English abstract of produced documents

in order to obtain the HdR

July 2007

COLETTE JOHNEN

In France, a researcher has to defend its HdR ("Licence to supervise Research") to become a full professor.

The HdR (abbreviation for "Licence to supervise Research") sanctions the recognition of the high scientific level of the candidate, the original character of its initiative in a domain of the science, its capacity to master a research strategy in a wide enough scientific or technological domain its capacity to supervise young researchers.

Curriculum Vitae of COLETTE JOHNEN

colette@lri.fr
www.lri.fr/∼colette

CURRENT POSITION

Since 1st Decembre 1988, I am Associated Professor at Computer Science department of University Paris-Sud 11, LRI.
I am member of Parallelism group of « Laboratoire de Recherche en Informatique » LRI, UMR 8623.

EDUCATION at University Paris-Sud 11

- *PhD*, defense in Decembre 1987

  Title of dissertation : « Algorithmic Analysis of Petri Nets : home space, rewriting systems » Grade : very honorable.
  PhD advisor : G. Berthelot. Referees : D. Frutos Escrig, G. Memmi.
  Examiners : G. Vidal-Naquet (chairman), C. Choppy.

- Master in Computer Science in 1984 at Univ. Paris-Sud *master dissertation :* Parallelism and rewriting system.

RESEARCH TOPICS

Since 1995, my main research topic is the design of distributed algorithms, more precisely the designed of self-stabilizing algorithms. The concept of self-stabilizing algorithms was introduced by Dijkstra in 1973. A system is self-stabilizing when regardless of its initial configuration, it eventually reach a legitimate configuration. Self-stabilizing protocols are thus, inherently tolerant to transient faults of the system. My specific research topics are :

- space efficient deterministic self-stabilizing token circulation algorithms [25], [22], [21], and [05].
- space efficient deterministic self-stabilizing construction of Breath First Search spanning trees [32], and [23].
- space optimal deterministic self-stabilizing leader election algorithm on rings [16].
- optimal self-stabilizing token circulation algorithms on unidirectional anonymous rings [14], and [12].

- self-stabilizing leader election algorithms on anonymous networks [20], [04], and [01].
- theoretical tools to assist the design and the proving of self-stabilizing algorithm [19], [18], [17], [15], and [03].
- self-stabilizing algorithms guaranteeing a *degraded* service during the stabilization phase (i.e. during the convergence of the system toward a legitimate configuration from an uncorrect configuration [13], [10], [07], and [06].
- elaboration of proof techniques and analysis techniques for probabilistic self-stabilizing algorithms [44], [31], and [08].
- comparing the various communication models for networks of processor based on registers [43], [30], and [11].

Definition of downloading strategies in Peer to Peer system (such that BitTorrent) that cope with rude users and that tolerate some faulty segments [09], and [02].

PhD STUDENTS

- *2004 - up to now :* Advisor for 80% of Le Huy Nguyen's PhD thesis (Dr. Pierre Fraigniaud is the co-advisor). Le Huy Nguyen will defense its PhD at the end of september 2007.
- *1997 - 2000 :* Advisor for 70% of Maria Gradinariu PhD thesis (with Prof. Joffroy Beauquier was the co-advisor). Now, Maria Gradinariu is Associated Professor at University Pierre et Marie Curie, Paris 6.

PROGRAM COMMITTEES

- *2003 and 2004 :* member of program committee of **OPODIS'03** and of **OPODIS'04**, the 7th and the 8th International Conference On Principles Of DIstributed Systems.
- *2003 :* member of program committee of **DISC'03**, the 17th International Symposium on Distributed Computing.
- *2003 :* member of program committee of **SSS'03**, the 6th Symposium on Self-Stabilizing Systems.

SUPERVISED MASTER THESIS

- *2007 :* Minh Tuan Ho (co-supervised with Le Huy Nguyen), title of the master thesis : « Simulation and evaluation of the robust self-stabilizing clustering algorithm [07] with NS-2 »
- *2003 :* Frédéric Majorczyk, title of the master thesis : « Study and simulation of scheduling

algorithms on Large Scale Distributed Systems »

- *2003 :* Samir Sebbah, title of the master thesis : « Simulation and evaluation of the self-stabilizing routing protocol [13] with NS-2 »
- *2002 :* Fayçal Khebizat, title of the master thesis : « conception and proof of a self-stabilizing algorithm building minimum weight spanning tree »
- *1998 :* Alain Fontaine, title of the master thesis : « routing transparent to host mobilities »

VISITING POSITIONS

- *July 2006 :* invited at Calgary university, Alberta, Canada, to work with Prof. Lisa Higham.
- *July 2005 :* invited at Calgary university, Alberta, Canada, to work with Prof. Lisa Higham.
- *May 2001 :* invited at Toronto university, Ontario, Canada, to work with Prof. Faith Fich.
- *1991-1992 :* invited researcher in Distributed Computing and Communications Laboratory (DCC lab) at University of Columbia, New-York, managed by Prof Yechiam Yemini.

INTERNATIONAL SEMINARS

- *May 2007 :* presentation of my work during « COST 295 - 1st Workshop on Dynamic Networks », Salerno, Italy
- *october 2004 :* invited to participate at the « 5-day Workshops Self-Stabilizing Distributed Systems » located in BIRS (Banff International Research Station), Banff, Canada.
- *1998 and 2000 :* invited to participate at the « self-stabilizing system seminar » at Schloss Dagstuhl (International Conference and Research Center for Computer Science), Germany, in octobre 1998 and in octobre 2000.

WORKSHOP ORGANIZATION

- *2002 :* co-organization with Prof. Joffroy Beauquier (LRI, Orsay) of seminar « Self-Stabilizing Systems ». This 5-day seminar took place at CIRM (Centre International de Rencontre Mathématique), Luminy, France from the 21th to the 25th October 2002. Forty researchers participated to this seminar.

PARTICIPATION to PhD Thesis examining committees

- *2005 :* member of examining committee of PhD of Joyce El Haddad, January 2005, her PhD

advisor was Prof. Serge Haddad, Univ. Paris Dauphine, Paris, France.

