Distributed Algorithms

Midterm Exam

28^{th} November, 2016

Name: _____

Sciper number: _____

Time Limit: 1 hour 45 minutes

Instructions:

- This exam is closed book: no notes, electronics, nor cheat sheets allowed.
- Write your name and SCIPER on *each page* of the exam.
- If you need additional paper, please ask one of the TAs.
- Read through each problem before starting to solve.
- When solving a problem, do not assume any known result from the lectures, unless it is explicitly stated that you might use some known result.

Good Luck!

Problem	Max Points	Score
1	14	
2	14	
3	14	
4	14	
Total	56	

1 Broadcast (*14 points*)

1.1 Question 1

1. Provide the properties of (a) Uniform reliable broadcast (U), and (b) an eventually perfect failure detector along with a brief and concise description of each property.

   Answer:
   Uniform reliable broadcast (U):
   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.

   Eventually perfect failure detector:
   EPFD1: Strong completeness: Eventually, every process that crashes is permanently suspected by every correct process.

   EPFD2: Eventual strong accuracy: Eventually, no correct process is suspected by any correct process.

2. Can we devise a uniform reliable broadcast algorithm with an eventually perfect failure detector but without assuming a majority of correct processes? If yes, then explain how to devise such an algorithm. If no, then give a counter-argument.

   Answer: No, a majority of correct processes is necessary. We explain why this is the case, using a system of four processes p, q, r, s by giving a so-called partitioning argument. Suppose it could indeed be implemented in this system when two out of the four processes may fail.

   Consider an execution where process p urb-broadcasts a message m and assume that r and s crash in that execution without receiving any message either from p or from q. Because of the validity property of uniform reliable broadcast, there must be a time t at which p urb-delivers message m.

   Consider now an execution that is similar to this one except that p and q crash right after time t, but r and s are correct. However, r and s have been falsely suspected by the failure detector at p and q, which is possible because the failure detector is only eventually perfect. In this execution, p has urb-delivered a message m whereas r and s have no way of knowing about the existence of m and they never urb-deliver it. This violates the uniform agreement property and shows that a majority of correct processes is necessary.

1.2 Question 2

1. Provide the properties of (a) Total order broadcast (T), and (b) Causal broadcast (C) along with a brief and concise description of each property.

   Answer:
   Total order broadcast (T):

   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).

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

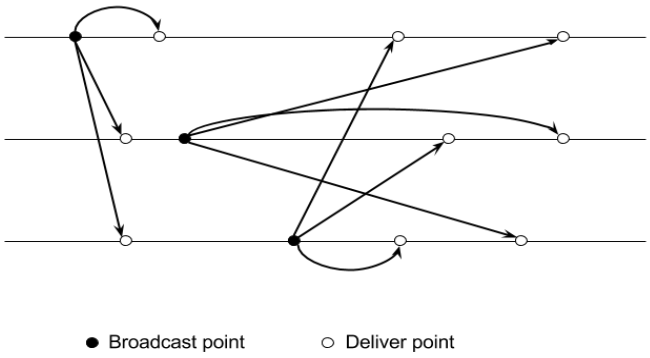
   CO2. No duplication: No message is delivered more than once.

   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. Consider the broadcast executions shown below. Which of the following statements are true about this execution? Use the notations from questions 1.1 and 1.2.



(a) C
(b) T
(c) U
(d) C & T
(e) T & U
(f) C & U
(g) C & T & U
(h) None of the above.

Answer: C & T & U (g)

2 Safety/Liveness and NBAC (*14 points*)

2.1 Question 1

Mark S for safety and L for liveness, X for neither. Justify your answer.

1. If a process p broadcasts a message m, then p eventually delivers m.

   S. if p crashes before delivering, the ppty is violated and cannot be satisfied later.

2. If a process p crashes, all other processes eventually stop working.

   L. if p crashes and some other process q keeps working, the property is violated but will be satisfied later, as soon as q stops.

