THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Shared Resources in Distributed Systems: Analytical Tools for Evaluation and Self-stabilizing Provisioning

IOSIF SALEM

Division of Networks and Systems
Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2018

Shared Resources in Distributed Systems: Analytical Tools for Evaluation and Self-stabilizing Provisioning

Iosif Salem

Copyright © Iosif Salem, 2018.

Technical report 152D
Department of Computer Science and Engineering
Distributed Computing and Systems Group
ISBN: 978-91-7597-682-2
Series number: 4363
in the series Doktorsavhandlingar vid Chalmers tekniska högskola.
Ny serie (ISSN0346-718X)

Division of Networks and Systems
Chalmers University of Technology
SE-412 96 GÖTEBORG, Sweden
Phone: +46 (0)31-772 10 00

Author e-mail: `iosif@chalmers.se`

Printed by Chalmers Reproservice
Göteborg, Sweden 2018

Shared Resources in Distributed Systems: Analytical Tools for Evaluation and Self-stabilizing Provisioning

Iosif Salem

Division of Networks and Systems, Chalmers University of Technology

ABSTRACT

Distributed computing is an established computing paradigm of modern computing systems. The nodes of a distributed system interact either by sharing resources or via a communication network. In both cases, provisioning of shared resources is a challenge, for example when resource demand and supply varies or when the system is prone to failures. Analytical tools for evaluating system performance and for provisioning shared resources enhance system design and implementations.

In this thesis, we develop analytical tools for the evaluation and self-stabilizing provisioning of shared-resources in distributed systems. We first focus on systems where resource demand and supply varies, and study cases of reusable and non-reusable resources. We study shared-object systems, where system nodes demand mutually exclusive access to a number of objects in a continuous fashion. We develop analytical tools for computing the expected delay and throughput of such systems, in a wide range of system utilization scenarios, including saturation points. Moreover, we study systems where nodes share energy resources, and focus on optimizing the available resources on a system-level. We develop online algorithms that use the flexibility on resource demand, to optimize the utilization of the available supply, and prove their competitive ratios.

Recovery from failures is necessary for provisioning shared resources. Dynamic and complex systems are often designed based on a failure model, but it is important that they recover even after the occurrence of unexpected failures, outside the failure model. Such failures can include topological changes in the network, stale information in the nodes' memory, communication failures, etc. These failures are further amplified by the system's asynchrony. In

these settings, we first focus on provisioning of network resources, in terms of network control and ordering of distributed events. We study Software-Defined Networks (SDNs) and specifically their control planes. We provide a self-stabilizing distributed algorithm for a fault-tolerant SDN control plane, that deals with communication failures, topological changes, as well as, with transient faults, that can bring the system in an arbitrary state. Moreover, we focus on ordering distributed events in asynchronous message-passing systems, in the absence of execution fairness. In these extreme asynchronous settings, we provide a practically-self-stabilizing distributed algorithm, that uses bounded memory and yet, can tolerate concurrent counter overflows, when counting distributed events, as well as transient faults.

Keywords: resource sharing, shared object systems, online algorithms, smart grid, distributed algorithms, self-stabilization, software-defined networks.

List of appended papers

Parts of the contributions presented in this thesis have previously appeared in the following manuscripts.

- **Iosif Salem**, Elad M. Schiller, Marina Papatriantafilou, Philippas Tsigas, "Shared-object System Equilibria: Delay and Throughput Analysis," in Proceedings of the *17th International Conference on Distributed Computing and Networking (ICDCN)*, Singapore, January 2016, pp. 30:1–30:10, ACM.
  The technical report of this paper appeared under the same title in *CoRR, abs/1508.01660*,
  `http://arxiv.org/abs/1508.01660`, 2015.
- Giorgos Georgiadis, **Iosif Salem**, Marina Papatriantafilou, "Tailor your curves after your costume: Supply-following demand in Smart Grids through the Adwords problem," in Proceedings of the *31st Annual ACM Symposium on Applied Computing (SAC),* Pisa, Italy, April 2016, p.p. 2127-2134.
  The technical report of this paper appeared as *Technical Report 2015:01*, Department of Computer Science and Engineering, Chalmers University of Technology, Göteborg, Sweden, 2015.
- Marco Canini, **Iosif Salem**, Liron Schiff, Elad M. Schiller, and Stefan Schmid, "Renaissance: Self-Stabilizing Distributed SDN Control Plane", technical report in *CoRR, abs/1712.07697*,
  `http://arxiv.org/abs/1712.07697`, 2017.

An earlier version of this work appeared as:
Marco Canini, **Iosif Salem**, Liron Schiff, Elad M. Schiller, and Stefan Schmid, "A Self-Organizing Distributed and In-Band SDN Control Plane", in the Proceedings of the *35th International Conference on Distributed Computing Systems (ICDCS)*, Atlanta, GA, USA, June 2017, p.p. 2656-2657, IEEE.

▷ **Iosif Salem** and Elad M. Schiller, "Practically-Self-Stabilizing Vector Clocks in the Absence of Execution Fairness", technical report in *CoRR, abs/1712.08205*, `http://arxiv.org/abs/1712.08205`, 2017.

in memory of Guido I. Salem

Acknowledgments

Reaching the point of finishing a PhD thesis requires the support of many people, to whom I express my deepest gratitude. It would be impossible to complete this thesis and bear the turbulent journey of a PhD without this support.

First, I would like to thank my thesis supervisor Assoc. Prof. Elad M. Schiller and my coadvisor Assoc. Prof. Marina Papatriantafilou for their support and influence, as well as for the opportunities that they gave me during my studies. I thank my research collaborators Prof. Philippas Tsigas, Dr. Georgios Georgiadis, Assoc. Prof. Stefan Schmid, and Dr. Liron Schiff. Also, I thank Asst. Prof. Pierre Leone for the opportunity of a research visit in the University of Geneva. I thank Assoc. Prof. Chryssis Georgiou, Assoc. Prof. Vana Kalogeraki, and Prof. Shlomi Dolev for the opportunity of presenting my work in their research groups and getting valuable feedback. I thank Prof. Peter Damaschke who was the CSE department's examiner of my thesis.

