UNIFESSPA
UNIVERSIDADE FEDERAL DO SUL E SUDESTE DO PARÁ

Universidade Federal do Sul e Sudeste do Pará

Sistemas Distribuídos

Prof.: Warley Junior

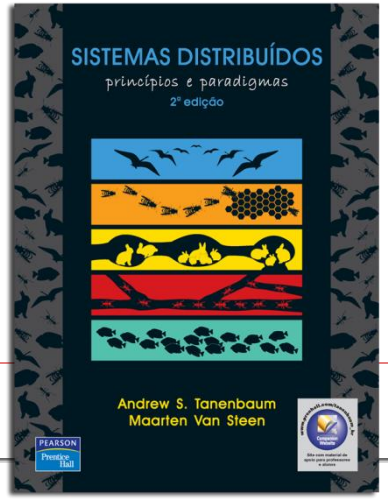
wmvj@unifesspa.edu.br

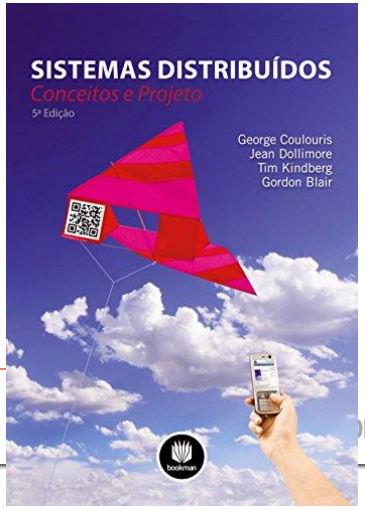
Agenda

- AULA 12:
- Sincronização
  - Relógios Físicos
  - Relógios Lógicos

Leitura Prévia

- COULOURIS, George. **Sistemas distribuídos: conceitos e projetos**. 4ª ed. Porto Alegre: Bookman, 2013.
  - Capítulo 14.
- TANENBAUM, Andrew S. **Sistemas distribuídos: princípios e paradigmas**. 2ª ed. São Paulo: Pearson Prentice Hall, 2007.
  - Capítulo 06.





Sincronização de relógios

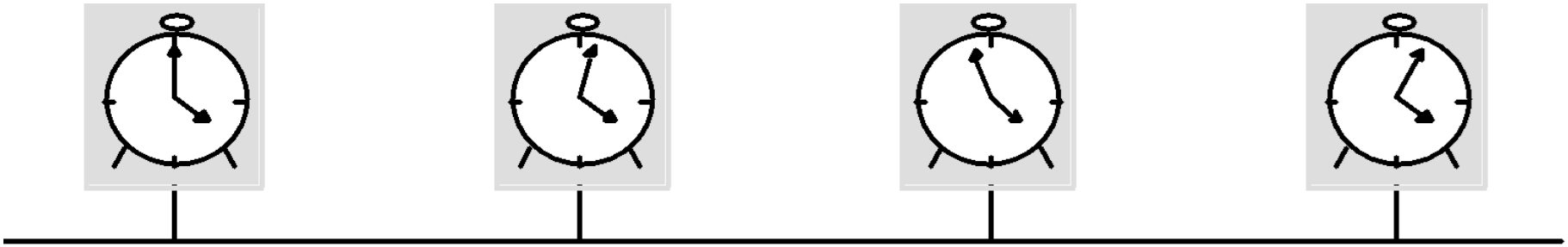
- Relógios são essenciais em computação
  - medir o tempo
  - identificar ordem de eventos
- Não há problemas de sincronização em sistemas centralizados
- Em sistemas distribuídos, obter um horário comum a vários computadores não é trivial.

Problema da Sincronização em SD

- Como os processos cooperam e sincronizam uns com os outros em um SD?
- Semáforos em sistemas individuais com uma única CPU.
- Esses métodos não funcionarão em sistemas distribuídos. Por quê?
  - Porque dependem implicitamente da existência de memória compartilhada.

Problema da Sincronização

- Dois processos que interagem usando semáforo.
- Devem ser capazes de acessar o semáforo.
- Se ocorrerem dois eventos em um SD.
- Como decidir sobre a ordem relativa dos eventos.
- Um evento precede outro evento?
- Difícil determinar se os eventos ocorrem em máquinas diferentes.



Problema da Sincronização

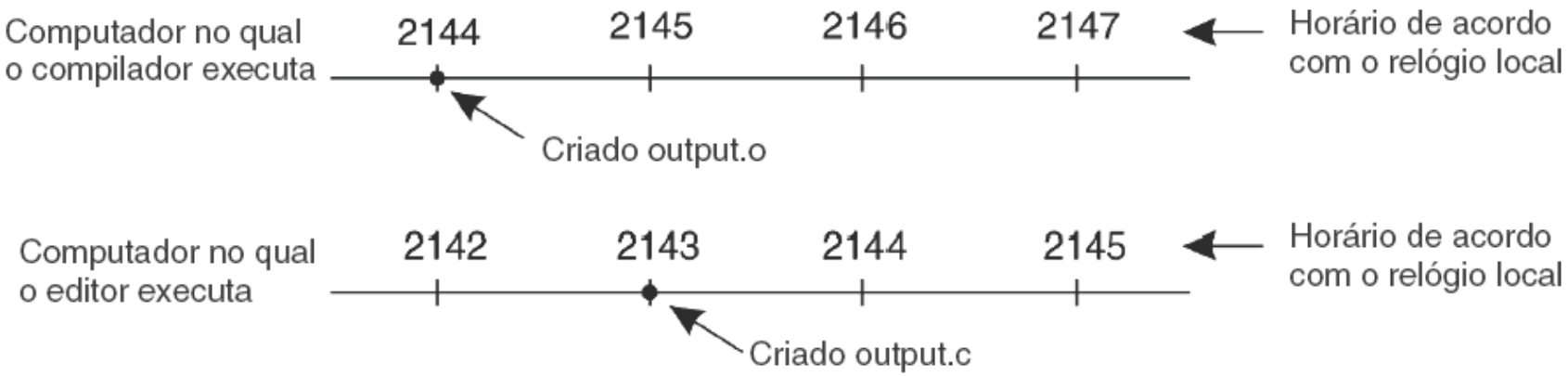
- Cada processador tem um componente chamado relógio.
- Relógios em um SD podem:
  - Acusar horas diferentes entre si (defasagem interna).
  - Acusar hora diferente da hora real externa (defasagem externa).
  - Ter velocidades diferentes (defasagem variável).

