodes adapted their state in response to variations of the local quantities $\mathbf{z}$ and converged quickly (less than 5 s) to the arithmetic mean (red-dashed line), a-posteriori calculated as column-sum of values in Table 4.3 divided by the number $|\mathcal{V}|$ of participating nodes.
Sensor nodes were able to recover from erroneous states induced by external interferences: An interfering signal (created on purpose from $t = 22$ s to $t = 33$ s) made the sensors temporarily underestimate the average. Once removed, each sensor converged to the desired value in approximately 20 seconds.
The chemical consensus algorithm induced perfectly predictable dynamics of sensors: In Figure 4.17(b), red-dashed lines represent the trajectories predicted through (4.36) when $\delta_{z,k}(s) = \delta_{z,k}$ reflects the impulse measurement variation (e.g. in time-domain, $\delta_{z,1}(t = 0) = 50$, and $\delta_{z,1}(t = 0^-) = \delta_{z,1}(t = 0^+) = 0$). The time-zoomed trajectories measured during the first nine seconds of the experiment are closed to that analytically predicted.

4.8 Further insights on the Chemistry-inspired consensus

This chapter was basically divided into two parts: we have shown how Artificial Chemistry can be beneficial to extract a model for achieving consensus in WSNs and how to exploit the derived analytical tools, and we have presented an implementation for this model to prove the realizability and the validity of our findings. In the following, we separately comment on these two specific parts, highlighting open issues and possible future research topics.

4.8.1 Artificial Chemistry for consensus in WSNs

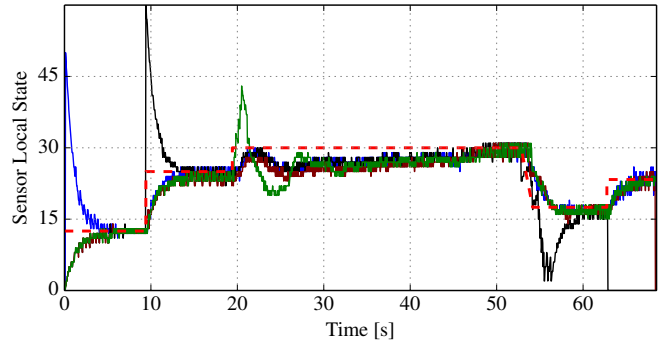
We have first shown analytically and by means of simulations that simple interaction mechanisms inspired by chemical systems can provide basic tools for achieving consensus in WSNs. However, some issues are still open.
In this work, we do not cover in detail the discretization aspect (studied for example in [50] and summarized in [66]), which may affect the final nodes' average-estimate. We limit to observe here that, by calibrating the amount of molecules that are produced per measured-quantity unit, designers can regulate the precision of the nodes' estimates and decide the amount of transmissions.[4.11]

Discretization

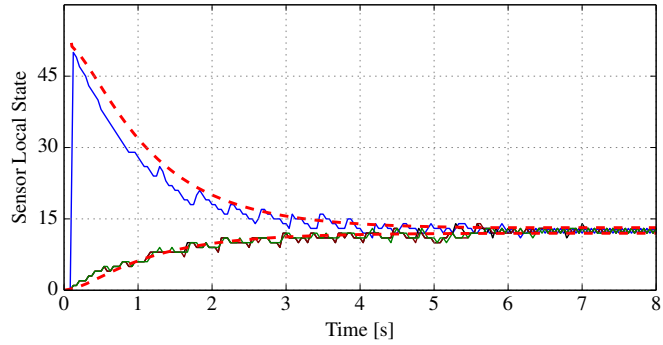
[4.11] In the experiments, we have thus coped anyway with the effect of discretization: we

Time instant t [seconds]	0	9.8	19.9	$53 \rightarrow 57$	63
Value of z_1	50	50	50	50	50
Value of z_2	0	50	50	$\rightarrow 0$	/
Value of z_3	0	0	20	20	0
Value of z_4	0	0	0	0	0

Table 4.3: *Local data z_i detected by node ν_i at time t during the experiment with a four-node hardware testbed. The results are plotted in Figure 4.17(a).*



(a) Nodes' state evolution



(b) Zoom: Analysis *vs.* measurements

Figure 4.17: *(a) Experiment results (4 colored, continuous lines) and arithmetic mean (red-dashed line) showing the sensors' state evolution over time. The β coefficient was set to 0.1. An interference was created on purpose from $t = 22$ s to $t = 33$ s. Refer to Table 4.3 for the related dynamics of the measured values $\mathbf{z}$. (b) time-zoomed trajectories of the sensors' state, measured in the experiments (colored-continuous lines) and predicted by means of (4.36) (red-dashed line).*

Another way to reduce computations/transmissions, and thus the resource consumption, consists in slowing down the virtual time characterizing the chemical model: transmissions decrease in number, this time at the cost of a proportionally-slower adaptation. Note that, in our simple implementation, lower rates mean also lower probability of collisions/interferences.

In this work, we considered a very simple communication model and did not refer to specific channel models and communication technologies. However, by means of simulations, we have tested the ability of a Chemistry-based system to recover from perturbations. Such perturbations may represent the generalized effect of fading (transient under-valued state of one or more sensors), multipath (transient over-valued state of one or more sensors), or non-reliable links (variation of the number of nodes and different participation times).

This work has focused on balanced digraphs only. In the presence of unbalanced graphs, the theoretical and experimental results illustrated in this chapter are no longer valid. Indeed in this case, sensors *converge* to a common value that differs from the average of the local measurements.[4.12] This result occurs because, at the equilibrium, molecules are still evenly distributed among all participating sensors but are "diffused" with different proportions, depending on the ratios between the number of receiving and transmitting neighbors. Future work could focus on the development of solutions able to take into account such different weights while still using chemical mechanisms of similar complexity to those adopted in this chapter.

Unconstrained consensus problems are always solvable

4.8.2 An implementation of Artificial Chemistry in WSNs

The second part of the chapter has shown that the $\mathcal{AC}$ can work under real-world conditions, and that the above framework can directly be used to develop a hardware implementation.

Although the experimental results are quite promising even under real-world conditions, the work presented in this chapter should not be seen as a finalized, ready-to-use commercial product for nowadays markets. We believe that further research in the implementation context may bring significant improvements in terms of robustness and speed.

For example, as already mentioned, the simple implementation we proposed does not consider the admission to the shared wireless medium. The consequence is the higher or lower likelihood to experience communication collisions and the consecutive underestimation of the average by nodes. We can "quantify" this phenomenon in terms of collision probability and possibly overcome it by letting the chemical reactor compensate the unwanted outflow created by

have had to use a number of molecules per unit quantity such as to obtain an acceptable estimation accuracy.

[4.12] For unbalanced graphs, the presented model does not represent a solution to *constrained* consensus problems but it still is a *solution to unconstrained* consensus problems, according to definitions in [180].

collisions. Refer to Appendix A.8 for further details on this topic. Here, we limit to observe that such a compensation corresponds in practice to complementing the $\mathcal{DAC}$ in (4.12) with a reaction that generates lost S-molecules at the speed at which collisions theoretically occur on average. The latter depends on a quantity each node knows, the number $|\mathcal{N}_i|$ of neighboring nodes.
A different implementation may use a frequency-division approach rather than the time-division one we have proposed in this chapter. By assigning to each node a specific reserved frequency-channel, we can ensure the absence of collisions, at least due to node-cross-interferences. However, this approach requires a more complex technology than that used for the presented testbed. Furthermore, we have to consider that, as the number of nodes increases, the required bandwidth does too. We could increase the frequency resolution of the transceiver in order to reduce the bandwidth associated to each channel. However, this inevitably increases the minimum physical time required for the acquisition of a sufficient amount of samples and thus, it is related to the maximum speed with which we can turn on-off the carrier. Refer to Appendix A.9 for further details.

4.9 Conclusion

In this chapter, we have demonstrated that the use of Chemistry to design and study solutions for the consensus problem represents a valid alternative to traditional methods. This work represents a first comprehensive treatment, ranging from "applied theory" to implementation.
We have introduced a set of simple interaction rules that, directly derived from intuitive patterns, allows achieving consensus in WSNs in a distributed manner, with no need for any synchronism and admission control mechanism. We have focused on balanced digraphs and only for this kind of networks, the proposed interaction rules solve constrained consensus problems, although they still always solve unconstrained consensus problems. From these interaction rules, we automatically extracted a fluid model which is the same as the one well established in the literature. The proposed final solution has been validated and compared with other solutions by means of experimental results, and further tested under real-world conditions by means of a four-node hardware testbed. Sensor nodes exhibited state-trajectories that closely match the analytically predicted ones. The numerical and experimental results show that the use of Artificial Chemistry for deriving, analyzing, and implementing communication protocols is not merely an intellectual exercise but an alternative approach, which may pave the way for the development of robust solutions, able to cope with the uncertainties of WSNs.
This chapter represents a first step towards the utilization of hardware to run artificial chemistries in the context of networking and communication. For the first time, data-exchange through artificial chemistries do not rely on complex layered communication infrastructures, where the main means to communicate

are packets. Instead, artificial chemistries are implemented on embedded microprocessors and lie at the lowest layer of the communication stack; chemical reactions in the dynamical system are directly linked to the RF-transceiver module. Although applying Chemistry to the consensus context may require further investigations, we hope this work serves as an incentive for the research community for further explorations in this context.

Chapter 5

Artificial Chemistry to Enable Programmable Control in Hardware

"When you first start off trying to solve a problem, the first solutions you come up with are very complex, and most people stop there. But if you keep going, and live with the problem and peel more layers of the onion off, you can often arrive at some very elegant and simple solutions."

Steve Jobs

In this chapter, we investigate the realizability of Chemistry-inspired mechanisms in hardware and propose a first implementation for this novel class of algorithms on Field-Programmable Gate Arrays. The introduced framework, or design-abstraction, can be used to build any (Chemistry-driven) hardware controllers, and it is promising to enable a high programmability of network-dynamics-related tasks. The user can define/modify at runtime the behavior of hardware controllers by simply configuring a few of memories (which define the driving chemical model). In this way, the programmability by the user is preserved both pre- and post-deployment.

This chapter is structured as follows: We first motivate this work in Section 5.1. Then, we explain in Section 5.2 the main components ("cores") we have created to build an Artificial Chemistry in hardware, and we complete the explanation by illustrating in Section 5.3 how these components are connected to realize a programmable Artificial Chemistry on a small FPGA-device. We show in Section 5.4 how such a platform can be configured to implement a rate controller for Internet traffic, and to realize a system of hardware devices that distributively compute the average function. We finally discuss obtained performances, open issues, and possible impact of this work in Section 5.5.

5.1 Introduction

Until now, in this thesis, we have proposed to control communication and networking systems by means of $\mathcal{AC}$s, by letting a virtual dynamical system (the $\mathcal{AC}$) evolve over time according to chemical kinetics, and by driving accordingly the behavior of nodes in computer and wireless sensor networks. This means each node has to be equipped with a "chemical engine" that, once configured with a certain chemical reaction network, implements and runs the reaction algorithm $\mathcal{A}$ so that the desired macro-behavior emerges. As we know from Chapter 2, the calculation of propensities and next-reaction times as well as the update of molecular concentrations entails priority computations to be performed at run-time. This may require a computational power that, according to which CPU the $\mathcal{AC}$ is implemented in and to the speed at which events have to be processed, turns out to be unaffordable. For example, the performance attained in Chapter 3, although satisfactory for end-systems, would be very costly when executed on the CPU of a network router with multiple network interfaces that are served at line speeds.

In fields like traffic management, there is often the need for processing at high speeds events such as packet arrivals and departures. For this reason, traditional traffic management solutions often need to include a hardware accelerator with a tailored internal structure for the specific task it has to perform. In general, a hardware accelerator is a co-processor that is capable of performing a certain task faster (performance), cheaper (power), and/or more precisely (computational accuracy) than a standard CPU does. Although hardware accelerators offer a reasonable performance improvement, a customized logic hardware is however required when aiming at speed and accuracy optimization [83, 259]. Without any sort of operating system and any software-based description, decision- and action-mechanisms that are implemented on raw hardware must be simple [138, 211].

In this final chapter, we show how to build complex systems while still maintaining their implementation simple enough to be built in hardware. This allows achieving ultimately fast processing in $\mathcal{AC}$-based communication and networking systems. We propose a hardware implementation of $\mathcal{AC}$s, and validate it on Field-Programmable Gate Arrays (FPGAs), taking advantage of the reliability of dedicated hardware circuitry, of the parallel execution, and of the lightning-fast performance.

Field-Programmable Gate Arrays (FPGAs)

An FPGA is a device that contains a matrix of reconfigurable gate array logic circuitry, surrounded by a periphery of I/O blocks. Once it is configured, the internal circuitry is connected in a way that creates a hardware implementation of the desired task (FPGA-technology is explained in details at the beginning of Section 5.2). Thus, FPGAs enable using dedicated hardware for processing logic and deter relying on an operating system or on a sequential representation of actions. Rather, FPGAs enable true computation parallelism by nature; different processing operations do not have to compete for the same resources. As a result, the performance of one part of the application is not affected

when additional processing is added, as opposed to what happens in software applications. Unlike Application Specific Integrated Circuits (ASICs), which are integrated circuits customized for a particular use with fixed hardware resources, FPGA-based systems can literally rewire their internal circuitry to allow reconfiguration after the system is deployed. Thus, FPGA-technology allows us to "program" (Chemistry-inspired) controllers on the fly. Besides the possibility to update the functionality and to re-configure a portion of the design, the FPGA-technology has also the advantage of substantially lower engineering costs. Nevertheless, for huge production volumes, the framework we propose here can be used to produce customized integrated circuits, with benefits in terms of speed and energy consumption. We do not use Complex Programmable Logic Devices (CPLDs) because of the limited logic resources, not sufficient to implement complex $\mathcal{AC}$s, and again because of the reduced flexibility of post-deployment configuration.
Before introducing how we have implemented $\mathcal{AC}$ on FPGA-technology, we comment on a few interesting works that propose hardware implementations in contexts that relate to our work, and we explain the space of our contributions.

5.1.1 Related works

In networking and communication systems, there is often the need for processing and computing at high rates. This is attested by the effort spent by the research community (e.g., [9, 42, 139]), as well as leading companies on the hardware market (e.g., [15, 196]), in finding solutions that can be fitted in efficient, tailored processors, or that can even be implemented in hardware. For example, the traditional forwarding and routing of packets are problems that often imply hardware implementations, where required decision and action mechanisms are accomplished at high speeds (line rates) by taking advantage of hardware technology such as FPGAs.
FPGAs have extensively been used for networking tasks due to benefits such as fast time-to-market, ability to fully exploit computational parallelism, and high processing speed. Nevertheless, networking functions on hardware remain neither easy to program nor fast to modify (particularly post-deployment). Although Hardware Description Languages (HDLs), such as VHDL [118] and Verilog [117], allow modeling, verification and automated synthesis of the digital designs, the process of hardware designing takes a long time, is laborious [138], and inhibits the implementation of complex control functions [7].
Motivated by the mentioned difficulties, soft-processors (processors composed of programmable logic on the FPGA) have frequently been proposed in the last years, see for example [139, 170, 202, 229]. These solutions still rely on FPGA-technology but use a software-based description, by means of CPUs embedded on FPGAs. Soft-processors are popular because they (*i*) are easy to program (e.g., by means of C-Language), (*ii*) are flexible (i.e., can be customized), and (*iii*) enable solutions which include complex decisions and actions. For example, the authors of [73] implemented Rate Control Protocol (RCP) routers in

Software-based sequentialized descriptions of the system

hardware but still performed the periodic control computations (e.g., equation-based bandwidth allocation and moving average of round-trip times) in software, and left to actual hardware implementations only simple tasks such as the time-stamping and the identification of packets (i.e., identifying whether an incoming packet is an RCP packet). Another motivation for the success of soft-processors is the fact that most networking researchers are used to high-level software descriptions and not trained in hardware design [211]. Even for those that are, packet processing and traffic-control designing in a hardware-description language is time consuming and error prone [138]. In spite of these attractive advantages, software-based approaches renounce however to much of the benefits (in terms of accuracy and speed) derived from running solutions on hardware. An advance in terms of speed has been done with OpenCL [13], a standardized C-language description for programming parallel tasks on processors embedded in FPGAs. Still this approach requires a strained transposition from the high-level software-based description to the actual low-level hardware implementation. Somehow, software-based approaches renounce to an elegant and simple solution that is actually close to the hardware representation.

5.1.1.1 Hardware solutions for traffic shaping

Few research works (e.g., [9, 106]) focus on full hardware implementation and deviate from software-based design. An outstanding project (from an academic perspective) for experimentation with hardware is NetFPGA [147, 177].

NetFPGA

NetFPGA is a line-rate, flexible, and open platform for research experimentation, which uses an FPGA-based approach to prototype networking devices, and allows users to develop solutions with capabilities (e.g., line-rate packet processing) that generally are not achievable with software-based approaches.[5.1] Most of the seminal works concerning hardware-based traffic shaping solutions have used the NetFPGA platform. Although the majority of them still perform update equations on the software layer of NetFPGA (e.g., [73,151]), a few works actually build complete control mechanisms in hardware. For example in [9], NetFPGA hardware is used to implement (*i*) a packet pacer that works at the NIC layer and (*ii*) phantom-queue modules that are used for Active Queue Management (AQM) purposes. As another example, authors of [106] presented the design and prototype of a hardware implementation of a packet pacer on the NetFPGA platform, with the aim of showing the benefit of actively smoothing traffic in small-buffer networks such as optical packet switching networks.

Traffic-Management IP-Cores

From a more commercial perspective, Xilinx Inc. offers a common, unified FPGA architecture that enables "next-generation traffic management" and packet-processing applications with a broad feature set [196, 255] – i.e., the possibility to have many hierarchical scheduling levels, hundreds of queues, burst equalization, Random Early Detection (RED) option, per-flow scheduling

[5.1] NetFPGA is a Gigabit Ethernet open platform for networking research which has been developed at Stanford University, consisting of a PCI card with a Xilinx Virtex 2Pro and Spartan 3 FPGAs, SRAM, DRAM and 4 Gigabit Ethernet ports.

logic[5.2], per-flow token bucket shaper, and throughput of GBps. Similarly, Intellectual Property cores (IP) from Altera Corp. assist designers in implementing traffic management FPGA-based solutions [14, 15]. Their IP incorporates five levels of scheduling to enable the implementation of hierarchical QoS and flexible queue configuration and mapping. It offers the possibility to work with more than one queue, with different shaping algorithms[5.3] and with different scheduling algorithms. Finally, Lattice Semiconductors Corp. produces low cost, low power FPGA-technology with a Traffic-Management-specific IP [140].

5.1.1.2 Hardware for Nature-inspired systems

In the context of Natural Sciences[5.4] (e.g., simulation of observed phenomena, solution to complex problems, or distributed computation via Nature-inspired mechanisms), research activity has mainly relied on software-based solutions. There exist only few Nature-inspired hardware-based solutions, although Nature-inspired approaches (which encompass usually basic subsystems interconnected via simple rules and actions) exhibit features such as the inherent processing parallelism that resemble hardware systems. One example is Artificial Neural Networks (ANNs) implemented on FPGAs, e.g., [82, 150, 161]. ANN are massively parallel computation systems that are based on simplified models of the human brain. Their complex classification capabilities, combined with properties such as generalization, fault-tolerance and learning make them attractive for a range of applications not possible with conventional computational approaches (e.g., video motion detection, hand-written character recognition and complex control tasks). In this context, FPGAs combine programmability (although this results in an intrinsic overhead and thus a limited logic density of FPGAs) with the increased speed of operations associated with parallel hardware solutions. Other examples of Bio-inspired systems on FPGAs can be found in [132, 171, 172], where Cellular Automata systems are implemented on FPGAs. Cellular Automata are a noteworthy candidate among parallel processing alternatives (e.g., for public-key cryptography, error-correction coding, image processing) and for simulating natural phenomena (e.g., moving wave patterns on living organisms' skin, gas and fluid dynamics, complex behaviors such as recognition and learning); they are characterized by simplicity, parallelism and distributiveness.

Artificial Neural Networks

Cellular Automata

There exist also works that propose to speed up simulations of chemical and biological systems by delegating computations to FPGA technology [126, 148, 209, 239, 261]. Motivated by the clear need for high-speed processing in "next-generation" communication networks, we have also developed a hardware platform where $\mathcal{AC}$s can be easily implemented and then used to control networking

[5.2] E.g., Round Robin, Weighted Round Robin, or Weighted Fair Queuing.

[5.3] Token bucket or leaky bucket.

[5.4] With "Natural Sciences" we mean branches of science dealing with the physical world, i.e., Physics, Chemistry, Geology, and Biology.

and communication systems' dynamics. In the following, we summarize specifically our contributions in this regard.

5.1.2 Space and scope of our contribution

$\mathcal{AC}$-based high-level abstraction for designing hardware controllers

In this chapter, we investigate the realizability of Chemistry-inspired algorithms in hardware, and propose a first hardware-implementation. We describe a very flexible platform for building $\mathcal{AC}$s in FPGAs and hence hardware mechanisms for communication and networking. Engineers can realize solutions for applications ranging from traffic management (e.g., a plug-and-play rate controller for Internet traffic) to distributed computation (e.g., distributed average computation in sensor networks), by simply changing the configuration of a few memories that defines the underlying chemical model.

We enable processing parallelism in control-mechanism design while maintaining analyzability and predictability of dependent-conditions and actions, but avoiding traditional sequential-description. We do not resort to any sort of software-based approaches, in favor of a parallelized solution that processes events at high speed and with high time-granularity. At the same time, we still guarantee a flexible and simple way to design hardware-controllers – designers only have to draw an appropriate reaction network that accommodates the (macro-) requirements of their project. By parallelizing most computations required from the reaction algorithm $\mathcal{A}$, computational performance is drastically improved – no matter the complexity of the chemical model, processing times can be limited to nanoseconds.

$\mathcal{AC}$s and $\mathcal{DAC}$s with chemical-like kinetics

We implement chemical-like systems (i.e., $\mathcal{AC}$s) in silicon – via basic logic gates and flip-flops, we realize networks of molecular species that interact continuously and in parallel, according to predefined reaction rules and external events. $\mathcal{AC}$ is not bound to a single chip only but rather it can be easily embedded in multiple FPGA-devices to perform tasks in less time or with the need for less powerful (in terms of logic) devices – a $\mathcal{DAC}$ running on multiple FPGAs. The dynamics of developed systems follow chemical laws and principles. This allows obtaining trajectories that are easily analyzable, while maintaining enough implementation simplicity.

To validate our work, we provide results obtained from experiments with Xilinx Spartan-6 XC6SLX9 FPGA. The used implementation optimizes the logic density rather than the computational speed. In this way, we were able to implement within a small FPGA complete $\mathcal{AC}$s that can process/control external events at high speed. The complexity of configurable chemical models (in terms of number of species, reactions, reactants, and products) is sufficient to create a wide spectrum of solutions. With this platform, we realize (i) a plug-and-play module to control the egress traffic of a linux machine (the hardware-implementation of the traffic rate controller introduced in Chapter 3), and (ii) a three-node wired-network, where nodes converge in a short time to the average of their initial value (the hardware-implementation of the chemical consensus algorithm explained in Chapter 4). While, no works have been published in

Plug-and-play rate controller for Internet traffic

Distributed average computation in wired networks

the context of hardware-implementation of consensus algorithms,[5.5] hardware rate control has been treated for example in [109,138,139,170,202]. In contrast with these studies, we do not try to directly solve complex traffic management tasks, and we can thus avoid implementation approaches that rely on software-description and merely reproduce processors on FPGA. At the same time, we offer the possibility to design within the same platform arbitrary runtime-programable controllers (as opposed to works, such as [9,106], that reduce the control mechanism to a fixed basic rule) – designers can define control mechanisms consisting of almost arbitrary (virtual and real) queues that interact with each other according to almost arbitrary rules.
This work can enable the programmability of high-efficiency tasks pertaining network dynamics in future network interface cards, routers, switches, and sensors. This may represent a significant contribution also to the research in Software Defined Networking (SDN) [185], where today's technologies like OpenFlow [159] and OpenStack [187] can access/modify parts of a router/node that pertain to topology creation and configuration, but have no means to control and configure the network dynamics. Dynamics-control operations are generally challenging because they (*i*) require a more frequent control/management intervention than the one required to setup a topology, and (*ii*) imply a deeper algorithmic complexity (from a programmatic point of view) than the one required to update lookup-tables and set-up flow filters. What we provide in the end is an abstraction/middleware to build run-time programmable hardware modules for dynamics-control operations.

5.2 Artificial chemistries on FPGA

We aim at a middleware/abstraction that enables configuring $\mathcal{AC}$-based networking mechanisms and control functions, which are implemented on FPGAs. Conceptually, FPGAs can be considered as an array of Configurable Logic Blocks (CLBs) – also referred to as "slices" or "logic cells" – that can be connected together through a vast interconnection matrix to form complex digital circuits. The two basic components of CLBs are Flip-Flops (FFs) and Look-Up Tables (LUTs). FFs are binary registers used to synchronize logic and save logical states between clock cycles within an FPGA circuit – on every clock edge, a FF latches the 1 or 0 (`true` or `false`) value on its input and holds that value constant until the next clock edge. LUT-memories allow registering the combinatorial output of the CLB – any combinatorial logic (ANDs, ORs, NANDs, XORs, *etc.*) is implemented as truth tables[5.6] within LUT memory. Other main components of FPGAs are the I/O blocks, which allow the circuit

FPGA-technology

[5.5] Apparently, there is no need for FPGA-technology in the context of consensus. This is because at the moment, consensus solutions are applied mainly in wireless sensor networks where the power consumption that FPGA-devices may require can be a concern.

[5.6] A truth table is a predefined list of outputs for every combination of inputs.

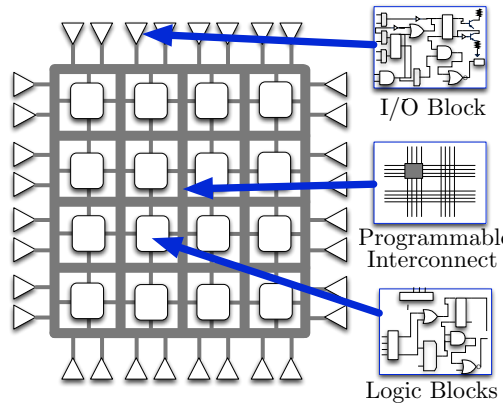


Figure 5.1: *Main components of the FPGA-technology (Figure inspired to a picture in [174]).*

to access to and be accessed from the outside world. Figure 5.1 outlines the main components of the FPGA-technology.

The function of the FPGA is defined by the user; the design is either programmed semi-permanently as part of a board assembly process, or it is loaded from an external device each time the FPGA is powered up. A user's design is implemented by specifying the simple logic function for each CLB and selectively closing the switches in the interconnect matrix. Each CLB combines a few binary inputs (typically between 3 and 10) to one or two outputs, according to a boolean logic function specified by the user.

The actual hardware design is defined via a textual hardware description language code (or by drawing schematic diagrams) which is then translated in places and routes. The design process is simplified by the use of libraries of more complex functionality-macros, made of arrays of CLBs (e.g., adders, multipliers), which provide common circuits, optimized for speed or logic utilization.

The programmability by the user of Chemistry-inspired mechanisms stems from the configuration of part of the FPGA. This simply means configuring memories that store values of (*i*) molecular concentrations, (*ii*) stoichiometric coefficients, and (*iii*) reaction-rate coefficients. The rest of the logic resources is instead semi-permanently programmed and used to (*i*) perform arithmetic operations in order to quantify the propensity $\mathbf{a}$ of reactions, (*ii*) realize timers in order to "materialize" chemical kinetics (i.e., let each reaction occur at the right time according to the LoMA), and (*iii*) realize the automated update of molecular concentrations, at instants defined by chemical kinetics (by reaction-rate coefficients) and according to reactions (to stoichiometric coefficients).

We have directly used the basic building blocks as provided by FPGA manufacturers (i.e., Xilinx's logic cores [251, 252]). We thus focus only on the missing

basic components to realize an $\mathcal{AC}$-platform, without explaining in detail the implementation of adders, multipliers, and other basic operator logic cores we have used (e.g., ripple carry array, row adder tree, carry save array, LUT, *etc.*). Still, we discuss their features that are relevant to our design choices.
According to the definition in Chapter 2, three main components define an $\mathcal{AC}$: species $\mathcal{S}$, reactions $\mathcal{R}$, and algorithm $\mathcal{A}$. We first describe the structure of the chemical model (i.e., we introduce $\mathcal{S}$ and $\mathcal{R}$) in Section 5.2.1, and then describe how chemical dynamics are imposed on the model (i.e., how the mass-action scheduler $\mathcal{A}$ is described via combinatorial logic) in Section 5.2.2.

5.2.1 Defining the structure of the chemical model

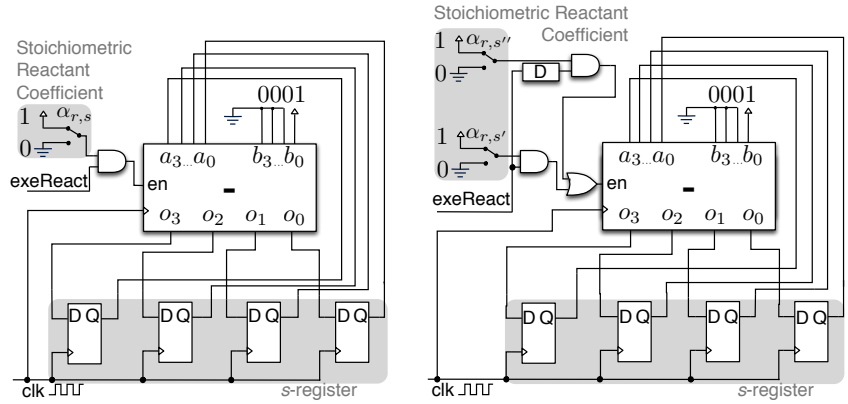
To run an $\mathcal{AC}$ we need to monitor its state, i.e., the species' state (the amount of virtual molecules of each species). Thus, we need registers (one for each species) that, once initialized, keep track of the run time number of corresponding molecules. To this end, we require a Random Access Memory (RAM) that has as many entries as the number $|\mathcal{S}|$ of species in the $\mathcal{AC}$, each of them having a reasonable number of bits to store a sufficient (limited) amount of molecules. For example, if we want to keep track of 8 species with a limited concentration range (possible number of molecules) between 0 and 255 molecules, we should use 8 times 8 FFs connected in cascade. Alternatively, we could take advantage of a part of the block RAM embedded in most of available FPGAs.

Concentration memory (c-RAM)

Reactions define via the stoichiometric matrix how species interact, i.e., how the RAM latching concentrations (c-RAM) is updated. This matrix is specified by the user and remains unchanged during the runtime evolution of the $\mathcal{AC}$, unless the user configures it differently. For example, the stoichiometric reactant coefficient $\alpha_{r,s}$ represents the amount of molecules of a certain species s consumed by a reaction r.[5.7] In the implementation of an $\mathcal{AC}$ in hardware, we simplify the meaning of this coefficient by allowing it to assume 1 (the species s *is* a reactant of reaction r) and 0 (the species s *is not* a reactant of reaction r) values only. In this way, we can implement the stoichiometric reactant coefficient as a simple 1-bit signal that enables or disables the subtraction of "1" from the register of the concerned species (i.e., the bank of FFs, representing a single location of the c-RAM). Figure 5.2(a) outlines such a scheme for a single reaction r and a single species s, whose concentration (amount of molecules) can be any integer value in the range $[0, 15]$. The 4 FFs constitute the cascaded 4-bit register for latching the species concentration value (highlighted by the gray area at the bottom of Figure 5.2(a)). The state of this register is modified by the subtracter-module, which decrements the species concentration by one molecule each time it is enabled via `en`-port. The subtracter's inputs are (i) the species concentration ($a_3 \dots a_0$), which is read from the 4-bit register, and (ii) the

Stoichiometry

[5.7] In the following, we refer to reactants only. The same argumentations, and thus implementation-schemes, are valid for implementing stoichiometric product coefficients $\beta_{r,s}$ too – the subtraction is simply replaced by an addition.



