A snap-stabilizing point-to-point communication protocol in message-switched networks.

ANR project SHAMAN

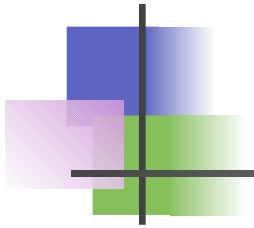
DUBOIS Swan

Accepted paper at IPDPS'09 (Rome, 25-29 may 2009)

- Joint work with :
  - Vincent VILLAIN
    (UPJV, Amiens)

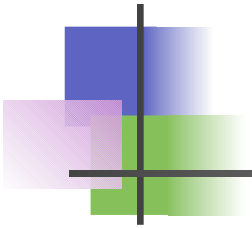
  - Alain COURNIER
    (UPJV, Amiens)

Map



1. Point-to-point communication.
2. Adopted scheme.
3. Self-stabilization.
4. Proposed algorithm.
5. Conclusion.

Part I : Point-to-point communication

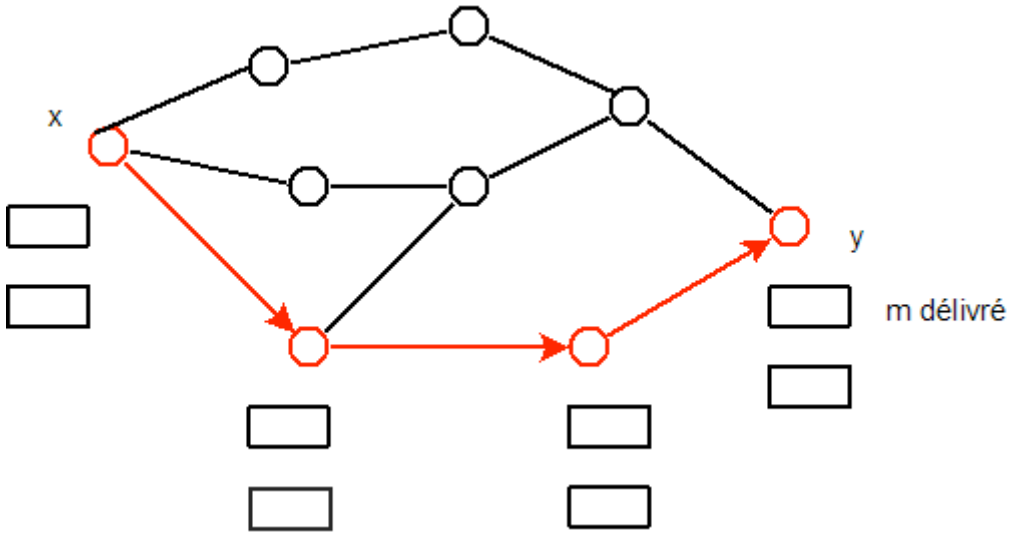


1. Problems to solve.
2. Message-switched network.
3. Forwarding problem.
4. Specification.

Problems to solve

- P wants to send a message to Q.
- First problem:
  - Determination of a path from P to Q.
  - Routing problem.
- Second problem:
  - Management of network ressources.
  - Forwarding problem.

Message-switched network



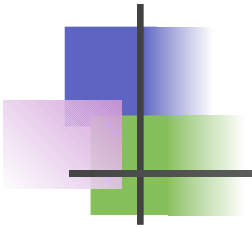
Forwarding problem

- Finite memory => finite number of buffers
- Management of a finite amount of ressources :
  - Deadlocks.
  - Livelocks.

Specification

- Safety :
  - Any emitted message is delivered to its destination once and only once in a finite time.
- Liveness :
  - Any message can be generated in a finite time.

