DISTRIBUTED ALGORITHMS 2013

SEMINAR

Archana Nottamkandath & István Haller

Vrije Universiteit Amsterdam

October 31, 2013
Exercises for today:

- 4.1, 4.2 (chapter Wave - Traversal)
- 4.3 (chapter Wave - Tree)
- 4.4, 4.5, 4.6 (chapter Wave - Echo)
- 5.1, 5.2, 5.3, 5.4, 5.5, 5.6 (chapter Deadlock detection)
Give an example of an execution of Awerbuch’s algorithm in which an information message and an acknowledgement are communicated through a tree edge.
REMINDER: AWERBUCHE’S ALGORITHM

- Each node maintains neighbor knowledge
- A node holding the token for the first time:
  - Decides on first destination to forward to
  - Informs all neighbors of the token reception, except for its father and the selected node
  - Forwards the token to the selected destination
- A node receiving the token back:
  - Forwards the token to any other unseen neighbors
  - If all neighbors seen, return token to parent
Give an example of an execution of Cidon’s algorithm in which two information messages and two tokens are communicated through the same channel in the network.
Acknowledgements are abolished: the token is forwarded without delay

Each node keeps track of the neighbor $mrs_u$ to which it forwarded the token last

If a node $u$ happens to receive the token from a node $v \neq mrs_u$:

- $u$ marks the edge $uv$ as used and purges the token
- Eventually $v$ will receive the info message from $u = mrs_v$ and will continue by sending the token to another neighbor
Argue that the tree algorithm takes at most $D$ time units to terminate, in case we take into account the time needed to communicate the decision to all processes.
Only works for **acyclic networks**

Each node:

- Waits until it received messages from all neighbors, **except one**, which becomes its father; then it sends to it
- If it receives a **message from its father**, it decides; it then sends the decision to all neighbors, except for the father
Consider an undirected network of $N > 3$ processes $p_0, ..., p_{N-1}$, where $p_1, ..., p_{N-1}$ form a ring and $p_0$ has a channel to all other processes. (Note that this network has diameter 2.) Give a computation of the echo algorithm on this network, with $p_0$ as initiator, that takes $N$ time units to complete.
Reminder: Echo Algorithm

- Initiator sends a message to all neighbors
- When a non-initiator receives a message for the first time, it makes the sender its father; then it sends a message to all other neighbors (forward)
- When a non-initiator has received a message from all neighbors, it sends a message to its father (backward)
- When the initiator has received a message from all neighbors, it decides
Argue that the echo algorithm takes at most $2 \times N - 2$ time units to terminate.
Suppose you want to use the echo algorithm in a network where duplication of messages may occur. Which modification should be made to the algorithm?

**Observation**

Each node keeps a counter of the neighbors from which it has received a message.
Suppose node $u$ sends a request to node $v$, then purges this request, and next sends another request to $v$. Let the purge message reach $v$ first, then the second request, and finally the first request. How should $v$ process these three messages?
We model communication and resource deadlocks using N-out-of-M model.

A (non-blocked) process can issue a request to $M_p$ other processes, and becomes blocked until $N_p$ of these requests have been granted.

Only non-blocked processes can grant a request.

In case of mutual exclusion, the resource grants a request, by inverting the request edge first.
Give a computation on a wait-for graph in which $free_u$ remains false for some non-initiator $u$ after running the Bracha-Toueg algorithm, while $u$ is not deadlocked in the basic algorithm.
REMINDER: BRACHA-TOUEG DEADLOCK DETECTION ALGORITHM

- Initially $\text{notified}_u = \text{false}$ and $\text{free}_u = \text{false}$ at all nodes $u$.
- The initiator starts a deadlock detection run by executing $\text{Notify}$.

\[ \text{_notify}_u := \text{true} \]
- for all $w \in \text{Out}_u$ send NOTIFY to $w$
- if $n_u = 0$ then $\text{Grant}_u$
- for all $w \in \text{Out}_u$ await DONE from $w$

\[ \text{grant}_u := \text{true} \]
- for all $w \in \text{In}_u$ send GRANT to $w$
- for all $w \in \text{In}_u$ await ACK from $w$
While a node is awaiting *DONE* or *ACK* messages, it can process incoming *NOTIFY* and *GRANT* messages.

**NOTIFY**
- If $\text{notified}_u = \text{false}$, then $u$ executes $\text{Notify}_u$
- $u$ sends back *DONE*

**GRANT**
- If $n_u > 0$, then $n_u := n_u - 1$; if $n_u$ becomes 0, then $u$ executes $\text{Grant}_u$
- $u$ sends back *ACK*

When the initiator has received *DONE* from all nodes in its *Out* set, it checks the value of its *free* field

- If it is still false, the initiator concludes it is deadlocked.
A way of considering it: a recursion of echo algorithms.

Initiator starts an echo algorithm with Notify.

Any node which can grant requests \((\text{requests} = 0)\), suspends response, starting an echo algorithm of its own with Grant.

As a result of the Grant waves, other nodes can join in with their own ones.

This recursive echo generation stops when no more requests can be granted.
Exercise 5.6

Suppose that the order in which resource requests are granted is predetermined. Give an example of a snapshot with a resource deadlock that isn’t discovered by the Bracha-Toueg algorithm. Show that in case of a nondeterministic selection which resource request is granted, the deadlock in your example may be avoided.

Observation

Bracha-Toueg grants all possible requests at once, independant of mutual exclusion.
EXERCISE 5.1

Let node \( v \) initiate a deadlock detection run, in which the waitfor graph below is computed. Give a computation of the Bracha-Toueg algorithm.
Let node $u$ initiate a deadlock detection run, in which the waitfor graph below is computed. Give a computation of the Bracha-Toueg algorithm.
Exercise 5.3

Let node $u$ initiate a deadlock detection run, in which the waitfor graph from the previous exercise is computed, with as only difference that $w$ is waiting for a 2-out-of-3 (instead of 1-out-of-3) request. Give a computation of the Bracha-Toueg algorithm.