RESUMO:

RELOGIOS LOGICOS

- SABER SE EVENTO OCORREU ANTES DE OUTRO
    - LOCAL: com relogio local,fácil ordenar os eventos. Basta verificar os momentos correspondentes de sua ocorrencia no relogio local
    - COM VARIAS MAQUINAS: Entre dois processos a interação se dá através de send(msg) e receive(msg). A transmissão da mensagem sempre ocorre antes da recepção da mensagem.
    - RELOGIOS LOGICOS: TIMESTAMP do receive deve ser sempre maior do que o timestamp do send!

A relação aconteceu-antes-de define uma ordem parcial dos eventos de um sistema distribuído

- Por que é 'parcial'? Porque deixa eventos concorrentes
- **send(msg) acontece antes de receive(msg)**

Paxos

- É justamente isso que o Paxos garante: todas as réplicas executam exatamente a mesma operação
- timestamps unicos
- ● Os processos do Paxos têm 3 papéis distintos
    - → Proposers: processos que propõem uma próxima requisição para ser atendida
    - → Acceptors: são os processos que entram em acordo decidindo qual a próxima requisição a ser atendida
    - → Learners: processos que aprendem uma decisão
- 1. 1a Fase: Proposer envia msg Prepare com ts para maioria dos acceptors;
        - Acceptor concorda em aceitar se não tiver recebido outro prepare com ts maior
        - Se ja havia recebido prepare com ts maior envia o ts maior, se ja tiver aceito com menor ts, envia OK com valor aceito!
- 2. 2a Fase -> Se o proposer recebeu OK de uma maioria fechou, envia Accept com o valor da requisição com o maior ts
    - Se tiver recebido valor de algum processo: adota o valor
    - Acceptor só aceita se não tiver recebido outro prepare request com ts ainda maior
    - Se propose não recebe OK de uma maioria, então incrementa o rs e continua tentando.

Exemplos

1.
2.
3.
4.

PROVA GABARITO PROVA 2 ERE4

1. Sobre os diferentes tipos de difusão responda:

(A) É correto afirmar que se as propriedades da difusão causal (Causal Broadcast) são atendidas, então as propriedades da difusão atômica (Atomic Broadcast) também são atendidas? Explique.

```
Resposta:
Não, a difusão causal garante apenas a ordem das mensagens entre as quais
há uma relação de causalidade, portanto há mensagens correntes para as
quais nenhuma ordem é definida. A difusão atômica, por outro lado, pede que
todas as mensagens estejam ordenadas.
```

- (B) É correto afirmar que se as propriedades da difusão FIFO (FIFO Broadcast) são atendidas, então as propriedades da difusão confiável (Reliable Broadcast) também são atendidas? Explique.

```
Resposta:
- Sim, a difusão FIFO apresenta todas as propriedades da difusão confiável
e, além disso, garante a ordem FIFO, segundo a qual todas as mensagens
recebidas de cada origem são entregues exatamente na ordem em que foram
transmitidas.
```

2. Além dos tipos clássicos de difusão (confiável, FIFO, causal e atômica) vimos também um tipo que simplifica a difusão confiável: a difusão de melhor esforço (best-effort broadcast). Responda: (A) Qual a simplificação que a difusão de melhor esforço traz sobre a difusão confiável? (B) Faz sentido usar este tipo de difusão? Em que circunstância?

```
Resposta:
- (A) A simplificação da difusão de melhor esforço sobre a difusão
confiável ocorre com respeito a uma falha (crash) da origem da mensagem. Na
difusão confiável se um processo correto entrega a mensagem, então todos os
processos corretos devem entregar aquela mensagem mesmo que a origem falhe.
Na difusão de melhor esforço, por outro lado, se a origem falha não é mais
necessário garantir a entrega.
- (B) Em algumas aplicações só faz sentido entregar a mensagem se a origem
permanece correta, caso falhe a necessidade não existe mais.
```

3. Sobre os Detectores de Falhas responda:

```
- (A) Duas propriedades são usadas para classificar os detectores de
falhas: quais são elas? O que significam?
    - As duas propriedades são a completude e precisão. De acordo com a
completude, o detector de falhas é capaz de detectar todos os processos que
realmente falham. De acordo com a precisão, o detector de falhas não
levanta falsas suspeitas de falhas de processos que estão efetivamente
corretos.
- (B) Uma das duas propriedades é difícil de ser atingida em sistemas
reais. Qual delas? Por que?
    - A propriedade difícial de ser atingida é a precisão, porque pode ser
impossível distinguir um processo correto mas lento de um processo falho.
Assim o processo correto e lento levanta uma falsa suspeita de falha.
```

