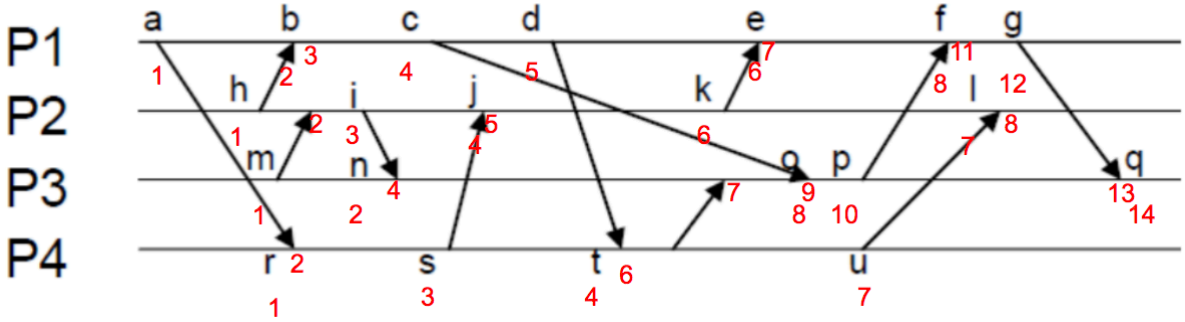
Gabriela Lemos Lúcidi Pinhão

Sistemas Distribuídos – Lista 4

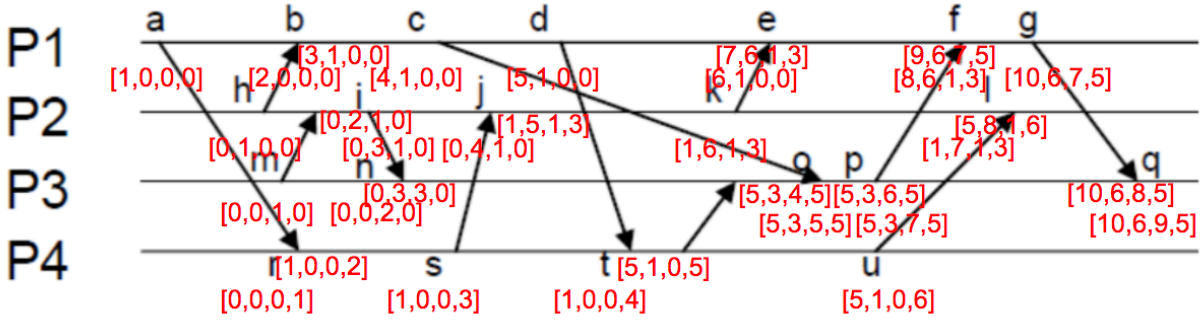
1)

1. A → S
2. T || K
3. B || K; B || N; B → U; K || N; K || U; N || U.
4. No seguinte desenho a linha do tempo de cada processo está abaixo da linha dele. Os valores deslocados para cima representam o valor do evento de chagada de mensagem. Os deslocados para baixo representam o evento pertencente ao processo.



A=1; B=2; C=4; D=5; E=6; F=8; G=12; H=1; I=3; J=4; K=6; L=7; M=1; N=2; O=8; P=10; Q=14; R=1; S=3; T=4; U=7;

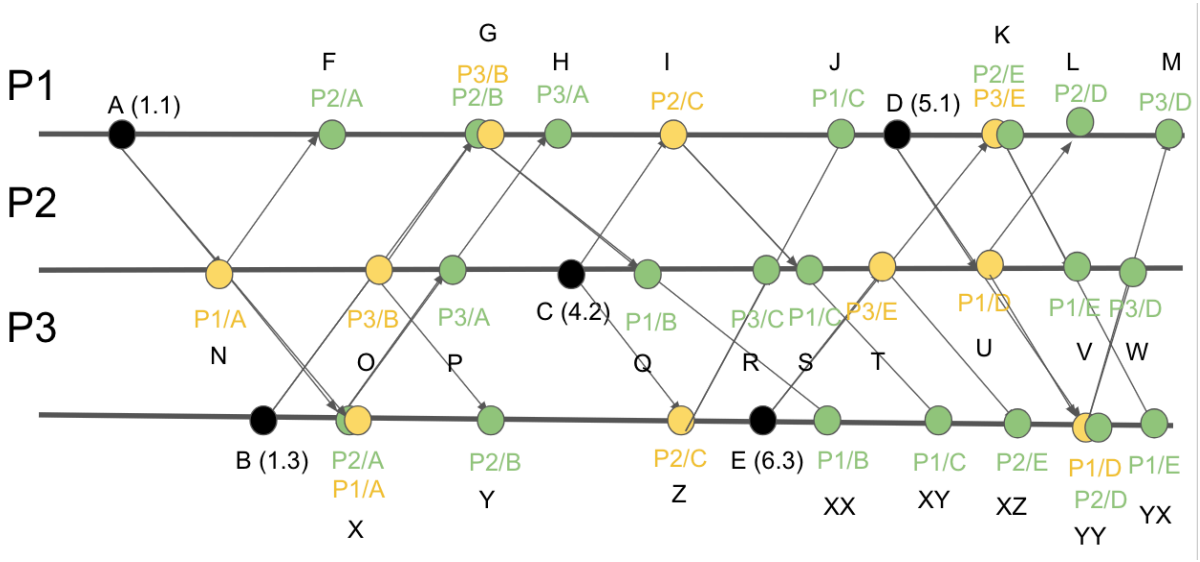
5. No seguinte desenho a linha do tempo de cada processo está abaixo da linha dele. Os valores deslocados para cima representam o valor do evento de chagada de mensagem. Os deslocados para baixo representam o evento pertencente ao processo.



