COMUNICAÇÃO ENTRE PROCESSOS
Comunicação de grupo

Comunicação entre processos (grupos)

- Comunicação *one-to-one*
  - Forma mais simples de comunicação entre processos
  - *point-to-point*, ou *unicast*
- Algumas aplicações → comunicação entre grupos de processos
  - oferecer facilidades para o programador
  - oferecer bom nível de desempenho
    - ex.:multicast melhor que relação a feixes de comunicação unicast
- Formas de comunicação em grupo
  - *one to many*
  - *many to one*
  - *many to many*

Comunicação entre processos (grupos)

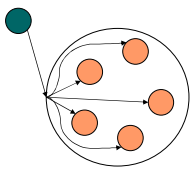
- Objetivo
  - Envio de mensagem para um grupo de processos através de uma *única operação*
- Grupo de processos
  - Coleção de processos que *agem* em conjunto
- Propriedades
  - *Mensagem enviada ao grupo é recebida por cada um dos seus membros*
  - Grupos devem ser *dinâmicos*

Comunicação entre processos (grupos)

- Formas de comunicação em grupo - *one-to-many*
  - também chamado *multicast*
  - *broadcast*: caso especial de *multicast* para todos processos em uma rede
  - exemplo:
    - gerente de servidores, todos oferecendo mesmo tipo de serviço
    - o gerente pode mandar mensagem a todos servidores perguntando por um servidor livre para assumir um pedido
    - seleciona primeiro que responde - resposta em unicast
    - gerente de servidores não tem que manter controle sobre servidores livres
  - outro exemplo:
    - achar servidor oferecendo um determinado tipo de serviço
    - mensagem em broadcast com "pergunta" - resposta em unicast

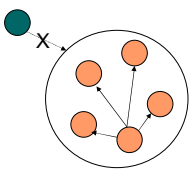
Comunicação entre processos (grupos)

- Formas de comunicação em grupo - *one-to-many*
  - Gerência de grupos
    - grupo aberto
      - qualquer processo pode mandar mensagens para o grupo como um todo
      - exemplo: servidores replicados para processamento distribuído formam grupo aberto pois clientes mandam pedidos para o grupo de servidores



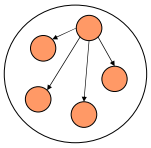
Comunicação entre processos (grupos)

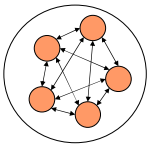
- Formas de comunicação em grupo - *one-to-many*
  - Gerência de grupos
    - grupo fechado
      - somente membros do grupo podem mandar mensagem para o grupo
      - ex.: grupo de servidores trabalhando em problema comum (ex.: nodos trocam informações sobre carga, para balanceamento - grupo pode ser fechado pois os nodos trocam informações somente entre eles)



Comunicação entre processos (grupos)

- Formas de comunicação em grupo - *one-to-many*
  - Gerência de grupos
    - Processos hierarquizados X Grupo simétrico





- Coordenador + trabalhador
- Falha do coordenador
- Tolerância a falhas
- Tomada de decisão complicada

Comunicação entre processos (grupos)

- Formas de comunicação em grupo - *one-to-many*
  - Gerência de grupos
    - sistema deve permitir criação e remoção dinâmica de grupos, bem como processos poderem entrar (*join*) e sair de grupos dinamicamente
    - mecanismo simples: servidor de grupo centralizado
    - todos pedidos mandados a este servidor
    - facilidade para manter informação atualizada sobre todos os grupos e exatamente os membros pertencentes aos grupos
    - problema tradicional em soluções centralizadas:
      - baixa confiabilidade - servidor de grupo falha - todo grupo fica comprometido

Comunicação entre processos (grupos)

- Formas de comunicação em grupo - *one-to-many*
  - Gerência de grupos
    - problemas de servidor de grupos centralizado:
      - baixa confiabilidade
      - baixa "escalabilidade" (potencial para crescer)
    - replicação do servidor de grupo para resolver tais problemas
      - overhead aumenta - manter informação dos grupos consistente em todos servidores replicados

Comunicação entre processos (grupos)

- Formas de comunicação em grupo - *one-to-many*
  - *Buffered* e *unbuffered multicast*
    - multicast é assíncrono:
      - não é realístico o enviador esperar que todos recebedores do grupo multicast estejam prontos para receber
      - o enviador pode não saber quantos recebedores existem no grupo
    - *unbuffered*: mensagem chega e processo recebedor não está pronto - kernel no recebedor descarta mensagem
    - *buffered*: mensagem armazenada para o processo receptor

