Tarea 7

Clase: Computación distribuida
Profesora: Karla Vargas

Cruz Perez Ramón -315008148
Marco Antonio Orduña Avila -315019928

2 de diciembre de 2020

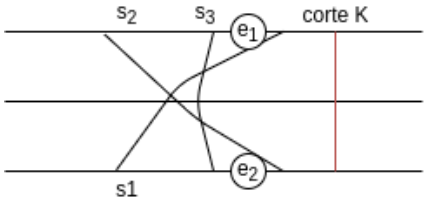
Responde las siguientes preguntas:

1. (2.5ptos) Demuestra que solo hay un corte consistente maximal que precede a cualquier corte.
   Por contradiccion:
   Sup. que existen dos cortes maximales S_1 y S_2.

   - Caso 1: Los cortes no se cruzan, en cuyo caso S_2 es más reciente:
     Sup. que S_1 esta muy cerca del corte k, entonces por ser maximal es el mas cercano, pero como S_2 tambien es maximal, entonces uno de los dos debe ser el mas cercano.
     Si, S_2 es el más cercano que S_1, entonces S_2 el maximal. ▼
     Si, es el caso que estan al misma distancia, entonces $S_1 = S_2$ ▼
     $\Longrightarrow$ Existe un unico maximal.

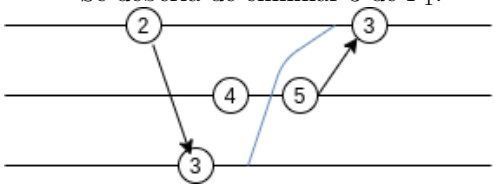
   - Caso 2: Si se cruzan, hay un envio de un mensaje despues de S_1:
     Como sabemos que cruzan, significa hay un evento$_1$ $\in S_1$ y evento$_1$ $\notin S_2$, y uno evento$_2$ $\in S_2$ y evento$_1$ $\notin S_1$
     Sup. que existe un S_3 que cubre esos dos eventos y es maximal, como cubre los dos eventos que y ademas es maximal significa que el S_3 es el unico maximal. ▼

     $\Longrightarrow$ Existe un unico maximal.

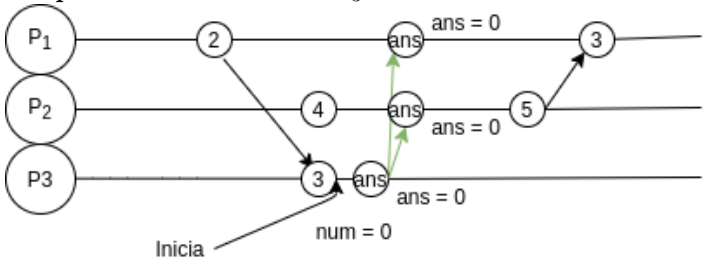


2. (2.5ptos)

   - ¿Cuáles de los siguientes cortes son consistentes? De los cortes que no fueron consistenes di cual sería su corte consistente maximal.
     -C_1: Es consistente, pues a todo los eventos ocurren en el pasado.
     -C_2: Es consistente, pues todos los envios ocurren el pasado.
     -C_3: **No** es consistente, pues ocurren una recepcion del mensaje, cuando el mansaje fue enviado desde el futuro.
     –C_3: Correccion.
     Se deberia de eliminar 3 de P_1.



- Ejecuta el algoritmo de snapshot en la ejecución anterior cuando el proceso p_3 recibe la instrucción de tomar un snapshot después de su evento 3. Los canales son FIFO.
  Despues de evento 3 en P_3



3. (2.5ptos) Diseña un algoritmo para hacer broadcast de cierto mensaje M cada T unidades de tiempo. IMPORTANTE: No es necesario distinguir entre cada ola de mensajes.

   Como es un sistema asincrono, se le asigna un reloj local a cada proceso, para que cada cierta unida de tiempo se envie el mensaje.
   Cuando el algoritmo A, inicia broadcast en paralelo:

Algorithm 1 New broadcast

```
1: Inicia
2: M = mensaje a enviar
3: T = tiempo a esperar
4: reloj = inicia()          tiempo en que esta el proceso
5: while True do
6:     wait until T == reloj
7:     send M a vecinos
8:     reloj = reset(0)      reinicio el reloj
9: end while
```

   Se reinicia el reloj para volver a tomar el tiempo.

4. (2.5ptos) ¿Para qué se usan los timeouts? Da ejemplos en la vida real donde sean usados

   Creo que un buen ejemplo es cuando se va a hacer una compra de algo en linea.
   Nos dan cierto tiempo para completar la compra, pues si nos tardamos mucho el sistema puede pensar que nos hemos ido, o simplemente ya no queremos la realizar la compra.
   En este caso nos tomaria como un proceso fallido.