Comunicação de Grupo: Disfusão Confiável e Atômica

Prof. Lau Cheuk Lung
E-mail: lau.lung@inf.ufsc.br
Departamento de Informática e Estatística
Universidade Federal de Santa Catarina

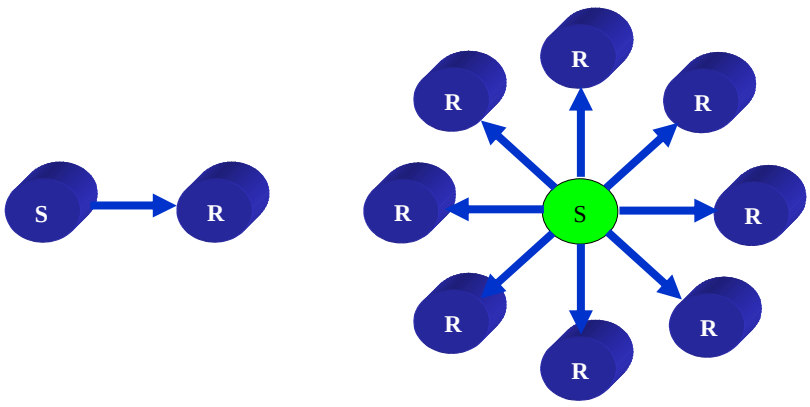
Introdução

- Outros paradigmas de interação : comunicação de grupo vários receptores
- Grupo: coleção de processos (objetos) que são associados a nível de aplicação por alguma propriedade ou interesse comum.
- Um sistema distribuído pode ter vários grupos sobrepostos ou não, seguindo as funcionalidades do sistema e suportados por um serviço de comunicação e gestão de grupo.
- Ponto chave :
  - Uma mensagem quando enviada a um grupo, todos os membros ativos devem recebê-la.
- Tolerância a Faltas, replicação de serviços, desempenho, aplicações de groupware.

Transparência de Grupo

- Em nível de middleware ou suporte de programação distribuída, o conceito de grupo permite um processo ou objeto tratar coleções de processos ou objetos como uma abstração única. O emissor não sabe o custo, quantos são e onde estão os membros.
  - One – to – many communication
- Outra maneira é o processo enviar a lista de receptores (endereços IP); um ponteiro para a lista no Send. A transparência de grupo é perdida.

Difusão de mensagem



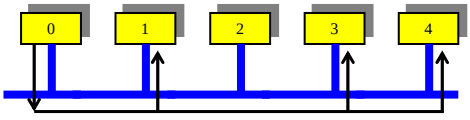
Comunicações ponto-a-ponto e um-para-muitos

Endereçamento

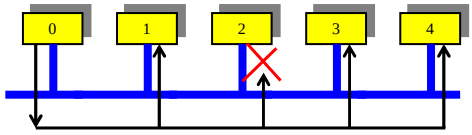
Grupos precisam ser distinguidos no sistema por endereços que dependem de serviços disponíveis na rede:

- redes com serviço de multicast (difusão seletiva) envio de um pacote só:
  - a definição de um endereço onde múltiplas máquinas identificariam seus processos como membros de um grupo
  - diferentes endereços IP multicast a diferentes grupos
- broadcast endereços que enviam pacotes a todas as máquinas
  - pode ser usado mas com eficiência menor; núcleo verificam se a mensagem é para participantes se fazem presentes na máquina.
- a partir de comunicações ponto a ponto sobre a rede: um pacote por membro do grupo (n membros ⇒ n unicasts). Middleware no emissor terá que ter a lista de endereços de processos do grupo.

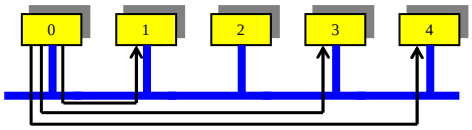
Difusão de mensagem



1 *Multicast*



1 *Broadcast*



3 *Unicast*

UDP e IP Multicast

UDP Multicast

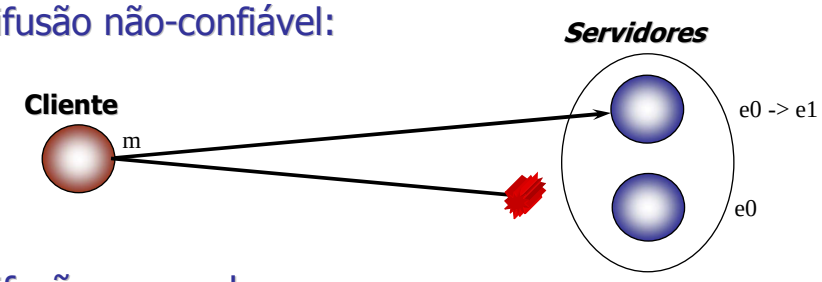
- Permite o envio simultâneo de datagramas a grupos de destinatários
- Grupos multicast são identificados por endereços IP de 224.0.0.0 a 239.255.255.255

