THE PROCESS GROUP APPROACH TO RELIABLE DISTRIBUTED COMPUTING

Kenneth P. Birman

“A abordagem de Grupos de Processos para a Computação Distribuída Confiável”

Trabalho realizado por:

João Cardoso	M2383
Vasco Santos	M2059
Ruben Costa	M2384

Introdução

- O artigo descreve 10 anos de investigação sobre o ISIS, um sistema que fornece ferramentas para a construção de software distribuído confiável.

- A tese que suporta o ISIS é a de que o desenvolvimento deste tipo de software pode ser simplificado, utilizando
  - **grupos de processos**
  - **ferramentas de programação em grupo**.

Grupo de processos

- Um *grupo de processos* é um conjunto de processos que podem ser tratados com uma única entidade para determinadas operações como
  - Sincronização;
  - Broadcast ou Multicast.

Grupo de processos

- A todos os processos de um grupo é associado um *process-group identifier*, que indica a que grupo um processo pertence. Este identificador corresponde ao PID do processo que foi o membro inicial do grupo.

- Um processo é sempre um membro de apenas um grupo de processos.

- Quando é criado um novo processo (filho), geralmente este é colocado no mesmo grupo que o processo pai.

- É habitual a programas criar novos grupos de processos, colocando processos-filho relacionados no mesmo grupo.

- Um processo pode mudar o seu próprio grupo, ou o dos seus processos-filho, criando um novo grupo, ou movendo-se a si ou os processos-filho para um grupo já existente.

Tipos de Grupos

- **Grupos Anónimos;**

  (*"Anonymous groups"*)

- **Grupos Explícitos**;

  (*"Explicit groups"*)

Grupos Anónimos

- Estes surgem quando uma aplicação publica os dados sob um determinado "tópico," e os outros processos subscrevem esse tópico.

- Para uma aplicação funcionar de forma automática e confiável, este grupos anónimos devem fornecer certas propriedades:
  - Deve ser possível enviar mensagens ao grupo através de um endereço do grupo
  - Se o remetente e os subscritores permanecerem operacionais, as mensagens só devem ser entregues uma vez. Se o remetente falhar, a mensagem deverá ser entregue ou a todos ou a nenhum dos subscritores.
  - As mensagens devem ser entregues a subscritores numa ordem consistente, que evite dependências, isto é processos que tenham de ficar á espera da entrega das mensagens a outros processos.
  - Deve ser possível a um subscritor, obter um histórico de um grupo.

Grupos Explícitos

- Um grupo é explicito quando os seus membros cooperam directamente: os processos sabem que são membros do mesmo grupo, e usam algoritmos que incorporam uma lista de membros do grupo.

- Os grupos explícitos de processos têm necessidades adicionais, decorrentes do seu uso de informação sobre a que grupo os processos pertencem (membership): quando ocorrem movimentações entre grupos, onde processos mudam de um grupo para outro, isto gera informação que necessita de ser publicada para todos os grupos.