- *2005 :* member of examining committee of PhD of Gabriel Antoine Louis Paillard, June 2005, his PhD advisor was Prof. Christian Lavault, Univ. Paris-Nord, Paris, France.

COORDINATIONS of SCIENTIFIC PROJECTS

- *2002 :* **AIRCAST**, Platform to explore issues about volatile hosts in wireless networks. The project was supported by the University Paris-Sud 11 for 15K€.
- *2002-2003 :* **STAR**, stabilization of computer networks using Internet standards. This project was financed by of STIC department of CNRS, as an Action « ATIP young researchers » for 25 K€.

PARTICIPATIONS to SCIENTIFIC PROJECTS

- *2004-2008 :* **FRAGILE**, Failure resilience and application guaranteed integrity in Large-scale Environnements, project of ACI « security and dependability ».
- *2003-2004 :* **Distributed Algorithms and Applications**, financed by STIC department of CNRS.
- *2002-2003 :***DYNAMO**, structural analysis and algorithmic for dynamic networks, financed by STIC department of CNRS.
- *1998-2000 :* **COMMOBIL**, routing and handover for mobile networks, financed by INRIA
- *1997-1998 :* **OCReM** - Optimization of communication overheads in mobile networks ; The project was supported by the University Paris-Sud 11.
- *1984-1985 :* I participated at ESPRIT European Union project « High level for Interactive Layout and Design » [33].

REVIEWS

For instance, since August 2005 to August 2006, I reviewed 5 papers for the following revues : Algorithmica, Journal of Parallel and Distributed Computing, Parallel Processing Letter, IEEE Transactions on Computers, Journal of Aerospace Computing, Information, and Communication. I also reviewed 6 papers submitted to the conferences OPODIS, ICDCS, DISC, or SSS.

- I do regular reviews for the following conferences : ACM Symposium on Principles of Distributed Computing (PODC), IEEE International Conference on Distributed Computing Systems (ICDCS), International Symposium on DIStributed Computing (DISC), IEEE International

Parallel and Distributed Processing Symposium (IPDPS), International Conference on Parallel and Distributed Systems (ICPADS), International Conference On Principles Of DIstributed Systems (OPODIS) et Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS).

- I do regular reviews for the following journals : Algorithmica, ed. Springer, Distributed Computing (DC), ed. Springer-Verlag Information Processing Letters (IPL), ed. Elsevier Journal of Parallel and Distributed Computing (JPDC), ed. Elsevier Parallel Processing Letter (PPL), ed. World Scientific Publishing IEEE Transactions on Parallel and Distributed Systems (TPDS), ed. IEEE Computer Society Press. IEEE Transactions on Computers (TC),ed. IEEE Computer Society Press.

SOFTWARE DEVELOPMENTS

- *1991-1992 :* I did participate at *Netmate* - Network Management Analysis and Testing Environment project [34]. Netmate software was developed and commercialized by SMARTS, System Management Arts Inc. has been acquired by EMC in 2005.
- *1984-1989 :* Software development of **Petrireve**, an analysis environement for Petri nets using rewriting technique [36].
  I integrated Petrireve to **Serpe**, an extensible structure for analysis of Petri nets. Moreover, I implemented an analyser of covering graph that was included at Serpe [35].
  Finally, I integrated the Serpe tools at **PAPETRI**, an environment for editing and analysing Petri nets. PAPETRI allows to work with several classes of nets [27]. Papetri was developed in collaboration with CEDRIC-CNAM lab. (Prof. Gérard Berthelot, University Evry, Val d'Essonne and Prof. Laure Petrucci, University Paris-Nord 13).

SKELETON of my « Licence to supervise research » THESIS

This document is not a translation of my HdR dissertation in English.

It is only an overview of the HdR dissertation, focused on my research results that are presented in detail in the HdR dissertation.

My HdR dissertation contains numerous references to related works, that I did omit in this document.

Introduction

A system distributed is a set of autonomous and communicating entities. Each entity (also called node, process, processor, machine...) has its own code, its own objectives, its execution speed ... . These entities exchange between them pieces of information. A major element of a system distributed is the communication graph (also called interconnection network, more simply network) which defines for each entity to which it can communicate. The objective of a distributed system is to provide services which could not be carried out by only one entity in term of functionality, availability, response time, or reliability ... .

The systems distributed are particularly prone to failures. Because each entity is likely to fail or to be erroneous. The traditional taxonomy of failure is :

- crash faults : either an element normally functions, or it does not do anything (it is broken down).
- intermittent failures : an element stops functioning for one arbitrary time period, then operates correctly. This type of failures corresponds to the message losing, the rupture of communication links.
- Byzantine faults or attacks : an element of the system has a un-standard behavior. This type of faults corresponds to existing bugs in the design and in the realization of the system. But also, the insertion of a « Trojan horse »in the system to disturb its functioning.

The distributed systems are also prone to the modifications of the interconnection graph : the entities can be nomad or mobile, communication links can be added or removed, and sub-networks can be connected or disconnected.

In short, a system distributed must be able to manage events of various natures which disturb its operation. It is thus important to have distributed algorithms adapted to a changing environment. There are two classes of algorithms tolerating disturbances or system failures.

- A masking algorithm ensures the continuity of the functionalities even in the presence of disturbances. This type of algorithms tolerates failures striking the system continuously. It is a very important guarantee which is expensive and complex to implement. Such a guarantee is proposed and satisfied only for critical systems whose dysfunctions can endanger human lifes (for example, control of the planes, of the nuclear power stations, ...). Usually, these algorithms tolerate only a restricted class of failures.
- An algorithm un-masking does not guarantee the continuity of the service, but it guarantees the resumption of the service after the disturbances end. This type of algorithms tolerate all kinds of disturbances during a finite time period. After the end of the gust of disturbances, the system acts correctly after a transitional phase (phase where the system does not function correctly, although there are no more disturbances).

E.W. Dijkstra [Dij74][1] defined the concept of self-stabilization as : *A system is self-stabilizing when regardless of its initial configuration, it eventually reach a legitimate configuration.* The self-stabilizing algorithms are un-masking; they tolerate an arbitrary series of transitory failures during a time period. After this series, a self-stabilizing algorithm modifies the system to reach a legitimate configuration. Once a configuration legitimate is reached, the system functions correctly, it carries out its tasks normally.

