Quelques contributions à l'auto-stabilisation

« Several contributions to self-stabilization »

Colette Johnen

Defense of « HdR » - November 2007

Edsger W. Dijkstra [1974]

- Edsger W. Dijkstra: Self-stabilizing Systems in Spite of Distributed Control. Commun. ACM 17(11): 643-644 (1974)

Task : « one circulating token in the ring » **is eventually reached from any initial configuration**

Self-Stabilization

Leslie Lamport's talk at PODC [1983]

- Extrait of Lamport's web site :«**I regard the resurrection of Dijkstra's brilliant work on self-stabilization to be one of my greatest contributions to computer science.**»

Self-stabilization System converges from any initial configuration to a correct configuration

Un-masking tolerance to
- any topology changes
- any transient failures

Self-stabilizing systems

Self-stabilizing system is now a subfield of distributed systems:

- 9th International Symposium on Stabilization, Safety, and Security of Distributed Systems
14th -16th November 07, Paris

- CiteSeer references ±500 papers

Several contributions to self-stabilization - outline

- Preamble
- Optimisation of memory space

Memory Space

Taxonomy of memory space required on each node :
- $O(1)$ bits – constant
- $O(\lg(\Delta))$ bits – Δ : node's degree [JB95, DJPV97, J97, DJPV00]
- $O(\lg(m_N))$ bits – m_N is the smallest integer that does not divide N [IJ90]
- $O(\lg(N))$ bits – N is the number of nodes

Several tracks to explore

Benchmark of distributed systems:
Token Circulation
Leader Election
on anonymous rings

[Dijkstra 74] : Self-stabilizing Deterministic token circulation on semi-uniform rings (unidirectional or bidirectional)

Several contributions to self-stabilization- outline

- Preamble
- Complexity in memory space
  Unidirectional rings
  Deterministic Algorithms

Deterministic Leader Election on Unidirectional Rings

System requirement :
centralized schedules, prime-size rings

[J. Burns and J. Pachl, 1989]
[J. Beauquier, M. Gradinariu, C. Johnen, 1999]

lg(N) bits ≤ Leader Election ≤ 2lg(N) bits

Deterministic Leader Election on Unidirectional Rings

System requirement :
centralized schedules, prime-size rings

[J. Beauquier, M. Gradinariu, C. Johnen, 1999]
[F. Ellen Fich, C. Johnen, 2001]

lg(N) bits ≤ Leader Election ≤ lg(N) + 4 bits

Leader Election by a deterministic algorithm : $\Theta(\lg(N))$

Deterministic Token Circulation on Unidirectional Rings

System requirement :
centralized schedules, prime-size rings

[J. Beauquier, M. Gradinariu, C. Johnen, 1999]
[F. Fich,C. Johnen, 01] + [F.F. Haddix, M. Gouda 96]

lg(N) – 1 bits ≤ Token Circulation ≤ lg(N) + 7 bits

Token Circulation by a deterministic algorithm : $\Theta(\lg(N))$

Several contributions to self-stabilization- outline

- Preamble
- Complexity in memory space
  Unidirectional rings
    Deterministic Algorithms
    Probabilistic Algorithms
      Token Circulation

History : Probabilistic Token Circulation on Unidirectional Rings

[1990, T. Herman] algorithm :

$Token_p$ → to flip a coin, *if* « head » *then* $Pass\_Token_p$

$Token_p \equiv v_p = v_l$
$Pass\_Token_p \equiv v_p := v_{l+1} \bmod (2)$

Odd size rings, 1 bit = $\lg(m_N)$ synchronous schedule

History : Probabilistic Token Circulation on Unidirectional Rings

- [1995, J. Beauquier, A. Cordier, S. Delaët] : any rings, memory-k bounded fair schedules
  - $\lg(m_N)$ bits
    - m_N is the smallest integer that does not divide N (N size of the ring)
- [1995-1997, H. Kakugawa, M. Yamashita] : any rings, centralized schedules
  - $2\lg(N)+1$ bits

