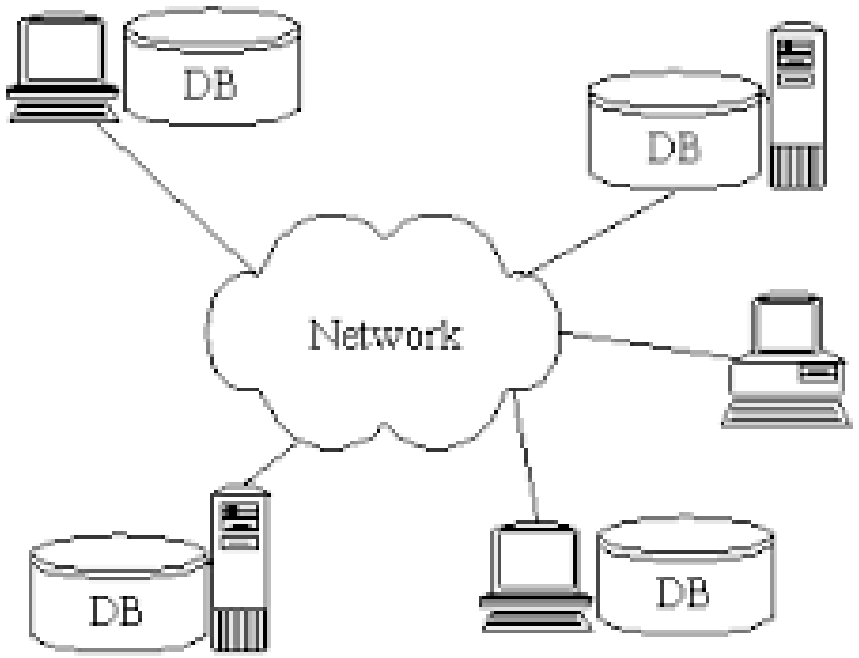
Sistemas Distribuídos
Aula 17

Roteiro

- Estado global distribuído
- Transações distribuídas
- *2-Phase Commit*
- Falhas e Deadlocks
- *3-Phase Commit*

Estado Global Distribuído

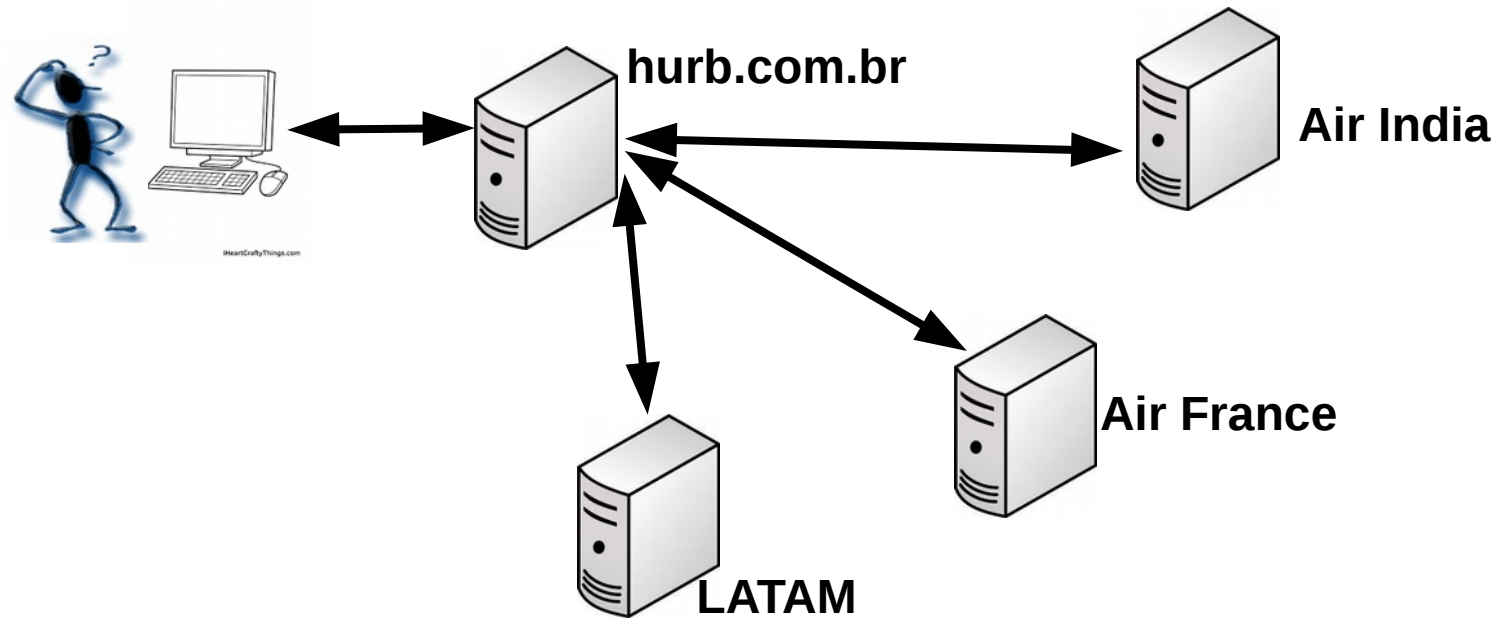
- *Two-Phase Locking*: mecanismo de controle de concorrência
  - assume que estado global está centralizado
- Considere estado global distribuído



