Sistemas Computacionais Distribuídos

Prof. Marcos José Santana

SSC-ICMC-USP

São Carlos, 2008

Grupo de Sistemas Distribuídos e Programação Concorrente

Departamento de Sistemas de Computação - SSC

Sistemas Computacionais Distribuídos

11a. Aula

Controle de Concorrência

Conteúdo

- Controle de concorrência
- Deadlock
- Protocolos de Commit

Controle de concorrência

- Acesso aos dados por mais de uma transação concorrentemente.
- Mecanismo para garantir a integridade das informações.
- Por que permitir concorrência?
  - Aumento de desempenho
  - Intercalação de operações

Controle de concorrência

- Operações intercaladas devem produzir o mesmo efeito se fossem realizadas serialmente, sem concorrência.
  - Teoria da Seriabilidade:

"toda execução de transações equivalente a uma execução serial é correta e chamada de serializável"

- Reservar um arq. Para uso de uma transação particular.

Controle de concorrência

- Técnicas Disponíveis
  - **Lock** (bloqueio)
  - **Timestamp**
  - **Controle otimista**

Controle de concorrência

- **Lock** (bloqueio)
  - Garante que um dado fique inacessível enquanto este está em um estado inconsistente.
  - Exclusivo ou Compartilhado.
  - Caso mais simples:
    - 1 lock ⟷ 1 transação
      - proprietário lock
  - Granularidade do bloqueio.

Controle de concorrência

- Transação deve:
  - Fazer lock no “dado” antes de fazer o acesso.
  - Não fazer lock antes de verificar se o dado já está em “lock”.
  - Liberar “obrigatoriamente” um dado que foi colocado em “lock”.

Controle de concorrência

- **Bloqueio Bifásico (TwoPhaseLocking)**

  - Fase de crescimento
    - processo adquire todos os bloqueios que ele necessita,
  - Fase de encolhimento
    - processo libera os bloqueios.

Controle de concorrência

- **Bloqueio Bifásico (TwoPhaseLocking)**
  - Escalonamentos formados pela intercalação das transações serão serializáveis.
  - **Deadlocks:** dois processos tentam adquirir o mesmo par de bloqueios, mas na ordem oposta.

Controle de concorrência

- **Timestamps**

  - Ordem de execução das transações conflitantes é determinada em tempo de execução.

  - Selecionar ANTES da execução uma ordenação entre as transações com marcadores de tempo.

Controle de concorrência

- **Timestamps ...**

  – Ordena todas as transações do sistema, sistema,utilizando marcadores de tempo.

  - W-timestamp(Q): maior marcador de tempo de qualquer transação que executa uma **atualização** (update (R)) com sucesso.

  - R-timestamp(Q): maior marcador de tempo de qualquer transação que execute uma **leitura** (fetch (R)) com sucesso.

Controle de concorrência

- **Timestamps ...**

– Se Ti emite um fetch(R) :

- Se TS(Ti) < W-timestamp(Q), Ti precisa ler um valor de Q que já foi sobrescrito → a operação fetch (R) é rejeitada e Ti é repetida.

- Se TS(Tj) >= W-timestamp(Q), a operação fetch (R) é executada, e R-timestamp(Q) é ajustado para o máximo de R-timestamp(Q) e TS(Ti).

Controle de concorrência

- **Timestamps ...**

- Se Ti emite um update (R)
  - Se TS(Ti) < R-timestamp(Q), o valor de Q que Ti estiver produzindo foi necessário anteriormente e foi assumido que nunca seria produzido $\longrightarrow$ a operação update (R) é rejeitada e Ti é refeita.

  - Se TS(Ti) < W-timestamp(Q), Ti está tentando gravar um valor obsoleto de Q $\longrightarrow$ a operação update (R) é rejeitada e Ti é repetida.

Controle de concorrência

- **Controle Otimista da Concorrência**

  – Controla arquivos lidos ou escritos.

  “ vá em frente e faça tudo o que você precisa fazer, sem prestar atenção naquilo que os outros estão fazendo; se houver algum problema, preocupe-se com ele mais tarde”.

- .

Controle de concorrência

- **Controle Otimista da Concorrência ...**

  – São otimistas porque confiam (a favor da eficiência) na tese de que a probabilidade de se ter um conflito entre transações concorrentes é muito pequena.

  – Lock é usado onde há maior probabilidade de conflito.