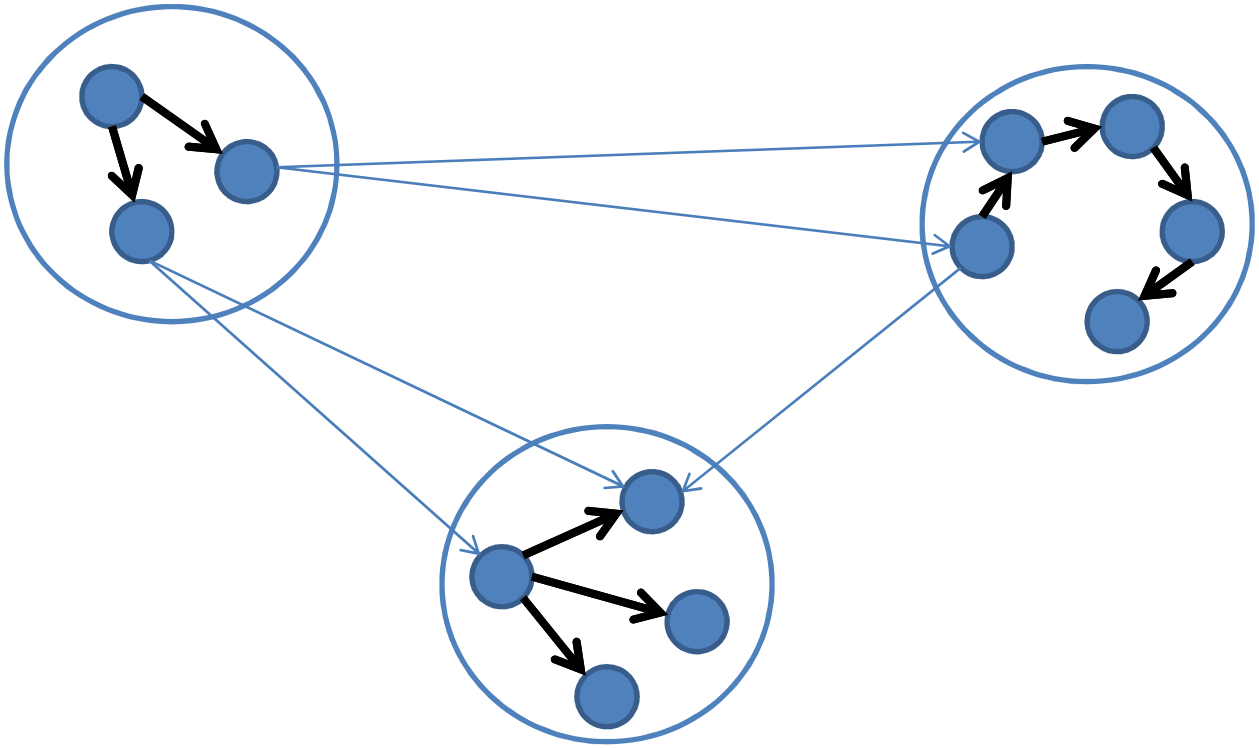
Problemas de Implementação

- Deste modo, surgem diversos problemas que têm de ser tidos em conta, para o desenvolvimento de software que implemente grupos de processos distribuídos:

  - **Suporte para a comunicação dentro do grupo e entre grupos**, endereçamento do grupo, atomicidade da comunicação, e ordenação da distribuição das mensagens.

  - **Sincronização:** Para obter um comportamento global correcto das aplicações de grupo, é necessária a sincronização da ordem de como as acções devem ocorrer, particularmente quando membros de um grupo agem independentemente, alterando dinamicamente informação partilhada.

Transmissão de mensagens



Tecnologias convencionais de transmissão de mensagens entre processos

- ***Datagramas***: Estes serviços descartam automaticamente mensagens corrompidas, mas não garantem que as mensagens sejam entregues; em certas condições as mensagens podem ser perdidas durante a transmissão, duplicadas ou entregues fora de ordem.

- ***Remote procedure call***: Nesta abordagem , a comunicação resulta da invocação de um procedimento que devolve um resultado. O RPC é um serviço relativamente confiável, mas quando uma falha ocorre, o remetente não é capaz de distinguir as possíveis causas: o destino poder ter falhado antes ou depois de ter recebido o pedido, ou a rede pode ter impedido ou atrasado a entrega do pedido ou da resposta.

- ***Data streams***: A comunicação é feita sobre canais que fornecem ferramentas de controlo e de entrega de mensagens de forma sequenciada e confiável. Por causa do pipelining, as streams geralmente superam o RPC quando uma aplicação envia grandes volumes de dados. Contudo, as normas descrevem regras que definem quando uma stream é quebrada utilizando as condições baseadas em timeout ou excesso de retransmissões.

Construindo Grupos sobre as tecnologias convencionais

- ***Endereçamento de Grupos***
- ***Tempo e dependência causal***
- ***Ordenação da entrega das mensagens***
- **Transferência de estados**
- **Tolerância a falhas**

Endereçamento de Grupos

- Um modo de abordar este problema envolve a implementação de um serviço de gestão de membros. Este serviço deve possuir uma lista de identificadores de grupo e as listas de membros que pertencem aos grupos.

- Este tipo de serviço pode ser implementado através de um programa que suporte RPC:
  - para registar um novo grupo, ou um novo membro de um grupo;
  - obter a lista de membros de um grupo;
  - ou ainda enviar mensagens para todos os membros de um grupo.