In HdR dissertation, I present a part of my research works concerning the self-stabilizing algorithms. I voluntarily limited myself to work concerning the three following topic :

- Theoretical study of the models of communication. In the chapter 1, the various models of communication per registers are presented and compared. To compare the power of these models, I study the realization of tolerant converters to failures from one model to another.
- Contribution to classic distributed problems. Two benchmark problems : the leader election and mutual exclusion are intensively studied in the chapter 2. I was interested in the memory capacity necessary to these two tasks on anonymous rings.
- Algorithms for the Ad hoc networks. In the chapter 3, the failure tolerant algorithms to manage Ad-Hoc networks are presented.

In the continuity of the three research topics presented in the dissertation, open research problems are proposed.

[1] [Dij74] : E.W. Dijkstra. Self stabilizing systems in spite of distributed control. *Communications of the Association of the Computing Machinery*, 17 :643–644, 1974.

Chapter 1 : Fault-tolerant Converters

To design and especially to prove the correctness of an algorithm distributed, it is of primary importance to define the model of communication set up between the processes : communication by messages, communication by registers. To establish that the communication is done via shared registers is not sufficient, it is also necessary to define semantics associated with the registers, the register types. I present in this chapter at the various models of communication per registers.

The objective of this chapter, is to compare these various models by presenting fault-tolerant converters of a model with other one, or by proving impossibility of building of such converter.

Chapter 1 : presented results

A network can be modeled by a graph where nodes represent processors and there is an edge between two nodes if and only if the corresponding processors communicate directly by reading or writing registers shared between them. Two variants are defined by specifying whether the registers are single-writer/multi-reader and located at the nodes (called state models) or single-writer/single-reader and located on the edges (called link models).
In the state network models, each processor p owns a single-writer multi-reader register, which is writable by p and readable by each of p's neighbours.
In the link network models, for each peer of neighbours (p, q), there are two single-writer single-reader registers : one is writable by p and readable by q ; the other ones is writable by q and readable by p.

The semantics of the shared registers used by the communicating processors further distinguishes possible models. L. Lamport [Lam86][2] defined three models of single-writer registers, differentiated by the possible outcome of read operations that overlap write operations. These three register types, in order of increasing power, are called safe, regular, and atomic. Program design is easier assuming atomic registers rather than regular or safe registers but the hardware implementation of an atomic register is costlier than the implementation of a weaker register.
A register is safe if each READ that does not overlap any WRITE returns the value of the latest WRITE that happens-before it, and otherwise returns any value.
A register is regular if it is safe and any READ that overlaps a WRITE returns the value of either the latest WRITE that happens-before it, or the value of some overlapping WRITE.
A register is atomic if it is regular, and if after that a READ, r, has returned the value written by w, then any subsequent READ, must return the value written by w or by any subsequent WRITE.

By specifying either state or link communication, via shared registers that are either safe, regular, or atomic, we arrive at six different network models that use locally shared registers. For example, the atomic-state model has atomic registers located at the nodes. The other models are named similarly.

[2][Lam86] : L. Lamport. On interprocess communication. *Distributed Computing*, 1(2) :77–101, 1986

Another parameter models is the fault-tolerance of the network. I consider wait-freedom, which captures tolerance of stopping failures of components of the network, or self-stabilization, which captures recovery of the network from transient errors of its components.

In [11], it is established that there is no general converter from atomic-state networks to atomic-link networks that preserves wait-freedom. The proof proceeds by showing that any such converter would require shared registers between *any* two processors ; that is, the graph modelling the network communication must be a complete graph. A consequence of this impossibility result is that there is no wait-free converter from atomic-state network model to regular-state model.

In [11], we also present a self-stabilizing converter from networks where neighbours communicate via atomic-state registers to networks where communications are done via atomic-link registers. S. Dolev, A. Israeli, and S. Moran. [DIM93][3] introduced the read/write atomicity model for self-stabilizing algorithms to capture the actual possible communication between processors. In this model, communications are done via atomic-link registers.

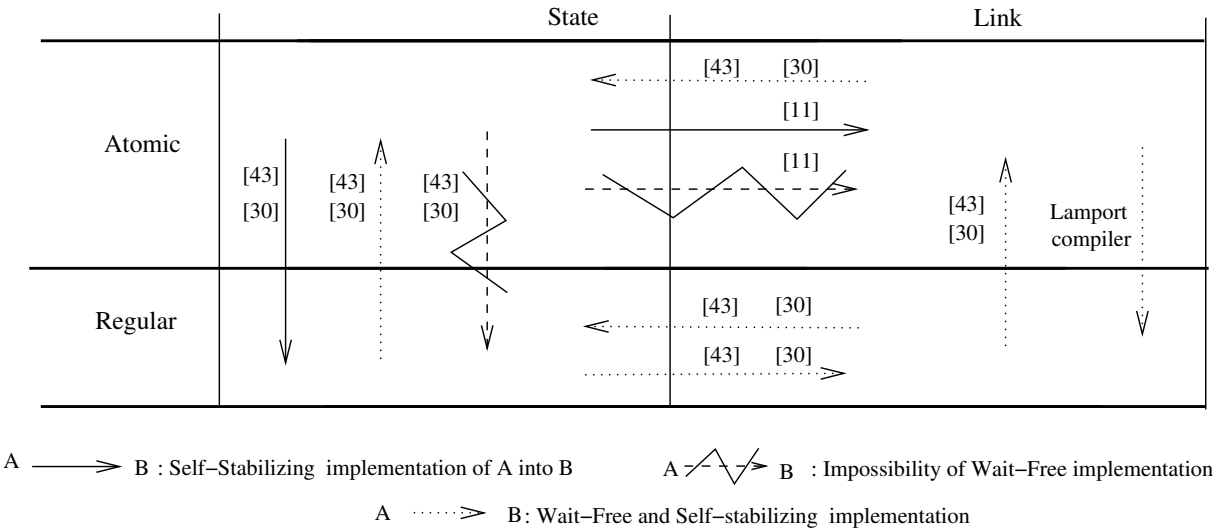

FIG. 1 – Obtained results about Fault-Tolerant converters

In [30] and [43], we present a self-stabilizing algorithm that converts any self-stabilizing algorithm for the atomic-state model to a self-stabilizing algorithm for the same problem in the regular-state model. This converter is also *silent*. That is, if, once registers have stabilized, the atomic-state algorithm does not require the participation of neighbours, then the transformed regular-state algorithm also does not require the participation of neighbours. As a consequence, our converter does not add significant overhead to communication.