Controle de concorrência

- **Controle Otimista da Concorrência ...**
- Melhor em áreas privadas.
  - Cada transação muda seu arquivo privado sem interferir nos outros e sem sofrer interferência.
- Caso falhe, todas as transações devem ser executadas novamente.
- Sistema fortemente carregado: maior probabilidade de ocorrência de falhas.

Controle de concorrência

- **Controle Otimista da Concorrência ...**

  - **Leituras**: completamente livres.

  - **Escritas**: severamente restritas.
    - 3 fases: Leitura,Validação e Escrita.

Controle de concorrência

- **Controle Otimista da Concorrência …**
  - **Fase de Leitura**: escritas são feitas em cópias locais do dado.
  - **Fase de Validação**: assegura que modificações não causam inconsistências. Parte complicada!
  - **Fase de Escrita**: cópias locais são tornadas globais (públicas). - Rápida!
  - **Se a Validação falha**… transação é abortada e reinicia como uma nova transação.

Controle de concorrência

- Deadlock

  – Quando duas ou mais transações encontram-se em estado de espera simultâneo, cada uma esperando pela outra para liberar um bloqueio antes que possa prosseguir.

Controle de concorrência

- Deadlock ...

  - 2 tipos de deadlocks distribuídos:
    - Ocorridos na comunicação entre processos.
    - Ocorridos na alocação de recursos.

  - Um sistema deve **detectar** e **resolver** deadlocks

Controle de concorrência

- Estratégias para tratar deadlocks:
  - Algoritmo do avestruz (ignorar o problema).
  - Detecção (permitir a ocorrência do deadlock, detectar sua ocorrência, e tentar a recuperação).
  - Prevenção (tornar estruturalmente impossível a ocorrência de deadlocks).
  - Evitar a ocorrência de deadlocks com uma política de alocação de recursos bastante cuidadosa.