I am honored to have an excellent group of external faculty members acting as examiners in the public defense of this thesis. I would like to thank Prof. Dr. Christian Scheideler (University of Paderborn) for being the faculty opponent, as well as the members of the grading committee; Prof. Roman Vitenberg (University of Oslo), MCF Lélia Blin (CNRS, Univ. d'Evry-Val-d'Essonne, LIP6), and Assoc. Prof. Ioannis Chatzigiannakis (Sapienza University of Rome).

Special thanks go to Yiannis Nikolakopoulos for the numerous discussions about more or less everything, as well as to Georgios Georgiadis for being a collaborator and a good friend (feel free to construct Venn diagrams Giorgo). My time in Chalmers would have been much different if I didn't have the luck of sharing an office with Thomas Petig and Valentin Tudor; unfortunately, most of our great office plans (e.g. installing a hammock) didn't go through. I thank the (past and present) colleagues in the Network and Systems division for a friendly and warm working environment. I thank Ali, Amir, Aras, Babis, Bapi, Beshr, Boel, Carlo, Daniel, Dimitris, Elena, Farnaz, Fazeleh, Georgia, Hannah, Ivan, Katerina M., Magnus, Mohamed, Nasser, Nhan, Oliver, Oscar, Paul, Peter L., Romaric, Tomas O., Tomas R., Valentin P., Vincenzo, Zhang, as well as the other members of the CSE department. Also, I thank Eva, Marianne, Tiina, Rolf, Peter H., and Rebecca for providing friendly and efficient administration.

I would like to thank the faculty members Olaf Landsiedel, Erland Jonsson, Raffaella Negretti, and Linda Bradley for our excellent collaboration in my TA duties. Also, I thank the students Michael Tran, Christos Profentzas and Rafael Constantinou, Emil Kristiansson and Johan Persson, and Guillermo Barredo for our collaboration in MSc theses or projects. I thank the projects *KARYON* (EU), *Chameleon-MAC* (VR) and *CHRONOS steg1* (Vinova) for funding my research.

Life in Göteborg would have been much different without the good friends I made here. I thank Stavros, Yiannis S., Petros, Vaggelis, Chloe, Vasilis, Suvi, Maria K. & Anton, Simona and little Mara, Christine L., Angelos, Christina and little Sia. Extra credits should go to Manos, for the numerous discussions about life, math, objectively bad jokes, etc. I must thank Marianne Pleen-Schreiber and her family for their great hospitality. Also, I thank the fellows from the PhD Pub; Eoin, Isabel, Aljoscha, Elke, Sigrid, Alla, Siavash, and Niklas.

I thank my good friends Dimitra, Eirik, Harris T., Harry M., Kanellos, Katerina K., Katerina S. Marilena, Nikos, Tina, and Vasiliki, that bear with me all the way back from our years in the Math department in Athens. I thank Zak and Eva for decades of discussions and support, and for pushing me to think in different perspectives. Also, I thank my family, my parents Guido and Sara, and my sister Louiza for their unconditional love and for never compromising their support. I thank Maria for her love and endless support, and for sharing all the good and hard times in these years and the years to come.

Iosif Salem
Göteborg, January 2018

Contents

List of Figures

Part I

INTRODUCTION

1

Introduction

1.1 Motivation

Overview Distributed systems are prevalent in our society. We interact with services that base on distributed systems through many computing devices that we use, and in turn these services require interaction of these devices with other systems (even though we, as users, are usually oblivious to these processes). This computing paradigm scales from smartphone applications communicating with servers or collaborative document editing, to the operation of a data center, where for example, your favorite social media host your data. Systems that are inherently distributed include nodes that need to interact via a network or by sharing resources in order to provide services.

For each service, each node of the system runs a program which determines the *local* actions of the node, as well as, the interaction with other nodes, to the end of providing the service. This interaction, may include coordination for sharing the system's resources, exchange of a node's state, discovering a change in the system's topology, etc. The complexity for the nodes to collectively achieve their tasks depends on a number of factors. Some of these factors are the architectural limitations of the system, faults that occur or have occurred in the past, the system's asynchrony (e.g. communication delays), the number of times a node needs to interact with other nodes for retrieving necessary information, the nodes' inability to predict the future system state (e.g. node additions or failures), the availability of resources, etc. [1].

This thesis aims to address challenges related to resource sharing in distributed systems by taking an analytical approach. We develop analytical tools for problems related to performance evaluation of shared resources systems, as well as for provisioning of shared resources in the presence of failures.

1.1.1 Shared-resources systems

A main paradigm of node interaction in a distributed system is the one in which the system's nodes interact by sharing resources [2, 3]. We refer to resources in an abstract manner, since in practice they can be, for example, the shared-memory of a single computer, distributed storage over multiple computers, energy resources that are generated in different sites of the power grid, cloud resources, network resources, etc. [1, 4]. Designing solutions for handling these shared resources in a distributed system is a challenge in an online setting, where future resource demand and availability is usually not known in advance. It is often the case that these solutions are evaluated experimentally, depending on a set of available data. Thus, analytical tools for studying the performance of such systems provide more flexibility both in the evaluation but also in the solution design. In this thesis we study two problems related to this context; one related to shared objects and another related to sharing energy resources.

Shared-object systems Consider a system that includes shared objects and nodes that interact by gaining and releasing access to subsets of these objects.

This abstraction can, for example, relate to shared-memory, where threads interact for gaining access to memory locations. The more the system is utilized, the higher the *contention* among its nodes, i.e., the events where more than one node needs to access a single object at the same time.