Problema da Sincronização

- Relógios sincronizados são necessários para uma série de aplicações:
  - Identificar atualidade de mensagens.
  - Aplicações de tempo real.
  - Controle de versões.

Sincronização de Relógio

- Em um sistema distribuído:
  - Alcançar acordo no tempo não é trivial!



Sincronização de Relógio

- A sincronização do relógio não precisa ser absoluta![Lamport, 1978]:
  - Se dois processos não interagem, seus relógios não precisam ser sincronizados.
  - O que importa não é que todos os processos concordem exatamente em que horas são.
  - Mas concordam com a ordem em que os eventos ocorrem.

Sincronização de Relógio

- Relógios Lógicos:
  - Necessidade apenas da consistência interna dos relógios?
  - Prover identificar a ordem dos eventos.
  - Não se os relógios estão próximos do tempo real.
- Relógios Físicos:
  - Para cenários onde os relógios devem ser apenas iguais.
  - E não devem se desviar do tempo real.

RELÓGIOS FÍSICOS

Relógios Físicos

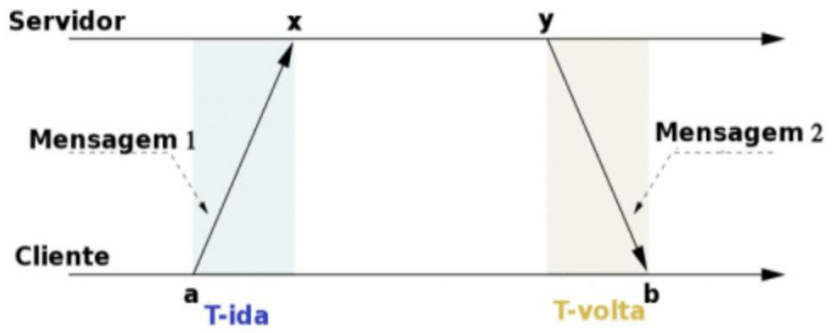
- GMT: *Greenwich Mean Time*
- BIH: *Bureau Internacional de I' Heure*
- TAI: *International Atomic Time*
- UTC: *Universal Coordinated Time*
- NIST:*National Institute of Standard Time*
- WWV: Estação de rádio de ondas curtas
- GEOS: *Geostationary Environment Operational Satellite*

Relógios Físicos - Problemas

- O tempo do relógio físico não pode voltar (atrasar)
  - Solução é diminuir a frequência do *clock*.
- É difícil medir com precisão o *delay* da rede.
- Atraso na rede mal calculado para computar o tempo certo no ajuste dos relógios.
- Algoritmos de como o de Berkeley tentam atenuar este problema.

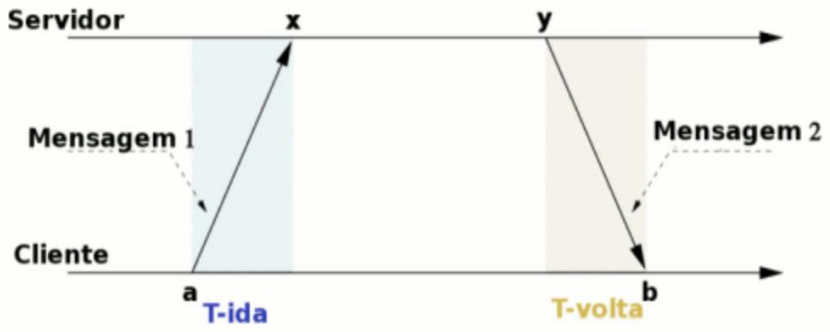
Network Time Protocol (NTP)

- Clientes consultam servidor de tempo que possui relógio de alta precisão
- Funcionamento:
  1. Uma máquina cliente envia mensagem para o servidor de tempo perguntando pela hora atual
  2. Servidor de tempo responde o mais rápido possível, com uma mensagem contendo a hora atual C_{UTC}
  3. O cliente obtém uma resposta e ajusta seu relógio
- Problemas?



Network Time Protocol (NTP)

- Clientes consultam servidor de tempo que possui relógio de alta precisão
- Funcionamento:
  1. Uma máquina cliente envia mensagem para o servidor de tempo perguntando pela hora atual
  2. Servidor de tempo responde o mais rápido possível, com uma mensagem contendo a hora atual C_{UTC}
  3. O cliente obtém uma resposta e ajusta seu relógio
- Problemas?
  - Há atraso no envio das mensagens na rede
  - O tempo pode retroceder

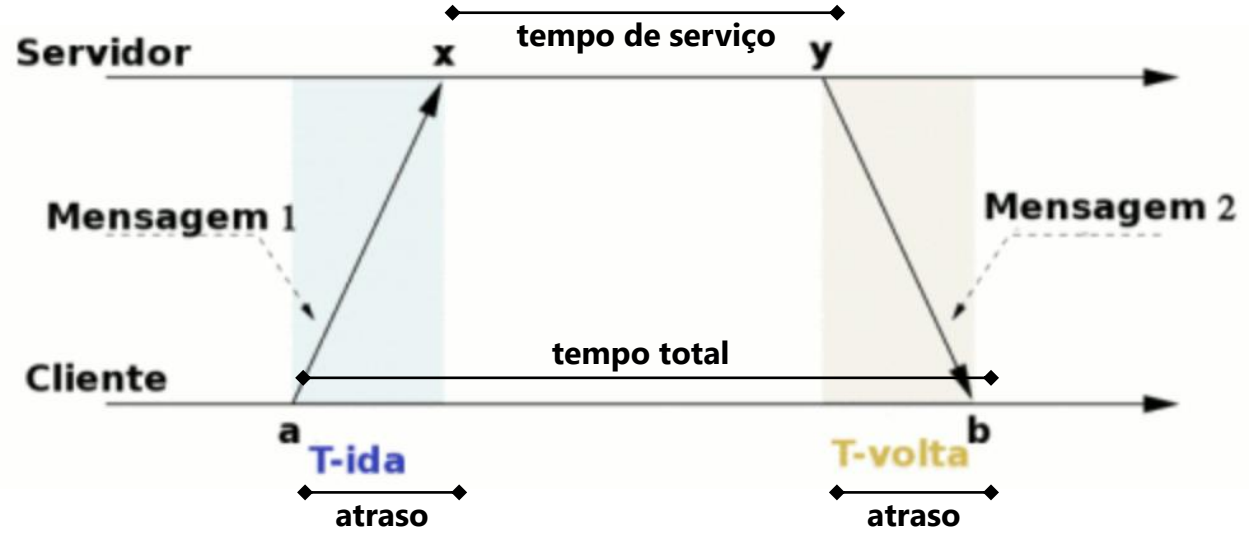


Network Time Protocol (NTP)