Probabilistic Token Circulation on Unidirectional rings

- [1999-2007, J. Beauquier, M. Gradinariu, C. Johnen] : any rings, any schedules

$\lg(m_N-2) - 1$ bits $\leq$ Token Circulation $\leq 2\lg(m_N)$ bits

Token Circulation by a probabilistic algorithm : $\Theta(\lg(m_N))$

Service Time

Service time is the upper bound of the waiting time of the token by a node

- [H90, BCD95, BGJ99, BGJ07]'s algorithm are based on the same technique :

« **randomly retard the token circulation** »

Service Time of these algorithms is unbounded

Bounded Service Time

Probabilistic Token Circulation	Service Time	Memory space
2000-2004, A. Datta, M. Gradinariu, S. Tixeuil	$(N+1)N^2$	$2\lg(N.m_N)$
2002, H. Kakugawa, M. Yamashita	2N	$\lg(N)+1$
2002, C. Johnen	N - optimal	$\lg(N+1)+1$
2004, C. Johnen	N^2	$2\lg(m_N)+2$

Requirements : any rings, any schedules

Several contributions to self-stabilization - outline

- Preamble
- Complexity in memory space
  - Unidirectional rings
    - Deterministic Algorithms
    - Probabilistic Algorithms
      - Token Circulation
      - Leader Election

Probabilistic Leader Election on Unidirectional Rings

- [1999-2007, J. Beauquier, M. Gradinariu, C. Johnen] : any rings, any schedules

$$\lg(m_N\text{-}2) - 1 \leq \text{Leader Election} \leq 3\lg(m_N)+1$$

Leader Election by a probabilistic algorithm : $\Theta(\lg(m_N))$

Several contributions to self-stabilization- outline

- Preamble
- Complexity in memory space
  Unidirectional rings
  Open Problem

Bidirectional anonymous rings

Token Circulation	Service Time	Memory Space
2000. J. Durand-Lose [IJ90] - convergence time: N^2	unbounded	$2\lg(m_N)+c$
2002, C. Johnen 1993 A. Israeli, M. Jalfon	N	$\lg(N+1)+11$
2004, C. Johnen 1993 A. Israeli, M. Jalfon	N^2	$2\lg(m_N)+12$
Conjecture	2N	$4\lg(m_N)+c'$

Several contributions to self-stabilization - outline

- Preamble
- Complexity in memory space
- Communication Models by registers

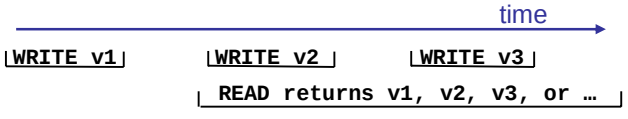
Single-Writer Register

- A register is a memory cell on which two types of operations are possible **READ** and **WRITE**
- On a Single-Writer register, only one processor can do the **WRITE** operation
- On a register, **READ** and **WRITE** operation are not atomic, they take some time

A **READ** operation a register R may overlap an **WRITE** operation on R

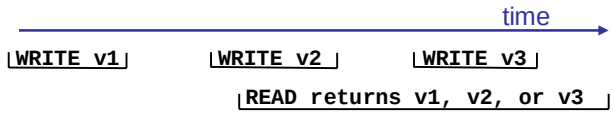
Single-Writer Safe Registers [Lamport 1986]

- On R, a **READ** operation that does overlap a **WRITE** operation returns any value (v1, v2, v3 or ….)



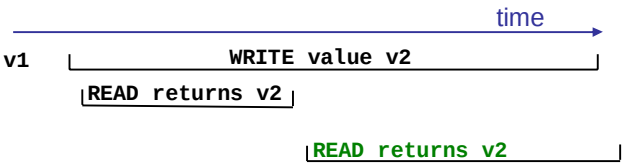
Single-Writer Regular Registers [Lamport 1986]

- On R, a **READ** operation that does overlap a **WRITE** operation returns the most recent preceding written value (v1) or any value written during overlapping **WRITE** operations (v2, v3)



Single-Writer Atomic Registers [Lamport 1986]

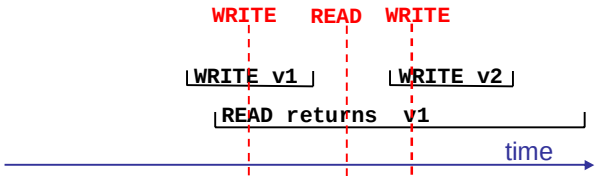
- Regular register such that if a **READ** operation returns the value written during the overlapping **WRITE** operation then any subsequence **READ** cannot return the most recent preceding written value (v1)