Contention management algorithms determine access to a shared object among a set of competing nodes [5–7]. The criteria for determining the node (or process) that will gain access to the shared object can rely on the number of objects that each competing node holds, an assignment of priorities over the nodes, a first-come first-serve order, a random choice, etc. A contention management scheme comes with some guarantees for the system's progress (e.g. how much a node can be stalled from gaining access to an object), throughput, and delay. Moreover, these algorithms are mostly evaluated experimentally and few analytical results exist in the literature (e.g. [8]).

Sharing energy resources Consider a system in which some nodes demand (non-reusable) resources and some nodes supply (and possibly demand) resources. In this system producing and consuming resources comes at a cost, resource demand and supply varies, and consumer nodes issue demands subject to a number of constraints (e.g. temporal or cost related). An example of such systems is the power grid, when considering energy resources and nodes of the grid that produce and consume energy.

The increasing inclusion of renewable energy generation in the power grid, the use of different energy carriers, the ability of the end-user to choose among different utility companies, as well as new consumption monitoring technologies are factors that are changing the traditional operation of the power grid [9]. Nodes of the grid can be both producers and consumers of energy (e.g. end users or companies that own an installation of renewables), and decide among different energy supply options for covering their energy demand. Even though energy supply is usually not a challenge (except for high demand peaks), using the grid's energy resources efficiently to reduce the production or consumption cost is non-trivial. That is, costs can be optimized by individual nodes, subsets of nodes, or in a system-level, by taking into account the varying supply and demand (and optionally the ability to forecast the future demand and supply). Solutions to the latter problem are nowadays supported by the ability to collect frequently consumption data from smart meters, i.e., metering devices that can connect remotely with the utility company (hence the term *smart grid* [9]).

1.1.2 Provisioning shared resources in distributed systems

Another prominent way of viewing distributed systems is to consider nodes that are interconnected via a network in order to provision resources, such as network resources or shared object replicas. For any task that the nodes need to solve, they need to communicate via the network, thus facing a list of challenges. Some of them are the number of times that each node needs to communicate with its neighbors, communication delays, the system's asynchrony, changes in the network topology, faults that occur or *stale* information that resides in the network, byzantine nodes, etc. [1, 2]. Here we mainly focus on the effect of stale information in asynchronous distributed systems in problems related to provisioning network resources and to ordering events in asynchronous distributed systems.

Provisioning network resources Since distributed systems lack central coordination (before any communication among the nodes occurs), distributed algorithms that run on these systems are dependent on the network's limitations and state. Most models of distributed computation in a network of nodes (e.g. local, congest, interleaving models [10–12]) use assumptions on the topology (e.g. fully connected graph, ring, arbitrary graph, etc.), the level of synchrony (i.e., how often nodes interact with the network), the delay in communication, the presence or not of distinct node identifiers, the fault model of the system, etc. These challenges are further amplified by limitations that come from the network's architecture, when we deploy distributed algorithms in practice.

Ordering events in asynchronous distributed systems A desirable property for any distributed system is the ability to argue about the order in which distributed events occurred. To that end, various notions of clocks have been used, e.g. clocks that count time [1] or distributed events [13–15]. Of course, nodes need to communicate in order to have both a common reference in their clock values, but also to learn about the local events that occurred in neighboring nodes. Such tasks become more challenging in the presence of failures, and various fault-models exist in the literature [1, 10, 16].

1.1.3 Thesis organization

This thesis consists of three parts and is organized as follows. We continue Part I (Introduction) by giving the background for the sections to follow (Section 1.2). Then, we present the two areas of focus of the thesis, including challenges and related work (sections 1.3 and 1.4). In the following, we formulate the two research questions that this thesis aims to address in each of the two areas of focus (Section 1.5), and explain how we contribute to these research questions through the appended papers, i.e., papers I–IV (Section 1.6). In Section 1.7 we conclude and discuss future directions.

In Part II of the thesis we append the complete technical reports of papers I–IV. In Part III, we discuss the technical contributions of the thesis and the analytical tools that we developed, which can be also used for relevant problems.

1.2 Background

In this section we give the necessary background, before proceeding with the overview of this thesis in the following sections. In Section 1.2.1 we give an introduction to Software-Defined Networks (SDNs) and in Section 1.2.2 we introduce Self-stabilization.

1.2.1 Software-defined networks

Computer networks (the Internet, data-center networks, enterprise networks, etc.) are a critical infrastructure. However, today's computer networks are often inflexible, complex and error-prone, raising concerns regarding their dependability. Recently, leading tech companies have reported major issues with their networks, due to misconfigurations [17–19]. Software-Defined Networks (SDNs) have emerged as a promising alternative, providing new opportunities for designing more dependable networks [20]. By outsourcing and consolidating the control over the data plane devices (switches, routers, basic middleboxes) to a logically centralized controller software, SDNs introduce interesting new flexibilities. In particular, the decoupling of the control plane from the data plane allows to innovate the former independently of the latter. Moreover, SDNs enable a principled and formal specification of the network configuration, also enabling an automated verification [21].

1.2.2 Self-stabilization

Consider a message-passing system, i.e., a set of nodes connected via a network, such that each node can be modeled as a finite-state machine and the network's communication channels have finite capacity. Message-passing systems can be designed to tolerate failures based on a fault model [16], such as topological changes (node or link failures) or communication failures. In addition to these failures, it is possible that failures outside the fault model can occur. We consider *transient faults*, i.e., any temporary violation of assumptions according to which the system and network were designed to behave, e.g., the corruption of the system state due to soft errors. We assume that these transient faults arbitrarily change the system state in unpredictable manners (while keeping the nodes' program code intact). Since these transient faults are rare, a common assumption is that all transient faults occurred before the start of the system execution.