(a) 1^{st}-order reactant (b) 2^{nd}-order reactant

Figure 5.2: *How to implement the stoichiometric reactant coefficient in hardware – species concentration update only. These are simplistic schemes where one reaction r and one species s (maximum number of molecules =15) are considered. (a) Scheme that allows a species to represent a 1^{st}-order reactant ($\alpha_{r,s} = 1$). (b) Scheme that allows a species to represent a 2^{nd}-order reactant ($\alpha_{r,s} = 2 \leftrightarrow \alpha_{r,s'} = \alpha_{r,s''} = 1$).*

fixed value "`0001`" ($=b_3 \ldots b_0$) representing the *one* molecule to subtract. As we have mentioned, the operation of the subtracter is triggered by a non-zero signal on the `en`-input. This happens only when the stoichiometric reactant coefficient $\alpha_{r,s}$ is set to one (in the scheme, the switch is set to Vcc-position, as depicted in Figure 5.2(a)), because of the AND-gate. The actual triggering signal is the other input of the AND-gate – the single-bit signal `exeReact` flags the execution of the reaction r by staying at high level for a clock-cycle (see the signal `clk`) as soon as reaction r is executed. In that case, the output of the subtracter is then $o_3 \ldots o_0 = a_3 \ldots a_0 - b_3 \ldots b_0$.

For a reaction r that consumes n molecules of the same species s at once (species s is an n^{th}-order reactant of reaction r), we have multiple (n) simplified stoichiometric reactant coefficients concerning the same species. Again, each of these stoichiometric reactant coefficients enables the subtracter-module, which subtracts one from the register (i.e., a specific location of the c-RAM) of the concerned species. Figure 5.2(b) outlines the scheme for a single reaction r and a single species s, which can be at most a 2^{nd}-order reactant. Differently from the previous scheme, in Figure 5.2(b) there are two switches for the two stoichiometric reactant coefficients $\alpha_{r,s'}$ and $\alpha_{r,s''}$, both related to reaction r and species s. The enabling of the subtraction is dependent on the position of these two switches: when both are set to Gnd, the input of the OR-gate is zero, and thus its output is zero too, disabling in this way the subtracter-module. On the contrary, when one of the two switches is set to Vcc, the subtracter is

enabled (the OR-gate has at least one high-valued input). The triggering of the subtraction-operation again relies on the `exeReact`-signal. In this scheme however, the subtraction can be triggered twice when $\alpha_{r,s'}$ and $\alpha_{r,s''}$ are both set to one (as depicted in Figure 5.2(b)): The delay-module (D-module) delays pulses on `exeReact`-signal by x clock's cycles. This means that, as soon as reaction r occurs, the signal on the subtracter's `en`-input is high for one `clk`-cycle, low for x `clk`-cycles, and then again high for one `clk`-cycle. This makes the subtracter remove twice the value "`0001`" from the concentration register of species s, each time the reaction r occurs.

The x `clk`-cycles delay must be sufficient to cover the latency of the subtracter-module, which depends on the way the subtracter is implemented. The subtracter module embeds a whole logic to perform a subtraction (addition). For this reason, a certain time elapses from when the `en`-signal is applied to when a valid output is available. For example, the latency introduced by a simple n-bit "ripple carry adder" is $2n$ times the gate delay, or n times the gate delay in the case of carry-propagation optimization as proposed in [146] (see also [41] for further details). The adder's latency can be reduced to one `clk`-cycle by parallelizing the logic at the cost of a higher utilization of CLBs and thus of the FPGA (this represents a design choice during the placing and routing phase).

We can extend the explained mechanism to enable updating many species. For example, the scheme in Figure 5.3 implements the update of multiple species concentrations – it includes a reaction r and a set of three species $\mathcal{S} = \{\mathrm{S}_1, \mathrm{S}_2, \mathrm{S}_3\}$, and gives the user the chance to define three reactants. The concentration of the three species is stored as a 4-bit number in three registers (the c-RAM is made of 3 locations of 4 bits each). Depending on stoichiometric reactant coefficients, these concentrations are possibly updated by subtracters once the reaction r occurs – these subtracters are enabled by the triggering `exeReact`-signal and by the stoichiometric reactant coefficients. For example in Figure 5.3, S_3-species represents a 2^{nd}-order reactant of the reaction r and S_2-species a 1^{st}-order reactant, whereas species S_1 is not a reactant. Once the reaction r occurs, the 2-bit down-counter enables sequentially the three "stoichiometric decoders" (the decoders in the top-right gray area). Because the inputs of two out of the three stoichiometric decoders are "`11`" and of the remaining decoder are "`10`", the S_3-related subtracter is enabled twice and the S_2-related subtracter is enabled once. Differently, the subtracter related to species S_1 is never enabled (because no stoichiometric decoder has inputs "`01`" and thus outputs "`0010`", which instead would enable the subtraction of "`0001`" from S_1-register). Once the 2-bit counter has reached the stable state "`00`", the directly-connected decoder produces "`0000`" as output until the reaction r fires again and thus disables all subtracters for the whole reaction interval. Subtracters are also disabled when stoichiometric coefficients are set to "`00`".

In the example in Figure 5.3, we have assumed that the subtracters have unitary latency and thus that no delay between updates of the same species is needed. In reality, the 2-bit counter is replaced by a n-bit counter. Then, by taking LSB and MSB bits of the counter, we enable an arbitrary interval between

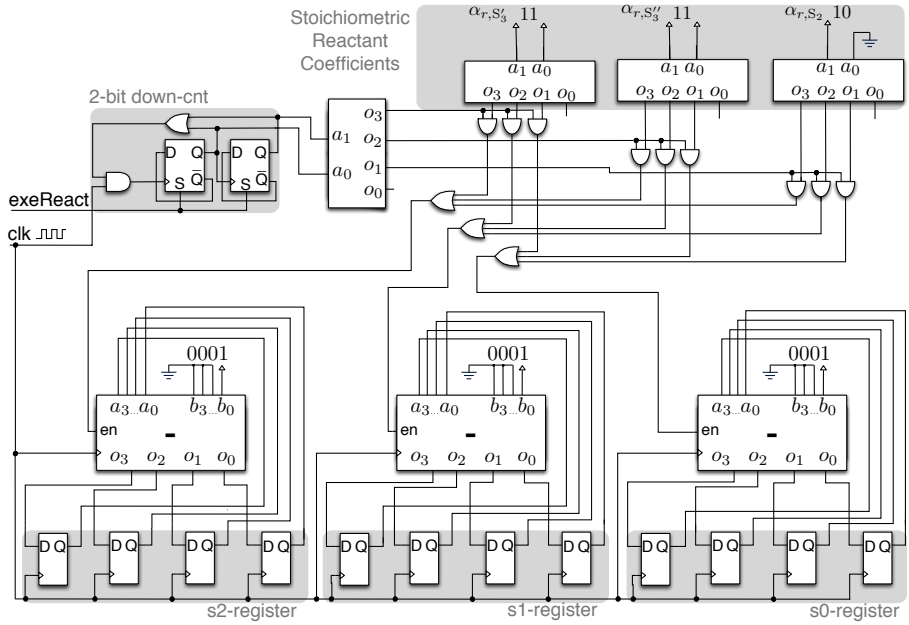


Figure 5.3: *How to generally implement the stoichiometric reactant coefficients in hardware – species concentration update only. The scheme includes one reaction r and three species S_1, S_2, S_3 (maximum number of molecules =15), and it allows three reactants (or fewer higher-order reactants) – with the shown configuration of $\alpha_{r,s}$ we have $r : 2S_3 + S_2 \rightarrow \dots$.*

subtractions in the same species. Furthermore, in the example we have assumed a maximum of three reactants. Instead, by having y as a maximum possible number of reactants, we have a $(\log_2 y)$-bit counter instead of the 2-bit counter.

For the sake of clarity, the scheme reproduces a mechanism where stoichiometric coefficients enable sequentially the concentrations' update. However, we have to perform sequentially only multiple updates of the same species (because we need to access the same register). Instead, we can update in parallel multiple species at the same time (because we access different registers). That is, we can improve the scheme in order to make the update of 1^{st}-order reactants parallel. To this end, the scheme should include also "stoichiometric decoders" that are directly enabled once the reaction occurs (i.e., the `exeReact`-signal feeds directly the AND-gates at decoders' outputs, without passing through the down-counter). Differently from the ones represented in the scheme, these decoders *must* be configured with different coefficients as inputs. For example, a reaction $r : 2{\cdot}\mathrm{S}_3{+}\mathrm{S}_2 \rightarrow \dots$ can be configured via two "delayed stoichiometric decoders" both set to "`11`", and another "direct stoichiometric decoder" set to "`10`". In this way, we can reduce by one third the time required to update

r-dependent species.
We can interpret the stoichiometric matrix as a memory. Reactant and product coefficients are stored in pseudo Read-Only Memories (ROMs) – memories that are not written during the normal execution of the $\mathcal{AC}$ but that can be modified by the users in the configuration phase (even at run-time). Each location contains the address of the concentration register for the species in question, i.e., the right location of the c-RAM. Each reaction with n reactants (or products) calls for n locations in the ROM. As a results, the two stoichiometric reactant and product memories (α- and β-ROM) must have n times $|\mathcal{R}|$ locations, where $|\mathcal{R}|$ is the total number of reactions. For example, referring again to Figure 5.3, we have that the stoichiometric reactant ROM is made of 1 times 3 locations (=1 reaction with the possibility to have 3 reactants) of 2 bits each (=possibility to address 4 species).

Stoichiometric coefficient memories (α- β-ROM)

5.2.2 Defining the dynamics of the chemical model

After having looked at the structure of the $\mathcal{AC}$, we have to focus on its dynamics. We have to realize a distributed/parallel mass-action scheduler that, like the reaction algorithm $\mathcal{A}$ for CPUs, enforces on the system the typical dynamics of a chemical reaction network.
According to the mass-action principle – see (2.6) – the scheduling time for each reaction (next reaction time) is given by the reciprocal of the reaction propensity a_r, namely the product between reactants' concentration and reaction coefficient:[5.8]

$$t_r = 1/a_r = 1/\Big(k_r \prod_{s\in\mathcal{S}} c_s^{\alpha_{r,s}}(t)\Big).$$

We thus require another bank of registers to store the $|\mathcal{R}|$ reaction coefficients, i.e. a pseudo ROM-memory[5.9] with contents defined by the user during the configuration of the $\mathcal{AC}$.

Reaction coefficient memory (k-ROM)

What is also still missing is part of the functionality that the stoichiometric reactant ROM has: selecting which species influence the scheduling of a certain reaction r. The implementation of this is similar to what we have shown previously for updating the species' concentration. The scheme in Figure 5.4 completes the one in Figure 5.3 – each stoichiometric decoder drives the select-inputs of a multiplexer. Based on that, the multiplexer forwards the related 4-bit concentration (for inputs $s_3 \dots s_0 = \texttt{1000}$ it forwards the value of the $s2$-register, for $s_3 \dots s_0 = \texttt{0100}$ the value of the $s1$-register, and for $s_3 \dots s_0 = \texttt{0010}$ the value of the $s0$-register) or the fixed value "0001" (for inputs $s_3 \dots s_0 = \texttt{0000}$). In the example, we have that S_3-concentration is carried by two out of the three output-busses, and S_2-concentration is carried by the remaining output-bus (S_1-concentration is not read from $s0$-register at all).

Stoichiometric *reactant* coefficients

[5.8] We implement in hardware the deterministic mass-action scheduler, see Section 5.5.1 for a discussion in this regard, based on the simplified propensity as introduced in Section 2.4.1.

[5.9] "Pseudo ROM" has the same meaning previously explained for α- and β-ROMs.

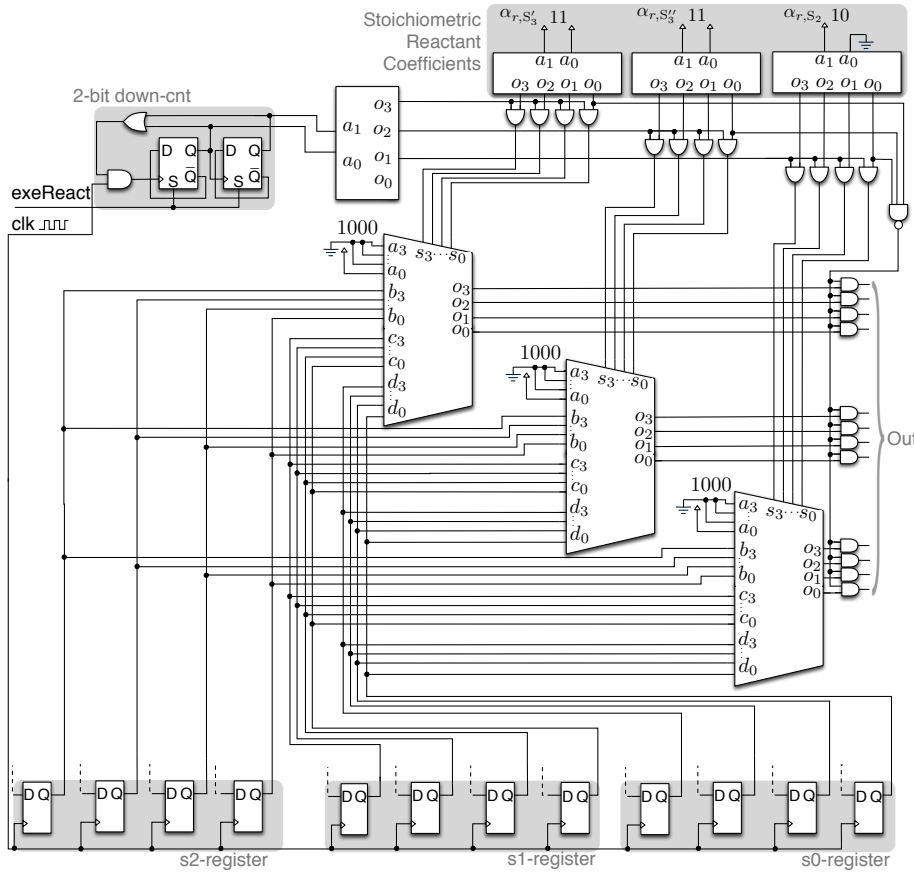


Figure 5.4: *How to generally implement the stoichiometric reactant coefficients in hardware – species-concentration selection for reaction-time computations. The scheme includes one reaction r and three species S_1, S_2, S_3 (maximum number of molecules =15), and it allows three reactants (or fewer higher-order reactants) – with the shown configuration of $\alpha_{r,s}$ we have $r : 2S_3 + S_2 \rightarrow \dots$.*

It is possible to define fewer stoichiometric reactant coefficients than the maximum provided for. If a $\alpha_{r,s}$-coefficient is set to "00", the related multiplexer selects and forwards the fixed value "0001", which is the identity element of the multiplication required to calculate the propensity. If none of the $\alpha_{r,s}$-coefficients is set ($\alpha_{r,s} = 00 \,\forall\, s$) then all the three output busses carry "0000", thus the calculated propensity will be zero too (because of the conjunct action of the NAND-gate and the "output AND-gates").

Reaction timers

Implementing timers in hardware is a very simple, economic task. Timers are registers (n-bit register = bank of n FFs) that, once initialized to a certain (integer) value, are decremented by one at either each clock-edge, each clock-

cycle, or each multiple of the clock-cycle. In hardware, we can obtain a very fine granularity of timers, with minimum times being as small as the clock's period (or even as half-period of the clock) – e.g., working at the frequency of 40 MHz (which represents a low value for a frequency in FPGA-technology!), we already obtain a granularity of 25 ns. The timer-accuracy does not represent a problem as it depends on the accuracy of the clock generator – e.g., working with an economic quartz (< 0.10 \$) we already have an accuracy of ± 50 ppm, which is much better than what is actually required in our applications and what is obtained relying on operating systems' interrupts in PCs. Timer-registers must be completed with comparator-modules that give indication about their expiration. Comparators are simple modules that produces a high-level on their 1-bit output when a certain relationship among its n-bit inputs exists. In this regard, we recall that equality is much less expensive than majority or minority conditions, and thus it represents the first design choice. For example, to implement the logic function $A = B$, for A and B being 4-bit signals, we need 4 XOR and one NOR gates (5 gates in total). Differently, implementing the logic functions $A < B$ and $A > B$ requires a much more complex circuitry made of 4 NAND, 23 AND, and 4 NOR gates (31 gates in total – see for example data-sheet [227]).

Reaction-time computation...

...floating-point representation

The calculation of reaction times represents a critical task that calls for multiplications and divisions. These operations, besides being expensive in terms of logic utilization and computational speed, lead to deal with high-dynamic values and signals. The challenge is multiplying low or high concentration-values and then dividing intermediate results in order to obtain times from propensities, without needing too many bits and without getting into significant truncation errors. Although not the most efficient solution, we opt for the floating-point representation. In this way, (i) we work with fixed sized variables and signals, e.g., for a single-precision floating-point representation we have to bring signals via 32-wire busses and store variables in 32-bit registers (see IEEE-754 Standard [116]), and (ii) we obtain a wide dynamic range and a good resolution of variables during the time computation (e.g., in the single-precision floating-point representation, we can represent numbers in the range $\sim \pm 2^{127}$ with a resolution of $\sim 2^{-23}$ the number represented [116]). The timer computation for a reaction r of at most x^{th}-order requires $(x-1)$ multiplications among the reactant concentrations, another multiplication to account for the reaction coefficient, and a division (reciprocal) to calculate the time from the propensity value. To resize the remaining times of those reactions that are still pending for execution, we further need a division to scale the old propensity value by the new one, and a final multiplication. Eventually, we have to interface the integer concentrations and times,[5.10] and the floating-point intermediate signals of the time computation.

...need for multiplications, divisions, and conversions

Most of the described operations can either be parallelized and performed by

[5.10] The realization of timers that work in floating-point representation would be a (pointless) effort. It is much more convenient to resort to conversions.

...some operations can be parallel

others require sequentiality

many modules, to achieve fast computations at the expense of a high logic utilization, or they can be performed in pipeline by exploiting one or few modules only. For example, the calculation of the propensity can be performed (*i*) sequentially by one multiplier, (*ii*) in parallel by using as many multipliers as the number of multiplications required, or (*iii*) by any intermediate solution that represents a tradeoff between efficiency and logic density / routing requirement. Other operations must be sequential and have to wait the worth of the latency of the upstream module providing them the inputs, e.g. the time-scaling must be subsequent to the remaining-time conversion and the new-propensity calculation.
The computation of reaction-times represents the most costly operation: it can either be performed independently for each reaction, in order to benefit the speed, or be computed sequentially by a single "`time-Calculator`"-module (in pipeline), in order to benefit the logic utilization.

5.3 Artificial chemistries on Xilinx Spartan-6 XC6SLX9 FPGA

By following the pipelining approach, we have implemented the whole $\mathcal{AC}$ on a relatively small, low-cost device: the Xilinx Spartan-6 XC6SLX9 FPGA (see [254] for a general overview of its features) mounted on the Avnet Spartan-6 LX9 MicroBoard [19]. Specifically, we comment on the schematics of the three main parts (modules) that constitute the $\mathcal{AC}$-platform: the `main`-module that manages the clock and connects the `AC`-module to the external peripherals (Section 5.3.1), the `AC`-module that implements the $\mathcal{AC}$ (Section 5.3.2), and the `time-Calculator`-module that serves to calculate the inter-reaction times (Section 5.3.3). Figure 5.5 illustrates the main components of the implemented middleware platform for programming chemical algorithms on hardware.
This section is meant to provide an in-depth treatment concerning the hardware platform we developed. In order to make the explanation understandable, we have had to simplify and/or omit some parts. Nonetheless, this section contains sufficient information such that system architects are able to reproduce their own $\mathcal{AC}$-platform in hardware. Our platform can be used to implement "arbitrary" $\mathcal{AC}$s without such a thorough knowledge. To this end, the first part of Section 5.3.2 is sufficient.
All presented schematics have been obtained/simplified relying on the output produced through Xilinx-ISE RTL-schematic tool [253], which translates the VHDL code into a graphical representation of what is synthesized in hardware. For the sake of clarity, although the implementation includes only logic primitives (e.g., logic gates, flip-flops, *etc.*), the figures shown in the next sections additionally include macro-blocks (modules) that represent logical functionalities (e.g., coding/decoding, managing of RAM memories, *etc.*). These macro-blocks are indicated with a square symbol whose apices are marked with triangles. Busses of wires are represented as double lines and modules' ports are identified

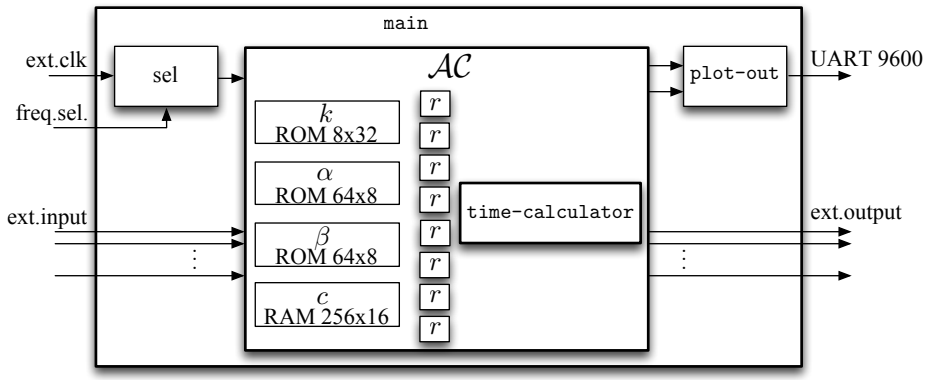


Figure 5.5: *Block diagram illustrating the main components of the middleware platform for programming chemical algorithms on Xilinx Spartan-6 XC6SLX9 FPGA.*

by name and size.

5.3.1 `main`-module

The `main`-module serves to connect the `AC`-module to the external peripherals (e.g., serial output via UART, input and output ports) and to manage the clock.

Working-frequency selection

The external clock is synthesized through a device-primitive by Xilinx, named "`clk-Synt`" in the schematic in Figure 5.6. This makes possible to select at run-time via an external switch (the `SW(3)`-switch that drives the `bufgmux`-multiplexer) between two frequencies at which the $\mathcal{AC}$ works (this directly affects the performance in terms of speed and power consumption).

Input and output peripherals

Any external event on port `EXT_EVENT`, such as a button being pressed or a specific triggering signal, is passed to the `AC`-module. Module `plot-out` includes the logic to send two 16-bit integer values via UART interface (which we use to send out and plot at run time on an external device – e.g, PC's monitor – a couple of selectable molecular concentrations). Note that the mentioned frequency selection does not affect `plot-out`-module, which is designed to constantly send data at a rate of 9600 bps.

5.3.2 `AC`-module

This module represents the principal part concerning the implementation of $\mathcal{AC}$ in hardware.

For the sake of clarity, we do not show the clock-inputs of modules and present a simplified schematic which consists of two reactions only. The schematic includes macro blocks whose functionality is explained without details of their implementation. Additionally, we do not show in the schematic how concen-

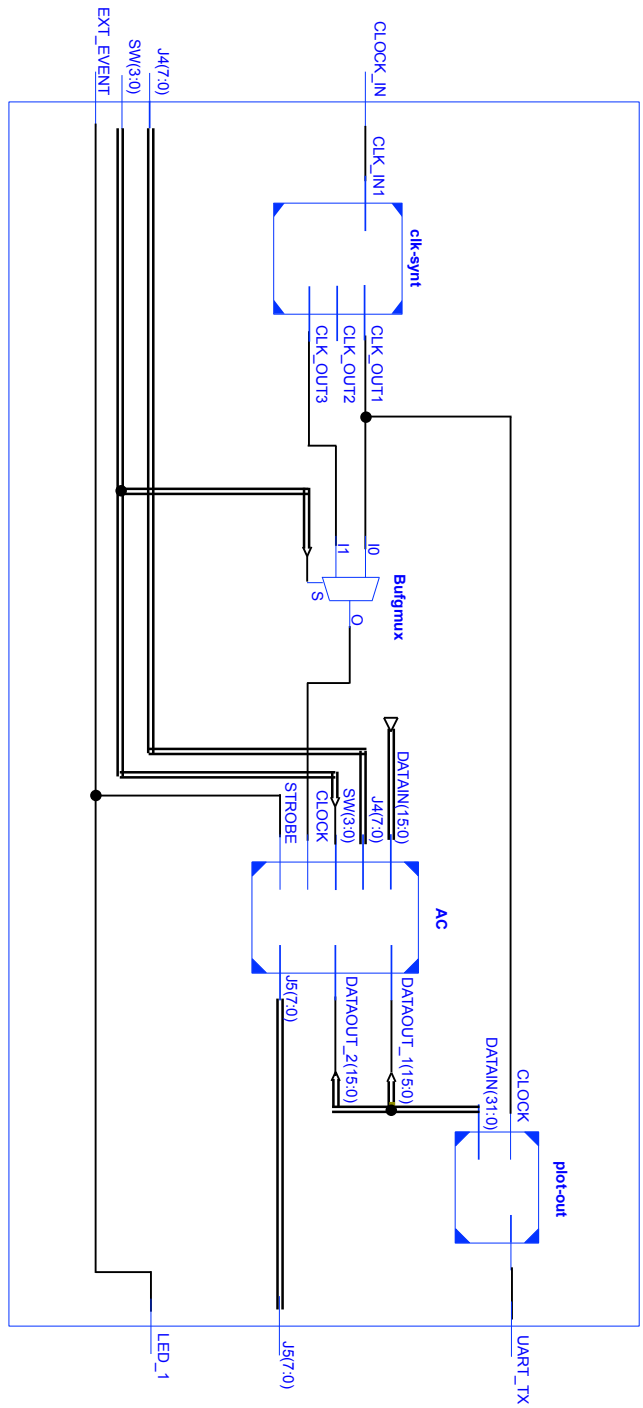


Figure 5.6: *Schematic: main-module.*

tration values are sent out to output ports DATAOUT1 and DATAOUT2, and we remove the ports that are not necessary for the AC-module's main functioning. Note further that all operations are synchronous (at clock's edge), and all multiplexing operations are followed by a latch that makes the output stable, although not represented in the schematic.

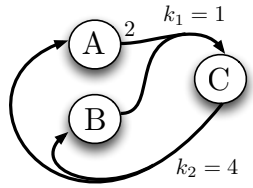
$\mathcal{AC}$ configuration (controller configuration)

The definition/configuration of the chemical model takes places through the configuration of the memory c-RAM, which stores 16-bit values of all molecular concentrations, and memories α-ROM and β-ROM, which store the stoichiometric reactant and product coefficients. Specifically, the stoichiometric memories contain for each reaction the address to access the concentration of dependent species, whose value is stored in the c-RAM (refer back to Figure 5.3 and Figure 5.4 for an explanation of the implemented addressing-mechanism). As represented in Figure 5.7, the stoichiometric memories have $\log_2(|\mathcal{S}|+1)$-bit width and $(|\Xi| \cdot |\mathcal{R}|)$-location depth, where $|\mathcal{S}|$ is the maximum allowed number of molecular species that can constitute a chemical model, $|\Xi|$ (by abusing the notation) is the maximum number of reactants and of products per each reaction, and $|\mathcal{R}|$ is the maximum number of reactions that can define a chemical model. Molecular concentrations of chemical species are stored in the c-RAM as 16-bit integer values. Memory c-RAM has 16-bit width and $(|\mathcal{S}|+1)$-location depth, where the first 16-bit location is reserved and set to "0...01", whereas the remaining $|\mathcal{S}|$ locations are initialized to zero unless configured otherwise. The α- and β- memories are also initialized to zero so as to point to the first (reserved) location of the c-RAM (i.e., with the address x0..0), unless configured otherwise according to the definition of the chemical model.

To illustrate how to define and configure an $\mathcal{AC}$, we consider the simple chemical model represented in Figure 5.7(a), which is composed by the species set $\mathcal{S} = \{\mathrm{A}, \mathrm{B}, \mathrm{C}\}$, and the reaction set $\mathcal{R} = \{r_1, r_2\}$ with $r_1 : 2\mathrm{A} + \mathrm{B} \xrightarrow{k_1} \mathrm{C}$ and $r_2 : \mathrm{C} \xrightarrow{k_2} \mathrm{A} + \mathrm{B}$, where $k_1 = 1$ $(\mathrm{mol}^2{\cdot}\mathrm{s})^{-1}$ and $k_2 = 4$ s^{-1}, and the initial concentration of species A is $c_\mathrm{A} = 100$ whereas species B and C are initially set to 0 molecules. All locations of the c-RAM have to be set to zero, except for c-RAM(0), which is reserved and set to x0001, and c-RAM(1), which contains the concentration value c_A of the A-species, i.e., x0064. Locations c-RAM(2) and c-RAM(3) are thought to contain the other two species concentrations c_B and c_C, as depicted in Figure 5.7(b).[5.11] Stoichiometric reactant coefficients ($\alpha_{r_1,\mathrm{A}} = 2$, $\alpha_{r_1,\mathrm{B}} = 1$, $\alpha_{r_1,\mathrm{C}} = 0$, and $\alpha_{r_2,\mathrm{A}} = 0$, $\alpha_{r_2,\mathrm{B}} = 0$ and $\alpha_{r_2,\mathrm{C}} = 1$) are stored in the α-ROM that, by assuming $|\mathcal{S}| = 255$, contains x01, x01, x02, in the first three locations, and x03 in the $(1{\cdot}|\Xi|)$-th location, as depicted in Figure 5.7(c). Similarly, stoichiometric product coefficients ($\beta_{r_1,\mathrm{A}} = 0$, $\beta_{r_1,\mathrm{B}} = 0$, $\beta_{r_1,\mathrm{C}} = 1$, and $\beta_{r_2,\mathrm{A}} = 1$, $\beta_{r_2,\mathrm{B}} = 1$ and $\beta_{r_2,\mathrm{C}} = 0$) are stored in the β-ROM that thus contains x03 in the first location, x01 in the $(1{\cdot}|\Xi|)$-th location, and x02 in the $(1{\cdot}|\Xi|+1)$-th location, as depicted in Figure 5.7(d).

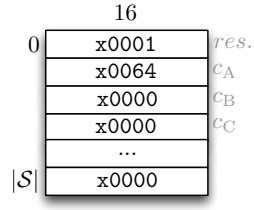
Reaction coefficients are stored in the k-ROM. To reduce the complexity of

[5.11] In this example, we have arbitrary decided to assign locations of the c-RAM in alphabetical order of species' names. Other schemes can be followed.

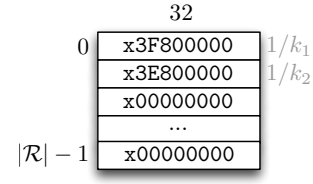


r_1 : $2\mathrm{A} + \mathrm{B} \xrightarrow{1} \mathrm{C}$
r_2 : $\mathrm{C} \xrightarrow{4} \mathrm{A} + \mathrm{B}$
$c_\mathrm{A}(\mathrm{t}{=}0){=}100, \quad c_\mathrm{B}(\mathrm{t}{=}0){=}\, c_\mathrm{C}(\mathrm{t}{=}0){=}0$

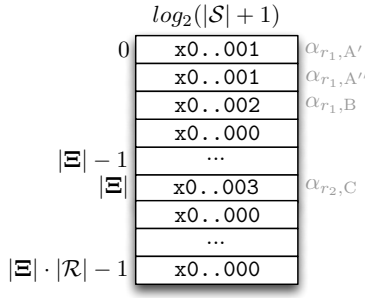
(a) Chemical Model



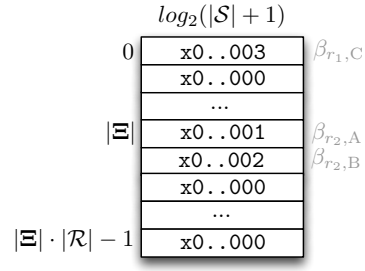
(b) `c-RAM`



(e) `k-ROM`



(c) α-`ROM`



(d) β-`ROM`

Figure 5.7: *Example of an $\mathcal{AC}$ configuration (configuration of memories). The $\mathcal{AC}$ whose model is depicted in Figure 5.7(a) is implemented on an FPGA by configuration of the* `c-RAM` *(Figure 5.7(a)), which keeps also track of the run-time state of the $\mathcal{AC}$, and by configuration of stoichiometric reactant and product coefficients stored respectively in* α`-ROM` *and* β`-ROM` *(Figure 5.7(c) and Figure 5.7(d)) and of reaction coefficients stored in* `k-ROM` *(Figure 5.7(e)).*

module `time-Calculator`, the `k-ROM` stores the $|\mathcal{R}|$ 32-bit single-precision floating-point values of the reciprocal of the reaction coefficient of each reaction. Referring back to the example in Figure 5.7, the `k-ROM` has the first location `k-ROM(0)` initialized to `x3F800000`, which represents the value $1/k_1 = 1$ in float notation, and `k-ROM(1)` initialized to `x3E800000`, which represents the value $1/k_2 = 0.25$ in float notation, as depicted in Figure 5.7(e).