Tempo e dependência causal

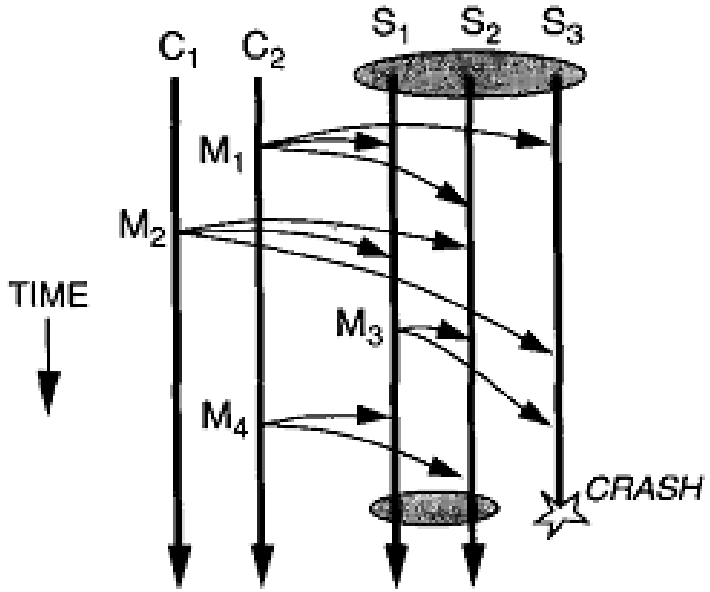
- A frase "atingindo todos os membros, ao mesmo tempo " levanta uma questão fundamental sobre a ordenação da distribuição das mensagens. O que leva á necessidade da implementação de um modelo temporal.

- Em algoritmos distribuídos, é usual ocorrerem dependências entre eventos. Por exemplo quando um processo, inicialmente atribui a uma variável x = 3 e de seguida atribui y = x; o evento responsável pela ultima operação vai depender do seu predecessor.

- Quando é dito que todos os membros de um grupo irão receber uma mensagem ao mesmo tempo, quer dizer que os eventos da distribuição da mensagem são concorrentes e totalmente ordenados de acordo com as acções dos membros do grupo.

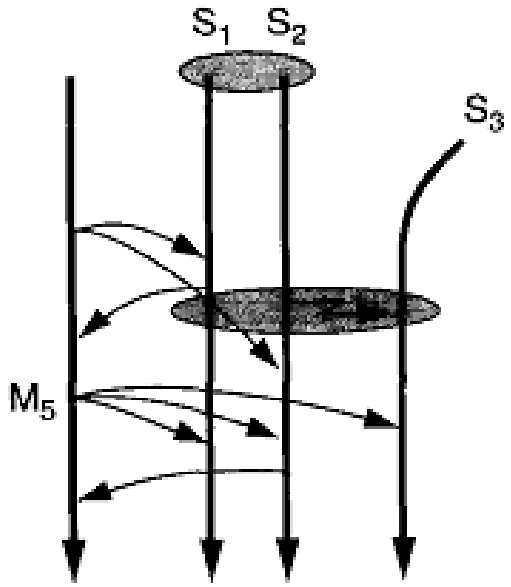
Ordenação da entrega das mensagens

- Considerando a figura , na qual duas mensagens M1 M2, M3 e M4 são enviadas para um grupo que consiste nos processos S1, S2, e S3.
- As mensagens M1 e M2 são enviadas concorrentemente e são recebidas em diferentes ordens por S2 e S3;
- M3 é enviada por S1 sem a M2 ter sido completamente enviada para todos os processos do grupo;
- M4 é enviada antes de ocorrer uma alteração dos membros de um grupo.



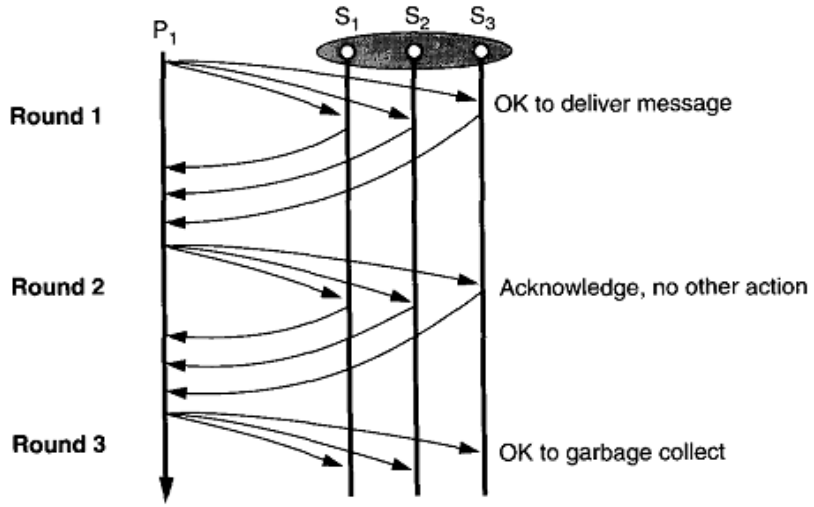
Transferência de estados

- Neste caso, é necessário transferir o estado do serviço para o processo S3, onde este pode representar um programa que reiniciou após uma falha, ou um servidor que seja adicionado para a redistribuição da carga.
- O estado de um servidor será uma estrutura de dados que irá conter a informação gerida pelo serviço, modificada pelas mensagens recebidas antes da adição do novo membro ao grupo.
- Neste caso ocorrerá uma inconsistência de dados pois S3 não possui toda a informação das mensagens