Problemas

1. Atraso no envio das mensagens na rede
   - Solução: ajustar relógio considerando estimativa de atraso
     - Resultado é melhorado usando histórico de média de atrasos
2. O tempo pode retroceder
   - Solução: corrigir a hora mudando o tempo gradativamente
     - Ex.: se cada interrupção da máquina adiciona 10ms ao relógio
       - Atrasar atualizando a cada 9ms ao invés de 10ms
       - Adiantar atualizando a cada 11ms ao invés de 10ms

Network Time Protocol (NTP)



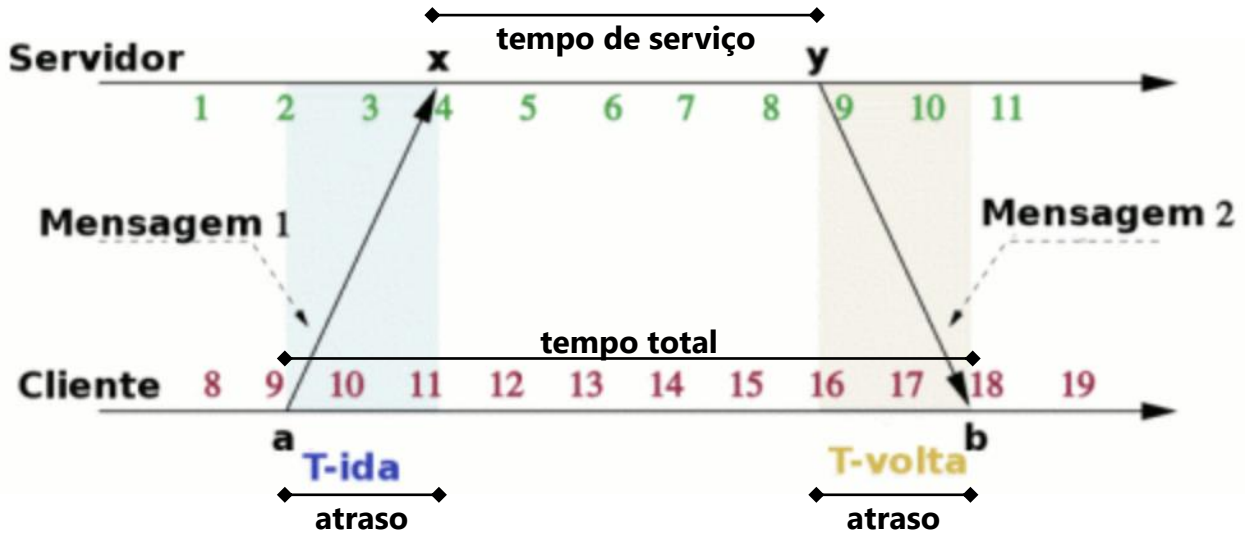
NTP

atraso: tempo para transmissão de mensagem

deslocamento: diferença de tempo entre os relógios

- **atraso = [(b – a) – (y – x)] / 2**
- deslocamento = [(x – a + atraso) + (y - b - atraso)] / 2
- **deslocamento = [(x – a) + (y – b)] / 2**

Network Time Protocol (NTP)



NTP

atraso: tempo para transmissão de mensagem

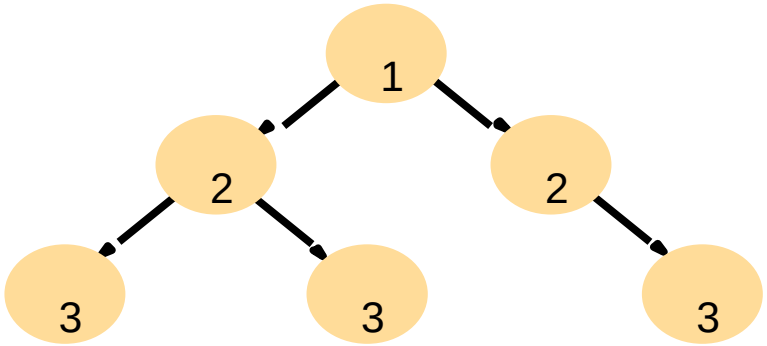
deslocamento: diferença de tempo entre os relógios

- **atraso = [(b – a) – (y – x)] / 2**
  - [(18 - 9) - (9 - 4)] / 2 = [9 - 5] / 2 = 4 / 2 = 2
- **deslocamento = [(x – a) + (y – b)] / 2**
  - [(4 - 9) + (9 - 18)] / 2 = [-5 -9] = -14 / 2 = -7

E agora, como ajustar o relógio?

Network Time Protocol (NTP)

- Cada computador pode possuir um nível de hierarquia (estrato)
  - Relógio de referência: nível 0
  - Servidor que possui o relógio de referência: nível 1
    - Se servidor A tem nível *k*, B obtém horário de A, B possui nível (*k*+1)
    - Prevalece o horário do servidor com menor estrato