UDP Multicast

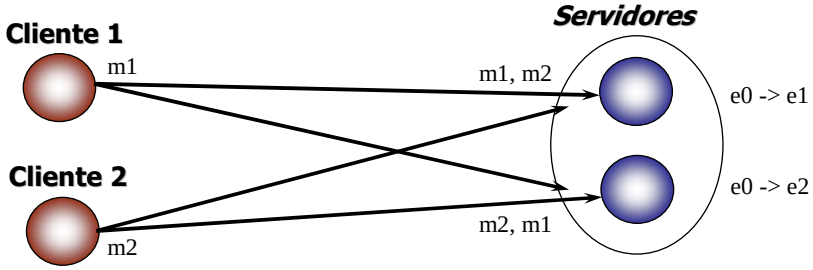
- Mais de um emissor pode mandar mensagens para o grupo (ou seja, mensagens vão de M emissores -> N receptores)
- Emissor não precisa fazer parte do grupo para enviar mensagens ao grupo, nem precisa saber quem são os seus membros; basta conhecer o endereço IP do grupo
- O receptor entra em um grupo (se torna um membro do grupo) e passa a receber as mensagens destinadas ao grupo

Problemas para difusão atômica

- Difusão não-confiável:

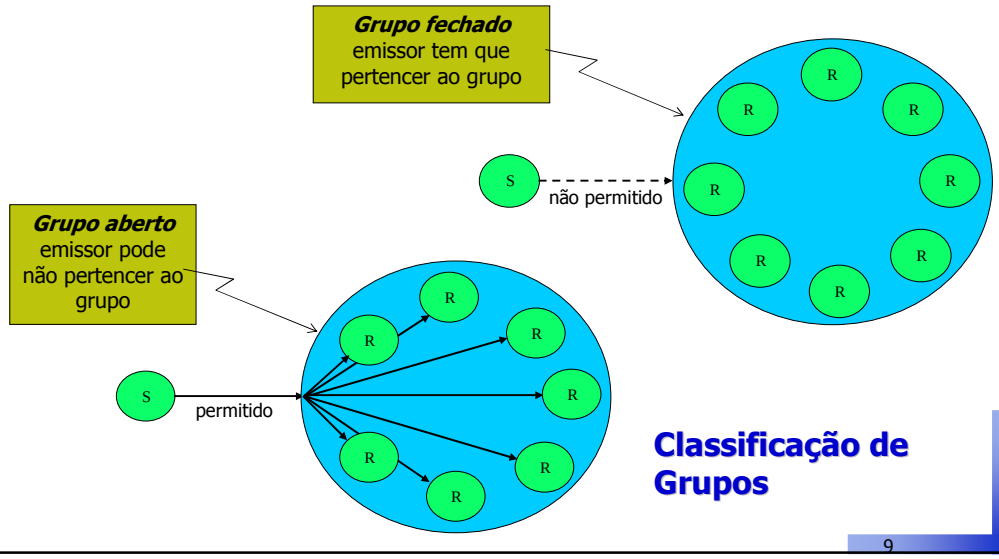


- Difusão sem ordem:



Grupos: classificação

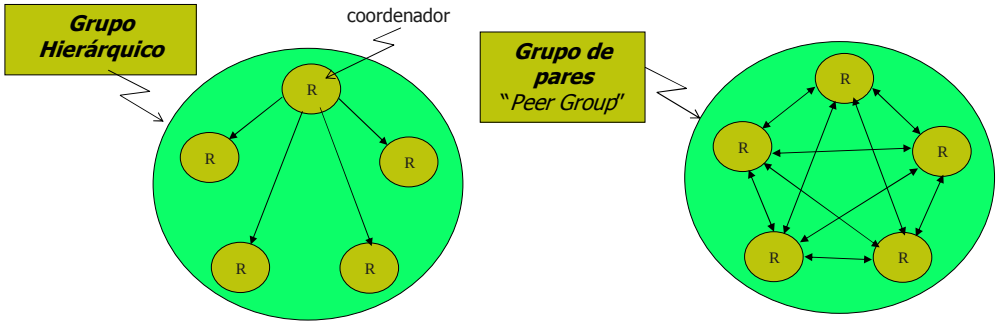
- **Grupos fechados X Grupos abertos**



Classificação de Grupos

Grupos: classificação

- Grupos simétricos X Grupos assimétricos



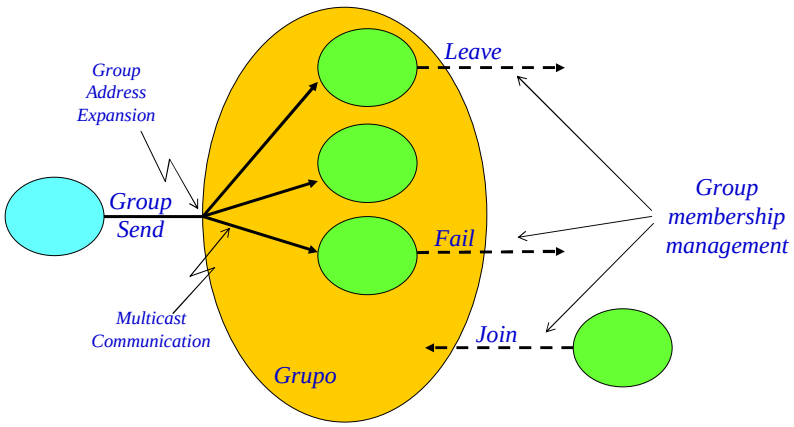
- Grupos simétricos: processos ou membros em papeis iguais na estrutura do grupo, grupo de pares
  - Vantagem sem ponto singular de falha
  - Desvantagem custos devido a gestão distribuída
- Grupos assimétricos: hierarquias de processos (objetos) custo só em tempo de falha; para decidir qualquer coisa é simples.