3. If a process p crashes, then no other process stops working.

   S. If p crashes and some process q stops, the property is violated for ever.

4. If a process p broadcasts a message m, then every correct process delivers m.

   L. If a process q does not deliver m, the property is violated, but can be satisfied as soon as q crashes.

5. If a particular process p broadcasts a message m, then no correct process delivers m.

   L. like the previous case, the property can be satisfied as soon as q crashes.

6. If a particular process p broadcasts a message m, then no process delivers m.

   S. in this case once violated, the property is violated forever because we would like that no other process (even if it crashes later) delivers the message from the particular process p.

2.2 Question 3

You saw in the class that you could use NBAC to implement a perfect failure detector (when one process can crash).

1. Recall the specifications of NBAC (no need to name them exactly).

NBAC1. (Agreement): No two processes decide differently. NBAC2. (Termination): Every correct process eventually decides. NBAC3. (Commit-Validity): 1 can only be decided if all processes propose 1. NBAC4. (Abort-Validity): 0 can only be decided if some process crashes or votes 0.

2. Which of those specifications you could tweak to obtain only an eventually perfect detector? (justify on how you could guarantee eventual eventual accuracy instead of strong accuracy)

Eventual accuracy: add "eventually" before NBAC4.

3 Consensus (*14 points*)

3.1 Question 1

Explain the four properties of consensus. Give four executions, each of which violates exactly one of the consensus properties.
Answer:
C1. Validity: Any value decided is a value proposed.
Validity violation: p_1 and p_2 propose 1. p_1 and p_2 decide 0.
C2. Agreement: No two processes decide differently.
Agreement violation: p_1 proposes 1 and p_2 proposes 0. p_1 decides 1 and p_2 decides 0.
C3. Termination: Every correct process eventually decides.
Termination violation: p_1 proposes 1 and p_2 proposes 0. p_1 decides 1 and p_2 never decides.
C4. Integrity: No process decides twice.
Integrity violation: p_1 proposes 1 and p_2 proposes 0. p_1 decides 1. p_2 decides 1 twice.

3.2 Question 2

Algorithm 1 implements a consensus using a perfect failure detector and best effort broadcast (beb). How would you change Algorithm 1 to make a uniform consensus? Explain your changes and apply them to the code.
Answer:
In Algorithm 1, a process decides in its corresponding round. First of all, we make the processes only broadcast their current value and not decide on its corresponding round. Secondly, the processes decide after exactly n round to ensure the uniform consensus. So, we go to the next round till $round == n$ and the process has not decided yet (indicated as a new parameter *decided*). The changes are made in Algorithm 2.

Algorithm 1 Consensus Using a Perfect Failure Detector and Beb

Upon event $< Init >$ **do**

1: $suspected = \emptyset$
2: $round = 1$
3: $currentProposal = nil$
4: $broadcast = false$
5: $delivered[] = false$

Upon event $< Crash, p_i >$ **do**

1: $suspected = suspected \cup \{p_i\}$

Upon event $< Propose, v >$ **do**

1: **if** $currentProposal == nil$ **then**
2: $\quad currentProposal = v$
3: **end if**

Upon event $< bebDeliver, p_{round}, value >$ **do**

1: $currentProposal = value$
2: $delivered[round] = true$

Upon event $delivered[round] == true$ **or** $p_{round} \in suspected$ **do**

1: $round = round + 1$

Upon event $p_{round} == self$ **and** $broadcast == false$ **and** $currentProposal \neq nil$

1: **trigger** $< Decide, currentProposal >$
2: **trigger** $< bebBroadcast, currentProposal >$
3: $broadcast = true$

Algorithm 2 Uniform Consensus Using a Perfect Failure Detector and Beb

Upon event $< Init >$ **do**

1: $suspected = \emptyset$
2: $round = 1$
3: $currentProposal = nil$
4: $decided = false$
5: $broadcast = false$
6: $delivered[] = false$

Upon event $< Crash, p_i >$ **do**

1: $suspected = suspected \cup \{p_i\}$

Upon event $< Propose, v >$ **do**

1: **if** $currentProposal == nil$ **then**
2: $\quad currentProposal = v$
3: **end if**