Via the `STROBE`-port, an external event is signaled to the $\mathcal{AC}$ (refer to the `AC`-module's scheme in Figure 5.8). There, the macro-block `c-Updater` is responsible for updating the concentrations. This updating is performed whenever an external event occurs or a reaction fires. In the case of an external event, the concerned species is (are) updated according to the value passed via the `DATAIN`-port. In the case of a firing reaction, the `r-Decoder` identifies the corresponding index r_i. Based on that, the `c-Updater` decreases/increases by one (decreases in the case of a reactant species, or increases in the case of a product species) those concentrations that are addressed by contents of stoichiometric memories' locations in the range $[i, i+|\Xi|)$. This update is sequential.[5.12]

`AC`-module functioning

Once the `c-Updater` has updated all concentrations, it triggers the module `time-Processor` through the output port `rdy`. The `time-Processor`-module contains a counter that goes through all reactions, reads from the `c-RAM` and passes to the `time-Calculator` the values of reactants' concentrations (those addressed by contents of the α`-ROM` only) and schedules the calculation of the next-reaction-time, for all reactions. The next section explains in detail the implementation of the `time-Calculator`.

The calculations of the reaction times are performed by a single module. This means that these calculations must be performed sequentially, making in this way the performances of the system (the maximum speed at which the chemical system can operate and interact with external events) dependent upon the maximum number of reactions. The use of as many `time-Calculator` as the number $|\mathcal{R}|$ of reactions reduces the total computational time to the time required to calculate one reaction only (and it simplifies the implementation too), at the cost of a higher logic utilization.

Via the `startTimer-Decoder`, the reactions' timers `r1-Timer` and `r2-Timer` are triggered. These modules are mere down counters, which are set to the value calculated by the `time-Calculator` (the 32-bit signal on the input `expTime`-port of timers represents the integer value of the next-reaction time) and decremented every clock (e.g., every 25 ns @ 40 MHz). The expiration of the reaction timer is captured by a 32-bit equality-comparator whose output – the normally-low level signal `exeReact` – rises to high level as soon as the timer reaches the value "`00...0`", and it remains constant for a clock duration. `r-Timer`-modules are activated by one-clock-long high level on its `strobe`-port, which triggers the count-down.

[5.12] For each reaction r_i, the addresses of the species to update are extracted from the locations of α- and β-ROMs in the range $[i, i+|\Xi|)$. Each address is used to read the concentration-value from the c-RAM, update it, and store again it in the RAM. For details on the updating mechanism, refer back to Section 5.2.1.

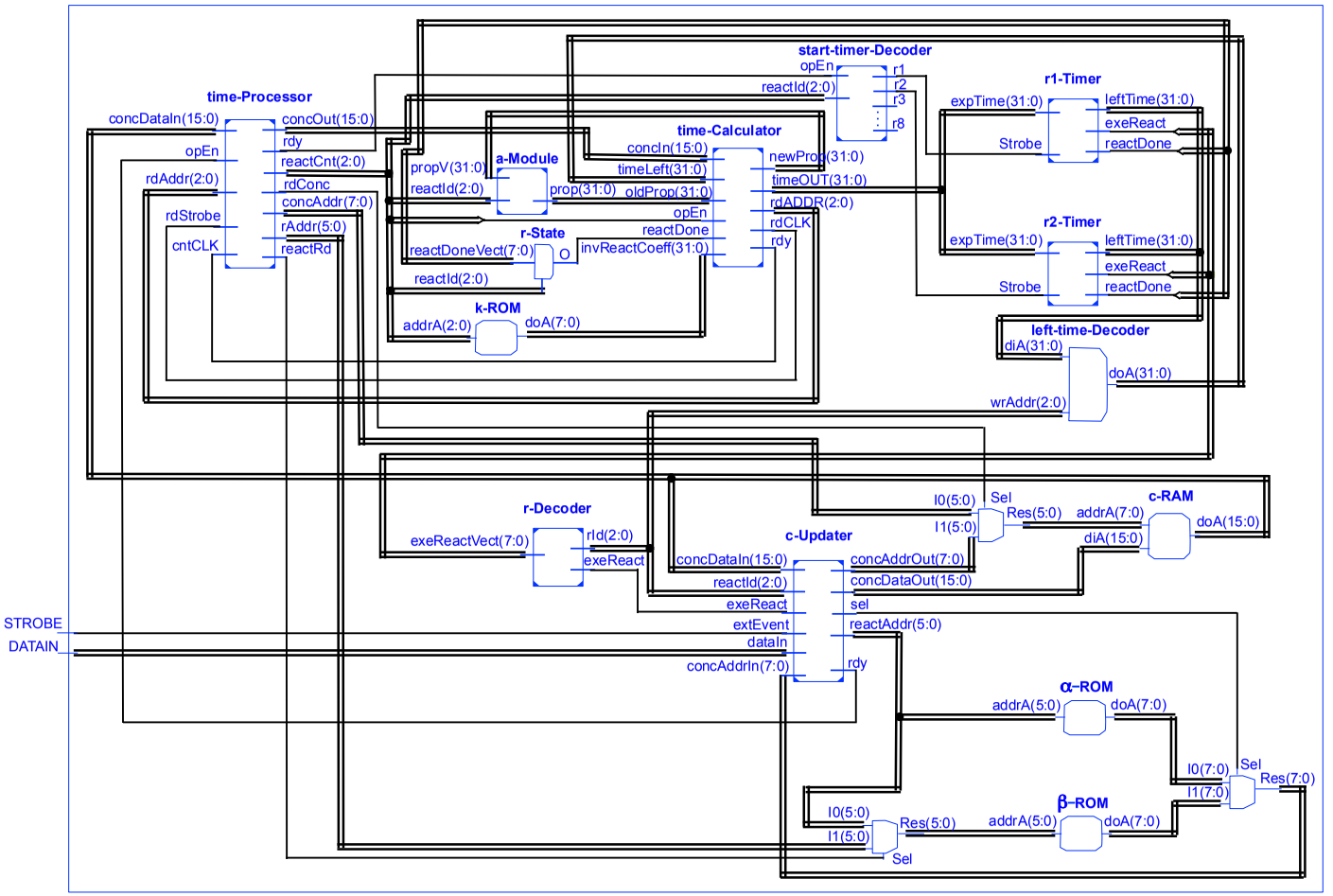


Figure 5.8: *Simplified schematic of the AC-module. Sizes of all busses and ports refer to the case $|\mathcal{S}|$=256, $|\mathcal{R}|$=8, $|\Xi|$=8 (concentration values expressed as 16-bit integer values). We represent two reactions only.*

The normally-high level signal `reactDone` remains high until a new reaction is scheduled. Each bit b_i of the corresponding `reactDoneVect`-signal gives information about the state of the corresponding reaction r_i: executed (i.e., high level) or still pending for execution (i.e., low level). Similarly, the `exeReactVect`-signal consists of $|\mathcal{R}|$ bits. Each high level of a certain bit b_i of this signal triggers all that processes that constitute the execution of the corresponding reaction r_i (i.e., concentration update and timer computation).
Although not represented in the schematic, `r-Timer`-modules are "frozen" once an event occurs (i.e., `r-Timer`-modules' clocks are suspended at the rising edge either of the `extEvent`-signal or of any `exeReact`-signal, and enabled again at the rising edge of any `strobe`-signal). This enables to correctly deal with almost concurrent events.
Finally, the `r-State`-decoder extracts the state of each reaction from the $|\mathcal{R}|$-bit `reactDoneVect`-signal. The `a-Module` stores the reciprocal values of propensities of all reactions as 32-bit (single-precision) floating-point values (the memory has therefore 32-bit width and ($|\mathcal{R}|$)-location depth), at each computation of reaction-times.[5.13] The `left-time-Decoder` extracts the integer value representing the remaining time of a selected reaction, and passes it to the `time-Calculator`.

5.3.3 `time-Calculator`-module

The last principal module is the `time-Calculator`, which computes the propensity of a reaction and produces on its output port `timeOUT` the 32-bit signal that represents the (integer) value of the time at which the reaction should be executed. Again, for the sake of clarity, in the schematic in Figure 5.9, we remove all ports and signals related to the clock of modules.
We have taken advantage of the Xilinx floating-point operator logic-core [251] to implement the modules constituting the `time-Calculator`. As mentioned, we have adopted the standard single-precision floating-point representation described in [251] (32 bits, with a 23-bit fraction and 8-bit exponent), which complies with the IEEE-754 Standard [116].
The inputs to this `time-Calculator`-module are (*i*) the 32-bit `oldProp`-signal that contains the floating-point value of the reciprocal propensity stored at the last time-computation, (*ii*) the 32-bit `invReactCoeff`-signal that represents the floating-point value of the reciprocal of the reaction coefficient, (*iii*) the 16-bit `concentrationIN`-signal that contains the integer value of reactants' concentration, (*iv*) the 32-bit `timeLeft`-signal that represents the integer value of the remaining time since the triggering event has occurred until the planned reaction execution, (*v*) the 1-bit flag `opNd` that triggers the whole `time-Calculator`, and finally (*vi*) the 1-bit flag `reactDone` that indicates if the next-reaction-time must be completely recalculated (in the case of a just-executed reaction)

Input signals

[5.13] We indicate `a-Module` as a module and not as a RAM because it contains also an input multiplexer, an output demultiplexer, and the inputs for enabling writing and reading operations. We remove these details from the scheme to make it better readable.

or instead, it must be rescaled based on `oldProp`- and `timeLeft`- signals (in the case of a reaction still pending for execution).

Interfacing float- and integer-world

Module `time-Calculator` performs all computations in floating-point. Thus, it requires (*i*) an integer to float converter, named "`int16Tof`", that transforms the 16-bit integer signal from the `concentrationIN`-input into a 32-bit floating-point signal, (*ii*) another integer to float converter, "`intTof`", that transforms the 32-bit integer signal from the `timeLeft`-input into a 32-bit floating-point signal, and on the output side, (*iii*) a float to integer converter, "`floatToInt`", that transforms the 32-bit floating-point time signal into a 32-bit integer time signal.

Propensity calculation

The time-computation process starts with the multiplication of reactants' concentrations. To reduce the logic utilization (at the cost of a lower computational speed), we iteratively use one floating-point multiplier module, "`fMult1`", to perform this calculation. For this reason, the schematic includes the `Cnt`-counter, which increments its output by one after the latency period required to (*i*) obtain the 16-bit `concentrationIN`-value once addressed through signal on `rdADDR`-output, (*ii*) convert that 16-bit integer signal to a 32-bit floating-point signal, and (*iii*) multiply the last `fMult1`-module's result by the converted version of the `concentrationIN`-input. At the first iteration, when the `time-Calculator` is triggered on the `opNd`-port, the input of the `fMult1`-module is fixed to one (see `mux1`-multiplexer).[5.14] The value (constant) `|Ξ|` defines the counting end.

Once the multiplication of all `|Ξ|` reactants' concentrations has been performed, the new reciprocal propensity is calculated through the floating-point divisor module `fDiv1`. This operation accounts for the reaction coefficient, whose reciprocal value constitutes the dividend. The output result of the `fDiv1`-module is passed via port `newProp` to the `AC`-module (where it is stored).

Time rescaling

According to the `reactDone`-flag (which is high in the case of an executed reaction), the result of module `fDiv1` is either (*i*) used directly as the floating-point value representing the next-reaction-time, or (*ii*) rescaled according to the values of the old propensity and the remaining time. The rescaling operation requires further (*i*) a floating-point divisor module `fDiv2`, which divides the new-propensity's reciprocal value (`fDiv1`-module's result) by the old propensity reciprocal value (`oldProp`), and (*ii*) a floating-point multiplier module `fMul2`, which scales this propensity ratio (`fDiv2`-module's result) by the value of the remaining time (`timeLeft`). Finally, the floating-point value of the reaction-time, either calculated from scratch or rescaled based on previous values, is converted to the 32-bit integer `timeOUT`-signal and passed to the `AC`-module illustrated in the previous section. Both calculation-from-scratch and rescaling operations lead to the same computational latency as the final conversion is triggered by the `rdy`-signal from module `fMul2`.

[5.14] This is done to avoid the first iteration of the `fMult1`-module to depend on its last (unknown) result.

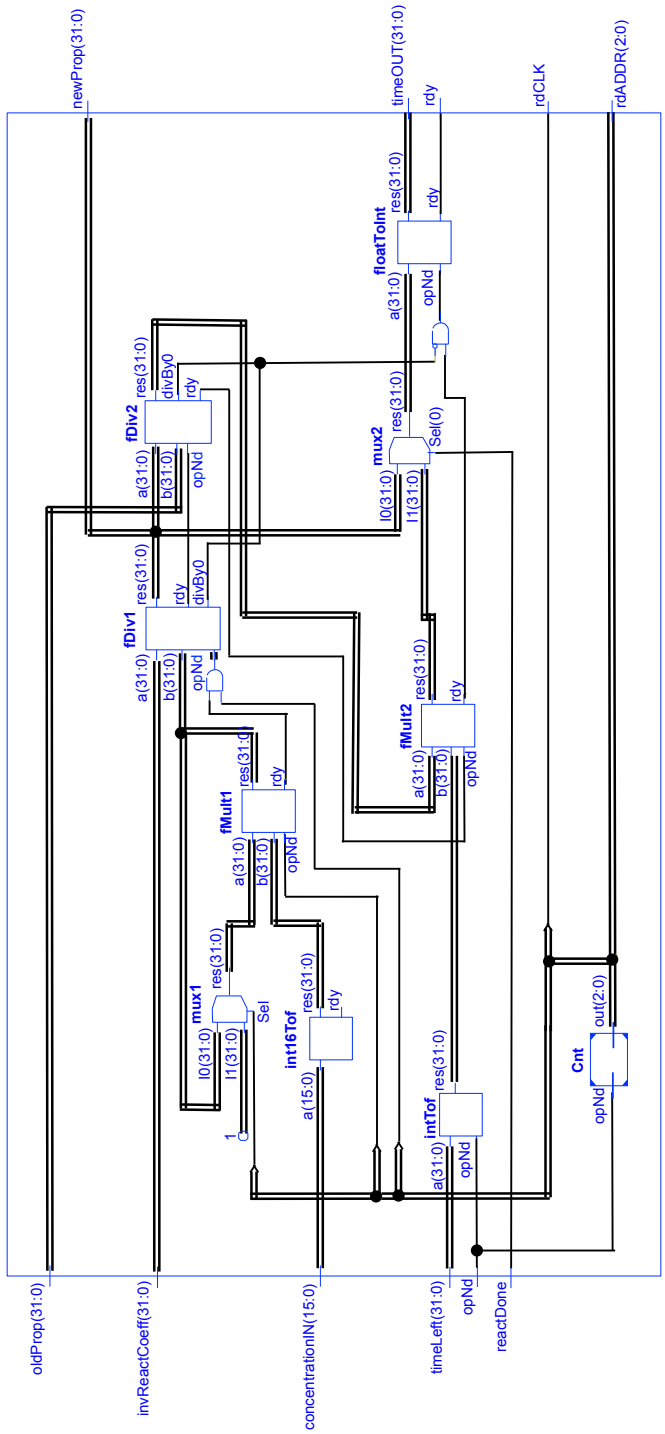


Figure 5.9: *Schematic: time-Calculator.*

5.4 Experimenting with Chemistry-inspired hardware controllers

The following experiments were performed by implementing the $\mathcal{AC}$-platform on Xilinx Spartan-6 XC6SLX9 FPGA, mounted on Avnet Spartan-6 LX9 MicroBoard. The supported $\mathcal{AC}$ had up to $|\mathcal{S}| = 256$ -1 species and $|\mathcal{R}| = 8$ reactions, each of them characterized by $|\Xi| = 8$ reactants and products.

5.4.1 Enzymatic control of Internet traffic with FPGAs

We configured the $\mathcal{AC}$ according to the Enzymatic model (see Section 3.3) and realized a runtime-programmable plug-and-play module to control the rate of a certain class of traffic in a real-world network.
Generally, the main bottlenecks for developing high-speed traffic management solutions are the following: (*i*) The need for external memory bandwidth in order to buffer data packets. (*ii*) The need for processing real-time complex scheduling algorithms in hardware. We left out the first aspect. Rather, we exploited the FPGA-platform described in Section 5.3 and configured it in order to realize a traffic controller. The key challenges of designing such a controller, especially on a hardware module, are (*i*) deciding the flows that require pacing, and (*ii*) determining the appropriate pacing rate [9]. We did not cover the aspects of packet inspection and identification. Rather, we realized in hardware a fast-processing controller able to implement concurrent actions and decisions. To this end, we simply needed to configure the introduced hardware-platform for $\mathcal{AC}$s, with the settings related to the Enzymatic rate controller introduced in Section 3.3.
In this experiment, whose setup is outlined in Figure 5.10, we have used the Enzymatic dynamics to control the outgoing traffic of a standard computer (Linux, Kernel 3.8.6). Practically, the chemical reaction network on the FPGA constitutes an external manager module that controls the service process of the egress link queue or a certain class of traffic. The scheduling policy of such a queue is non-work-conserving (i.e., the queue is not served as soon as possible). Rather, the reaction system (implemented on the FPGA) triggers the service of the computer's queue. Each time a packet, and thus a certain amount of bytes, arrives at the egress queue, the packet is enqueued waiting to be transmitted, and the reaction system on the FPGA is notified. As soon as a specific reaction (the output reaction) of the chemical system fires, a fixed amount of bytes is allowed to be dequeued in the Linux computer. When this amount is enough to cover the size of the packet at the head of the queue, that packet is dequeued and transmitted (FIFO-policy).
In the system that we have implemented, the communication between computer and FPGA exploits the parallel port, on the computer's side, and one of the external connectors, on the LX9-board's side. To this end, we have extended and used the Parapin kernel module [75] in order to use the PC's parallel port

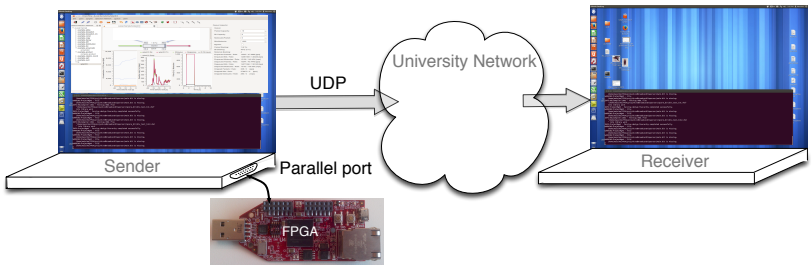


Figure 5.10: *Setup of the experiment on hardware traffic rate control: The outgoing traffic of a standard computer was controlled via an Enzymatic rate controller implemented on FPGA. The communication between computer and FPGA exploited the parallel port on the computer's side (as a connector only, and not as a communication standard), and one of the external connectors on the LX9-board's side.*

as a generic digital I/O interface: (*i*) handling the interrupts of specific parallel-port's pins, and (*ii*) writing directly to the parallel port registers. Each state transition of a specific pin of the parallel port generates an interrupt, which then activates the functionalities in the kernel-space to account for the new amount of bytes authorized to be sent, and to possibly dequeue and send the queue-head packet. Vice-versa, a state-change of a specific pin of the LX9-board's connector (which is induced by the state transition of the connected parallel port pin) triggers the update of a concentration (input concentration) in the reaction system. As we let every 1 KByte of enqueued traffic be notified and we had 100 ns as a minimum time between two consecutive transitions, the maximum manageable load was 10 GBps.

We configured the Enzymatic rate controller according to the $\mathcal{AC}$ presented in Section 3.3 and summarized in Figure 5.11. We had four species only, to be initialized as $c^0_{\mathrm{S}} = c^0_{\mathrm{ES}} = c^0_{\mathrm{P}} = 0$, $c^0_{\mathrm{E}} = e_0$.[5.15] We decided to fix the rate limit to 0.5 GBps and to create an S-molecule for each KByte of data, and dually to allow the transmission of 1 KByte for each produced P-molecule. Thus, we set $e_0 = 25$ Kmol. To this end, we set the third element of the `c-RAM` to `x61A8`, and left the other fields equal to `x0000`, except for the first (reserved) field which was equal to `x0001`.[5.16] According to Figure 5.11(a), the stoichiometric matrix is

$$\boldsymbol{\Xi} = \begin{bmatrix} -1 & 0 \\ -1 & 1 \\ 1 & -1 \\ 0 & 1 \end{bmatrix}.$$

[5.15] It does not really matter how we initialized species S and P, and how we distributed the mass e_0 among species E and ES. The stable state depends on the e_0-value only (and reaction coefficients k_1 and k_2). It is important that $c^0_{\mathrm{E}} + c^0_{\mathrm{ES}} = e_0$.

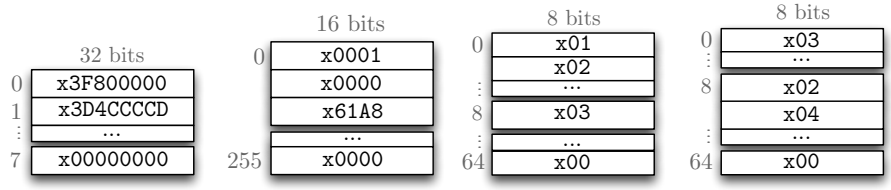
[5.16] We arbitrary decided to assign the `c-RAM`-locations according to the species' order we used in the definition in Figure 5.11(a).

$$\mathcal{AC} = \Big\{ \{\mathrm{S}, \mathrm{E}, \mathrm{ES}, \mathrm{P}\},\ \{r_1, r_2\},\ \mathcal{A} \Big\}$$

$$r_1: \quad \mathrm{S} + \mathrm{E} \xrightarrow{1} \mathrm{ES} \qquad r_2: \quad \mathrm{ES} \xrightarrow{20} \mathrm{E} + \mathrm{P}$$

$$c^0_{\mathrm{E}} = e_0 = 25000 \qquad c^0_{\mathrm{S}} = c^0_{\mathrm{ES}} = c^0_{\mathrm{P}} = 0$$

(a) Chemical Model



(b) `k-ROM` (c) `c-RAM` (d) α-`ROM` (e) β-`ROM`

Figure 5.11: *FPGA-configuration for traffic rate control (configuration of the memories for an Enzymatic rate controller). The $\mathcal{AC}$ whose model is formalized in Figure 5.11(a) was configured on an FPGA through reaction coefficients stored in `k-ROM` (Figure 5.11(b)), and stoichiometric reactant and product coefficients stored respectively in α`-ROM` and β`-ROM` (Figure 5.11(d) and Figure 5.11(e)); it was initialized through the `c-RAM` (Figure 5.11(c)).*

Thus, by considering the reactions in the order we have presented in Figure 5.11(a), we set the stoichiometric reaction coefficients as α`-ROM(0)= x01`, α`-ROM(1)= x02` and α`-ROM(8)= x03`, whereas the stoichiometric product coefficients as β`-ROM(0)= x03`, β`-ROM(8)= x02` and β`-ROM(9)= x04`.

The results shown in Figure 5.12 refer to a scenario where the UPD traffic was bursty (the traffic was generated via "Iperf" tool). In a first phase (in the first 10 s), the load (green-continuous line) was enough to exceed the predefined threshold (red-dashed line). In this phase, we can recognize the effect of the control mechanism of the Enzymatic rate controller that guaranteed the egress rate (blue-continuous line) to respect the threshold. In a second phase ($\sim$ 14-21 s), the load was still bursty but this time, it was on average lower than the predefined rate limit. In this case, the Enzymatic rate controller allowed the output rate to follow on average the load but filtered out its spikes, contributing in this way to have a more "stable" output.

We were able to configure the behavior of the controller post deployment. We modified the rate limit to different values by changing the e_0 value loaded in the third location of the *c*-`RAM`, e.g., `xC350` instead of `x61A8` to fix the maximum rate to 1 GBps. We changed also the filtering effect of the traffic controller by modifying the coefficient k_2 and accordingly e_0 amount as

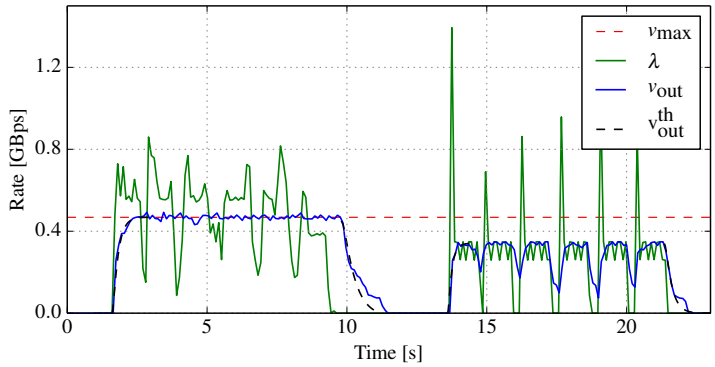


Figure 5.12: *Results of the experiment on UDP-traffic control via the hardware Enzymatic-controller formalized in Figure 5.11. v_{max} represents the limit we predefined by means of parameters $e_0 = 25$ Kmol and $k_2 = 20\ s^{-1}$ (1 KByte was mapped to 1 molecule and vice-versa), λ is the load presented at the network layer, v_{out} is the actual traffic that the Enzymatic reaction allowed, and v^{th}_{out} is the theoretical trajectory derived from analysis in Section 3.3.3.3.*

$v_{\max}/k_2$. Figure 5.13 shows two output trajectories $v_{\text{out}}(t)$ experienced in the latter experiment, when we halved the coefficient k_2 (set the second location of k-`ROM` to `0x3DCCCCCD`), and accordingly doubled e_0 in order to maintain the rate limit to 0.5 GBps by setting the third location of the c-`RAM` to `x61A8`.

As we have seen in Section 3.3, the same experiment could be repeated also by implementing a configurable chemical reaction network as a Linux-kernel module that controls the service process of the egress queue. For the experiments presented in Section 3.3, we used a manager Linux-kernel module that controls the service process of the egress link queue or a certain class of traffic. In this way, the computational effort due to implementing the $\mathcal{AC}$ falls on kernel itself. This has two disadvantages: (*i*) With high reaction rates (e.g., already working with an Enzymatic rate controller with 200 MBps as a limit and concentration update every received KByte, in an Intel(R) Core(TM)2 CPU 6600@2.40GHz, 1024MByte 667MHz DDR2 RAM), the computer suffers from the computational cost derived from running the $\mathcal{AC}$. That is, CPU-resources move away from application processing, and the traffic controller itself risks to represent a performance bottleneck. Instead, performing all mechanisms and computations for running the $\mathcal{AC}$ on an FPGA brings about remarkable advantages: The implementation of an $\mathcal{AC}$ on FPGA completely frees the kernel from Chemistry-related computations. The $\mathcal{AC}$-based controller is able to process events much faster thanks to the computational parallelism and to the high clock rate at which $\mathcal{AC}$-related operations are performed; in the discussed experiment, the controller worked properly with rates $>$ 200 MBps. (*ii*) The other disadvantage is that the computational effort varies according

Higher processing rates

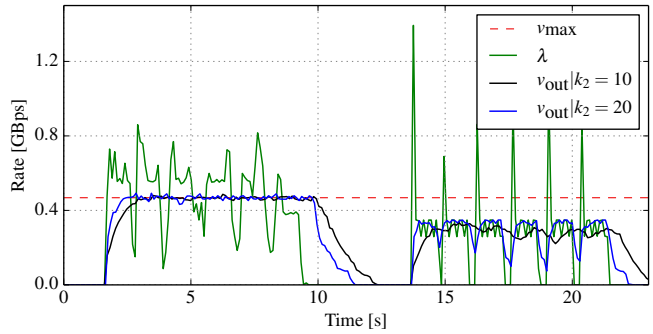


Figure 5.13: *Results of the experiment on the programmability of the hardware Enzymatic-controller formalized in Figure 5.11: v_{out} is the actual Enzymatic-controlled traffic for two cases: $k_2 = 10$ and $k_2 = 20$ s^{-1}. v_{max} represents the limit we predefined by means of parameter $e_0 = 500/k_2$ Kmol, (1 KByte is mapped to 1 molecule and vice-versa), λ is the load presented at the network layer.*

to the complexity of the reaction network, i.e., the number of species and of reactions, and their order. In our experiment, the performance still depended on the complexity of the reaction network. As already mentioned, this was due to the way we had to implement the platform on the small Xilinx Spartan-6 XC6SLX9 FPGA. To fully decouple the reaction model complexity from the computational cost, we need the parallel computation of reaction times (which represents the most time consuming process). This is possible with slightly more advanced FPGA technologies (see Section 5.5.1).

5.4.2 Distributed computation with multiple FPGAs

We configured multiple hardware devices according to the chemical consensus algorithm described in Chapter 4, and let the devices interact such that they constituted a unique reaction network ($\mathcal{DAC}$), and computed distributively, and in parallel, a specific task (i.e., distributed computation of the average-function).

We experimented with a network of $|\mathcal{V}| = 3$ nodes, connected according to a complete topology, where each node ν_i communicates with both its neighbors $\nu_j \in \mathcal{N}_i$ (see Figure 4.4(a) for a graphical illustration).[5.17] Each node ν_i was an Avnet Spartan-6 LX9 MicroBoard, which was configured according to the chemical consensus model formalized in Figure 5.14. For the sake of clarity, in Figure 5.14(a) in the definition of reactions $r_{B',j}$ and $r_{X,j}$, we have neglected the term concerning product-molecules in neighbors ν_ι of node ν_j that are not node ν_i ($\nu_\iota \in \mathcal{N}_j | \nu_\iota \neq \nu_i$ with $\nu_j \in \mathcal{N}_i$). We recall that input reactions $r_{B',j}$

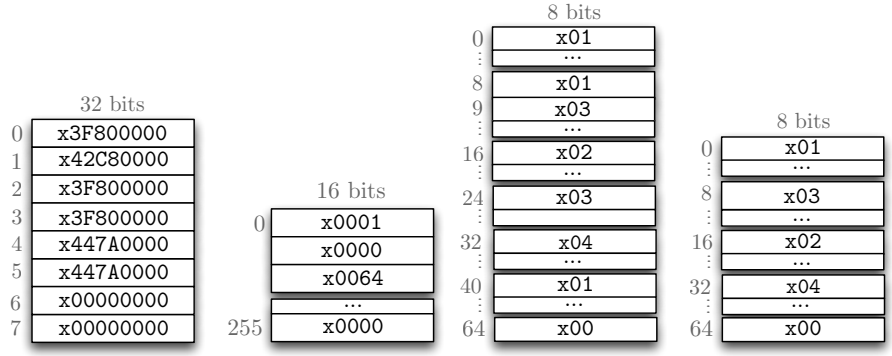
[5.17] In this section, we use the symbolism introduced in Chapter 4.

$$\mathcal{AC} = \Big\{\{\mathrm{S},\mathrm{X},\mathrm{Y},\mathrm{Z}\},\ \{r'_B, r''_D, r_X, r_Y, r_Z, r_A\},\ \mathcal{A}\Big\}$$

$$\begin{array}{llll}
r_{B',i}: & \mathrm{S}_i \xrightarrow{1} \mathrm{S}_i + \sum_{j\in\mathcal{N}_i} \mathrm{S}_j & r_{D'',i}: & \mathrm{S}_i + \mathrm{Y}_i \xrightarrow{0.01} \mathrm{Y}_i \\
r_{X,i}: & \mathrm{X}_i \xrightarrow{1} \mathrm{X}_i + \sum_{j\in\mathcal{N}_i} \mathrm{Y}_j & r_{Y,i}: & \mathrm{Y}_i \xrightarrow{1} \emptyset \\
r_{Z,i}: & \mathrm{Z}_i \xrightarrow{0.001} \mathrm{S}_i + \mathrm{Z}_i & r_{A,i}: & \mathrm{S}_i \xrightarrow{0.001} \emptyset \\
r_{B',j}: & \mathrm{S}_j \xrightarrow{1} \mathrm{S}_j + \mathrm{S}_i & r_{X,j}: & \mathrm{X}_j \xrightarrow{1} \mathrm{X}_j + \mathrm{X}_i
\end{array}$$

$$c^0_{\mathrm{X}} = 100 \quad c^0_{\mathrm{S}} = c^0_{\mathrm{Y}} = c^0_{\mathrm{Z}} = 0$$

(a) Chemical Model



(b) `k-ROM` (c) `c-RAM` (d) α`-ROM` (e) β`-ROM`

Figure 5.14: *FPGA-configuration (configuration of memories) for distributed average computation. The $\mathcal{AC}$ whose model is formalized in Figure 5.14(a) was configured on an FPGA through reaction coefficients stored in `k-ROM` (Figure 5.14(b)), and stoichiometric reactant and product coefficients stored respectively in α`-ROM` and β`-ROM` (Figure 5.14(d) and Figure 5.7(e)); it was initialized through the `c-RAM` (Figure 5.14(c)).*

and $r_{X,j}$ are executed at the neighbor node ν_j and produce molecules S_i and X_i in the node ν_i (as well as in other neighbors ν_ι).