Pertinência (*membership*)

Grupos dinâmicos: necessidade de um gerenciamento de grupo
- Grupos podem ser criados e destruídos em tempo de execução
- Um processos (objeto) pode se juntar ou sair de um grupo

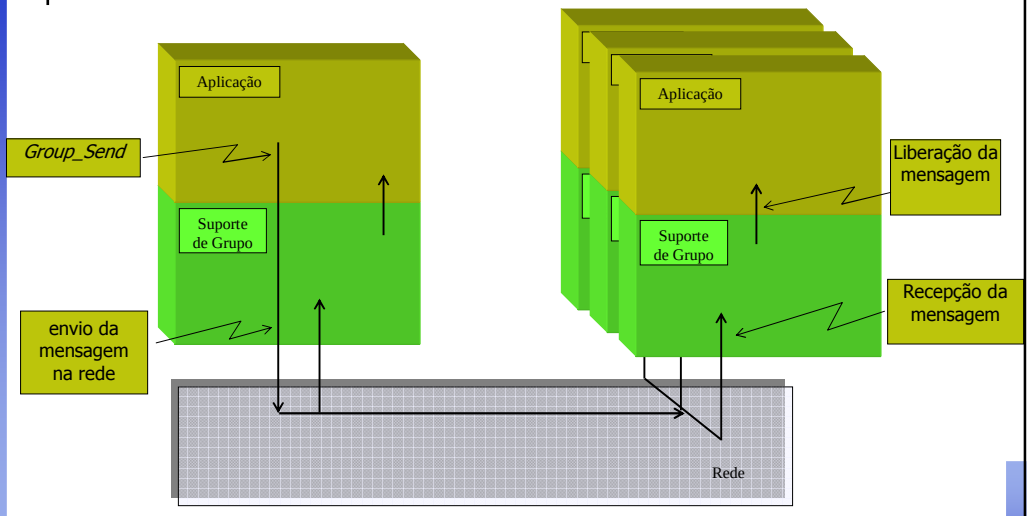
- Como manter as informações de pertinência?
  - Um serviço ⇒ solução centralizada
  - Uma solução distribuída ou descentralizada
- Protocolos distribuídos de *membership* ⇒ vista consistente dos componentes do grupo (*view*) em diferentes membros
  - Detectores de falhas dão o suporte ao serviço de membership
  - A entrada e saídas operações sincronizadas com as mensagens enviadas ao grupo. As informações de *membership* são enviadas pelos mesmos fluxos de mensagens de aplicação e estão sujeitas as mesmas propriedades de entrega.

Comunicação de Grupo



- Grupo aberto no qual um processo fora do grupo envia uma mensagem e sem conhecer o *membership*.
- O serviço de comunicação de grupo tem que gestionar mudanças no membership enquanto um *multicast* ocorre simultaneamente.

Modelo de comunicação de grupo



Comunicação de Grupo: Modelo

Algoritmo de difusão confiável

- Algoritmos de multicast confiável partem quase sempre da premissa de primitivas de multicast construídas sobre serviços subjacentes de comunicação ponto a ponto confiável e ordenação FIFO (First In First Out).

Com $p,q \in \wp$ e $m \in M$

{ R-*multicast*(*G*,*m*)}
    for all $p \in G$ **do**
        send (*m*,*p*);
    end for;

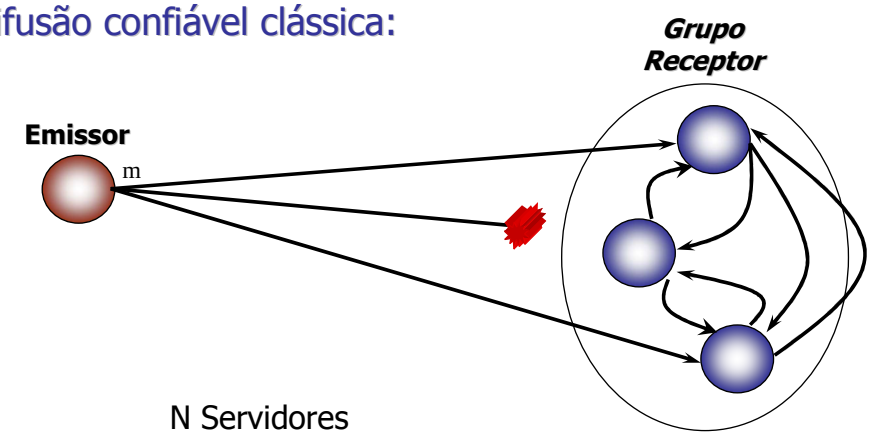
{ R-*deliver* (*m*)}
    receive(*m*) for the first time
    if *sender*(*m*) $\neq p$ **then**
        R-multicast(*G*,*m*)
    end if
    R-*deliver*(*m*)

Algoritmo de multicast Confiável

Alternativa: Algoritmos probabilísticos

Algoritmo de difusão confiável

- Difusão confiável clássica:



N Servidores

Requer N^2-N Mensagens

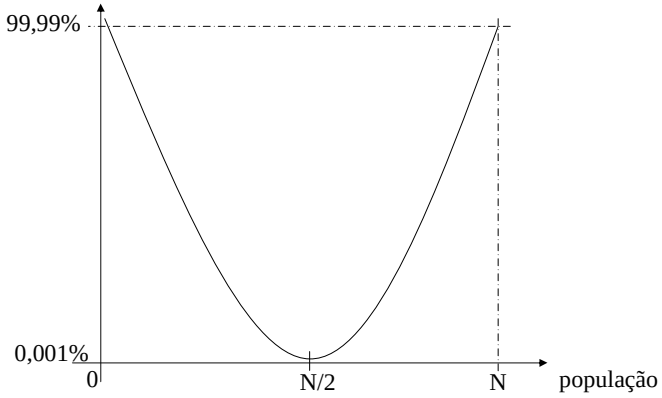
Grupos: propriedades

Propriedade *all-or-nothing* (confiabilidade da comunicação de grupo)

- acordo (ou atomicidade): a mensagem enviada ao grupo deve alcançar todos os processos membros ou nenhum destes.
  - protocolo de difusão confiável (*reliable broadcast*) garante propriedade *all-or-nothing*. implementação difícil várias trocas de mensagens
- Não interessa o número de perdas de pacotes e *crashes* de máquinas todos os processos corretos vão terminar recebendo a mensagem.

Algoritmo de difusão probabilístico

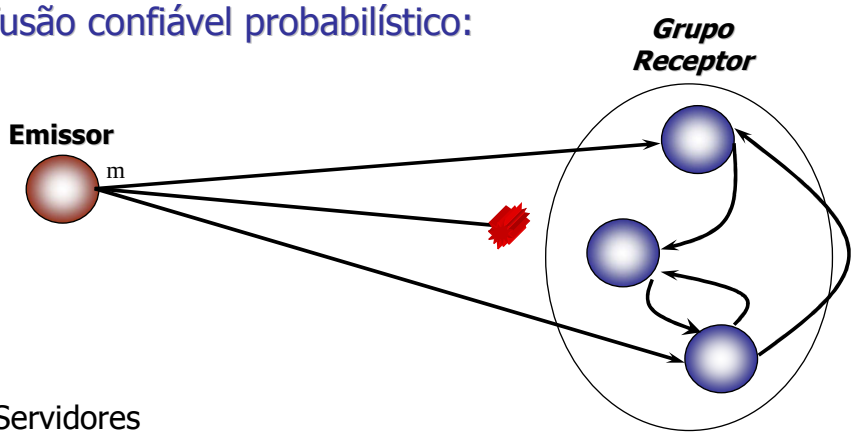
- Modelo epidêmico:



Como uma doença contagiosa se espalha em uma população

Algoritmo de difusão confiável

- Difusão confiável probabilístico:



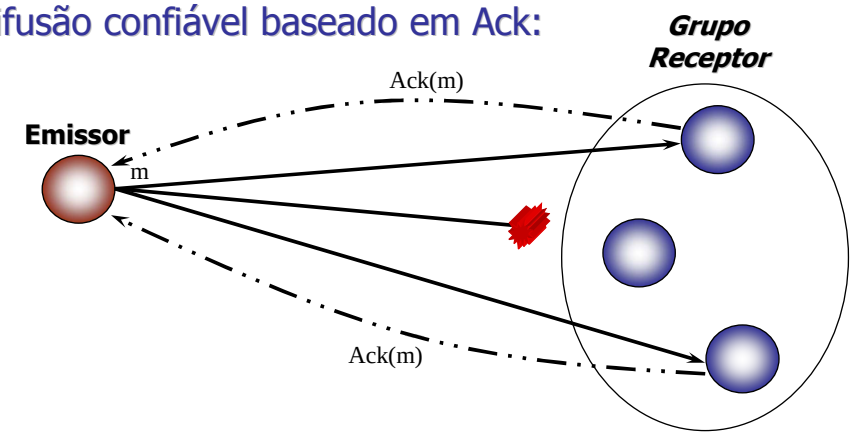
N Servidores

Requer ~$N^2/2$ mensagens

Quanto maior N mais próximo de 100%

Difusão confiável baseado em Ack e Nack

- Difusão confiável baseado em Ack:



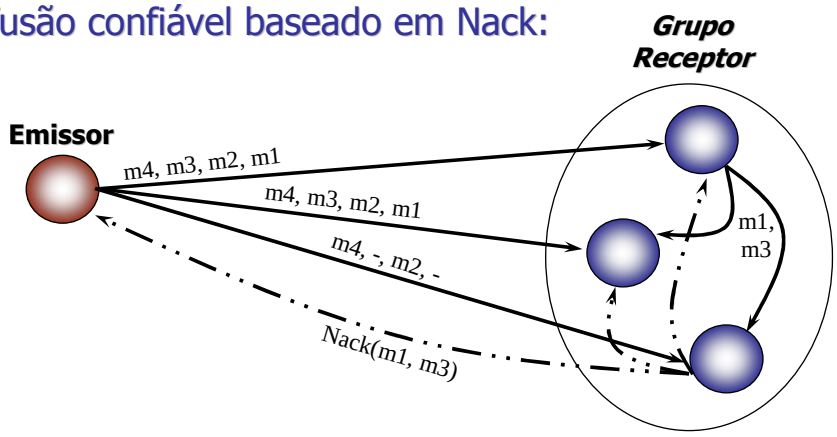
O emissor envia a mensagem e espera pelo reconhecimento (Ack) de cada receptor. Se o emissor não recebe um Ack de um receptor após um *timeout*, ele retransmite a mensagem para este.