4. Com base nas duas propriedades dos detectores de falhas que você descreveu na questão anterior, os detectores de falhas são inicialmente classificados em 8 classes. Entretanto estas classes podem ser trivialmente reduzidas pela metade, isto é, para 4 classes. Responda: como é feita esta redução do número de classes de detectores?

```
Resposta: Metade das classes se enquadram na completude forte (todos
processo que falha é detectado por todos os processos corretos após um
tempo) e a outra metade se enquadra na completude fraca (após um tempo,
pelo menos 1 processo correto detecta um processo que falha). A redução das
classes é feita trivialmente, transformando a completude fraca em forte:
garantindo que o processo correto comunique a falha detectada aos demais
processos corretos.
```

5. Na nossa disciplina estudamos dois algoritmos para a difusão confiável (Reliable Broadcast). Um deles utiliza um detector de falhas. Responda:
    - (A) Explique como o detector de falhas ajuda a difusão confiável?
        - Se os processos corretos têm acesso a um detector de falhas, podem saber se a origem da difusão confiável permanece correta, neste caso não precisam retransmitir a mensagem.
    - (B) Das duas propriedades da questão anterior, quais delas o detector deve atender? Por que? Resposta:
        - Basta que o detector de falhas tenha a propriedade de completude: um processo falho deve ser detectado. A precisão não é importante: se há uma falsa suspeita de falha da origem, o que acontece é apenas que mais mensagens são transmitidas, não afetando a correção da difusão.
6. A Replicação Máquina de Estados (RME) define que um conjunto de réplicas de um processo executam o mesmo conjunto de ações, na mesma ordem, garantindo a consistência entre as réplicas. Explique como o consenso e a difusão atômica são usados para implementar a RME.

- Resposta: Os processos "clientes" enviam múltiplas requisições de forma concorrente para as réplicas utilizando difusão atômica. A difusão atômica garante que todos os processos executam as mesmas requisições na mesma ordem, utilizando o consenso para garantir que a cada passo a mesma requisição é executada por todas as réplicas.

PROVA 2 2019/1

1. Considere 4 processos (A, B, Ce D) executando as instruções abaixo. Aplique os relógios lógicos ordenando temporalmente as instruções executadas. [usa ordem lexicografica]
   - A: instr A1; A recv D; instr A2; A send B; instr A3;
   - B: B send C; instr B1; B recv A; instr B2;
   - C: instr C1; C recv B; instr C2;
   - D: instr D1; D send A; instr D2;

```
- A: instr A1[1]; A recv D [3]; instr A2 [4]; A send B [5]; instr A3 [6];
- B: B send C[1]; instr B1 [2]; B recv A [6]; instr B2 [7];
- C: instr C1[1]; C recv B [2] ; instr C2 [3];
- D: instr D1[1]; D send A [2]; instr D2 [3];
```

ORDEM TOTAL: inst A1 => B send C => inst C1 => inst D1 => inst B1 => C recv B => D send A => A recv D => inst C2 => inst D2 => inst A2 => A send B => inst A3 => B recv A => inst B2

2. Sobre os diferentes tipos de broadcast estudados explique:

Slide Broadcast

```
- (A) Broadcast atômico é FIFO? Explique.
    -
    - Não, no broadcast atômico todas as mensagens são entregues por todos
os processos na mesma ordem, porém essa ordem pode ser qualquer ordem.
Desde que seja a mesma em todos os processos.
    - Já no caso do FIFO, entrega as mensagens em ordem de um mesmo
processo origem, através da ordem que foram transmitidaes

- (B) Broadcast confiável (reliable) é best-effort? Explique.
    -
    - Não. No caso do Broadcast Confiável garante a entrega para todos os
processos... Se um processo recebeu a mensagem transmitida, mesmo que o
processo transmissor falhe, este pode realizar o envio para todos os
processos da mensagem.
        -
    - Best-Effort - Não tem ordem.
        - Além disso, Entretanto: se o processo origem falha tendo
transmitido a mensagem para alguns processos e não para outros, é isso
mesmo _Se a origem falha, pode ser que alguns processos não recebam e
portanto não entreguem a mensagem_
- (C) Broadcast causal é FIFO? Explique. inverso é vdd
    -
    -  Não! Causal broadcast garante que as mensagens são entregues de
maneira a preservar a causalidade. Se uma mensagem  m1 causou o envio de
uma mensagem m2, então m1 deve ser entregue antes de m2 em todos os
processos. Já no FIFO, garante que as mensagens enviadas por um mesmo
processo são entregues na ordem em que foram enviadas.
- (D) Broadcast atômico é causal? Explique.
    -
    - Não, a difusão causal garante apenas a ordem das mensagens entre as
quais há uma relação de causalidade, portanto há mensagens correntes para
as quais nenhuma ordem é definida. A difusão atômica, por outro lado, pede
que todas as mensagens estejam ordenadas.
```