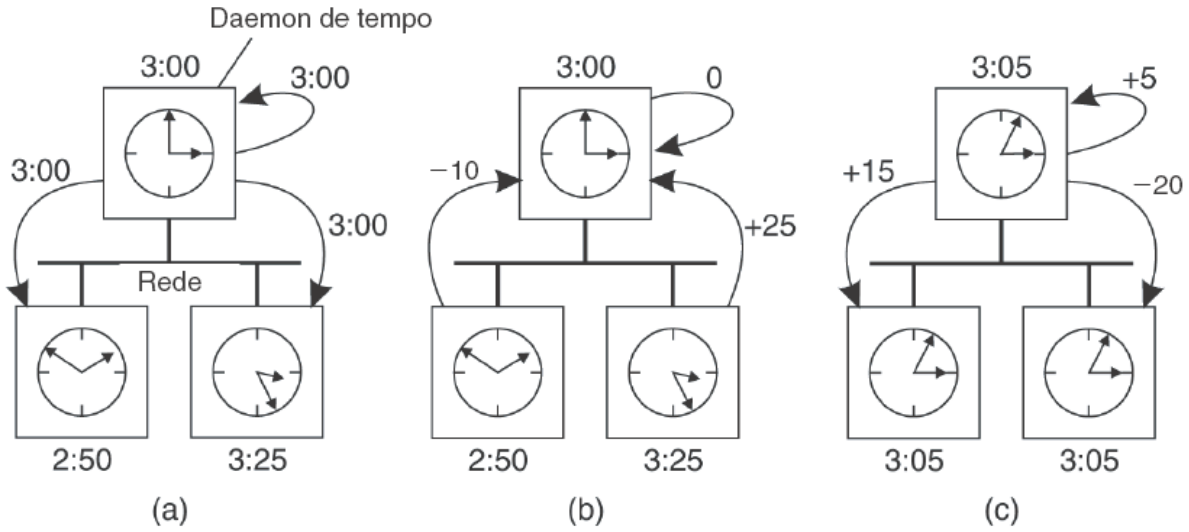


Algoritmo de Berkeley

- Útil quando não há um relógio de referência
  - O importante é que as máquinas tenham tempos aproximados
- Funcionamento:
  - O servidor responsável (*daemon de tempo*) consulta todas as máquinas de tempos em tempos, obtendo o horário de cada uma
  - Depois calcula a média dos horários e informa a cada computador o deslocamento de tempo a ser feito
    - O servidor pode inclusive mudar a própria hora

Algoritmo de Berkeley

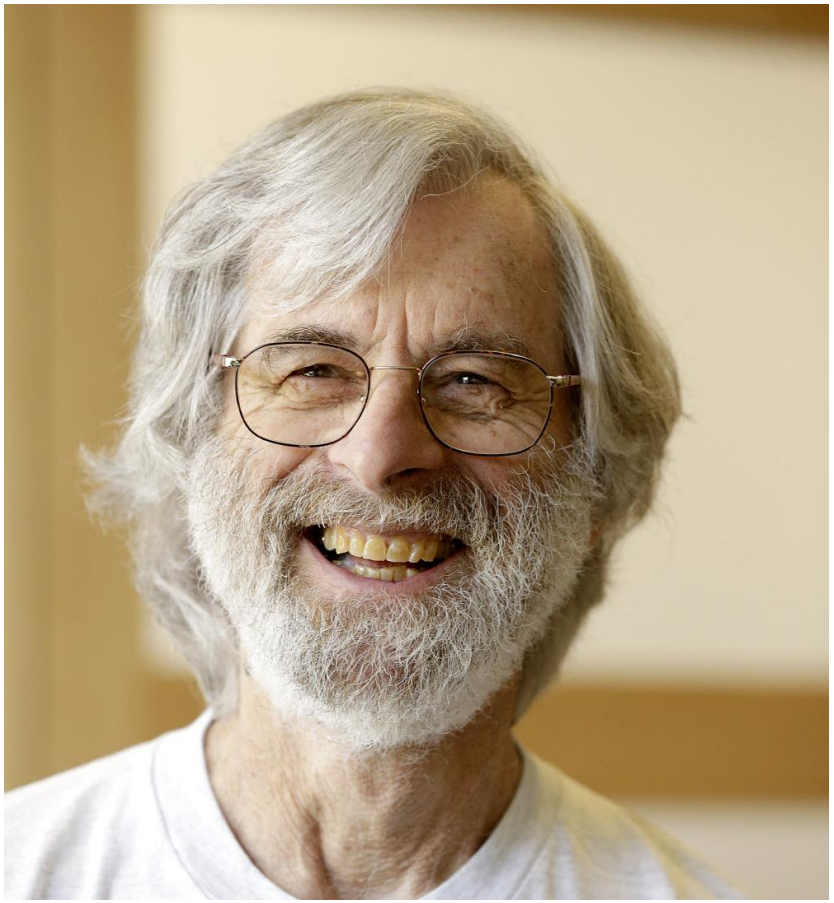
- Não dispõe de uma máquina com receptor de Tempo Universal Coordenado.
- O Servidor de Tempo requer periodicamente de cada máquina, o tempo do seu relógio.
- O servidor de tempo calcula a média e diz para cada máquina como ajustar seu relógio.



RELÓGIOS LÓGICOS

Relógios Lógicos

- Proposto por Leslie Lamport (o cara de SD)
- **Objetivo**: estabelecer a ordem de eventos independentemente da hora real
- Embora a sincronização de relógios seja possível, não precisa ser absoluta:
  - É necessário sincronizar processos que não interagem entre si? **Não!**



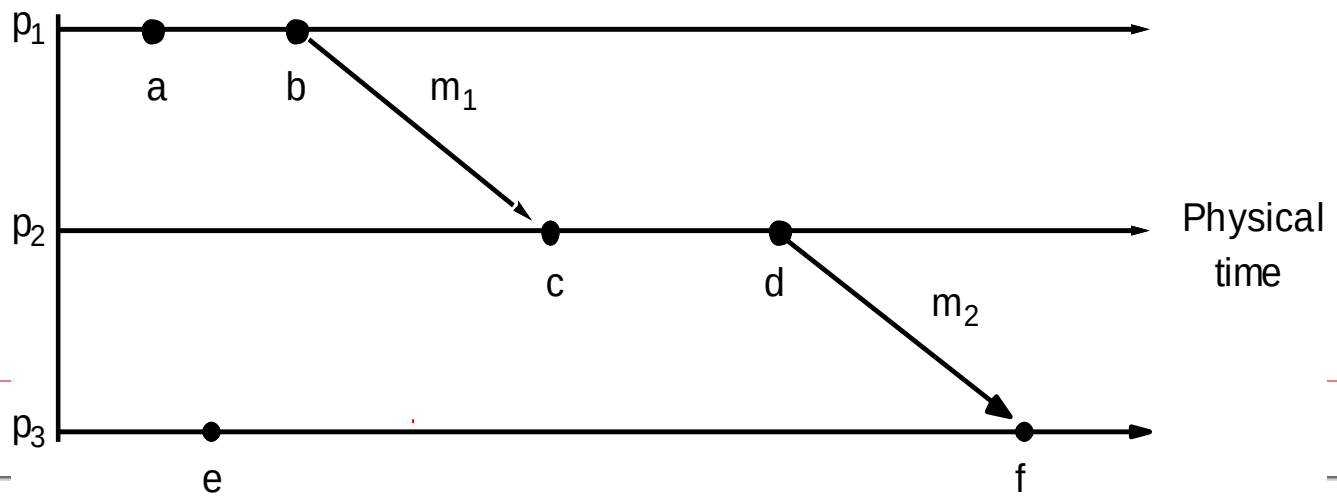
Relógios Lógicos [Lamport, 1978]