Part II : Adopted scheme



1. Deadlock-free controller.
2. Buffer graph.
3. Theorem.
4. How to solve forwarding problem ?

Deadlock-free controller

- No control on messages moves

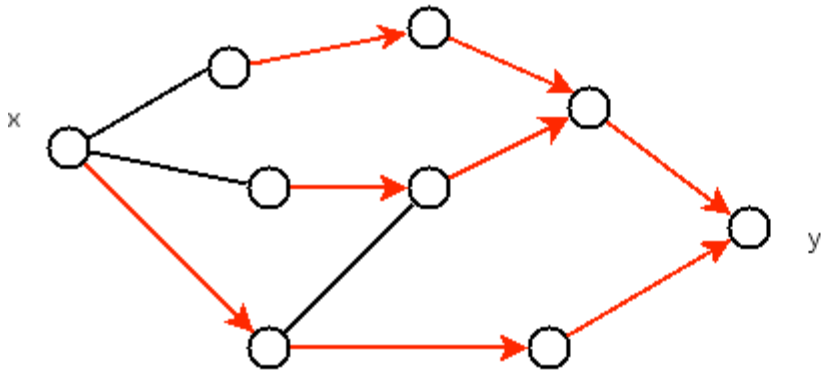
  => risk of deadlocks

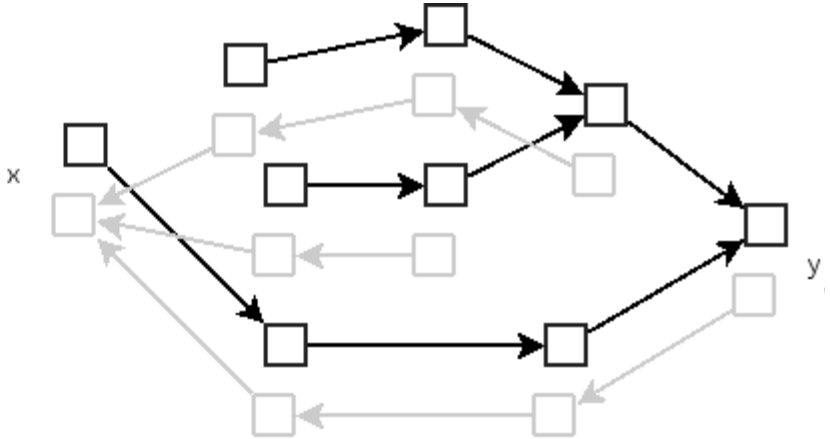
- Design of an algorithm A to control messages moves dynamically.

- If A avoid deadlocks, A is a deadlock free controler.

Buffer graph

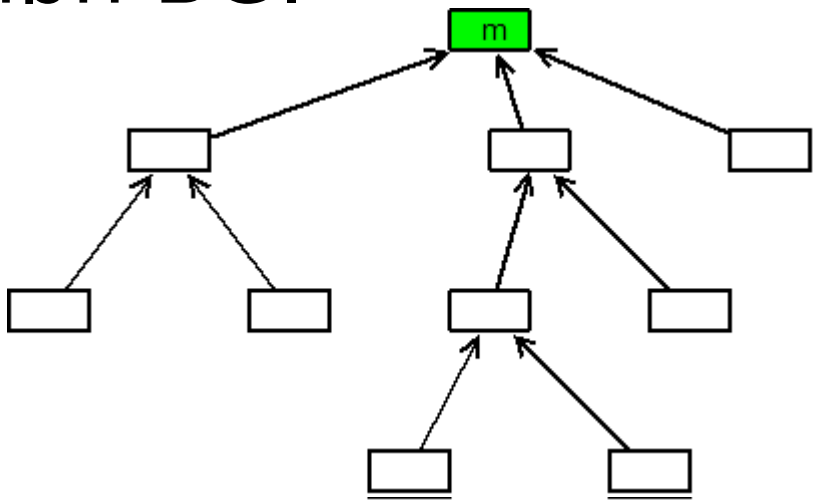
- Key idea: representing all paths followed by messages with a graph.





Theorem

- Construction of a controler C based on a buffer graph BG.



- [MS78,TS81] :
  - C is deadlock-free if and only if BG is acyclic.

How to solve forwarding problem ?

- Given a deadlock-free controler C, we can solve forwarding problem if :
  - C is fair.
  - C does not loose messages.

