27760 Utku Alkan PA2 Report

Firstly, I checked if the input is valid and then initializations are done (semaphores, barrier, etc.). Then I created threads according to the input. So, the thread function will have string "A" if the team is TeamA and the function will have "B" if the team is TeamB as arguments.

It is important to note that I created my own semaphore implementation with the help of the lecture slides. So, the value of the semaphores can be decremented below 0 means that the value will represent the waiting thread's count.

Now I will explain my thread creation function. It is also important to note that I created two semaphores for team A and team B; both of their value is initialized to 0.

Firstly, it locks with locker mutex so no other thread can enter. Then it has to continue until it enters the block when its team is checked. Then the semaphores' pseudocode is like the following

if(fan's team is A):

check if 1 A semaphore and 2 B semaphore are waiting

if true => post all of that 3 semaphores since with the new fan, 1 car is completed (2 A, 2 B)

if false => check if 3 A semaphore is waiting

if true => post all of the 3 A semaphores since with the new fan, 1 car is completed (4 A)

if false => fan should wait in teamAsemaphores since it cannot form a complete car with the waiters

if(fan's team is B):

check if 1 B semaphore and 2 A semaphore are waiting

if true => post all of that 3 semaphores since with the new fan, 1 car is completed (2 A, 2 B)

if false => check if 3 B semaphore is waiting

if true => post all of the 3 B semaphores since with the new fan 1 car is completed (4 B)

if false => fan should wait in teamBsemaphores since it cannot form a complete car with the waiters

There are two important details that are not visible in pseudocode. The first one is, the driver variable becomes true if a car is completed. This will be beneficial later. And the second one is, in my implementation of semaphores, sem_wait does not only waits; it also unlocks the locker mutex which was locked at the top of the function. This is very important and useful since now any other fan can enter the function and continue doing the same waiting process until one fan wakes the waiting fans.

After all of these are finished, I implemented a barrier with 4 threads. So, it waits for all four threads which will constitute the car. After all of them arrive, with the help of the first note we mentioned just now, one of them enters to the "driver==true" block and defines itself as the driver. Others will skip this "if block" since the driver

becomes false instantly. Also, it will unlock the locker which was locked by the last thread and has never waited. Since it never waited, the locker mutex was never unlocked so we had to unlock it.

Then, all of the threads are joined by a for loop so they will finish before the main thread.