HW 2

Name: Jiacheng Zhao

Resources. (All people, books, articles, web pages, etc. that have been consulted when producing your answers to this homework)

Textbooks and instructor's slides

https://courses.engr.illinois.edu/cs425/fa2011/hw/hw2-sol.pdf

On my honor, as an Aggie, I have neither given nor received any unauthorized aid on any portion of the academic work included in this assignment. Furthermore, I have disclosed all resources (people, books, web sites, etc.) that have been used to prepare this homework. This work is my own and is written in my own words.

Signature: JIACHENG ZHAO

Problem 1. Exercise 15.4

Solution.

Suppose processes Pa and Pb both sent a message to Master for grant in entering the CS and Pa sent earlier than Pb did. However, due to the message propagation delay Pb's message arrived earlier than Pa's. In that case Master would sent the token to Pb and Pb entered the CS before Pa, although Pa sent the request first. That's a situation in which two requests are not processed in happened-before order.

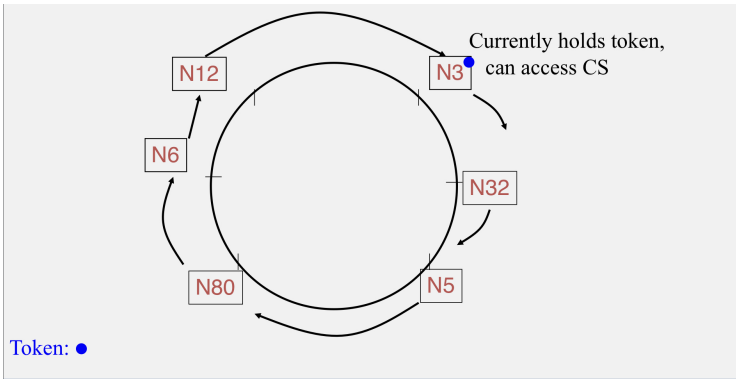
Problem 2. Exercise 15.5

Solution.

We can set a timer each time Master sends the token to a client. The length of the timer is long enough for any client to finish the CS mission. If the timer fires and a client is still holding the token, then we can assume that the this client has already crushed. In that case we should let Master invalidate the previous token and generate a new token, and send the new one to the clients according to queue, from which the crushed client should be removed. Since the crushed client can not send a request for token grant, it won't disturb the Master's queue any more. After some time if the previous crushed client send back the old token, Master should still regard the token as invalid and discard it. However, Master should send a massage to this client telling that the token has been renewed and change the length of its own timer. If this client sends a request for token grant in the future, Master could add it to the queue, but only with the new token.

Problem 3. Exercise 15.6

Solution.



Take the scenario above as an example. Currently N3 holds the token and is accessing the CS. Now suppose this time N12 needs to enter the CS, but it does not have the token right now. So N12 has to wait until N6 passes the token to it.

After some time N3 exits the CS and passes the token to N32. However, at the same time N5 needs to enter the CS as well. Since N32 does not need to enter the CS, it passes the token to N5 and N5 enters the CS successfully. In the case described above, although N12 needs to enter the CS before N5, N12 enters CS later than N5 because of its position. This scenario shows that processes are not necessarily granted entry to the critical section in happened-before order.

Problem 4. Exercise 15.7

Solution.

From the problem we can know that in a system each process uses the CS many before another process requires it. However, according to the Ricart-Agrawala algorithm, a process will immediately reply to other processes' request message if it finishes the CS mission. In that case, suppose process P1 is using the CS. After finishing the mission, it replies to others' messages. However, after reply it will send a request to everyone telling that it needs to enter the CS, since normally a process should use the CS more than once. However, P1 has to wait until all the other finish their mission and then it can enter the CS. These will sufficiently increase the client delay, which is really inefficient.

To improve this situation, we can add a timer when each process finishes its CS mission. After exiting, We can let the process not reply to others immediately. Instead, let the process wait for a little while to see if it needs to reuse the CS. If so, start the CS again and also set a timer after exiting. If the process does not do anything until the timer fires, then the process is allowed to reply to other's requests. This mechanism allows a process to reuse the CS instead of replying to others immediately after exiting and improve the efficiency.

Since any process will eventually reply to others' request(a process will eventually finish using the CS), my adaptation satisfies liveness condition ME2.

Problem 5. Exercise 15.9

Solution.

In case of the partitions, the network of progresses will form subgroups. When subgroups are formed, they will run bully algorithm to elect a leader specially for this subgroup. However, when the partitions are over and network merges, these progresses should run the bully algorithm again to elect a new leader for the entire progresses.