[3][DIM93] : S. Dolev, A. Israeli, and S. Moran. Self-stabilization of dynamic systems assuming only Read/Write atomicity. *Distributed Computing*, 7(1) :3–16, 1993.

For all the remaining relationships (both possibilities and impossibilities) among the four models that use atomic and regular registers under either self-stabilizing and wait-free requirements, we either observe that they have been answered by existing research or by my works. These results are summarized in the figure 1.

Chapter 2 : About the memory space required by self-stabilizing algorithms

E. Dijkstra, in its seminal paper of 1974, has introduced the self-stabilizing concept by the presentation of self-stabilizing token circulation algorithms on rings.

This paper has driven lot of research works on self-stabilizing token circulation and leader election on anonymous rings. In this chapter, I present my contribution to this research topic.

Chapter 2 : presented results

Deterministic algorithms. Let us study the problem of deterministic self-stabilizing leader election on anonymous rings. It is part of the common knowledge [20] that, on anonymous rings, there is no deterministic self-stabilizing leader election or token circulation algorithm (1) under a distributed daemon, or (2) on ring having a composite size (i.e. the ring size is not prime number). Only on prime-size rings, under a centralized daemon, one may design a deterministic self-stabilizing leader election algorithm or a deterministic self-stabilizing token circulation algorithm.

The question addressed here is to determine the amount of memory space needed by a deterministic self-stabilizing leader election or token circulation algorithm on unidirectional prime size rings, under a centralized daemon. We are not claiming that this particular question is important because of applications (a prime-size ring is not a realistic assumption). Rather we seek better insight into the nature of self-stabilization on unidirectional networks.

In [20], a lower bound is proven : N states per processor for any deterministic self-stabilizing leader election algorithm on unidirectional rings of size N. The application of the proof technique to self-stabilizing token circulation allows us to obtain a lower bound of $(N-1)/2$ states per processor.

In [16], a deterministic self-stabilizing leader election algorithm for unidirectional rings is presented in which the number of states is $10N$. Hence, [16] algorithm matches the number of bits of storage used at each processor to within a small additive constant of the number required by the lower bound.

M.G. Gouda and F.F. Haddix, in [GH96][4], have proposed a deterministic self-stabilizing token circulation using 8 states per processor for unidirectional rings with a leader. The fair composition of [GH96] and [16] algorithms provides a deterministic self-stabilizing token circulation on anonymous rings. The obtained algorithm requires $80(N)$ states per processor (or $7+\lg(N)$ bits). [16] algorithm combined with algorithm of [GH96] matches the number of bits of storage used at each processor to within a small additive constant of the number required by the lower bound.

Probabilistic algorithm. The number of states per processor required by a probabilistic self-stabilizing leader election algorithm under the unfair distributed scheduler on anonymous and

[4][GH96] : M.G. Gouda and F.F. Haddix. The stabilizing token ring in three bits. *Journal of Parallel and Distributed Computing*, 35(1) :43–48, 1996

unidirectional rings is greater than $(m_N - 2)/2$, it was proven in [20]. Thus each processor needs to allocate $\ln(m_N - 2) - 1$ bits to the functioning algorithm. m_N is the smallest integer not dividing the ring size N. Notice that the value of m_N is constant on average.

J. Beauquier, A. Cordier, and S. Delaët, in [BCD95][5] have proposed a probabilistic self-stabilizing token circulation algorithm on anonymous rings. Algorithm of [BCD95] is suitable for any ring size. In [BCD95], it is proven this algorithm converges under the fair and memory k-bounded scheduler. A scheduler is memory k-bounded, if its choices depends only on the last k previous computation step. The scheduler acts as if it has a short-term memory limited to k last recent computation steps.

In [01], it has been proven that algorithm of [BGD95] converges under k-bounded schedules. A k-bounded scheduler produces only k-bounded computations. A computation is k-bounded [20] if along any sequence where a processor p is continuously enabled, any other processor q performs at most k actions before p performs an action. In [08], we has established that k-bounded scheduler is more powerful than the fair and memory k-bounded scheduler, because (1) any computation produced by the fair and memory k-bounded scheduler can be produced by the k-bounded scheduler, and (2) the converse is not true.

A self-stabilizing token circulation under the unfair distributed scheduler is presented in [20]. The memory space needed on each processor by the algorithm of [20] is $2\lg(m_N)$. The cross-over composition, presented in [15], and the algorithm of [BCD95] was used to establish the algorithm of [20].

In algorithm of [20], the technique used to discard multiple tokens is « to randomly retard the token circulation ». A processor having a token randomly decides to pass or not the token. The drawback of this technique is the service time. *Service time* is the maximal time in term of computation steps required by the algorithm to perform a token circulation, once the system is stabilized. Once the ring is stabilized (i.e. there is only one token in the ring), the only token also delays its moves. If the delay is unbounded, the upper bounded of the service time is infinite : a processor may never get the token because the token stay forever on the same processor. The algorithm presented in [12] overcomes this drawback. Once stabilized, the service time depends only on the schedule. Under the synchronous scheduler, a token circulation requires N computation steps. Under any schedule, N token circulations require at most $O(N^3)$ computation steps. The memory space needed on each processor by the algorithm [12] is $2\lg(m_N) + 2$ bits.

H. Kakugawa and M. Yamashita in [KY97][6] have presented the first algorithm where the service time is optimal. The idea is to lock forever a single token when the ring has several tokens. Once, there is only one token in the ring ; this token is not delayed or locked, then the service

[5][BCD95] : J. Beauquier, S. Cordier, and S. Delaët. Optimum probabilistic self-stabilization on uniform rings. *WSS95, the 2nd Workshop on Self-Stabilizing Systems*, pages 15.1–15.15, 1995.