Tolerância a falhas

- Este protocolo utiliza três rondas de RPCs como esta ilustrado na figura.
- Durante a 1ª ronda, o remetente envia a mensagem para os destinatários, que acusam a sua recepção. Aqui, os destinatários vão guardar uma cópia da mensagem caso seja necessário completar o protocolo, substituindo o remetente.
- Na 2ª ronda, se não ocorreram falhas, o remetente informa os destinatários que vai entrar na 3ª ronda. Estes acusam a recepção desta mensagem e tomam nota.
- Durante a 3ª ronda, cada destinatário descarta toda a informação sobre a mensagem - apagando a copia efectuada e outra informação mantida.



Sincronização Virtual

- Objectivo:
  - Integração de mecanismos de programação em grupo num único ambiente.
  - Encorajar os programadores a assumir um estilo puramente sincronizado de execução distribuída.
  - É constituida por duas partes:
    - *Close Synchrony* (Sincronização Estrita)
    - *Asynchronous Pipelining*

Close Synchrony

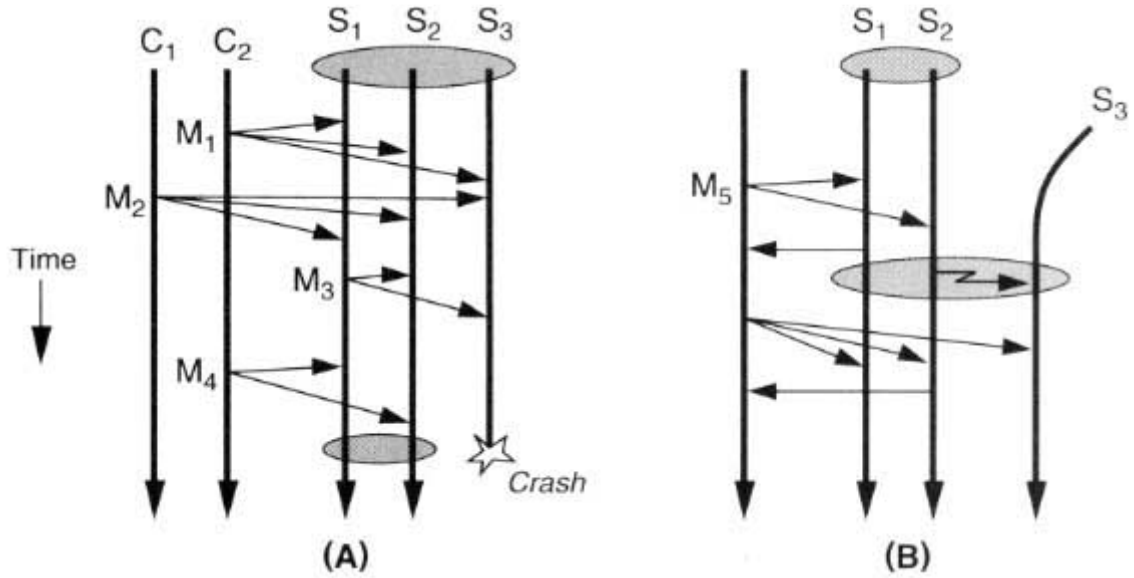
- Propriedades:
  - Execução de um processo consiste numa sequência de eventos, os quais podem ser computações internas, transmissões de mensagens, entregas de mensagens ou alterações nos membros dos grupos que o cria ou se junta.
  - Uma execução glogal do sistema consiste num conjunto de execuções de processos. Ao nivel global, um pode falar sobre mensagens enviadas como multicasts a grupos de processos.

Close Synchrony

- Propriedades:
  - Quaisquer dois processos que recebem os mesmos multicasts ou observam as mesmas alterações nos membros de um grupo, vêem os eventos locais correspondentes na mesma ordem relativa.
  - Um multicast para um grupo de processos é entregue a todos os membros. Os eventos enviados e entregues consideram-se que ocorreram com um único e instantâneo evento.

