Lecture Notes in Computer Science 8255

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Teruo Higashino Yoshiaki Katayama
Toshimitsu Masuzawa Maria Potop-Butucaru
Masafumi Yamashita (Eds.)

Stabilization, Safety, and Security of Distributed Systems

15th International Symposium, SSS 2013
Osaka, Japan, November 13-16, 2013
Proceedings

Springer

Volume Editors

Teruo Higashino
Osaka University, Suita, Japan
E-mail: higashino@ist.osaka-u.ac.jp

Yoshiaki Katayama
Nagoya Institute of Technology, Japan
E-mail: katayama@nitech.ac.jp

Toshimitsu Masuzawa
Osaka University, Suita, Japan
E-mail: masuzawa@ist.osaka-u.ac.jp

Maria Potop-Butucaru
University Paris 6, France
E-mail: maria.gradinariu@lip6.fr

Masafumi Yamashita
Kyushu University, Fukuoka, Japan
E-mail: mak@csce.kyushu-u.ac.jp

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-319-03088-3 e-ISBN 978-3-319-03089-0
DOI 10.1007/978-3-319-03089-0
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: Applied for

CR Subject Classification (1998): D.4.5, D.4.7, F.3.1, F.1, C.2.4, K.6.5

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The papers in this volume were presented at the 15th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS), held during November 13–16, 2013, in Osaka, Japan.

SSS is an international forum for researchers and practitioners working on the design and development of distributed systems with self-* properties: (classical) self-stabilizing, self-configuring, self-organizing, self-managing, self-repairing, self-healing, self-optimizing, self-adaptive, and self-protecting. Research in distributed systems is now at a crucial point in its evolution, marked by the importance of dynamic systems such as peer-to-peer networks, large-scale wireless sensor networks, mobile ad hoc networks, cloud computing, robotic networks, etc. Moreover, new applications such as grid and Web services, banking and e-commerce, e-health and robotics, aerospace and avionics, automotive, industrial process control, etc. have joined the traditional applications of distributed systems.

The theory of self-stabilization has been enriched in the last 30 years by high-quality research contributions in the areas of algorithmic techniques, formal methodologies, model theoretic issues, and composition techniques. All these areas are essential to the understanding and maintenance of self-* properties in fault-tolerant distributed systems.

This year the program was organized into several tracks reflecting most topics related to self-* systems. The tracks were: (1) Self-Stabilization, (2) Fault-Tolerance and Dependability, (3) Formal Methods and Distributed Systems, (4) Ad Hoc, Sensors, Mobile Agents and Robot Networks, and (5) P2P, Social, Self-Organizing, Autonomic and Opportunisitic Networks.

We received 68 submissions from 20 countries. Each submission was reviewed by at least three Program Committee members with the help of external reviewers. Out of the 68 submissions, 23 papers were selected as regular papers, and 12 papers were accepted as brief announcements. Among the 23 regular papers, we considered two papers for special awards. The best paper award was given to Heger Arfaoui, Pierre Fraigniaud, and Andrzej Pelc for "Local Decision and Verification with Bounded-Size Outputs," and the best student paper award was given to Fabienne Carrier, Ajoy K. Datta, Stéphane Devismes, Lawrence L. Larmore, and Yvan Rivierre for "Self-Stabilizing (f,g)-Alliances with Safe Convergence." This year, we were very fortunate to have two distinguished tutorial speakers, Onur Altintas (Toyota InfoTechnology Center) and Shlomi Dolev (Ben-Gurion University of the Negev), and two distinguished keynote speakers, Kazuo Iwano (Mitsubishi Corporation) and Michel Raynal (Institut Universitaire de France and IRISA, Université de Rennes 1).

On behalf of the Program Committee, we would like to thank all the authors who submitted their work to SSS. We sincerely acknowledge the tremendous time and effort the program track chairs and the Program Committee members

invested in the symposium. We are also grateful to the external reviewers for their valuable and insightful comments and to EasyChair for tremendously simplifying the review process and the generation of the proceedings. We also thank the Steering Committee members for their valuable advice and the Organizing Committee members for their time and effort to ensure a successful meeting. Finally, we greatly appreciate the support from the Graduate School of Information Science and Technology at Osaka University, and the Support Center for Advanced Telecommunications Technology Research (SCAT).