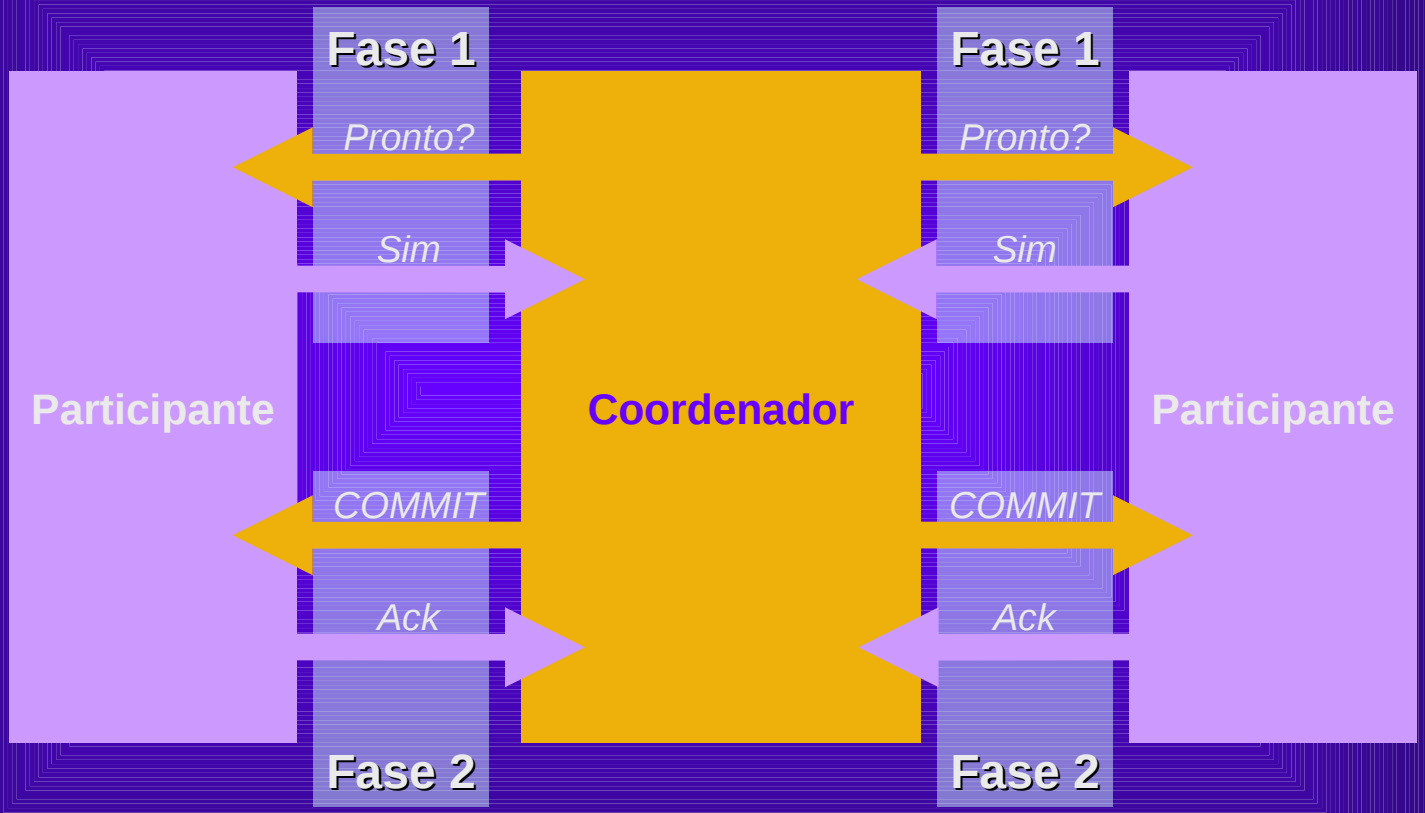
Protocolos de commit

- Commit em duas fases:

