Dependable Distributed Systems
Master of Science in Engineering in Computer Science

AA 2022/2023

Lecture 27 – Exercises
November 30th, 2022

Ex 1: Let us consider a distributed system composed of N processes executing the algorithm reported in figure

---

Algorithm 3.12: Broadcast with Sequence Number

Implements:
FIFOReliableBroadcast, **instance** *frb*.

Uses:
ReliableBroadcast, **instance** *rb*.

upon event $\langle$ *frb*, *Init* $\rangle$ **do**
  $lsn := 0$;
  $pending := \emptyset$;
  $next := [1]^N$;

upon event $\langle$ *frb*, *Broadcast* | m $\rangle$ **do**
  $lsn := lsn + 1$;
  trigger $\langle$ *rb*, *Broadcast* | [DATA, *self*, m, *lsn*] $\rangle$;

upon event $\langle$ *rb*, *Deliver* | p, [DATA, s, m, sn] $\rangle$ **do**
  $pending := pending \cup \{(s, m, sn)\}$;
  while exists $(s, m', sn') \in pending$ such that $sn' = next[s]$ **do**
    $next[s] := next[s] + 1$;
    $pending := pending \setminus \{(s, m', sn')\}$;
    trigger $\langle$ *frb*, *Deliver* | s, m' $\rangle$;

Let us assume that

1. up to f processes may be Byzantine faulty and
2. A Byzantine process is not able to compromise the underline Reliable Broadcast primitive (i.e., when a Byzantine process sends a message through the rbBroadcast interface, the message will be reliably delivered to every correct process).

For each of the following properties, discuss if it can be guaranteed when f=1 and motivate your answer (also by using examples)

- *Validity:* If a correct process *p* broadcasts a message *m*, then *p* eventually delivers *m*.
- *No duplication:* No message is delivered more than once.
- *No creation*: If a process delivers a message *m* with sender *s*, then m was previously broadcast by process *s*.
- *Agreement:* If a message *m* is delivered by some correct process, then *m* is eventually delivered by every correct process.
- *FIFO delivery:* If some process broadcasts message *m1* before it broadcasts message *m2*, then no correct process delivers *m2* unless it has already delivered *m1*.

Ex 2: Let us consider a distributed system composed of N processes executing the algorithm reported in figure

---

Algorithm 3.2: Lazy Reliable Broadcast

Implements:
ReliableBroadcast, **instance** *rb*.

Uses:
BestEffortBroadcast, **instance** *beb*;
PerfectFailureDetector, **instance** $\mathcal{P}$.

upon event $\langle$ *rb*, *Init* $\rangle$ **do**
    correct := Π;
    from$[p]$:= $[\emptyset]^N$;

upon event $\langle$ *rb*, *Broadcast* $|$ m $\rangle$ **do**
    trigger $\langle$ *beb*, *Broadcast* $|$ [DATA, *self*, m] $\rangle$;

upon event $\langle$ *beb*, *Deliver* $|$ p, [DATA, s, m] $\rangle$ **do**
    if $m \notin$ *from*$[s]$ **then**
        trigger $\langle$ *rb*, *Deliver* $|$ s, m $\rangle$;
        from$[s]$:= *from*$[s] \cup \{m\}$;
        if $s \notin$ *correct* **then**
            trigger $\langle$ *beb*, *Broadcast* $|$ [DATA, s, m] $\rangle$;

upon event $\langle$ $\mathcal{P}$, *Crash* $|$ p $\rangle$ **do**
    correct := *correct* $\setminus \{p\}$;
    forall $m \in$ *from*$[p]$ **do**
        trigger $\langle$ *beb*, *Broadcast* $|$ [DATA, p, m] $\rangle$;

Let us assume that up to f processes may be Byzantine faulty with a symmetric behaviour i.e., a Byzantine process can change the content of the message it is going to send, but it cannot send different values to different processes when invoking the bebBroascast (they can lie but in a consistent way).

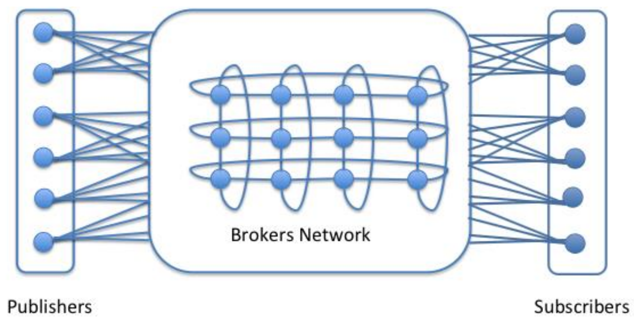
For each of the following properties, discuss if it can be guaranteed when f=1 and motivate your answer (also by using examples)

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

Ex 3: Consider a distributed system constituted by *n* processes $\prod=\{p_1, p_2 \ldots p_n\}$ with unique identifiers that exchange messages through perfect point-to-point links and are arranged in a unidirectional ring (i.e., each process p_i can exchange messages only with process $p_{(i+1) \bmod n}$ and stores its identifier in a local variable `next`).
Each process p_i knows the initial number of processes in the system (i.e., every process p_i knows the value of *n*).

1. Assuming that processes are not going to fail, write the pseudo-code of an algorithm that implements a consensus primitive
2. Let's now assume that processes may crash and that each correct process has access to a perfect failure detector. Discuss the issues of the implementation provided in point 1 and describe how you should modify the algorithm to make it fault tolerant
3. Considering the algorithm proposed in point 1, discuss which properties may be compromised by the presence of one Byzantine process in the ring.

Ex 4: Let us consider a distributed system composed by publishers, subscribers and brokers. Processes are arranged in a network made as follows and depicted below:



1. Each publisher is connected to k brokers through perfect point-to-point links;
2. Each subscriber is connected to k brokers through perfect point-to-point links;
3. Each broker is connected to k brokers through perfect point-to-point links and the resulting broker network is k-connected [1](4-connected in the example);

Answer to the following questions:

1. Write the pseudo-code of an algorithm (for publisher, subscribers and broker nodes) implementing the subscription-flooding dissemination scheme assuming that processes are not going to fail.
2. Discuss how many crash failures the proposed algorithm can tolerate.
3. Modify the proposed algorithm in order to tolerate f Byzantine processes in the broker network and discuss the relation between f and k.