Seminar Report: Muty

Xavier Caballero & Albert Capdevila

March 20, 2022

1 Open questions

1.1 Lock1: Deadlock

a) What unwanted halting situation can occur when contention increases?
Quan tenim un nivell de contenció molt elevat es pot produir un **deadlock**. Fent diverses proves, hem vist que quan el temps de sleep i de work és igual a 50 ms es produeixen una gran quantitat de deadlocks. El motiu és que quan hi ha una contenció tant alta, s'incrementa la possibilitat d'enviar una request al mateix moment i que totes les instàncies es quedin en el estat de wait esperant una resposta. Si ens trobem en aquesta situació, després d'esperar el temps de withdrawal (8000 ms), el worker enviarà un missatge de release.

b) Indicate two situations which can lead to a process withdrawal (i.e., a process gets tired of waiting to be granted access to the critical section). Assume that processes do not fail.
Les següents situacions poden provocar un withdrawal:

- Quan es produeix un deadlock. (Com s'ha comentat a la pregunta anterior)
- Quan el temps de work és major al de withdrawal. En aquest cas els processos que estiguin en estat de waiting rebran un missatge de release per part del worker, ja que el temps de work és major al de withdrawal.

1.2 Lock2: Resolving deadlock

c) Justify how your code guarantees that only one worker is in the critical section at any time.
Per tal de que un treballador pugui entrar a la regió crítica necessita que tots els locks acceptin la seva request. En aquesta solució, podríem tenir la següent situació (suposem un escenari amb 3 locks):

1. Tenim els 3 locks en estat open.
2. El lock amb prioritat 2 demana entrar a la regió crítica i passa al estat de waiting.
3. El lock amb prioritat 1 envia el OK.

4. Passat un temps, el lock amb prioritat 1 demana entrar a la regió crítica.
5. El lock amb prioritat 2 rebrà la request del 1 i com el identificador és menor enviarà el OK.
6. El lock amb prioritat 3 envia el OK al 1 i al 2.
7. Tant el lock 1 com el 2 han rebut el OK dels altres locks i entren a la regió crítica.

Per evitar que hi hagi més d'un procés a la regió crítica, enviarem el OK i una request amb una referència nova per tal d'assegurar que només entra un procés a la regió crítica. En el cas de l'exemple anterior, el lock amb prioritat 2 enviarà el OK i una request amb una referència nova per tal de garantir que només hi ha un treballador a la regió afectada.

d) Do the situations described in the previous question b) still lead to a process withdrawal in this lock implementation?
Com que no tenim deadlocks en aquesta solució, la primera situació ja no pot provocar cap withdrawal. En el cas de la segona, com que no es té cap control del temps de work podem continuar tenint withdrawals.

e) What is the main drawback of this lock implementation?
El principal inconvenient d'aquesta solució és que el repartiment per entrar a la regió critica és molt poc equitatiu. És a dir, les instàncies amb el identificador més petit (més prioritàries) entraren moltes més vegades a la regió crítica que les que tenen un identificador més gran. Com es pot veure en la següent imatge, la diferència és molt gran entre el John (identificador 1) i el George (identificador 4). El primer ha entrat 121 vegades mentre que el segon només 11.

```
John: 121 locks taken, 14.652892561983471 ms (avg) for taking, 0 withdrawals
Paul: 52 locks taken, 114.0576923076923 ms (avg) for taking, 0 withdrawals
Ringo: 101 locks taken, 28.73267326732673 ms (avg) for taking, 0 withdrawals
George: 11 locks taken, 788.2727272727273 ms (avg) for taking, 0 withdrawals
```

Figure 1: Inconvenient principal del Lock2

1.3 Lock3: Lamport's time

f) Justify if this lock implementation requires sending an additional request message (as well as the ok message) when a process in the waiting state receives a request message with the same logical time from another process with higher priority.
En aquest cas no cal enviar una nova request com a la solució anterior. Si pensem en el escenari comentat a la pregunta c, ara el procés més prioritari quan rebi una request actualitzarà el rellotge amb el màxim entre el timestamp que rep i el seu clock. Després, quan vulgui enviar una request primer incrementarà una unitat el seu rellotge i finalment enviarà el missatge. Per tant, en aquest cas ens assegurem que el procés menys prioritari no enviarà el OK perquè el timestamp del procés prioritari serà major.

g) Do the situations described in the previous question b) still lead to a process withdrawal in this lock implementation?
En aquest cas passa el mateix que a la solució anterior. El primer cas no pot passar i el segon si perquè no es té control del temps de work.

h) Note that the workers are not involved in the Lamport's clock. According to this, would it be possible that a worker is given access to a critical section prior to another worker that issued a request to its lock instance logically before (assuming happened-before order)? (Note that workers may send messages to one another independently of the mutual-exclusion protocol).
Aquesta situació no podria passar perquè l'accés a la regió crítica és en funció de quan es realitza la petició i en cada request s'ha d'enviar el timestamp per tal de controlar tot això. Per tant, és impossible que tinguem la situació de l'enunciat en el lock3.

2 Personal opinion

Aquest seminari és molt útil per tal d'entendre diferents algoritmes d'exclusió mútua i quins problemes poden presentar. A més a més, al tenir una interfície gràfica amb el estat dels diferents treballadors podem veure més fàcilment si la nostre implementació funciona bé. Per aquestes raons, pensem que aquest seminari es pot incloure el pròxim curs.