A=[1,0,0,0]; B=[2,0,0,0]; C=[4,1,0,0]; D=[5,1,0,0]; E=[6,1,0,0]; F=[8,6,1,3]; G=[10,6,7,5]; H=[0,1,0,0]; I=[0,3,1,0]; J=[0,4,1,0]; K=[1,6,1,3]; L=[1,7,1,3]; M=[0,0,1,0]; N=[0,0,2,0]; O=[5,3,5,5]; P=[5,3,7,5]; Q=[10,6,9,5]; R=[0,0,0,1]; S=[1,0,0,3]; T=[1,0,0,4]; U=[5,1,0,6];

6. Não existe.
7. Nada.
8. X → Y

2)



P1

Após A: A (1.1);
Após F: A (1.1);
Após G: A (1.1), B (1.3) / Confirmações de B;
Após H: A (1.1), B (1.3) / Confirmações de A→Executa A→Executa B
Após I: C (4.2);
Após J: C (4.2) / Confirmações de C → Executa C;
Após D: D (5.1);
Após K: D (5.1), E (6.3) / Confirmações de E;
Após L: D (5.1), E (6.3)
Após M: D (5.1), E (6.3) / Confirmações de D→Executa D→Executa E;

P2

Após N: A (1.1);
Após O: A (1.1), B (1.3);
Após P: A (1.1), B (1.3) / Confirmações de A→Executa A;
Após C: B (1.3), C (4.2);
Após Q: B (1.3), C (4.2) / Confirmações de B→Executa B;
Após R: C (4.2)
Após S: C (4.2) / Confirmações de C→Executa C;
Após T: E (6.3);
Após U: D (5.1), E (6.3);
Após V: D (5.1), E (6.3) / Confirmações de E;
Após W: D (5.1), E (6.3) / Confirmações de D→Executa D→Executa E;

P3

Após B: B (1.2);
Após X: A (1.1), B (1.2) / Confirmações de A→Executa A;
Após Y: B (1.2);
Após Z: B (1.2), C (4.2);
Após E: B (1.2), C (4.2), E (6.3);
Após XX: B (1.2), C (4.2), E (6.3) / Confirmações de B→Executa B;
Após XY: C (4.2), E (6.3) / Confirmações de C→Executa C;

Após XZ: E (6.3);
Após YY: D (5.1), E (6.3) / Confirmações de D→Executa D;
Após YX: E (6.3) / Confirmações de E→Executa E;

3) Uma vantagem é o bom desempenho pois apenas 3 mensagens são trocadas por acesso a região crítica. A desvantagem é que é um ponto único de falha, pois se o coordenador falhar, nenhum processo pode ser executado.

4) A vantagem é que se um de seus vizinhos sair do anel o nó já sabe qual será seu novo vizinho. Ao mesmo tempo, quando um nó sair da rede, mais mensagens deverão ser trocadas para avisar todos os nós que sabem de sua existência. Num contexto de intermitência onde os nós podem sair da rede com frequência, o aumento do número de mensagens trocadas pode ser significativo.

5) Considerando um sistema de 5 nós, se o líder falhar, o primeiro que detectar sua falha vai mandar uma mensagem de eleição para todos os 3 nós (inclusive ele) com seu ID. Se houver algum nó com ID maior, vai mandar mensagem de OK de volta ao nó que iniciou a eleição e começa uma nova eleição. Quando o nó que declarou a primeira eleição receber um OK, ele desiste da eleição. O processo se repete até o nó de maior ID declarar uma eleição. Neste caso, não vai receber nenhuma mensagem de OK e se tornará líder, enviando uma mensagem a todos os nós avisando que é o novo líder.
   No melhor caso, o primeiro nó a declarar uma eleição é o de ID maior, então enviará 3 mensagens de eleição, não vai receber nenhuma mensagem de OK, se tornará líder e enviará 3 mensagens avisando que é o novo líder, num total de 6 mensagens.
   No pior caso, o primeiro nó a declarar uma eleição é o de menor ID. Este enviará 3 mensagens de eleição e receberá 3 mensagens de OK. O próximo enviará 3 mensagens de eleição e receberá 2 mensagens de OK. O próximo enviará 3 mensagens de eleição e receberá 1 mensagem de OK. O próximo enviará 3 mensagens de eleição e não receberá mensagens de OK, tornando-se líder. Assim, envia 3 mensagens avisando aos nós que é o novo líder. O total de mensagens será 21.