A simple wired infrastructure

Communications occurred via wires, and nodes exchanged virtual molecules via their I/O ports (see Figure 5.15 for the experiment setup). Specifically each time reaction $r_{B',i}$ fired at node ν_i, an output pin of the LX9-board was kept at high level for a clock cycle (25 ns @ 40 MHz), and that level was carried via wire to the input pin of its neighboring LX9-boards $\nu_j \in \mathcal{N}_i$. The 25ns-long high level on the configured input port induced the immediate creation of an S_j-molecule in the neighbor. Dually, the firing of reaction $r_{B',j}$ in one of the

neighbors $\nu_j \in \mathcal{N}_i$ induced the creation of an S_i-molecule in the node ν_i. The same mechanism was used for the r_X-reactions and the Y-species, by using of different I/O pins.
In order to configure the FPGA with the $\mathcal{AC}$ defined in Figure 5.14(a), we configure the platform according to the settings in Figure 5.14(b-e). Each node ν_i was forced to measure a local quantity z_i at a specific time, i.e., each time the local quantity z_i changed by a $\delta_{z,i}$-amount, $\delta_{z,i}$ molecules of species S were injected or removed, accordingly. The experiment's target was seeing nodes that distributively calculate the average of z-quantities:

$$z_{\text{avg}}(t) = \frac{1}{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{V}|} z_i(t). \tag{5.1}$$

Figure 5.16 shows the results for the described experiment setup: continuous colored lines are the (run-time) states of nodes whereas the red-dashed line is the arithmetic mean (ideal value z_{avg}, *a-posterieri* calculated according to (5.1)). All nodes converged to the arithmetic mean of the local quantities.
The convergence time is independent of the technology used to implement the $\mathcal{AC}$. Rather, the convergence time depends on how the virtual time, which characterizes the chemical dynamical system, is mapped to the physical time at which events actually occur. However, the implementation in hardware enables a much wider range for such a mapping – by accepting a higher number of transmissions, we can reduce the convergence time by orders of magnitude. Of course, keeping the design parameters fixed, the convergence time is determined by the algebraic connectivity of the network graph (see Chapter 4).

Wider computational speed range

By comparing the result from the wireless testbed shown in Section 4.7 with those here reported, we observe a great improvement in the accuracy of the average-values estimated by nodes. The difference between the outcome of these two real-world experiments is due to how communications among network nodes take place. The wired setup, here explained, enables more stable, error-free communications than that experienced in the wireless testbed.
With the performed experiment we have shown that, even when having only small-sized FPGAs available, it is possible to implement arbitrary reaction models by interconnecting multiple FPGAs.

5.5 Further insights on hardware artificial chemistries

In this final section, we first discuss the features of the realized platform in terms of logic utilization and computational speed, and we indicate how to improve the speed by using bigger (on-the-market) devices. Then, we comment on possible improvements and how future works could address these aspects. Finally, we discuss the impact that the Chemistry-inspired approach may have as a high-level abstraction to synthesize hardware controllers.

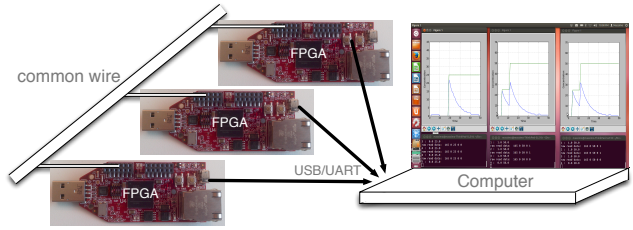


Figure 5.15: *Setup of the experiment on distributed computation on hardware: A network of $|\mathcal{V}| = 3$ nodes, connected according to a complete topology, where each node was an Avnet Spartan-6 LX9 MicroBoard, which was configured according to the $\mathcal{AC}$ formalized in Figure 5.14 and communicated with both its neighbors. The computer had the sole role of monitoring the run-time state of designed species.*

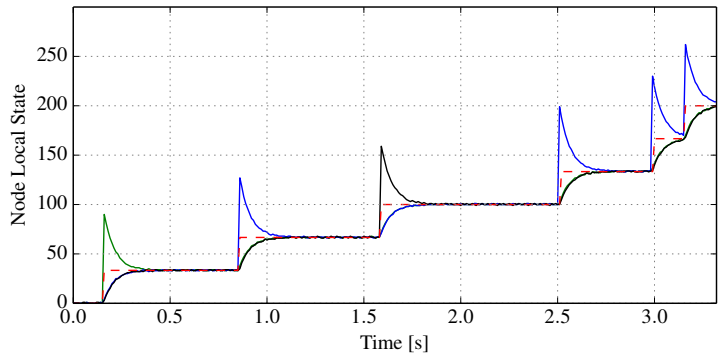


Figure 5.16: *Results of the experiment on distributed computation on hardware: Each node (FPGA-board) of a complete 3-node network was configured according to the $\mathcal{AC}$ formalized in Figure 5.14 – the chemical consensus model. Continuous lines represent the nodes' state whereas the red-dashed line represents the (ideal) arithmetic mean.*

5.5.1 Which logic device fits our need?

The $\mathcal{AC}$-platform has been implemented on Xilinx Spartan-6 XC6SLX9 FPGA. The used FPGA is the second smallest device of Spartan-6 family. With the current implementation, we get around 70% utilization of slice LUTs (most critical feature we had to take into account for syntheses, mapping and routing). As suggested by Xilinx, mapping and routing process succeed only when the utilized logic is less than $\sim 75\%$ of what available. Even with this small FPGA, we were able to setup a chemical model made of up to $|\mathcal{S}| = 255$ species and $|\mathcal{R}| = 8$ reactions, each of them characterized by $|\Xi| = 8$ reactants and products.

We have used the 40 MHz-clock produced by CDCE913 chip on LX9 Mi-

croBoard, and made two working frequencies available: 40 MHz and 80 MHz, although higher frequencies could be used (directly 160 MHz and, by optimizing circuitry choices in this respect, up to 320 MHz). With the current implementation, which overlooks the computational efficiency to benefit the logic utilization, these two working frequencies allow us to correctly process external events (e.g. packet reception) as fast as ~ 10 μs and ~ 5 μs, respectively. At these speeds, the system is still able to process correctly two *sporadic* events that differ in time of less than 50 ns, with the constraint that events must last at least ~ 5 ns.

Computational performances can drastically be improved by parallelizing the computation: The ability to process events as fast as tens of nanoseconds can be obtained by associating a `time-Calculator`-module to each reaction and further parallelizing the multiplication process among reactants' concentrations. This would have the cost of a higher logic utilization: An $\mathcal{AC}$-platform, which has similar capabilities to that we implemented on XC6SLX9 FPGA in terms of maximum number of species (255), reactions (8), reactants (8), and products (8), would require around four times the logic utilization we currently require. Although not fitting in a XC6SLX9 FPGA, such a high-performance $\mathcal{AC}$-platform can be implemented in on-the-market devices such as the XC6SLX45 FPGA (still Spartan-6 family). Further, it is possible with today's technology to have a high-performance $\mathcal{AC}$-platform with 32 reactions each with 32 reactants and products (a complete $\mathcal{AC}$ that would definitively satisfy requirements and needs for any known application in the field of Chemistry-inspired algorithms), for example by using Spartan-6 XC6SLX150 FPGA (also a device which is already available on the market since a few years).

Parallelization...

...orders of magnitude of improvement

...implementable on available chips!

5.5.2 Possible improvements and future works

The configurable platform we have introduced offers already an advanced tool to easily build control mechanisms in hardware. However, as we have argued, the realized system cannot fully exploit the computational parallelism. A straightforward step to improve the platform is parallelizing the calculation of reaction times. This requires the mere application of guidelines we have given in Section 5.2, simplifying the schemes illustrated in Section 5.3, and finally implementing the whole system in a bigger FPGA-device (e.g., XC6SLX45, XC6SLX150).

We believe we can enhance the platform (e.g., reduce the logic utilization) by avoiding working with the floating-point representation. The implementation may rely on fixed-point representation and integer arithmetic only. To this end, the time-related computations must carefully be mapped into a limited dynamic range, and scaled through each function in the path, by paying attention to the rounding and saturation steps.

Towards fixed-point representation

To the best of our knowledge, this is the first framework for complete $\mathcal{AC}$s on hardware. However, our platform may be improved to support random dynamics – so far, the scheduling of the reactions can be deterministic only.

Random dynamics

Future versions may include the possibility to enforce random trajectories by means of reaction intervals that are, for example, exponentially distributed, as required to simulate actual chemical systems, or normally distributed, as required to control the level of randomness characterizing the system's dynamics. Including stochasticity may be simple as there exists a research branch that studies methods and proposes solutions to generate random variables on FPGAs (e.g., [142, 143]). But of course, this would have the cost of a higher logic utilization (e.g., $\sim$ 5 to 10 % of the utilization of a device such as the Xilinx XC6SLX9 FPGA).

5.5.3 Chemistry-inspired high-level abstraction for the design of hardware controllers

General-purpose programming languages...

...not suitable for hardware design

Although there are vendors that advertise C-to-VHDL compilers, one can actually not compile to hardware any equation-based control-mechanism written in a general-purpose programming language. This would raise the level of abstraction and, as a result, the design would get bulky and slow. The reason behind this is the considerable engineering effort that is necessary to convert the software into something that will run at a reasonable speed in FPGA-technology. In fact, VHDL-based logic synthesis is an efficient method for designing complex hardware that does not suit well the purpose of describing regular structures like finite-state machines [207]. The design of hardware modules should instead match closer the features of hardware technologies.

The $\mathcal{AC}$-platform can be seen as an abstraction/middleware for generating VHDL-synthesizable code, thus for synthesizing a gate-level description for FPGA-technology. Designers are able to build hardware controllers that are based on more or less complex equations, by keeping under control the possible intricate web of dependencies that arises. Such a high-level design abstraction (*i*) reduces the design time, (*ii*) includes a graphical as well as a rule-based description, (*iii*) facilitates describing any sort of proportional, derivative, linear and non-linear functionality, and (*iv*) enables importing directly principles, laws, and models from the surrounding nature. The analyzability and the predictability of implemented modules improve too: a quick validation of the controller can be performed empirically by simulating the chemical model on simulators, and formally by means of the analytical tools borrowed from Chemistry.

There exist tools that exploit a high-level abstraction to describe a system by means of a model-based description rather than a complex, low-level VHDL or Verilog description (see for example the system generator born from the collaboration of Xilinx Inc. and Mathworks Inc.). Model-based approaches for hardware design require a translation step to convert complex functionalities into a HDL-representation. The obvious challenges and issues concern length, quality, and readability of the generated code. Furthermore, these tools provide solutions to specific tasks and cannot (easily) be adapted to realize arbitrary control systems [32], much less once the controller has already deployed. With

our approach, any distributed or centralized controller is realized by means of simple rules (reactions) to update registers (species) according to analyzable dynamics (dictated by the mass-action principle). In this way, the user is able to implement hardware controllers by configuring a few memories (which can be done at runtime), without the need for an actual translation of algebraic functions into logic gates and their interconnections (which are instead programmed semi-permanently as part of the assembly process).

5.6 Conclusion

In this chapter, we have pushed the implementation of Artificial Chemistry down to hardware – an in-silicon chemical-like system, made of virtual molecules that react with each other following chemical-like trajectories. Besides the mere scientific curiosity, this is interesting and beneficial in the context of Chemistry-inspired networking and communication systems.

We propose a full-hardware implementation where the complexity derived from control mechanisms, interactions, and feedback loops is approached with, and kept under control thanks to, the chemical metaphor. With the proposed approach, control systems are built in hardware by drawing interaction graphs (chemical reaction networks). Designers can let hardware perform complex equations rather than basic, fixed rules only. They define equation-based mechanisms by simply modifying the configuration of a few memories.

Our approach challenges the conventional belief that identifies the software-based approach as the only way of handling the complexity derived from abundant, dependent, conditional statements in control-applications. We do not define control algorithms by means of general programming languages although, apparently, doing this would be much faster and require less effort than directly resorting to hardware descriptions. Our main motivation is that software descriptions require anyway a transposition into hardware. This generally leads to complex sub-optimal implementations that make the system bulky and slow. We adopt instead a model-based approach that describes directly a system by simple, analyzable rules and registers. The adopted Chemistry-inspired description is close to the software-representation, thus it makes the algorithm description an easy task. At the same time it is close to the hardware-representation, so that the described algorithm can be built directly in silicon.

This chapter should reassure engineers that are concerned about the computational cost derived from running artificial chemistries. One can safely take advantage of Chemistry-inspired algorithms for high-speed networking tasks, such as traffic shaping in next-generation networks. Costs of Chemistry-related computations are kept negligible thanks to the possibility to build complete artificial chemistries in hardware – chemical controllers are bounded to independent chips that can afford high-speed processing, and do not require any computational resource from the main processors.

In this chapter, we should have appeased also the concerns about the logic

resources required to implement more or less complex artificial chemistries on FPGAs. Indeed, we have demonstrated that simple reaction networks, sufficient for most networking tasks, still fit in small FPGAs that are available today on the market. Furthermore, we have shown that more complex reaction networks can still be run on multiple FPGAs – there is no practical difference between chemical reaction networks that are confined into one device and those that span across multiple devices. That is, we can configure a distributed artificial chemistry that takes advantage of multiple devices at the same time, to gain in (logic-)capacity and/or speed.

Chapter 6

Conclusion

"In literature and in life we ultimately pursue, not conclusions, but beginnings."

Sam Tanenhaus

With this chapter, we conclude our treatment of Artificial Physical Chemistry applied to the design and the analysis of networking and communication systems. Nevertheless, we would like this chapter, and in general this thesis, to represent a starting point for a research line that devotes time and effort to applying the Chemistry-inspired approach, in all its flavors, to different areas of communication engineering.
This closing chapter is structured as follows: We first recall generic features of Chemistry-inspired algorithms/systems in Section 6.1. We summarize our main contributions in Section 6.2. Finally, we comment on possible directions of future works in Section 6.3.

6.1 Chemistry-inspired engineering of communication and networking systems

The most important feature of Chemistry-inspired systems is the strict relationship between their functional and dynamical aspect. In general terms, systems can be described/defined by their state space and by the rule that determines the transitions in that state space. In the classical approach, this rule is given explicitly by the computer program in order to accomplish specific functional tasks, without paying attention to emergent dynamics exhibited by the system. On the contrary, in the Chemistry-inspired approach the rule *emerges* from the combination of distributed interactions governed by an underlying law (i.e., the law of mass action). That is, with the Chemistry-inspired approach, we program the dynamics in order to achieve, at equilibrium, predefined functional tasks.

With the Chemistry-inspired approach, designers adopt a flow-based perspective on communication processes, and they design algorithms by defining algorithms' macro behavior (e.g., how a rate control algorithm should react in face of an imminent congestion of the shared resource). This is done by describing the system's dynamics in terms of (distributed) chemical reaction networks. At the same time, designers do not have to handle directly micro-states' changes (e.g., the effect of receiving/enqueueing a single packet) and event timing (e.g., when to transmit the next packet so as to comply with the maximum aggregate rate limit). Once the reaction network is designed, the reaction algorithm schedules correctly (according to the underlying law – law of mass action) the program that spans the designed reaction network. This differs from the full control of each micro-state's change that is required by traditional protocols, where the control of the dynamic behavior is explicitly enforced by handling event timers and race conditions in the protocols' state machine.

Emergence

With the Chemistry-inspired approach, designers can implement *emergent-control* strategies, in which the controlled behavior emerges at a global scale (macro-level) from a set of (usually simple) local rules that are executed by the different components of the whole system/network; the control function is not enforced by a single module but rather emerges as asymptotically stable equilibrium state of the whole system. Emergent-control strategies can fulfill the seemingly paradoxical need of today's and future communication environments for combining self-organization (e.g., automation of network management and resource allocation in the Internet) and dynamics control (e.g., traffic control to provide reliable and predictable services for exchanging/storing users' data). Like in the distributed rate control and in the consensus achievement, a global coordination arises out of the local interactions between the system's components. This process is spontaneous and not governed/controlled by any internal or external module but is rather completely distributed. Some key features of traditional (feedback) control, such as global state and feedback loop, do not appear explicitly at the macro level. Instead, thanks to the underlying mass-action kinetics, local micro-level feedbacks and micro-rules give rise to a controlled macro-behavior of the system.

Robustness

Besides emergence, the Chemistry-inspired approach enables/facilitates system *robustness*, i.e. the ability of the system to maintain its functionalities in the presence of perturbations, stress, or errors. Like actual biological systems, Chemistry-inspired algorithms can well tolerate violations of their nominal inputs and variations of their operating environment (and even when they are not, they "fail gracefully"). More formally, from the point of view of dynamics, robustness is the ability of the system to find the attractor (a region of the state space which attracts trajectories of the system, i.e. a steady state) when perturbed from the current trajectory. This happens if the perturbation maintains the basin of attraction (the set of all states whose trajectories lead to that attractor/steady state). Therefore, robustness calls for large basins of attraction. Traditional computational systems instead exhibit usually small basins of attraction, such that any perturbation is likely to move the system to

a different basin, hence to a different attractor, resulting in a different (wrong or, at least, not predicted) computation/result. For example, in gossip and consensus algorithms, the reception of a single corrupted packet can compromise the correct computation of the target function. On the contrary, (macroscopic) solutions of Chemistry-inspired algorithms are steady states with larger basins of attraction. Perturbations move the system only a small distance in its state thanks to stoichiometric rules that force the system to move by "small steps" only. Hence, perturbations likely maintain the basin of attraction and thus, they do not lead to a completely different behavior/result than that expected in nominal conditions. This holds true also for small perturbations of the system's parameters, given that the parameter's variation does not lead to crossing bifurcation points (i.e., does not induce a change in the system's behavior). For example, the effect of one or a few missed/corrupted receptions by sensors that implement the Chemistry-inspired consensus algorithm is not even appreciable in the final result of the distributed computation.

Rate-encoded information

The mass-action principle allows Chemistry-inspired systems and algorithms to convey information by rate modulation. In traditional computer networking, a piece of information is encoded and processed symbolically: information is encoded as a sequence of binary numbers and stored in packets, which are then transmitted to convey the information to the recipient. In mass-action driven systems, the information, which reflects also (part of) the state of the system, is conveyed by the rates of distributed reactions and thus by the rates of transmissions among spatially distributed nodes of the communication network. This information can represent data-messages, as in the example of the chemical consensus algorithm (that indeed can be implemented directly as a communication protocol), or control-messages that are exchanged for free orthogonally to the symbolic payload, as in the case of the distributed traffic rate controller.

Simplified analysis process

Thanks to the mass-action principle, the analysis of Chemistry-inspired systems is simple and accurate. On one hand, the law of mass action enforces a direct mapping between the state and the state's variation of the system. This enables extracting automatically the fluid model description (i.e., a set of ordinary differential equations) of the system from its execution model (i.e., the reaction network that defines it). One the other hand, the law of mass action makes the system's dynamics more fluid. For example, thanks to the mass-action-based scheduling of packets in packet-switched networks, packets transmissions are paced such that their interactions at queues are explicit and predictable. In this way, the fluid model approximates accurately the system's dynamics. The analysis process of networking and communication systems is also simplified thanks to the direct use of chemical theory, which provides new analytical tools to complement traditional methods of control and queueing theory. For example, we have demonstrated the stability of the chemical consensus algorithm by means of the Deficiency Zero theorem, which is a novel, unusual tool in the field of networking and communication engineering.

6.2 Contributions and findings

In the thesis, we have focussed on system dynamics and have investigated *computation and processing* from a "Physical Chemistry perspective". We have revisited some of the definitions and principles of Chemistry and have adapted the way of reproducing artificial chemistries on traditional computers so as to have at hand powerful tools to engineer communication and networking systems. We have shown some potentialities and benefits of Chemistry-inspired engineering in different domains, which range from rate control to distributed computation, from software-based mechanisms to hardware controllers. We have illustrated how it simplifies essential development-stages of communication and networking systems, such as the *analysis* of system's stability, system's sensitivity, and system's equilibrium points, as well as the *design* of control systems that are distributed and embedded in the environment. For the first time, we have demonstrated that the chemical approach is not relegated to pure theoretical modeling exercises or to fuzzy hypothetical systems far away from the actual deployment. On the contrary, it can be applied, today, to realize (*i*) algorithms for controlling the access to shared resources, (*ii*) linux-kernel and hardware modules for controlling the Internet traffic, and (*iii*) mechanisms for distributive computing in wireless sensor networks. Furthermore, the underlying chemical theory can be used to account for practical issues related to the actual deployment of algorithms, by taking advantage of the intrinsic robustness and self-organization of Chemistry-inspired mechanisms.

Distributed traffic rate control in packet-switched networks

We have shown how the Chemistry-inspired approach can be beneficial in managing traffic in packet-switched networks. Mass-action-served queues guarantee predictable and controllable interactions among packet flows. This stems from the strict relationship between the system's state and dynamics, between queue-fill levels and scheduling rates. We have formalized the stationary features of mass-action-based queueing systems, based on steady-state analysis. The chemical approach also provides us with an intuitive pattern that embodies one of the most used techniques in traffic shaping: rate limit/control. In this regard, we have investigated (analytically, by simulations, and by experiments with real traffic) the use of a well-known chemical model (the Enzymatic reaction model) as a local traffic controller to limit the traffic rate up to a maximum value and to control the level of potential traffic bursts. Starting then from this simple but effective mechanism, we developed a class of distributed rate controllers that can be easily parametrized, extended, and customized, for various purposes and in different operational environments, such as to enable service differentiation among user's flows at the client side, to distributively coordinate aggregate flows, to allocate capacities to admitted flows in an intra-domain network, and to regulate the access to a resource at the server end. For such a class of distributed rate controllers, we have provided an in-depth analysis and theory-based guidelines for tuning a possible implementation, and a first evaluation in a small network of wire- and wireless-connected computers.

We have shown that the Chemistry-inspired approach can be used to develop communication protocols in the context of distributed computation in Wireless Sensor Networks (WSNs). Our contributions in this field are mainly four: (*i*) We have revisited known theoretical results in the consensus context by means of new tools directly imported from Chemistry. (*ii*) We have shown that the same tools can be used to derive a set of interaction rules that allow achieving consensus in WSNs, in a distributed manner. At the same time, we have used these tools to formally assess the qualities (convergence and stability) of the derived system. (*iii*) We have validated the performance of the proposed model by means of simulations and a four-node hardware testbed. (*iv*) In doing this, we have demonstrated that Chemistry-inspired protocols can lie at the lowest layer of the communication stack; chemical reactions control directly Radio-Frequency (RF) modules. The emergent and simple communication protocol lets nodes exchange their data in an asynchronous admission-free manner, with no need for symbolically encoded information and for packet exchanges.

Communication protocol for distributed computation in WSNs

We have demonstrated that Chemistry-inspired algorithms can be built in hardware. This represents a remarkable step towards the adoption of this class of algorithms in real communication and networking environments, where high-speed computation and processing capabilities are required more and more frequently. We have given guidelines on how to decouple the computational time related to running the underlying chemical model from the complexity of the chemical model itself. Covering the opposite side of the design spectrum, we have shown how to "fit" artificial chemistries to relative small hardware devices, advantaging logic density over computational speed.

High-speed high-resolution Chemistry-inspired systems on hardware

The investigation on the hardware implementation of Chemistry-inspired systems has brought to another important contribution: a user-friendly high-level abstraction for the design of hardware controllers. The developed platform can be used as a facility to generate a logic-gate-level description, synthesizable on FPGA-technology (Field-Programmable Gate Array). With this platform, more or less complex equation-based control-mechanisms can be built in hardware. Designers are able to realize distributed or centralized control modules by simply defining chemical species and reaction rules. These modules exploit the processing parallelism of hardware, and thus achieve high-speed processing and high time-granularity capabilities. At the same time, the analyzability and the programmability of dependent-conditions and actions are preserved. This contrasts with most of the existing control solutions that have to rely on software-based sequentialized descriptions because of the difficulty to program dependent-conditions and actions through a hardware-based parallelized description.

User-friendly high-level abstraction for hardware design

6.3 Future works

We would like this doctoral work to be an incentive to tackle open issues of current and future communications by means of the Chemistry-inspired approach – an approach that, from the earliest stages of the development process of algorithms, protocols, or mechanisms, (*i*) promotes dynamics awareness, besides traditional concerns on functional aspects, (*ii*) supports the development of systems with a-priori defined and well predictable dynamics, (*iii*) provides analytical and formal tools, and at the same time (*iv*) is simple and intuitive thanks to a direct use (sometimes combined) of elementary motifs and well understood patterns. We are convinced that, in this thesis, we have identified only some of the possible research areas (e.g., distributed computation in wireless sensor networks, traffic shaping in packet-switched networks) where this novel, unconventional approach can successfully and beneficially be applied. However, we hope that other researchers and scientists recognize in the here-proposed framework a valuable means for progressing in their specific fields of interest and knowledge. In addition, we just have embarked on pursuable paths to an actual usage of Chemistry-inspired solutions, and further investigations are required to actually influence and define future industrial R&D topics.
An interesting project would be for example developing a Chemistry-inspired hardware design environment, which supports the graphical as well as the rule-based design, offers a fast validation prior deployment, and provides a theory-based facility for tuning the module under construction. Such a project is no more than the natural evolution of what we have proposed in this thesis in the context of hardware implementation of Artificial Chemistry. In fact, we believe that it is possible to develop a product, ready for today's market, for the realization of runtime-programmable hardware controllers. Users would be assisted in the design process by (*i*) a graphical environment, where it is possible to design/define the behavior of the future hardware module by drawing chemical reaction networks, and (*ii*) a rule-based environment, where it is possible to design/define the hardware module's behavior by writing a set of chemical reactions. The platform should include analytical tools of Chemistry and control theory in order to offer to the user information about the dynamics of the module under construction (based on the automatically extracted fluid model). This analysis-based tool should assist the user in finding a setting/tuning (or even suggest the user a possible optimized setting/tuning) in order to accommodate specific demands on the module's behavior. Finally, the platform should offer the possibility to verify through simulations the (dynamical) behavior of the module under construction. Note that with "simulating the behavior", we mean the reproduction of the chemical model (system of species interconnected according to reaction rules) by means of a reaction algorithm. The simulation of the actual hardware implementation of the chemical model is not needed instead. This is because the building process is standardized and does not involve each time the translation from chemical description language to hardware description language. That is, the behavior

of the hardware module is guaranteed to faithfully reflect the dynamics of the underlying chemical model.

Transparent engineering framework

A decisive step towards large-scale use of Chemistry-inspired engineering could be the development of an environment in which the designer can define specific features of the dynamic behavior that the system should exhibit, and get, as output, a chemical model that meets these specifics. For example, the designer might define a specific mask for the frequency-domain features of the future system and get a specific chemical reaction network as output. The position of each singularity of the defined frequency response would contribute to the definition of the output (linear) chemical model. Such a facility would reduce the effort required, at the first stage of the Chemistry-inspired engineering process, to design a chemical model that meets the project specifics. It would even make the underlying chemical metaphor transparent to designers – kinetics and interaction laws could be used to realize systems exhibiting the specified dynamical behavior, by designers with no related expertise.

In-network management mechanisms (packet-switched networks)

Another wide noteworthy field of research is the in-network control of traffic in packet-switched networks by means of chemical kinetics and principles. The mere adoption of the mass-action principle for scheduling dequeueing/transmission of packets in routers leads to an exact and strict relationship between (*i*) the queue fill level, (*ii*) the queue latency, and (*iii*) the dequeueing/transmission rate of packets. Interestingly, these three elements, on which any queueing-theory-based description of systems relies, represent also the key points of past and recent research on queue management. We believe that the import of laws and concepts from Physical Chemistry into this field may be successful and give rise to remarkable solutions from a design/implementation perspective and, mostly, in terms of analysis/predictability. We also believe that the exploration of Chemistry-inspired in-network traffic control can bring important contributions to the creation of programmable, manageable, and adaptable networking infrastructure, a hot topic addressed by the Software-Defined Networking (SDN) community. The Chemistry-inspired approach would complete SDN by broadening the programmability of routers and switches to enable *programmable network dynamics*. And, although there may soon be software-based solutions for the "programmable control" of traffic dynamics, the Chemistry-inspired approach still represents a means of high programmability in both software and hardware environments, which is worthwhile to research on.

Appendix A

Supplementary material

This additional chapter contains supplementary material. Although its contents are related to the main part of the thesis, they are not essential for the understanding of the manuscript, would have risked to break the logical writing thread if included in the main body, or need a deep investigation. For this reason we postpone them to this very closing part.
This chapter is structured as follows: We introduce in Section A.1 a new attraction-based law that can simplify patterns for designing networking mechanisms. We provide the step-by-step analysis of the Enzymatic rate controller in Section A.3 and that of the Distributed Traffic Rate Control (DTRC) mechanism in Section A.4. In Section A.5, we study the use of the DTRC-mechanism as a random media access algorithm and compare it with the traditional ALOHA protocol. We discuss in Section A.6 how to enable service-class differentiation within the DTRC-mechanism and report early experiments' results. In the context of Chemistry-inspired distributed computation, we give results about simulations of the chemical consensus model in clustered and inline networks in Section A.7, introduce a possible approach to account for the runtime communication collisions in Section A.8, and an alternative frequency-division implementation of the chemical consensus algorithm in Section A.9.

A.1 A novel attraction-based law: LoPA

We have already extended the potentiality of the chemical paradigm for the design of communication/networking algorithms by enabling deviating from laws and models that describe phenomena in real Chemistry. In Section 2.4, we have introduced the possibility to choose different distributions to generate random inter-reaction times (deterministic times are possible too), with benefits that range from scenario to scenario and from application to application (e.g., deterministic reaction times are useful to obtain network dynamics that are theoretically free from oscillations and that match exactly analytical predictions).
Here, we further extend the chemical paradigm through the introduction of

Attraction-based reaction law...

a new reaction law: Law of Pseudo Attraction (LoPA). LoPA states that *the reaction rate is proportional to the concentration (quantity) c of all product molecules.* LoPA essentially relates the reaction rates with the product concentrations through the following propensity function

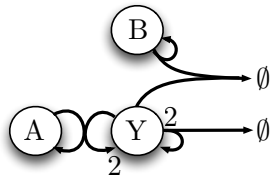
...propensity

$$v_r = k_r \times \prod_{s=1}^{|\mathcal{S}|} c_s^{\beta_{sr}}, \tag{A.1}$$

where k_r is the reaction coefficient, $|\mathcal{S}|$ is the size of the set $\mathcal{S}$ of all the present species, c_s is the concentration of the s-species, and β_{sr} is the stoichiometric reaction coefficient related to s-species and reaction r, i.e. the amount of s-molecules produced by the r-reaction.
LoPA can simplify design patterns for Chemistry-inspired algorithms and definitely broadens capabilities of engineers and designers. This novel law, which does not have meaning in the context of actual Physical Chemistry, forces reaction kinetics to depend on the amount of molecules of product species, and not on the amount of reactant molecules as instead dictated by the discussed LoMA. The cost for such an attraction-based law is the need for information that can be not available locally. When the product species lies in a different node (possibly reflecting also another physical location – distributed reactions), the rate of the reaction depends on a value (propensity) that is not known locally. This is not a problem when the state of product is available at the node where the reaction occurs. It requires instead the exchange of state information when the product species lies in a physical location, which is away from the node where the reaction occurs and when the (chemical) state information of that neighbor is not available locally.
To evaluate possible benefits for the introduced law, in the next section, we compare two motifs to compute the difference between any two quantities, one model is obtained relying on the LoMA (the "difference motif" proposed by Meyer in [163]), the other is obtained relying on the LoPA.

A.1.1 Difference motif: LoPA *vs.* LoMA

In [163], Meyer introduced the difference motif, a chemical model to compute the difference between two quantities. Figure A.1 shows the related reaction network and reaction equations. The two input quantities are encoded into the molecular concentrations c_A and c_B of species A and B; after a transient time, the species Y contains a number of molecules equals to $c_\mathrm{Y} = c_\mathrm{A} - c_\mathrm{B}$.
By applying the attraction law LoPA, we can simplify such a scheme and make its dynamics reducible to a linear model. In Figure A.2, we have three reactions (r_1, r_3, and r_4) that are governed by the LoMA, and one reaction (r_2) that is governed by the LoPA. Like in the previous chemical model, the network of reactions subtracts the number of B-molecules from the number of A-molecules and presents the result in the number of Y-molecules, when condition $c_\mathrm{A} > c_\mathrm{B}$ is satisfied.

