ECE428 Homework 2

Due: 11:59 p.m. on Monday March 20th, 2023

This assignment has 4 questions with 60 points in total. The solutions must be typed, and submitted via Gitlab. However, the diagrams can be hand-drawn. You must acknowledge any sources used to arrive at your solutions, other than the course materials and textbook. All homework assignments are expected to be an individual work, so no collaborations are allowed.

Question 1: The only things guaranteed in life are FIFO channels *[20 points]*

Consider a system with pair-wise FIFO communication channels. Explain, which of the ordering properties – FIFO, causal, and/or total ordering – will be satisfied in the following four scenarios. For each ordering property, either explain (in one or two sentences) why it will be satisfied, or provide a counter-example, for example, using a diagram. The same counter-example can be used for different scenarios. For B-multicast over FIFO channels, explain whether and why it automatically satisfies causal ordering, or provide a counterexample.

(a) B-multicast in a situation where there are no process failures;

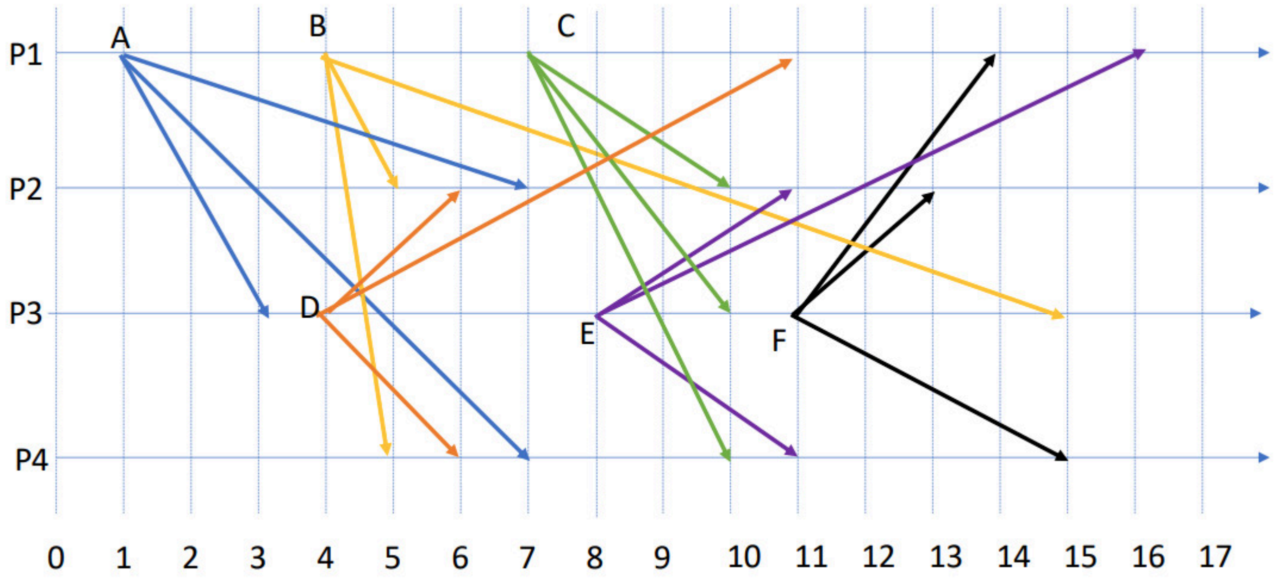
(b) R-multicast in a situation where there are no process failures;

(c) R-multicast in a situation where process failures may occur;

(d) The sequence number-based FIFO multicast algorithm discussed in class.

Questions 2-4 are on the next pages.

Question 2: You're out of order! This whole trial is out of order! *[18 points]*



(6) (a) Consider the event diagram above. To assure the FIFO multicast delivery order, which messages will have to be delayed in a holdback buffer? For these messages, what is the earliest point at which they can be delivered? For simplicity, assume that messages multicast are self-delivered at the sending process instantaneously.

(6) (b) Consider the same diagram, but now suppose we wanted to assure a causal multicast delivery order. Which messages would have to be delayed?

(6) (c) Still considering the same diagram, assume ISIS total ordering has been used. For every message and process, write down the process's proposed priority for the message, and what the final priority for the message will be. Assume that no other messages have been seen, and the proposed priorities all start at 1. It can be also assumed that reply messages with their proposed priorities get delivered after time 17.

Question 3: Trickle-up Multicast *[7 points]*

Consider an R-multicast algorithm running in a multicast group of 100 nodes.

(2) (a) How many copies of a message will be sent at each multicast?

(2) (b) The R-multicast was modified, so that upon receiving a multicast request, the re-multicast is sent only to the higher-numbered processes, as indicated in this Python implementation on the next page:

```
def r_multicast(message, group):
  for member in group:
     unicast(member, message)

def receive_message(message, group, sender):
   if message not in received_set:
      received_set.add(message)
      if sender.id != myid:
        for member in group:
           # new
           if member.id > myid:
             unicast(member, message)
      deliver(message)
```

How many messages will be sent using this modification at each multicast?

(3) (c) Change the Python code above to guarantee a reliable delivery. Assume that once the call to unicast() returns, the message will always be delivered to the recipient, even if the sender may later crash. The proposed code modification should send the same number of messages as the original code given in part (b).

Question 4: There's Only Room for One Sheriff in This Critical Section *[15 points]*

Consider the following table of processes, which lists at what time (since the system starts) the processes request to enter a critical section by calling enter, and how long each process spends in a critical section after it is admitted:

Process	Time critical section is requested	Time spent in critical section
P_3	10 ms	20 ms
P_2	15 ms	10 ms
P_1	20 ms	15 ms
P_4	25 ms	30 ms
P_5	40 ms	25 ms

(5) (a) Suppose that mutual exclusion is managed by a central server algorithm, with P_1 being the leader. In this system, assume a one-way delay of 8 ms between any pair of processes. Note that P_1's messages to itself for requesting/granting/releasing critical section access take a negligible time (0 ms). List what time each process enters the critical section. You may want to include a diagram for a partial credit.

(5) (b) Suppose now that the processes are in a token ring, with the following structure:

$$P_1 \rightarrow P_4 \rightarrow P_2 \rightarrow P_3 \rightarrow P_5 \rightarrow P_1$$

Assume that at time 0 ms, the token is at P_1, and one-way delay between any processes is 8 ms. When would each process enter its critical section?

(5) (c) Assume now that the processes are using Ricart-Agrawala mutual exclusion. Assuming again a one-way delay of 8 ms, when will each process enter the critical section? All processes' local Lamport timestamps are set to 0 at time 0 ms, and no messages other than those used in the Ricart-Agrawala exclusion are sent between any processes.