- Definiu uma relação denominada acontece antes.
- $a \rightarrow b$, (significa *a* acontece antes de *b*)
- Se a e b são eventos de um mesmo processo p, e a ocorre antes de b ($a^p \rightarrow b$) então $a \rightarrow b$
- A relação é transitiva. Logo, se $a \rightarrow b$ e $b \rightarrow c$, então $a \rightarrow c$
- O tempo em que um evento acontece é dado por $C(a)$
  - Logo, $C(a) < C(b)$
- O tempo do relógio deve sempre "andar" para frente!

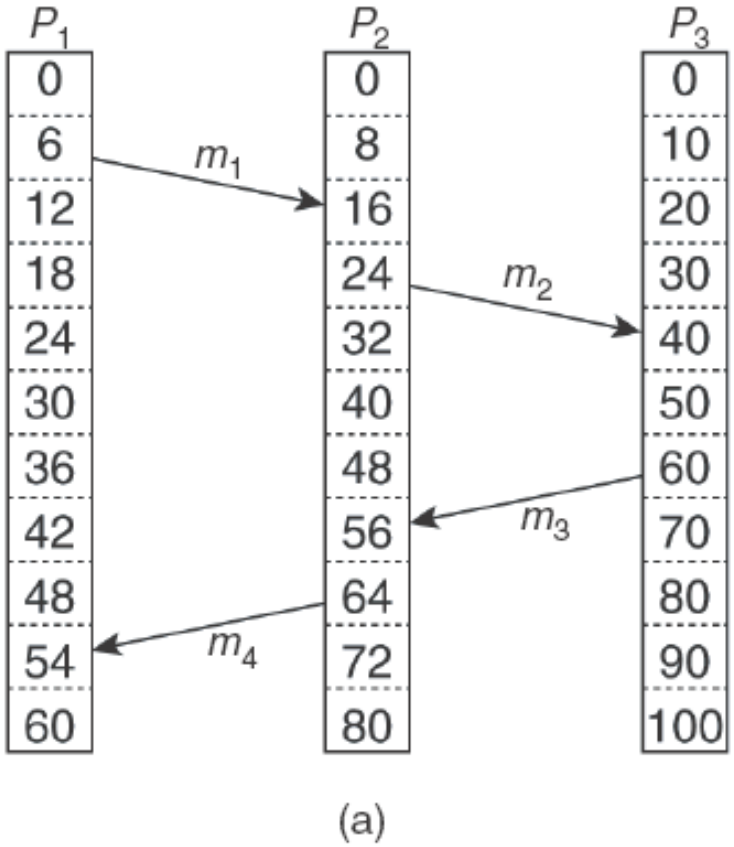
-

Relógios Lógicos [Lamport, 1978]

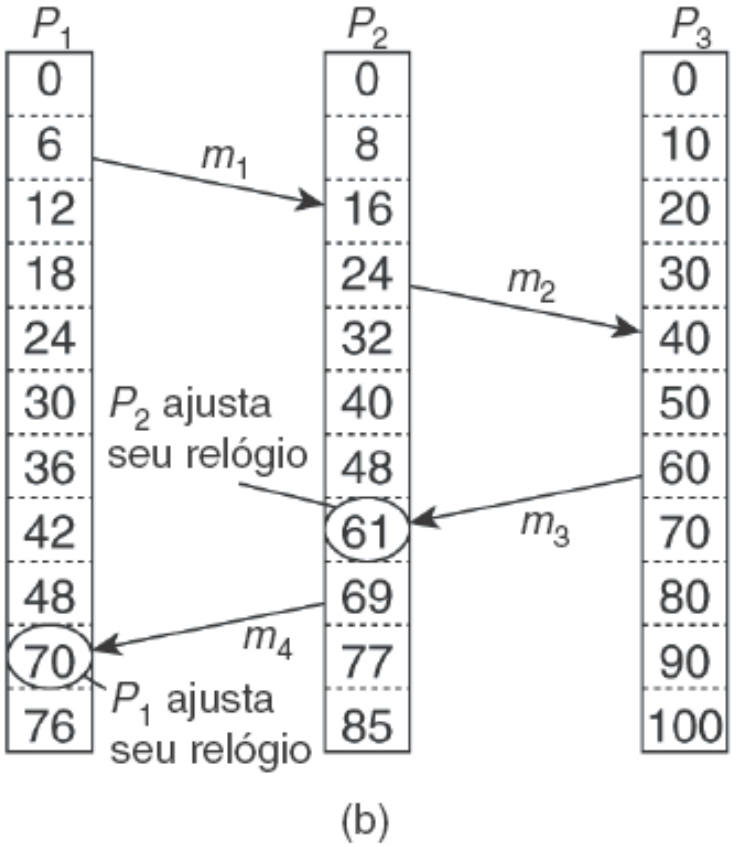
- Nem todos os eventos podem ser relacionados através da relação "acontece antes".
- Consideremos *a* e *e* (processos sem a existência de mensagens entre os processos)
- São definidos como processos concorrentes; a||e



Relógios Lógicos [Lamport, 1978]



(a)



(b)

