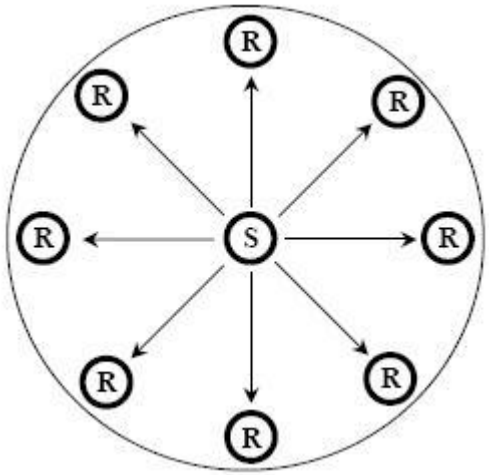
Comunicação em Grupo

(Broadcast - um emissor envia a mensagem para todos os nós do sistema)

Multicast – um emissor, um grupo de processos como receptores



Exemplos de aplicação:

- ferramentas de trabalho cooperativo
- difusão de som/imagem para um grupo de utilizadores registados

Estas aplicações requerem suporte para multicast e também ter a informação sobre "quem" pertence ao grupo (*group membership*).

Nos exemplos anteriores, a noção de grupo de processos é usada explicitamente como suporte à funcionalidade da aplicação.

Diz-se que o grupo é visível

O grupo pode ser usado para implementar requisitos não funcionais.

Exemplo: Um grupo ser usado para nos referirmos colectivamente a um conjunto de réplicas do mesmo componente que executam a mesma acção. Isto é, para técnicas de replicação que permitam assegurar a continuidade do serviço em caso de falha de algum dos componentes

-A aplicação acede ao componente como se fosse um só e não um conjunto de réplicas

Neste caso o grupo diz-se invisível

Grupos e “Views”

Os grupos podem ainda classificar-se em:

Grupos abertos:

- Qualquer processo no sistema pode enviar mensagens para o grupo

- Usados em casos de replicação de serviços

Grupos Fechados:

- Só os membros do grupo podem enviar mensagens para o grupo
- Processos fora do grupo não podem enviar mensagens ao grupo como um todo. (Apenas a membros individualmente)

- Usados em processamento paralelo

Grupos e “Views”

Uma plataforma de grupos possui dois tipos de serviços:

- um serviço de membros do grupo (Group Membership)

- um serviço de comunicação em grupo (Group Communication)

Serviço de membros de grupo, proporciona,

- capacidade para criar, e se tornar membro de, um grupo

- manutenção de um registo sobre quais os actuais membros do grupo, isto é sobre a actual “group view”

(Dentro de uma “group view” os processos têm um identificador único, pelo qual são ordenados)

Serviço de membros de grupo

Objectivos:

- Fornecer de forma dinâmica a constituição do grupo (group views)

- Fornecer primitivas para um processo se juntar ao grupo / deixar o grupo

Serviço de membros de grupo

Deve garantir:

- Exactidão (accuracy) – a informação fornecida reflecte o cenário físico

- Consistência – a informação fornecida é a mesma a todos os processos

Serviço de Comunicação em Grupo

Serviços que permitem fazer multicast de mensagens para todos os membros do grupo (ou para um subconjunto)

Deve garantir:

- A entrega das mensagens (fiabilidade)
- Que é respeitada a ordem pretendida

Quando um membro entra/sai de um grupo,

- Uma nova vista (view) é enviada a todos os membros do grupo

Protocolo Multicast

(responsável pela entrega de mensagens a todos os membros de um grupo)

Principais componentes:

- Endereçamento (routing)
  - Selecciona o caminho da mensagem desde a origem aos destinos
  - Encontra o caminho que minimiza o número de mensagens trocadas e a latência do multicast
  - Sempre que possível deve ser usado hardware multicast

- Tolerância na omissão
  - Lida com as mensagens que são perdidas ou corrompidas na infra-estrutura física, através da transmissão redundante ou retransmissão
  - As omissões são toleradas usando ackowledgments para detectar erros seguido da retransmissão das mensagens perdidas

Protocolo Multicast

Principais componentes: ...