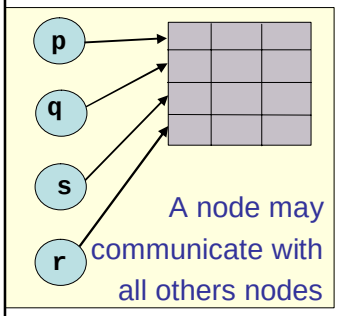
Property of Atomic register

- A sequence of operations on an atomic register is linearizable [HW90]

  each operation appears to happen instantaneously at some point during its execution



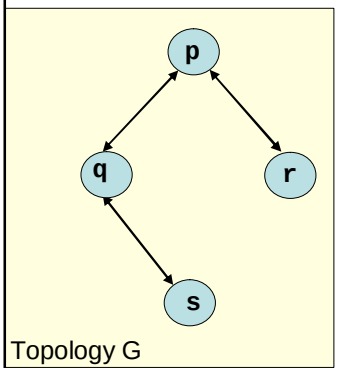
Shared Memory Model



[L86] [HW90] [HV95] [LTV96] [HPT02] : wait-free implementation of atomic register by regular or safe ones

A wait-free operation is always done in a finite number of steps

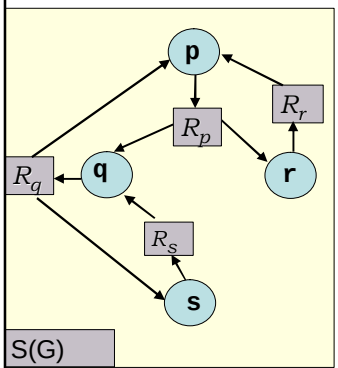
Networks



A node can only communicate with its neighbors

Ex: *p* can only communicate with *q* and *r*

State network model

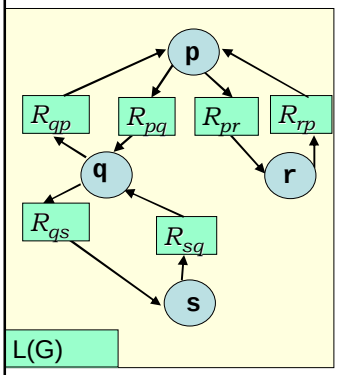


A node *p* has a single-writer multi-reader register R_p

R_p is readable by *p*'s neighbors

R_p is writable only by *p*

Link network model



A node *p* has several single-writer single-reader registers (one per neighbor)

R_{pq} is readable by *q*

R_{pq} is writable by *p*

Bibliography

- Self-stabilizing algorithms in **ATOMIC-LINK** network model :

[S. Dolev, A. Israeli, S. Moran 1993]
called R/W atomicity model

- Self-stabilizing algorithms in **ATOMIC-STATE** network model :

[M. Mizumo, M. Nesterenko 1998]
[G. Antonoiu, P.K. Srimani 1999]
[M. Nesterenko, A. Arora 2002]

Several contributions to self-stabilization - outline

- Preamble
- Complexity in memory space
- Model of communication by registers
  Compilers

Compiler from Com1 to Com2

S1=(G, Com1, *Algo)* → S2=(G, **Com2, τ(*Algo*)**)

READ(*R*) and **WRITE**(*R,v*) operations in S1 are replaced in S2, by two programs τ(**READ**(*R*)) and τ(**WRITE**(*R,v*)) using Com2

τ is a compiler iff S2 = (G, Com2, τ(*Algo*))
is a syntactically and semantically valid transformation of S1 = (G, Com1, *Algo)*

Self-stabilizing compilers

	Atomic	Regular	Safe
State multi-reader	Silent, obstruction-free C. Johnen, L. Higham 07		
Link Single reader	L. Higham, C. Johnen 06		

Wait-free compilers

	Atomic	Regular	Safe
State multi-reader	L. Higham, C. Johnen 07		
Link Single reader	L. Higham, C. Johnen 06	L. Lamport 1986	L. Lamport 1986