Arquitetura: relógios lógicos

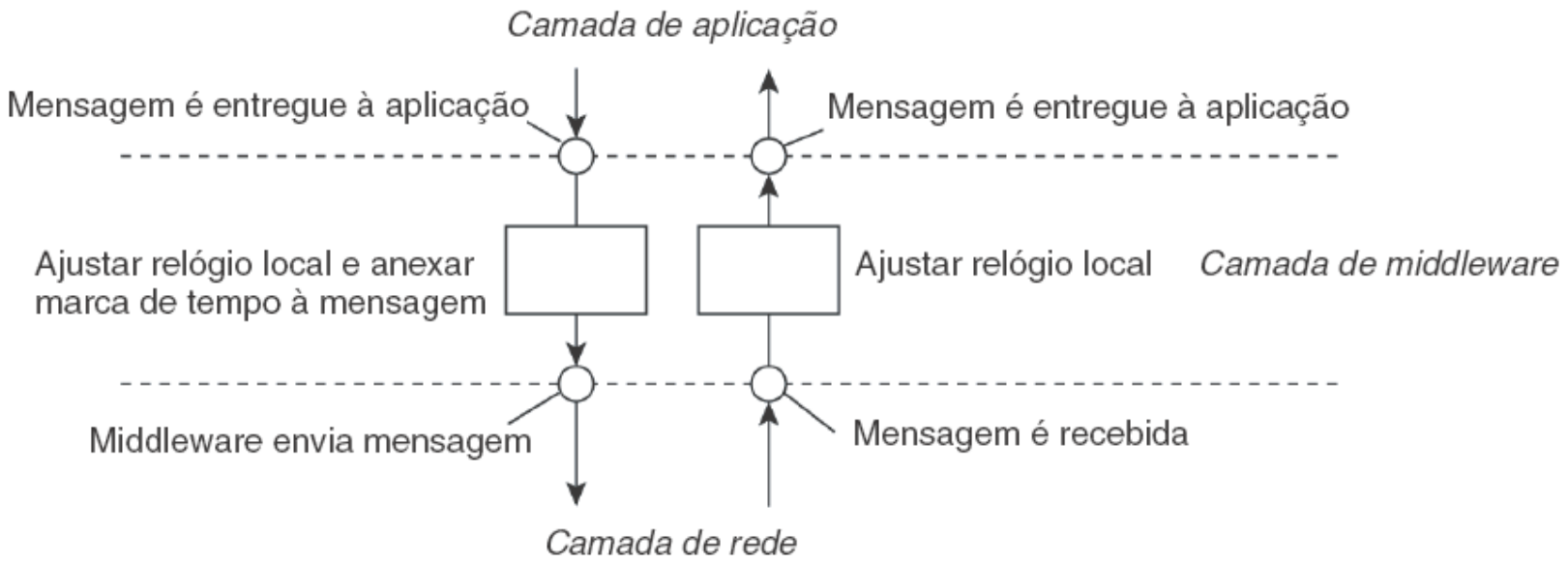
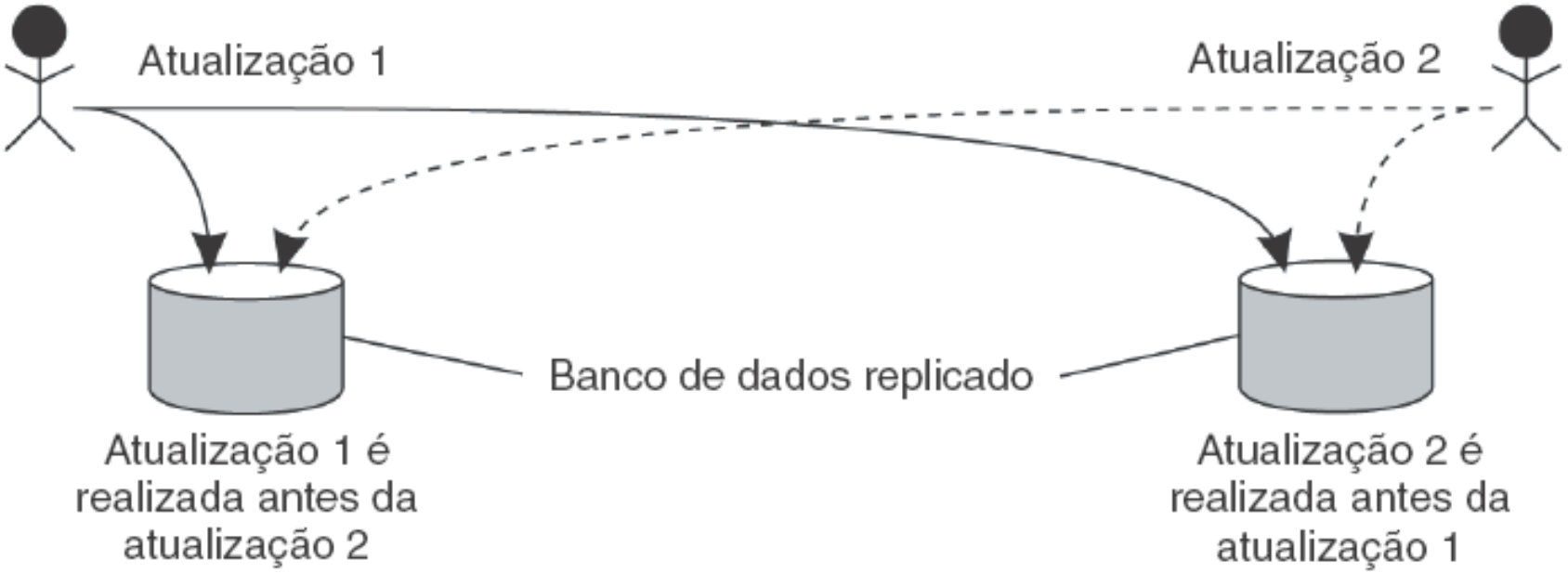


Figura 6.10 Posicionamento de relógios lógicos de Lamport em sistemas distribuídos.

