Distributed Algorithms, Final Exam Solution

January 2016

Part I
Broadcast (12 points)

Answer the following questions in the provided spaces.

Question 1 (5x1 = 5 points)

Which of the following properties are safety properties and which are liveness? Mark an "S" or "L" next to each property **clearly** with a *one line* explanation.

1. For any message m_1 delivered by a process p_1, if there exists another process p_2 that delivered m_1 and then delivered a message m_2, then p_1 also delivers m_2. <u>L</u>

2. For every pair of processes p_1 and p_2 that deliver a message m_1, if p_1 delivers a message m_2 before delivering m_1, then p_2 also delivers m_2 before delivering m_1. <u>S</u>

3. Every process that crashes is eventually detected. <u>L</u>

4. No process is detected before it crashes. <u>S</u>

5. No two processes deliver the same message. <u>S</u>

Question 2 (3x1 = 3 points)

Mention the properties for the following:

1. Causal broadcast (C)
   Answer:
   CO1. Validity: If p_i and p_j are correct, then every message broadcast by p_i is eventually delivered by p_j.

   CO2. No duplication: No message is delivered more than once.

   CO3. No creation: No message is delivered unless it was broadcast

CO4. Agreement: For any message m, if a correct process delivers m, then every correct process delivers m.

CO5: **Causal order:** If any process p_i delivers a message m_2, then p_i must have delivered every message m_1 such that $m_1 -> m_2$.

2. Total order broadcast (T)
Answer:
TOB1. Validity: If p_i and p_j are correct, then every message broadcast by p_i is eventually delivered by p_j.

TOB2. No duplication: No message is delivered more than once.

TOB3. No creation: No message is delivered unless it was broadcast

TOB4. Agreement: For any message m, if a correct process delivers m, then every correct process delivers m.

TOB5. **Total order:** The processes must deliver all messages according to the same order (i.e., the order is now total).

3. Uniform reliable broadcast (U)
Answer:
URB1. Validity: If p_i and p_j are correct, then every message broadcast by p_i is eventually delivered by p_j.

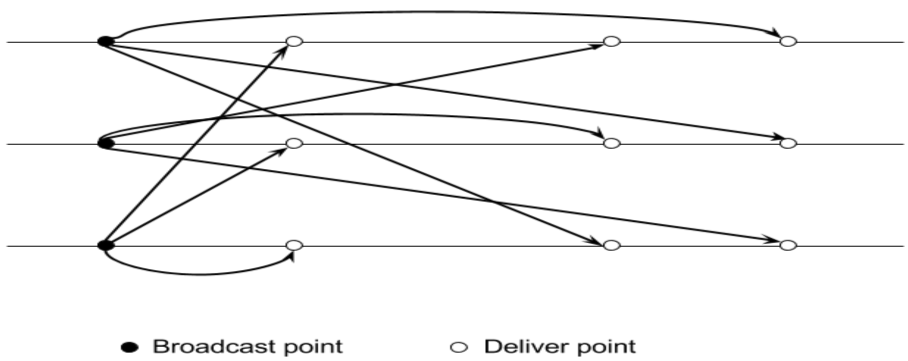
URB2. No duplication: No message is delivered more than once.

URB3. No creation: No message is delivered unless it was broadcast

URB4. **Uniform Agreement:** For any message m, if a process delivers m, then every correct process delivers m.

Question 3 (2 points)

Consider the broadcast executions shown below. Which of the following statements are true about this execution? Use the notations from question 2.

