Distributed Algorithms

Fall 2019

Midterm Solutions, 18/11/2019

Matteo Monti <matteo.monti@epfl.ch>
Athanasios Xygkis <athanasios.xygkis@epfl.ch>

Exercise 1a (1/2)

State the interface and properties of Terminating Reliable Broadcast

Interface:

- Broadcast(m): Broadcasts a message m to all processes. It is only executed by a process s, which is known to all other processes.
- Deliver(m): Delivers a message m or the symbol φ.

Exercise 1a (2/2)

Properties:

- *Integrity*: If a process delivers a message m, then m is either φ or m was broadcast by s.
- *Validity*: If the sender s is correct and broadcasts a message m, then s eventually delivers m.
- *Agreement*: For any message m, if a correct process delivers m, then every correct process eventually delivers m.
- *Termination*: Every correct process eventually delivers exactly one message.

Exercise 1b (1/3)

Perfect Failure Detector (Given for explanation purposes. Not necessary for a complete answer):

Strong accuracy:
No correct process is ever suspected:

$$\forall F, \forall t \in \mathbb{R}, \forall p \in correct(F), \forall q : p \notin H(q, t)$$

Strong completeness:
Eventually, every faulty process is permanently suspected by every correct process:

$$\forall F, \forall p \in faulty(F), \exists t \in \mathbb{R}, \forall q \in correct(F), \forall t' \geq t : p \in H(q, t')$$

Where:

- F is a function that to each time t associates the set of processes that crashed at time t.
- faulty(F) is the set of processes that eventually crash.
- correct(F) is the set of processes that never crash.
- H(p, t) is the output of the failure detector of process p at time t.

Exercise 1b (2/3)

Eventually Perfect Failure Detector (Given for explanation purposes. Not necessary for a complete answer):

Eventually strong accuracy:
There is a time after which no correct process is suspected by any correct process:

$$\forall F \exists t \in \mathbb{R}, \forall t' \geq t, \forall p \in correct(F), \forall q \in correct(F) : p \notin H(q, t')$$

Strong completeness:
Eventually, every faulty process is permanently suspected by every correct process:

$$\forall F, \forall p \in faulty(F), \exists t \in \mathbb{R}, \forall q \in correct(F), \forall t' \geq t : p \in H(q, t')$$

Where:

- F is a function that to each time t associates the set of processes that crashed at time t.
- faulty(F) is the set of processes that eventually crash.
- correct(F) is the set of processes that never crash.
- H(p, t) is the output of the failure detector of process p at time t.

Exercise 1b (3/3)

Therefore, the two failure detectors differ in terms of their accuracy.

- A correct process is never suspected by any correct process in a perfect failure detector,
- In an eventually perfect failure detector correct processes may transiently be suspected by other correct processes.

Exercise 1c (1/4)

It is impossible to implement Terminating Reliable Broadcast without a Perfect Failure Detector.
In order to show this, we use a sequence of instances of Terminating Reliable Broadcast to implement a Perfect Failure detector for a designated process s.

Code for s:

Upon event <s, Init>:
    Initialize trb[0]
    Trigger <trb[i].Broadcast, “I live!”>

Upon event <trb[i].Deliver, “I live!”>
    Initialize trb[i + 1]
    Trigger <trb[i + 1].Broadcast, “I live!”>

Exercise 1c (2/4)

Code for p ≠ s:

Upon event <trb[i].Deliver, “I live!”>
    Initialize trb[i + 1]

Upon event <trb[i].Deliver, φ>
    Trigger <P.Crash, s>

Exercise 1c (3/4)

Strong accuracy

If s is correct then, for all i, we have that every correct eventually trb[i].Delivers “I live!”.
We prove this by induction:

- Upon initializing, s trb[0].Broadcasts “I live!”.
- By Validity and Agreement of Terminating Reliable Broadcast, if s trb[i].Broadcasts “I live!”, every correct process eventually trb[i].Delivers “I live!”.
- Upon trb[i].Delivering “I live!”, s trb[i + 1].Broadcasts “I live!”.

Therefore, no correct process ever (i.e., for any i) trb[i].Delivers φ. Since a correct process triggers P.Crashes s only upon trb[i].Delivering φ, no correct process ever suspects s.

Exercise 1c (4/4)

Strong completeness

If s is faulty, then for some i s never trb[i].Broadcasts “I live!”. By contradiction, if s trb[i].Broadcasts “I live” for every i, then i never crashes, i.e., at no point in time s stops executing.