Self-stabilization is a design criterion that requires a system, which may start in an arbitrary state, to return to a correct behavior within a bounded period and was introduced by Dijkstra [22]. That is, for any execution, the system is guaranteed to reach a legitimate state (according to a task's specification) within a bounded time, and continue being in a legitimate state for the remainder of the execution. Self-stabilizing (distributed) algorithms have been developed for a large variety of systems, e.g. self-stabilizing algorithms for peer-to-peer networks [23], mobile robots [24], etc. [10].

Asynchronous message-passing systems cannot always fulfill Dijkstra's stabilization requirements (often referred to as strong self-stabilization). Adversarial schedulers can allow stale information (e.g. due to transient faults) to reside in the system for an unbounded prefix of any execution, and then appear to violate the system's safety requirements. Thus, research has focused on more relaxed stabilization criteria. *Pseudo-stabilization* [10, 25] deals with the above inability by bounding the number of times the system violates safety in an infinite system execution. Moreover, *practically-self-stabilizing systems* [26–29] require a bounded number of safety violations during any *practically infinite* period of a system execution. A practically infinite execution [28, 29] is an execution of bounded but extreme size, say, of 2^b sequential processor steps, where $b = 64$ or an even a larger integer, as long as a constant number of bits can represent it. These relaxed notions of self-stabilization are relevant for this thesis, however more proposals exist in the literature, e.g. the ones in [30–32].

1.3 Analytical tools for evaluating the performance of distributed systems

Overview In this section we focus on analytical tools that evaluate algorithms for sharing resources in dynamic distributed systems. We present the motivation, challenges, and related work in the context of relevant problems that we study in this thesis.

1.3.1 Analytical performance evaluation of shared-object systems

Motivation, challenges, and related work Consider a system that consists of a set of computing entities, which we call threads and a number of reusable objects. Each thread runs a sequential program (a job), for which it has to acquire a subset of these objects in order to perform an operation for a bounded time. Once the operation completes, the thread releases access to the job's objects and waits until another job is assigned to it (e.g. by a scheduler).

The order of object acquisition plays a crucial role in the progress of such systems. For example, it is important to avoid deadlocks or livelocks, i.e., situations in which two or more processes make no progress with or without changing their state, unless they are interrupted. A simple solution for avoiding such *race conditions* is to force the threads to follow the same order (e.g. ascending or descending) when acquiring their jobs' objects. Also, as the workload of threads increases, it is more probable that threads compete for accessing the same objects.

Working systems that follow the discussed paradigm include multi-word compare-and-swap (CASN) operations and fine-grained locking implementations in shared-memory systems [33–36], as well as transactional memories [37, 38]. A common way to model such systems is to consider a generalization of the dining philosophers problem, as in [39, 40], in which every job includes a fixed set of objects that it may need. This problem has well-known results studying the worst-case job delays, which may even be exponential on the system's size, i.e., the chromatic number of the resource graph [8, 39]. In this graph, the vertices (objects) are connected if there is, at least, one thread that may request them both at any point in time. In practice, the expected delay and throughput is rather different than the worst case and, therefore, computer experiments are the common way for evaluating the system performance.

1.3.2 Optimizing resource allocation on a system level: the smart grid case

Motivation, challenges, and related work The power grid is rapidly shifting nowadays towards a dynamic market of energy resources. Until recently, utility companies were the main suppliers of energy and their operation followed the utility service paradigm, i.e., all demands must be satisfied irrespective of the available supply (demand-following supply [41]). However, this paradigm becomes very costly for the utility companies in very high demand peaks, since they need to maintain their production in higher levels than the average consumption. This need to reduce high production costs, can be achieved by shifting the demand curve to follow the supply curve, as much as possible.

From a consumer point of view, there is a wide range of available energy supply services, either from different utility companies, or from local-scale energy production and brokering, or from own generated resources (e.g. photovoltaic arrays). The choice between all of these options is based on information about the sources, usually price-related (*pricing signal* [42]). The common thread underlying both real world practice and relevant research is that *no single actor has full control over all pricing signals* [9]. Therefore, the standard model of energy utilization can no longer guarantee an efficient system-level utilization.

1.4 Analytical tools for self-stabilizing provisioning of distributed systems

Overview In this section we focus on analytical tools for self-stabilizing provisioning of resources in distributed systems. We present the motivation, challenges, and related work in the context of relevant problems that we study in this thesis.

1.4.1 Fault-tolerant Software-Defined Network control planes

Motivation, challenges, and related work Software-Defined Networks (SDNs) have emerged as a promising alternative, providing new opportunities for designing more dependable networks. SDNs outsource the control over the data plane devices (switches, routers, basic middleboxes) to a logically centralized software entity. We refer to that entity as the SDN *control plane*. This decoupling allows a more flexible network design, by enabling the development of the control plane independently of the data plane (e.g. automated verification [21]).

Since control is logically centralized, designing fault-tolerant SDN control planes is crucial. To that end, it is important that the control plane is physically distributed, in order to provide robustness. That is, a *decentralized control plane* can tolerate controller failures by relying on multiple and redundant controllers. Moreover, decentralized control planes can improve scalability and performance (latency).

Decoupling the control plane from the date plane raises the challenge of the control plane quickly reacting to data plane events. This becomes more challenging when control is done *in-band*, i.e., the control plane is part of the network (e.g. network attached servers). Even though most deployments of SDNs rely on out-of-band control [43–45], where control plane packets are carried by a dedicated management network, in-band control is desirable for many reasons. Except for the economical and connectivity benefits, as well as the benefit of not having to maintain a separate management network, they enhance fault-tolerance (by redundancy). That is, control traffic can also be forwarded with data plane traffic, instead of using only the dedicated management ports of the switches (as in out-of-band). Of course, these benefits come with the challenge of demultiplexing control and data traffic at the switches.