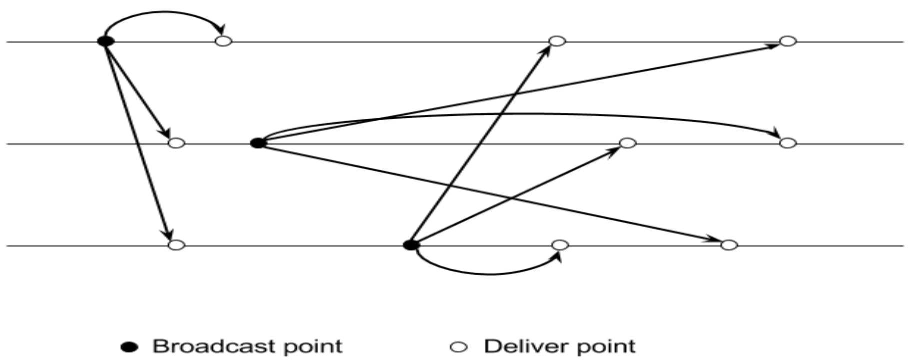


1. C
2. T
3. U
4. C & T
5. T & U
6. C & U
7. C & T & U
8. None of the above.

Answer: C & U (6)

Question 4 (2 points)

Consider the broadcast executions shown below. Which of the following statements are true about this execution? Use the notations from question 2.



1. C
2. T
3. U
4. C & T
5. T & U
6. C & U
7. C & T & U
8. None of the above.

Answer: C & T & U (7)

Part II

Consensus, NBAC, TRB, GM (15 points)

Question 1 (5x2 points)

Indicate which sentence is true and which is false with a T and F about three different consensus algorithms, respectively. Explain your answer.

Algorithm I (nonuniform consensus with P): The processes exchange and update proposals in rounds and decide on the value of the non-suspected process with the smallest id.

Algorithm II (uniform consensus with P): The processes exchange and update proposal in rounds, and decide after n rounds.

Algorithm III (uniform consensus with majority): The processes alternate in the role of a coordinator until one of them succeeds in imposing a decision.

1. Algorithm I can satisfy uniform consensus if we use Uniform Reliable Broadcast instead of Best Effort Broadcast.
   Answer: F.

2. In algorithm II, the decided value is proposed by the correct process with lowest ID.
   Answer: F.

3. Using a $\diamond P$ in algorithm II does not violate Agreement property but Termination.
   Answer: F.

4. In algorithm III, if p_i is leader and decides, the decided value is actually proposed by p_i.
   Answer: F.

5. Algorithm III finishes in at least N rounds because a process processes its value in a round only if it is the leader in that round.
   Answer: F. If all the processes are correct and none of them is suspected in the first round, the leader can impose a decision in the first round.

Question 2 (4x0.5 point)

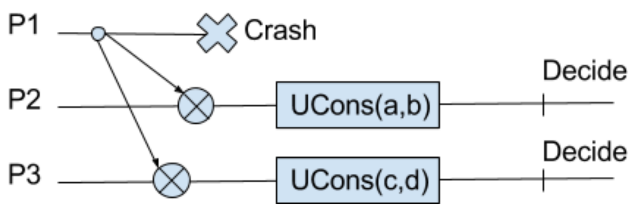
Explain the difference between each of the following pairs of abstractions in terms of their respective properties.

1. Consensus and Non Blocking Atomic Commit (NBAC).
2. Reliable Broadcast and Terminating Reliable Broadcast.
3. Group Membership and perfect failure detector (P).
4. View Synchrony and uniform view Synchrony.

Question 3 (3x1 point)

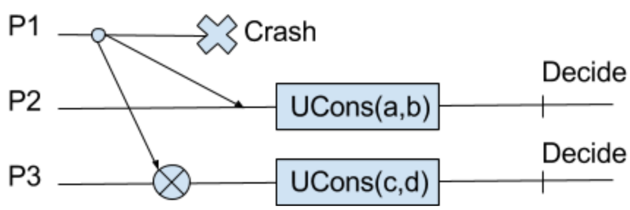
Each of the below executions represents a NBAC abstraction implemented using Uniform Consensus. All the processes propose 1 in the beginning. Write all the possible values (0, 1, or 0/1) for **a, b, c** and **d** in each execution. ($\otimes$ shows the sender is detected to have crashed and the message is lost.)

1.



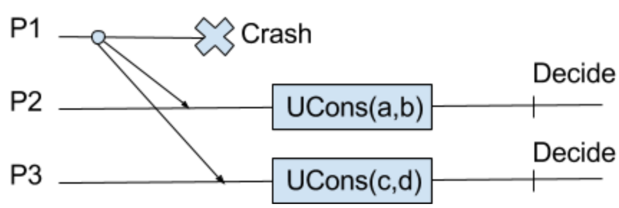
Answer: $a = b = c = d = 0$

2.



