CS425/ECE428 Homework 2

Due: 11:59 p.m. on Friday, October 1, 2021

This assignment has 3 pages and 4 questions, worth a total of 60 points. Solutions must be submitted via Gradescope. Solutions must be typed, not hand-written, but you may include hand-drawn diagrams.

You must acknowledge any sources you used to arrive at your solutions, other than the course materials and textbook. If you work in a group on homework assignments, please list the names of your collaborators, but make sure to write your own solution.

1. The only things guaranteed in life are FIFO channels . . . . *20 points*

   Consider a system that has FIFO pair-wise communication channels. Explain which of the ordering properties—FIFO, causal, and/or total ordering—will be satisfied in the following scenarios. For each ordering property, either explain *briefly*—1–2 sentences—why it will be satisfied or provide a counterexample (you will probably want to use a diagram). You may use the same counterexample for multiple answers.

   For example, for B-multicast over FIFO channels, explain whether it will automatically satisfy causal ordering (and why) or provide a counterexample.

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

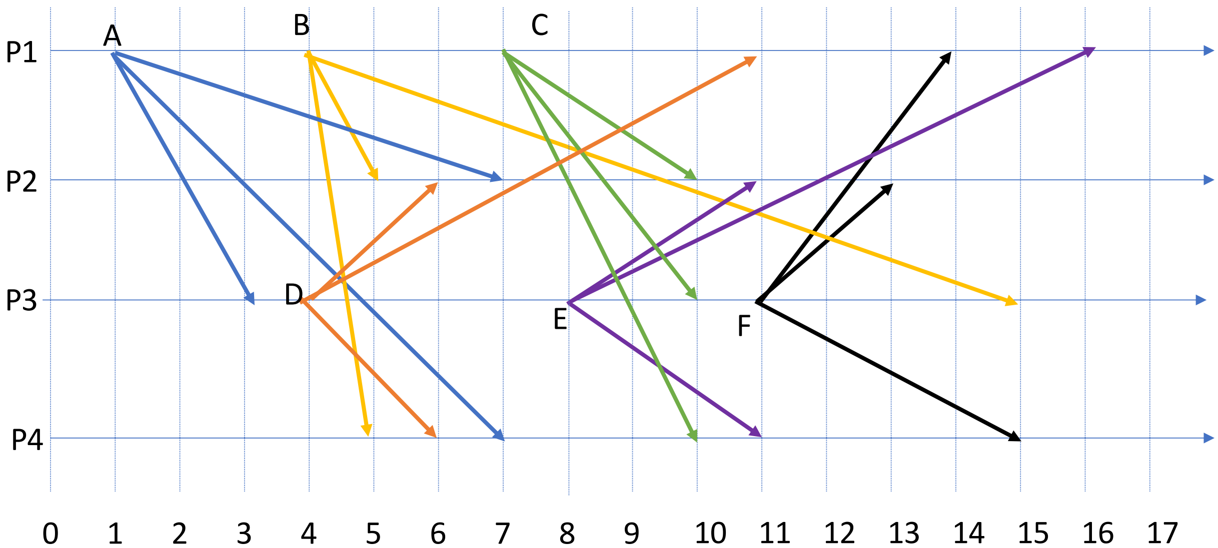
   (b) R-multicast in a situation where there are no process failures

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

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

2. You're out of order! This whole trial is out of order! . . . . *18 points*

(6) (a) Consider the following diagram. To assure a 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. Again, which messages would have to be delayed?

(6) (c) Still using the same diagram, consider an implementation of ISIS total ordering. For each message, write down what each process's proposed priority is for the message, and what the final priority for the message will be. You should assume that no other messages have been seen, so the proposed priorities start at 1. You may also assume that the reply messages with the proposed priorities all get delivered after time 17.

3. Trickle-up Multicast . . . . *7 points*

Consider the R-multicast algorithm running in a multicast group with 100 nodes.

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

(2) (b) Suppose we modified R-multicast so that, upon receiving a multicast, the re-multicast is sent only to higher numbered processes:

```
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) How would you need to change this code to guarantee reliable delivery? You can assume that once the call to `unicast()` returns the message will always be delivered to the recipient even if the sender later crashes. Your modification should send the same number of messages as in part (b) above.

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 system start) each process requests 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 negligible time (0 ms). List what time each process enters the critical section. You may want to include a diagram for partial credit purposes.

(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) Now suppose that the processes are using Ricart-Agrawala mutual exclusion. Again assuming an 8 ms one-way delay, when would each process enter the critical section? You can assume that all processes' local Lamport timestamps are set to 0 at time 0 ms and that no messages other than those used in Ricart-Agrawala are sent between the processes.