. Controlo de fluxo

- Minimiza a perda de dados provocada pela falta de espaço do buffer dos receptores, ou dos routers intermédios

. Ordem

. Recuperação de falhas

- Impõe uma ordem predefinida e um critério de fiabilidade em relação às mudanças de "view"

O middleware pode reordenar localmente as mensagens recebidas atrasando a entrega das que já recebeu para garantir as propriedades desejadas.
Para esse efeito mantém uma fila local de mensagens recebidas e ainda não entregues

Multicast Fiável em grupos dinâmicos:

- Uma mensagem enviada para um grupo ou é entregue a todos os membros corretos (que não falham) ou a nenhum

- Membros que falham podem ter recebido a mensagem ou não

- Implementação simples (e errada): o emissor emite para cada um dos elementos do grupo de forma fiável (acks+retransmissões) . ➔ "ack implosion"

Em caso de falha do emissor, é necessário que os elementos do grupo propaguem as mensagens recebidas para os elementos que ainda não as receberam

Multicast Fiável em grupos dinâmicos:

- Uma vista (view) é o conjunto de elementos de um grupo num dado momento.

Nos sistemas de comunicação em grupo fiáveis, a mudança de vista é efetuada através do envio de uma mensagem multicast

Sincronia virtual (virtual synchrony)

Um sistema de multicast fiável implementa sincronia virtual se:

-as mudanças de vista são entregues em todos os processos pela mesma ordem

-o conjunto de mensagens entregues entre a entrega de cada duas vistas consecutivas é idêntico para todos os processos que observam as duas vistas

-Isto é, uma mensagem, não pode ser entregue no contexto de uma vista diferente daquela em que foi enviada.

O problema do consenso

Exemplos:

- Commit de um transação num sistema de bases de dados distribuídas
- Eleger um lidar de um conjunto de processos
- ...

Objectivo

Fazer com que um conjunto de processos concorde num único valor que dependa do valor inicial de cada um dos participantes.

O problema do consenso

Exemplo: Linha de empacotamento de produtos, com várias máquinas de empacotamento.

Quando chega um novo produto e duas ou mais máquinas estão disponíveis, é necessário decidir qual delas vai empacotar o produto.

Uma solução: Escalonador centralizado.

Solução descentralizada: As máquinas terão de chegar a um consenso sobre qual delas vai tratar o próximo produto.

O problema do consenso

Protocolo: - Todas as máquinas têm um número

- Quando chega um produto:

- Se uma máquina está livre, propõe o seu número.

- Se uma máquina está ocupada: - Continua com o seu serviço;
Se recebe alguma proposta de outra máquina, vai apoiá-la

Como várias máquinas podem estar livres ao mesmo tempo, cada processo pode receber várias propostas.