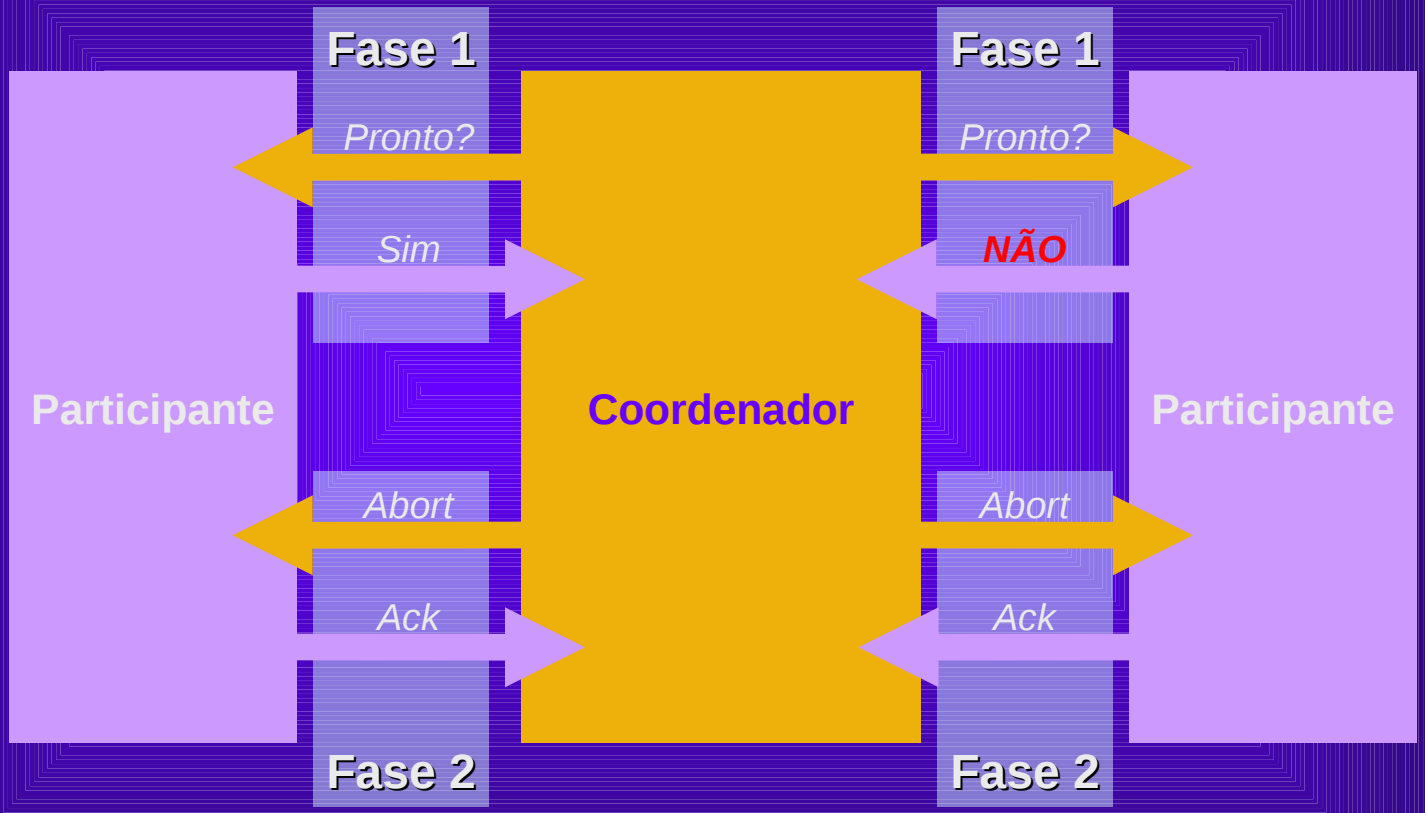
  – um coordenador

    - Uso de *stable storage*
    - Nenhum nó pára para sempre
    - Todos podem falar com todos