While the benefits of separating the control plane with the data plane have been well founded in the literature [43, 44, 46–48], the question of how connectivity between these two planes is maintained (i.e., the communication channels from controllers to switches and between controllers) has not received much attention. This raises several concerns regarding the availability of the SDN architecture. For example, it is a challenge to guarantee that the SDN control plane can always establish a route between any pair of switches and controllers, given a physically connected data plane. To that end, connections can be made with the fault-tolerance literature, in order to guarantee the provisioning of network resources.

1.4.2 Ordering distributed events in asynchronous systems that are prone to failures

Motivation, challenges, and related work Self-stabilizing systems [10, 22] recover to a legitimate state after the occurrence of an arbitrary combination of failures. Distributed systems that are self-stabilizing rely on fairness assumptions regarding communication and scheduling (execution fairness), as well as on assumptions regarding synchrony. Communication is fair when a message that is sent infinitely often is received infinitely often [10]. Similarly, an execution is fair when every step that is applicable infinitely often is executed infinitely often [10] (hence no processor can crash after the start of the system's execution). In asynchronous systems, recovery is often designed and measured based on synchronization rounds or similar notions [39]. However, when studying systems in which any of these assumptions do not hold, more relaxed notions of stabilization are often used, such as pseudo-stabilizing [10, 25] or practically-self-stabilizing algorithms [26–29] (cf. Section 1.2.2).

Providing solutions in asynchronous distributed systems, in the absence of mechanisms for synchronization or roll-back is a challenge, especially in the presence of failures. For example, when ordering distributed events, it is important to develop algorithms that use bounded storage, and tolerate failures (whether they are included in the failure model [16] or not) as well as arbitrary processor rates. In the absence of execution fairness, processors may crash even after the starting configuration, hence relying on synchronization rounds is no longer possible. Therefore, standard solutions for ordering distributed events, such as vector clock algorithms [13, 15], need to be redesigned to cope with these extreme asynchronous settings. Since vector clocks include a wide range of applications, such as constructing distributed snapshots [3] or using them as building blocks in various conflict-free replicated data types (CRDTs) [49], it is important to provide vector clock algorithms that overcome the discussed challenges.

1.5 Research questions

We consolidate and position the challenges of sections 1.3 and 1.4 in two research questions that we present in the following. In Section 1.6 we discuss how this thesis addresses these research questions.

Research question 1 (RQ 1)**.** How to evaluate analytically the performance of algorithms for resource-sharing in distributed systems, in which resource demand and supply varies?

Research question 2 (RQ 2)**.** How to deal effectively with the effect of transient faults in an asynchronous message-passing system, in which changes in the topology can occur at any time?

1.6 Thesis contribution

Overview We present the contributions of this thesis with respect to the challenges and the related work (sections 1.3 and 1.4), as well as, the research questions in Section 1.5.

1.6.1 Analytical performance evaluation of resource allocation systems (RQ 1)

Analytical performance evaluation of shared-object systems (Paper I) We study shared-object systems, which consist of a fixed number of threads and objects. Threads carry out jobs by acquiring access to subsets of objects, on which they perform operations of bounded time. We assume that jobs are assigned to the threads following known exponential distributions (arrival rates) and threads acquire their jobs' objects in an ascending (object) order. For such systems we estimate analytically the expected job delay and throughput.

We estimate the system's performance in a wide range of scenarios. Our analysis provides estimates of the job delay and throughput, when the job arrival rates match the job completion rates. In these cases, the system is in a *shared-Object System Equilibrium* (OSE). The existing literature often focuses on peak utilization scenarios, i.e., saturation points. However, saturation points are special cases of OSEs, where (1) the system is in equilibrium and (2) any increase in the job arrival rates cannot increase any further the job completion rates. Thus, our analysis covers a wider range of system equilibria.

For a given $\varepsilon > 0$ and an OSE, we say that the system is in an ε-*OSE* when the completion rate of any job differs from the one of an OSE by at most ε. We develop (polynomial-time) algorithms for estimating delay and throughput in ε-OSEs. To that end, we study the conditions for a given shared-object system to be in an OSE as well as contention-related properties of OSEs, i.e., the *expected* job delay and completion rate, as well as the time in which each thread blocks other threads and by that prevents them from making progress. We then propose a procedure for finding ε-OSEs, if such exist in the given system.

Optimizing resource allocation on a system level: the smart grid case (Paper II) We consider the energy dispatch problem, where energy demands are issued by consumer sites (at arbitrary intervals) and must be satisfied within a certain time range (timeslot) by the energy supply sites. These demands can have flexibility regarding the timeslot in which they must be satisfied, restrictions on the energy carrier to satisfy them (e.g. thermal or electric), and (optionally) produce energy storage for later use.

To the end of shaping the demand curve to follow the supply, we introduce the concept of *energy budget* (or simply *budget*) for every timeslot, by combining energy and price information. Intuitively, budgets reflect the ability and cost of supplying energy for every timeslot. With this approach, we reformulate the energy utilization problem as a budget utilization problem. That is, by maximizing the utilization of the available budget we achieve both to force the demand curve to follow (as much as possible) the supply curve, and also to reduce high demand peaks through adaptive scheduling.

This modeling approach allows us to connect to research fields such as bipartite matching and scheduling, and to use these tools for solving the budget utilization problem. In fact, we propose a novel modeling of the energy dispatch problem based on the Adwords problem [50], in which a set of bidders with given budgets, place bids for a newly revealed adword. To that end, we utilize a proposed modeling from previous work [51] that maps the energy dispatch problem to an online scheduling problem. We provide solutions that are orthogonal to pricing schemes and prove their online guarantees (competitive ratio [52]). Moreover, we identify an extension of the ADWORDS problem that includes dynamic budgets, and address it through our algorithms.