Algoritmo de difusão confiável

- Difusão confiável baseado em Ack
  - Nesta abordagem, a correção de erro (retransmissão da mensagem perdida) é feita somente pelo emissor.
    - Problema: e se o emissor cai (*crash*) depois da primeira transmissão, quem vai fazer a correção de erro ?
  - Uma mensagem de reconhecimento (Ack) é enviada por cada receptor para cada mensagem recebida -> muito tráfego;
    - O ideal seria que todos processos receptores pudessem fazer a correção de erro também.
  - Como se calcula o *timeout* de espera para cada receptor ?
    - Testes para medir o atraso de uma mensagem -> timeout fixo;
    - Timeout adaptativo -> timeout variável.

Algoritmo de difusão confiável

- Difusão confiável baseado em Nack:



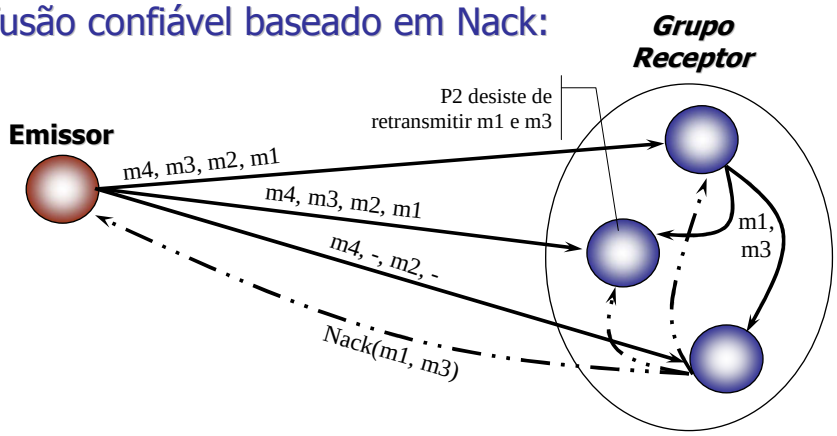
O emissor envia a mensagem para cada receptor. Se um receptor notar a falta de alguma mensagem de uma seqüência, ele envia um Nack para todos processos (emissor e receptores) pedindo retransmissão.

Algoritmo de difusão confiável

- Difusão confiável baseado em Nack
  - A retransmissão pode ser feita por qualquer processo.
  - Como evitar a explosão de retransmissões ?
    - Cada processo ao receber um Nack, espera um tempo aleatório antes de fazer a correção de erro.
      - Se durante esse tempo ele perceber que outro processo já fez a retransmissão solicitada no Nack, ele cancela.
    - Buffer de mensagens recebidas: cada processo deve guardar as mensagens recebidas no buffer para poder retransmiti-las caso algum processo solicite (Nack).
      - Mas o buffer tem tamanho limitado, quando sei que posso eliminar alguma mensagem do buffer ??
      - Uma mensagem *m* só pode ser eliminada do buffer do processo *p* quando *p* tiver a certeza que todos outros processos já receberam esta mensagem.
        - Se todos já receberam *m*, pra que guardá-la no buffer ?☺

Algoritmo de difusão confiável

- Difusão confiável baseado em Nack:

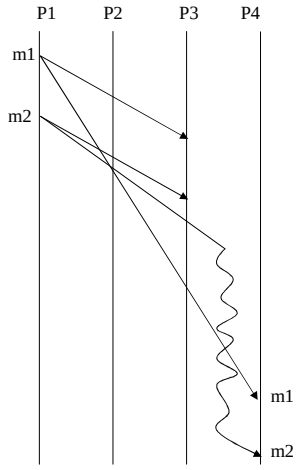


Para descarte de mensagens no buffer, todos processos mandam de tempos em tempos uma mensagem aos outros indicando as mensagens que já receberam.

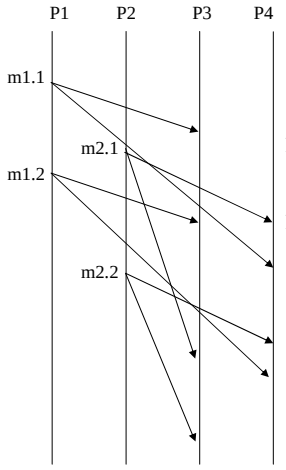
Difusão confiável com ordem FIFO

- A ordem FIFO é baseada na ordem de emissão. Pode ser implementado com número de seqüência na mensagem.