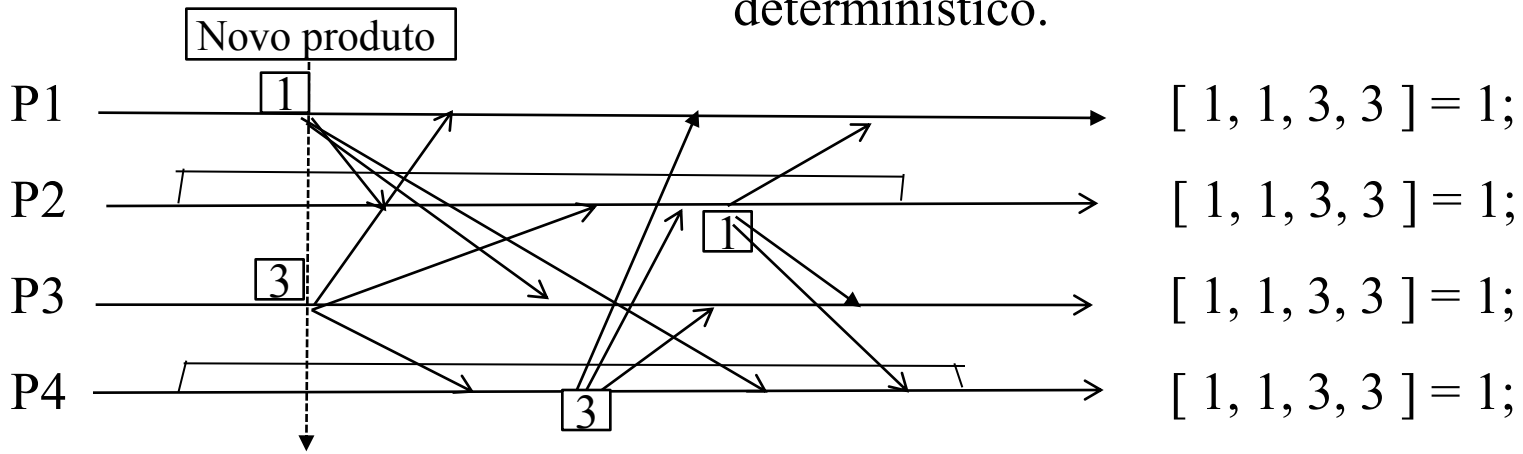
O problema do consenso

Protocolo: ...

O consenso é atingido quando, recolhendo todas as propostas,

- Eliminamos os duplicados;
- Ordenamos as propostas pelo número da máquina;
- Retiramos a primeira proposta da lista.

Como todas as máquinas recebem o mesmo número de propostas, o algoritmo é determinístico.



O problema do consenso

Um protocolo de consenso deve satisfazer as condições:

– **Consistência** – todos os agentes concordam com o mesmo valor e todas as decisões são definitivas.

– **Validade** – o valor acordado tem de ser o input de um dos agentes

– **Terminação –** no final de um número finito de etapas cada agente escolhe um valor (acção)

Consenso tolerante a falhas

O que acontece se uma das máquinas falha?

Se uma máquina falha no passo de disseminar a sua proposta, algumas máquinas podem receber a proposta e outras não !!!!

Um protocolo para resolver o consenso, tem que evitar que um processo decida por um valor enquanto não tiver a garantia de que esse será o único valor que poderá ser decidido pelos outros processos.

Consenso tolerante a falhas

Consideremos a seguinte situação:

O processo com menor identificador é, de forma estática, definido como o coordenador e envia o seu valor inicial a todos os restantes processos.

Esse valor é o resultado do consenso.

(Todos os processos decidem pelo mesmo valor e esse valor é um dos valores iniciais.)

Se não houver falhas, o consenso é trivial.

Em caso de falha de um ou mais processos?

Consenso tolerante a falhas

-Se o processo coordenador falha durante a disseminação do seu valor, alguns processos podem decidir pelo valor recebido enquanto outros continuam à espera !!!

- Eleger outro coordenador?

- E se uns processos recebem o valor de um coordenador, e outros processos recebem o valor do seguinte?

Consenso tolerante a falhas

Comecemos por supor que:

-Quando um processo recebe o valor inicial do coordenador, altera o seu próprio valor para o valor recebido.

-Garante-se assim que, se este processo vier a assumir o papel de coordenador, irá propor o valor recebido do coordenado anterior.

Consenso tolerante a falhas

Algoritmo:

- O coordenador envia o seu valor a todos os outros processos.

- Esses processos não decidem imediatamente o resultado do consenso, mas actualizam o seu valor inicial e enviam um acknowledgement (ACK) de volta para o coordenador.

Quando o coordenador recebe um ACK de todos os processos (em funcionamento) ele sabe que o seu valor é conhecido por todos (diz-se que o valor está **locked**).

Consenso tolerante a falhas

- Após receber todos os ACK, envia para todos os processos a mensagem “decided”.