November 2013

Maria Gradinariu Potop-Butucaru
Teruo Higashino
Toshimitsu Masuzawa
Masafumi Yamashita

Organization

General Chair

Toshimitsu Masuzawa	Osaka University, Japan

Program Chairs

Maria Gradinariu Potop-Butucaru	University Pierre et Marie Curie (Paris 6), France
Masafumi Yamashita	Kyushu University, Japan
Teruo Higashino	Osaka University, Japan

Program Co-chairs

Taisuke Izumi	Nagoya Institute of Technology, Japan
Ralf Klasing	CNRS and University of Bordeaux, France
Sandeep Kulkarni	Michigan State University, USA
Zvi Lotker	Ben-Gurion University of the Negev, Israel
Achour Mostéfaoui	University of Nantes, France
Christian Scheideler	University of Paderborn, Germany
Oliver Theel	Carl von Ossietzky University of Oldenburg, Germany
Sébastien Tixeuil	UPMC Sorbonne Universites and IUF, France
Tatsuhiro Tsuchiya	Osaka University, Japan
Koichi Wada	Hosei University, Japan

Local Arrangements Chairs

Hirotsugu Kakugawa	Osaka University, Japan
Fukuhito Ooshita	Osaka University, Japan

Publication Chair

Yoshiaki Katayama	Nagoya Institute of Technology, Japan

Publicity Chairs

Doina Bein	Pennsylvania State University, USA
Francois Bonnet	JAIST, Japan

Sylvie Delaët	LRI, University of Paris-Sud 11, France
Taisuke Izumi	Nagoya Institute of Technology, Japan
Tomoko Izumi	Ritsumeikan University, Japan

Program Committee

Self-Stabilization

Chairs: Sébastien Tixeuil and Koichi Wada

Sylvie Delaët	LRI, University of Paris-Sud 11, France
Stéphane Devismes	University of Grenoble, France
Emmanuel Godard	Aix-Marseille University, France
Ted Herman	University of Iowa, USA
Taisuke Izumi	Nagoya Institute of Technology, Japan
Colette Johnen	University of Bordeaux, France
Sayaka Kamei	Hiroshima University, Japan
Jun Kiniwa	University of Hyogo, Japan
Mikhail Nesterenko	Kent State University, USA
Fukuhito Ooshita	Osaka University, Japan
Riko Jacob	ETH Zurich, Switzerland
Stefan Schmid	TU Berlin and Telekom Innovation Laboratories, Germany
Mordo Shalom	Tel-Hai College, Israel
Josef Widder	Technische Universität Wien, Austria
Yukiko Yamauchi	Kyushu University, Japan
Shmuel Zaks	Technion, Israel

Fault-Tolerance and Dependability

Chairs: Achour Mostéfaoui and Tatsuhiro Tsuchiya

Bernadette Charron-Bost	Ecole Polytechnique, France
Xavier Defago	JAIST and LIP6, UPMC, CNRS
Martin Hutle	Fraunhofer AISEC, Germany
Michiko Inoue	Nara Institute of Science and Technology, Japan
Kenichi Kourai	Kyushu Institute of Technology, Japan
Mikel Larrea	University of the Basque Country UPV/EHU, Spain
Fernando Pedone	University of Lugano, Switzerland
Lucia Penso	Universität Ulm, Germany
Luis Rodrigues	Universidade Técnica de Lisboa, INESC-ID, Portugal
Oliver Theel	Carl von Ossietzky University of Oldenburg, Germany
Roman Vitenberg	University of Oslo, Norway

Formal Methods and Distributed Systems

Chairs: Sandeep Kulkarni and Oliver Theel

Borzoo Bonakdarpour	University of Waterloo, Canada
Stéphane Devismes	University of Grenoble, France
Ali Ebnenasir	Michigan Technological University, USA
Vijay Garg	University of Texas at Austin, USA
Bernd Hauck	C1 WPS
Oday Jubran	Carl von Ossietzky University of Oldenburg, Germany
Sven Koehler	Tel Aviv University, Israel
Sayan Mitra	University of Illinois at Urbana Champain, USA
Achour Mostéfaoui	University of Nantes, France
Scott A. Smolka	Stony Brook University, USA
Volker Turau	Humburg University of Technology, Germany

Ad Hoc, Sensors, Mobile Agents, and Robot Networks

Chairs: Ralf Klasing and Zvi Lotker

Ittai Abraham	Microsoft Research Silicon Valley, USA
Keren Censor-Hillel	Technion, Israel
Colin Cooper	King's College London, UK
Jurek Czyzowicz	Université du Québec en Outaouais, Canada
Robert Elsässer	University of Salzburg, Austria
Thomas Erlebach	University of Leicester, UK
Pierre Fraigniaud	CNRS and University of Paris Diderot, France
Satoshi Fujita	Hiroshima University, Japan
Leszek Gąsieniec	University of Liverpool, UK
Sun-Yuan Hsieh	National Cheng Kung University, Taiwan, ROC
Adrian Kosowski	INRIA Bordeaux, France
Fabian Kuhn	University of Freiburg, Germany
Shay Kutten	Technion, Israel
Gopal Pandurangan	Nanyang Technological University and Brown University, Singapore/USA
Rajmohan Rajaraman	Northeastern University, USA