[6][KY97] : H. Kakugawa and M. Yamashita. Uniform and self-stabilizing token rings allowing unfair daemon. *IEEE Transactions on Parallel and Distributed Systems*, 8(2) :154–162, 1997.

time is optimal. The algorithm of [KY97] having an optimal service time requires a centralized scheduler. By delaying the token circulation, H. Kakugawa and M. Yamashita in [KY02][7] have adapted the algorithm of [KY97] to run under the unfair distributed scheduler. The waiting time of a token is bounded by 2, thus the service time of the algorithm in [KY02] is $2N$. It is not optimal. In [14], using a technique similar to the one used in the algorithm of [KY97], I present an algorithm that deals with unfair distributed schedulers and ensures an optimal service time. Once the system is stabilized, after N computation steps, each processor has obtained the token one time. The algorithms of [KY97], [KY02], and [14] require on each processor $O(\lg(N))$ bits.

In [01], a self-stabilizing leader election on anonymous rings of any size is proposed. The action of [01] algorithm has to be synchronized with the holding of circulating token : a process carries out an action of [01] algorithm only if it has a token, and the end of the action, it transmits the token to its right-hand side neighbour. Algorithm [01] can be combined with any self-stabilizing token circulation algorithm on unidirectional rings. In [01], we analyzed in detail the combination of this algorithm with the token circulation algorithm presented in [20]. The obtained algorithm required $3\lg(m_N)+1$ bits of space memory space on each processor, it converges under the unfair distributed scheduler.

[7] [KY02] : H. Kakugawa and M. Yamashita. Uniform and self-stabilizing fair mutual exclusion on unidirectional rings under unfair distributed daemon. *Journal of Parallel and Distributed Computing*, 62(5) :885–898, 2002.

Chapter 3 : self-stabilizing algorithms for Ad-Hoc networks

The objective of this chapter is to present my work related to the design of the self-stabilizing protocols adapted to very dynamic networks (i.e. network whose the topology changes very frequently) as ad hoc networks, large scale network, ...

The robustness concept was proposed by M.G. Gouda and J. Cobb [8]. The motivation for robust stabilization is that a system should react gracefully to the input changes - preserving a safety predicate in the presence of the input changes, or at least it should converges very quickly to a configuration satisfying the safety predicate. A robust self-stabilizing system ensures, in priority, the safety of its actions or some basic tasks ; then it evolves to eventually do all its tasks properly. Formally, a self-stabilizing protocol is robust if and only if a predicate $\mathcal{SP}$ on the configurations, named the safety predicate, verifying the following properties, exists.

- $\mathcal{SP}$ is closed.
- Once, the predicate $\mathcal{SP}$ is satisfied, the system can do some basic tasks or at least avoid erroneous actions.
- The convergence time to a configuration verifying predicate $\mathcal{SP}$ should be really faster than the convergence time to a legitimate configuration.

Robust self-stabilizing protocols deal with more efficiency to the input changes that the plain self-stabilizing protocols. A self-stabilizing protocol converges automatically from any configuration to a legitimate configuration. No property is guaranteed during the phase of convergence. Thus the system can have a completely arbitrary behavior during this phase. In the worst case, the duration of this phase of convergence is proportional to the network diameter, for leader election, token circulation, building routing tables, and some clustering protocols, ... This is why, the self-stabilizing protocols are less adapted to very dynamic networks than robust self-stabilizing protocols. The ad hoc networks, the mobile networks, the networks of very big sizes are parts of this category.

Chapter 3 : presented results

Clustering. An ad hoc sensor network means partitioning the nodes into clusters, each one with a cluster-head and (possibly) some ordinary nodes. S. Basagni has proposed, GDMAC, a Generalized Distributed and Mobility-Adaptive Clustering protocol in [Bas99][9]. The cluster-heads are selected according to a node's parameter (called weight). The higher is the weight of a node, the more suitable this node is for the role of cluster-head. In ad hoc sensor networks, amount of bandwidth, memory space or battery power of a processor could be used to determine

[8][CG02] : J.A. Cobb and M.G. Gouda. Stabilization of general loop-free routing. *Journal of Parallel and Distributed Computing*, 62(5) :922–944, 2002.

[9][Bas99] : S. Basagni. Distributed and mobility-adaptive clustering for multimedia support in multi-hop wireless networks. *VTC'99, the IEEE 50th International Vehicular Technology Conference*, pages 889–893, 1999.

weight values. Inside a cluster, the communications to the cluster-head have to be as fast, reliable as possible, because this kind of communication is a key part of network architecture services. Ordinary nodes should to be at distance 1 of their cluster-head, to communicate easily with their cluster-head.
To limit the number of switchings (ordinary nodes going from a cluster to another one), and the number of resignations (cluster-heads scattering their cluster), GDMAC builds clusters having the following properties, named *Ad hoc clustering properties*, defined as follow :

1. Every ordinary node affiliates with a neighbor that is cluster-head and which has higher weight than its weight (affiliation condition).

2. Every ordinary node v affiliates with a quasi-optimal cluster-head : formally, for every cluster-head $z \in N_v : w_z \leq w_{Clusterhead_v} + h$ (cluster-head condition).

3. A cluster-head has at most k neighboring cluster-heads (k being an integer, $0 \leq k < n$) (k-neighborhood condition).

The first requirement ensures that each ordinary node has direct access to at least one cluster-head (the one of the cluster to which it belongs), thus allowing fast intra and inter cluster communications. The second requirement guarantees that each ordinary node always stays with a cluster-head that gives it a « good » service. By varying the threshold parameter h it is possible to reduce the switching overhead associated to the passage of an ordinary node from its current cluster-head to a new neighboring one when it is not necessary. When $h = 0$ we simply obtain that each ordinary node affiliates with the neighboring cluster-head with the highest weight. Finally, the third requirement allows us to have up to k neighboring cluster-heads, $0 \leq k < n$. When $k = 0$ we obtain that no cluster-head can be neighbors.

A self-stabilizing version of GDMAC is presented in [10].