Protocolos de commit – duas fases



Protocolos de commit – duas fases

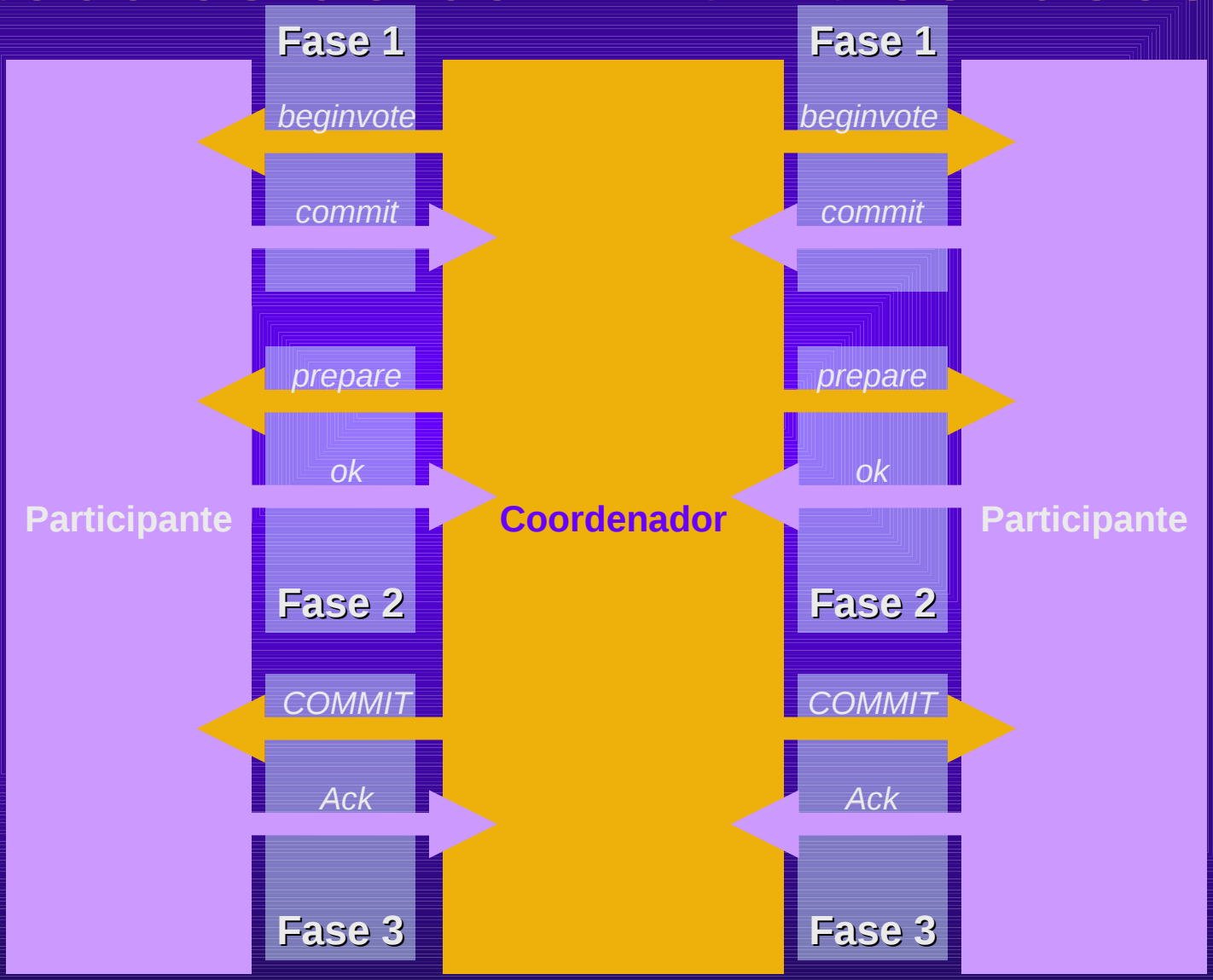


Protocolos de commit

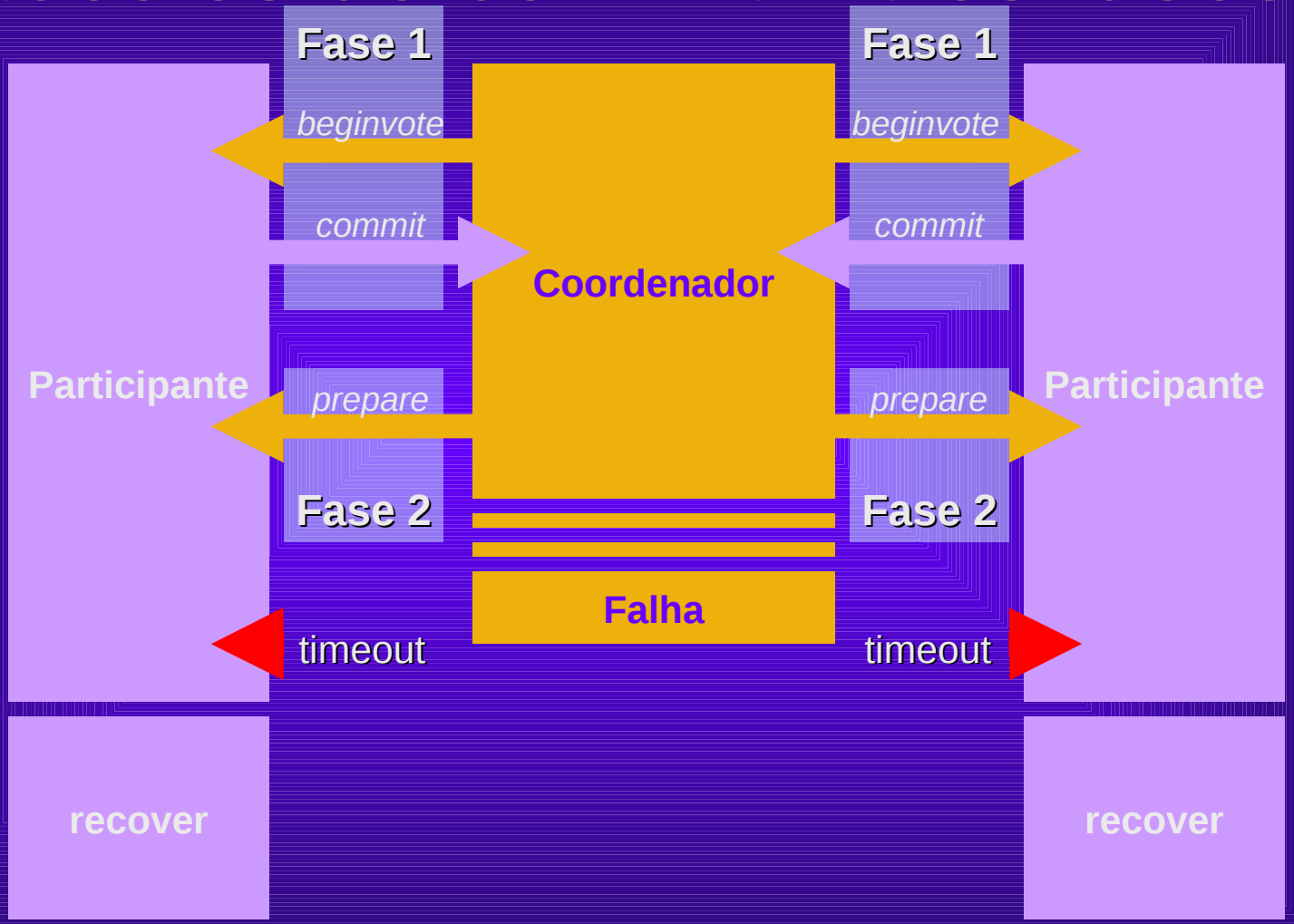
- Commit em três fases