P2P, Social, Self-Organizing, Autonomic and Opportunistic Networks

Chairs: Taisuke Izumi and Christian Scheideler

Gregory Chockler	University of London, UK
Anwitaman Datta	Nanyang Technological University, Singapore

John Douceur	Microsoft Research
Pascal Felber	University of Neuchatel, Switzerland
Kalman Graffi	University of Düsseldorf, Germany
Qiang-Sheng Hua	Tsinghua University, China
Valerie King	University of Victoria, Canada
Lukasz Krzywieki	Wroclaw University of Technology, Poland
Laura Ricci	University of Pisa, Italy
Nicolas Schiper	Cornell University, USA
Siddhartha Sen	Princeton University, USA
Thorsten Strufe	TU Darmstadt, Germany
Hirozumi Yamaguchi	Osaka University, Japan

Steering Committee

Anish Arora	Ohio State University, USA
Ajoy K. Datta	University of Nevada, USA
Shlomi Dolev (Chair)	Ben-Gurion University of the Negev, Israel
Sukumar Ghosh	University of Iowa, USA
Mohamed Gouda	University of Texas at Austin, USA
Ted Herman	University of Iowa, USA
Toshimitsu Masuzawa	Osaka University, Japan
Vincent Villain	Université de Picardie Jules Verne (UPJV), France

Additional Reviewers

Altisen, Karine	Bridgman, John
Burman, Janna	Chauhan, Himanshu
Chen, Lin	Cournier, Alain
Datta, Ajoy K.	Daum, Sebastian
Duggirala, Parasara Sridhar	Faghih, Fathiyeh
Falcone, Ylies	Foreback, Dianne
Függer, Matthias	Gouda, Mohamed
Gu, Zhaoquan	Huang, Zhenqi
Hung, Wei-Lun	Izumi, Tomoko
Jubran, Oday	Konnov, Igor
Lafourcade, Pascal	Matsumae, Susumu
Niebert, Peter	Paes Leme, Renato
Petit, Franck	Rahaman Molla, Anisur
Robinson, Peter	Singh, Abhishek
Weiss, Stephane	Zemmari, Akka

Invited Papers

Tutorial on Vehicular Networking

Onur Altintas

Toyota InfoTechnology Center
6-6-20 Akasaka, Minato-ku, Tokyo Japan 107-0052

Abstract. Vehicular networking serves as one of the most important enabling technologies required to implement a myriad of applications related to vehicles, vehicle traffic, drivers, passengers and pedestrians. In this tutorial we will look into applications and use cases of vehicular networking with select examples from US, Europe and Japan. We will follow by looking into the requirements of applications ranging from safety to infotainment. Next we will cover some of the deployment plans and field tests around the world. System level approaches and a brief comparison of V2V, V2R, V2I communications will be given, followed by an overview of the standardization activities. We will provide a comparison of IEEE 802.11p/WAVE, ETSI (Europe) and Japan (ARIB) standards. Before concluding, we will take a glimpse at the recently emerging reality of electric vehicles and issues surrounding them. Finally we will conclude with open issues that require further research.

Practically Stabilizing and Secure Replicated State Machines
(Tutorial Abstract)

Shlomi Dolev

Department of Computer Science,
Ben-Gurion University of the Negev, Beer-Sheva, Israel
dolev@cs.bgu.ac.il

The tutorial focuses in two paradigms for reliable and secure distributed computation using multi-party computation. Replicated state machine and secure multi-party commutation. Recent results in self-stabilizing replicated state machine and in communicationless multi-party computation will be described.

Practically Stabilizing Replicated State Machine. Replicated state machines are used in practice to overcome faults in distributed systems. The Chubby, ZooKeeper (see e.g., [2]) that are used by Google and Yahoo are based on distributed implementation of replicated state machine. Other data centers also use the repeated consensus abstraction achieved by distributed algorithms such as Paxos [23]. Paxos ensures safety, namely, when a step is distributively selected (from a set of proposed steps) to be executed by a machine, all the other active machines will (eventually) execute this step too. Liveness is conditional (as [22] proved that asynchronous consensus does not exist) to the synchrony level encapsulated by the definition of an unreliable failure detection distributed algorithm [5]. The unreliable failure detector tries to exploit heuristics on the relative speed of responsiveness of machines to give (unreliable) hints on the machine that are suspected to be failed. The hints are used by a quorum of active machines to safely decide and proceed in implementing the common abstract state machine. The abstract state machine is distributively implemented by the machines using their replicas.