In [07], a robust version of self-stabilizing GDMAC is presented. The robustness property guarantees that, starting from an arbitrary configuration, in one asynchronous round, the network verifies clustering safety property. The *clustering safety property* is defined as : (1) the network is partitioned into clusters, and (2) each cluster has a leader that performs cluster-head tasks. Once a network verifies the clustering safety property, services using the clustering architecture works properly, even if there are not very efficient because the cluster partition is not optimal. Thus, once a degraded service is established, the service stay available during the convergence phase toward a legitimate configuration. thus, there is not service break during self-stabilizing phase. The stabilization time is commensurate with the network diameter, as any weight-based clustering construction.

In [06], a self-stabilizing clustering protocol taking into consideration the number of nodes a cluster-head can handle, and transmission power, mobility, and battery power of the nodes is presented. The protocol guarantees that network nodes are partitioned into clusters : each cluster has at most $SizeBound$ nodes. Thus, this protocol achieves load balancing by guarantee a

threshold ($SizeBound$) on the number of nodes that a cluster-head handle. Therefore, none of the cluster-heads are overloaded at any time. The cluster-heads are selected according to their weight. A trivial solution guaranteeing the first requirements is that every node is the head of cluster having no other member. This solution does not provide an useful hierarchical organization. Therefore, to minimize the number of cluster-heads in a neighborhood, several cluster-heads are neighbor only if it is necessary : no cluster-head can join another cluster and gives up its responsibility. To resume the protocol builds clustering satisfying the *well-balanced clustering properties* :

1. Every ordinary node always affiliates with one cluster-head of its neighborhood which has higher weight than its weight (affiliation condition).

2. There is at most $SizeBound$ nodes in a cluster (size condition).

3. If a cluster-head v has a neighboring cluster-head u such that $w_u > w_v$ then the size of u's cluster is $SizeBound$ (cluster-head neighboring condition).

Routing table. In [13], a robust self-stabilizing version of Distance-Vector routing protocol is presented. The protocol guarantees the following safety predicate : « the weight of a packet decreases at each hop, the weight of a packet being the route cost from the router holding the packet to the receiving host ». Once the safety predicate is satisfied,

- it stays satisfied even if the routes are modified to minimize their costs.
- Thus, any packet reaches its final destination after a bounded number of hops, the bound is the initial route cost from the sender to the receiver.

Open problems

Fault-tolerant Converters

What remains open is whether or not there is a self-stabilizing converter from networks (state or link) with regular registers, to the corresponding network with only safe registers. Interestingly, Lamport's construction of single-writer single-reader single-bit regular registers from single-writer single-reader single-bit safe registers fails to be self-stabilizing (explained in [HPT02][10]). We conjecture that this shortcoming can be rectified at the expense of wait-freedom.

In [DH01][11], S. Dolev and T. Herman proposed a version of the K-states algorithm in the regular-link model. The K-states algorithm, [Dij74], is a self-stabilizing token circulation on the unidirectional rings. In [DH01], it is proven that there is not version of the K-states algorithm using a single safe register by communication link. It would be interesting to propose a version of the algorithm of Dijkstra using several safe registers by communication link, or to prove that such an algorithm cannot be designed.

Probabilistic self-stabilizing algorithms

The results obtained on the memory capacity needed by probabilistic leader election algorithms and token circulation are synthesized in the figure 2.

ring type	problem	service time	lower bound of memory space	upper bound of memory space
unidirectional	Circulation	non optimal	$\lg(m_N - 2) - 1$	$2.\lg(m_N) + 2$
unidirectional	Circulation	optimal	$\lg(m_N - 2) - 1$	$\lg(N + 1) + 2$
unidirectional	Election	-	$\lg(m_N - 2) - 1$	$3.\lg(m_N) + 1$
bidirectional	Circulation	non optimal	?	$2.\lg(m_N) + 12$
bidirectional	Circulation	optimal	?	$\lg(N + 1) + 11$
bidirectional	Election	-	?	$3.\lg(m_N) + 10$

FIG. 2 – Summary of the memory space required

Is bidirectional links, an advantage to ensure the circulation of a token ? In deterministic environment, the answer is clearly yes, in regard of the memory space required per processus. For unidirectional rings, number of states per processus is proportional to ring size ; for bidirectional ring, a constant number of states, for any ring sizes, is sufficient.

[10][HPT02] : J.H. Hoepman, M. Papatriantafilou, and P. Tsigas. Self-stabilization of wait-free shared memory objects. *Journal of Parallel and Distributed Computing*, 62(5) :818–842, 2002.

[11][DH01] : S. Dolev and T. Herman. Dijkstra's self-stabilizing algorithm in unsupportive environments. In *WSS01, the 5th International Workshop on Self-Stabilizing Systems, Springer LNCS : 2194*, pages 67–81, 2001.

What happen in probabilistic environment ? In [IJ90][12], A Israli and M Jalfon established a lower bound on the number of states needed per process by a probabilistic self-stabilizing token circulation algorithm on bidirectional anonymous rings : $\sqrt{m_n}$. To my knowledge, there is not self-stabilizing algorithm on bidirectional rings matching this bound. To design such an algorithm should be a challenging task.

Another open question is the cost of the optimal service time. On unidirectional rings, it seems that the counterpart to obtain an optimal service time is the memory space requirement. Does this conterpart hold on bidirectional rings ? Is it possible to design an algorithm of token circulation having an optimal service time, requiring less $\lg(N)$ bits of memory space on each process ?

A completely different approach is to use a deterministic of token circulation algorithm on bidirectional rings semi-uniform (i.e. ring having a leader). Such an algorithm is proposed E Dijkstra [Dij74]. The algorithm requires 2 bits of memory space and the time of service is $2N$. But, it is necessary to design an effective leader election algorithm on the bidirectional rings. For example, an algorithm whose time of convergence is $O(N^2)$; or an algorithm requiring less $3\lg(m_N)$ bits of memory space.

What can it be said about the memory space necessary to a probabilistic leader election algorithm on bidirectional rings ? I have established a lower bound near of $(m_N - 2)/2$ states by process on unidirection rings in [20]. I conjecture that this lower bound can be improven.

