Tarea 4

Ian Israel García Vázquez Armando Ramírez González
Ricardo Montiel Manriquez Christopher Alejandro Escamilla Soto
Jonás García Chavelas

11 de noviembre de 2021

1. Ejercicios

1. *a*) Demuestra que el algoritmo 1 soluciona el problema del consenso, tolerando $t < n$ fallas de tipo paro, done n es el número de procesos en el sistema.

   Para demostrar que el algoritmo soluciona el problema de consenso debemos demostrar que cumpla con lo siguiente:

   1) **Terminación**

      Podemos observar que el algoritmo termina en $t+1$ rondas, ya que para una ejecución donde haya t fallas de tipo paro tal que $t < n$, en la ronda 1 serán enviados $0 < n-t$ mensajes por los procesos correctos (que no fallan). Depués de ello, los procesos correctos se comunicarán $r = t$ para llegar a consenso ejecutando la instrucción $rec[t-1] == rec[t]$. Por lo que el algoritmo terminará en la ronda $r = t+1$.

   2) **Válidez**

      El valor es propuesto por un procesos p_k, ya que después de $t+1$ rondas $flag$ toma el valor de $true$, por lo que $prop$ está dada por $max(vista)$.

   3) **Acuerdo**

      Sugpongamos que después de $t+1$ rondas el algoritmo no termina, sino que ejecuta otra ronda.

      Como el algoritmo se ejecutó $t+1$ rondas, sabemos por la condición de válidez que los procesos eligieron una propuesta que es el $max(vista)$, entonces al ejecutar una ronda más el valor elegido por los procesos en la ronda $t+1$ se mantiene.

      Por lo tanto existe un acuerdo.

      Por lo tanto el algoritmo 1 soluciona el problema del consenso.

   b) ¿Es cierto que los procesos correctos terminan en a lo más $max(f+2, t+1)$ rondas en el algoritmo 1? Argumenta tu respuesta. Recuerda que $f \leq t$ es el n número de fallas que realmente ocurren en una ejecución dada.

   Si $f = t$, como f es el número de fallas reales, entonces a la ronda $f+2$, el proceso dejará de fallar y terminará correcto, en caso de que $f < t$ entonces t, en el paso $t+1$ si $f+2 < t$ podemos asegurar que los procesos han terminado correctamente, pues nuevamente f son las fallas reales.

Esto significa que no podemos tener más fallos del máximo marcado por $max(f+2, t+1)$, pues este solo tolera $t < n$ fallas de tipo paro.

c) Haz un análisis del número máximo de mensajes que se envían en una ejecución del algoritmo 1. Tu cota debe estar en función de n y f.

Sean una gráfica completa K_n con n el número de procesos y sea t el número máximo de fallas de tipo paro que puede haber, tal que $t < n$.

En la ronda inicial $r = 1$ cada proceso correcto envía un mensaje, es decir, son enviados $n - t$ mensajes en la ronda inicial. En las siguientes t rondas, nuevamente, cada proceso correcto enviará un mensaje por lo que en cada ronda r serán enviados $n - t$ mensajes, entonces en las primeras t rondas se habrán enviado $t * (n - t)$ mensajes.

En la ronda $t + 1$ ningún mensaje será enviado ya que es la ronda en la que se llega a consenso.

Por lo tanto en una ejecución del algoritmo 1 el número máximo de mensajes que serán enviados es de $t * (n - t)$, para n procesos y a lo más t fallas.

2. Demuestra que el algoritmo 2 soluciona el problema del consenso en una gráfica **G** arbitraria, tolerando $t < k(G)$ fallas del tipo paro. $k(G)$ denota la conexidad por vértices de G, es decir, el mínimo número de vértices que se tienen que quitar de G para desconectarla. Entonces, si hay menos $k(G)$ fallas de los procesos, la gráfica que queda sigue siendo conexa.
Tip: Piensa que tanto tarda en fluir la entrada mínima más pequeña, a pesar de las fallas que puedan ocurrir.