- Quando um processo recebe a mensagem “decided” decide pelo valor que contém.

Nota: a solução anterior só funciona se existir uma detector de avarias fiável. É necessário, saber se as mensagens chegam a todos os processos que estão em funcionamento.

Num sistema completamente assíncrono, não existe uma solução determinística para o problema.

[Fischer, M. , Lynch N. and Paterson, “Impossibility of Distributed Consensus with One Faulty proccess” Jounal of the ACM, 32, 374:382, 1985.

Atómic Commitment

Problema: garantir que um conjunto de operações em diferentes processos ou são todas executadas (**commit**) ou nenhuma é executada (**abort**)

- É um caso particular de consenso
- Pode ser útil mesmo quando processos não falham, por exemplo se operações individuais podem falhar porque não se verificam determinadas condições

Atómic Commitment

Formalismo :

- Cada participante pode decidir um de dois resultados: commit/ abort

- Cada participente deverá votar/propor um destes 2 resultados

AC1: Todos os participantes que decidem tomam a mesma decisão

AC2: Se algum participante decide **commit**, então todos os participantes votaram nesse sentido

AC3: Se todos os participantes votarem a favor de **commit** e não houver falhas, então todos os participantes decidem **commit**

AC4: cada participante decide no máximo uma vez, i. é, a sua decisão é irreversível

Two-Phase Commit

- O protocolo usa um processo especial, o **coordenador**

- Os processos que deverão executar as acções são os **participantes**

. O coordenador pode não ser participante. Se fôr, deverá executar quer o protocolo do coordenador quer o protocolo dos participantes.

- Protocolo em duas fases:

Fase 1

- O coordenador envia uma mensagem, VOTE-REQUEST, a cada um dosparticipantes, e espera pela resposta do todos os participantes.

- Cada um dos participantes emite um voto VOTE-COMMIT ou VOTE-ABORT

Two-Phase Commit

Fase 2

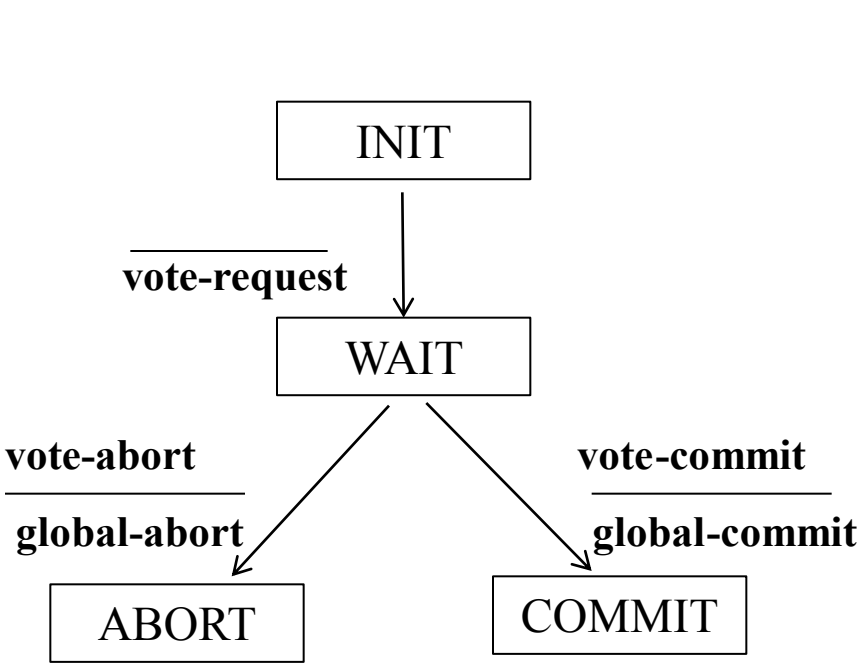
- O coordenador decide COMMIT se tiver recebido VOTE-COMMIT de todos os participantes, senão decide ABORT

-Quando recebem a decisão do coordenador, os participantes atuam de acordo executando ou não as operações, e enviam um ACK.