- Estado do sistema é a *união* dos estados locais
- Transações leitura/escrita em diferentes locais
- 2PL não serve; não temos memória compartilhada entre locais

Exemplo com Transporte Aéreo

- Passagem para Delhi
  - melhor preço: Rio – São Paulo – Paris – Delhi
  - trechos com LATAM, Air France, Air India
- Sistemas transacionais distintos
  - além do sistema da agência de viagens
- Como garantir a compra dos três trechos?



Transações Distribuídas

- **Ideia**: oferecer propriedades ACID no sistema transacional distribuído
- **Problema**: transação atualiza estado em diferentes locais, de forma condicionada
- **A**tomicidade: tudo ou nada para a transação distribuída!

- Como garantir ACID?

Protocolo para coordenar a execução da transação!

2-Phase Commit

- 2PC: protocolo para commit atômico distribuído
  - não confundir com 2PL (*nada a ver*)
- Processo que inicia transação age como coordenador
  - outros são participantes (outros = processos que possuem valores lidos/escritos pela transação)
- Duas fases
  - Preparar e responder: processos localmente decidem se podem efetuar a transação
  - Executar (*commit*): processos mudam seu estado local

Preparar e Responder

- Coordenador envia diferentes partes da transação (subtransação) para diferentes processos
- Cada processo se prepara para executar a subtransação
  - adquire todos os locks necessários (2PL aqui)
- Cada processo responde ao coordenador
  - Sim: tudo certo para execução (segurando os locks)
  - Não: algo está errado, abortar a transação (liberando os locks)

Executar

- Coordenador recebe respostas de todos processos
- Se todas positivas, envia mensagem *commit*
  - caso contrário, envia mensagem abort
- Ao receber Commit: processo efetua transação, libera locks, responde com OK
- Ao receber Abort: processo aborta transação, libera locks, responde com OK

Exemplo Bancário

- Transferência bancária entre bancos distintos (ex. DOC)
- Transferência executada por processo em L1, conta de origem em L2, conta de destino em L3
  - L1 age como coordenador, envia subtransação retirada para L2, envia subtransação depósito para L3

Em L2

```
retirada(c, v) {
  acquire(c)
  s = get_saldo(c)
  se (s >= v)
    s = s-v
    U = ”put_saldo(c,s)”
    votar commit
    se (commit=OK)...
  caso contrario
    votar abort
    release(c)
}
```

Em L3

```
deposito(c, v) {
  acquire(c)
  s = get_saldo(c)
  s = s + v
  U = “put_saldo(c, s)”
  votar commit
  se (commit=OK)...
}
```