3. Vimos dois algoritmos de Reliable Broadcast: o algoritmo básico e o algoritmo baseado em Detector de Falhas Perfeito. Sabendo que na prática é impossível garantir a precisão (accuracy) do detector, como você compara os dois algoritmos em termos do número de mensagens necessárias no pior caso? E no melhor caso?
    - RB com Detector de falhas: Melhor caso N pior caso N^2.
        - Se a mensagem é recebida de um processo correto: não precisa re-encaminhar!
        - Mas se o processo origem está falho: reenvia para todos os processos
            - Propriedades:
            1. Completude: se um processo que executou rbcast(msg) falhou, ele tem que ser detectado [como falho] para garantir que a msg seja entregue por todos os processos corretos (validade)
            2. Precisão: Não precisa. se não satisfaz a precisão, então pode ser que ocorra uma falsa suspeita: um processo correto é suspeito de ter falhado, mais mensagens transmitidas, ok.
    - RB: n^2 mensagens (todos mandam para todos)

4. (A) Vimos dois tipos de algoritmo de consenso, o consenso regular e o consenso uniforme. Explique a diferença entre os dois precisamente. -
   -
   (B) Em linhas gerais qual característica do algoritmos de consenso regular vistos em sala impede que o acordo uniforme seja garantido? - Requer a precisão, pois caso contrário não espera ACK de processo incorretamente suspeito. Assim pode violar o acordo uniforme. - Acordo uniforme: Se uma mensagem m é entregue por um processo [falho ou correto] então m é entregue por todos os processos corretos do sistema. - Consenso: todos os processos 'decidem' pelo mesmo valor, que é um daqueles que foram propostos inicialmente.
5. Considere um sistema com 7 Generais Bizantinos, no qual dois generais comandados (nodos 5 e 6) são traidores. O comandante (nodo 0) envia a ordem RECUAR. Enquanto que os dois traidores enviam sempre ATACAR. Mostre as ações executadas pelos nodos 1 e 5, detalhando todas as maiorias executadas. ?

PROVA 2 2013/1

1. Considere 4 processos (A, B, C e D) executando as instruções abaixo. Aplique os relógios lógicos ordenando temporalmente as instruções executadas.
   - A: send D; instr Al; receive C; instr A2;
   - B: send C; comandoB1; receive D; instr B2;
   - C: instr C1; receive B; instr C2; send A;
   - D: instr D1; receive A; instr D2; send B;

```
- A: send D [1]; instr Al [2]; receive C [5]; instr A2 [6];
- B: send C [1]; comandoB1 [2]; receive D [5]; instr B2 [6];
- C: instr C1 [1]; receive B [2]; instr C2 [3]; send A [4];
- D: instr D1 [1]; receive A [2] ; instr D2 [3]; send B [4];
```

ORDENAÇÃO TOTAL:

send D => send C => instr C1 => instr D1 => inst A1 => comando B1 => receive B => receive A => inst C2 => inst D2 => send A => send B => receive C => receive D => inst A2 => inst B2

2.
   - (A) Utilizando uma matriz de blocos causais e assumindo canais de comunicação FIFO é possível detectar que um determinado bloco está completo. O que um processo participante pode fazer quando detecta que um bloco está completo?
   - (B) Além de determinar que um bloco está completo, é possível também determinar que um bloco está estável. Em que contexto a estabilidade dos blocos é utilizada? Dica: pense nas propriedades garantidas pela estabilidade.
3. Responda:
   - (A) Quais propriedades diferem a difusão confiável (Reliable Broadcast) da difusão atômica (Atomic Broadcast)?
     - Na difusão confiável deve ser garantido a relação de causalidade entre mensagens, já na difusão atômica,

- (B) É correto afirmar que se as propriedades da difusão causal (Causal Broadcast) são atendidas, então as propriedades da difusão FIFO (FIFO Broadcast) também são atendidas?
  - Não, causal garante apenas a relação de causalidade entre processos. Não garante ordem FIFO de entrega de um mesmo processo.