In this modelling, bids for adwords (that utilize the budget) reflect the current ability and cost to serve a demand in a specific timeslot. When the bids are very small compared to the budgets, we solve the utilization problem with $(1 - \frac{1}{e})$-competitive ratio. When the bids can be comparable to the budgets (e.g. due to local-scale generation), we solve the energy utilization problem with $\frac{1}{2}$-competitive ratio.

1.6.2 Analytical tools for self-stabilizing provisioning of distributed systems (RQ 2)

Bootstrapping the control-plane of a software-defined network in the presence of failures (Paper III) We design a self-stabilizing software-defined network control plane, i.e., an SDN that recovers from controller, switch, and link failures, as well as a wide range of communication failures. To that end, we model the SDN control plane as an asynchronous distributed system and rely on message passing for communication. We consider a failure model that includes fail-stop failures of controllers, link failures, and a wide range of communication failures, including omission, packet duplication, and packet reordering. We assume that up to κ concurrent link failures can occur at any point in time, for some parameter $\kappa \in \mathbb{Z}^+$, as well as access to a (link) failure detector. Moreover, we also assume that transient faults (e.g., the corruption of the packet forwarding rules or malicious changes to the availability of links, switches, and controllers) can bring any execution of the system to an arbitrary starting system state (while keeping the program code intact).

We develop an algorithm to bootstrap and maintain connectivity in an inband and distributed SDN control plane, in the presence of the failures mentioned above. Our algorithm maintains control flows between any controller and any other node in the network (switch or controller). In fact, in the presence of at most κ link failures, we achieve bounded communication delays. We assume no external (out-of-band) support of the control plane, controllers that can fail-stop, and yet provide bounded time recovery after the occurrence of an arbitrary combination of failures. Once the control plane exhibits bounded communication delays, the controllers can coordinate their network operations (e.g. traffic balancing or installing data-flows between hosts).

To that end, we provide a (distributed) self-stabilizing algorithm for decentralized SDN control planes, that recovers from arbitrary combinations of failures, given that the network topology is $(\kappa + 1)$-edge-connected and includes at least one (non-failed) controller. First, we show that our algorithm recovers from transient faults. Starting from an arbitrary state, the system recovers within time $O(D^2N)$ to a legitimate state, where N is the number of nodes in the system and D is the maximum system diameter (regardless of link failures). In a legitimate state, no stale information exists in the memory of the system nodes (e.g. regarding unreachable controllers or stale rules) and the switches store rules that (1) facilitate $O(D)$ flows between any controller and any other node in the network, and (2) maintain bounded communication delays in the event of at most κ concurrent link failures.

We show bounds on the memory requirements of the controllers and the switches. In a legitimate state, the number of packet forwarding rules at every switch are at most N_C times the optimal, where N_C is the number of controllers. We also show that starting from a legitimate state, the system can recover from a wide range of topological changes within $O(D)$ time.

Ordering events in asynchronous systems that are prone to failures (Paper IV) We design a highly fault-tolerant distributed algorithm for vector clocks, in the absence of execution fairness. We consider asynchronous message passing systems, in which node and communication failures, as well as transient faults can occur. Specifically, we assume crash failures of nodes, that can optionally perform undetectable restarts (i.e., resume with the same state as before crashing, possibly having lost incoming messages and without being aware that a crash occurred), as well as packet failures, such as omission, duplication, and reordering [16]. Moreover, we assume that transient faults can bring the system in an arbitrary starting state, while leaving the program code intact. Since transient faults are rare, we assume that they occur before the beginning of a system execution.

We present a practically-self-stabilizing vector clock algorithm that deals with the failures mentioned above and does not require synchronization guarantees, nor uses mechanisms for synchronization or roll-back, *even during the period of recovery from failures not included in the failure model*. To that end, we interpret the requirements of practically-self-stabilizing algorithms, by demanding that for every practically-infinite [29] system execution, the number of safety violations is insignificant with respect to the the execution size. We use existing practically-self-stabilizing labeling schemes [26, 28] for constructing a data structure of $O(N^3)$ size (where N is the number of processors) that supports the vector clock functionalities, and yet tolerates the studied failures.

Our solution uses bounded memory for every processor in the system. Our proposed vector clock data structure considers $3N$ integers and two labels [28] per vector, where each label's size is in $O(N^3)$. We rely on bounded counters for recording the system's events, and present elegant techniques for dealing with concurrent counter overflows, such that counter increments (i.e. events) are never lost, even though vector clocks with different labels might exist in the system. Hence, by counting events correctly, we show that it is possible to reason about causality, during a legal execution. We show that for every practically-infinite execution, at most $O(N^8\mathscr{C})$ safety violations occur, where $\mathscr{C}$ is a bound on the capacity of the communication channels.

1.7 Conclusion and future directions

This thesis focuses on the research area of sharing resources in distributed systems. We focus on problems that relate to analytical tools for evaluating the performance of resource allocation algorithms in such systems, as well as, for provisioning shared resources in a self-stabilizing manner. Regarding the first area of focus, we provide performance guarantees for systems in which resources are allocated in an online fashion, in terms of expected and worst case performance. Regarding the second area of focus, we provide algorithms for overcoming the effects of transient faults, in addition to those of the failure model, even in the presence of topological changes that occur in the system.

Some areas and problems that connect to the ones studied in this thesis are, for example, (i) the analytical performance evaluation of shared-object systems that follow acquisition schemes different than the sequential one that was studied in Paper I, (ii) scheduling solutions for matching energy supply and demand in the smart grid that study the effect of price fluctuations in computing the load allocation, as well as, connections to smart city frameworks and infrastructures [53] (cf. Paper II), (iii) combining in-band and out-of-band control depending on network sub-regions (cf. Paper III), and (iv) practically-self-stabilizing algorithms related to other CRDT primitives than vector clocks (cf. Paper IV).

Bibliography

[1] Andrew S. Tanenbaum and Maarten Van Steen, *Distributed systems: principles and paradigms*, Prentice-Hall, 2007.