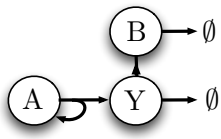


(a) Reaction Network

$$r_1 : \mathrm{A} + \mathrm{Y} \xrightarrow{1} \mathrm{A} + 2\mathrm{Y}$$
$$r_2 : \mathrm{Y} + \mathrm{B} \xrightarrow{1} \mathrm{B}$$
$$r_3 : 2\mathrm{Y} \xrightarrow{1} \mathrm{Y}$$

(b) Reaction Equations

Figure A.1: *The network of LoMA-reactions subtracts the number of B-molecules from the number of A-molecules and presents the result in the number of Y-molecules. If $c_B > c_A$, the presented quantity is $c_Y = 0$.*



(a) Reaction Network

$$r_1 : \mathrm{A} \xrightarrow{1} \mathrm{A} + \mathrm{Y}$$
$$r_2 : \mathrm{Y} \xrightarrow{1} \mathrm{B}$$
$$r_3 : \mathrm{B} \xrightarrow{1} \emptyset$$
$$r_4 : \mathrm{Y} \xrightarrow{1} \emptyset$$

(b) Reaction Equations

Figure A.2: *Reactions r_1, r_3, and r_4 are governed by the usual LoMA, whereas reaction r_2 is governed by the new Law of Pseudo Attraction (LoPA). The network of reactions subtracts the number of B-molecules from the number of A-molecules and presents the result in the number of Y-molecules. If $c_B > c_A$, the presented quantity is $c_Y = 0$.*

To look how this computational result can be achieved, let look at the fluid model of both reaction systems and study their behaviors at steady states. For the traditional LoMA-model in Figure A.1, the following set of ODEs is valid

$$\dot{c_\mathrm{A}} = 0 \tag{A.2a}$$
$$\dot{c_\mathrm{B}} = 0 \tag{A.2b}$$
$$\dot{c_\mathrm{Y}} = -c_\mathrm{A}c_\mathrm{Y} + 2c_\mathrm{A}c_\mathrm{Y} - c_\mathrm{B}c_\mathrm{Y} - 2c_\mathrm{Y}^2 + c_\mathrm{Y}^2. \tag{A.2c}$$

From (A.2c), by collecting c_Y and then simplifying with respect to c_Y, we get that $c_\mathrm{Y}^* = c_\mathrm{A}^* - c_\mathrm{B}^*$. The set of ODEs for the attraction/repulsion model is instead

$$\dot{c_\mathrm{A}} = 0 \tag{A.3a}$$
$$\dot{c_\mathrm{B}} = c_\mathrm{B} - c_\mathrm{B} \tag{A.3b}$$
$$\dot{c_\mathrm{Y}} = c_\mathrm{A} - c_\mathrm{B} - c_\mathrm{Y}. \tag{A.3c}$$

Again we get that at steady state, $c_\mathrm{Y}^* = c_\mathrm{A}^* - c_\mathrm{B}^*$. By looking at the models as well as the fluid model describing their behaviors, it is visible that the

complexity of the difference motif when applying an attraction law next to the repulsive LoMAis reduced.

We envision the possibility to extend the chemical paradigm for designing networking and communication algorithms with any macroscopic law (e.g. LoMA, LoPA) and any microscopic distribution (e.g. exponential, normal). However, we believe a fixed feature that any sort of extension must preserve is the direct coupling between states and rates (that is enforced by reaction laws), which must permit to automatically build a deterministic ODE model that describes the algorithm's dynamics.

A.2 Further related works on traffic shaping

In this section, we extend the discussion started in Section 3.1 on the various building blocks that (have) contribute(d) significantly to shaping the traffic in packet-switched networks. We summarize the mechanisms that are used today in the Internet to keep network dynamics under control and obtain an efficient and fair use of network resources. In the following, we briefly comment on end-to-end (Section A.2.1) and in-network mechanisms (Section A.2.2).

A.2.1 End-to-end transport control

Transport-layer protocols

Today, in the Internet, there are many mechanisms that contribute to managing the traffic and keep network dynamics under control. A prominent role is played by transport-layer protocols that provide end-to-end communication services for applications, such as capacity sharing and flow multiplexing. Among them, the Transmission Control Protocol (TCP) [199] is the mostly used. TCP abstracts away the application's communication from the underlying network's conditions (it detects lost, duplicated, or out-of-order packets, it requests retransmission of lost data, rearranges out-of-order data, and tries to minimize network congestion while maximizing the goodput) and provides a reliable, bidirectional virtual channel between any two hosts over the Internet. There exist a few other alternative transport-layer protocols, albeit much less in use today. The Stream Control Transmission Protocol (SCTP) (see [186] for a sound introduction to its features) is a session-oriented[A.1] protocol that was born to accomplish the requirements of telephony signaling transport. Like TCP, it is rate adaptive and provides an in-order, error-free, and reliable transport service.

[A.1] In a session-oriented mechanism, the relationship between the communication endpoints is created prior to data being transmitted, and this relationship is maintained until all data transmission has been successfully completed.

Moreover, it supports functions for multi-streaming[A.2] and multi-homing[A.3]. A less sophisticated protocol is the User Datagram Protocol (UDP) [198], which provides a simple messaging service that is fast but can offer only an unreliable best-effort delivery, without dynamics-aware mechanisms. More recently, the Datagram Congestion Control Protocol (DCCP) [133] has been introduced to merge, in a single protocol, congestion-control (like TCP) and timeliness (like UDP) by renouncing requirements of reliability and in-order-delivery, so as to meet the needs of applications with timing constraints on data-delivery (e.g., streaming media, multiplayer online games and Internet telephony).

Transmission Control Protocol (TCP)

Most traffic of today's Internet however is carried by TCP. One of the main elements of TCP is its congestion-control algorithm, which governs the dynamics of the communication between endpoints. TCP tries to maximize the capacity utilization while avoiding to congest the end-to-end path. To this end, it continuously probes the network for spare capacity and controls the rate of packet flows at the source, with a window-based mechanism. Packets are sent at a rate that is periodically regulated relying on a dynamically-adapted window, such that roughly one window's worth of packets is transmitted every Round-Trip Time (RTT) [149].[A.4] Over the last thirty years, the congestion-control mechanism has been at the core, or in any case an important aspect, of TCP variants, and in fact is what differentiates them. TCP variants differ in the type of feedback signal used to detect congestion (e.g., delays, packet losses), how this feedback is interpreted in various operational environments (e.g., wireless networks, high-speed long-delay networks, data-centers, mobile ad hoc networks), and how they are optimized for specific applications (e.g., low-priority data-transfer service for background traffic, web-traffic).

Algorithms of transport-layer congestion-control

Modern congestion control emerged from the need for overcoming the "congestion collapse" phenomenon.[A.5] By having a network-aware rate limiting mechanism, a TCP sender estimates the number of data packets the network can accept for delivery without becoming congested. TCP Tahoe introduced the first set of algorithms to address the challenges of congestion control [37,121,199]: "Slow Start" and "Congestion Avoidance" constitute two distinct main operational phases of TCP Tahoe to detect available network

[A.2] Multi-streaming function allows data to be partitioned into multiple streams that can be independently-sequenced delivered, so that a message loss in any stream will only initially affect delivery within that stream, and not delivery in other streams. Compared to strict sequence-preservation (like in TCP), this reduces the delay when a message loss or a sequence error occurs within the network.

[A.3] Multi-homing function enables a single SCTP endpoint to support multiple IP addresses, thus it increases the potential survival of the session in case of network failures – i.e., enhancing the network's robustness.

[A.4] The Round-Trip Time (RTT) is the time elapsed since a packet is sent until the related acknowledgment (ACK) is received at the source.

[A.5] According to the "congestion collapse" phenomenon [93], if the offered load in an uncontrolled distributed sharing-system exceeds the total system capacity, the successful traffic will go to zero as load increases.

resources and adjust the transmission rate of the TCP flow to the detected limits respectively. In the Slow Start phase, the reception of an acknowledgment (ACK packet) is considered as an invitation to send double the amount of data that has been acknowledged (multiplicative increase policy). In Congestion Avoidance phase, a much more conservative policy is followed in response to received ACK packets and to the detection of packet losses – the congestion window increases by one only if all data packets have been successfully delivered during the last RTT (additive increase policy), and it is halved after a loss (multiplicative decrease policy). Fast Retransmit algorithm [222] was introduced to reduce the loss-detection time by requiring that TCP receivers accept and reply to all out-of-order data packets with reports of the last in-order packet (a duplicate ACK [37]), so as to report the packet-loss to the sender within the RTT interval.[A.6]

Later, TCP variants (mostly) have varied in how the algorithms react to the detection of losses. For example, TCP-Reno [11] introduced the conceptual difference between major (a loss detection through RTO) and minor (a loss detection through duplicate ACKs) congestion events, and the Fast Recovery phase; the sender retransmits the missing packet that has been signaled by three duplicate ACKs, and waits for an ACK of the entire transmit window before returning to congestion avoidance or, on timeout, enters the slow-start state. The shorter recovery period and the possibility to transfer data within the recovery period make the use of the network more efficient. TCP New-Reno [85, 86] further improves the efficiency in heavy-load environments (for multiple loss events) by taking into account the dependency of sequentially detected loss and avoiding accordingly to reduce excessively the rate. Further incremental improvements include extensions of reporting capabilities of the receiver (such as Selective ACK [156], to notify blocks of successfully delivered data packets, and Duplicate-SACK (DSACK) [88], to notify the reception of duplicate packets for the same sequence number) in order to make the recovery from errors much faster (Forward Acknowledgment [155]) or to correctly deal with the phenomenon of packet reordering (Reordering-Robust TCP [266]).[A.7]

New solutions further exploit enhanced reporting capabilities of the network itself; DCTCP [8], a TCP version for data-center environments, take advantage of in-network ECN-marking (see later) to throttle flows by reducing the congestion window in proportion to the extent of congestion. To successfully work in high-speed long-delay networks (i.e. networks with large Bandwidth-Delay Product), TCP has been adapted to rapidly discover the right congestion window size. For example, Binary Increase Congestion (BIC) control algorithm [257] introduces an additional operational phase, "Rapid Convergence", to induce,

[A.6] Otherwise, the packet loss is detected at the end of the Retransmission TimeOut (RTO).

[A.7] In real networks, packets are reordered due to different reasons, ranging from simple erroneous software or hardware behavior (bugs, misconfigurations, or malfunctions) to in-network mechanisms (diverse packets handling services, internal rescheduling at routers) and packet processing parallelism. Packet reordering makes a simple loss detection mechanism not feasible (out-of-order delivery $\neq$ packet loss).

through a binary search manner, a very fast (logarithmic) convergence time of the optimal congestion window size.[A.8] The enhanced version of BIC, and probably most popular version today, is CUBIC [105], which maintains the scalability and intra-fairness properties of its predecessor but also guarantees (*i*) RTT-fairness through RTT-independent congestion-window growth functions, and (*ii*) at least the same performance of the standard (Reno) congestion control through a dedicated mechanism.
There have been TCP extensions that have enhanced the loss-based congestion assessment. In wireless networks, where packet losses are not only related to congestion, the Westwood algorithm [154] enables the sender-side to distinguish between a congestion-related and an unrelated (random) loss without any support from the network, by estimating the last relevant flow rate and using this as a baseline to distinguish the loss type. There have also been variants that introduced completely new alternative mechanisms to loss-based congestion assessment for estimating the network's state. TCP Vegas [38] modifies the original TCP behavior implementation to proactively detect congestion from the estimated amount of enqueued packets at the bottleneck router. Like in the older attempt DUAL [243], Vegas uses RTT measurements to infer the bottleneck queue's length, and mitigates extremely the oscillatory patterns in network dynamics.[A.9]

A.2.2 In-network queueing/scheduling

TCP congestion control mechanisms [10] govern network dynamics from the edges of the network. Traffic management, service differentiation, and queue management [55] are the other aspect of controlling network dynamics, inside the network. Mechanisms are used within the network (in the routers) (*i*) to leverage the endpoints' congestion avoidance mechanisms, (*ii*) to enhance the congestion-feedback signal, (*iii*) to address issues of burstiness, synchronization, or unfairness, and (*iv*) to simply differentiate and prioritize traffic classes. It is useful to distinguish between two kinds of router algorithms (related to traffic control), which are closely related but address different objectives: "queue management policies" versus "scheduling algorithms". To a rough approximation, the first ones aim at addressing issues related with the (optimized) performance and utilization of router queues. The second ones determine which packet to send next and are used primarily aiming to manage the allocation

[A.8] Like other algorithms for high-*bandwidth-delay-product* networks (HS-TCP), BIC has also Limited Slow Start to avoid a large number of packets being lost when the congestion window is increased too fast. Similarly, BIC had up- and down-limited binary convergence to avoid the search range to be too wide or too narrow.

[A.9] We have cited only a few works in the context of TCP's congestion avoidance that (*i*) have contributed (conceptually, with standardized solutions, with significant implementation enhancements, *etc.*) to the current use of the Internet, (*ii*) apply complete end-to-end mechanism and do not rely on active participation of other elements within the network, and (*iii*) are cited throughout the thesis in experiments or discussions. For a wider but still very compact overview on the topic, refer to [5].

of bandwidth among flows. Scheduling algorithms and queue management are complementary.

A.2.2.1 (Active) queue management

"Tail-drop" is the simplest way for controlling the queue length. It consists in setting a maximum length (in terms of packets or bytes) for each queue, accepting packets for the queue until the maximum length is reached, and rejecting (drop) subsequent incoming packets until the queue decreases when a packet from the queue has been transmitted. Although this method had served the Internet well for years [205], it has two main drawbacks: (*i*) The "lock-out" phenomenon, often resulting from synchronization or other timing effects, makes one or few flows (connections) monopolize the queue space, preventing other flows from accessing it. (*ii*) Queues can remain (almost) full for long periods of time, a condition that eventually contributes to the reduction of the overall throughput.[A.10] In contrast with these two phenomena, ideally the sizing of queues should reflect the size of expected traffic bursts, and their steady-state fill level should be maintained low to achieve higher throughput, flow admissibility, and lower end-to-end delay [55].

The solution to the "full-queue" problem is dropping packets at routers before a queue becomes full, so that endpoints can respond to imminent congestion before buffers overflow – Active Queue Management (AQM). With AQM, the links play an active role in congestion control and avoidance. This brings various important benefits [55]: (*i*) reducing the number of packets dropped in routers (i.e., by working with full queues, there is no room left to absorb packet bursts, which are an intrinsic feature of traffic in packet networks [245]), (*ii*) reducing the delays experienced by flows, (*iii*) preventing lock-out behaviors by ensuring place in the buffers for incoming packets, and (*iv*) alleviating the recently identified bufferbloat condition[A.11].

Active Queue Management (AQM)

Traditionally, AQM mechanisms rely on the queue level to proactively signal congestion before buffers overflow – e.g., Random Early Detection (RED) [87] and its variants GRED [84], CHOKe [190], and DRED [192]. This is done probabilistically by dropping packets with a probability that increases as the estimated average queue size grows. Other proposals adjust the dropping probability based on packet-losses/marking (e.g., BLUE [81]), or on the arrival packet rate and on a predefined bandwidth allocations (e.g., SFED [124]). SHRED [56] uses instead an estimated average congestion window and the congestion window extracted from the arriving packet, whereas RED-Worcester [194] tries to reflect the average nature of the traffic and respectively targets a correct queue

[A.10] A burst of packets arriving in a full queue causes multiple packets to be dropped, which can make in turn flows to throttle back in a globally-synchronized fashion, inducing a subsequent period of lowered link utilization – i.e., global synchronization problem.

[A.11] *Bufferbloat* refers to having "excessively" large and frequently full buffers in the network that causes high latency and packet delay variation, leading to a reduced overall throughput.

length (small queues for delay-sensitive and bigger queues for throughput-sensitive traffic).[A.12] Recently, a second generation of AQM algorithms has appeared, which aim particularly at addressing the bufferbloat issue. They rely on queue delays for their operation instead of queue lengths: CoDel [178] relies on the direct monitoring of the buffer latency via time-stamping each packet on ingress, PIE [189] relies on the queue length and the dequeueing rate and infers the latency by exploiting Little's law.
Finally, next to AQM mechanisms, the introduction of Explicit Congestion Notification (ECN) [89] lets each link participate in congestion control by notifying explicitly users (i.e., packet "marking") when it detects congestion, as opposed to implicit signaling by dropping the packet.[A.13] ECN marking provides faster feedback and reduces the number of packets dropped by a TCP connection thereby retransmissions by TCP senders, latency, and jitter.

A.2.2.2 Scheduling algorithms

Scheduling algorithms affect the way in which packets from different sources interact with each other and, in turn, they influence the collective behavior of flow control algorithms. The simplest algorithms is the First Come First Served (FCFS), which is a single queue scheduler and in which the order of arrival of packets also determines the order in which they are forwarded over the output link. However, a packet scheduler should (*i*) provide some measure of isolation between, and fairness among, the competing flows, (*ii*) bound the total delay experienced by a packet in transit, and (*iii*) have a low operational complexity that guarantees high working speed. For this reason, multiple queue schedulers have been proposed. A first attempt in this direction was the Fair Queueing (FQ) algorithm [173], which maintains separate round-robin-served queues[A.14] for packets from each individual source, avoiding in this way a source from arbitrarily increasing its bandwidth share or the delay experienced by other sources. Pure round-robin scheduling however fails to guarantee a fair allocation in the case of packets with variable size. Instead, the Generalized Processor Sharing (GPS) [191] (an ideal, but not directly implementable, scheduling discipline), assumes fluid flows (i.e., infinitely divisible) and achieves perfect fairness and isolation among competing flows with variable sized packets.
A big category of scheduling algorithms includes time-stamp based algorithms. These scheduling algorithms try to emulate a GPS-system by computing a timestamp for each packet and then transmitting them in increasing order

[A.12] We have cited a few of the many proposals in the context of AQM. We refer the reader to [4] for a survey on works until 2011.

[A.13] With ECN-extension to IP/TCP, upon detecting incipient congestion, an ECN router set a bit in the packet header to one so as to notify the user that a link on its route is congested. The user then reacts to the mark as if a packet has been lost. Thus, the link avoids dropping the packet and still manages to convey congestion information to the user.

[A.14] A Round-Robin service rotationally selects packets to send out from all flows that have queued packets, ignoring empty queues.

of their timestamps. For example, the Weighted Fair Queuing (WFQ) [62] algorithm computes the timestamp of a packet as the time it would finish being serviced under a reference GPS server. Self-Clocked Fair Queuing [101], Virtual Clock [264], Leap Forward Virtual Clock [224] improve the timestamp computational efficiency by avoiding maintaining a reference GPS server and computing coarser timestamps. Time-stamp-based scheduling algorithms have good delay and fairness properties but generally need to sort packets by their deadlines, and therefore suffer from complexity logarithmic in the number of flows.

A much simpler algorithm to implement is Deficit Round Robin (DRR) [216]. DDR algorithm assigns to each flow (*i*) a quantum size proportionally to the weight of the flow and (*ii*) a deficit counter that measures the current unused portion of the allocated bandwidth. Packets of backlogged flows are transmitted in rounds, and in each round, each backlogged flow can transmit up to an amount of data equal to the sum of its quantum and deficit counter. The unused portion of this amount is carried over to the next round as the value of the deficit counter. As a result, DRR has a very simple implementation that requires low processing effort but has poor delay and output burstiness properties. Evolutions to the DRR-scheduling have been proposed. For example, to improve the burstiness problem, the Smoothed Round Robin [103] spreads the quantum allocated to a flow over an entire round by using a Weight Spread Sequence. Targeting better delay bounds, Bin Sort Fair Queuing [54] integrates notions of timestamp scheduling and uses an approximate bin-sorting mechanism to schedule packets.

An important issue with multiple-queue algorithms is performance and memory consumption as the maximum possible number of flows increases. Stochastic Fairness Queueing (SFQ) [158] scheduler is an evolution of FQ, which has been designed to be CPU- and flow- friendly. It employs a (constantly changing) hashing algorithm that statistically divides traffic over a limited number of queues, which are served in a round-robin style.

Work-conserving

All discussed scheduling algorithms operate under a *work-conserving* discipline. This means the server of a queue is never idle until there is a packet to be sent in the queue. The benefit of adopting work-conserving disciplines is the low *average* packet delay. The scheduling tries to exploit the maximum path capacity by letting the upper transmission rate be inherently limited by the network's bottleneck features (e.g., bottleneck link's capacity). The downside, which up until recently was not seriously considered,[A.15] is the uncontrollability of the performance exhibited by the overall communication system. Flow interactions at shared network elements (e.g., switches and routers) are "left to chance".

Non-work-conserving

By contrast, *non-work conserving* disciplines delay the transmission of packets to eligible times. This makes the traffic out of switches more predictable and, at the same time, reduces the delay jitter experienced by a connection and the

[A.15] Non-work-conserving scheduling was already proposed in the nineties by Zhang *et al.* [262, 263].

burstiness of traffic, allowing in this way smaller buffers at output queues.

A.3 Step by step analysis of the Enzymatic traffic controller

This section provides details about the analysis of the Enzymatic traffic controller introduced in Section 3.3. The procedure follows exactly the one we explained in Section 2.3.
The dynamics of the chemistry-inspired rate controller graphically represented in Figure A.3 are described by a system of coupled ODEs, which can be easily extracted from the reaction sets:

Fluid model

$$\dot{c_{\mathrm{S}}} = \lambda - k_1 c_{\mathrm{S}} c_{\mathrm{E}} \tag{A.4a}$$
$$\dot{c_{\mathrm{ES}}} = k_1 c_{\mathrm{S}} c_{\mathrm{E}} - k_2 c_{\mathrm{ES}} \tag{A.4b}$$
$$\dot{c_{\mathrm{E}}} = k_2 c_{\mathrm{ES}} - k_1 c_{\mathrm{S}} c_{\mathrm{E}}. \tag{A.4c}$$

By solving (A.4) with respect to species concentrations, when the left-hand side is equal to 0, we find steady-state concentrations:

Steady state

$$c_{\mathrm{S}}{}^{*} = \frac{\lambda}{k_2 e_0 - \lambda} \frac{k_2}{k_1} \tag{A.5}$$
$$c_{\mathrm{E}}{}^{*} = \frac{k_2 e_0 - \lambda}{k_2} \tag{A.6}$$
$$c_{\mathrm{ES}}{}^{*} = \frac{\lambda}{k_2}. \tag{A.7}$$

Since there exists a conserved moiety E-ES (i.e. a closed loop where the sum of the concentrations of involved species is constant: $c_{\mathrm{ES}} + c_{\mathrm{E}} = const. = e_0$), if we impose that molecular concentrations must be positive quantities, we limit the validity of (A.5) to the following region:

Steady-state results validity

$$\lambda < k_2 e_0. \tag{A.8}$$

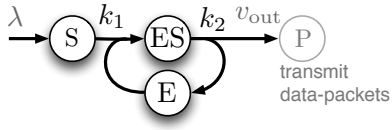
The next step is the definition of the rate vector as $\mathbf{v} = [k_1 c_{\mathrm{S}} c_{\mathrm{E}} \; k_2 c_{\mathrm{ES}} \; \lambda]$ and of the stoichiometric matrix (reaction network topology) as

Reaction network structure

$$\boldsymbol{\Xi} = \begin{bmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ -1 & 1 & 0 \end{bmatrix}$$

whose rows represent the species $\mathbf{s} = [\mathrm{S} \; \mathrm{ES} \; \mathrm{E}]$ and columns represent the reactions $\mathbf{r} = [r_1 \; r_2 \; r_\lambda]$.[A.16] For example, the first column expresses the fact that reaction r_1 consumes an S-molecule and an E-molecule and produces an

[A.16] We abuse the notation and represent the arrival process as a reaction $r_\lambda : \emptyset \xrightarrow{\lambda} \mathrm{S}$.



$$r_1: \quad \mathrm{S} + \mathrm{E} \xrightarrow{k_1} \mathrm{ES} \qquad (A.9)$$

$$r_2: \quad \mathrm{ES} \xrightarrow{k_2} \mathrm{E} + \mathrm{P} \qquad (A.10)$$

Figure A.3: *The Enzymatic traffic controller: graphical reaction network and reaction set.*

ES-molecule. Each structural conservation, e.g. the moiety E-ES, corresponds to linearly dependent rows in the stoichiometric matrix. As suggested in [201], we avoid redundant terms and proceed taking into account the reduced stoichiometric matrix $\boldsymbol{\Xi}_R$ that can be obtained removing the row related to E-species from $\boldsymbol{\Xi}$, i.e. the last row:

$$\boldsymbol{\Xi}_R = \begin{bmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix}.$$

As it is indicated in [210], the relationship between $\boldsymbol{\Xi}$ and $\boldsymbol{\Xi}_R$ is formalized by the link matrix $\mathbf{L}$ (such that $\boldsymbol{\Xi} = \boldsymbol{\Xi}_R\mathbf{L}$), which makes explicit how dependent species' concentrations, $\mathbf{c}_d = [c_{\mathrm{E}}]$, can be derived from independent species' concentrations, $\mathbf{c}_i = [c_{\mathrm{S}}\ c_{\mathrm{ES}}]$:

$$\mathbf{L} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix}.$$

The link matrix is composed by the identity matrix whose size equals the number of independent species (rank($\boldsymbol{\Xi}$)), and an additional row vector describing the relationship between dependent and independent species. In this case, there is one moiety only and the additional row vector expresses that the dependent species E can be derived by subtracting ES-species' concentration c_{ES} from the initial amount that characterizes the moiety (e_0). Namely, species can be represented in terms of independent species only as $\mathbf{c} = \mathbf{L}\mathbf{c}_i + \mathbf{T}$ where $\mathbf{T}$ is an appropriate constant vector – i.e. referring to our rate-controller example we have $[c_{\mathrm{S}}\ c_{\mathrm{ES}}\ c_{\mathrm{E}}]^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} c_{\mathrm{S}} \\ c_{\mathrm{ES}} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ e_0 \end{bmatrix}$.

In order to represent the Enzymatic reaction model shown in Figure A.3 as a LTI system, we must evaluate the behavior of the model around steady states (nominal states) and input signal values. Namely, we calculate the Jacobians of the rate vector with respect to state vector $\mathbf{c}$ and perturbation vector $\mathbf{p}$ respectively. The state vector $\mathbf{c}$ includes independent species' concentrations $\mathbf{c}_i$ and dependent species' concentrations $\mathbf{c}_d$, and, assuming we want to study the Enzymatic controller's response to variations on packet arrival rate, we fix

the parameter vector to $\mathbf{p} = [\lambda]$:

Elasticity coefficients

$$\frac{\partial \mathbf{v}}{\partial \mathbf{c}} = \begin{bmatrix} k_1 c_{\mathrm{E}} & 0 & k_1 c_{\mathrm{S}} \\ 0 & k2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\frac{\partial \mathbf{v}}{\partial \mathbf{p}} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T.$$

Control matrixes

We know that the state matrix $\mathbf{A}$ defines how a perturbation on the concentrations affects their future changes and the input matrix $\mathbf{B}$ indicates how external perturbations cause fluctuations of the system state. By using the reduced form of the stoichiometric matrix $\boldsymbol{\Xi}_R$, we define $\mathbf{A}$ and $\mathbf{B}$ as follows:[A.17]

$$\mathbf{A} = \boldsymbol{\Xi}_R \cdot \left.\frac{\partial \mathbf{v}}{\partial \mathbf{c}}\right|_{(\mathbf{c}^*, \mathbf{p}^*)} \cdot \mathbf{L}$$

$$\mathbf{B} = \boldsymbol{\Xi}_R \cdot \left.\frac{\partial \mathbf{v}}{\partial \mathbf{p}}\right|_{(\mathbf{c}^*, \mathbf{p}^*)}.$$

The output matrix $\mathbf{C}$ and the feedforward matrix $\mathbf{D}$ must be chosen according to the response we want to analyze. Since we look at the fluctuations of the reaction rates with respect to a perturbation of the input (the output vector reports variations on reaction rates),[A.18] we define

$$\mathbf{C} = \left.\frac{\partial \mathbf{v}}{\partial \mathbf{c}}\right|_{(\mathbf{c}^*, \mathbf{p}^*)} \cdot \mathbf{L} \qquad \mathbf{D} = \left.\frac{\partial \mathbf{v}}{\partial \mathbf{p}}\right|_{(\mathbf{c}^*, \mathbf{p}^*)}.$$

Again, the output matrix $\mathbf{C}$ includes the relationship between dependent and independent species, which is formalized by the link matrix $\mathbf{L}$.
We can now calculate the TF $\mathbf{H}(s)$ in the Laplace domain, which describes the transient behavior of output deviations $\mathbf{y}$ from the nominal state with respect to perturbations on the chosen input $\mathbf{u} = [\lambda]$, namely the system sensitivity.
Since the triggering rate for packet transmissions is the rate of r_2-reaction $v_{\mathrm{out}} = v_2 (= k_2 c_{\mathrm{ES}})$, we can focus on the last row of the TF matrix, the row vector that describes how v_2 deviates from steady state:

Transfer functions in Laplace domain

$$H(s) = \begin{cases} \frac{k_1 k_2 {c_{\mathrm{E}}}^*}{s^2 + s(k_2 + k1({c_{\mathrm{S}}}^* + {c_{\mathrm{E}}}^*)) + k_1 k_2 {c_{\mathrm{E}}}^*} & (\lambda < k_2 e_0 = R) \\ 0 & (\lambda \geq k_2 e_0 = R, c_{\mathrm{S}} \to \infty) \end{cases}$$

[A.17] This definition is a direct consequence of describing the state-evolution as

$$\dot{\mathbf{x}}(t) = \boldsymbol{\Xi}_R \cdot \partial \mathbf{v} / \partial \mathbf{c}|_{(\mathbf{c}^*, \mathbf{p}^*)} \cdot \mathbf{L} \cdot \mathbf{x}(t) + \boldsymbol{\Xi}_R \cdot \partial \mathbf{v} / \partial \mathbf{p}|_{(\mathbf{c}^*, \mathbf{p}^*)} \cdot \mathbf{u}(t).$$

[A.18] The output vector is

$$\mathbf{y}(t) = \partial \mathbf{v} / \partial \mathbf{c}|_{(\mathbf{c}^*, \mathbf{p}^*)} \cdot \mathbf{L} \cdot \mathbf{x}(t) + \partial \mathbf{v} / \partial \mathbf{p}|_{(\mathbf{c}^*, \mathbf{p}^*)} \cdot \mathbf{u}(t).$$

$$H(s) = \begin{cases} \frac{k_1(k_2 e_0 - \lambda)}{s^2 + s\left(k_2 + k_1\left(\frac{k_2 e_0 - \lambda}{k_2} + \frac{k_2 \lambda}{k_1(k_2 e_0 - \lambda)}\right)\right) + k_1(k_2 e_0 - \lambda)} & (\lambda < k_2 e_0 = R) \\ 0 & (\lambda \geq k_2 e_0 = R, c_S \to \infty) \end{cases}$$

The roots of the denominator of $H(s)$ (poles) are

$$s = 1/2(-\sqrt{(k_1({c_E}^* + {c_S}^*) + k_2)^2 - 4{c_E}^* k_1 k_2} - k_1({c_E}^* + {c_S}^*) - k_2)$$
$$s = 1/2(\sqrt{(k_1({c_E}^* + {c_S}^*) + k_2)^2 - 4{c_E}^* k_1 k_2} - k_1({c_E}^* + {c_S}^*) - k_2).$$

Note that being k-coefficients and molecular concentrations positive quantities, we have that $\sqrt{(k_1({c_E}^* + {c_S}^*) + k_2)^2 - 4{c_E}^* k_1 k_2} < k_1({c_E}^* + {c_S}^*) + k_2$ for all values. Thus, we further assert that poles have always negative real parts.

A.4 Step by step analysis of the distributed traffic rate controller

This section gives details about the transient (sensitivity) analysis of the Distributed Traffic Rate Controller (DTRC) introduced in Section 3.4.1. The procedure follows exactly the one we explained in Section 2.3 and the obtained results are those used in the analysis of the DTRC Section 3.4.2.