Exemplo 1: ordem FIFO



Exemplo 2: ordem FIFO



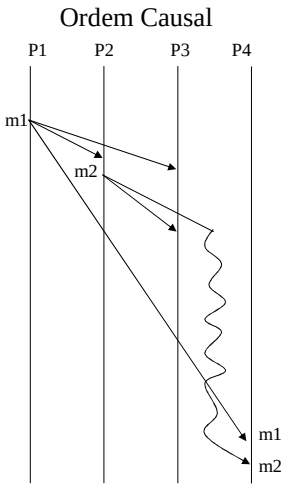
P3 recebe: m1.1, m1.2, m2.1 e m2.2

P4 recebe: m2.1, m1.1, m2.2 e m1.2

Em ambos casos a relação FIFO é atendida em P3 e P4.

Difusão confiável com ordem causal

- A ordem causal pode ser implementada colocando no cabeçalho da mensagem o(s) identificador(es) das mensagens que precedem a atual.



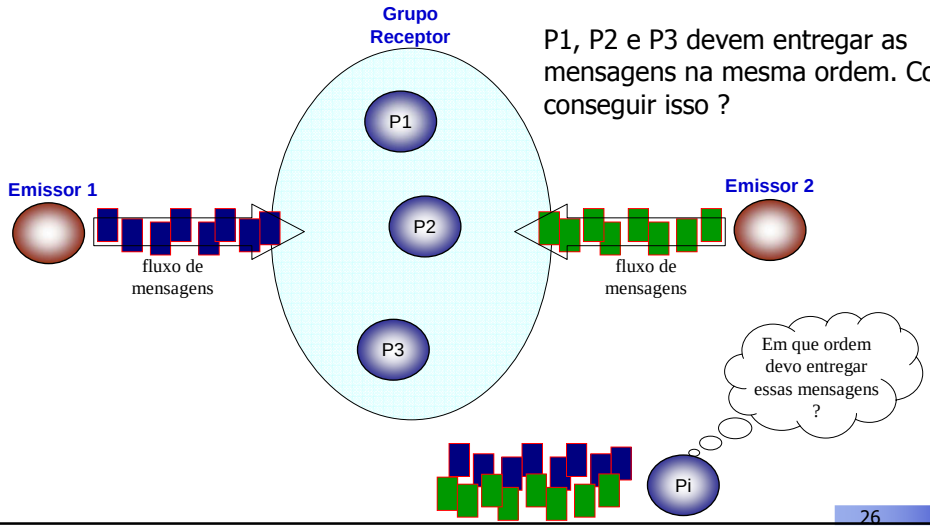
Relação de precedência m1 -> m2

A chegada da mensagem m1 fez com que P2 emitisse m2, logo m1 precede m2.

M_d(M_a, M_b, M_c) -> M_d só pode ser liberada para a aplicação se o processo tiver liberado M_a, M_b e M_c, pois essas mensagens precedem M_d.

Ordem total

- A ordem total visa garantir que todos processos recebam o mesmo conjunto de mensagens e na mesma ordem, baseada em qualquer critério.

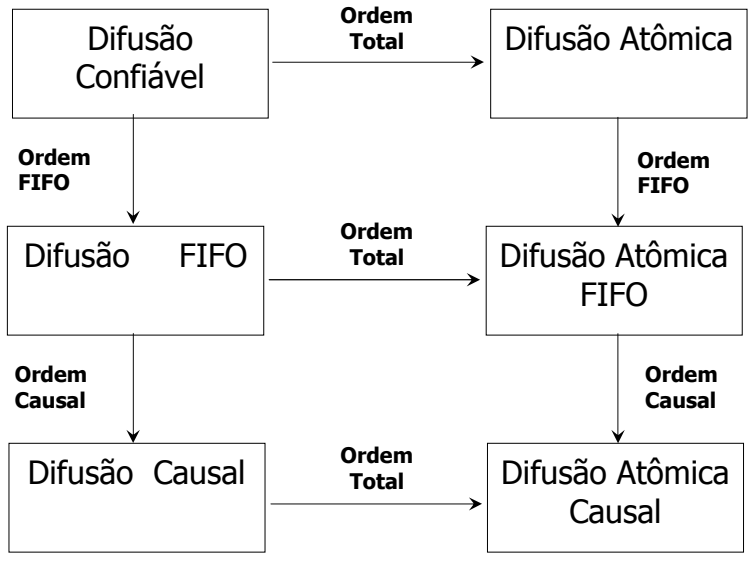


P1, P2 e P3 devem entregar as mensagens na mesma ordem. Como conseguir isso ?

Ordens totais

- ordem cronológica ou ordem de tempo global das emissões
  - Liberação da mensagem na ordem exata de emissão difícil de implementar esta ordem
- ordem consistente o sistema define uma ordem total qualquer das mensagens ⇒ todos os membros liberam segundo esta ordem.
- Protocolo que reúne as propriedades de ordem total (consistente ou de tempo global) e *all-or-nothing* é chamado difusão atômica.
- Outras ordens existem (ordem fifo, ordem causal, etc)