Retomando

El problema del consenso

Cada proceso tiene una entrada, un valor que propone para el consenso. Buscamos un algoritmo que proponga lo siguiente:

a) **Terminación**: Todo proceso que es correcto, elige una propuesta.

b) **Validez**: Todo valor elegido fue propuesto por un proceso

c) **Acuerdo**: Todo par de valores elegidos son idénticos

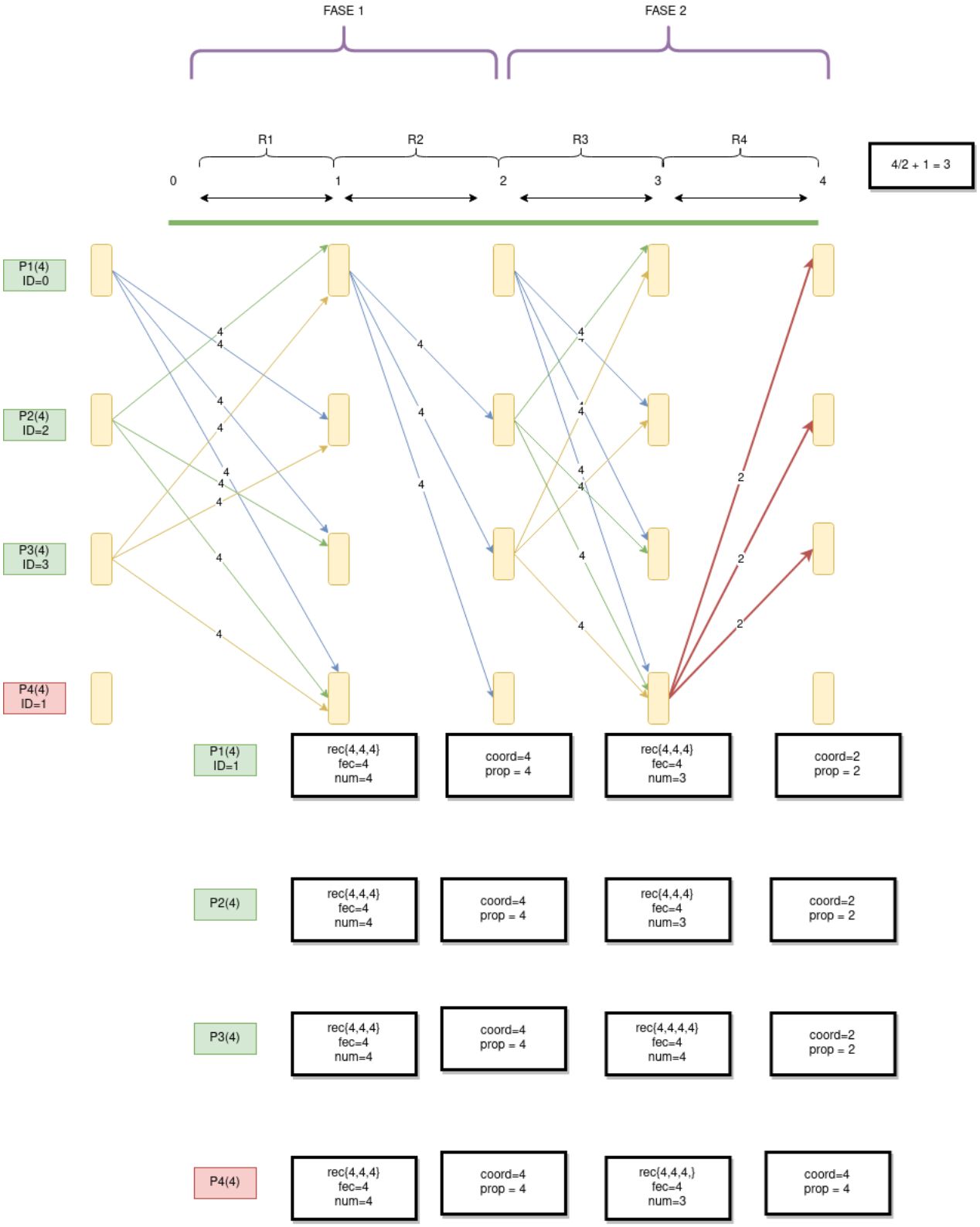
Resolviendo **Afirmación**: El algoritmo 2 soluciona el problema del consenso

a) **Terminación**: Cada ronda r se elige una propuesta del conjunto de los props, esto durante n-veces.