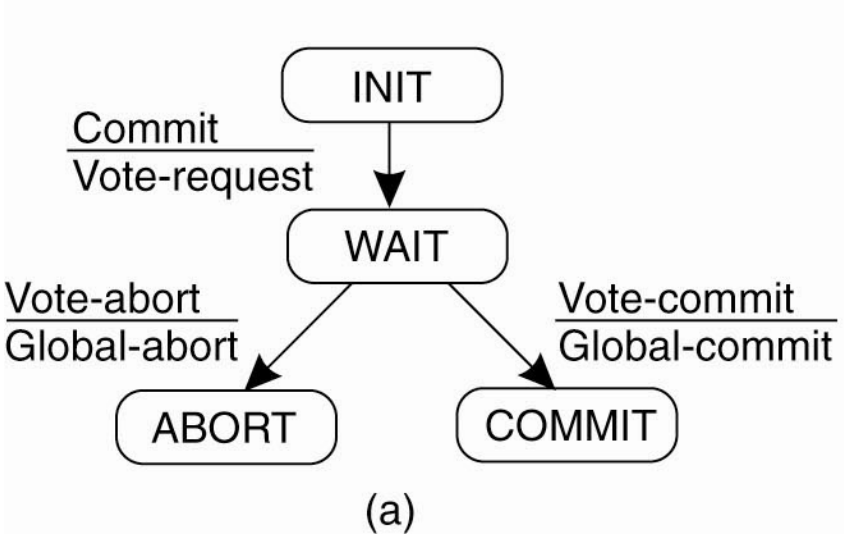
Exemplo Bancário

- L2 responde a L1 dizendo se pode ou não efetuar a retirada (votar commit ou abort)
- L3 responde a L1 dizendo que pode efetuar o depósito (votar commit)
- L1 envia commit para L2 e L3, caso L2 vote commit
  - caso contrário, L1 envia abort
- L2 e L3 usam 2PL para transação local, aguardam resposta do coordenador (commit ou abort)
  - liberar locks ao final

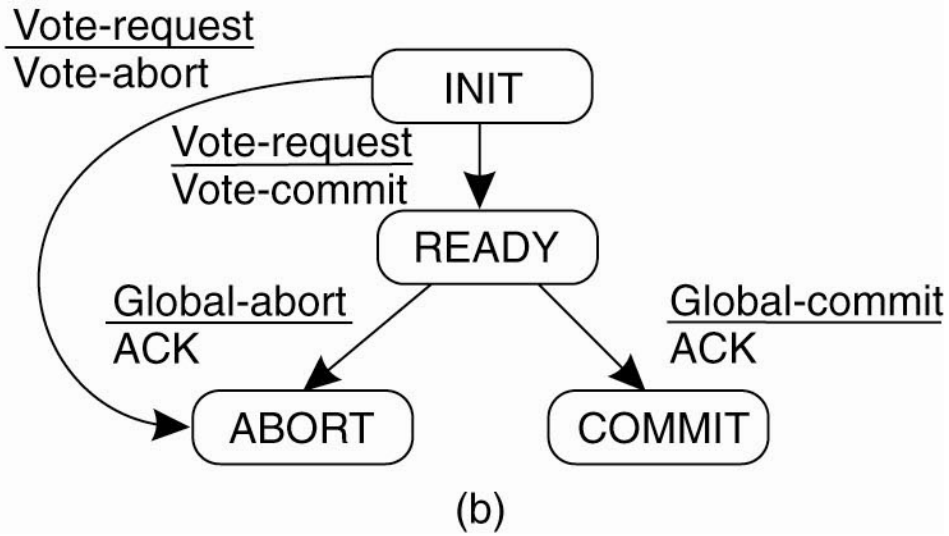
- **Resolvido!**
- Mas como lidar com eventuais falhas?