Close Synchrony

- É uma garantia poderosa. De facto, elimina todos os problemas identificados anteriormente:



Close Synchrony

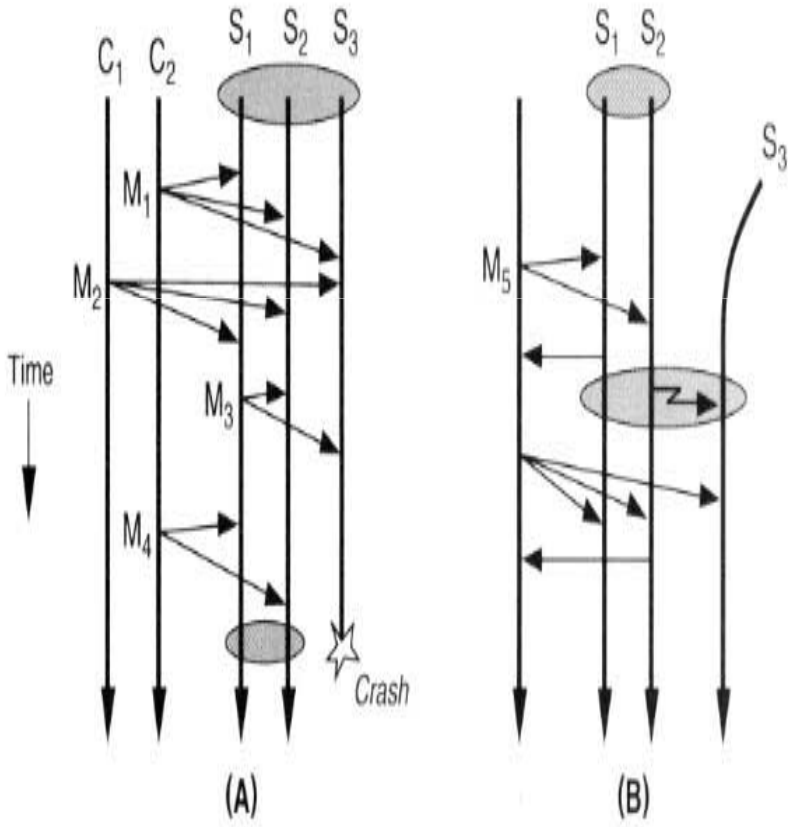
- Garantia de fiabilidade em comunicações fracas:
  - Um subsistema de comunicação estritamente sincronizado aparece ao programador como completamente fiavel.
- Expansão do endereço de grupo
  - Numa execução estritamente sincronizada, os membros de um grupo de processos é fixado no instante lógico quando é entregue um multicast.

Close Synchrony

- Ordenação de entregas para mensagens concorrentes
  - Numa execução estritamente sincronizada, os multicasts concorrentes enviados são eventos distintos.

Close Synchrony

– Ordenação de entregas para sequências de mensagens relacionadas
  - Na figura, parte (A), processo S1 envia a mensagem M3 depois de receber M2, por isso, M3 pode estar dependente casualmente de M2.



Close Synchrony

- Transferência de estado
  - Transferência de estado ocorre num intante bem definido no tempo do modelo. Se um membro do grupo faz um chekpoint do estado do grupo no momento em que é adicionado um novo membro, ou envia algo baseado no estado do novo membro, o estado será bem definido e completo.

Close Synchrony

- Atomicidade de avarias
  - O modelo de sincronização estrita trata um multicast como um único evento lógico e relata as avarias através das alterações nos membros do grupo que são ordenadas em respeito ao multicast. O comportamento de “tudo ou nada” de um multicast atómico é, por isso, implícito pelo modelo.

Close Synchrony

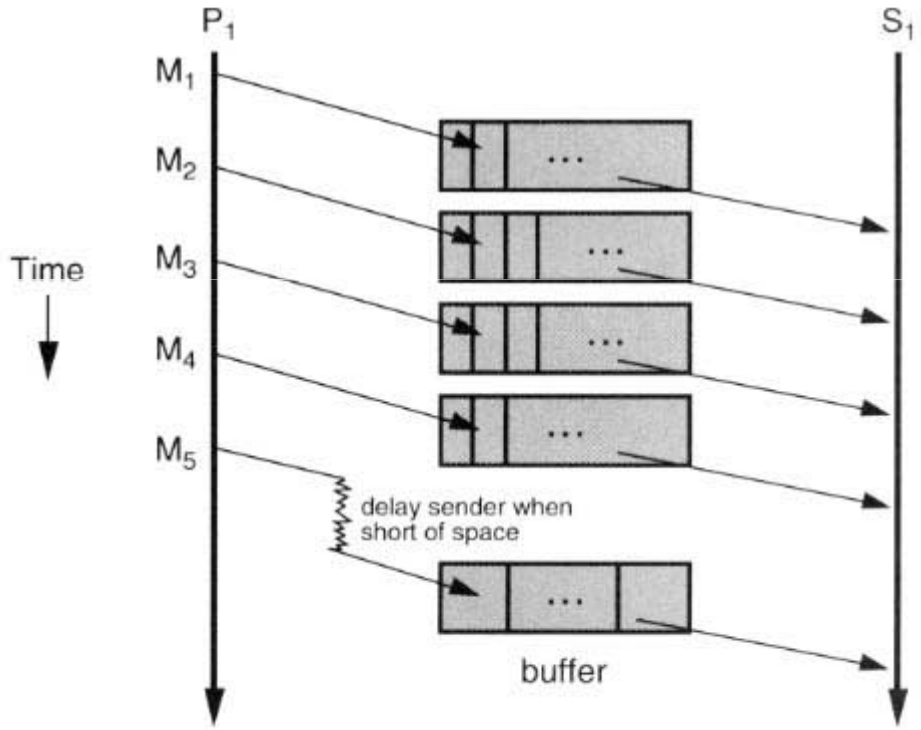
- Limitações:
  - Não pode ser aplicado na presença de avarias;
  - Bastante pesado.