Comunicação entre processos (grupos)

- Formas de comunicação em grupo - *one-to-many*
  - Semântica *send-to-all* e *bulletin-board*
    - *send-to-all*: cópia da mensagem é mandada para cada processo do grupo, e a mensagem é armazenada até sua recepção
    - *bulletin-board*: mensagem é endereçada a um canal; processos recebedores copiam mensagem do canal. Dita mais flexível pois:
    - a relevância de uma mensagem a um recebedor particular depende do estado do recebedor
      - mensagens não aceitas após um período de tempo podem não ser mais úteis; seu valor depende do estado do enviador
      - ex.: server manager procurando servidor para pedido - com bborad somente servidores aptos para aceitar request leriam do canal de multicast - mensagem poderia ser retirada do canal assim que o enviador aquele pedido não tenha mais necessidade

Comunicação entre processos (grupos)

- Formas de comunicação em grupo - *one-to-many*
  - Confiabilidade flexível em mulicast
    - aplicações diferentes tem diferentes requisitos de confiabilidade
    - **0-*reliable***: enviador não espera resposta de nenhum recebedor.
      Ex.: *time signal generator*
    - **1-*reliable***: enviador espera resposta de 1 recebedor - qualquer um.
      Ex.: server manager a procura de um servidor
    - ***m-out-of-n-reliable***: o grupo consiste de n recebedores e o enviador espera uma confirmação de m (1<m<n) dos n recebedores.
      Ex.: algoritmos de consenso por maioria - usados para controle de consistência de informações replicadas usam este tipo de confiabilidade, com m=n/2
    - ***all-reliable***: o enviador espera resposta de todos os recebedores do grupo.
      Ex.: mensagem para atualizar réplicas de arquivo em todos servidores de arquivos envolvidos (grupo)

Comunicação entre processos (grupos)

- Formas de comunicação em grupo - *one-to-many*
  - ***Multicast Atômico***
    - confiabilidade *all-reliable*
    - característica *all-or-nothing* - ou entrega mensagem a todos, ou a nenhum
    - método 1:
      a) transmite a todos;
      b) espera ack de todos;
      c) depois de timeout, retransmite aos ainda não confirmados
      d) volta para b) considerando só os que não confirmaram ainda
      e) quando todos confirmaram, envio em multicast acabou
    - ? E se falhas acontecem ?
      - No enviador durante o multicast ?

Comunicação entre processos (grupos)

- Formas de comunicação em grupo - *one-to-many*
  - ***Multicast Atômico***
    - Método 2:
      - enviador
        - Cada mensagem tem um identificador para distingui-la das demais
        - o enviador a manda para o multicast group
        - uso de timeout e retransmissões - em falta de confirmação
      - recebedor:
        - verifica identificador da mensagem para ver se é nova
        - se não for nova, descarta
        - se for nova manda a mesma mensagem para o multicast group em multicast atômico - uso de timeout e retransmissões

Comunicação entre processos (grupos)

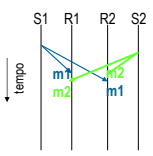
- Formas de comunicação em grupo - *one-to-many*
  - ***Multicast Atômico***
    - Método 2:
      - acontece um *flooding* da mensagem
      - caro - muitas mensagens
        - deve-se optar por multicast atômico somente quando realmente necessário
      - garante que todos processos sobreviventes do grupo "eventualmente" receberão a mensagem, mesmo que o processo enviador falhe durante o multicast
      - "eventualmente" - vai receber, porém não se sabe com que atraso.

Comunicação entre processos (grupos)

- Formas de comunicação em grupo - *many-to-one*
  - recepção seletiva e não seletiva
  - não seletiva
    - recebedor quer receber de qualquer um do grupo
  - seletiva
    - recebedor quer controlar dinamicamente de quem receber
    - ex.: processo buffer recebe mensagens de produtores e consumidores;
      se buffer cheio -> aceitar mensagens só de consumidores
      se buffer vazio -> aceirar mensagens só de produtores
      outro caso -> aceitar mensagens dos dois

Comunicação entre processos (grupos)