6) O CSMA não impede colisões, apenas diminui a probabilidade de acontecerem. Isso porque se dois dispositivos podem detectar que o canal está livre ao mesmo tempo pode haver uma condição de corrida e, assim, haver colisões. O algoritmo diminui essa probabilidade pois faz com que os dispositivos aleatoriamente esperem um tempo antes de ocupar o canal. Nós final da espera, aleatoriamente fica, em espera novamente ou não. Isso diminui as chances de dois dispositivos ocuparem o canal ao mesmo tempo, havendo colisões, mas não evita. Pode acontecer de dois detectarem que o meio está vazio ao mesmo tempo e aleatoriamente sejam escolhidos para não esperar, o que acarretará em colisões.

7) Considera-se n o número de vizinhos do nó.
Receber mensagem de eleição: No pior caso, o nó escolhido é vizinho de todos os nós, menos do que declarou a eleição e, neste caso pode receber o(n) mensagens.
Transmitir mensagem de eleição: No pior caso, o nó escolhido é o que declarou a eleição e é vizinho de todos os nós da rede. Portanto, o número de mensagens enviadas é o(n).
Receber mensagem de "já tem pai": No pior caso, o nó escolhido é vizinho de todos os nós da rede e vai receber mensagem de todos os nós menos do seu pai. O número de mensagens é o(n).
Transmitir mensagem de "já tem pai": No pior caso, o nó escolhido é vizinho de todos os nós da rede e vai mandar mensagem para todos os nós menos para o seu pai. O número de mensagens é o(n).
Receber mensagem de resultado: No pior caso, o nó escolhido é vizinho de todos os nós da rede e todos são seus filhos. O número de mensagens é o(n).
Transmitir mensagem de resultado: Vai sempre transmitir apenas uma mensagem de resultado, ao seu pai, ou nenhuma, se for o nó que declarou a eleição.

8) ACID é um conjunto de propriedades ideais a um sistema transacional. Seu nome é o formado das iniciais de tais propriedades: **A**tomicidade (Toda transação do sistema deve ser executada por completo e, caso contrário, deve ser abortada por completo), **C**onsistência (Toda transação preserva certas regras criadas pelo dono dos dados. Ex: Um banco pode ditar que uma transação não pode deixar o saldo de uma conta bancária negativo), **I**solamento (Transações executam como se fossem únicas no sistema, sem ter interferência de outras) e **D**urabilidade (Estado global do sistema independe de fatores externos. Ex: faltou luz no lugar em que está o servidor).

9) Se após a linha acquire(c1) ocorrer uma troca de contexto para uma função transferencia(c2, c1, v), essa travará o lock c2 e será impedida de continuar pois o lock c1 está travado pela primeira transação. Quando a primeira transação voltar a executar, estará travada pelo lock c2, ocorrendo deadlock. A solução é impor uma ordenação global de locks:

```
transferencia(c1, c2, v) {
   minc = min(c1,c2)
   maxc = max(c1,c2)
   acquire(minc)
   se (retirada(c1,v) >= 0)
        acquire(maxc)
        deposito(c2,v)
        release(minc)
        release(maxc)
        retorna 0
   release(minc)
```

```
    retorna -1
}
```

10) É um mecanismo de controle de concorrência que garante atomicidade em sistemas transacionais. Seu funcionamento envolve dois tipos do locks para cada objeto: Read locks, que permite outro read lock mas não um write lock. E o write lock, que não permite nenhum outro lock. É dividido em duas fases: Fase 1 (Expanding), onde os locks são adquiridos, mas nenhum é liberado; Fase 2 (Shrinking), onde locks são liberados, mas nenhum é adquirido.

    Para a variação Strong Strict Two Phase Locking (SS2PL) a fase 1 envolve:

    - Adquirir locks de leitura e determinar tudo que é necessário para alterar estado global;
    - Gerar lista com mudanças no estado global;
    - Adquirir locks de escrita.

    A fase 2 envolve:

    - Atualizar estado global, se tudo correu bem;
    - Abortar a transação sem modificar estado global, se algum imprevisto ocorreu;
    - Liberar todos os locks.