Asynchronous Pipelining

- É possivel obter alto desempenho em sistemas distribuidos através de interacções assincronas.
- O emissor de uma mensagem pode continuar a sua execução sem esperar pela entrega.
- O emissor é bloqueado apenas quando a taxa de produção de dados excede a taxa de consumo, ou quando o emissor precisa de esperar por uma resposta ou outro tipo de input.

Asynchronous Pipelining

- Vantagem:
  - A latência (atraso) entre o emissor e o destinatário não afecta a taxa de transmissão – o sistema trabalha de forma igual a um pipeline, permitindo a ambos manterem-se continuamente activos.



Asynchronous Pipelining

- A execução em sincronização estrita opõe tal *pipelining*, pois atrasa a execução do emissor até a mensagem ser entregue.

Sincronização Virtual

- Em resumo, um sistema com sincronização virtual permite execuções assincronas para as quais existe alguma execução sincronizada estritamente, indistinguivel da assincrona.
- Isto siginifica que para cada aplicação os eventos precisam de ser sincronizados apenas ao nivel onde a aplicação é **sensivel à ordenação de eventos**.
- Ou seja, é possivel entregar mensagen em ordens diferentes em processos diferentes sem que a aplicação dê conta.

Sensibilidade na ordenação em Sistemas Distribuídos

- "Quando a sincronização pode ser relaxada num sistema distribuido sincronizado virtualmente?"
- Duas formas de ordenar bastantes úteis:
  - ABCAST – Ordenação Total
  - CBCAST – Ordenação Causal

ABCAST vs CBCAST
(ABCAST)

- Vantagens:
  - Garante que as réplicas se mantenham idênticas ao longo da execução do sistema;
  - Bastante forte;
  - Fácil de trabalhar.
- Limitações:
  - Díficil de implementar;
  - Bastante pesada;
  - Adiciona latência.

ABCAST vs CBCAST (CBCAST)

- Vantagens:
  - Não é sujeita à latência;
  - Melhor desempenho;
  - Pode ser usado quando qualisquer multicasts conflituosos são ordenados de forma única ao longo de um única corrente causal.
- Limitações:
  - Mais fraco que o ABCAST (permite que mensagens enviadas concurrentemente sejam entregues a diferentes destinos em ordens diferentes).

Beneficios da Sincronização Virtual

- Em resumo, os beneficios do modelo de sincronização virtual são:
  - Permite que seja desenvolvido código fonte assumindo um modelo simplificado de execução sincronizada estritamente;
  - Suporta uma noção significativa de estado de grupo e transferência de estado, ambos quando grupos gerem dados replicados e quando uma computação é particionada dinâmicamente entre membros do grupo;
  - Comunicação *pipelined* assincona;

Beneficios da Sincronização Virtual

- Tratamento da comunicação, das alterações nos membros de um grupo de processos e avarias através de um únicp modelo de execução orientado a eventos;
- Gestão de avarias através de uma lista de membros de um sistema consistemente apresentada e integrada com o subsistema de comunicação.

Limitações da Sincronização Virtual

- Disponibilidade reduzida durante as avarias de partição LAN: apenas permite progresso numa única partição e, por isso, tolera no máximo (n/2)-1 avarias simultâneas, sendo n o número de sitios concorrentes em funcionamento;
- Tem o risco de classificar incorrectamente um sitio ou processo em funcionamento como ter tido uma falha (falsos negativos).