  - Evita bloqueio de recursos no caso de falha
  - Incorporação de timeout

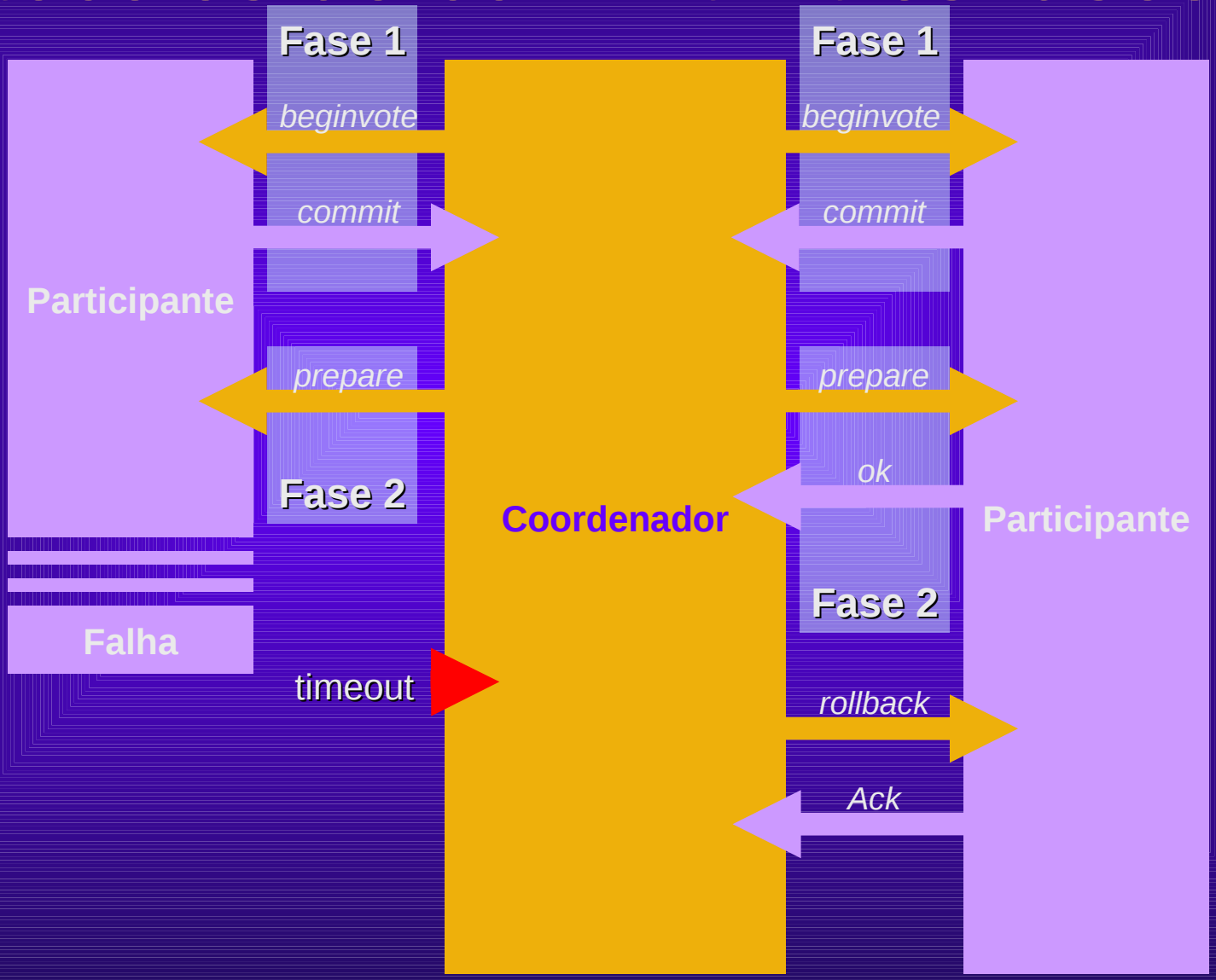
Protocolos de commit – três fases



Protocolos de commit – três fases



Protocolos de commit – três fases



Fim!