Answer: $a = 1, c = 0, b = d = 0/1$

3.



Answer: $a = b = c = d = 1$

Part III

Process Failures and Shared Memory (13 points)

Question 1

Consider the fail-silent system model, i.e., where processes may fail by crashing, and the failure detection *cannot* be detected, so no failure detector can be used.

1. Does any implementation of a regular register require a majority of correct processes in this case? Explain your answer.
2. Now suppose an eventually perfect failure detector is available – is a correct majority still required? Explain your answer.

Answer

See solution 4.8 in the course book, pg. 197-198.

Question 2

Mark each of the following statements with either:

`T`, if the statement is true, or

`F`, if that statement is false.

1. Consider any algorithm running in system of n total processes. The algorithm does not make use of any failure detector, thus this system must comprise at least $n = 2 \cdot f + 1$ processes in order to tolerate a predefined subset of f processes crashing. [F]
2. The Uniform Reliable Broadcast abstraction ensures that, despite a process crashing while it broadcasts a message m, the correct processes still eventually deliver m. [F]
3. A regular register execution where no crashes occur is equivalent with an atomic register execution. [F]
4. Consider a Reliable Broadcast algorithm. If all sending processes are assumed to be correct, then this algorithm actually satisfies the Uniform variant of Agreement. [T]

5. If sending processes in a Causal Broadcast algorithm are assumed to be correct, then this algorithm actually satisfies the Uniform Causal Broadcast specification. [T]

Question 3. A regular register array.

(4.5 points)

Traditional registers, as defined in the class, are meant to store one single object (e.g., a value). A *register array*, in contrast, is able to store multiple objects.

We consider a regular register array algorithm $\mathcal{A}$ which relies on m underlying regular registers, $[R_0, ..., R_{m-1}]$. Each underlying register stores an object. We assume that all objects have the form of $\langle key_i/value_i \rangle$ tuples, with $i \in \{0, .., m-1\}$. Thus, each object is uniquely associated with one of the underlying registers, such that object with key key_i is associated register R_i.

Additionally, we require that $\mathcal{A}$ runs on N processes, $p_0, .., p_{N-1}$, where $N = M \cdot 2$. To ensure reliability, this algorithm guarantees that every underlying register is replicated at two processes, as follows. Processes are first grouped in pairs, $\{p_0, p_1\}, \{p_2, p_3\}$, ..., $\{p_{N-2}, p_{N-1}\}$. And then each process pair replicates one of the registers, such that the first pair of processes handles (i.e., replicates) register R_0, the second pair handles register R_1, and so on.

For handling any `read` or `write` operation on a key key_i, a process either triggers that operation on its underlying register (if key_i is associated to its underlying register), or it forwards the operation to one of the two processes handling the register associated with key_i.

The specification of a regular register array is as follows:
Name: Regular register array.
Properties: Same as properties *Termination* and *Validity* of regular registers.
Requests:

`read(`k_i`)` $\rightarrow v_i$,
`write(`k_i, v_i`)`, where k_i is they key of the object, and v_i is the value.

Give an implementation of $\mathcal{A}$, assuming that the m underlying regular registers are already provided.

Algorithm 1 A regular register array.

```
Implements:
    Regular register array (A)
Uses:
    RegularRegister (rr_i)
upon event ⟨read | k_x⟩ at process p_i do
    if (i = 2 · x)||(i = 2 · x + 1) then
        trigger ⟨rr_i, read | k_x⟩
    else
        p = x * 2 # p is one of the processes handling k_i
        forward ⟨A, read | k_x⟩ to process p
    end if

upon event ⟨write | k_x, v⟩ at process p_i do
    if (i = 2 · x)||(i = 2 · x + 1) then
        trigger ⟨rr_i, write | k_x, v⟩
    else
        p = x * 2 # p is one of the processes handling k_i
        forward ⟨A, write | k_i, v⟩ to process p
    end if
```