Falhas e 2PC

- Usar temporizadores (*timers*) nas esperas
  - permite sair do estado bloqueado
- Usar armazenamento permanente (*log*)
  - permite voltar ao último estado local
- Diagrama de transição de estados do 2PC
  - cada transição: mensagem(evento)/resposta(ação)



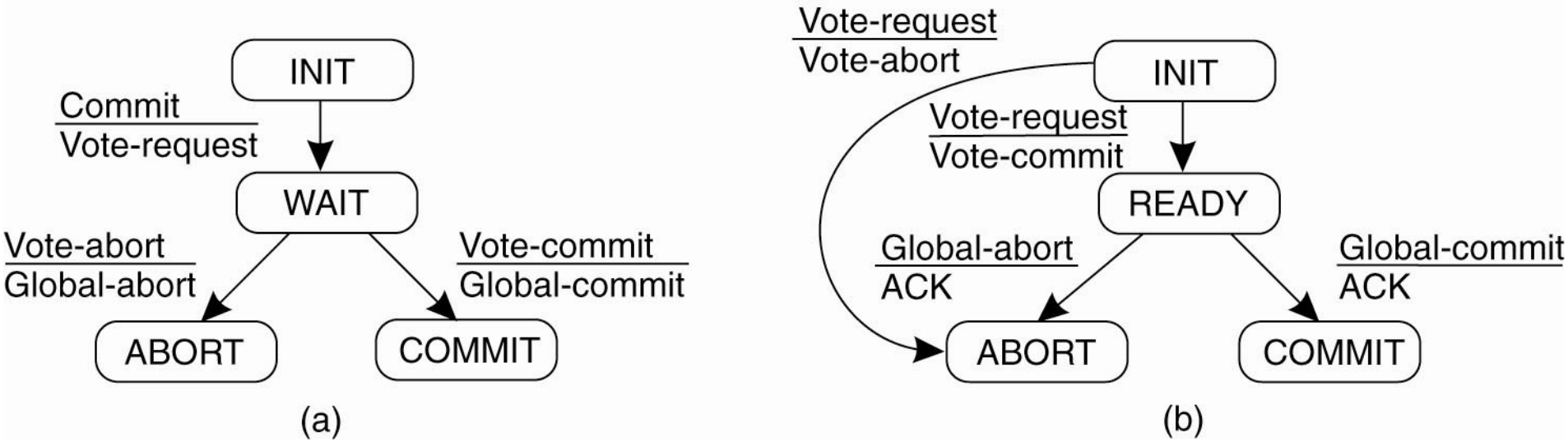
coordenador



participante

Falhas e 2PC

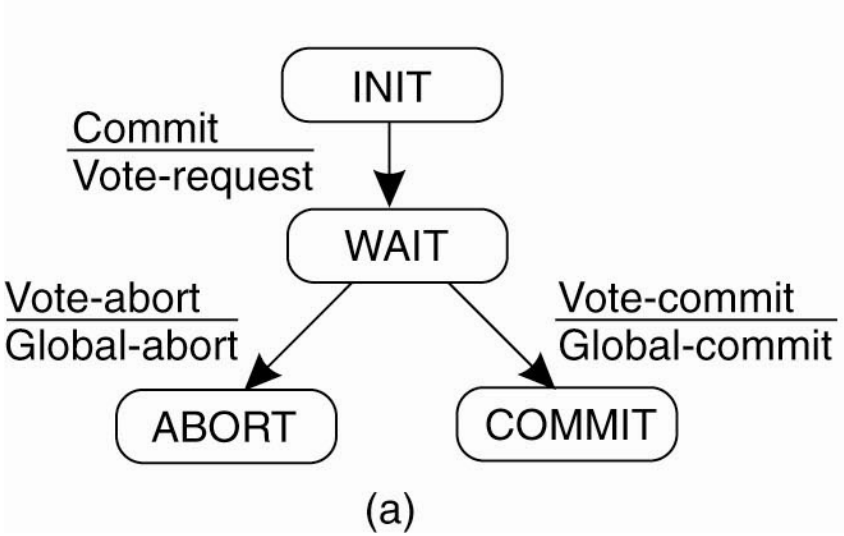
- O que ocorre se processo P falha em INIT?
  - coordenador não recebe resposta, envia abort
- O que ocorre se processo P falha em READY?
  - ao recuperar, P descobre se transação foi abort ou commit, e faz o mesmo
- O que ocorre se C falha em INIT?
  - nada, processos P ainda não receberam a transação