Reaction network structure

We start from the structure of the reaction network implementing DTRC's functionalities; the stoichiometric matrix of the chemical model shown in Figure A.4 is

$$\mathbf{\Xi} = \begin{bmatrix} 0 & 1 & 1 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

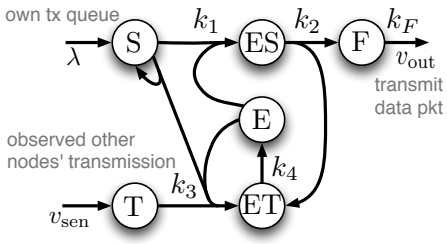
whose rows are related to the species vector $\mathbf{s} = [\text{ET S T F ES E}]$ and columns are related to the reaction vector $\mathbf{r} = [r_1\ r_2\ r_3\ r_4\ r_{\text{int}}\ r_{\text{ext}}\ r_{\text{out}}]$.

We know that the mass conservation principle applies to the reaction loop ES-ET-E and imposes the following constraint:

$$E_0 = c_{\text{ES}} + c_{\text{E}} + c_{\text{ET}} = const. \tag{A.11}$$

Each structural conservation (e.g. conserved moiety expressed in (A.11)) in the network, correspond to linearly dependent rows in the stoichiometric matrix. We can thus proceed taking into account the reduced stoichiometric matrix $\mathbf{\Xi}_R$ that, for instance, can be obtained removing the row related to E-species from $\mathbf{\Xi}$, i.e. the last row:

$$\mathbf{\Xi}_R = \begin{bmatrix} 0 & 1 & 1 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$



$$
\begin{aligned}
r_{\text{int}} &: \quad \emptyset \xrightarrow{\lambda} \text{S} \\
r_{\text{ext}} &: \quad \emptyset \xrightarrow{v_{\text{sen}}} \text{T} \\
r_1 &: \quad \text{S} + \text{E} \xrightarrow{k_1} \text{ES} \\
r_2 &: \quad \text{ES} \xrightarrow{k_2} \text{ET} + \text{F} \\
r_3 &: \quad \text{T} + \text{E} + \text{S} \xrightarrow{k_3} \text{ET} + \text{S} \\
r_4 &: \quad \text{ET} \xrightarrow{k_4} \text{E} \\
r_{\text{out}} &: \quad \text{F} \xrightarrow{k_{\text{F}}} \emptyset
\end{aligned}
$$

Figure A.4: *Adaptive Distributed Traffic Rate Controller (DTRC): graphical reaction network and reaction set.*

As it is suggested in [210], the relationship between $\boldsymbol{\Xi}$ and $\boldsymbol{\Xi}_R$ is formalized by the link matrix $\mathbf{L}$, which makes explicit how dependent species' concentrations, $\mathbf{c}_d = [c_{\text{E}}]$, can be derived form independent species' concentrations, $\mathbf{c}_i = [c_{\text{ET}}\ c_{\text{S}}\ c_{\text{T}}\ c_{\text{F}}\ c_{\text{ES}}]^T$:

$$
\mathbf{L} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 \\
-1 & 0 & 0 & 0 & -1
\end{bmatrix}.
$$

The link matrix is composed by the identity matrix whose size equals the number of independent species, and, in this case since there is one moiety only, a row vector expressing that the dependent species' concentration c_{E} can be derived by subtracting the concentration of species ET and ES from the initial amount that characterizes the moiety (e_0).

Since we are interested in how the protocol responds to a certain internal demand (i.e. λ) and external inflow (i.e. v_{sen}), the vector of perturbations is defined as $\mathbf{p} = [\lambda\ v_{\text{sen}}]$. As a next step, by following [201], we approximate the behavior of the model to linear model whose input vector $\mathbf{u}(t) = \mathbf{p}(t) - \mathbf{p}^*$ denotes the deviations from input signal values and whose state vector $\mathbf{x}(t) = \mathbf{c}(t) - \mathbf{c}^*$ denotes the deviations from nominal state:

$$
\dot{\mathbf{x}}(t) = \boldsymbol{\Xi}_R \cdot \left.\frac{\partial \mathbf{v}}{\partial \mathbf{c}}\right|_{(\mathbf{c}^*, \mathbf{p}^*)} \cdot \mathbf{L} \cdot \mathbf{x}(t) + \boldsymbol{\Xi}_R \cdot \left.\frac{\partial \mathbf{v}}{\partial \mathbf{p}}\right|_{(\mathbf{c}^*, \mathbf{p}^*)} \cdot \mathbf{u}(t). \tag{A.12}
$$

The terms $\frac{\partial \mathbf{v}}{\partial \mathbf{c}}$ and $\frac{\partial \mathbf{v}}{\partial \mathbf{p}}$ are the Jacobians of the vector of reaction rates (namely, $\mathbf{v} = [k_1 c_{\text{S}} c_{\text{E}}\ k_2 c_{\text{ES}}\ k_3 c_{\text{T}} c_{\text{E}} c_{\text{S}}\ k_4 c_{\text{ET}}\ \lambda\ v_{\text{sen}}\ k_F c_{\text{F}}]$) with respect to state vector $\mathbf{c}$

and perturbation vector $\mathbf{p}$, respectively:

Elasticity coefficients

$$\frac{\partial \mathbf{v}}{\partial \mathbf{c}} = \begin{bmatrix} 1 & k_1 c_\mathrm{E} & 0 & 0 & 0 & k_1 c_\mathrm{S} & 0 & 0 \\ 0 & 0 & 0 & 0 & k_2 & 0 & 0 & 0 \\ 0 & k_3 c_\mathrm{E} c_\mathrm{T} & k_3 c_\mathrm{E} c_\mathrm{S} & 0 & 0 & k_3 c_\mathrm{S} c_\mathrm{T} & 0 & 0 \\ k_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & k_F & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\frac{\partial \mathbf{v}}{\partial \mathbf{p}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}.$$

Since we have linearized the model, we can make use of the well-known state-space description of Linear Time Invariant (LTI) systems:

$$\dot{\mathbf{x}}(t) = \mathbf{A} \cdot \mathbf{x}(t) + \mathbf{B} \cdot \mathbf{u}(t) \tag{A.13a}$$

$$\mathbf{y}(t) = \mathbf{C} \cdot \mathbf{x}(t) + \mathbf{D} \cdot \mathbf{u}(t). \tag{A.13b}$$

The definitions of the state matrix $\mathbf{A}$ and the input matrix $\mathbf{B}$ slightly differ from the ones given in Section 2.3.3.2 as we have to use the reduced form of the stoichiometric matrix $\mathbf{\Xi}_R$:

$$\mathbf{A} = \mathbf{\Xi}_R \cdot \left.\frac{\partial \mathbf{v}}{\partial \mathbf{c}}\right|_{(\mathbf{c}^*, \mathbf{p}^*)} \cdot \mathbf{L} \qquad \text{and} \qquad \mathbf{B} = \mathbf{\Xi}_R \cdot \left.\frac{\partial \mathbf{v}}{\partial \mathbf{p}}\right|_{(\mathbf{c}^*, \mathbf{p}^*)}.$$

Being interested in the fluctuations of reaction rates with respect to changes of input signals, we define output and feedforward matrixes as follows:

$$\mathbf{C} = \left.\frac{\partial \mathbf{v}}{\partial \mathbf{c}}\right|_{(\mathbf{c}^*, \mathbf{p}^*)} \cdot \mathbf{L} \qquad \text{and} \qquad \mathbf{D} = \left.\frac{\partial \mathbf{v}}{\partial \mathbf{p}}\right|_{(\mathbf{c}^*, \mathbf{p}^*)}.$$

($\mathbf{C}$ must include the relationship between dependent and independent queues, i.e the link matrix $\mathbf{L}$.)
By assuming that the initial conditions are set to steady state (i.e. there are no initial perturbation: $\mathbf{x}_0 = \mathbf{0}$), the TF of the LTI system results in

$$\mathbf{H}(s) = \frac{\mathbf{y}(s)}{\mathbf{u}(s)} = \mathbf{C}\,(s \cdot \mathbf{I} - \mathbf{A})^{-1}\,\mathbf{B} + \mathbf{D}.$$

Relationships λ-v_out and v_sen-v_out: $H_\mathrm{o\text{-}i}$ and $H_\mathrm{o\text{-}e}$

The TF matrix $\mathbf{H}$ has 14 response coefficients, describing the transient behavior of all (seven) output deviations $\mathbf{y}$ from the nominal state to perturbations on all (two) input signals $\mathbf{u}$. Since the trigger for the transmission is r_out-reaction, i.e., $v_\mathrm{out} = v_\mathrm{out}(= k_F c_\mathrm{F})$, we focus on the last row of the TF-matrix, a vector $[H_\mathrm{o\text{-}i} \;\; H_\mathrm{o\text{-}e}]$ reporting the transient behavior of steady-state output deviations of r_out-reaction's rate with respect to perturbations on inputs λ and v_sen.

A.5 Enzymatic Media Access Control: E-MAC

In this section, we investigate the use of the distributed Enzymatic-controller model (already introduced in Section 3.4 in the context of distributed traffic shaping algorithms) to build a Media Access Control (MAC) protocol. We exploit the key-ability of the Enzymatic model to limit the media access rate in order to optimize the system in terms of "stability". Our Enzymatic Media Access Control protocol (E-MAC) guarantees the same maximum throughput of the Pure ALOHA protocol but improves the performances in terms of stability. Like current Media Access Control (MAC) protocols, the E-MAC requires collision detection to be able to retransmit colliding packets. Differently from the Pure ALOHA protocol, the E-MAC requires also a mechanism to detect the amount of transmissions performed by the other participants. The delivery of such an information does not have particular constraints in terms of time. Indeed, differently from the Carrier Sense Multiple Access (CSMA) family, it is not important to know when exactly the media is busy but it is sufficient an average estimate of its use.
Our approach belongs to the contention-type or random transmission protocols in which the success of a transmission is not guaranteed in advance. The reason is that whenever two or more users are transmitting on the shared channel simultaneously, a collision occurs and the data cannot be received correctly. In this case, packets may have to be transmitted and retransmitted until eventually they are correctly received. The E-MAC is very close to the concepts of the pure ALOHA protocol, and it is motived by the instability exhibited by the pure ALOHA protocol. For these reasons, in the next, we first summarize the main features of ALOHA (via analysis) (Section A.5.1). We then show how these features can be preserved/enhanced by applying the distributed Enzymatic-controller model (Section A.5.2).

A.5.1 The ALOHA protocol

The ALOHA family of protocols is probably the richest family of multiple access protocols. Its popularity is due first to its seniority, and second to its simplicity. Many local area networks of today implement some sophisticated variants of this family's protocols.
The pure ALOHA protocol is very simple [1]: A newly generated packet is transmitted immediately hoping for no interference by others. If two or more packet transmissions overlap in time, a collision is caused and none of the colliding packets is received correctly and they have to be retransmitted. The users whose packets collide with one another are called the colliding users. At the end of every transmission, each user knows whether its transmission was successful or a collision took place and, in case of an unsuccessful transmission, every colliding user schedules its retransmission to a random time in the future, independently of what other users do. This randomness is required to ensure that the same set of packets does not continue to collide indefinitely.

In the pure ALOHA, the channel is error-free without capture: whenever a transmission of a packet does not interfere with any other packet transmission, the transmitted packet is received correctly. Moreover, the model considers a single-hop system with an infinite population that generates packets of equal length according to a Poisson process with rate λ pkt/s.[A.19] Since the population is infinite, each packet can be considered as if it belongs to a different user. Hence, each newly arrived packet can be assigned to an idle user (i.e. one that does not have a packet to retransmit). This assumption enables to interchange the roles of users and packets and to consider only the points in time when packet transmission attempts are made. By observing the channel over time, we define a point process consisting of scheduling points, i.e., the points in which packets are scheduled for transmission. The scheduling points include both the generation times of new packets and the retransmission times of previously collided packets. Let the rate of the scheduling points be g pkt/s. The parameter g is referred to as the offered load to the channel or the transmission rate. Clearly, since not all packets are successful on their first attempted transmission, $g > \lambda$.

The exact characterization of the scheduling points process is extremely complicated, but can be overcome by assuming that this process is a Poisson process (with rate g, of course). Note that a Poisson process implies independence between events in non-overlapping intervals, which cannot be the case here because of the dependence between the interval containing the original transmission and the interval containing a retransmission of the same packet. It can be shown, however, that if the retransmission schedule is chosen uniformly from an arbitrarily large interval then the number of scheduling points in any interval approaches a Poisson distribution. As we anticipated, the Poisson assumption is used because it makes the analysis of ALOHA-type systems tractable and predicts successfully their maximal throughput.

We assume that all packets stay on the media for T seconds. Namely, we assume that the channel has limited capacity of $R = 1/T$ pkt/s. Let consider a packet (new or old) scheduled for transmission at some time t. This packet will be successful if no other packet is scheduled for transmission in the interval $(t - T, t + T)$, so-called vulnerable period. Thus, the probability of successful transmission is the probability of no scheduled packet in an interval of length $2T$ is $P_{\text{succ}} = e^{-2gT}$. [A.20] Thus, the rate of successfully transmitted packets is simply gP_{succ}. The *throughput* μ, which is the fraction of time that carries useful information, turns out to be $\mu = Tge^{-2gT}$. We can further normalize the transmission rate g with respect to the capacity limitation R pkt/s of the transmission and obtain the well-known equation of the throughput in terms

[A.19] In this, section we refer to rates as average values.

[A.20] The transmission is modeled as Poisson process with parameter g, thus the probability that k packets are being transmitted in the period $2T$ is $P(k) = \frac{(g)^k}{k!}e^{-g\cdot 2T}$.

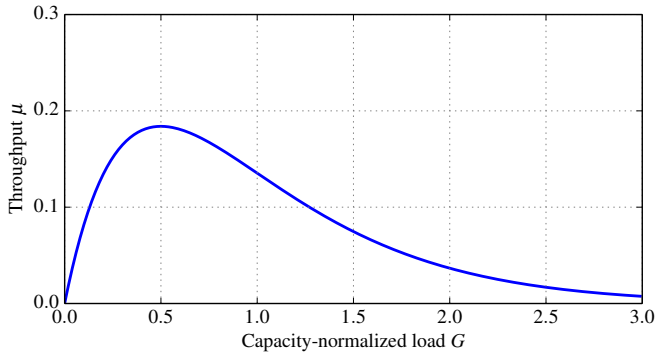


Figure A.5: *Pure ALOHA: Throughput* vs. *offered load.*

of normalized channel load $G = g/R$:

$$\mu = Ge^{-2G}. \tag{A.14}$$

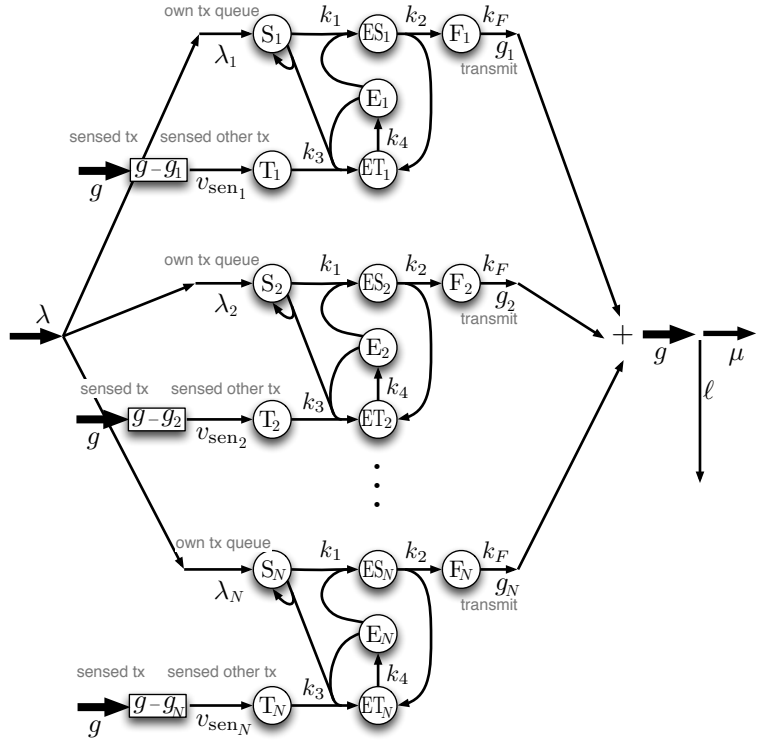
ALOHA throughput

The relation between μ and G is depicted in Figure A.5, which is typical to many ALOHA type protocols. We can observe that the maximal throughput[A.21], $\mu_{\max} = 0.5/e \sim 0.18$, is obtained at $G = 1/2$.
By referring to Figure A.5, we can also argue about the "*stability*" performance of the Pure ALOHA protocol: for $G < 1/2$ the protocol is (conditionally) stable, while for $G > 1/2$ it is conditionally unstable, meaning that if the offered load g increases beyond the point $R/2$, the system will continue to drift towards lower throughputs, eventually reaching zero.

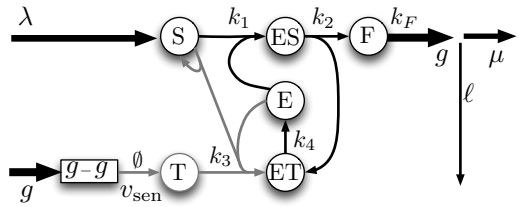
A.5.2 The E-MAC protocol

We propose to adopt a stochastic protocol similar to the ALOHA protocol, which has an improved stability thanks to an upper-limited media access rate. The scenario is very similar to that described in Section 3.4: we have N nodes that want to access a shared resource. The latter has a limited capacity of maximum R pkt/s. To guarantee the maximum throughput of $\mu_{\max} \sim 0.18$, we limit the aggregate access rate to $G = 1/2$. By extracting inter-reaction times from the exponential distribution (a design choice and not a modeling assumption), we can safely consider the transmission process of one or more users as a Poisson process (for the ALOHA protocol, we can just *assume* independent exponentially-distributed scheduling times in order to make the analysis tractable). We can thus reuse the previously found results regarding the throughput in the ALOHA case, and those found in Section 3.4.2 that are related to the distributed Enzymatic model (DTRC model). To proceed

[A.21] The maximal value of the throughput is also referred to as the capacity of the pure ALOHA channel.

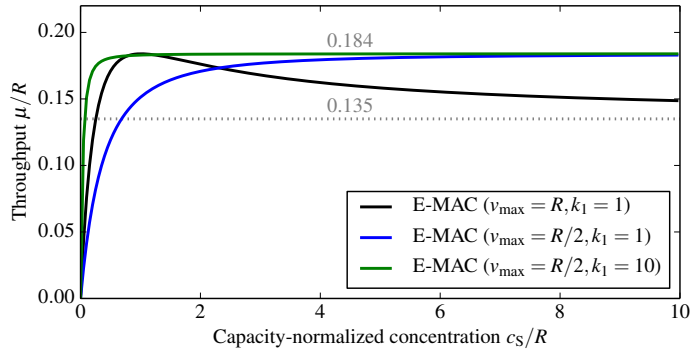


(a) Explicit model

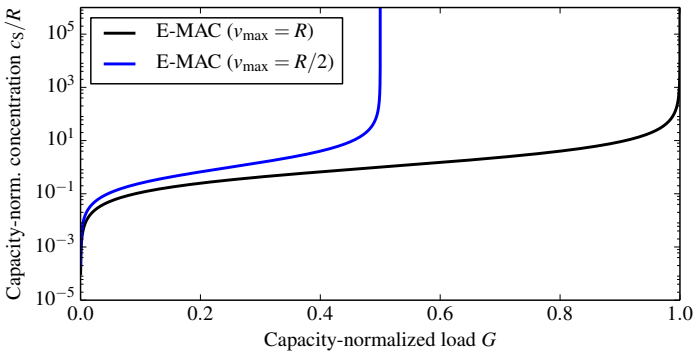


(b) Equivalent global model

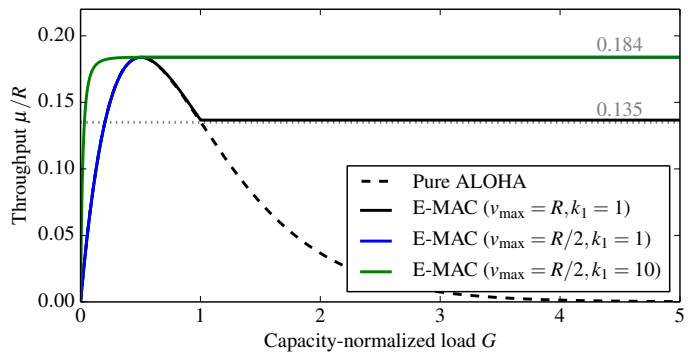
Figure A.6: *Reaction network model to control the* access *of N nodes to a shared media. (a) The reaction network describes the whole system in detail. (b) The equivalent chemical model has as input the aggregated load $\lambda = \sum_i \lambda_i$ and as output the aggregated access rate $g = \sum_i g$. The sensing rate is null by assuming no measurement errors: $v_{sen} = g - \sum_i g_i = 0$.*



(a) Throughput *vs.* concentration



(b) Concentration *vs.* load



(c) Throughput *vs.* load

Figure A.7: *E-MAC: (a) Throughput over aggregate concentration of S-molecules, of a system of N Enzymatic-controlled nodes. (b) Dependency of concentration c_S upon the load rate λ. (c) Throughput over the load rate and comparison with ALOHA throughput.*

further, we rewrite the whole system made of N controlled nodes (shown in Figure A.6(a)) as a unique reaction system with rates λ and g (or $G = Rg$ in the normalized version) made of the sum of local rates, as depicted in Figure A.6(b). Note that, in this equivalent global model, the "sensed rate" is zero, i.e., $v_{\text{sen}} = 0$. Indeed, the v_{sen}-quantity is taken as the difference between the measured rate accounting for all nodes' transmission and the actual aggregated rate of nodes' transmission, which have the same value if we assume no sensing errors. The throughput μ as a function of "load" concentration c_{S} turns out to be

E-MAC throughput

$$\mu = Ge^{-2G}\Big|_{\substack{k_2 \gg k_4 \\ k_4 E_0 = 1/2}} = \frac{0.5k_2k_1c_{\text{S}}}{k_2k_4 + (k_2 + k_4)k_1c_{\text{S}}} e^{-\frac{k_2k_1c_{\text{S}}}{k_2k_4+(k_2+k_4)k_1c_{\text{S}}}}. \quad \text{(A.15)}$$

We recall that the concentration c_{S} increases without bound as soon as the load rate λ is too high, i.e., $\lambda > v_{\max} = R/2$. Otherwise, the amount of S-molecules is finite and equal to the following value (for $v_{\text{sen}} = 0$):

$$c_{\text{S}} = \frac{\lambda k_2 k_4}{E_0 k_2 k_4 - \lambda(k_2 + k_4)}\Big|_{k_2 \gg k_4} = \frac{\lambda k_4}{E_0 k_4 - \lambda} \qquad \big(\lambda < v_{\max} = R/2\big). \quad \text{(A.16)}$$

Figure A.7 shows the meaning of equations (A.15) and (A.16): The normalized throughput of a system of N Enzymatic-controlled nodes has the same maximum value of a system of ALOHA nodes (i.e., $\mu \sim 0.184$). However, by limiting the aggregated access rate to half the media capacity R by means of DTRCs, we can stabilize the throughput to this maximum value and avoid the throughput to drop to zero for high loads (in contrast with the ALOHA case, where the throughput drops to zero when the load increases above the "sweet" spot $G = 0.5$). By trying to exploit the maximum capacity and setting the DTRCs to have $v_{\max} = R$, the normalized throughput for high loads decreases to $1/e^2 \sim 0.135$. As we knew from the analysis in Section 3.4.2, we can make the system more sensible to the load, and thus obtain higher throughput values for lower load values, by increasing the k_1-coefficient. This however can bring to oscillatory behavior when the sensing of the media is affected by high delay.

A.6 Service-class differentiation with DTRCs

The Distributed Traffic Rate Controller (DTRC) enables a `DiffServ`-like mechanism [223]. We can still refer to the introduced reaction model (see Figure A.4) and weight each traffic class by differentiating the value of reaction coefficient k_5 from flow to flow. In this section, we first explain the reasons behind this choice and then report early experiment's results.

We want that a prioritized (having higher weight) flow is advantaged over other flows having lower weights. Intuitively, in order to do this, we cannot simply alter the input rate at which enqueued bytes are notified to the chemical model (i.e., differ the mapping byte-to-molecule from node to node). Indeed, the control mechanism has been designed to act fairly and limit the output

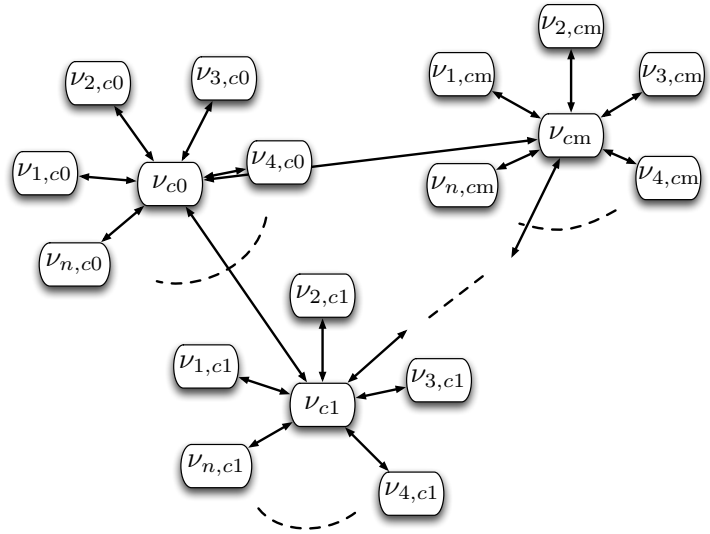
	$k_{5,4} = 1$(no prio)	$k_{5,4} = 30$	$k_{5,4} = 50$	$k_{5,4} = 100$
$\lambda_1 = 100$	$v_{\text{out},1} = 100$	$v_{\text{out},1} = 100$	$v_{\text{out},1} = 100$	$v_{\text{out},1} = 100$
$\lambda_2 = 300$	$v_{\text{out},2} = 266$	$v_{\text{out},2} = 230$	$v_{\text{out},2} = 180$	$v_{\text{out},2} = 100$
$\lambda_3 = 100$	$v_{\text{out},3} = 100$	$v_{\text{out},3} = 100$	$v_{\text{out},3} = 100$	$v_{\text{out},3} = 100$
$\lambda_4 = 600$	$v_{\text{out},4} = 267$	$v_{\text{out},4} = 340$	$v_{\text{out},4} = 440$	$v_{\text{out},4} = 600$
$\lambda_5 = 400$	$v_{\text{out},5} = 266$	$v_{\text{out},5} = 230$	$v_{\text{out},5} = 180$	$v_{\text{out},5} = 100$

Table A.1: *Traffic class prioritization at host 4 via $k_{4,4}$-setting, with $k_{4,i} = 1$ at all other hosts ($i \neq 4$), in a 5-host network. All local average loads λ_i and transmission rate $v_{out,i}$ are in pkt/s. Rate values have been averaged over 20s-period and rounded.*

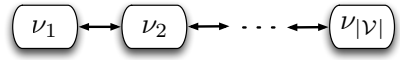
rate v_{out} proportionally to the difference of input rate λ and the fair sharing value. Thus, thanks to r_3 reaction, an increase of λ_i to prioritize a certain flow i, would not have any appreciable effect on $v_{\text{out},i}$. We can neither differentiate the amount of concentrations characterizing the conservation loop as it would results in a modification of the predefined limit on the aggregate rate, thus against our ultimate goal of limiting the aggregate rate to the predefined value. The flow prioritization stems from the parametrization of the reaction model in Figure A.4, by calibrating opportunely the reaction coefficients. However, we must keep in mind that these coefficients impact on static and dynamical features of the controller: k_F-coefficient affects the cutoff frequency, k_4 modifies the predefined limit on the aggregate rate, k_1, k_2 and k_3 affect the response speed in the unsaturated regime. The only left coefficient that allows weighting traffic flows without altering the other features of the controller is the k_5-coefficient. Indeed, besides avoiding system stale states, the k_5-coefficient controls the speed of the linear reaction draining T-species and thus affects the accounting of others' transmissions. Thus, by differentiating k_5-values among the different DTRCs, we are able to enforce different forwarding regimes for the different traffic classes, while still satisfying the aggregate rate limit and guaranteeing the efficiency of the system.[A.22]

In simulations, we were able to advantage limited flows by increasing k_5 from 1 up to 100 s^{-1} , without affecting performances of non-limited flows (i.e., with output transmission rate lower than the rate share). How much we increased k_5 determined how much the prioritized host increased its transmission rate, depending also on its local load. We simulated five hosts competing for a shared resource. We set the aggregate rate limit to 1000 pkt/s and used the settings previously reported in Table 3.7. We prioritized flows via differentiation of k_5-value and studied the effectiveness of the this prioritization-mechanism in saturated as well as unsaturated regime. We considered the situation where hosts had stable generation rate λ and transmission rate v_{out}. In Table A.1, we compare the measured average values of generation rate λ_i at host i, and achieved transmission rate $v_{\text{out},i}$ for the corresponding $k_{5,i}$-value. The average rate values have been calculated over a period of 20 s.

[A.22] The differentiation of k_5-value has impact also on the speed at which inactive nodes (flows) return to the initial state. However, this effect is not appreciable in the output rates.



(a) Clustered network



(b) Inline network

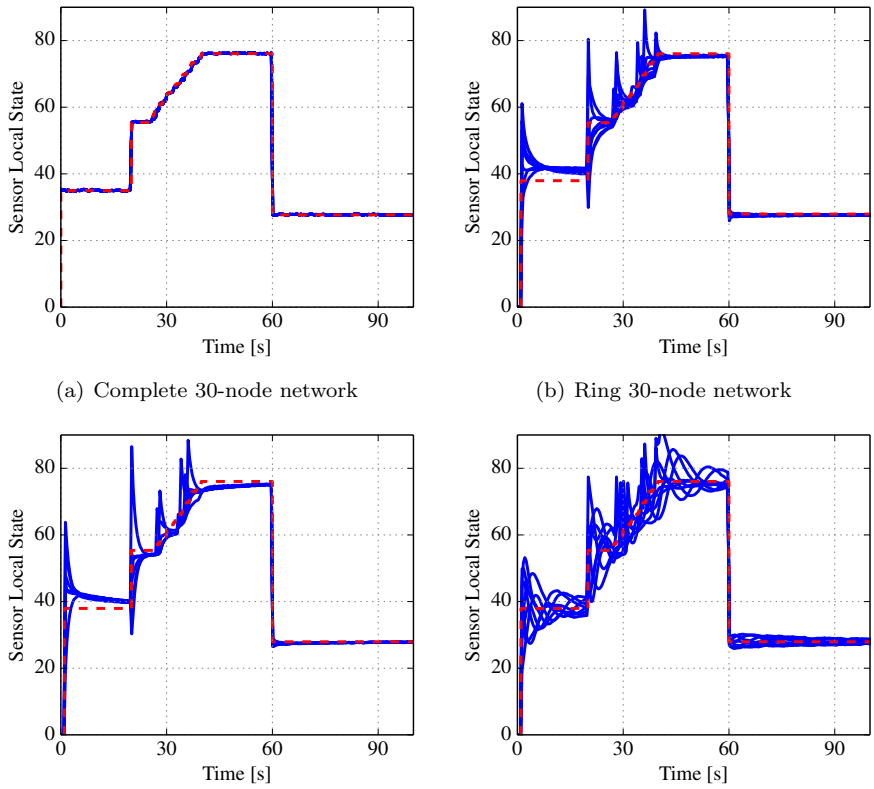
Figure A.8: *Two network topologies – (a) m clusters made of 1 cluster head and n satellite node, and (b) $|\mathcal{V}|$ aligned nodes.*

A.7 Chemical consensus in clustered and inline networks

This section describes results from a few further experiments that aimed to test the chemical consensus algorithm introduced in Chapter 4.

We compare results about experiments in ring and complete networks, already shown in Chapter 4, with those obtained in clustered and inline networks. In *clustered* networks (Figure A.8(a)), sensors formed groups of star-connected nodes. The central node of each cluster, assumed to have higher performance in terms of reception and transmission, was able to communicate with the next cluster-head node. For *inline* networks (Figure A.8(b)), we positioned nodes on a virtual line, middle nodes communicated with the two neighbors whereas the edge-nodes communicated with the only neighbor.

Figure A.9 shows the results obtained in a dynamic scenario where nodes sensed a value that changed four times during the experiment. The inline topology



(a) Complete 30-node network

(b) Ring 30-node network

(c) Cluster (4+1)x6-node net.