Chubby, ZooKeeper and in fact Paxos start in a consistent initial configuration and preserve consistency (in particular safety properties) by arguing that machines take actions according to the program (algorithm) and proving that these actions preserve consistency. Unfortunately, such a time-lined proof is very fragile in distributed systems, as unpredictable faults can temporarily cause the system configuration to be in inconsistent state, possibly caused by: accepting messages that are corrupted (while the error correcting code attached to them did not identify them as such), electricity spikes, single event upsets and crosstalks that flip the value of bits, and in fact any temporal violation of the assumptions made by the system designer [24, 7, 15, 6]. Self-stabilization is a property of systems, a property that ensures convergence to the desired behavior from any arbitrary configuration, where a configuration is described by a cartesian product of values, an arbitrary value to each variable in the system. The system

is proven to have an attractor, which is the desired behavior. Thus, the fragility of the proof thread of claims becomes robust, even when unexpected (illegal) set of actions are taken, the system converges to the desired behavior following the undesired actions. Note that there exist systems that tolerate Byzantine faults [4], where even malicious actions of a subset of the machines is tolerated, and in particular errors in their programs. Here too the system consistency is still preserved by the correctly acting machines, and the correctness proof is inductive, starting from a consistent state and preserving consistency when taking steps, rather than being, attracted from any configuration to converge (here in the presence of the Byzantine participants) to behave as desired [21]. Such systems are (still) too expensive to be widely used as the communication and complexity overhead is costly.

One important ingredient of the replicated state machine is the use of a sequence number for the steps. The sequence number is *practically unbounded* which encapsulate the fact that when the system starts with sequence number 0 it will take more than the life time of the system to exhaust the sequence number. Recently a line of works in the scope of self-stabilization, argue that it is sufficient to ensure that a self-stabilizing system converges to exhibit the desired behavior for such a practically unbounded executions as well. As compared with the original non-stabilizing specifications the system acts correctly "only" for practically infinite period (say "only" for one million years). In some sense the practically stabilizing notion can be viewed as an extension of the pseudo-stabilizing notion where the number of divergences from the desired behavior in an infinite execution is bounded, whereas in practically stabilizing systems the scope is almost or practically infinite execution rather than strictly infinite. In the pseudo stabilizing case the pigeon hole principle ensures an infinite execution with no divergence from the desired behavior, while in the case of practically stabilization, the pigeon hole principle ensures the existence of practically infinite execution in which the system does not diverge from the desired behavior [16, 1, 3]. In addition we have to recall that the assumption of fault-free infinite suffix in which the system stabilizes is only an abstraction, as the system should converge in every long enough fault-free period. Finally, self-stabilizing replicated state machines have been considered in different scopes using different techniques in [14, 10, 18, 20, 9, 8].

Communicationless Practically Unbounded Secure Multi Party Computation. Decomposing automaton into several automata such that the original automaton operations are encoded in the operations of the new automata is the approach that will be described. Now the settings is different as every machine receives (secret shares of) the same (streaming) inputs, and a secure computation should be carried by the machines, optimally without revealing any information to the participating machines. The motivation comes from cloud computing where a user would not like to reveal neither the data nor the processing to the cloud machines, while still using their storage and computation power.

Consider an input sequences that is (at least, practically) infinite. One challenge we have is to cope with a split of inputs, such that no information is revealed from the input received by a subset of the automaton portions. Another related challenge is to allow the adversary to record the state and inputs of a subset of the automaton portions for a finite or preferably infinite sequences. Additional challenge is to overcome corruptions in several automaton portions. At last proactive security issues are inherent consideration for systems with unbounded computation length.

The tutorial summarizes several recent works in which a dealer wants to delegate a computation to processes in the cloud by sending them a stream of inputs. The dealer is able to harvest the result by collecting the states of the processes at any given time, while processes have limited or no information concerning the current state of the computation. In particular the following solutions will be described:

- Reactive secret sharing, that changes the secret according to unbounded sequence of common inputs, where no communication among the (dynamic set of) participants is allowed, a fully secure solution for simple functions but somewhat non perfectly secure solution for any function [19].
- Dynamic online multiparty computation, in which a dynamic group of participants that should not know the sequence of inputs they process nor the program computed. The solution is based on a secret share based implementation of oblivious Turing machine [11].
- Infinite execution with no communication among the participants where the input is revealed to all participants. We prove that any automaton can be executed without revealing any information concerning the current state of the automaton. The construction is based on Krohn-Rhodes decomposition technique. Using pseudo random sequence, we present a simpler efficient technique for securing the current state of the automaton [12, 13].
- Computation of a class of automata and in particular automata for general string matching, in which both the inputs and the state are information theoretically secure [17].