Relação entre as primitivas de difusão confiável



Mecanismos de ordenação total

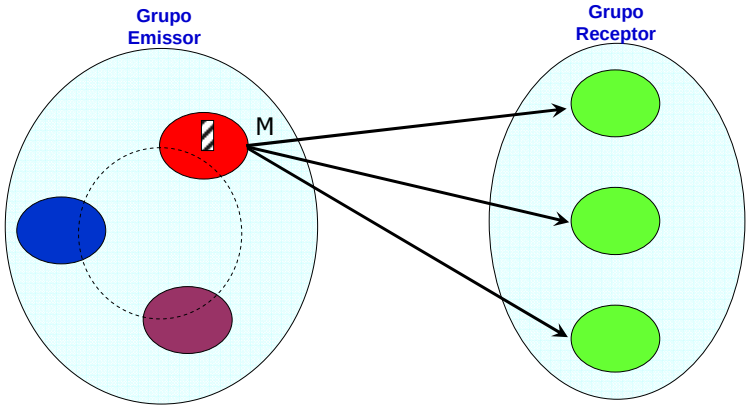
- Histórico da comunicação: baseado no algoritmo de ordenação de eventos do Lamport;
- Baseado em privilégio: o token dentro de um anel lógico de processos. O processo de posse do token tem direito de multicast;
- Baseado em seqüenciador (fixo ou móvel): a mensagem é enviada pro seqüenciador e ele repassa a mensagem pro grupo;
  - Sequenciador repassa a mensagem (Tandem);
  - Sequenciador define a ordem de entrega (ISIS);
- Acordo no destino: a mensagem em enviada pro grupo que executa um protocolo de acordo para decidir sobre a ordem de entrega da mensagem.

Mecanismos de ordenação total

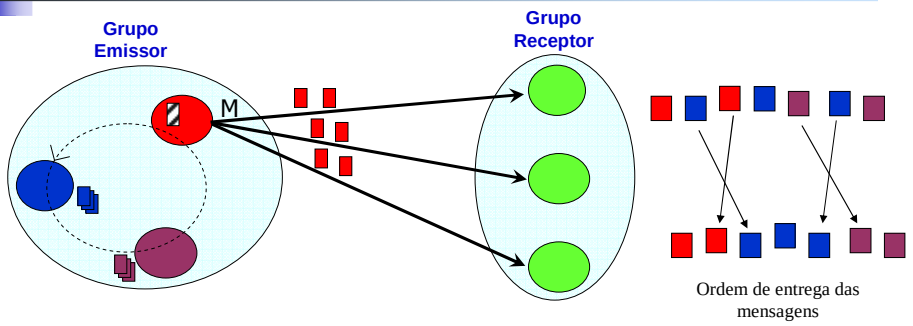
- Histórico da comunicação: baseado no algoritmo de ordenação de eventos do Lamport;
- Baseado em privilégio: o token dentro de um anel lógico de processos. O processo de posse do token tem o privilégio de *multicast*;
- Baseado em seqüenciador (fixo ou móvel): a mensagem é enviada pro seqüenciador e ele repassa a mensagem pro grupo;
  - Seqüenciador repassa a mensagem (Tandem);
  - Seqüenciador define a ordem de entrega (ISIS);
- Acordo no destino: a mensagem em enviada pro grupo que executa um protocolo de acordo para decidir sobre a ordem de entrega da mensagem.

Baseado em Privilégio

- Nesta técnica, os emissores formam um anel virtual onde circula um *token*. O emissor que estiver de posse do *token* tem o privilégio de difundir suas mensagens.



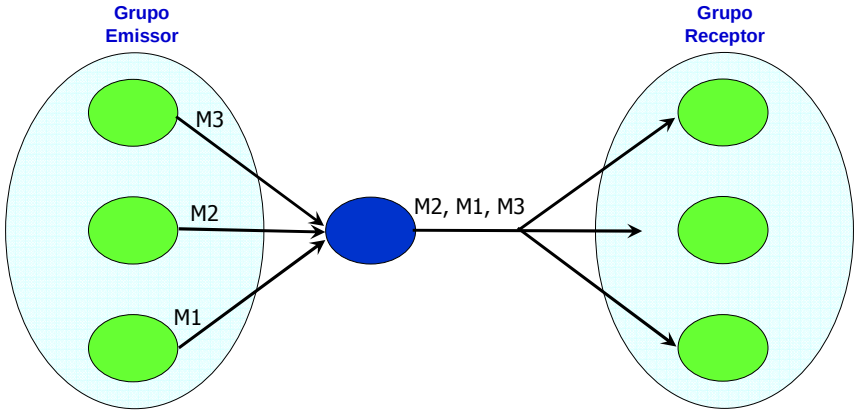
Baseado em Privilégio



- Problemas dessa abordagem:
  - Perda do token: processo que está usando o token falha;
  - Para um grupo com muitos membros, um emissor tem que esperar muito para poder enviar uma mensagem;
  - O grupo receptor tem que saber a seqüência do anel lógico dos receptores.