Self-stabilizing algorithms for Ad-Hoc networks

The management of large Ad-Hoc networks (i.e. of the networks of thousands or million nodes interacting) cannot be managed like small size systems. When there are few elements, the input changes (crash fault, transient failure, removal of elements, mobility of nodes, ...) are relatively rare. In large scale system, the average time between two input changes does not allow to recover all its functionalities. The construction of robust self-stabilzing algorithms takes all its interest within this framework.

To design a robust version of protocol of [07]. The robustness property would guarantee that, starting from an arbitrary configuration, in few asynchronous rounds, the network verifies clustering safety property.

The principal task of network management is to ensure the packets routing from the sender to receiver. To ensure a pro-actif routing mecansim, two pieces are necessary (1) a protocol of routing table construction, and (2) a mechanism of packets transfer.

Clustering is often used in order to reduce the quantity of exchanged data to build the routing tables. The packets routing is carried out in two times. Initially, the packet is conveyed to the

[12][IJ90] : A. Israeli and M. Jalfon. Token management schemes and random walks yield self-stabilizing mutual exclusion. *PODC90, the 9th Annual ACM Symposium on Principles of Distributed Computing*, pages 119–131, 1990.

recipient cluster. Finally, the head of this cluster undertakes the transfert of the packet to the receiver. A node needs to keep in its routing table only a route bound per cluster. Thus, the size of the routing tables is reduced and its maintenance is facilitated.

It would be interesting to conceive a complete protocol structure which (1) will be without service break during the convergence phase, and (2) which will have the advantages of a routing mecansim based on node clustering. Once the service is established, the continuity of the service is guaranteed, even during the system evolution toward an optimal operation. Concretely, as soon as a route exists between two nodes, then the packets routing between these two nodes should be ensured.

In the case of routing mecanism based on clusters, the nodes must maintain a routing table and must forward the packets. The routing tables have small size. Nevertheless, this type of approach requires the participation of all nodes to the packets forwarding. A interesting approach is to determine bridges between the cluster-heads such that only cluster-head and bridges participate to the packets routing process. It is possible to improve the existing self-stabilizing construction of a bridges in two research line : (1) to propose a self-stabilizing protocol minimizing the number of bridges, and (2) to propose a robust self-stabilizing protocol, i.e. a protocol without service failure.

PUBLICATIONS of COLETTE JOHNEN

Papers in International Journals with review process

[01] J. Beauquier, M. Gardinariu et C. Johnen. Randomized self-stabilizing and space optimal leader election under arbitrary scheduler on rings, accepted to *Journal of Parallel et Distributed Computing*, 2007 also LRI technical report 1225, 1999.

[02] T. Herman and C. Johnen. Strategies for peer-to-peer downloading. *Information Processing letters*, 94(5) : 203–209, 2005.

[03] C. Johnen, L. O. Alima, A. K. Datta, et S. Tixeuil. Optimal snap-stabilizing neighborhood synchronizer in tree networks. *Parallel Processing Letters*, 12(3&4) : 327–340, septembre 2002.

[04] J. Beauquier, M. Gradinariu, C. Johnen, and J. Durand-Lose. Token-based self-stabilization uniform algorithms. *Journal of Parallel et Distributed Computing*, 62(5) : 899–921, mai 2002.

[05] A. K. Datta, C. Johnen, F. Petit and V. Villain. Self-stabilizing depth-first token circulation in arbitrary rooted networks. *Distributed Computing*, 13(4) : 207–218, octobre 2000.

Papers in International Conferences with review process

[06] C. Johnen et L.H. Nguyen. Self-stabilizing bounded size clustering algorithm. In *ISPA07, the 5th International Symposium on Parallel and Distributed Processing and Applications*, 2007

[07] C. Johnen and L. H. Nguyen. Robust self-stabilizing clustering algorithm. In *OPODIS'06, the 10th International Conference On Principles Of DIstributed Systems, Springer LNCS 4305*, pages 408–422, 2006.

[08] J. Beauquier, C. Johnen, and S. Messika. All k-bounded policies are equivalent for self-stabilization. In *SSS'06, the 8th International Symposium on Stabilization, Safety, and Security of Distributed Systems, Springer LNCS : 4280*, pages 82–94, 2006.

[09] K. G. Dastidar, T. Herman, and C. Johnen. Safe Peer-to-Peer self-downloading. In *SSS'06, the 8th International Symposium on Stabilization, Safety, and Security of Distributed Systems, Springer LNCS 4280*, pages 324–334, 2006.

[10] C. Johnen and L. H. Nguyen. Clustering algorithm for ad hoc networks. In *AlgoSensors'2006, the 2nd International Workshop on Algorithmic Aspects of Wireless Sensor Networks, LNCS 4240*, pages 83–94, 2006.

[11] L. Higham and C Johnen. Relationships between communication models in networks using atomic registers. In *IPDPS'06, the 20th IEEE International Parallel and Distributed Processing Symposium*, 2006.

[12] C. Johnen. Bounded service time and memory space optimal self-stabilizing token circulation protocol on unidirectional rings. In *IPDPS'04, the 18th IEEE International Parallel and Distributed Processing Symposium*, 2004.

[13] C. Johnen and S. Tixeuil. Route preserving stabilization. In *SSS03, the 6th Symposium on Self-Stabilizing Systems, Springer LNCS :2704*, pages 183–197, 2003.

[14] C. Johnen. Service time optimal self-stabilizing token circulation protocol on anonymous unidirectional. In *SRDS'02, the 21st Symposium on Reliable Distributed Systems*, pages 80–89, ed. IEEE Computer Society Press, 2002.

[15] J. Beauquier, M. Gradinariu, et C. Johnen. Cross-over composition - enforcement of fairness under unfair adversary. In *WSS'01, the 5th International Workshop on Self-Stabilizing Systems, Springer LNCS :2194*, pages 19–34, 2001.

[16] F. E. Fich et C. Johnen. A space optimal, deterministic, self-stabilizing, leader election algorithm for unidirectional rings. In *DISC'01, the 15th International Symposium on Distributed Computing, Springer LNCS :2180*, pages 224–239, 2001.