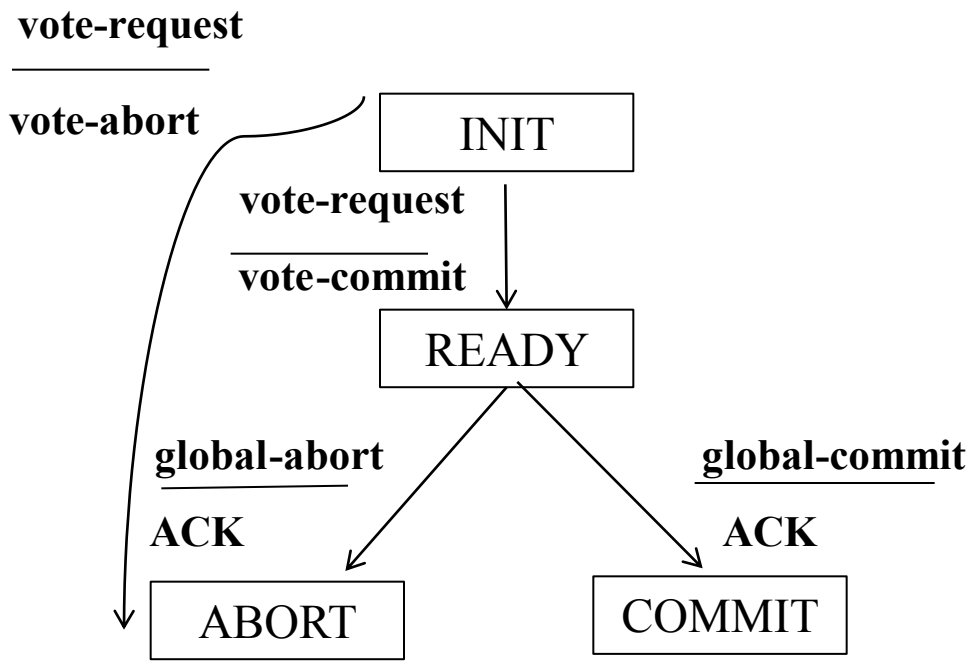
Avarias em processos podem ser detetadas por timeouts

O protocolo deve prever as acções a executar se ocorrerem timeouts

Two-Phase Commit



a) coordenador



b) participante

Two-Phase Commit

Timeout e recuperação

Coordenador – só espera por mensagens no estado WAIT

- Decide ABORT e envia GLOBAL-ABORT

Participante – pode bloquear em INIT

- Decide ABORT e passa ao estado correspondente

Participante – pode bloquear em READY

- Precisa contactar os outros participantes para determinar o resultado
- Poderá ter de bloquear à espera que o coordenador recupere, para tomar conhecimento da decisão

Two-Phase Commit

Timeout e recuperação

O protocolo satisfaz as propriedades AC1 a AC4 em presença de:

- Falhas de comunicação

- Falhas nos nós, desde que não sejam arbitrárias

O principal problema é a possibilidade de os participantes bloquearem quando o coordenador falha.