(d) Inline 30-node network

Figure A.9: *Simulation measurements (10 out of 30 nodes): Computed averages (blue-continuous line) and arithmetic mean (red-dashed line) over time. The results are related to simulations where sensed quantity changed four times:*

t:	*0s*	*20 s*	$\in \mathcal{U}[25s, 40s]$	*60s*
z_i:	$\in \mathcal{U}[5, 30]$	$\in \mathcal{U}[50, 60]$	$\in \mathcal{U}[75, 77]$	$\in \mathcal{U}[25, 35]$.

induced very poor convergence speed due to the oscillatory trajectory each node's state followed. The cause is the sort of reflection that the spreading of state variation experienced when arrived at the boundary of the network. We found that it is important *where* and *how many* measurement changes occur – depending on which node changed the measurement, the oscillation damping was faster or slower (but always experienced).

A.8 Chemical consensus algorithm: collision estimation

In this section, we estimate the probability of collisions that affect the described implementation of the chemical consensus model for WSNs (explained Chapter 4). The theoretically predicted rate at which collisions likely occur may constitute the input of a correcting mechanism that compensates the underestimation due to interferences among nodes. For the sake of analyzability, in this section, we make assumptions about the distributions of nodes' access instants. We remind the reader that the quality of these assumptions may be affected by the design choice concerning the inter-reaction times.

We can assume the starting time of sensors as independent from each other and being a random variables uniform in the interval $[0, T_{\max}]$. We further assume that the transmission time of each node is still a uniform random variable in the the same interval $[0, T_{\max}]$. To make the analysis simpler, we will calculate the probability of non collision P_{NC} from which the probability of collision P_C can be derived as $P_{\text{C}} = 1 - P_{\text{NC}}$. Additionally we will refer to τ as the pulse duration, under the assumption $\tau \sim \tilde{\tau} << T_{\max}$ (both pulses, representing molecules S and X, have approximatively the same time-length).

In the case of a single node, the probability of non collision is unitary since the node is the only one that can occupy the channel and multiple transmissions are not possible. In the case of two nodes, they have to share the resource and thus, once one of the two has occupied a certain period τ from t'_1 to t''_1, the other has not the whole $T_{\max}$ available but rather it should avoid to start transmitting within the period $[t'_1, t''_1]$. The vulnerability period is 2τ and thus the probability of non collision is $P_{\text{NC}}\big(|\mathcal{V}| = 2\big) = 1 - \frac{2\tau}{T_{\max}} = \frac{T_{\max} - 2\tau}{T_{\max}}$. In the case of three nodes, we also have to keep in mind that the first two nodes occupying the channel could use two periods for their transmission that are close in time but not adjacent: node ν_1 transmits in the interval $[t'_1, t''_1]$ and node ν_2 in the interval $[t'_2, t''_2]$, where $0 < t'_2 - t''_1 < \tau$. The worst case is when the first two nodes occupy two periods that are spaced by an amount slightly less than the pulse duration τ. In 3-node case, the probability of non-collision is $P_{\text{NC}}\big(|\mathcal{V}| = 3\big) = \frac{T_{\max} - 2\tau}{T_{\max}} \cdot \frac{T_{\max} - 4\tau}{T_{\max}}$. By following a similar argumentation we can write the general equation that quantify the probability of non-collision in

a system with $|\mathcal{V}|$ sensor nodes:

$$P_{\mathrm{NC}}\big(|\mathcal{V}| = M\big) = \frac{T_{\max} - 2\tau}{T_{\max}} \cdot \frac{T_{\max} - 4\tau}{T_{\max}} \cdot \ \ldots \ \cdot \frac{T_{\max} - (M-1)2\tau}{T_{\max}}$$

and thus the final probability to have a collision in a $|\mathcal{V}|$-sensor system:

$$P_{\mathrm{C}}\Big(|\mathcal{V}| = M\Big) = 1 - \prod_{i=0}^{M-1} \left(\frac{T_{\max} - 2i\tau}{T_{\max}}\right). \tag{A.17}$$

Collision probability

It is not important how many nodes the network counts but rather how the sensor nodes are connected. In a random network or in any other network where the node's connectivity changes from node to node, the collision probability of a node is dependent on the number of neighbors the node counts:

$$P_{\mathrm{C,i}}\Big(|\mathcal{N}_i| = N_i\Big) = 1 - \prod_{i=0}^{N_i-1} \left(\frac{T_{\max} - 2i\tau}{T_{\max}}\right). \tag{A.18}$$

Dependency on the number of neighbors

In a ring network, where the interference is related to the communication between adjacent nodes only, the collision probability is $P_{\mathrm{C}} = 1 - \frac{T_{\max}-2\tau}{T_{\max}}$. The collision probability, and thus the related underestimation of the average, can be reduced by increasing the minimum transmission period $T_{\max}$ or by using shorter pulses. The first approach makes the working range smaller or the convergence time bigger. The second approach increases the required bandwidth. A third possible approach is to slow down the virtual time that characterizes the chemical model, keeping τ-value fix. In this way, (*i*) the collision probability is reduced while the range of representable values does not change, and (*ii*) the number of transmissions (as well as the energy consumption) is reduced. The counterpart is a slower adaptation of a node as the dynamical system ($\mathcal{AC}$) works at a virtual time that is slower than the experienced physical time at which events actually occur.

Collision compensation

The fact that collisions may occur does not represent a problem: Even though collisions make the system open (i.e., collisions represent a non-selective dilution flow that removes molecules from the system), we can let the chemical reactor compensate this unwanted outflow. Practically, this can be implemented by adding a reaction to the chemical model described in Figure 4.12 – a reaction $r_{C,i}$ that generates lost molecules at the speed at which collisions of pulses $g(\tau)$ occur on average:

$$r_{C,i}\colon \quad \emptyset \ \xrightarrow{P_{\mathrm{C,i}}} \ \mathrm{S}_i. \tag{A.19}$$

Note that $P_{\mathrm{C,i}}$ depends (always) on quantities (information) the node knows anyway. Thus, the deployment of the additional correcting mechanism through reaction (A.19) is handy.

The use of two predefined lengths of pulses that are extremely short compared to ones used for traditional packet-based communications makes our system

robust against external interferences. Pulses that have a duration different from the prefixed τ and $\tilde{\tau}$ values are easily identifiable as external interference and thus can be filtered out without affecting the chemical reactor's state.
Another problem we face to is the eventuality to have adjacent pulses that have an overall duration that is multiple of a valid value τ and $\tilde{\tau}$ (e.g., two adjacent τ-pulses, which keep the signal high for exactly $\tilde{\tau}$ us).

A.9 Chemical consensus algorithm: frequency-division implementation

In this final section, we discuss a different implementation of the chemical consensus algorithm than the time-division one proposed in Section 4.7.1. In a frequency-division approach, the chemical dynamical system of a node ν_i continuously adapts the output rate $v_{B,i}$ in order to reflect the local estimates $c_{\mathrm{S},i}$ and, with this rate, its transmitter modulates with OOK-scheme the signal that is transmitted on a reserved narrow-band channel centered on f_i, related to that node ν_i. The channel can be assigned basing on a predefined or a random frequency allocation. An I-Q receiver filters out the out-of-band signals and brings the signal to a lower intermediate frequency. An Analog-to-Digital Converter (ADC) brings the signal in the digital world, where the spectrum of the received signal is obtained through the Fast Fourier Transform (FFT). This spectrum will contain as many (non-sporadic) components as the number $|\mathcal{N}_i|$ of neighboring nodes, and the aggregate rate $v_{\mathrm{rec},i}$ will be the sum of the rates characterizing each component.
The quality of the RF front-end and of the ADC defines the performance of the system in terms of maximum number of participating nodes and total required bandwidth. The frequency resolution of the receiver-transmitter sub-system defines the number of channels for a certain required bandwidth. However, also in the case we assume no limitations of the receiver-transmitter sub-system, we still have to take into account that the frequency resolution is also related to the minimum physical time required for the acquisition of a sufficient amount of samples and thus it is related to the maximum speed with which we can turn on-off the carrier.
Both time- and frequency-division approaches differ in how the nodes share the common resource and the way how the information about the execution of broadcast reaction r_B is encoded. From the technology point of view, the two approaches are roughly similar (both need a superheterodyne receiver in order to obtain sufficient sensibility) but differ in the complexity required to process the received signal (different methods to encode and extract the information about the number $|\mathcal{N}_i|$ of neighbors and the aggregate reception rate v_{rec}).

Bibliography

[1] N. Abramson, "The ALOHA system: another alternative for computer communications," in *Proc. of the American Federation of Information Processing Societies (AFIPS) Joint Computer Conference*, Houston (TX), USA, Nov 1970, pp. 281–285.

[2] H. Abu-Libdeh, Y. Vigfusson, K. Birman, and M. Balakrishnan, "Ajil: Distributed rate-limiting for multicast networks," Technical Report, Cornell University, 2008.

[3] A. Adamatzky, O. Holland, N. G. Rambidi, and A. Winfield, "Wet artificial brains: Towards the chemical control of robot motion by reaction-diffusion and excitable media," in *Proc. of the European Conference on Advances in Artificial Life*, Lausanne, Switzerland, Sep 1999, pp. 304–313.

[4] R. Adams, "Active queue management: A survey," in *IEEE Communications Surveys and Tutorials*, vol. 15, no. 3, Jul 2013, pp. 1425–1476.

[5] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock, "Host-to-host congestion control for TCP," in *IEEE Communications Surveys and Tutorials*, vol. 12, no. 3, 3rd quarter 2010, pp. 304–342.

[6] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the performance of TCP pacing," in *Proc. of the IEEE INFOCOM*, Tel Aviv, Israel, Mar 2000, pp. 1157–1165.

[7] M. Alizadeh, B. Atikoglu, A. Kabbani, A. Lakshmikantha, R. Pan, B. Prabhakar, and M. Seaman, "Data center transport mechanisms: Congestion control theory and IEEE standardization," in *Proc. of the Annual Allerton Conference on Communication, Control, and Computing*, Monticello (IL), USA, Sep 2008, pp. 1270–1277.

[8] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center TCP (DCTCP)," in *ACM Computer Communication Review*, vol. 40, no. 4, Oct 2010, pp. 63–74.

[9] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakara, A. Vahdat, and M. Yasuda, "Less is more: Trading a little bandwidth for ultra-low latency in the data center," in *Proc. of the USENIX Conference on Networked Systems Design and Implementation*, San Jose (CA), USA, Apr 2012, pp. 253–266.

[10] M. Allman, V. Paxson, and E. Blanton, "TCP congestion control," Draft Standard RFC 5681, Internet Engineering Task Force (IETF), Sep 2009.

[11] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," Draft Standard RFC 2581, Internet Engineering Task Force (IETF), Apr 1999.

[12] O. Alparslan, S. Arakawa, and M. Murata, "Node pacing for optical packet switching," in *Journal of Photonic Network Communications*, vol. 22, no. 2, Oct 2011, pp. 172–179.

[13] Altera Corporation, "Implementing FPGA design with the OpenCL standard," White paper WP-01173-3.0, Nov 2013.

[14] Altera Corporation, "Enabling 100G traffic management," http://www.altera.com/end-markets/wireline/applications/traffic/wil-traffic.html, ret. Jan 2014.

[15] Altera Corporation, "Enabling quality of service with customizable traffic managers," Altera White Paper WP-STXIITRFC-1.0, Sep 2005.

[16] A. Amokrane, M. F. Zhani, R. Langar, R. Boutaba, and G. Pujolle, "Greenhead: Virtual data center embedding across distributed infrastructures," in *IEEE Transactions on Cloud Computing*, vol. 1, no. 1, Jan-Jun 2013, pp. 36–49.

[17] D. F. Anderson, "Global asymptotic stability for a class of nonlinear chemical equations," in *SIAM Journal on Applied Mathematics*, vol. 68, no. 5, 2008, pp. 1464–1476.

[18] P. Antoniou and A. Pitsillides, "A bio-inspired approach for streaming applications in wireless sensor networks based on the Lotka-Volterra competition model," in *Journal of Computer Communications*, vol. 33, no. 17, Nov 2010, pp. 2039–2047.

[19] Avnet Inc., "Xilinx Spartan-6 FPGA LX9 microboard," User Guide, Rev C, Aug 2011.

[20] K. Avrachenkov, M. E. Chamie, and G. Neglia, "A local average consensus algorithm for wireless sensor networks," in *Proc. of the IEEE Conference on Distributed Computing in Sensor Systems (DCOSS)*, Barcelona, Spain, Jun 2011, pp. 1–6.

[21] T. C. Aysal, M. E. Yildiz, A. D. Sarwate, and A. Scaglione, "Broadcast gossip algorithms: Design and analysis for consensus," in *Proc. of the IEEE Decision and Control (CDC)*, Cancún, México, Dec 2008, pp. 4843–4848.

[22] T. C. Aysal, M. E. Yildiz, A. D. Sarwate, and A. Scaglione, "Broadcast gossip algorithms for consensus," in *IEEE Transactions on Signal Processing*, vol. 57, no. 7, Jul 2009, pp. 2748–2761.

[23] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," in *Proc. of the ACM SIGCOMM*, Toronto (ON), Canada, Aug 2011, pp. 242–253.

[24] H. Ballani, K. Jang, T. Karagiannis, C. Kim, D. Gunawardena, and G. O'Shea, "Chatty tenants and the cloud network sharing problem," in *Proc. of the USENIX Conference on Networked Systems Design and Implementation (NSDI)*, Lombard (IL), USA, Apr 2013, pp. 171–184.

[25] J. Banâtre, P. Fradet, and Y. Radenac, "Principles of chemical programming," in *Electronic Notes in Theoretical Computer Science, Elsevir*, vol. 124, no. 1, Mar 2005, pp. 133–147.

[26] J. Banâtre and D. Le Métayer, "Programming by multiset transformation," in *Magazine Communications of the ACM*, vol. 36, no. 1, Jan 1993, pp. 98–111.

[27] J. Banâtre and D. Le Métayer, "A new computational model and its discipline of programming," Technical Report RRN 566, Institut National de Recherche en Informatique et en Automatique (INRIA), Sep 1986.

[28] W. Banzhaf, P. Dittrich, and H. Rauhe, "Emergent computation by catalytic reactions," in *Nanotechnology*, vol. 7, no. 4, Dec 1996, pp. 307–314.

[29] A. L. Barabási and R. Albert, "Emergence of scaling in random networks," in *Science*, vol. 286, no. 5439, Oct 1999, pp. 509–512.

[30] S. Barbarossa and G. Scutari, "Bio-inspired sensor network design," in *IEEE Signal Processing Magazine*, vol. 24, no. 3, May 2007, pp. 26–35.

[31] A. Basu, A. Lin, and S. Ramanathan, "Routing using potentials: a dynamic traffic-aware routing algorithm," in *Proc. of the ACM SIGCOMM*, Karlsruhe, Germany, Aug 2003, pp. 37–48.

[32] K. V. Beeck, F. Heylen, J. Meel, and T. Goedemé, "Comparative study of model-based hardware design tools," in *Proc. of the European Conference on the Use of Modern Information and Communications Technologies (ECUMICT)*, Ghent, Belgium, Mar 2010, pp. 295–306.

[33] F. Bénézit, A. G. Dimakis, P. Thiran, and M. Vetterli, "Order-optimal consensus through randomized path averaging," in *IEEE Transactions on Information Theory*, vol. 56, no. 10, Oct 2010, pp. 5150–5167.

[34] G. Berry and G. Boudol, "Chemical abstract machine," in *Proc. of ACM SIGPLAN-SIGACT symposium on principles of programming languages (POPL)*, San Francisco (CA), USA, Jan 1990, pp. 81–94.

[35] D. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.

[36] S. P. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," in *IEEE Transactions on Information Theory*, vol. 52, no. 6, Jun 2006, pp. 2508–2530.

[37] R. Braden, "Requirements for Internet hosts – communication layers," Standard RFC 1122, Internet Engineering Task Force (IETF), Oct 1989.

[38] L. S. Brakmo and L. L. Pearson, "TCP Vegas: End to end congestion avoidance on a global Internet," in *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 13, no. 8, Oct 1995, pp. 1465–1480.

[39] G. E. Briggs and J. B. S. Haldane, "A note on the kinetics of enzyme action," in *Biochemical Journal*, vol. 19, no. 2, Mar 1925, pp. 338–339.

[40] R. Brooks, "Coherent behavior from many adaptive processes," in *Proc. of the International Conference on Simulation of Adaptive Behavior*, Brighton, UK, Aug 1994, pp. 22–29.

[41] N. Burgess, "Fast ripple-carry adders in standard-cell CMOS VLSI," in *Proc. of the IEEE Symposium on Computer Arithmetic (ARITH)*, Tübingen, Germany, Jul 2011, pp. 103–111.

[42] S. Byma, G. J. Steffan, and P. Chow, "NetThreads-10G: Software packet processing on NetFPGA-10G in a virtualized networking environment," in *Proc. of the International Conference on Field Programmable Logic and Applications*, Porto, Portugal, Aug 2013, pp. 1–1.

[43] R. H. Byrne, J. T. Feddema, and C. T. Abdallah, "Algebraic connectivity and graph robustness," Technical Report SAND2009-4494, Sandia National Laboratories Albuquerque (NM) and Livermore (CA), USA, Jul 2009.

[44] K. Cai and H. Ishii, "Average consensus on general digraphs," in *Proc. of the IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, Orlando (FL), USA, Dec 2011, pp. 1956–1961.

[45] Y. Cai, S. Y. Hanay, and T. Wolf, "A study of the impact of network traffic pacing from network and end-user perspectives," in *Proc. of the IEEE International Conference on Computer Communications and Networks (ICCCN)*, Maui, Hawaii, Aug 2011, pp. 1–6.

[46] Y. Cai, S. Y. Hanay, and T. Wolf, "Practical packet pacing in small-buffer networks," in *Proc. of the IEEE International Conference on Communications (ICC)*, Dresden, Germany, Jun 2009, pp. 1–6.

[47] Y. Cai, B. Jiang, T. Wolf, and W. Gong, "A practical on-line pacing scheme at edges of small buffer networks," in *Proc. of the IEEE INFOCOM*, San Diego (CA), USA, Mar 2010, pp. 1424–1432.

[48] Y. Cai, T. Wolf, and W. Gong, "Delaying transmissions in data communication networks to improve transport-layer performance," in *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 29, no. 5, May 2011, pp. 916–927.

[49] R. Carli, A. Chiuso, L. Schenato, and S. Zampieri, "Distributed Kalman filtering using consensus strategies," in *Proc. of the IEEE Conference on Decision and Control (CDC)*, New Orleans (LA), USA, Dec 2007, pp. 5486–5491.

[50] R. Carli, F. Fagnani, P. Frasca, and S. Zampieri, "Gossip consensus algorithms via quantized communication," in *Automatica*, vol. 46, no. 1, Jan 2010, pp. 70–80.

[51] M. Caulfield, "CDNI rate pacing," Draft Standard 00 draft-caulfield-cdni-rate-pacing, Internet Engineering Task Force (IETF), Oct 2013.

[52] C. Charalambous and S. Cui, "A biologically inspired networking model for wireless sensor networks," in *IEEE Network*, vol. 24, no. 3, Jun 2010, pp. 6–13.

[53] Q. Chen and O. W. W. Yang, "On designing self-tuning controllers for AQM routers supporting TCP flows based on pole placement," in *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 22, no. 10, Dec 2004, pp. 1965–1974.

[54] S. Cheung and C. S. Pencea, "BSFQ: Bin sort fair queuing," in *Proc. of the IEEE INFOCOM*, New York (NY), USA, Jun 2002, pp. 1640–1649.

[55] D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, "Recommendations on queue management and congestion avoidance in the Internet," Informational RFC 2309, Internet Engineering Task Force (IETF), Apr 1998.

[56] M. Claypool, M. Hartling, and R. Kinicki, "Active queue management for Web traffic," in *Proc. of the IEEE International Conference on Performance, Computing, and Communications*, Phoenix (AZ), USA, Apr 2004, pp. 531–538.

[57] D. Cohen and J. Muylaert-Filho, "Introducing a calculus for higher-order multiset programming," in *Lecture Notes in Computer Science – Coordination Languages and Models (Springer Ed.)*, vol. 1061, Apr 1996, pp. 124–141.

[58] I. Councill, L. Giles, and P. Teregowda, "Consensus propagation," in *IEEE Transactions on Information Theory*, vol. 52, no. 11, Nov 2006, pp. 4753–4766.

[59] G. Craciun and M. Feinberg, "Multiple equilibria in complex chemical reaction networks: II. The species-reaction graph," in *SIAM Journal on Applied Mathematics*, vol. 66, no. 4, Mar 2006, pp. 1321–1338.

[60] G. Craciun and M. Feinberg, "Multiple equilibria in complex chemical reaction networks: I. The injectivity property," in *SIAM Journal on Applied Mathematics*, vol. 65, no. 5, May 2005, pp. 1526–1546.

[61] K. Curtis and J. Milks, "Managing power - supply system noise for EMC compliance," White paper, Microchip Technology Inc., http://www.microchip.com/stellent/groups/designcenter_sg/documents/market_communication/en028069.pdf, ret. Apr 2014.

[62] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a Fair Queueing algorithm," in *ACM Computer Communication Review*, vol. 19, no. 4, Sep 1989, pp. 1–12.

[63] P. Denantes, F. Bénézit, P. Thiran, and M. Vetterli, "Which distributed averaging algorithm should I choose for my sensor network?" in *Proc. of the IEEE INFOCOM*, Phoenix (AZ), USA, Apr 2008, pp. 986–994.

[64] C. Di Napoli, M. Giordano, Z. Németh, and N. Tonellotto, "Using chemical reactions to model service composition," in *Proc. of the ACM International Workshop on Self-Organizing ARchitectures (SOAR)*, Washington (DC), USA, Jun 2010, pp. 43–50.

[65] E. W. Dijkstra, "Self-stabilizing systems in spite of distributed control," in *Communications of the ACM*, vol. 17, no. 11, Nov 1974, pp. 643–644.

[66] A. G. Dimakis, S. Kar, J. M. F. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," in *Proceedings of the IEEE*, vol. 98, no. 11, Nov 2010, pp. 1847–1864.

[67] A. G. Dimakis, A. D. Sarwate, and M. J. Wainwright, "Geographic gossip: Efficient averaging for sensor networks," in *IEEE Transactions on Signal Processing*, vol. 56, no. 3, Mar 2008, pp. 1205–1216.

[68] P. Dittrich, "Chemical computing," in *Lecture Notes in Computer Science – Unconventional Programming Paradigms (Springer Ed.)*, vol. 3566, 2005, pp. 19–32.

[69] P. Dittrich and W. Banzhaf, "Self-evolution in a constructive binary string system," in *Artificial Life*, vol. 4, 1998, pp. 203–220.

[70] P. Dittrich and P. S. di Fenizio, "Chemical organization theory: towards a theory of constructive dynamical systems," in *Bulletin of Mathematical Biology*, vol. 69, no. 4, 2007, pp. 1199–1231.

[71] P. Dittrich, J. Ziegler, and W. Banzhaf, "Artificial chemistries – a review," in *Artificial Life*, vol. 7, 2001, pp. 225–275.

[72] A. D. Dominguez-Garcia and C. N. Hadjicostis, "Distributed strategies for average consensus in directed graphs," in *Proc. of the IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, Orlando (FL), USA, Dec 2011, pp. 2124–2129.

[73] N. Dukkipati, G. Gibb, N. McKeown, and J. Zhu, "Building a RCP (Rate Control Protocol) test network," in *Proc. of the IEEE Annual Symposium on High-Performance Interconnects*, Stanford (CA), USA, Aug 2007, pp. 91–98.

[74] N. Dukkipati and N. McKeown, "Why flow-completion time is the right metric for congestion control," in *ACM Computer Communication Review*, vol. 36, no. 1, Jan 2006, pp. 59–62.

[75] J. Elson, "PARAPIN: A parallel port pin programming library for linux," Technical Documentation, Information Sciences Institute, University of Southern California, http://www.circlemud.org/jelson/software/parapin/docs/parapin.html, ret. Mar 2000.

[76] M. Enachescu, A. Goel, T. Roughgarden, Y. Ganjali, and N. Mckeown, "Routers with very small buffers," in *Proc. of the IEEE INFOCOM*, Barcelona, Spain, April 2006, pp. 1–11.

[77] A. Erramilli, A. Neidhardt, and I. Saniee, "Performance impacts of multi-scaling in wide area TCP/IP traffic," in *Proc. of the IEEE INFOCOM*, Tel Aviv, Israel, Mar 2000, pp. 352–359.

[78] Exar Corporation, "PowerXR EMI reduction technique utilizing spread-spectrum clock dithering," White paper ANP-37, Jul 2013.

[79] M. Feinberg, "Complex balancing in general kinetic systems," in *Archive for Rational Mechanics and Analysis*, vol. 49, no. 3, Dec 1972, pp. 187–194.

[80] M. Feinberg and F. M. Horn, "Dynamics of open chemical systems and the algebraic structure of the underlying reaction network," in *Chemical Engineering Science*, vol. 29, 1974, pp. 775–787.

[81] W. Feng, K. G. Shin, D. D. Kandlur, and D. Saha, "The BLUE active queue management algorithms," in *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, Aug 2002, pp. 513–528.

[82] P. Ferreira, P. Ribeiroa, A. Antunesa, and F. M. Dias, "A high bit resolution FPGA implementation of a FNN with a new algorithm for the activation function," in *Neurocomputing*, vol. 71, no. 1-3, Dec 2007, pp. 71–77.

[83] B. H. Fletcher, "FPGA embedded processors: Revealing true system performance," Xilinx White Paper ETP-367, 2005.

[84] S. Floyd, "Recommendations on using the gentle variant of RED," http://www.icir.org/floyd/red/gentle.html, ret. Jan 2014.

[85] S. Floyd and T. Henderson, "The NewReno modification to TCP's fast recovery algorithm," Experimental RFC 2582, Internet Engineering Task Force (IETF), Apr 1999.

[86] S. Floyd, T. Henderson, and A. Gurtov, "The NewReno modification to TCP's fast recovery algorithm," Standard RFC 3782, Internet Engineering Task Force (IETF), Apr 2004.

[87] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," in *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, Aug 1993, pp. 397–413.

[88] S. Floyd, J. Madhavi, M. Mathis, and M. Podolsky, "An extension to the Selective Acknowledgement (SACK) option for TCP," Standard RFC 2883, Internet Engineering Task Force (IETF), Jul 2000.

[89] S. Floyd, K. K. Ramakrishnan, and D. Black, "The addition of explicit congestion notification (ECN) to IP," Standard RFC 3168, Internet Engineering Task Force (IETF), 2001.

[90] W. Fontana, "Algorithmic Chemistry," in *Proc. of the Workshop on Artificial Life (ALIFE)*, vol. 5, Redwood City (CA), USA, 1992, pp. 159–210.

[91] W. Fontana and L. Buss, "The arrival of the fittest: Toward a theory of biological organization," in *Bulletin of Mathematical Biology*, vol. 56, no. 1, 1994, pp. 1–64.

[92] M. Franceschelli, A. Giua, and C. Seatzu, "Distributed averaging in sensor networks based on broadcast gossip algorithms," in *IEEE Sensors Journal*, vol. 11, no. 3, Mar 2011, pp. 808–817.

[93] M. Gerla and L. Kleinrock, "Flow control: a comparative survey," in *IEEE/ACM Transactions on Communications*, vol. 28, no. 4, Apr 1980, pp. 553–574.

[94] M. A. Gibson and J. Bruck, "Efficient exact stochastic simulation of chemical systems with many species and many channels," in *Journal Physical Chemistry A*, vol. 104, no. 9, Feb 2000, pp. 1876–1889.

[95] D. T. Gillespie, "The chemical Langevin equation," in *Journal of Chemical Physics*, vol. 113, no. 1, Apr 2000, pp. 297–306.

[96] D. T. Gillespie, "Stochastic simulation of chemical kinetics," in *Annual Review of Physical Chemistry*, vol. 58, Apr 2007, pp. 35–55.

[97] D. T. Gillespie, "A general method for numerically simulating the stochastic time evolution of coupled chemical reactions," in *Journal Computational Physics*, vol. 22, no. 4, Dec 1976, pp. 403–434.

[98] D. T. Gillespie, "Exact stochastic simulation of coupled chemical reactions," in *Journal Physical Chemistry*, vol. 81, no. 25, Dec 1977, pp. 2340–2361.

[99] D. T. Gillespie, "A rigorous derivation of the chemical master equation," in *Physica A*, vol. 188, no. 1-3, Sep 1992, pp. 404–425.

[100] M. Goldenbaum, H. Boche, and S. Stańczak, "Nomographic gossiping for f-consensus," in *Proc. of the International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, Paderborn, Germany, May 2012, pp. 130–137.

[101] S. J. Golestani, "A self-clocked fair queueing scheme for broadband applications," in *Proc. of the IEEE INFOCOM*, Toronto (ON), Canada, Jun 1994, pp. 636–646.

[102] F. Guillemin and J. Boyer, "Analysis of the M/M/1 queue with processor sharing via spectral theory," in *Queueing Systems*, vol. 39, no. 4, Dec 2001, pp. 377–397.

[103] C. Guo, "SRR: An O(1) time-complexity packet scheduler for flows in multiservice packet networks," in *IEEE/ACM Transactions on Networking*, vol. 12, no. 6, Dec 2004, pp. 1144–1155.

[104] V. Gupta, B. Hassibi, and R. M. Murray, "On sensor fusion in the presence of packet-dropping communication channels," in *Proc. of the IEEE Conference on Decision and Control, and European Control Conference (CDC–ECC)*, Seville, Spain, Dec 2005, pp. 3547–3552.

[105] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," in *ACM SIGOPS Operating Systems Review - Research and developments in the Linux kernel*, vol. 42, no. 5, Jul 2008, pp. 64–74.

[106] S. Y. Hanay, A. Dwaraki, and T. Wolf, "High-performance implementation of in-network traffic pacing," in *Proc. of the IEEE International Conference on High Performance Switching and Routing (HPSR)*, Cartagena, Spain, Jul 2011, pp. 9–15.

[107] G. Hasegawa and M. Murata, "TCP symbiosis: Congestion control mechanisms of TCP based on Lotka-Volterra competition model," in *Proc. of the ACM Workshop on Interdisciplinary Systems Approach in Performance Evaluation and Design of Computer and Communications Sytems (Interperf)*, Pisa, Italy, Oct 2006, pp. 1–10.

[108] Y. Hatano and M. Mesbahi, "Agreement over random networks," in *IEEE Transactions On Automatic Control*, vol. 50, no. 11, Nov 2005, pp. 1867–1872.

[109] S. Heithecker and R. Ernst, "Traffic shaping for an FPGA based SDRAM controller with complex QoS requirements," in *Proc. of the ACM/IEEE Annual Design Automation Conference (DAC)*, Anaheim (CA), USA, Jun 2005, pp. 575–578.

[110] C. V. Hollot, V. Misra, D. Towsley, and W. Gong, "A control theoretic analysis of RED," in *Proc. of the IEEE INFOCOM*, Anchorage (AK), USA, Apr 2001, pp. 1510–1519.

[111] C. V. Hollot, V. Misra, D. Towsley, and W. Gong, "On designing improved controllers for AQM routers supporting TCP flows," in *Proc. of the IEEE INFOCOM*, Anchorage (AK), USA, Apr 2001, pp. 1726–1734.

[112] F. Horn, "On a connexion between stability and graphs in chemical kinetics. I. Stability and the reaction diagram," in *Proceedings of the Royal Society of London – Series A, Mathematical and Physical Sciences*, vol. 334, no. 1598, 1973, pp. 299–312.

[113] F. Horn, "On a connexion between stability and graphs in chemical kinetics. II. Stability and the complex graph," in *Proceedings of the Royal Society of London – Series A, Mathematical and Physical Sciences*, vol. 334, no. 1598, 1973, pp. 313–330.

[114] F. Horn, "Stability and complex balancing in mass-action systems with three short complexes," in *Proceedings of the Royal Society of London – Series A, Mathematical and Physical Sciences*, vol. 334, no. 1598, 1973, pp. 331–342.

[115] F. Horn and R. Jackson, "General mass action kinetics," in *Archive for Rational Mechanics and Analysis*, vol. 47, no. 2, 1972, pp. 81–116.

[116] IEEE Computer Society, "IEEE standard for floating-point arithmetic," IEEE Std 754 - 2008 (revision IEEE Std 754 - 1985), 2008.

[117] IEEE Computer Society, "IEEE standard for Verilog hardware description language," IEEE Standard 1364-2005, Apr 2006.

[118] IEEE Computer Society, "IEEE standard VHDL language reference manual," IEEE Standard 1076-2008, Jan 2009.

[119] B. Ingalls, "A frequency domain approach to sensitivity analysis of biochemical networks," in *Journal Physical Chemistry B*, vol. 108, no. 3, Jan 2004, pp. 1143–1152.