[2] Hagit Attiya and Jennifer Welch, *Distributed computing: fundamentals, simulations, and advanced topics*, vol. 19, John Wiley & Sons, 2004.

[3] Maurice Herlihy and Nir Shavit, *The art of multiprocessor programming*, Morgan Kaufmann, 2011.

[4] Vinay Setty, Roman Vitenberg, Gunnar Kreitz, Guido Urdaneta, and Maarten Van Steen, "Cost-effective resource allocation for deploying pub-/sub on cloud," in *Distributed Computing Systems (ICDCS), 2014 IEEE 34th International Conference on*. IEEE, 2014, pp. 555–566.

[5] Rachid Guerraoui, Maurice Herlihy, and Bastian Pochon, "Towards a theory of transactional contention managers," in *Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing, PODC 2006, Denver, CO, USA, July 23-26, 2006*, Eric Ruppert and Dahlia Malkhi, Eds. 2006, pp. 316–317, ACM.

[6] Maurice Herlihy, Victor Luchangco, Mark Moir, and William N Scherer III, "Software transactional memory for dynamic-sized data structures," in *Proceedings of the twenty-second annual symposium on Principles of distributed computing*. ACM, 2003, pp. 92–101.

[7] Virendra J. Marathe and Michael L. Scott, "A qualitative survey of modern software transactional memory systems," *University of Rochester Computer Science Dept., Tech. Rep*, 2004.

[8] Nancy A. Lynch, "Upper bounds for static resource allocation in a distributed system," *Journal of Computer and System Sciences*, vol. 23, no. 2, pp. 254–278, 1981.

[9] Sarvapali D. Ramchurn, Perukrishnen Vytelingum, Alex Rogers, and Nicholas R. Jennings, "Putting the 'smarts' into the smart grid: a grand challenge for artificial intelligence," *Communications of the ACM*, vol. 55, no. 4, pp. 86–97, 2012.

[10] Shlomi Dolev, *Self-stabilization*, MIT press, 2000.

[11] Laurent Feuilloley and Pierre Fraigniaud, "Survey of distributed decision," *CoRR*, vol. abs/1606.04434, 2016.

[12] Jukka Suomela, *Distributed Algorithms*, Aalto University, Finland, 2016, Available at: `https://users.ics.aalto.fi/suomela/da`.

[13] Colin J Fidge, "Timestamps in message-passing systems that preserve the partial ordering," 1987.

[14] Leslie Lamport, "Time, clocks, and the ordering of events in a distributed system," *Communications of the ACM*, vol. 21, no. 7, pp. 558–565, 1978.

[15] Friedemann Mattern et al., "Virtual time and global states of distributed systems," *Parallel and Distributed Algorithms*, vol. 1, no. 23, pp. 215–226, 1989.

[16] Chryssis Georgiou and Alexander A Shvartsman, "Cooperative task-oriented computing: Algorithms and complexity," *Synthesis Lectures on Distributed Computing Theory*, vol. 2, no. 2, pp. 1–167, 2011.

[17] GitHub, "`github.com/blog/1346networkproblemslastfriday`," in *The GitHub Blog*, 2016.

[18] Joab Jackson, "Godaddy blames outage on corrupted router tables," in *PC World*, 2011.

[19] Ram Mohan, "Storms in the cloud: Lessons from the amazon cloud outage," in *Security Week*, 2011.

[20] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[21] Carolyn Jane Anderson, Nate Foster, Arjun Guha, Jean-Baptiste Jeannin, Dexter Kozen, Cole Schlesinger, and David Walker, "NetkAT: semantic foundations for networks," in *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*, Suresh Jagannathan and Peter Sewell, Eds. 2014, pp. 113–126, ACM.

[22] Edsger W. Dijkstra, "Self-stabilizing systems in spite of distributed control," *Communications of the ACM*, vol. 17, no. 11, pp. 643–644, 1974.

[23] Thomas Clouser, Mikhail Nesterenko, and Christian Scheideler, "Tiara: A self-stabilizing deterministic skip list and skip graph," *Theoretical Computer Science*, vol. 428, pp. 18–35, 2012.

[24] Lélia Blin, Maria Gradinariu Potop-Butucaru, and Sébastien Tixeuil, "On the self-stabilization of mobile robots in graphs," in *International Conference On Principles Of Distributed Systems*. Springer, 2007, pp. 301–314.

[25] James E. Burns, Mohamed G. Gouda, and Raymond E. Miller, "Stabilization and pseudo-stabilization," *Distributed Computing*, vol. 7, no. 1, pp. 35–42, 1993.

[26] Noga Alon, Hagit Attiya, Shlomi Dolev, Swan Dubois, Maria Potop-Butucaru, and Sébastien Tixeuil, "Practically stabilizing swmr atomic memory in message-passing systems," *Journal of Computer and System Sciences*, vol. 81, no. 4, pp. 692–701, 2015.

[27] Peva Blanchard, Shlomi Dolev, Joffroy Beauquier, and Sylvie Delaët, "Practically self-stabilizing paxos replicated state-machine," in *Networked Systems - Second International Conference, NETYS 2014, Marrakech, Morocco, May 15-17, 2014. Revised Selected Papers*, Guevara Noubir and Michel Raynal, Eds. 2014, vol. 8593 of *Lecture Notes in Computer Science*, pp. 99–121, Springer.

[28] Shlomi Dolev, Chryssis Georgiou, Ioannis Marcoullis, and Elad Michael Schiller, "Practically stabilizing virtual synchrony," *CoRR*, vol. abs/1502.05183, 2015. An earlier version appeared in the Proceedings of the 17th International Symposium Stabilization, Safety, and Security of Distributed Systems, SSS 2015, Edmonton, AB, Canada, August 18-21, 2015.