Wait-free and Self-stabilizing compilers Conjecture

	Atomic	Regular	Safe
State multi-reader			
Link Single reader	«L. Lamport 86 » →	→	

Several contributions to self-stabilization - outline

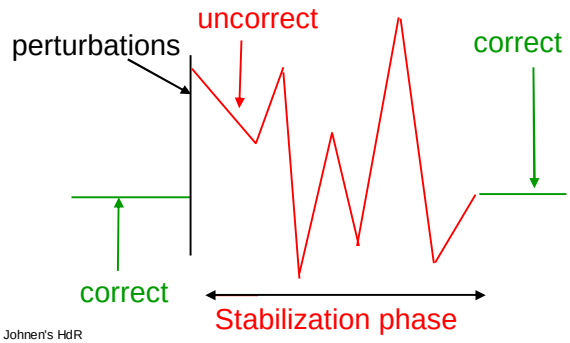
- Preamble
- Complexity in memory space
- Model of communication by registers
- Algorithmic for ad-hoc networks

Ad-Hoc Networks

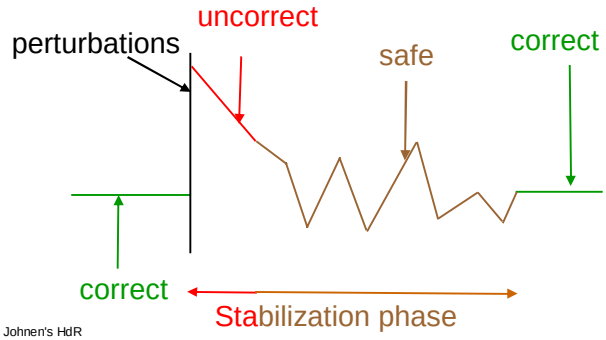
- A temporal, independent multi-hops network that provides peer-to-peer connectivity
- Network's topology may change rapidly and unpredictably
- Needs for an efficient management
  - Self-configuration
  - Self-maintenance
  - Self-healing

Self-stabilizing network management

Drawbacks of Self-stabilizing Protocols



Robust Self-Stabilizing Protocols [J. Cobb, M. Gouda, 2001]



Clustering Protocols

- Clustering : to aggregate nodes into set (named clusters)

J.-F. Myoupo and al. 2004
F. Theoleyre, F. Valois 2005
D. Simplot-Ryl and al. 2007
K. Al Agha and al. 2006

Clustering technique is a tool to increase scalability :
- routing information
- data aggregation

Clustering for Ad-Hoc networks

- Easy inter-cluster management :
  - Cluster-head, 1-hop cluster
- Maintaining stable clusters :
  - Weight-based clustering : Cluster-head selection is based on node's weight –
  - The cluster-head is the node of the cluster having the highest weight

➢Time to build the weight-based clusters is O(*D*), where *D* is the network diameter

GDMAC

GDMAC [S. Basagni 1999] is a generic algorithm building weight-based 1-hop clusters

Computation of a suitable weight can be done by several technique: [GT95, BKL01, CDT02, MBF04, BKAGB06, RRGA06]

- power battery of the node
- node mobility
- node degree/transmission power

Robust and self-stabilizing version of GDMAC [L. Nguyen, C. Johnen 2006]

- In a Safe configuration, the network is correctly partitioned :
  - Each cluster has a node acting as a cluster-head
  - Each node belongs to a 1-hop cluster
- Convergence time to a safe configuration is 1 round
- The safety predicate is still verified even if nodes change their weight

Project : robust and self-stabilizing network management

Cluster based routing protocol :

– Robust and SS clustering
[L. Nguyen, C. Johnen 2006]

– Robust and SS routing schema
[C. Johnen, S. Tixeuil 2003]

- In safe configuration, guarantee the packet's transporting from any node to any nodes
- Robustness to changes of node's weights and link's cost.

Projects

- Complexity in memory space
  - Token circulation and leader election on Bidirectional anonymous rings
- Model of communication by registers
  - Wait-free and self-stabilizing Compilers
- Algorithmic for ad-hoc networks
  - Robust and self-stabilizing network management

Thank you