[120] International Telecommunication Union, "Traffic control and congestion control in B-ISDN," ITU-T Recommendation I.371 – Series I: Integrated Services Digital Network, Mar 2004.

[121] V. Jacobson, "Congestion avoidance and control," in *ACM Computer Communication Review*, vol. 18, no. 4, Aug 1988, pp. 314–329.

[122] T. Jahnke and W. Huisinga, "Solving the chemical master equation for monomolecular reaction systems analytically," in *Journal of Mathematical Biology*, vol. 54, no. 1, Jan 2007, pp. 1–26.

[123] V. Jeyakumar, M. Alizadeh, D. Mazières, B. Prabhakar, C. Kim, and A. Greenberg, "EyeQ: practical network performance isolation at the edge," in *Proc. of the USENIX Conference on Networked Systems Design and Implementation*, Lombard (IL), USA, Apr 2013, pp. 297–312.

[124] A. Kamra, S. Kapila, V. Khurana, V. Yadav, R. Shorey, H. Saran, and S. Juneja, "SFED: A rate control based active queue management discipline," Technical Report 00A018, IBM India Research Laboratory, 2000.

[125] P. Karn, C. Bormann, G. Fairhurst, D. Grossmana, R. Ludwig, J. Mahdavi, G. Montenegro, J. Touch, and L. Wood, "Advice for Internet subnetwork designers," Best Current Practice RFC 3819, Internet Engineering Task Force (IETF), Jul 2004.

[126] J. F. Keane, C. Bradley, and C. Ebeling, "A compiled accelerator for biological cell signaling simulations," in *Proc. of ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA)*, Monterey (CA), USA, Feb 2004, pp. 233–241.

[127] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proc. of the Annual IEEE Symposium on Foundations of Computer Science*, Cambridge (MA), USA, Oct 2003, pp. 482–491.

[128] D. G. Kendall, "Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded Markov chain," in *The Annals of Mathematical Statistics*, vol. 24, no. 3, Nov 1953, pp. 338–354.

[129] J. Kenyeres, M. Kenyeres, M. Rupp, and P. Farcaš, "WSN implementation of the average consensus algorithm," in *Proc. of the European Sustainable Wireless Technologies (European Wireless)*, Vienna, Austria, Apr 2011, pp. 139–146.

[130] S. Keshav, "A control-theoretic approach to flow control," in *Proc. of the ACM SIGCOMM*, Zurich, Switzerland, Sep 1991, pp. 3–15.

[131] D. Klein, J. Hespanha, and U. Madhow, "A reaction-diffusion model for epidemic routing in sparsely connected MANETs," in *Proc. of the IEEE INFOCOM*, San Diego (CA), USA, Mar 2010, pp. 1–9.

[132] T. Kobori, T. Maruyama, and T. Hoshino, "A Cellular Automata system with FPGA," in *Proc. of the IEEE Annual Symposium on Field-Programmable Custom Computing Machines (FCCM)*, Rohnert Park (CA), USA, Apr 2001, pp. 120–129.

[133] E. Kohler, M. Handley, and S. Floyd, "Datagram Congestion Control Protocol (DCCP)," Standard RFC 4340, Internet Engineering Task Force (IETF), Mar 2006.

[134] J. R. Koza, "Spontaneous emergence of self-replicating and evolutionarily self-improving computer programs," in *Artificial Life III – SFI Studies in the Sciences of Complexity*, vol. VII, 1994, pp. 225–262.

[135] P. Kreyssig and P. Dittrich, "Emergent control," in *Organic Computing – A Paradigm Shift for Complex Systems, Ser. Autonomic Systems, Springer Ed.*, vol. 1, 2011, pp. 67–78.

[136] M. Kriegleder, R. Oung, and R. D'Andrea, "Asynchronous implementation of a distributed average consensus algorithm," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, Nov 2013, pp. 1836–1841.

[137] M. Kriegleder, R. Oung, and R. D'Andrea, "Distributed altitude and attitude estimation from multiple distance measurements," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Portugal, Oct 2012, pp. 3626–3632.

[138] M. Labrecque, "Overlay architectures for FPGA-based software packet processing," Ph.D. Dissertation, Graduate Department of Electrical and Computer Engineering University of Toronto, Toronto (ON), Canada, 2011.

[139] M. Labrecque, G. J. Steffan, G. Salmon, M. Ghobadi, and Y. Ganjali, "NetThreads routing edition: Programming NetFPGA with threaded software," in *Proc. of the NetFPGA Developers Workshop*, Stanford University (CA), USA, Aug 2010, pp. 1–6.

[140] Lattice Semiconductor Corp., "Lattice announces low cost programmable SPI-4.2 solution," http://ir.latticesemi.com/phoenix.zhtml?c=117422&p=irol-newsArticle&ID=1472678&highlight, ret. Jan 2014.

[141] D. Le Métayer, "Higher-order multiset programming," in *Proc. of the AMS DIMACS Workshop on Specifications of Parallel Algorithms, DIMACS series in Discrete Mathematics*, Princeton (NJ), USA, May 1994, pp. 179–200.

[142] D. U. Lee, W. Luk, J. D. Villasenor, G. Zhang, and P. H. W. Leong, "A hardware Gaussian noise generator using the Wallace method," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 8, Aug 2005, pp. 911–920.

[143] D. U. Lee, J. D. Villasenor, W. Luk, and P. H. W. Leong, "A hardware Gaussian noise generator using the Box-Muller method and its error analysis," in *IEEE Transactions on Computers*, vol. 55, no. 6, Jun 2006, pp. 569–671.

[144] P. D. Leenheer and D. Angeli, "Monotonicity and convergence in chemical reaction networks," in *Proc. of the IEEE Conference on Decision and Control, and European Control Conference (CDC–ECC)*, Seville, Spain, Dec 2005, pp. 2362–2367.

[145] W. Li and H. Dai, "Cluster-based fast distributed consensus," in *Proc. of the IEEE Acoustics, Speech and Signal Processing (ICASSP)*, Honolulu (HI), USA, Apr 2007, pp. 185–188.

[146] H. Ling, "High-speed binary adder," in *IBM Journal of Research and Development*, vol. 25, no. 3, Mar 1981, pp. 156–166.

[147] J. W. Lockwood, N. McKeown, G. Watson, G. Gibb, P. Hartke, J. Naous, R. Raghuraman, and J. Luo, "NetFPGA – an open platform for gigabit-rate network switching and routing," in *Proc. of the IEEE International Conference on Microelectronic Systems Education (MSE)*, San Diego (CA), USA, Jun 2007, pp. 160–161.

[148] L. Lok, "The need for speed in stochastic simulation," in *Nature Biotechnology*, vol. 22, 2004, pp. 964–965.

[149] S. H. Low, F. Paganini, and J. C. Doyle, "Internet congestion control," in *IEEE Control Systems Magazine*, Feb 2002, pp. 28–43.

[150] P. Lysaght, J. Stockwood, J. Law, and D. Girma, "Artificial neural network implementation on a fine-grained FPGA," in *Lecture Notes in Computer Science – Field-Programmable Logic Architectures, Synthesis and Applications (Springer Ed.)*, vol. 849, 1994, pp. 421–431.

[151] N. Malangadan and G. Raina, "Rate based feedback: some experimental evaluation with NetFPGA," in *Proc. of the IEEE International Conference on Communication (ICC)*, Kyoto, Japan, Jun 2011, pp. 1–6.

[152] F. A. Malekzadeh, R. Mahmoudi, and A. H. van Roermund, "Dithering," in *Analog Circuits and Signal Processing – Analog Dithering Techniques for Wireless Transmitters*, vol. 3, 2013, pp. 9–23.

[153] S. Mascolo, "Classical control theory for congestion avoidance in high-speed Internet," in *Proc. of the IEEE Conference on Decision and Control (CDC)*, vol. 3, Phoenix (AZ), USA, Dec 1999, pp. 2709–2714.

[154] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth estimation for enhanced transport over wireless links," in *Proc.of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, Rome, Italy, Jul 2001, pp. 287–297.

[155] M. Mathis and J. Mahdavi, "Forward acknowledgement: refining TCP congestion control," in *ACM Computer Communication Review*, vol. 26, no. 4, Oct 1996, pp. 281–291.

[156] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanov, "TCP selective acknowledgment options," Standard RFC 2018, Internet Engineering Task Force (IETF), Oct 1996.

[157] N. Matsumaru and P. Dittrich, "Organization-oriented chemical programming for the organic design of distributed computing systems," in *Proc. of the International Conference on Bio-Inspired mOdels of NEtwork, Information and Computing systems (BIONETICS)*, Cavalese, Italy, Dec 2006, pp. 1–6.

[158] P. E. McKenney, "Stochastic fairness queuing," in *Proc. of the IEEE INFOCOM*, San Francisco (CA), USA, Jun 1990, pp. 733–740.

[159] N. McKeown, G. Parulkar, T. Anderson, L. Peterson, H. Balakrishnan, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," ONF White Paper, Mar 2008.

[160] D. A. McQuerry, "Stochastic approach to chemical kinetics," in *Journal of Applied Probability*, vol. 4, 1967, pp. 413–478.

[161] S. G. Merchant and G. D. Peterson, "Evolvable block-based neural network design for applications in dynamic environments," in *Journal VLSI Design - Special issue on selected papers from the midwest symposium on circuits and systems*, Jan 2010, pp. 1–24.

[162] M. Mesbahi, "On state-dependent dynamic graphs and their controllability properties," in *IEEE Transactions on Automatic Control*, vol. 50, no. 3, Mar 2005, pp. 387–392.

[163] T. Meyer, "On chemical and self-healing networking protocols," Ph.D. Dissertation, Faculty of Computer Science, University of Basel, Switzerland, 2011.

[164] T. Meyer and C. F. Tschudin, "Chemical networking protocols," in *Proc. of the ACM Workshop on Hot Topics in Networks (HotNets)*, New York (NY), USA, Oct 2009, pp. 1–6.

[165] T. Meyer and C. F. Tschudin, "A theory of packet flows based on law-of-mass-action scheduling," in *Proc. of the IEEE International Symposium on Reliable Distributed Systems (SRDS)*, Irvine (CA), USA, Oct 2012, pp. 341–351.

[166] T. Meyer, L. Yamamoto, and C. Tschudin, "An artificial chemistry for networking," in *Bio-Inspired Computing and Communication*, vol. 5151, 2008, pp. 45–57.

[167] L. Michaelis and M. L. Menten, "The kinetics of invertase action (die Kinetik der Invertinwirkung)," in *Biochemische Zeitschrift*, vol. 49, Jan 1913, pp. 333–369.

[168] M. Monti, "A signal processing approach to the analysis of chemical networking protocols," M.Sc. Thesis, Department of Information Engineering, University of Pisa, Italy, Jul 2010.

[169] Y. Moon, D. Jeong, and G. Kim, "Clock dithering for electromagnetic compliance using spread spectrum phase modulation," in *Proc. of the IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco (CA), USA, Feb 1999, pp. 186–187.

[170] B. Moyer, "Packet subsystem on a chip," in *Xcell Journal*, vol. 56, First Quarter 2006, pp. 10–13.

[171] S. Murtaza, A. G. Hoekstra, and P. M. A. Sloot, "Performance modeling of 2D Cellular Automata on FPGA," in *Proc. of the International Conference on Field Programmable Logic and Applications (FPL)*, Amsterdam, The Netherlands, Aug 2007, pp. 4–78.

[172] S. Murtaza, A. G. Hoekstra, and P. M. A. Sloot, "Cellular automata simulations on a FPGA cluster," in *International Journal of High Performance Computing Applications*, vol. 25, no. 2, May 2011, pp. 193–204.

[173] J. Nagle, "On packet switches with infinite storage," Informational RFC 970, Internet Engineering Task Force (IETF), Dec 1985.

[174] National Instruments corporation, "FPGA fundamentals," Technical Report Nr.6983, May 2012.

[175] B. Nazer, A. G. Dimakis, and M. Gastpar, "Neighborhood gossip: Concurrent averaging through local interference," in *Proc. of the IEEE Acoustics, Speech and Signal Processing (ICASSP)*, Taipei, Taiwan, Apr 2009, pp. 3657–3660.

[176] B. Nazer, A. G. Dimakis, and M. Gastpar, "Local interference can accelerate gossip algorithms," in *IEEE Journal on Selected Topics in Signal Processing*, vol. 5, no. 4, Aug 2011, pp. 876–887.

[177] NetFPGA Working Group, "The NetFPGA project," http://netfpga.org, ret. Jan 2014.

[178] K. Nichols and V. Jacobson, "Controlling queue delay," in *Magazine Communications of the ACM*, vol. 55, no. 7, May 2012, pp. 42–50.

[179] M. Nokleby, W. U. Bajwa, R. Calderbank, and B. Aazhang, "Gossiping in groups: Distributed averaging over the wireless medium," in *Proc. of the Annual Allerton Conference on Communication, Control, and Computing*, Monticello (IL), USA, Sep 2011, pp. 1242–1249.

[180] R. Olfati-Saber, J. A. Fax, and R. Murray, "Consensus and cooperation in networked multi-agent systems," in *Proceedings of the IEEE*, vol. 95, no. 1, Jan 2007, pp. 215–233.

[181] R. Olfati-Saber, "Ultrafast consensus in small-world networks," in *Proc. of the American Control Conference*, vol. 4, Portland (OR), USA, Jun 2005, pp. 2371–2378.

[182] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: algorithms and theory," in *IEEE Transactions on Automatic Control*, vol. 51, no. 3, Mar 2006, pp. 401–420.

[183] R. Olfati-Saber and R. M. Murray, "Graph rigidity and distributed formation stabilization of multivehicle systems," in *Proc. of the IEEE Conference on Decision and Control (CDC)*, Las Vegas (NE), USA, Dec 2002, pp. 2965–2971.

[184] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," in *IEEE Transactions on Automatic Control*, vol. 49, no. 9, Sep 2004, pp. 1520–1533.

[185] ONF Project Group, "Open Networking Foundation (ONF)," https://www.opennetworking.org/, ret. Feb 2014.

[186] L. Ong and J. Yoakum, "An introduction to the Stream Control Transmission Protocol (SCTP)," Informational RFC 3286, Internet Engineering Task Force (IETF), May 2002.

[187] OpenStack Project Group, "OpenStack," http://www.openstack.org/, ret. Jan 2014.

[188] V. N. Padmanabhan and R. H. Katz, "TCP fast start: A technique for speeding up web transfers," in *Proc. of the IEEE Globecom Internet Mini-Conference*, Sydney, Australia, Oct 1998, pp. 41–46.

[189] R. Pan, P. Natarajan, C. Piglione, M. S. Prabhu, V. Subramanian, F. Baker, and B. V. Steeg, "PIE: A lightweight control scheme to address the bufferbloat problem," Draft Standard 00 draft-pan-aqm-pie, Internet Engineering Task Force (IETF), Dec 2012.

[190] R. Pan, B. Prabhakar, and K. Psounis, "CHOKe - a stateless active queue management scheme for approximating fair bandwidth allocation," in *Proc. of the IEEE INFOCOM*, Tel Aviv, Israel, Mar 2000, pp. 942–951.

[191] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," in *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, Jun 1993, pp. 344–357.

[192] E. Park, H. Lim, K. Park, and C. Choi, "Analysis and design of the virtual rate control algorithm for stabilizing queues in TCP networks," in *Computer Networks*, vol. 44, no. 1, Jan 2004, pp. 17–41.

[193] G. Paŭn, "Computing with membranes," in *Journal of Computer and System Sciences*, vol. 61, no. 1, 2000, pp. 108–143.

[194] V. Phirke, M. Claypool, and R. Kinicki, "RED-Worcester - traffic sensitive active queue management," in *Proc. of the IEEE International Conference on Network Protocols*, Paris, France, Nov 2002, pp. 194–195.

[195] D. C. Plummer, "An Ethernet Address Resolution Protocol – or – converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware," Standard RFC 826, Internet Engineering Task Force (IETF), Nov 1982.

[196] N. Possley, "Traffic management in Xilinx FPGAs," Xilinx White Paper WP244 (v1.0), Apr 2006.

[197] E. Post, "Formal reductions of the combinatorial decision problem," in *American Journal of Mathematics*, vol. 65, 1943, pp. 197–215.

[198] J. Postel, "User Datagram Protocol," Internet Standard RFC 768, Internet Engineering Task Force (IETF), Aug 1980.

[199] J. Postel, "Transmission Control Protocol – DARPA Internet program protocol specification," Standard RFC 793, Internet Engineering Task Force (IETF), Sep 1981.

[200] B. Raghavan, K. V. Vishwanath, S. Ramabhadran, K. Yocum, and A. C. Snoeren, "Cloud-control with distributed rate limiting," in *Proc. of the ACM SIGCOMM*, Kyoto, Japan, Aug 2007, pp. 337–348.

[201] C. V. Rao, H. M. Sauro, and A. P. Arkin, "Putting the "control" in metabolic control analysis," in *Proc. of the Dynamics and Control of Process Systems (DYCOPS)*, vol. 2, Cambridge (MA), USA, Jul 2004, pp. 1001–1006.

[202] K. Ravindran, N. Satish, Y. Jin, and K. Keutzer, "An FPGA-based soft multiprocessor system for IPv4 packet forwarding," in *Proc. of the IEEE International Conference on Field Programmable Logic and Applications*, Tampere, Finland, Aug 2005, pp. 487–492.

[203] C. Reder, "Metabolic control theory: A structural approach," in *Journal Theoretical Biology*, vol. 135, no. 2, 1988, pp. 175–201.

[204] H. Rodrigues, J. R. Santos, Y. Turner, P. Soares, and D. Guedes, "Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks," in *Proc. of the USENIX Workshop on I/O Virtualization (WIOV)*, Portland (OR), USA, Jun 2011, pp. 1–8.

[205] S. Ryu, C. Rump, and C. Qiao, "Advances in active queue management (AQM) based TCP congestion control," in *Telecommunication Systems - Modeling, Analysis, Design and Management*, vol. 25, no. 3-4, Mar 2004, pp. 317–351.

[206] M. Saerens, F. Fouss, L. Yen, and P. Dupont, "The principal components analysis of a graph, and its relationships to spectral clustering," in *Lecture Notes in Computer Science – Machine Learning (Springer Ed.)*, vol. 3201, 2004, pp. 371–383.

[207] V. Salapura and V. Hamann, "Implementing fuzzy control systems using VHDL and statecharts," in *Proc. of the European Design Automation Conference and Exhibition (EURO-VHDL and EURO-DAC)*, Geneva, Switzerland, Sep 1996, pp. 53–58.

[208] V. Saligrama, M. Alanyali, and O. Savas, "Distributed detection in sensor networks with packet loss and finite capacity links," in *IEEE Transactions on Signal Processing*, vol. 54, no. 11, Nov 2006, pp. 4118–4132.

[209] L. Salwinski and D. Eisenberg, "In silico simulation of biological network dynamics," in *Nature Biotechnology*, vol. 22, Jul 2004, pp. 1017–1019.

[210] H. M. Sauro and B. Ingalls, "Conservation analysis in biochemical networks: computational issues for software writers," in *Biophysical Chemistry*, vol. 109, no. 1, Apr 2004, pp. 1–15.

[211] G. Schelle and D. Grunwald, "CUSP: A modular framework for high speed network applications on FPGAs," in *Proc. of the ACM/SIGDA international symposium on Field-Programmable Gate Arrays (FPGA)*, Monterey (CA), USA., Feb 2005, pp. 246–257.

[212] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz, "Ant-based load balancing in telecommunications networks," in *Journal of Adaptive Behavior*, vol. 5, no. 2, Fall 1996, pp. 169–207.

[213] D. Shah, *Gossip Algorithms*. Foundations and Trends in Networking (NOW Ed.), 2009, vol. 3, no. 1.

[214] W. Shaochuan and M. G. Rabbat, "Broadcast gossip algorithms for consensus on strongly connected digraphs," in *IEEE Transactions on Signal Processing*, vol. 61, no. 16, Aug 2013, pp. 3959–3971.

[215] G. Shinar, U. Alon, and M. Feinberg, "Sensitivity and robustness in chemical reaction networks," in *SIAM Journal Applied Mathematics*, vol. 69, no. 4, Jan 2009, pp. 977–998.

[216] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round robin," in *IEEE/ACM Transactions on Networking*, vol. 4, no. 3, Jun 1996, pp. 375–385.

[217] O. Simeone and U. Spagnolini, "Distributed time synchronization in wireless sensor networks with coupled discrete-time oscillators," in *EURASIP Journal on Wireless Communications and Networking*, Mar 2007, pp. 1–13.

[218] O. Simeone, U. Spagnolini, G. Scutari, and Y. Bar-Ness, "Physical-layer distributed synchronization in wireless networks and applications," in *Physical Communication*, vol. 1, no. 1, Mar 2008, pp. 67–83.

[219] V. Sivaraman, H. Elgindy, D. Moreland, and D. Ostry, "Packet pacing in short buffer optical packet switched networks," in *Proc. of the IEEE INFOCOM*, Barcelona, Spain, Apr 2006, pp. 1–11.

[220] E. D. Sontag, "Structure and stability of certain chemical networks and applications to the kinetic proofreading model of T-cell receptor signal transduction," in *IEEE Transactions on Automatic Control*, vol. 46, no. 7, Jul 2001, pp. 1028–1047.

[221] S. Stepney, "Nonclassical computation – a dynamical systems perspective," in *Handbook of Natural Computing, Springer*, vol. 4, 2012, pp. 1979–2025.

[222] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery algorithms," Standard RFC 2001, Internet Engineering Task Force (IETF), Jan 1997.

[223] Y. E. Sung, C. Lund, M. Lyn, S. G. Rao, and S. Sen, "Modeling and understanding end-to-end class of service policies in operational networks," in *ACM Computer Communication Review*, vol. 39, no. 4, Oct 2009, pp. 219–230.

[224] S. Suri, G. Varghese, and G. Chandranmenon, "Leap forward virtual clock: A new fair queuing scheme with guaranteed delays and throughput fairness," in *Proc. of the IEEE INFOCOM*, Kobe, Japan, Apr 1997, pp. 557–565.

[225] A. S. Tanenbaum, *Computer Networks*. Prentice Hall, 4th Ed., 2003.

[226] M. Teich, *A documentary history of biochemistry, 1770-1940*. Fairleigh Dickinson University Press, 1992.

[227] Texas Instruments Inc., "4-bit magnitude comparators," Datasheet SDLS123, Mar 1974.

[228] E. Thereska, H. Ballani, G. O'Shea, T. Karagiannis, A. Rowstron, T. Talpey, R. Black, and T. Zhu, "IOFlow: A software-defined storage architecture," in *Proc. of the ACM Symposium on Operating Systems Principles (SOSP)*, Farmington (PA), USA, Nov 2013, pp. 182–196.

[229] J. G. Tong, I. D. L. Anderson, and M. A. S. Khalid, "Soft-core processors for embedded systems," in *Proc. of the International Conference on Microelectronics (ICM)*, Dhahran, Saudi Arabia, Dec 2006, pp. 170–173.

[230] C. F. Tschudin, "Fraglets – A metabolistic execution model for communication protocols," in *Proc. of the Annual Symposium on Autonomous Intelligent Networks and Systems (AINS)*, Menlo Park (CA), USA, Jul 2003, pp. 1–6.

[231] C. F. Tschudin and T. Meyer, "Programming by equilibria," in *15. Kolloquium Programmiersprachen und Grundlagen der Programmierung (KPS)*, Oct 2009, pp. 37–46.

[232] J. N. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," in *IEEE Transactions Automatic Control*, vol. 31, no. 8, Sep 1986., pp. 803–812.

[233] A. Turing, "The chemical basis of morphogenesis," in *Philosophical Transactions of the Royal Society of London – Series B, Biological Sciences*, vol. 237, no. 641, Aug 1952, pp. 37–72.

[234] J. S. Turner, "New directions in communications (or which way to the information age?)," in *IEEE Communications Magazine*, vol. 24, no. 10, Oct 1986, pp. 8–15.

[235] M. Ullaha and O. Wolkenhauer, "Investigating the two-moment characterisation of subcellular biochemical networks," in *Journal of Theoretical Biology*, vol. 260, no. 3, Oct 2009, pp. 340–352.

[236] R. van Haalen and R. Malhotra, "Improving TCP performance with bufferless token bucket policing: A TCP friendly policer," in *Proc. of the IEEE Workshop on Local and Metropolitan Area Networks (LANMAN)*, Princeton (NJ), USA, Jun 2007, pp. 72–77.

[237] N. G. van Kampen, *Stochastic Processes in Physics and Chemistry*. Elsevier Ed., 3rd Ser., 2007.

[238] D. D. Van Slyke and G. E. Cullen, "The mode of action of urease and of enzymes in general," in *Journal of Biological Chemistry*, vol. 19, no. 2, Oct 1916, pp. 141–180.

[239] P. B. Vanguri, "An FPGA based implementation of the exact stochastic simulation algorithm," M.Sc. Thesis, University of Tennessee, Dec 2010.

[240] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," in *Proc. of the SIMUTOOLS*, Marseille, France, Mar 2008, pp. 1–10.

[241] P. Waage, C. M. Gulberg, and H. I. Abrash (Translator), "Studies concerning affinity," in *Journal of Chemical Education*, vol. 12, no. 63, 1986, pp. 1044–1047.

[242] X. Wang and D. Eun, "Local and global stability of TCP-NewReno/RED with many flows," in *Journal of Computer Communications*, vol. 30, no. 5, Mar 2007, pp. 1091–1105.

[243] Z. Wang and J. Crowcroft, "Eliminating periodic packet losses in the 4.3-Tahoe BSD TCP congestion control algorithm," in *ACM Computer Communication Review*, vol. 22, no. 2, Apr 1992, pp. 9–16.

[244] D. J. Watts and S. H. Strogatz, "Collective dynamics of "small-world" networks," in *Nature*, vol. 393, Jun 1998, pp. 440–442.

[245] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson, "Self-similarity through high-variability: Statistical analysis of Ethernet LAN traffic at the source level," in *Proc. of the ACM SIGCOMM*, Cambridge (MA), USA, Aug 1995, pp. 100–113.

[246] K. Winstein and H. Balakrishnan, "TCP ex machina: computer-generated congestion control," in *ACM Computer Communication Review*, vol. 43, no. 4, Oct 2013, pp. 123–134.

[247] O. Wolkenhauer, M. Ullah, W. Kolch, and K. Cho, "Modeling and simulation of intracellular dynamics: Choosing an appropriate framework," in *IEEE Transactions on Nanobioscience*, vol. 3, no. 3, Sep 2004, pp. 200–207.

[248] Z. Wu, Z. Guan, Z. Wu, and T. Li, "Consensus based formation control and trajectory tracing of multi-agent robot systems," in *Journal of Intelligent and Robotic Systems*, vol. 48, no. 3, Mar 2007, pp. 397–410.

[249] F. Xiao and L. Wang, "Asynchronous consensus in continuous-time multi-agent systems with switching topology and time-varying delays," in *IEEE Transactions on Automatic Control*, vol. 53, no. 8, Sep 2008, pp. 1804–1816.

[250] L. Xiaoand and S. Boyd, "Fast linear iterations for distributed averaging," in *Systems and Control Letters*, vol. 53, Feb 2003, pp. 65–78.

[251] Xilinx Inc., "LogiCORE IP floating-point operator v6.0," Application Note DS816, Jan 2012.

[252] Xilinx Inc., "LogiCORE IP adder/subtracter v11.0," Application Note DS214, Mar 2011.

[253] Xilinx Inc., "RTL and technology schematic viewers tutorial," Manual UG685 (v13.1), Mar 2011.

[254] Xilinx Inc., "Spartan-6 family overview," Product Specifications DS160 (v2.0), Oct 2011.

[255] Xilinx Inc., "Ultimate flexibility and features without sacrificing price, performance, or power," http://www.xilinx.com/applications/wired-communications/network-processing/, ret. Jan 2014.

[256] N. Xiong, L. T. Yang, Y. Zhang, Y. Zhou, and Y. Li, "Design and analysis of a stable queue control scheme for the Internet," in *Proc. of the IEEE Conference on Embedded and Ubiquitous Computing (EUC)*, Shanghai, China, Dec 2008, pp. 378–384.

[257] L. Xu, K. A. Harfoush, and I. Rhee, "Binary increase congestion control (BIC) for fast, long distance networks," in *Proc. of the IEEE INFOCOM*, Hong Kong, China, Mar 2004, pp. 2514–2524.

[258] L. Yamamoto, "Evaluation of a catalytic search algorithm," in *Studies in Computational Intelligence – Nature Inspired Cooperative Strategies for Optimization (NICSO)*, vol. 284, 2010, pp. 75–87.

[259] P. Yiannacouras, J. G. Steffan, and J. Rose, "Soft vector processors vs. FPGA custom hardware: measuring and reducing the gap," in *Proc. of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Monterey (CA), USA, Feb 2009, pp. 277–277.

[260] H. Yin, P. Wang, T. Alpcan, and P. G. Mehta, "Hopf bifurcation and oscillations in a communication network with heterogeneous delays," in *Automatica*, vol. 45, no. 10, Oct 2009, pp. 2358–2367.

[261] M. Yoshimi, Y. Osana, T. Fukushima, and H. Amano, "Stochastic simulation for biochemical reactions on FPGA," in *Proc. of the International Conference on Field Programmable Logic and Application (FPL)*, Leuven, Belgium, Aug 2004, pp. 105–114.

[262] H. Zhang and D. Ferrari, "Rate-controlled service disciplines," in *Journal of High Speed Networks*, vol. 3, no. 4, 1994, pp. 389–412.

[263] H. Zhang and D. Ferrari, "Improving utilization for deterministic service in multimedia communication," in *Proc. of the IEEE International Conference on Multimedia Computing and Systems*, Boston (MA), USA, May 1994, pp. 295–304.

[264] L. Zhang, "Virtual clock: A new traffic control algorithm for packet switching networks," in *ACM Computer Communication Review*, vol. 20, no. 4, Sep 1990, pp. 19–29.

[265] L. Zhang, S. Shenker, and D. D. Clark, "Observations on the dynamics of a congestion control algorithm: the effects of two-way traffic," in *ACM Computer Communication Review*, vol. 21, no. 4, Sep 1991, pp. 133–147.

[266] M. Zhang, B. Karp, S. Floyd, and L. L. Peterson, "RR-TCP: A reordering-robust TCP with DSACK," in *Proc. of the IEEE International Conference on Network Protocols*, Atlanta (GA), USA, Nov 2003, pp. 95–106.

[267] M. Zheng, M. Goldenbaum, S. Stanczak, and H. Yu, "Fast average consensus in clustered wireless sensor networks by superposition gossiping," in *Proc. of the IEEE Wireless Communications and Networking Conference (WCNC '12)*, Paris, France, Apr 2012, pp. 2009–2014.

[268] J. Ziegler and W. Banzhaf, "Evolving control metabolism for a robot," in *Artificial Life*, vol. 7, no. 2, 2001, pp. 171–190.

Index

Curriculum Vitae

Personal informations

Name	Massimo Monti
Date of birth	01 October 1984
Place of birth	Empoli (FI), Italy
Nationality	Italian

Education

2010 - 2014	PhD candidate at the Department of Mathematics and Computer Science, University of Basel, Switzerland. Joint PhD with the Department of Information Engineering, University of Pisa, Italy.
2008 - 2010	Master of Science (MSc - 120 ECTS) in Information Engineering, University of Pisa, Pisa, Italy.
2003 - 2007	Bachelor of Science (BSc - 180 ECTS) in Information Engineering, University of Pisa, Pisa, Italy.
1999 - 2003	High school diploma in Electronics and Telecommunication, I.T.I.S. G. Marconi, Pontedera, Italy.

Work experience

2010 - 2014	Research and teaching at the Department of Mathematics and Computer Science, University of Basel, Switzerland.
Aug - Dec 2007	Assistant in consultancy in Digital Signal Processing for Electromagnetic monitoring, EMC and Safety sectors, Elettronica Monti, Italy.
Jun - Jul 2007	Assistant in consultancy in Digital Signal Processing for Electromagnetic monitoring, EMC and Safety sectors, Narda STS s.r.l. Italy, Italy.

Titles and awards

2013	Best paper award at IFIP/IEEE IM Distributed Autonomous Network Management Systems (DANMS) workshop.
2011	Title of Ingegnere dell'Informazione (Qualification for Information Engineering Practice, earned passing State examinations).
2003	Title of Perito Industriale (equipollent to EU Diploma of post-secondary level of Industrial Engineer, earned passing State examinations).

"Real knowledge is to know the extent of one's ignorance."

Confucius