Upon event $< bebDeliver, p_{round}, value >$ **do**

1: $currentProposal = value$
2: $delivered[round] = true$

Upon event $delivered[round] == true$ **or** $p_{round} \in suspected$ **do**

1: **if** $round == n$ **and** $decided == false$ **then**
2: $\quad$ **trigger** $< Decide, currentProposal >$
3: $\quad decided = true$
4: **else**
5: $\quad round = round + 1$
6: **end if**

Upon event $p_{round} == self$ **and** $broadcast == false$ **and** $currentProposal \neq nil$

1: **trigger** $< bebBroadcast, currentProposal >$
2: $broadcast = true$

4 Terminating Reliable Broadcast (TRB) and Group Membership (GM) (*14 points*)

4.1 Question 1 – Multiple choice questions

For each of the items below, identify all the choices which are correct (true).

1. The perfect failure detector P in relation with TRB:

   (a) P is both necessary and sufficient for the TRB algorithm.

   (b) P is not necessary for the TRB algorithm.

   (c) P is necessary for the TRB algorithm.

   (d) None of the above.

   Answer: (a) and (c).

2. The GM abstraction ensures:

   (a) The **uniform variant of agreement** on each view (j, M) which they install.

   (b) **Integrity**: If a process p installs a view (j, M), then $p \in M$.

   (c) **Validity**: Once a process installs a view (j, M), then any process p, such that $p \notin M$, is a crashed process.

   Answer: (a), (b), (c).

4.2 Question 2

1. Explain the difference between the **Agreement** and the **Uniform Agreement** properties in the context of Terminating Reliable Broadcast (TRB).

   Answer: In the non-uniform version of agreement, a process may deliver m (or φ) and then crash, without requiring correct processes to also deliver m (or φ).

2. In the description of the TRB problem, it is stated that:

   > Process *src* [i.e., the broadcasting process] is supposed to broadcast a message m (distinct from φ).

   Explain the purpose of the special message φ by answering these questions:

   A. How would a process interpret the delivery of this message φ?

   B. Why is it important that the broadcast message m <u>is distinct</u> from the special message φ? Namely, sketch or explain an execution of a problematic situation that can arise if $m == \varphi$. Explain, in this context, which property – if any – of TRB breaks in such a case.

   Answer:
   A. The process interprets the delivery of φ as the failure of the sender to successfully broadcast the indented a message m. In most situations, this is interpreted as the fact that the sender *src* crashed.
   B. In case $m == \varphi$ is allowed by TRB, then it is possible that a process believes the sender to have crashed, and consequently treat m as a notification that a failure occurred in the system – but no failures actually occurred. This ambiguity would prevent the implementation of useful applications on top of TRB: this abstraction becomes useless in many cases, e.g., to implement failure detection. Surprisingly, none of the four properties of GM breaks, but higher-level application semantics may break if the application expects φ to be a special message.

3. Consider a modified, weaker algorithm for GM, called wGM, which lacks the **Local Monotonicity** property of GM.

   A. Describe the properties of wGM.

   B. Explain briefly the different between wGM and the perfect failure detector P.

   Hint: Informally, **Local Monotonicity** ensures that subsequent views have increasing view numbers and smaller process sets.

   Answer:
   A. Agreement, Completeness, and Accuracy – see their complete definitions in the class' slides.
   B. The most important distinction is the lack of coordinated views in P; this is given by the Agreement property in wGM, which is inherited from GM.

4. Consider a distributed algorithm that runs on small sensors and builds on the GM protocol and the uniform reliable broadcast (urb) protocol. A sensor may fail at any time with crash-stop. This sensor algorithm uses GM to maintain the set of active processes and it uses urb to periodically exchange *sensor data* (i.e., signals about their surroundings) among all the active processes. Is it possible, given the properties provided by GM, for a process to urb.deliver sensor data which was sent from some process p, after p is no longer in the view of active processes? Explain why or why not.

   Answer:
   Yes, it is possible. The installing of new views through GM at processes is not coordinated with the delivery of messages through urb. Thus, it may happen that p crashes, GM detects it and all processes install a view which does not include p anymore, and in parallel urb is still delivering messages from p. This is a motivating example for a different abstraction: View-Synchronous Broadcast.