Part III : Self-stabilization

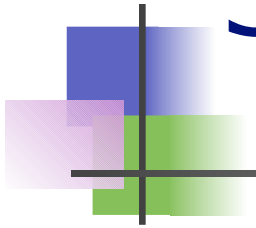
1. Definition.
2. Snap-stabilization.

Definition [D74]

- Starting from an abitrary state, the system recovers a correct behavior in a finite time.
- Arbitrary state can modelise effects of transient faults on the system.

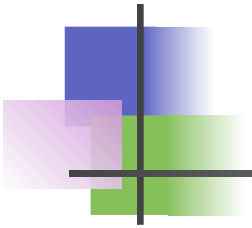
=> Self-stabilizing algorithms can repair themselves from any number of transient faults.

Snap-stabilization [BDPV99]



- Starting from an abitrary state, the system always has a correct behavior.
- Consequence : the first querry must be satisfied.
- Forwarding problem : request = emission of a new message.

Part IV : Proposed algorithm



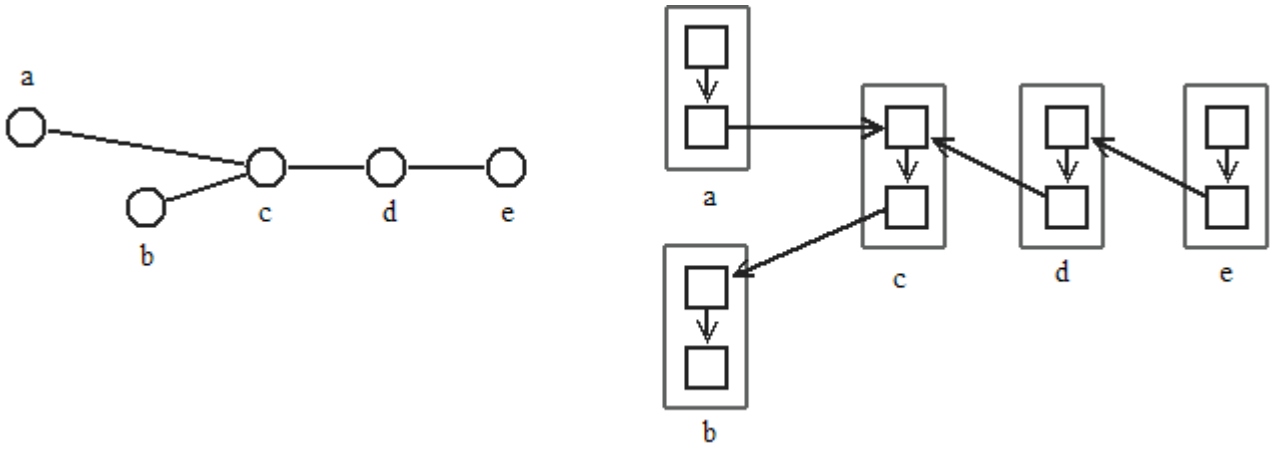
1. Problems of the former scheme.
2. Buffer graph.
3. Avoidance of duplication.
4. Avoidance of fusion.
5. Global example.

Problems of the former scheme

- First idea :
  - Keep the buffer graph.
  - Forwarding messages in spite of routing tables moves.
- Problems :
  - Duplication of a message.
  - Fusion of two differents messages.

Buffer graph

- A tree per destination.
- Two buffers per processor per tree :
  - A reception buffer.
  - An emission buffer.