Ferramenta ISIS

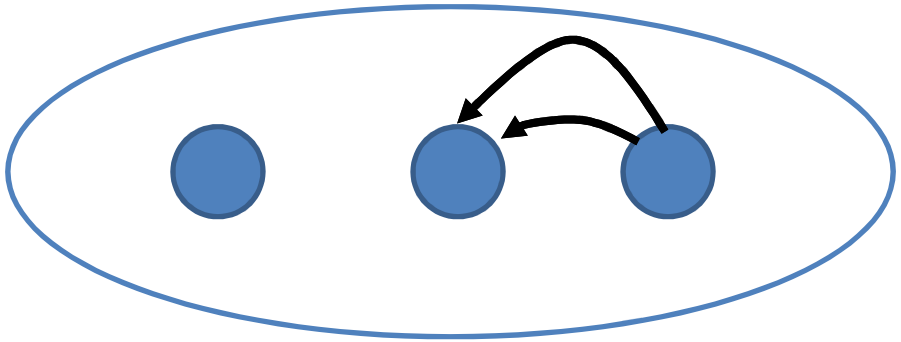
- A ferramenta ISIS é uma colecção de mecanismos de alto nível para a formação e gestão de grupos de processos e a implementação de software baseado em grupos

Ferramenta ISIS

- A eficiência de um sistema distribuído é limitada pela informação disponível para os protocolos de comunicação.
- O que levou a que uma troca entre simplicidade da interface e disponibilidade de informações precisas, tivesse que ser feita.
- Devido a isto a ferramenta ISIS apresenta quatro estilos de grupos de processos que diferem no modo como os processos interagem com o grupo.

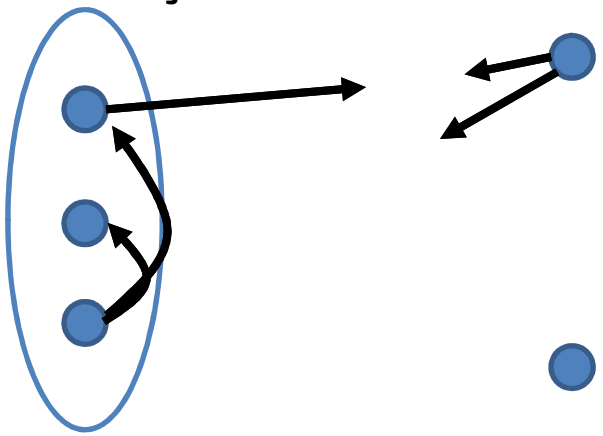
Estilos de grupos

- Peer group
  - Estes surgem quando um conjunto de processos cooperam de perto, por exemplo na replicação de dados.
  - Frequentemente utilizada como input para algoritmos utilizados na manipulação de pedidos.



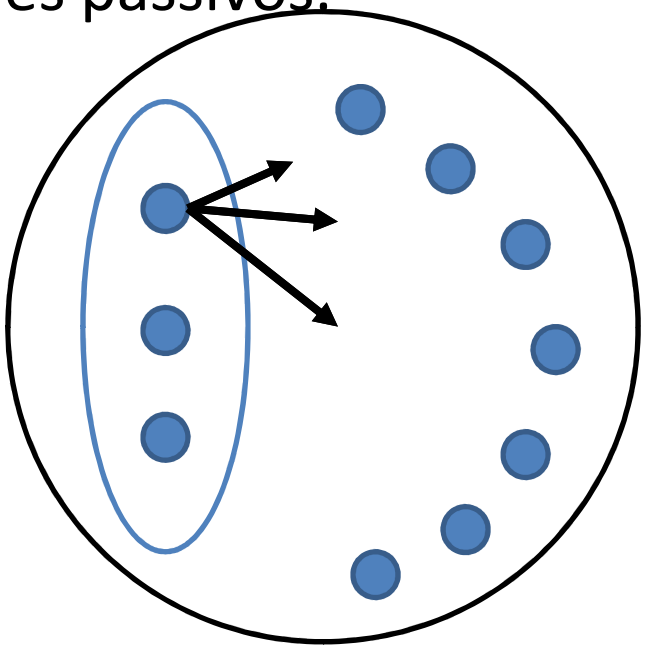
Estilos de grupos

- Client-server group
  - Qualquer processo pode comunicar com qualquer grupo, dado o nome do grupo e a permissão apropriada.
  - Caso um processo, não membro do grupo, provoca multicast repetidamente ao grupo, é obtido um melhor despenho através do registo do processo como um cliente ao grupo. Isto permite que o sistema optimize o protocolo de endereçamento.