4. O algoritmo de exclusão mútua distribuída de Ricart-Agrawala pode ser considerado uma evolução natural do algoritmo de Lamport, inclusive tendo em vista que elimina a premissa dos canais de comunicação FIFO. Explique por que os canais FIFO são necessários para o algoritmo de Lamport e por que não são necessários para o algoritmo de Ricart-Agrawala.
   - FIFO: Necessário no Lamport: 1- Requisição RC <request (tsi, i)> e enfileira na RQi [requisições ordenadas pelo relógio lógico].2. Ao receber a requisição de i, j responde com Reply(tsj, j). 3 - O processo entra na RC se a sua requisição for a 1a da sua fila e já recebeu mensagens com timestamp maior de todos os demais processos. 4. Ao terminar a execução o processo i retira a req da RQi e envia RELEASE -> todos retiram a requisição das suas filas.
     - **3** :Safety pois o canal de comunicação é FIFO (Não vão chegar mensagens antigas)
   - RICART-AGRAWALA: Não precisa de canal FIFO ! O REPLY só é recebido quando tem permissão. Quando receber todos -> garantido que não há requisições antigas pendentes.

5. Considere um sistema com 7 Generais Bizantinos, no qual dois generais comandados (nodos 1 e 6) são traidores. O comandante (nodo 0) envia a ordem ATACAR. Enquanto que os dois traidores enviam sempre RECUAR. Mostre as ações executadas pelos nodos 2 e 6, detalhando todas as maiorias executadas.

- Para resolver o problema utilizando o algoritmo de Paxos, vamos considerar um cenário com 7 generais, onde o nodo 0 é o líder (proposer) que envia a ordem RECUAR, e dois nodos (5 e 6) são traidores que sempre enviam a ordem ATACAR. Os nodos corretos devem chegar a um consenso usando Paxos, um algoritmo de consenso tolerante a falhas bizantinas. Paxos funciona em duas fases principais: preparação (prepare) e aceitação (accept).

1. Fase 1: Preparação (Prepare) Proposer (Nodo 0) envia uma mensagem prepare(n) a todos os nodos (1 a 6). Aqui, n é o número da proposta, que deve ser único e maior que qualquer número de proposta anteriormente visto por qualquer acceptor.

2. Fase 2: Aceitação (Accept)

- Acceptors (Nodos 1 a 6) respondem ao prepare(n) com uma promessa (promise(n)) de não aceitar qualquer proposta com um número menor que n. Se eles já aceitaram uma proposta anterior, eles incluem o valor da maior proposta aceita.
- Proposer (Nodo 0) coleta promessas da maioria dos acceptors. Se obtiver promessas suficientes, envia uma mensagem accept(n, v) a todos os nodos, onde v é o valor proposto (neste caso, RECUAR).
- Análise das Ações dos Nodos 1 e 5
- Vamos detalhar o que acontece com os nodos 1 e 5:
- Nodo 1 (Correto)

- Recebe prepare(n) de 0 e responde com promise(n).
- Recebe accept(n, RECUAR) de 0 e aceita a proposta. Nodo 1 mantém RECUAR como seu valor decidido.

- Nodo 5 (Traidor)
  - Recebe prepare(n) de 0. Como traidor, nodo 5 pode responder de forma maliciosa. Pode:
  - Não responder ao prepare.
  - Responder com um valor falso.
  - Recebe accept(n, RECUAR) de 0. Como traidor, pode tentar influenciar outros nodos enviando ATACAR, mas isso não afetará os nodos corretos, pois a maioria dos nodos já terá concordado com RECUAR.

Processo Completo de Paxos

- Proposer (Nodo 0) envia prepare(n) para todos.
- Nodos 1, 2, 3, 4, 5, 6 recebem prepare(n). Nodos 1, 2, 3, 4 respondem com promise(n).
- Nodo 0 coleta promessas e, tendo uma maioria (4 nodos), envia accept(n, RECUAR) para todos.
- Nodos 1, 2, 3, 4, 5, 6 recebem accept(n, RECUAR). Nodos 1, 2, 3, 4 aceitam RECUAR. Nodos 5 e 6 (traidores) podem enviar ATACAR, mas isso não mudará a decisão da maioria.
- Maioria e Decisão
- Para Paxos funcionar, é necessário que a maioria (quorum) dos nodos esteja de acordo. Com 7 nodos, a maioria é de pelo menos 4 nodos corretos.
- Nodo 1: Recebe promessas de 2, 3, 4 e aceita RECUAR.
- Nodo 5: Como traidor, pode tentar enviar ATACAR, mas a maioria já decidiu por RECUAR.
- Conclusão
  - Utilizando Paxos, os nodos corretos (1, 2, 3, 4) decidirão RECUAR, pois essa é a ordem enviada pelo proposer (nodo 0) e a maioria dos nodos corretos aceitou essa proposta. Os traidores (5, 6) podem tentar causar confusão enviando ATACAR, mas não conseguirão mudar a decisão da maioria. Portanto, a decisão final será:
    - Nodo 1: Decide RECUAR.
    - Nodo 5: Como traidor, pode reportar ATACAR, mas isso não afetará a decisão da maioria dos nodos corretos, que é RECUAR.