Solução: ➔ Three-phase commit

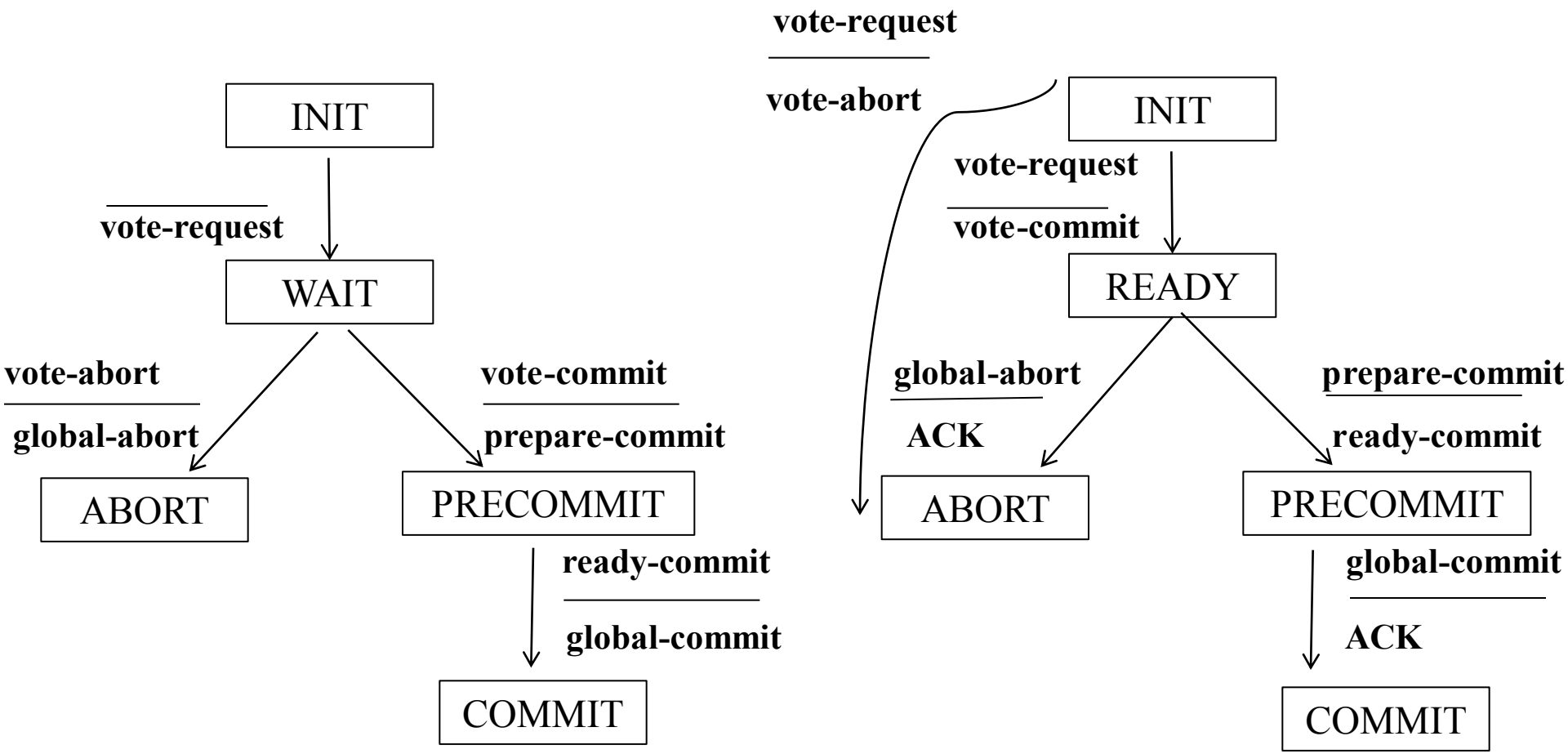
Three-Phase Commit

Evita o bloqueio, introduzinto uma fase intermédia entre as duas fase do two-phase commit, onde o coordenador dá a conhecer a intenção de fazer commit.

O novo estado, garante que não há um estado com transição para o estado COMMIT e para o estado ABORT

Isto é, não há um estado onde não seja possível tomar a decisão e ao mesmo tempo seja possível transitar para o estado de COMMIT

Three -Phase Commit



a) coordenador **b) participante**

Three -Phase Commit

O protocolo garante que não há bloqueio no caso do coordenador falhar, desde que haja uma maioria de participantes que acordem no resultado:

- Se a maioria dos participantes estiver no estado READY pode decidir ABORT
- Se a maioria dos participantes estiver no estado PRECOMMIT pode decidir COMMIT