[29] Shlomi Dolev, Ronen I. Kat, and Elad Michael Schiller, "When consensus meets self-stabilization," *Journal of Computer and System Sciences*, vol. 76, no. 8, pp. 884–900, 2010.

[30] Alain Bui, Ajoy K Datta, Franck Petit, and Vincent Villain, "Snap-stabilization and pif in tree networks," *Distributed Computing*, vol. 20, no. 1, pp. 3, 2007.

[31] Stéphane Devismes, Sébastien Tixeuil, and Masafumi Yamashita, "Weak vs. self vs. probabilistic stabilization," in *Distributed Computing Systems, 2008. ICDCS'08. The 28th International Conference on*. IEEE, 2008, pp. 681–688.

[32] Mohamed Gouda, "The theory of weak stabilization," *Self-Stabilizing Systems*, pp. 114–123, 2001.

[33] Phuong Hoai Ha and Philippas Tsigas, "Reactive multiword synchronization for multiprocessors," in *Parallel Architectures and Compilation Techniques, 2003. PACT 2003. Proceedings. 12th International Conference on*. IEEE, 2003, pp. 184–193.

[34] Phuong Hoai Ha, Philippas Tsigas, Mirjam Wattenhofer, and Rogert Wattenhofer, "Efficient multi-word locking using randomization," in *Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*. ACM, 2005, pp. 249–257.

[35] Timothy Harris, Keir Fraser, and Ian Pratt, "A practical multi-word compare-and-swap operation," *Distributed Computing*, pp. 265–279, 2002.

[36] H*a*kan Sundell, "Wait-free multi-word compare-and-swap using greedy helping and grabbing," *International Journal of Parallel Programming*, vol. 39, no. 6, pp. 694–716, 2011.

[37] Maurice Herlihy and J Eliot B Moss, *Transactional memory: Architectural support for lock-free data structures*, vol. 21, ACM, 1993.

[38] Nir Shavit and Dan Touitou, "Software transactional memory," *Distributed Computing*, vol. 10, no. 2, pp. 99–116, 1997.

[39] Nancy A. Lynch, *Distributed Algorithms*, Morgan Kaufmann, 1996.

[40] Marina Papatriantafilou and Philippas Tsigas, "On distributed resource handling: Dining, drinking and mobile philosophers," in *On Principles Of Distributed Systems, Proceedings of the 1997 International Conference,*

Chantilly, France, December 10-12, 1997, Alain Bui, Marc Bui, and Vincent Villain, Eds. 1997, pp. 293–308, Hermes.

[41] Randy H. Katz, David E. Culler, Seth Sanders, Sara Alspaugh, Yanpei Chen, Stephen Dawson-Haggerty, Prabal Dutta, Mike He, Xiaofan Jiang, Laura Keys, et al., "An information-centric energy infrastructure: The berkeley view," *Sustainable Computing: Informatics and Systems*, vol. 1, no. 1, pp. 7–22, 2011.

[42] Wenxian Yang, Rongshan Yu, and Milashini Nambiar, "Quantifying the benefits to consumers for demand response with a statistical elasticity model," *IET Generation, Transmission & Distribution*, vol. 8, no. 3, pp. 503–515, 2014.

[43] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer, "Achieving High Utilization with Software-Driven WAN," in *SIGCOMM*, 2013.

[44] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jon Zolla, Urs Hölzle, Stephen Stuart, and Amin Vahdat, "B4: Experience with a Globally-Deployed Software Defined WAN," in *SIGCOMM*, 2013.

[45] Teemu Koponen, Keith Amidon, Peter Balland, Martín Casado, Anupam Chanda, Bryan Fulton, Igor Ganichev, Jesse Gross, Natasha Gude, Paul Ingram, Ethan Jackson, Andrew Lambeth, Romain Lenglet, Shih-Hao Li, Amar Padmanabhan, Justin Pettit, Ben Pfaff, Rajiv Ramanathan, Scott Shenker, Alan Shieh, Jeremy Stribling, Pankaj Thakkar, Dan Wendlandt, Alexander Yip, and Ronghua Zhang, "Network Virtualization in Multi-tenant Datacenters," in *NSDI*, 2014.

[46] Martin Casado, Michael J. Freedman, Justin Pettit, Jianying Luo, Nick McKeown, and Scott Shenker, "Ethane: Taking Control of the Enterprise," in *SIGCOMM*, 2007.

[47] Mark Reitblatt, Nate Foster, Jennifer Rexford, Cole Schlesinger, and David Walker, "Abstractions for Network Update," in *SIGCOMM*, 2012.

[48] Nedeljko Vasić, Dejan Novaković, Satyam Shekhar, Prateek Bhurat, Marco Canini, and Dejan Kostić, "Identifying and Using Energy-Critical Paths," in *CoNEXT*, 2011.

[49] Marc Shapiro, Nuno M. Preguiça, Carlos Baquero, and Marek Zawirski, "Conflict-free replicated data types," in *Stabilization, Safety, and Security of Distributed Systems - 13th International Symposium, SSS 2011, Grenoble, France, October 10-12, 2011. Proceedings*, 2011, pp. 386–400.

[50] Aranyak Mehta, "Online matching and ad allocation," *Foundations and Trends in Theoretical Computer Science*, vol. 8, no. 4, pp. 265–368, 2013.

[51] Giorgos Georgiadis and Marina Papatriantafilou, "Dealing with storage without forecasts in smart grids: Problem transformation and online scheduling algorithm," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, New York, NY, USA, 2014, SAC '14, pp. 518–524, ACM.

[52] Allan Borodin and Ran El-Yaniv, *Online computation and competitive analysis*, cambridge university press, 2005.

[53] Evangelos Theodoridis, Georgios Mylonas, and Ioannis Chatzigiannakis, "Developing an iot smart city framework," in *Information, intelligence, systems and applications (iisa), 2013 fourth international conference on*. IEEE, 2013, pp. 1–6.