References

1. Noga Alon, Hagit Attiya, Shlomi Dolev, Swan Dubois, Maria Potop-Butucaru, Sebastien Tixeuil, "Pragmatic Self-stabilization of Atomic Memory in Message-Passing Systems", *SSS*, pp. 19-31, 2011.
2. Flavio Paiva Junqueira and Benjamin Reed, "The life and times of a zookeeper", PODC, 2009.
3. Peva Blanchard, Shlomi Dolev, Joffroy Beauquier, Sylvie Delaet, "Self-Stabilizing Paxos", CoRR abs/1305.4263, 2013.
4. Miguel Castro, Barbara Liskov, "Practical byzantine fault tolerance and proactive recovery", *ACM Trans. Comput. Syst.* 20(4), 2002.
5. Tushar Deepak Chandra, Sam Toueg, "Unreliable Failure Detectors for Reliable Distributed Systems", *J. ACM*, 43(2), 1996.

6. Sylvie Delaet, Shlomi Dolev, Olivier Peres, "Safe and Eventually Safe: Comparing Self-stabilizing and Non-stabilizing Algorithms on a Common Ground", *OPODIS*, pp. 315-329, 2009. Also, "Safer Than Safe: On the Initial State of Self-stabilizing Systems", *SSS* 2009: 775-776.
7. Shlomi Dolev, *Self-Stabilization*, MIT Press 2000.
8. Shlomi Dolev, "Dynamic Multi-party Computation Forever for Swarm and Cloud Computing and Code Obfuscation", *ALGOSENSORS*, 2011.
9. Shlomi Dolev, Ori Gersten, "A framework for robust active super tier systems". *STTT* 12(1): 53-67, 2010.
10. Shlomi Dolev, Seth Gilbert, Limor Lahiani, Nancy Lynch, Tina Nolte, "Timed Virtual Stationary Automata for Mobile Networks", *9th International Conference on Principles of Distributed Systems* (OPODIS), December, 2005. Also Technical Report MIT-LCS-TR-979a, MIT CSAIL, Cambridge, MA 02139, 2005.
11. Shlomi Dolev, Juan Garay, Niv Gilboa, Vladimir Kolesnikov, "Swarming Secrets", *47th Annual Allerton Conference on Communication, Control, and Computing*, 2009. Also brief Announcement PODC pp. 231-232, 2010.
12. Shlomi Dolev, Juan A. Garay, Niv Gilboa, Vladimir Kolesnikov, "Secret Sharing Krohn-Rhodes: Private and Perennial Distributed Computation", *ICS* 2011.
13. Shlomi Dolev, Juan A. Garay, Niv Gilboa, Vladimir Kolesnikov, Yelena Yuditsky, "Towards Efficient Private Distributed Computation on Unbounded Input Streams,", pp. 69-83, ACNS 2013.
14. Shlomi Dolev, Seth Gilbert, Nancy A. Lynch, Alexander A. Shvartsman, Jennifer L. Welch, "GeoQuorums: implementing atomic memory in mobile ad hoc networks", *Distributed Computing*, 18(2), 2005.
15. Shlomi Dolev, Yinnon A. Haviv "Self-Stabilizing Microprocessor: Analyzing and Overcoming Soft Errors", *IEEE Trans. Computers* 55(4): 385-399, 2006.
16. Shlomi Dolev, Ronen I. Kat, Elad Michael Schiller, "When consensus meets self-stabilization", *J. Comput. Syst. Sci.* 76(8), 2010.
17. Shlomi Dolev, Niv Gilboa and Ximing Li, "Accumulating Automata and Cascaded Equations Automata, for Communicationless Information Theoretically Secure Multi-Party Computation Over (Practically) Unbounded Input Streams", submitted for publication, 2013.
18. Shlomi Dolev, Limor Lahiani, Nancy Lynch, and Tina Nolte, "Self-Stabilizing Mobile Location Management and Message Routing", *Proc. of the 7th International Symposium on Self-Stabilizing Systems*, (SSS 2005), LNCS 3764, pp. 96-112, 2005. Also Technical Report MIT-LCS-TR-999, Massachusetts Institute of Technology, 2005.
19. Shlomi Dolev, Limor Lahiani, Moti Yung, "Secret swarm unit: Reactive k-secret sharing", *Ad Hoc Networks* 10(7), 2012.
20. Shlomi Dolev, Elad Schiller, Jennifer L. Welch, "Random Walk for Self-Stabilizing Group Communication in Ad Hoc Networks", *IEEE TMC* 5(7), 2006.
21. Shlomi Dolev, Jennifer L. Welch, "Self-stabilizing clock synchronization in the presence of Byzantine faults", *JACM* 51(5), 2004.
22. Michael J. Fischer, Nancy Lynch and Mike Paterson, "Impossibility of Distributed Consensus with One Faulty Process", *J. ACM*, 32:2, pp. 374-382, 1985.
23. Leslie Lamport, "The Part-Time Parliament", *ACM TOCS*, 16 (2), 1998.
24. Eric C. Rosen. "Vulnerabilities of network control protocols: an example", *SIGCOMM Comput. Commun. Rev.*, 11(3):10 16, July 1981.
25. Rodrigo Rodrigues, Barbara Liskov, Kathryn Chen, Moses Liskov, David A. Schultz, "Automatic Reconfiguration for Large-Scale Reliable Storage Systems", *IEEE TDSC* 9(2): 145-158, 2012.

