Sample Questions

1. Provide an implementation of FIFO-order perfect point-to-point links on top of perfect point-to-point links using sequence numbers.
   Hint: FIFO-order abstraction guarantees that if message m is broadcasted before m' then it will be received before m'.
   Answer:
   FIFO-order abstraction:

Implements:
FIFOPerfectPointToPointLinks, **instance** *fpl*.

Uses:
PerfectPointToPointLinks, **instance** *pl*.

upon event $\langle$ *fpl*, *Init* $\rangle$ **do**
  forall $p \in \Pi$ **do**
    $lsn[p] := 0$;
    $next[p] := 1$;

upon event $\langle$ *fpl*, *Send* | q, m $\rangle$ **do**
  $lsn[q] := lsn[q] + 1$;
  trigger $\langle$ *pl*, *Send* | q, $(m, lsn[q])$ $\rangle$;

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

2. Explain the properties of the perfect failure detector and precisely write the difference between a perfect failure detector and an eventual perfect failure detector in terms of the interface and the properties.

   Answer:
   Perfect failure detector:

Module:

Name: PerfectFailureDetector, **instance** $\mathcal{P}$.

Events:

Indication: $\langle$ $\mathcal{P}$, *Crash* | p $\rangle$: Detects that process p has crashed.

Properties:

PFD1: *Strong completeness:* Eventually, every process that crashes is permanently detected by every correct process.

PFD2: *Strong accuracy:* If a process p is detected by any process, then p has crashed.

The main difference between an eventual PFD with a PFD in terms of the interface is that the eventual PFD has a Recover interface in which it can restore faith in a crashed process. In terms of the proper-

ties, eventual PFD provides only *eventual* accuracy. It guarantees that eventually no correct process is suspected.

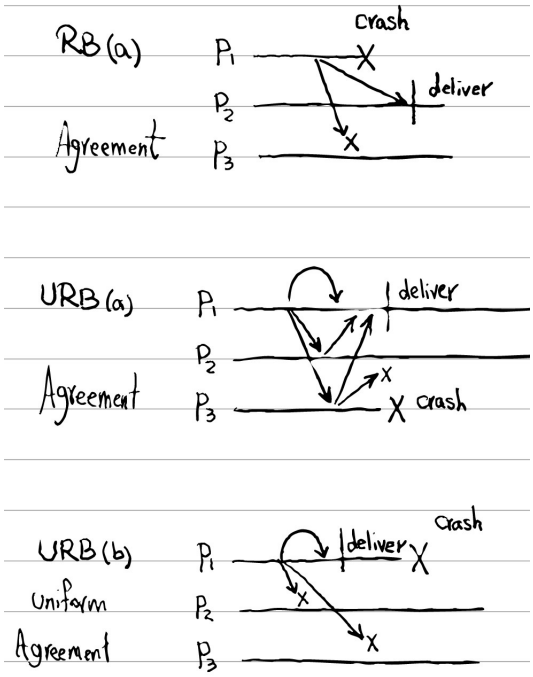
3. Modify the uniform reliable broadcast algorithm such that it does not use the failure detector but assumes that a majority of processes are correct.

   Answer:
   Instead of tracking correct processes and delivering a message when the ack is received from all of them, a process delivers a message when it receives an ack from a majority of processes. There will be at least one correct process in that majority and that process is enough to ensure delivery for all correct processes.

4. What happens in the reliable broadcast (RB) and uniform reliable broadcast (URB) algorithms if the (a) completeness, (b) accuracy property of the failure detector is violated? Show example execution diagrams if any of the four properties of broadcast is violated. Answer for RB.(a), RB.(b), URB.(a), and URB.(b) separately.

   Answer:
   RB(a) Agreement is violated
   p1 is crashed and has sent a message to p2 but not p3. Because of the incompleteness of the failure detector, p2 is never informed of the crash of p1 and does not rebroadcast the message. Therefore, p3 never receives it.
   RB(b) Correct.

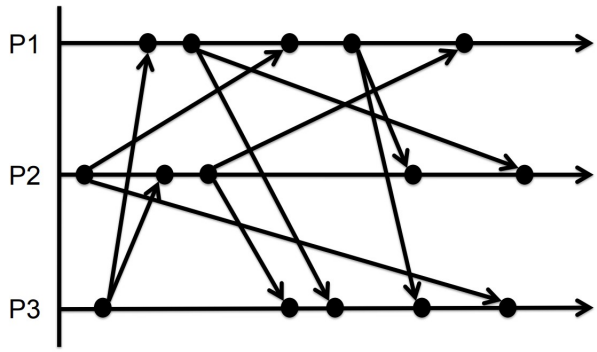
   URB(a) Validity and agreement are violated.
   The process may wait for a crashed process for ever. p3 crashes and can send a message to p1 but not p2. P1 delivers the message. Since the failure detector is not complete is does not inform p2 about the crash of p3. P2 waits for p3 for ever and does not deliver the message.
   URB(b) Uniform agreement is violated.
   p1 falsely suspects p2 and p3 to have crashed. Process p1 eventually delivers m and crashes afterward. It might be the case that p2 and p3 have never delivered and have no way of knowing about m.



5. Can we devise a causal broadcast algorithm that ensures only the nonuniform variant of causal order property: "No correct process p delivers a message m_2 unless p has already delivered every message m_1 such that $m_1 \rightarrow m_2$?"

   Answer:
   No. In short, because causality is a safety property and the algorithm does not know when processes crash, it has to preserve it for any process. Let's break it down more. Assume by contradiction that some algorithm does not ensure the causal delivery property but ensures its nonuniform variant. This means that the algorithm has some execution where some process p delivers some message m without delivering a message m' that causally precedes m. In this case, p might very well be correct which violates even the nonuniform variant.

6. Using the causal broadcast algorithm discussed in the class, write down the timestamps (vector clocks) at the point of each request and each receipt (assume vector clocks are all zero in the beginning). Also, mark receipts that are buffered, along with the points at which they are delivered to the application.



   Answer:
   P1(in order of the events):
   Receipt, (0,0,1)
   Request, (1,0,1)
   Receipt, (1,1,1)
   Request, (2,1,1)
   Receipt, (2,2,1)
   P2(in order of the events):
   Request, (0,1,0)
   Receipt, (0,1,1)
   Request, (0,2,1)
   Buffered as (2,1,1), Timestamp still (0,2,1)
   Receipt (1,2,1)
   Receipt (2,2,1) (previously buffered)
   P3(in order of the events):
   Request, (0,0,1)
   Buffered as (0,2,1), Timestamp still (0,0,1)
   Receipt (1,0,1)
   Buffered as (2,1,1), Timestamp still (1,0,1)
   Receipt (1,1,1)
   Receipt (2,1,1) (previously buffered)
   Receipt (2,2,1) (previously buffered)

7. Assume that five processes sending requests to one another using causal broadcast abstraction covered in the class. Below are the current vector clocks for each process:
   Process A (5, 4, 3, 4, 1)
   Process B (1, 5, 4, 3, 2)
   Process C (2, 1, 5, 4, 3)
   Process D (3, 2, 1, 4, 4)
   Process E (4, 3, 2, 2, 5)
   If Process D sends a message, which process(es) can accept it immediately? Choose all processes that can accept it *immediately*.

   Answer:
   None.
   Be aware of the difference between causal order and FIFO order. The question is asking for the causal broadcast. Since none of the processes have a vector clock greater or equal to the vector clock at D at the time of sending, none of the processes can deliver it immediately.