[17] M. Gradinariu et C. Johnen. Self-stabilizing neighborhood unique naming under unfair scheduler. In *Euro-Par'01, Parallel Processing, Springer LNCS :2150*, pages 458–465, 2001.

[18] I. Lavallée, C. Lavault, et C. Johnen. Fair and reliable self-stabilizing communication. In *OPODIS'00, the 4th International Conference On Principles Of DIstributed Systems*, pages 163–176, 2000.

[19] C. Johnen, L. O. Alima, A.K. Datta, et S. Tixeuil. Self-stabilizing neighborhood synchronizer in tree networks. In *ICDCS'99, the 19th IEEE International Conference on Distributed Computing Systems*, pages 487–494, 1999.

[20] J. Beauquier, M. Gradinariu et C. Johnen. Memory space requirements for self-stabilizing leader election protocols. In *PODC'99, the 18th ACM Symposium on Principles of Distributed Computing*, pages 199–208, 1999.

[21] G. Alari, J. Beauquier, A. K. Datta, C. Johnen, and V. Thiagarajan. Fault-tolerant token passing algorithm on tree networks. In *IPCCC'98, the 1998 IEEE International Performance Computing, and Communications Conference*, pages 44–50, 1998.

[22] A. K. Datta, C. Johnen, F. Petit, and V. Villain. Self-stabilizing depth-first token circulation in arbitrary rooted network. In *SIROCCO98, the 5th International Colloquium On Structural Information and Communication Complexity*, pages 32–46, 1998.

[23] C. Johnen. Memory-efficient self-stabilizing algorithm to construct BFS spanning trees (long version of paper in PODC'07). In *WSS'07, the 3rd Workshop on Self-Stabilizing Systems*, pages 125–140. CARL, 1997.

[24] C. Johnen, G. Alari, J. Beauquier, and A. K. Datta. Self-stabilizing depth-first token passing on rooted networks. In *WDAG'97, the 11th International Workshop Proceedings Distributed Algorithms, Springer LNCS :1320*, pages 260–274, 1997.

[25] C. Johnen and J. Beauquier. Space-efficient distributed self-stabilizing depth-first token circulation. In *WSS'95, the 2nd Workshop on Self-Stabilizing Systems*, pages 4.1–4.15, 1995.

[26] C. Johnen and F. Mourlin. Analysis of the communication structure of OCCAM2 programs using petri nets. In *Transputers'92, Advanced Research and Industrial Applications*. IOS, May 1992.

[27] G. Berthelot, C. Johnen et L. Petrucci. *Computer-Aide Verification '90*, volume 3 of *D.I.M.A.C.S.*, chapter PAPETRI : Environment for the analysis of PETRI nets. A.C.M./A.M.S., 1991.

[28] C. Johnen. Algorithmic verification of home spaces in Place/Transition systems. In *12th IMACS world congress*, 1988.

[29] C. Choppy et C. Johnen. Petrireve : Proving petri net properties with rewriting systems. In *RTA'85, the 1st International Conference on Rewriting Techniques and Applications, LNCS :202*, pages 271–286, 1985.

Short Papers in International conferences with review process

[30] C. Johnen and L. Higham. Fault-tolerant implementations of atomic-state communication model in weaker Networks. In *DISC'07, the 21th International Symposium on Distributed Computing*, 2007.

[31] J. Beauquier, C. Johnen, and S. Messika. Brief announcement : Computing automatically the stabilization time against the worst and the best schedulers. In *DISC'06, the 20th International Symposium on Distributed Computing, Springer LNCS :4167*, pages 543–547, 2006.

[32] C. Johnen. Memory efficient, self-stabilizing algorithm to construct BFS spanning trees (brief announcement) In *PODC'97, the 16th Annual ACM Symposium on Principles of Distributed Computing*, page 288, August 1997.

Contrat Reports

[33] J. Beauquier, C. Johnen, E. Pelz, and N. Polian. *Clocked Petri nets*, 1985. final report of agreement « High level for Interactive Layout and Design ».

Software manuals

[34] C. Johnen. *NETMATE Object Model SYBASE technical report.* Columbia University, New-York, 1992.

[35] P. Fraisse, C. Johnen et N. Treves. Serpe : an extensible structure for analysis of petri nets. Technical Report 302, L.R.I, Université of Paris-Sud, Orsay, France, September, 1986.

[36] C. Johnen and C. Choppy. *Welcome to the Petrireve Environment*, 1985. manual of Petrireve.

Papers in French Journals with review process

[37] C. Johnen, F. Petit, and S. Tixeuil. Autostabilisation et protocoles réseau. *Technique et Science Informatique*, 23(4) : 1027–1056, 2004.

[38] I. Lavallée, C. Lavault, and C. Johnen. *chapter : Exorcisme ou communication fiable et équitable autostabilisée*, pages 9–21. in parallélisme, réseaux et répartition. J.-F. Myoupo, Hermes, 1998.

Papers in French Conferences with review process

[39] C. Johnen. Service time of self-stabilizing token circulation protocol on anonymous unidirectional rings - extended abstract -. In *AlgoTel'03, les 5ième Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*, 2003.

[40] D. Barth, C. Johnen et V. Vèque. Routage et reroutage dans un réseau mobile. In *AlgoTel'00, les 2èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*, 2000.

[41] A. Finkel and C. Johnen. Construction efficace du graphe de couverture minimal : Application à l'analyse de protocole. In *CFIP'91, le Colloque Francophone sur l'Ingénierie des Protocoles*, 1991.

[42] C. Johnen and G. Memmi. D'espace d'accueil dans les réseaux de petri. In *CFIP'91 le Colloque Francophone sur l'Ingénierie des Protocoles*, 1991.

LRI Technical reports

[43] L. Higham and C. Johnen. Self-stabilizing implementation of atomic register by regular register in networks framework. Technical Report 1449, L.R.I, 2006.

[44] J. Beauquier, C. Johnen, and S. Messika. Computing automatically the stabilization time against the worst and the best schedulers. Technical Report 1448, L.R.I, 2006.

[45] J. Beauquier et C. Johnen. Analyze of randomized self-stabilizing algorithms under non-deterministic scheduler classes. Rapport Technique 1327, L.R.I, 2002.