- Formas de comunicação em grupo - *many-to-many*
  - aspectos já discutidos para one-to-many e many-to-one se incluem +
  - **entrega ordenada de mensagens**
  - entrega das mensagens em ordem aceitável para a aplicação
  - ex.: 2 enviadores mandam mensagens para atualizar mesmo registro da base de dados para dois servidores mantendo cada um uma réplica da base de dados. Se mensagem recebida em ordens diferentes pelos servidores, teremos resultados diferentes - inconsistência
  - ordenação de mensagens requer mecanismo de sequenciamento (serialização)



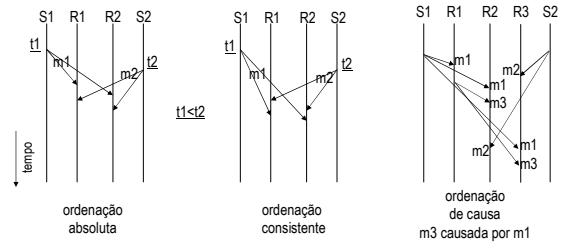
Comunicação entre processos (grupos)

- Formas de comunicação em grupo - *many-to-many*
  - em *one-to-many*: sequenciar multicasts
    - enviador inicia próximo multicast só depois de acabar o que já está em curso
  - em *many-to-one*: mensagens são recebidas pelo recebedor na ordem em que chegam da rede ...
  - em *many-to-many ?*
    - Vários enviadores e vários recebedores em diversos pontos
    - rede apresenta atrasos diferentes dependendo das posições dos processos ...
    - falhas de links, roteadores ...
    - como garantir mesma percepção de ordem para os vários recebedores ?

Comunicação entre processos (grupos)

- Ordenação de mensagens
  - Ordenação absoluta
    - quando diversas mensagens são transmitidas para um grupo, as mensagens chegam para todos os membros do grupo na mesma ordem que foram enviadas
  - Ordenação consistente
    - quando diversas mensagens são transmitidas para um grupo, as mensagens chegam para todos os membros do grupo na mesma ordem
  - Ordenação de causa
    - garante que se o evento de envio de uma mensagem causa o evento de envio de outra mensagem, então as mensagens são enviadas a todos receptores na mesma ordem.

Comunicação entre processos (grupos)



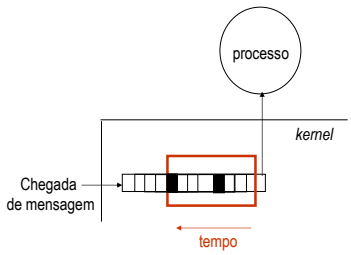
ordenação absoluta | ordenação consistente | ordenação de causa — m3 causada por m1

Comunicação entre processos (grupos)

- Ordenação de mensagens - ordenação absoluta - método
  - usar *timestamps* globais como identificadores das mensagens
  - supor sincronização dos relógios
  - kernel do recebedor mantém mensagens recebidas
  - usa janela deslizante de tempo para entregar para o processo - tempo fixo e ajustado considerando o maior atraso na comunicação entre processos do grupo
  - mensagens dentro da janela esperam para serem entregue pois mensagem com *timestamp* menor pode chegar
  - quando passa o tempo da janela, mensagens podem ser entregue pois garante que as diferenças de atraso no envio dos diversos processos para o recebedor serão cobertas pelo tempo de espera da janela

Comunicação entre processos (grupos)

- Ordenação de mensagens - ordenação absoluta - método

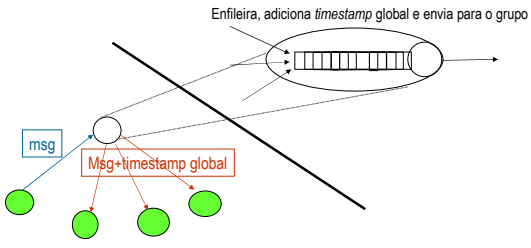


Comunicação entre processos (grupos)

- Ordenação de mensagens - ordenação consistente - método
  - sincronização de relógios - dificuldade de implementar
  - ordenação absoluta - muitas aplicações não necessitam
  - para muitas aplicações basta que as mensagens sejam entregues em uma ordem consistente, ou seja: se um processo percebe mensagem m1 antes de m2 então todos processos envolvidos também percebem m1 antes de m2 (mesmo que m2 tenha sido enviada antes de m1 !)
  - método 1: seqüenciador
    - enviadores mandam mensagens para o grupo para um seqüenciador
    - seqüenciador associa número de seqüência a cada mensagem e manda por multicast
    - recebedor salva mensagens e espera seqüência se completar para entregar em ordem