Concurrency-Related Distributed Recursion

Michel Raynal

*Institut Universitaire de France & †IRISA, Université de Rennes 1 (France)
raynal@irisa.fr

Recursion. Recursion is a powerful algorithmic technique that consists in solving a problem of some size (where the size of the problem is measured by the number of its input data) by reducing it to problems of smaller size, and proceeding the same way until we arrive at basic problems that can be solved directly. This algorithmic strategy is often capture by the Latin terms "*divide ut imperes*".

Recursive algorithms are often simple and elegant. Moreover, they favor invariant-based reasoning, and their time complexity can be naturally captured by recurrence equations. In a few words, recursion is a fundamental concept addressed in all textbooks devoted to sequential programming (e.g., [5, 7, 10] to cite a few). It is also important to say that, among the strong associations linking data structures and control structures, recursion is particularly well suited to trees and more generally to graph traversal [5].

Recursive algorithms are also used since a long time in parallel programming. In this case, parallel recursive algorithms are mainly extensions of sequential recursive algorithms, which exploit data independence. Simple examples of such algorithms are the parallel versions of the quicksort and mergesort sequential algorithms.

Recursion and distributed computing. In the domain of distributed computing, the first (to our knowledge) recursive algorithm that has been proposed is the algorithm solving the Byzantine general problem [9]. This algorithm is a message-passing synchronous algorithm. Its formulation is relatively simple and elegant, but it took time to understand its deep nature.

Similarly to parallelism, recursion has been used in distributed algorithms to exploit data independence or provide time-efficient implementations of data structures. As an example, the distributed implementation of a store-collect object described in [2] uses a recursive algorithm to obtain an efficient tree traversal, which provides an efficient adaptive distributed implementation.

Capture the essence of distributed computing. The aim of real-time computing is to ensure that no deadline is missed, while the aim of parallelism is to allow applications to be efficient (crucial issues in parallel computing are related to job partitioning –flow graphs– and scheduling). Differently, when considering distributed computing, the main issue lies in mastering the uncertainty created by the multiplicity and the geographical dispersion of computing entities, their asynchrony and the possibility of failures.

At some abstract level and from a "fundamentalist" point of view, such a distributed context is captured by the notion of a task, namely, the definition of a distributed computing unit which capture the essence of distributed computing [8]. Tasks are the distributed counterpart of mathematical functions encountered in sequential computing (where some of them are computable while others are not).

The talk: recursive algorithms for computable tasks. This invited talk is on the design of recursive algorithms that compute tasks. A seminal related work can be found in [6]. It appears that, for each process participating to a task, the recursion parameter x is not related to the size of a data structure but to the number of processes that the invoking process perceives as participating to the task computation. In a very interesting way, it follows from this feature that it is possible to design a general recursion pattern, which can be appropriately instantiated for particular tasks.

When designing such a pattern, the main technical difficulty come from the fact that processes may run concurrently, and, at any time, distinct processes can be executing at the same recursion level or at different recursion levels. To cope with such an issue, recursion relies on an underlying data structure (basically, an array of atomic read/write registers) which keeps the current state of each recursion level.