b) **Válidez**: En MIN se encuentra el conjunto de prop recibidos de cualquiera de los vecinos y de sus respectivos vecinos.

c) **Acuerdo**: El total de iteraciones corresponde a n; pero como $t < k(G)$, supongamos el caso de un árbol con $|V| = n$ y pensando que han fallado todos los procesos que tenían que fallar, $|E| = n-1$, suponiendo el peor de los casos, el valor mínimo se encuentra en un nodo con un único vecino, tendrá que recorrer vértice por vértice, lo que implica que tomará a lo más $n - 1$ iteraciones, y como son n iteraciones, se cumplirá el acuerdo y todos los nodos cumplirán con el **Acuerdo**

3. Da una ejecución del algoritmo 3 para $n = 4$ procesos y $t = 1$ fallas bizantinas, en la que los procesos correctos no lleguen a un consenso, a pesar de que los cuatro procesos, íncluido el Bizantino, empiecen con la misma propuesta y el bizantino sea el último de los coordinadores de la ejecución. Explica tu respuesta.


FASE 1
FASE 2
R1
R2
R3
R4
0
1
2
3
4
4/2 + 1 = 3
P1(4)
ID=0
P2(4)
ID=2
P3(4)
ID=3
P4(4)
ID=1
P1(4)
ID=1
rec{4,4,4}
fec=4
num=4
coord=4
prop = 4
rec{4,4,4}
fec=4
num=3
coord=2
prop = 2
P2(4)
rec{4,4,4}
fec=4
num=4
coord=4
prop = 4
rec{4,4,4}
fec=4
num=3
coord=2
prop = 2
P3(4)
rec{4,4,4}
fec=4
num=4
coord=4
prop = 4
rec{4,4,4,4}
fec=4
num=4
coord=2
prop = 2
P4(4)
rec{4,4,4}
fec=4
num=4
coord=4
prop = 4
rec{4,4,4,}
fec=4
num=3
coord=4
prop = 4

Aquí se muestra la ejecución del algoritmo 3 para 4 procesos y uno de ellos siendo bizantino, el diagrama esta dividido en las 2 rondas de cada fase, como cosas importantes a notar:

Usamos dos propiedades de de los proceso bizantinos.

La primera propiedad: Dejar de enviar mensajes cuando debería de enviar, y esto lo podemos notar en la ronda 1, el proceso bizantino no envió su mensaje inicial y repitió este mismo comportamiento durante la ronda 3 donde nuevamente tendría que enviar un mensaje al inicio de la ronda.

La segunda propiedad: Mentir, enviar información contradictoria a procesos distintos y esto lo podemos notar en la ultima ronda de la fase 2 pues miente sobre la información que devisa enviar, enviando un 2 en vez de 4.

Es importante notar que para que el proceso bizantino sea el ultimo de los coordinadores en la ejecución y como t=1 tendríamos que tener al proceso bizantino con ID = 1 de esta forma el proceso bizantino queda como el ultimo en la ejecución.

En la ultima ronda cuando el coordinador es el proceso bizantino, este manda un mensaje con información incorrecta y como la condicional de la ronda 4 no se cumple obliga a que la propuesta de los procesos no bizantinos sea modificada por el mensaje recién recibido por el proceso bizantino.

Al final de la ejecución podemos ver que el proceso bizantino logra entorpecer la decisión final de todos lo procesos esto gracias a sus dos propiedades y a las circunstancias que logra modificar.