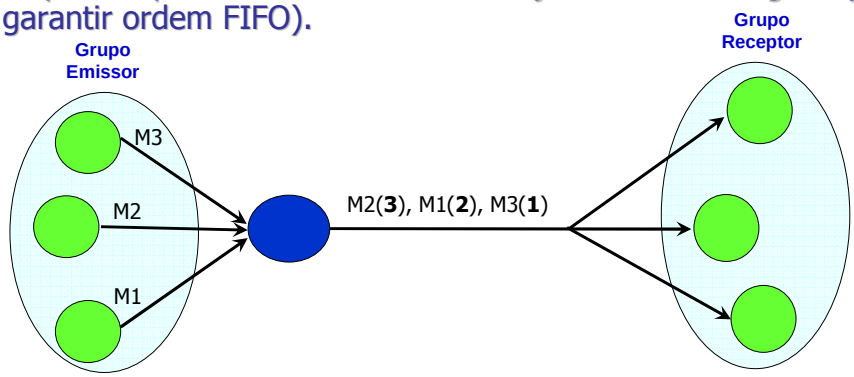
Seqüenciador Fixo

- Nesta técnica, os emissores enviam suas mensagens para um seqüenciador fixo. Este decide em que ordem as mensagens deverão ser entregues.



Seqüenciador Fixo

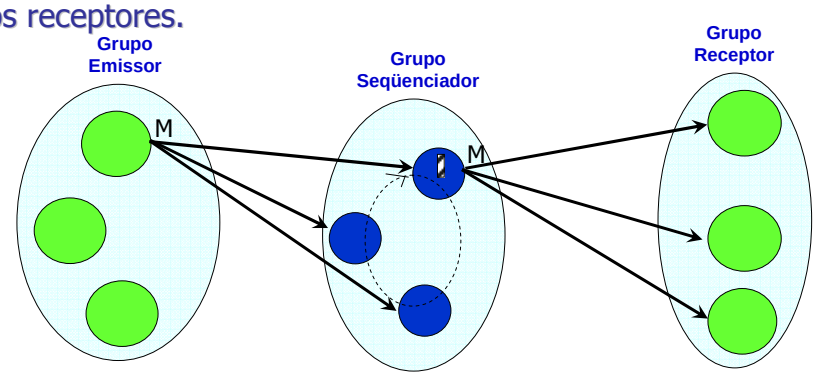
- Para garantir que os receptores entreguem as mensagens de acordo com a ordem estabelecida pelo seqüenciador um número de seqüência pode ser colocado no cabeçalho das mensagens (para garantir ordem FIFO).



- Problema: o seqüenciador é um simples ponto de falha. Se falhar todo o grupo fica indisponível.

Seqüenciador Móvel

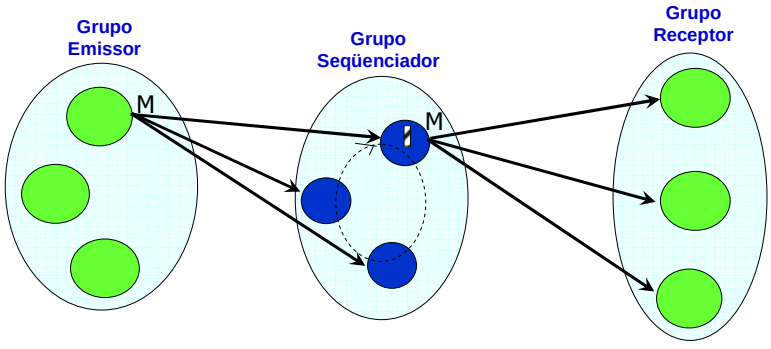
- Quando um emissor deseja enviar uma mensagem ao grupo receptor, ele difunde a mensagem no grupo seqüenciador. O seqüenciador que possuir o privilégio (token) repassa a mensagem aos receptores.



- O grupo seqüenciador não precisa ter muitos membros (2 a 4 membros).

Seqüenciador Móvel

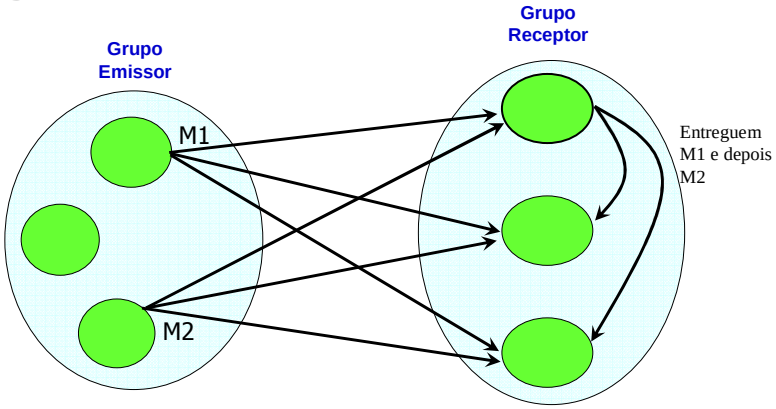
- Esta abordagem resolve o problema dos dois algoritmos anteriores: o grupo emissor poder ser escalável e não há um simples ponto de falha.



- Problemas dessa abordagem:
  - Perda do token: processo que está usando o token falha;
  - O grupo receptor tem que saber a seqüência do anel lógico dos seqüenciadores.

Seqüenciador ISIS

- Esta é uma variante de seqüenciador em que no grupo receptor tem um coordenador na qual, a cada período de tempo, envia uma mensagem aos outros indicando a ordem de entrega das mensagens.



Acordo no Destino

- As mensagens são difundidas (de forma confiável) diretamente para todos os membros do grupo receptor. Cada receptor faz uma proposta sobre a ordem de entrega e a decisão pode ser tomada pelo voto majoritário.