After having introduced the general recursion pattern, the talk will instantiate it to solve two tasks, namely, the write-snapshot task [3] and the renaming task [1, 4]. Interestingly, the first instantiation of the pattern is based on a notion of linear time (there is single sequence of recursive calls, and each participating process executes a prefix of it), while the second instantiation is based on a notion of branching time (when considering the recursion tree, a process executes a prefix of a single branch, while the whole set of branches captures all the possible process execution paths).

In addition to its methodological dimension related to the new use of recursion in a distributed setting, the talk has a pedagogical flavor in the sense that it focuses on and explains fundamental notions of distributed computing. Hence, an aim of this talk is to provide the reader with a better view of the nature of fault-tolerant distributed recursion when the processes are concurrent, asynchronous, prone to crash failures, and communicate through read/write registers.

Where to find the technical content. The technical content of this invited talk can be found in [12], and in Chapters 8 and 9 of [13]. A topological perspective of distributed recursion can be found in [11].

References

1. Attiya H., Bar-Noy A., Dolev D., Peleg D. and Reischuk R., Renaming in an asynchronous environment. *Journal of the ACM*, 37(3):524-548, 1990.
2. Attiya H., Fouren A., and Gafni E., An adaptive collect algorithm with applications. *Distributed Computing*, 15(2): 87-96, 2002.

3. Borowsky E. and Gafni E., Immediate atomic snapshots and fast renaming. *Proc. 12th ACM Symposium on Principles of Distributed Computing (PODC'93)*, pp. 41-51, 1993.
4. Castañeda, Rajsbaum S., and Raynal M., The renaming problem in shared memory systems: An introduction. *Computer Science Review*, 5(3):229-251, 2011.
5. Dahl O.J., Dijkstra E.W., and Hoare C.A.R., *Structured programming*. Academic Press, 220 pages, 1972 (ISBN 0-12-200550-3).
6. Gafni E. and Rajsbaum S., Recursion in distributed computing. *Proc. 12th Int'l l Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS '10)*, Springer LNCS 6366, pp. 362-376, 2010.
7. Harel D. and Feldman Y., *Algorithmics: the spirit of computing* (third edition). Springer, 572 pages, 2012 (ISBN 978-3-642-27265-3).
8. Herlihy M.P., Rajsbaum S., and Raynal M., Power and limits of distributed computing shared memory models. To appear *Theoretical Computer Science*, 2013/2014.
(http://dx.doi.org/10.1016/j.tcs.2013.03.002),
9. Lamport L., Shostak E., and Pease M.C., The Byzantine general problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382-401, 1982.
10. Mehlhorn K. and Sanders P., *Algorithms and data structures*. Springer, 300 pages, 2008 (ISBN 978-3-540-77977-3).
11. Onofre J.-C., Rajsbaum S., and Raynal M., A topological perspective of recursion in distributed computing. *Mexican Conference on Discrete Mathematics and Computational Geometry*, Oaxaca (Mexico), November 2013.
12. Rajsbaum S. and Raynal M., An introductory tutorial to concurrency-related distributed recursion. *Research report #2006*, IRISA, Université de Rennes (France), to appear in *Electronic Bulletin of EATCS*, 20 pages (October 2013/February 2014).
13. Raynal M., *Concurrent programming: algorithms, principles and foundations*. Springer, 515 pages, 2013 (ISBN 978-3-642-32026-2).

Table of Contents

Dependability and Fault-tolerance

Transactional Encoding for Tolerating Transient Hardware Errors 1
Jons-Tobias Wamhoff, Mario Schwalbe, Rasha Faqeh, Christof Fetzer, and Pascal Felber

Universal Model Simulation: BG and Extended BG as Examples 17
Petr Kuznetsov

Helical Entanglement Codes: An Efficient Approach for Designing Robust Distributed Storage Systems 32
Verónica Estrada Galiñanes and Pascal Felber

Concurrent Wait-Free Red Black Trees 45
Aravind Natarajan, Lee H. Savoie, and Neeraj Mittal

Self-Stabilization I

Self-stabilizing *(f,g)*-Alliances with Safe Convergence 61
Fabienne Carrier, Ajoy K. Datta, Stéphane Devismes, Lawrence L. Larmore, and Yvan Rivierre

A Self-stabilizing Algorithm for Maximal p-Star Decomposition of General Graphs 74
Brahim Neggazi, Volker Turau, Mohammed Haddad, and Hamamache Kheddouci

Space Complexity of Self-Stabilizing Leader Election in Population Protocol Based on k-Interaction 86
Xiaoguang Xu, Yukiko Yamauchi, Shuji Kijima, and Masafumi Yamashita