By the Termination of Terminating Reliable Broadcast, every correct process eventually trb[i].Delivers a message. By the Integrity of Terminating Reliable Broadcast, every correct process eventually trb[i].Delivers φ. As a result, every correct process eventually suspects s.

Exercise 1d

State the interface and properties of Consensus:

Interface:

- Propose(v): Proposes value v for consensus
- Decide(v): Outputs a value v of consensus

Properties:

- *Termination*: Every correct process eventually decides some value.
- *Validity*: If a process decides v, then v was proposed by some process.
- *Integrity*: No process decides twice.
- *Agreement*: No two correct processes decide differently.

Exercise 1e (1/3)

Provide an algorithm A that implements consensus among N processes, subject to an arbitrary number of failures, using at most N instances of Terminating Reliable Broadcast. Prove the correctness of A.

Algorithm:

```
Upon event <c, Init>:
        proposed := []      // List of pairs (process_id, proposed_value)

        for j in 1 to N:
                Initialize TRB_j     //  TRB_j is an instance of Terminating Reliable Broadcast,
                                     //   in which process with id=j is the designated sender.
```

Exercise 1e (2/3)

Algorithm (cont'd):

upon event <c, Propose | v >:
    Trigger <TRB_i, Broadcast | (i, v)>

upon event <TRB_j, Deliver | m>:
    proposed.append(m) // Remember that m has the form <process_id, proposed_value> or
                        // the symbol φ (if the broadcasting process crashed)

upon event length(proposed) == n:
    removed_φ := filter(lambda x: x != φ, proposed) // Remove φ entries from the list
    min_pid_pair := min(removed_φ, key=lambda x: x.first) // Find the pair with the
                                                           // minimum process id
    return min_pid_pair.second

Note: Any deterministic function on proposed can be used instead of *min*.

Exercise 1e (3/3)

Proof of correctness:

- *Termination:* Every correct process terminates if it appends N messages to proposed and the list does not contain only φ.
  - The messages appended are the messages delivered by the TRB instances. By the termination property of TRB, the process (being correct) will deliver exactly one message per TRB instance. Therefore the process will append N messages in proposed.
  - By the Validity property of TRB, the correct process that terminates will have its own Broadcast value in the proposed list. Hence this list contains at least one non-φ value.
- *Validity*: The decision is a value from the proposed list. Every value in that list is a value delivered by an instance of the TRB. Processes propose values by broadcasting in an instance of TRB.
- *Integrity*: A process decides when the length of proposed becomes N. This can only occur once, since we only append in the proposed list.
- *Agreement*: Two correct processes have identical proposed lists. This is guaranteed by the Termination and agreement property of TRB.

Exercise 2a

State the interface and properties of Best-Effort-Broadcast:

Interface:

- Broadcast(m): Broadcasts a message to all processes.
- Deliver(p, m): Delivers a message m broadcast by process p.

Properties:

- *Validity*: If a correct process broadcasts a message m, then every correct process eventually delivers m.
- *No duplication*: No message is delivered more than once.
- *No creation*: If a process delivers a message m with sender s, them m was previously broadcast by process s.

Exercise 2b (1/2)

Provide an algorithm A that implements Best-Effort-Broadcast among N processes, subject to an arbitrary number of failures such that: the maximum per-process communication complexity of A is O(N); the latency of A is O(1). Prove the correctness of A.

Algorithm:

We rely on an instance of Perfect Links pl.

```
upon event <beb, Broadcast | m>:
        forall q in Π do:
                trigger <pl, Send | q, m>

upon event <pl, Deliver | p, m>:
        trigger <beb, Deliver | p, m>
```

Exercise 2b (2/2)

Correctness proof:

- The No-duplication/No-creation properties of BEB are satisfied by the corresponding properties of Perfect Links.
- By the Reliable delivery property of the Perfect Links and the fact that:
  - The sender sends the message to all
  - Every correct process that receives a message Delivers that message.

Complexity:

- The sender needs to contact all other processes, therefore the communication complexity is O(N). Other processes do not send any message. Therefore the *maximum* communication complexity is O(N).
- The latency is 1 unit of time (O(1)), because the sender sends the messages in a for-loop. Computation is instantaneous, therefore it is as if the messages are sent in parallel.

