# Sistemas Distribuídos - COS470
# 2016/1

## Terceira Lista de Exercícios

ATENÇÃO! Para ajudar no treinamento para as provas faça as listas de forma que todas as respostas estejam devidamente comentadas.

**Questão 1:** Cite e explique dois desafios que precisam ser resolvidos na implementação de RPC.

**Questão 2:** Explique todos os passos envolvidos em uma chamada RPC assíncrona.

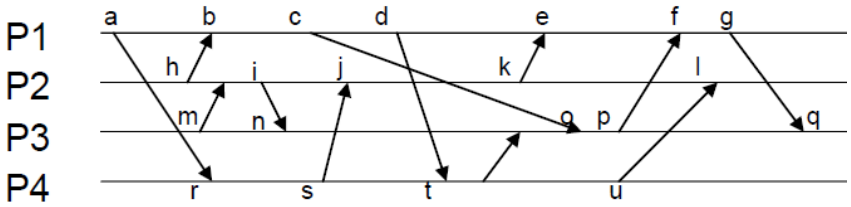
**Questão 3:** Qual o problema inerente a sistemas distribuídos que relógios sincronizados ajudam a resolver? Dê um exemplo do problema.

**Questão 4:** Considere o problema de manter a hora sincronizada entre dois relógios. Cite e explique os dois aspectos fundamentais que dificultam a sincronização.

**Questão 5:** Explique como funciona o mecanismo básico (visto em aula) para sincronização de relógios quando um computador deseja usar o relógio de outro como referência. Utilize os horários dos relógios e mostre como o relógio deve ser acertado.

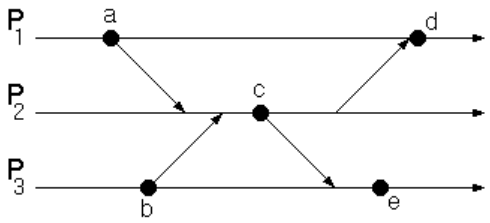
**Questão 6:** Explique sucintamente como funciona o Network Time Protocol (NTP).

**Questão 7:** Considere a figura abaixo com quatro processos e alguns eventos: Assumindo que os




valores de relógio lógico inicialmente são zero:

1. Indique dois eventos, $x$ e $y$, em processos diferentes, para os quais vale a relação $x \rightarrow y$, ou seja, $x$ "ocorreu antes"de $y$.
2. Indique dois eventos, $x$ e $y$, em processos diferentes, para os quais vale a relação $x||y$, ou seja, $x$ "ocorreu concorrentemente"a $y$.
3. Defina a relação de tempo lógico ($\rightarrow$ ou $||$) entre cada par do seguinte conjunto de eventos $E = \{b, k, n, u\}$.
4. Determine o valor do relógio lógico de Lamport para cada evento.
5. Determine o valor do relógio lógico de vetor para cada evento.
6. Indique dois eventos, $x$ e $y$, para os quais $L(x) < L(y)$ mas que $V(y) < V(x)$, onde $L(\cdot)$ e $V(\cdot)$ correspondem aos valores dos relógios lógicos de Lamport e de vetor para os eventos.
7. Para dois eventos quaisquer, $x$ e $y$, o que podemos concluir se $L(x) < L(y)$?
8. Para dois eventos quaisquer, $x$ e $y$, o que podemos concluir se $V(x) < V(y)$?




**Questão 8:** Considere a figura acima com três processos e alguns eventos indicados: Utilize o algoritmo para ordenação total de eventos (*globally ordered multicast*) para definir a ordem em que os eventos indicados serão processados. Mostre o progresso do algoritmo, indicando como suas filas locais mudam com as mensagens e eventos.

**Questão 9:** Cite e explique uma vantagem e uma desvantagem do algoritmo de exclusão mútua centralizado.

**Questão 10:** No algoritmo de exclusão mútua em anel, qual a vantagem de um nó conhecer seus dois próximos vizinhos no anel, ao invés de apenas o próximo? Seria vantajoso conhecer mais de dois vizinhos?