Self-Healing of Byzantine Faults 98
Jeffrey Knockel, George Saad, and Jared Saia

Leader Election and Centers and Medians in Tree Networks 113
Ajoy K. Datta and Lawrence L. Larmore

Formal Methods and Distributed Systems

Local Decision and Verification with Bounded-Size Outputs 133
Heger Arfaoui, Pierre Fraigniaud, and Andrzej Pelc

How Good is Weak-Stabilization? 148
Narges Fallahi and Borzoo Bonakdarpour

Verifying Livelock Freedom on Parameterized Rings and Chains 163
Alex Klinkhamer and Ali Ebnenasir

Certified Impossibility Results for Byzantine-Tolerant Mobile Robots . . . 178
Cédric Auger, Zohir Bouzid, Pierre Courtieu, Sébastien Tixeuil, and Xavier Urbain

P2P, Social, Self-Organizing, Autonomic and Opportunistic Network

Self-stabilizing Balancing Algorithm for Containment-Based Trees 191
Evangelos Bampas, Anissa Lamani, Franck Petit, and Mathieu Valero

On the Effectiveness of Punishments in a Repeated Epidemic Dissemination Game 206
Xavier Vilaça and Luís Rodrigues

Linearizing Peer-to-Peer Systems with Oracles 221
Rizal Mohd Nor, Mikhail Nesterenko, and Sébastien Tixeuil

Self-Stabilization II

Synchronous Counting and Computational Algorithm Design 237
Danny Dolev, Janne H. Korhonen, Christoph Lenzen, Joel Rybicki, and Jukka Suomela

An Asynchronous Self-stabilizing Approximation for the Minimum Connected Dominating Set with Safe Convergence in Unit Disk Graphs 251
Sayaka Kamei, Tomoko Izumi, and Yukiko Yamauchi

Automated Addition of Fault-Tolerance under Synchronous Semantics 266
Yiyan Lin, Borzoo Bonakdarpour, and Sandeep Kulkarni

Ad-hoc, Sensors, Mobile Agents and Robot Networks

Naming and Counting in Anonymous Unknown Dynamic Networks 281
Othon Michail, Ioannis Chatzigiannakis, and Paul G. Spirakis

Gathering Asynchronous Oblivious Agents with Restricted Vision in an Infinite Line 296
Samuel Guilbault and Andrzej Pelc

Counting the Number of Homonyms in Dynamic Networks 311
G.A. Di Luna, R. Baldoni, S. Bonomi, and Ioannis Chatzigiannakis

Localizability of Wireless Sensor Networks: Beyond Wheel Extension ... 326
Buddhadeb Sau and Krishnendu Mukhopadhyaya

Brief Announcement I and II

Memory Efficient Self-Stabilizing k-Independent Dominating Set Construction 341
Colette Johnen

Modeling and Analyzing Timing Faults in Transaction Level SystemC Programs 344
Reza Hajisheykhi, Ali Ebnenasir, and Sandeep S. Kulkarni

Low-Communication Self-stabilizing Leader Election in Large Networks 348
Thamer Alsulaiman, Andrew Berns, and Sukumar Ghosh

Self-stabilizing Byzantine Resilient Topology Discovery and Message Delivery 351
Shlomi Dolev, Omri Liba, and Elad M. Schiller

Self-stabilizing TDMA Algorithms for Wireless Ad-Hoc Networks without External Reference 354
Thomas Petig, Elad M. Schiller, and Philippas Tsigas

Zone-Based Synthesis of Strict 2-Phase Fault Recovery 357
Fathiyeh Faghih and Borzoo Bonakdarpour

Analyzing Convergence and Reachability of Asynchronous Iterations.... 360
Yoshisato Sakai

Ring Exploration by Oblivious Robots with Vision Limited to 2 or 3 363
Ajoy K. Datta, Anissa Lamani, Lawrence L. Larmore, and Franck Petit

Scalable Estimation of Network Average Degree..... 367
Taisuke Izumi and Hironobu Kanzaki

Synthesizing Round Based Fault-Tolerant Programs Using Genetic Programming..... 370
Ling Zhu and Sandeep Kulkarni

Self-stabilizing DAG-Constructing Protocols with Application to Geocast in MANET 373
Koichi Ito, Yoshiaki Katayama, Koichi Wada, and Naohisa Takahashi

An Agile and Stable Neighborhood Protocol for WSNs 376
Gerry Siegemund, Volker Turau, Christoph Weyer, Stefan Lohs, and Jörg Nolte

Author Index 379