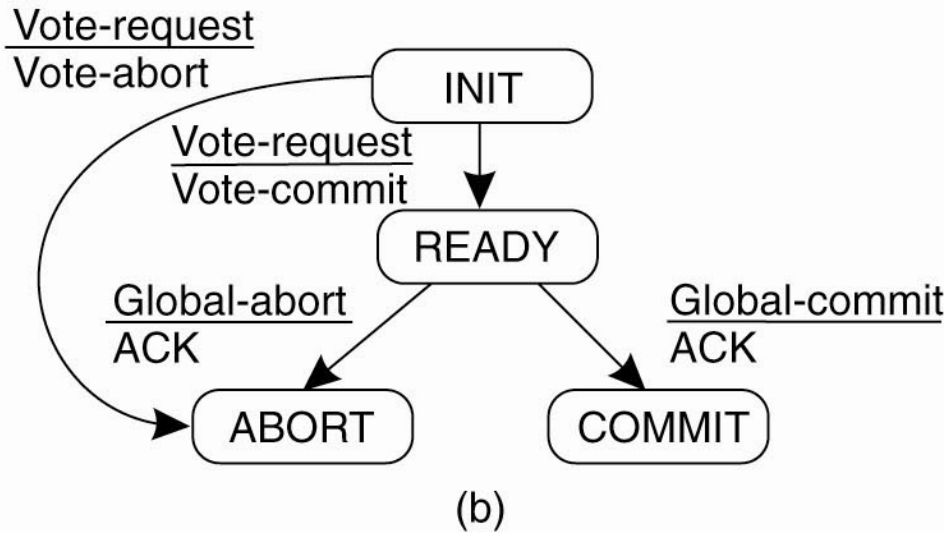
coordenador **participante**

Falhas e 2PC

- O que ocorre se C falha em WAIT?
  - alguns processos podem ter votado COMMIT e outros ABORT
  - processo em READY não pode decidir por COMMIT ou ABORT (não sabe o voto dos outros)
- Processos dependem de C para tomar uma decisão
  - aguardam até C recuperar da falha
- 2PC é bloqueante em C



coordenador



participante

Deadlocks

- Mesmo na ausência de falhas, *deadlocks* é um problema que precisa ser tratado
- Execução distribuída das (sub)transações pode causar uma dependência cíclica nos locks
  - T1=transf(c1,c2,v1), T2=transf(c2,c1,v2)
  - “retirada” de T1 pega lock de c1 em L1
  - “retirada” de T2 pega lock de c2 em L2
  - “depósito” de T1 espera liberar lock de c2 em L2
  - “depósito” de T2 espera liberar lock de c1 em L1
- Processos não podem votar (aguardam locks)
  - coordenadores aguardam indefinidamente

Deadlocks!

Resolvendo *Deadlocks*

- Processos utilizam temporizadores ao esperar por locks
  - votam abort quando temporizador dispara
- Coordenador utilizar temporizador no estado WAIT
  - envia global abort quando temporizador dispara
- Coordenador pode tentar executar transação novamente
  - após global abort, locks dos processos são liberados, nova tentativa pode encontrar locks livres

***Livelocks* à vista!**

- Coordenador pode tentar repetidas vezes, sem sucesso
  - transação pode de fato ser impossível
- Solução: tentar número máximo de vezes
  - aborta transações, mas evita *livelocks*