**Questão 11:** Em um sistema transacional, o que é ACID? Explique também seu significado.

**Questão 12:** Considere um sistema bancário transacional e a seguinte implementação da função que transfere da conta $c1$ para a conta $c2$ o valor $v$.

```
transferencia(c1, c2, v) {
   acquire(c1)
   se (retirada(c1,v) >= 0)
     acquire(c2)
     deposito(c2,v)
     release(c1)
     release(c2)
     retorna 0
   release(c1)
   retorna -1
}
```

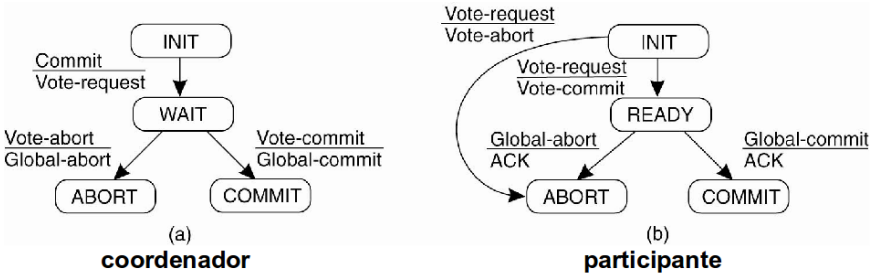
Explique o que pode acontecer com esta implementação. Como você corrigiria a implementação?

**Questão 13:** Para que serve a técnica de *Two Phase Locking* (2PL)? Explique sucintamente como a mesma funciona.

**Questão 14:** Para que serve o protocolo de *Two Phase Commit* (2PC)? Explique sucintamente como o mesmo funciona.

**Questão 15:** O protocolo *Two Phase Commit* (2PC) evita *deadlocks* em sistemas transacionais distribuídos? Explique sua resposta. Em caso negativo, como podemos lidar com *deadlocks*.

**Questão 16:** Considere o diagrama de transição de estados do protocolo *Two Phase Commit* (2PC):




1. Explique o que acontece quando um processo participante falha no estado INIT. Como o protocolo recupera desta falha?
2. Explique o que acontece quando um processo participante falha no estado READY. Como o protocolo recupera desta falha?
3. Explique o que acontece quando o coordenador falha no estado WAIT. Como o protocolo recupera desta falha?

**Questão 17:** Explique os conflitos *read-write* e *write-write* que surgem quando temos sistemas distribuídos com dados replicados.

**Questão 18:** Considere as seguintes execuções de instruções em diferentes processos, cada qual com sua memória local (assuma que inicialmente os valores das variáveis são zero). Indique quais casos (execuções) respeitam o modelo de consistência sequencial, indicando uma possível ordenação para as instruções.

1. P1: W(x,1);
   P2: R(x,0); R(x,1)
2. P1: W(x,1);
   P2: R(x,1); R(x,0);
3. P1: W(x,1);
   P2: W(x,2);
   P3: R(x,1); R(x,2);
4. P1: W(x,1);
   P2: W(x,2);
   P3: R(x,2); R(x,1);
5. P1: W(x,1);
   P2: W(x,2);
   P3: R(x,2); R(x,1);
   P4: R(x,1); R(x,2);
6. P1: W(x,1); R(x,1); R(y,0);
   P2: W(y,1); R(y,1); R(x,1);
   P3: R(x,1); R(y,0);
   P4: R(y,0); R(x,0);
7. P1: W(x,1); R(x,1); R(y,0);
   P2: W(y,1); R(y,1); R(x,1);
   P3: R(y,1); R(x,0);

**Questão 19:** Em se tratando de sistemas tolerante a falhas, qual é a diferença entre disponibilidade (*availability*) e confiabilidade (*reliability*)? Dê um exemplo.

**Questão 20:** Considere um componente com $MTTF = 2.5$ ano e $MTTR = 32$ horas. Considere o uso de componentes redundantes para projetar um sistema cujo componente tem disponibilidade de 99.99%. Assumindo que falhas deste componente são independentes no sistema, determine o número de componentes redundantes necessários.

**Questão 21:** Considere a organização de componentes redundantes TMR (*Triple Modular Redundancy*). Explique o que ocorre nos seguintes casos:

1. Exatamente um componente e um votador falha em cada linha.
2. Dois votadores falham na mesma coluna.

**Questão 22:** Explique por que falhas bizantinas são mais difíceis de lidar do que falhas que travam (*crash failures*).

**Questão 23:** Considere o protocolo de acordo bizantino com três participantes sendo um deles operando em falha bizantina. Mostre que o protocolo falha, ou seja, que os processos que não estão em falha não chegam a um consenso sobre o identificador do outro.