Multicast Totalmente Ordenado



Ordenação total de eventos

- *Multicast* que entrega mensagens na mesma ordem a cada receptor
  - Mensagens transportam marca lógica de tempo de seu remetente
  - Mensagens são ordenadas em fila de cache local pela marca lógica de tempo
  - Após receber mensagem, reconhecimento (*ack*) é enviado em multicast
  - Mensagens só são entregues à aplicação após reconhecimento de todos os processos e quando forem a primeira mensagem da fila
    - processo com menor *id* tem prioridade quando clocks são iguais

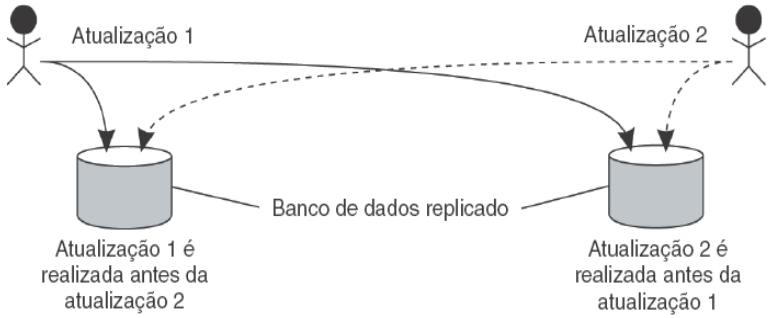
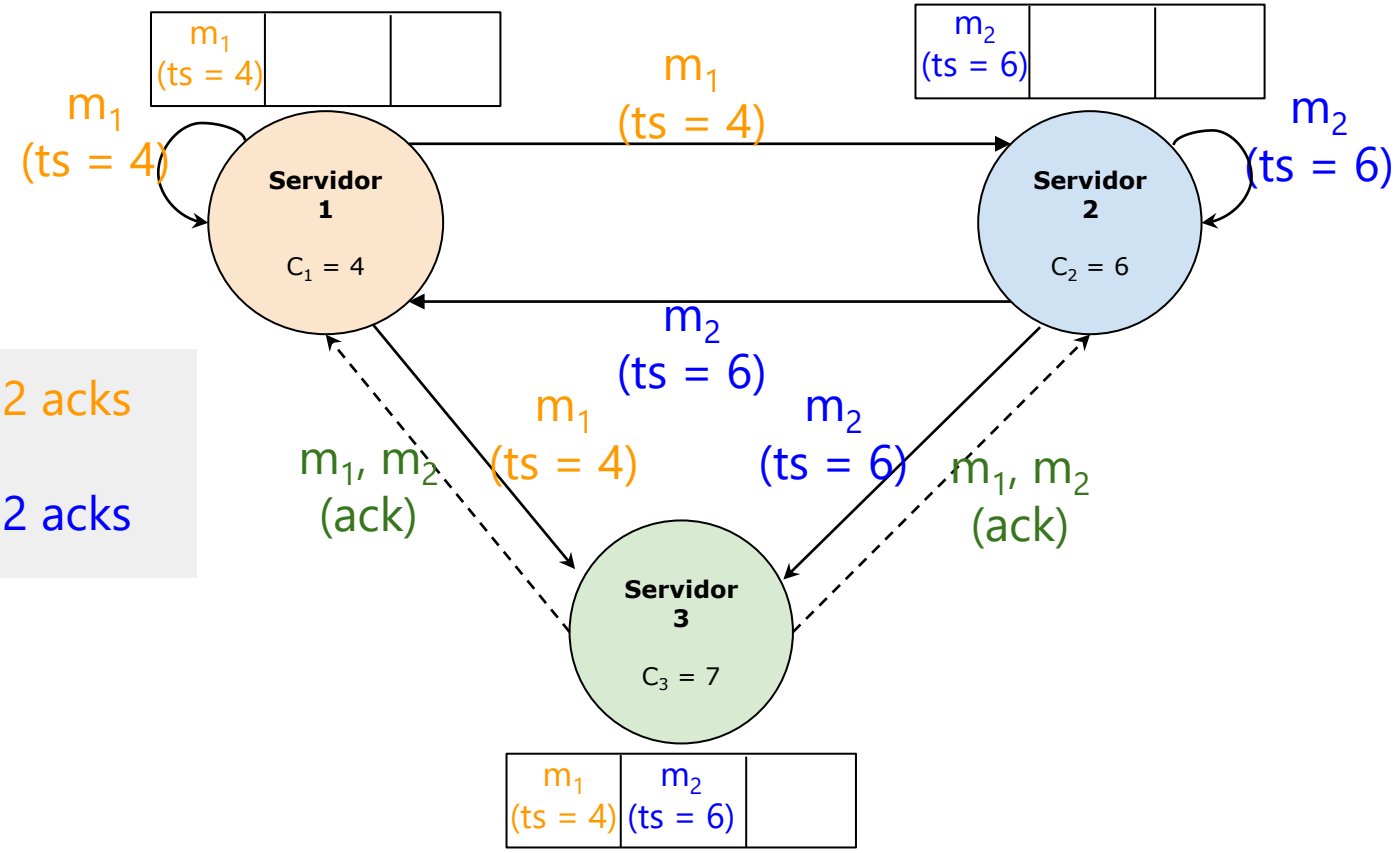