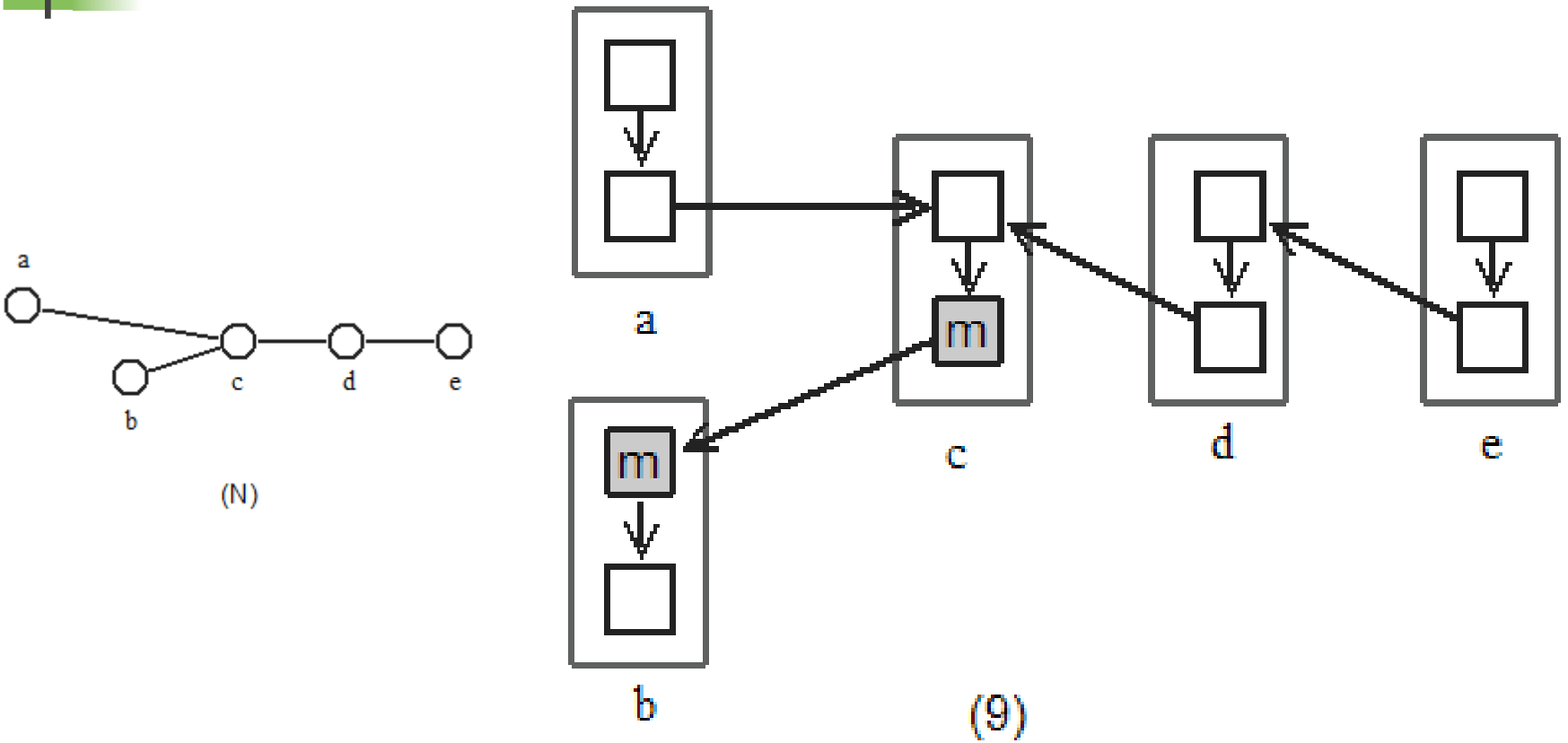
Avoidance of duplication

- Internal forwarding of m allowing only when m has been erased of the former emission buffer.
- Erasing from an emission buffer only if the message has been copied in ONE reception buffer.
- Erasing of m when it has been copied in a « bad » reception buffer.

Avoidance of fusion

- Use of a flag (p,c) where :
  - p is the last processor crossed by m.
  - c is a color. c is computed at each internal forwarding :
    - c is a color holds by no message in emission buffers of neighbors.
- If two messages have different color, they are considered different.

Global example



PART V : Conclusion

1. General remarks.
2. Perspectives.

General remarks

- First snap-stabilizing forwarding protocol.
- No signifiant overcost (space and time) with respect to the fault-free adapted protocol.

Perspectives

- Determination of the minimal number of buffers needed for solve the problem.
- Adapt other graph buffers.
- Work on other switching schemes.
- More practical model.

Questions ?