Comunicação entre processos (grupos)

- Ordenação de mensagens - ordenação consistente - método
  - método 1: seqüenciador



Comunicação entre processos (grupos)

- Ordenação de mensagens - ordenação consistente - método
  - método 1: seqüenciador
    - centralização: sujeito a falhas - prejudica todo grupo
  - protocolo ABCAST:
    - enviador associa número de seqüência crescente temporário à mensagem e manda em multicast
    - cada recebedor retorna um número de seqüência que propõe para a mensagem recebida
      - max(Fmax,Pmax) + 1 + i/N
      - Fmax: número máximo final já acordado no grupo, guardado pelo processo i
      - Pmax: número máximo de seqüência já proposto por este recebedor (processo i)
      - i: número do processo
      - N: número total de processos no grupo

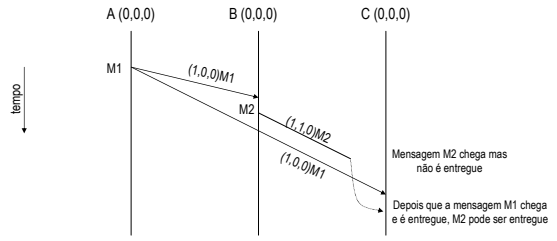
Comunicação entre processos (grupos)

- Ordenação de mensagens - ordenação consistente - método
  - protocolo ABCAST:
    - quando enviador recebe as propostas de número de sequencia de todos os recebedores (confirmação implícita), então ele opta pelo maior número de sequencia proposto e manda commit com o número escolhido
    - número de seq. é garantidamente único devido ao termo i/N (no caso de processo estar acontecendo simultaneamente em dois enviadores, garante-se com i/N que não haverão propostas com mesmo nro. Seq.)
    - mensagens com commit podem ser entregues aos processos na ordem conforme o número de seqüência associado

Comunicação entre processos (grupos)

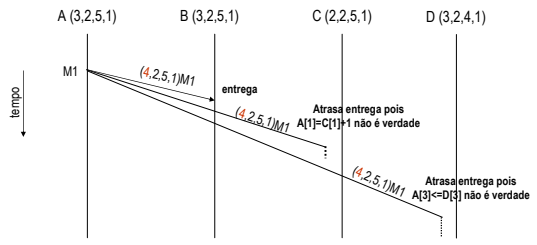
- Algoritmo para implementar ordenação de causa - CBCAST:
  - grupo com *n* membros
  - cada membro tem um vetor com *n* elementos
  - *i-ésimo* elemento do vetor representa a última mensagem recebida do processo *I*
  - elementos dos vetores são inicializados com 0 (zero)
  - ao enviar uma mensagem processo incrementa seu elemento no vetor
  - quando mensagem chega ao receptor, este verifica se ela depende de outra mensagem
    - condição 1: S[i] = R[i] + 1
      // garante que recebedor não perdeu nenhuma mensagem do enviador
    - condição 2: S[j] <= R[j] para todo i diferente de j
      // garante que enviador não recebeu nenhuma mensagem que o recebedor ainda não
      // recebeu - ou seja: recebedor tem que ter recebido mesmo conjunto (ou mais)
      // de mensagens que enviador - mas contrário não é necessário
    - se condições 1 e 2 **não** Verdadeiro então armazenar mensagem e avaliar novamente na chegada de outra mensagem
    - se condições 1 e 2 **ok**: entregar mensagem para aplicação - está ordenada causalmente

Comunicação entre processos (grupos)



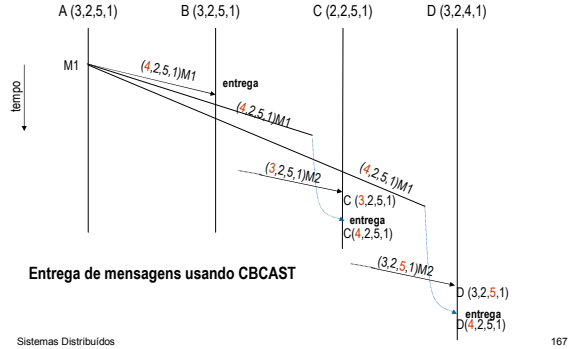
Entrega de mensagens usando CBCAST

Comunicação entre processos (grupos)



Entrega de mensagens usando CBCAST

Comunicação entre processos (grupos)



Entrega de mensagens usando CBCAST