Exercise 2c (1/6)

Provide an algorithm A that implements Best-Effort-Broadcast among N processes, under the assumption that only the sender process can crash, such that: The maximum per-process communication complexity if A is O(1); the latency of A is O(logN). Prove the correctness of A.

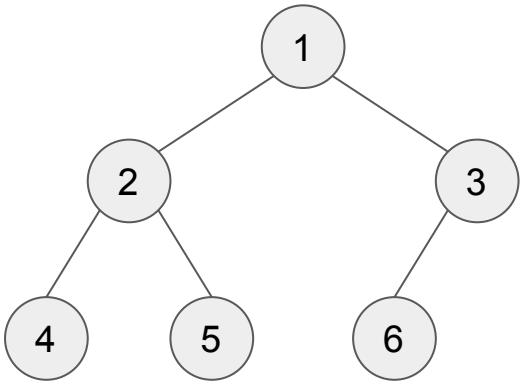
Without loss of generality, let p_1 be the designated sender. We arrange the processes in a complete binary tree topology. Process p_1 is at the root of the tree.

NB. A complete binary tree is a binary tree in which every level, except possibly the last, is completely filled, and all nodes are as far “left” as possible.

We assume that each process is aware of the location of processes in the tree, therefore each process knows which are its children (at most two).

Exercise 2c (2/6)

In an complete binary tree we can define the following encoding, which makes the pseudocode much simpler (not necessary for a complete solution):



For a node at position Π[i] of the array:

- The parent of Π[i] is Π[i/2]
- The left child of Π[i] is Π[2i]
- The right child of Π[i] is Π[2i+1]

Exercise 2c (3/6)

Given the previous encoding, the algorithm we propose is the following:

```
upon event <beb, Broadcast | m>: // Executed only by designated process p_1.
        Trigger <pl, Send | Π[1]>              // Send to itself


upon event <pl, Deliver | p, m>:
        if 2*p ≤N :
                trigger <pl, Send | Π[2*p], m>        // Send to children if they exist

                if 2*p+1 ≤N :
                        trigger <pl, Send | Π[2*p+1], m>     // If the left children doesn't exist,
                                                             // then the right children doesn't exist either.

        trigger <beb, Deliver | m>             // Deliver
```

Exercise 2c (4/6)

Correctness proof:

- The no duplication and no creation property of the the broadcast are ensured by the perfect links abstraction and the fact that no process forwards a message it did not receive.
- Due to our assumption, the validity property becomes: If the sender is correct and broadcasts a message m, then every other process eventually delivers the message m.
- If the sender doesn’t fail, then the children of the sender will receive the message. Since no other process can fail, the message will be disseminated to all processes.
- More precisely, every process will receive the message from its parent, which (recursively) will receive the message from its own parent, etc. Every process that receives the message also delivers it (because it is correct).
    - Note: The curious student can prove this formally by using induction on the length of the path from the root to the node of interest.

Exercise 2c (5/6)

Height of binary tree: is the largest number of edges in a path from the root node to a leaf node.

Perfect binary tree: A perfect binary tree is a binary tree in which all interior nodes have two children and all leaves have the same depth.

Lemma: A perfect binary tree of height H has $2^{H+1} - 1$ nodes ($H \geq 0$).

Proof:
Let's denote the number of nodes in level i by N_i. Then following recursive formula holds:

$N_0 = 1$
$N_i = 2N_{i-1}$, $i>0$

We can prove easily by induction that the closed form of this formula is $N_i=2^i$.

Therefore a tree of height H has
$S=N_0 + N_1 + \ldots + N_H = 2^0 + 2^1 + \ldots + 2^H$= (geometric progression)= $2^{H+1} - 1$ nodes

Exercise 2c (6/6)

Using the previous lemma, we see that a complete binary tree of height H>0 has:

- More nodes than a perfect binary tree of height H-1. Therefore, it has > 2^H-1 nodes
- No more nodes than a perfect binary tree of height H. Therefore, it has ≤2^{H+1}-1 nodes.

Also, a complete binary tree of H=0 has always 1 node.

As a result, a complete binary tree of N nodes has a height H(N) that satisfies the following inequality: $\log_2(N+1) - 1 \leq H(N) \leq \log_2(N) \Rightarrow H(N) = \lfloor \log_2(N) \rfloor$

Therefore, the latency of the algorithm is $O(\log_2(N))$.