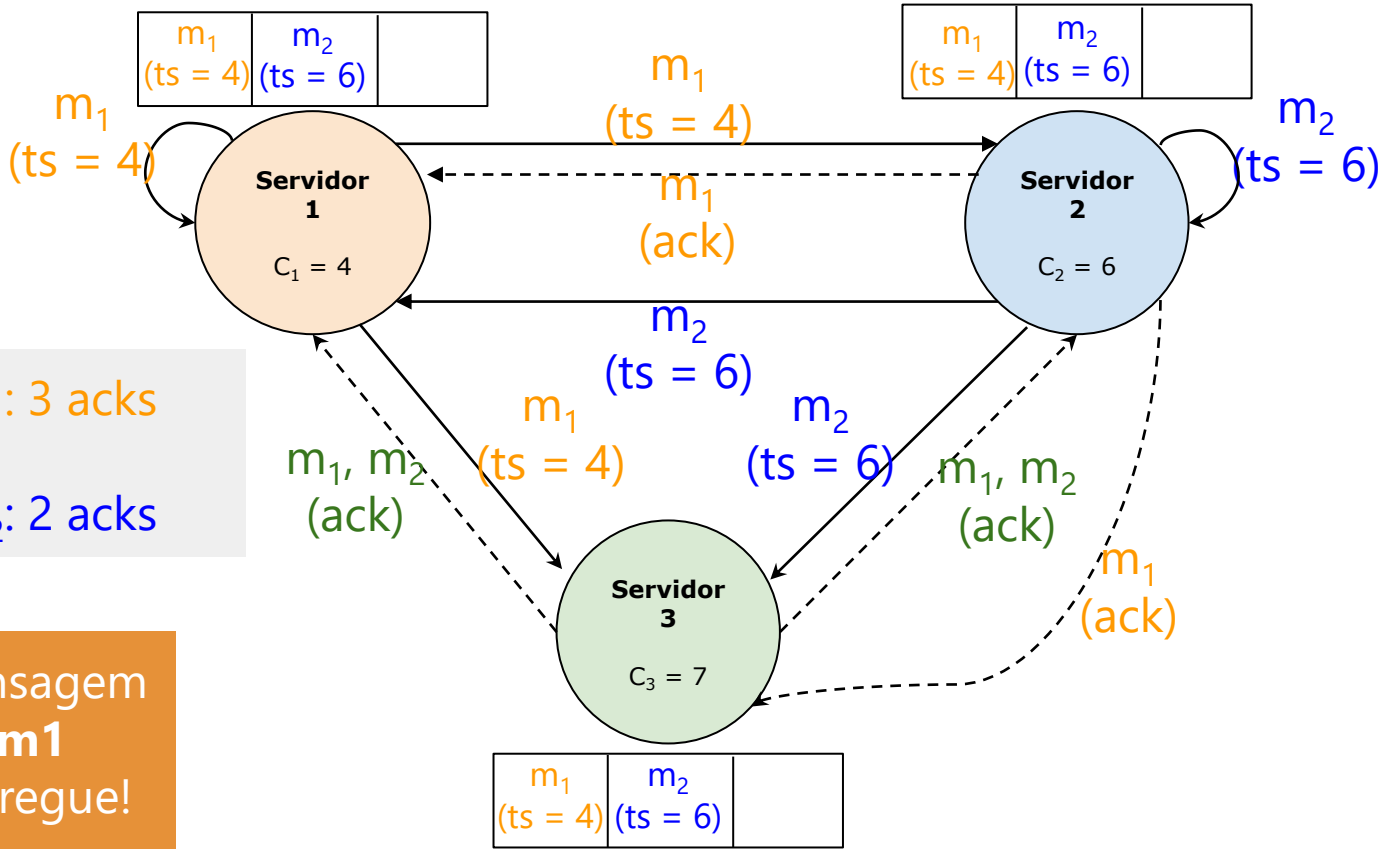
Figura 6.11 Atualização de banco de dados replicado que o deixa em estado inconsistente.

Ordenação total de eventos



- m_1: 2 acks
- m_2: 2 acks

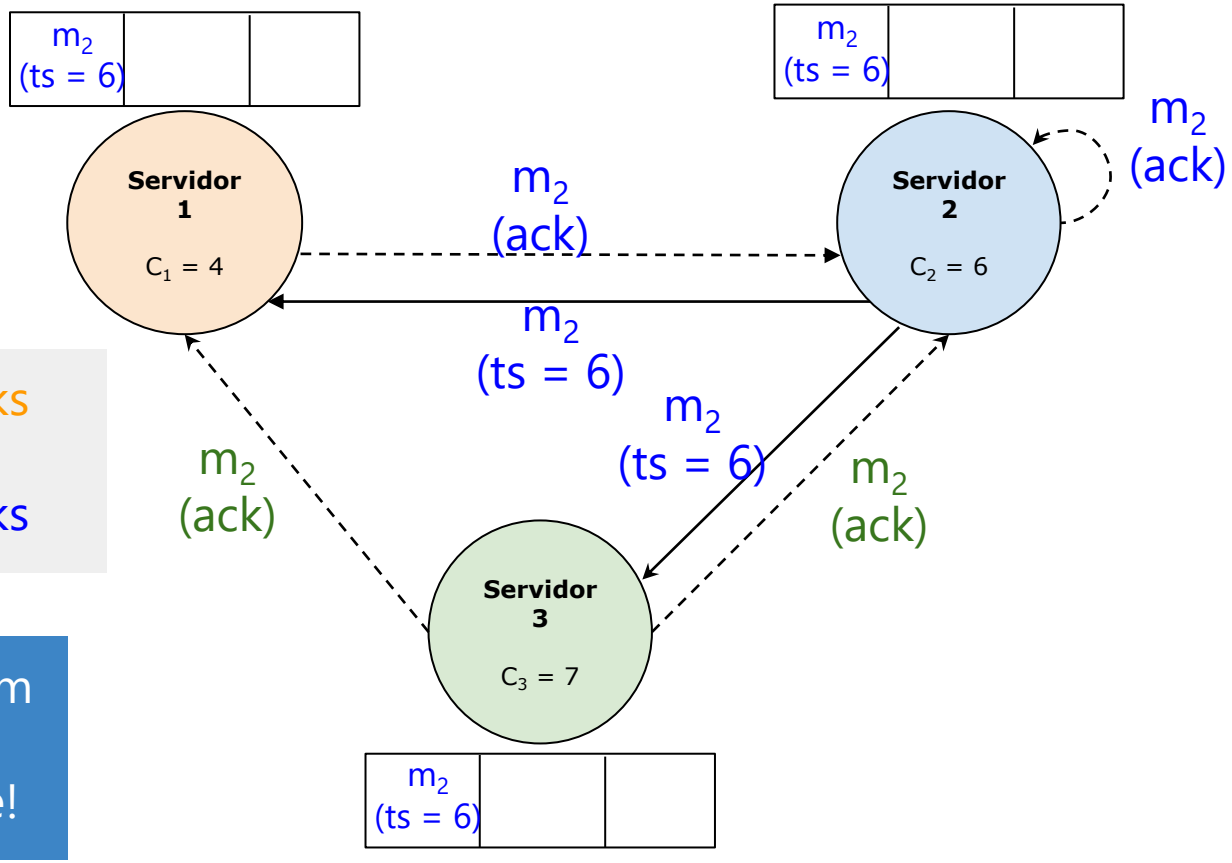
Ordenação total de eventos



- m_1: 3 acks
- m_2: 2 acks

mensagem **m1** entregue!

Ordenação total de eventos



- m_1: 3 acks

- m_2: 3 acks

mensagem **m2** entregue!