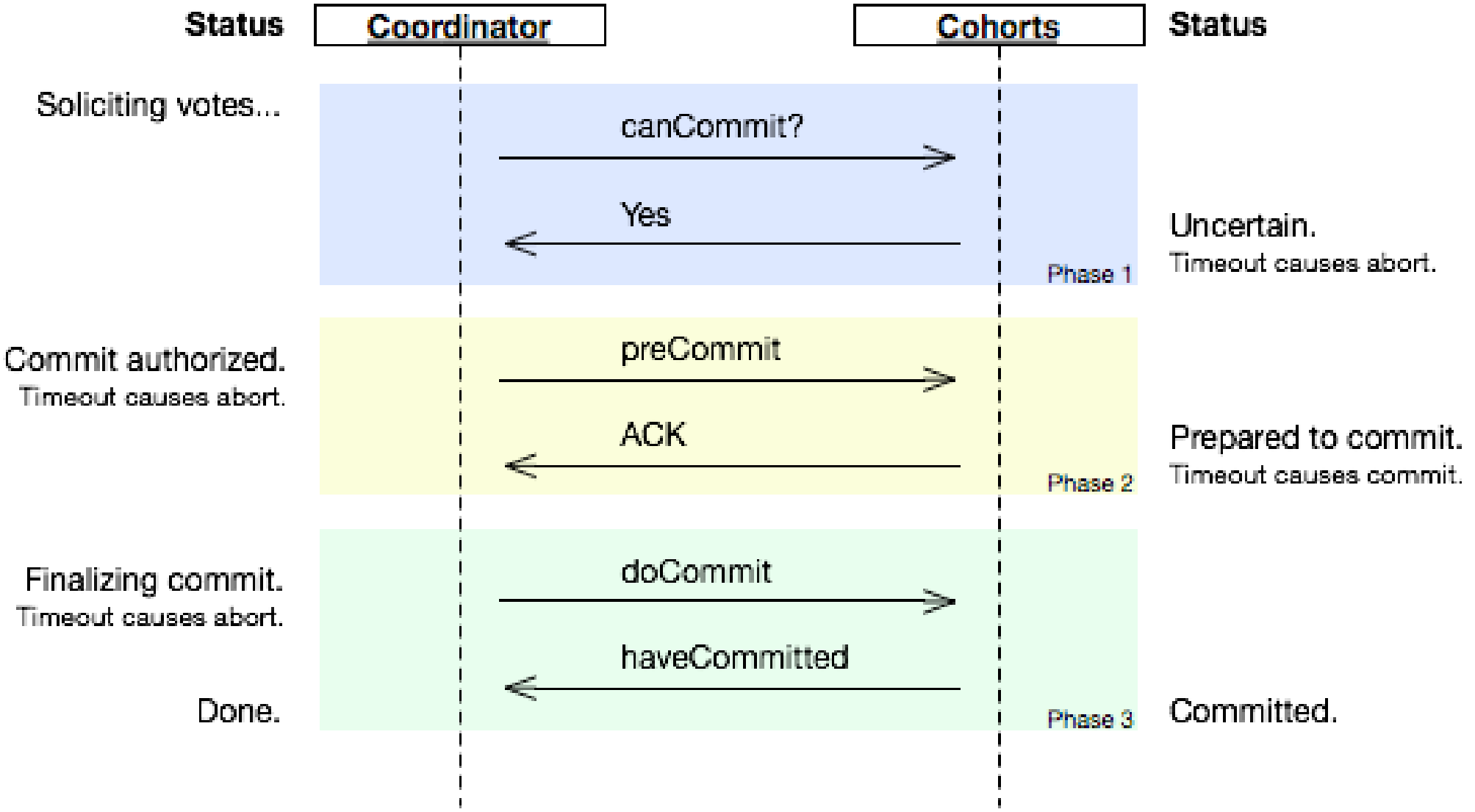
Three-Phase Commit

- Principal desvantagem do 2PC?

Forte dependência no coordenador

- *Three-Phase Commit* (3PC)
  - muito similar ao 2PC
  - fase extra permite commit, sem que coordenador confirme commit global
    - não depende do coordenador
  - uso explícito de temporizadores nos processos para decisões de abort e commit

Three-Phase Commit



Vantagens e Desvantagens

- Não é bloqueante no coordenador
  - permite concluir transação (commit ou abort)
- Temporizadores resolvem naturalmente possíveis deadlocks
- Permite recuperar alguns tipos de falhas que P2C não permite
- Demandam três RTT (*Round-Trip Time*) entre coordenador e participantes
  - seis mensagens por participante