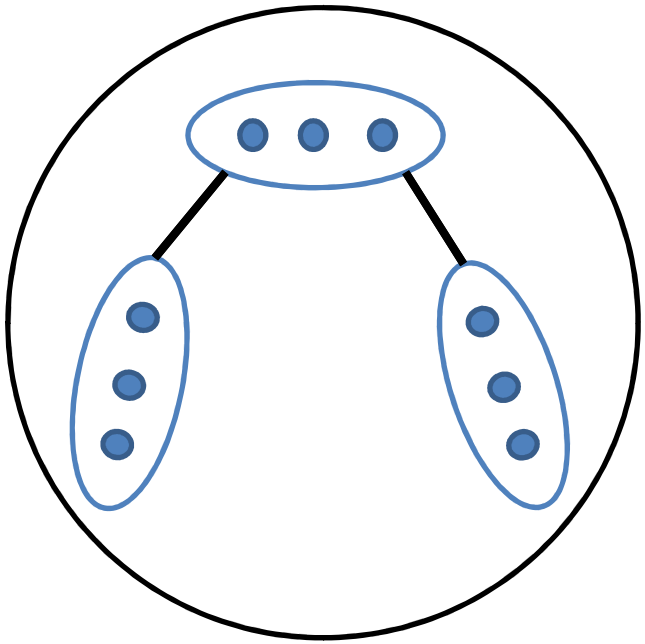
Estilos de grupos

- Diffusion group
  - Um diffusion group é um Client-server group no qual o cliente regista-se a ele próprio, mas em que os membros do grupo enviam mensagens para todo o conjunto de clientes e os clientes são dissipadores passivos.



Estilos de grupos

- Hierarchical group
  - Um hierarchical group é um grupo constituído por vários subgrupos.
  - Aplicações que utilizam este tipo de grupo, inicialmente contacta com a raiz e são depois redireccionados para um dos subgrupos.



Interface da ferramenta

- Muitos dos utilizadores prefere utilizar soluções menos eficientes do que arriscar erros.
- Por essa razão a ferramenta inclui implementação assíncrona da programação distribuída mais importante. Isto inclui:
  - Ferramenta de sincronização
  - Ferramenta para a gestão de replicação de dados
  - Ferramenta para tolerância a falhas primary-backup, que proporciona load-balancing através da utilização de diferentes membros do grupo como primários para diferentes pedidos.

Onde a ISIS é usada

Computação finaceira

- Uma grande parte dos utilizadores desta ferramenta está preocupada com sistemas de processamento financeiro. Em que é usado um modelo de cliente – servidor, onde o servidor filtra e analisa as streams de dados.

Onde a ISIS é usada

Computação finaceira

- Tolerância a falhas neste sistema é referente a dois pontos:
  - Sistemas financeiros precisam reiniciar rapidamente componentes que tenham falhado, e reorganiza-los para que o serviço não sofra interrupções nem de hardware ou software.
  - Existem funções especificas que necessitam de tolerância a falhas ao nível dos ficheiros ou da base de dados, de modo a garantir que depois de reiniciado o sistema de gestão da informação, seja capaz de recuperar os dados com o mínimo de custos.

Onde a ISIS é usada

Computação finaceira

- A abordagem normalmente utilizada é a de representar os serviços chave usando um grupo de processos, para que caso um falhe outro possa responder por ele.
- A redundância também pode ser explorada para melhorar o tempo de resposta.

Onde a ISIS é usada

Computação finaceira

- Uma firma pode trabalhar com dezenas ou até centenas de modelos financeiros, para prever os comportamentos do mercado. No entanto apenas um pequeno subconjunto destes serviços vai ser preciso num dado tempo.
- Sistemas como estes que variam dinamicamente o conjunto de serviços, geralmente consistem no chamado processor pool, em que os serviços podem ser inicializados quando necessário.

Onde a ISIS é usada

Bases de dados

- Os clientes das bases de dados acedem a informação através de uma camada de software que multicasts actualizações (usando ABCAST) para um conjunto de servidores, enquanto faz queries directamente ao servidor menos carregado
- Os servidores são supervisionados por um grupo de processos que informa os clientes das mudanças de carga no servers pool, e supervisiona o reinicio do servidor que foi vitima de uma falha, a partir de um checkpoint e as subsequentes actualizações

Onde a ISIS é usada

Outros ISIS aplicações

- Esta ferramenta tem sido aplicada em aplicações militares, sistemas médicos, gráficos e realidade virtual, sismologia, telecomunicações, entre outros.

Conclusão

- A programação orientada para grupos de processos, têm como objectivo trazer uma onda de avanços em computação distribuída e fiável, e de aplicações que operam em plataformas distribuídas.
- Ferramentas como a ISIS, vêm ajudar utilizadores inexperientes, a implementar grupos de processos de modo a que o sistema resultante seja muito simples e fácil de usar