Furthermore, each process sends the message to at most two other processes (its children), therefore the communication complexity of each process is O(1).

Exercise 2d (1/5)

Provide an algorithm A that implements Best-Effort-Broadcast among N processes, under the assumption that at most f << N can crash, such that: The maximum per-process communication complexity if A is O(f); the latency of A is O(logN - logf). Prove the correctness of A.

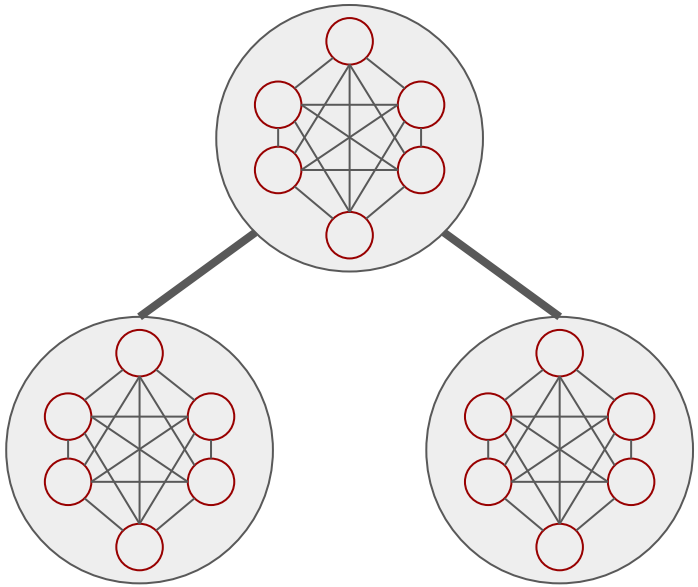
With Exercise 2b, we proved that we can implement Best-Effort Broadcast among N processes, subject to an arbitrary number of failures, with O(N) communication complexity and O(1) latency. We do so by having the sender broadcast the message to every process in the system.

With Exercise 2c, we proved that we can implement Best-Effort Broadcast among N processes, if all the “gossiping” processes are correct, by organizing them into a complete binary tree.

In order to achieve validity, we organize our processes in $\lceil N/(f + 1) \rceil$groups of (f + 1) processes. We then organize the groups in a complete binary tree. A correct process forwards the messages it receives to the members in: its group; the two groups that are children of its group.

Exercise 2d (2/5)

In an our topology we define the following encoding, which makes the pseudocode much simpler (not necessary for a complete solution):



- The group of Π[i] is $\lfloor i/(f + 1) \rfloor$.
- The elements of group k G(k) are {(f + 1)k, (f + 1)k + 1, … (f + 1)k + f}.
- The left child of of G(k) is G(2k).
- The right child of G(k) is G(2k + 1).

Exercise 2d (3/5)

```
upon event <beb, Init>
        delivered = false

upon event <beb, Broadcast | m>:  // Executed only by designated process Π[1].
        Trigger <pl, Send | Π[1]>                // Send to itself

upon event <pl, Deliver | p, m>: // On process Π[i].
        k =⌊i/(f + 1)⌋
        for all g in {k, 2k, 2k + 1}
                for all j in {(f + 1)g, …, (f + 1)g + f}
                        if j <= N
                                trigger <pl, Send | Π[j], m>
        if delivered = false
                delivered = true
                trigger <beb, Deliver | Π[1], m>
```

Exercise 2d (4/5)

Integrity follows from the integrity of perfect links and the fact that no process forwards a message it didn't previously receive.

No duplication follows from the use of the delivered flag.

Let us assume that p_1 is correct. We prove **Validity** as follows:

- If any correct process in group k eventually delivers m, then every correct process in groups k, 2k and 2k+1 eventually delivers m. This follows from the fact that a correct process in group k forwards m to all the members of groups k, 2k, 2k+1 and that links are perfect.
- In every group there is at least one correct process. This follows from the observation that (f + 1) > f and the pigeonhole principle.

The **communication complexity** of the algorithm is O(f): indeed, every correct process receives at most (f + 1) messages and sends at most (3f + 3) messages.

Exercise 2d (5/5)

Following from what we proved in Exercise 2c, the **latency** of the algorithm is log(G), where G <= N/(f + 1) is the number of groups. Indeed, if a correct process in group k delivers m, then every correct process in groups 2k and 2k+1 delivers m within one message delay.

The result immediately follows from O(log(N/(f + 1)) = O(logN - logf).