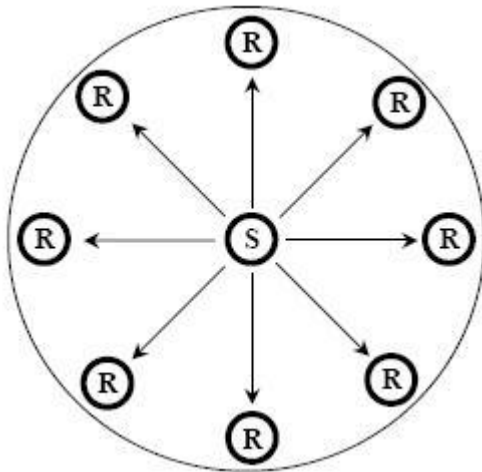


Comunicação em Grupo

(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 acknowledgments 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 problema do consenso

Exemplos:

- Commit de uma transação num sistema de bases de dados distribuídas
- Eleger um líder 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.

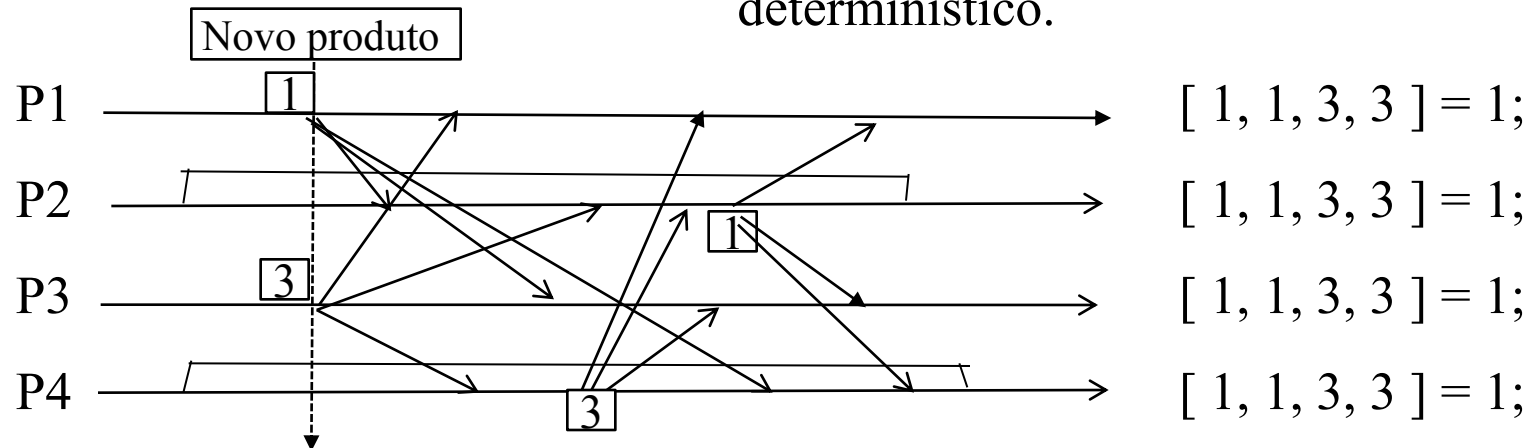
O problema do consenso

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?

-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

Começemos 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 coordenado, irá propor o valor recebido do coordenado anterior.

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 coordenado 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 um 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 process” Journal of the ACM, 32, 374:382, 1985.