SISTEMAS DISTRIBUÍDOS

Capítulo 8 - Sistemas de comunicação indireta

Nota prévia

A estrutura da apresentação é semelhante e utiliza algumas das figuras do livro de base do curso

G. Coulouris, J. Dollimore, T. Kindberg, G. Blair
Distributed Systems - Concepts and Design,
Addison-Wesley, 5th Edition, 2011

Para saber mais:

secção 6.1-6.4, 15.4 – apenas parte relativa aos tipos de multicast (não entrar na parte de implementação)

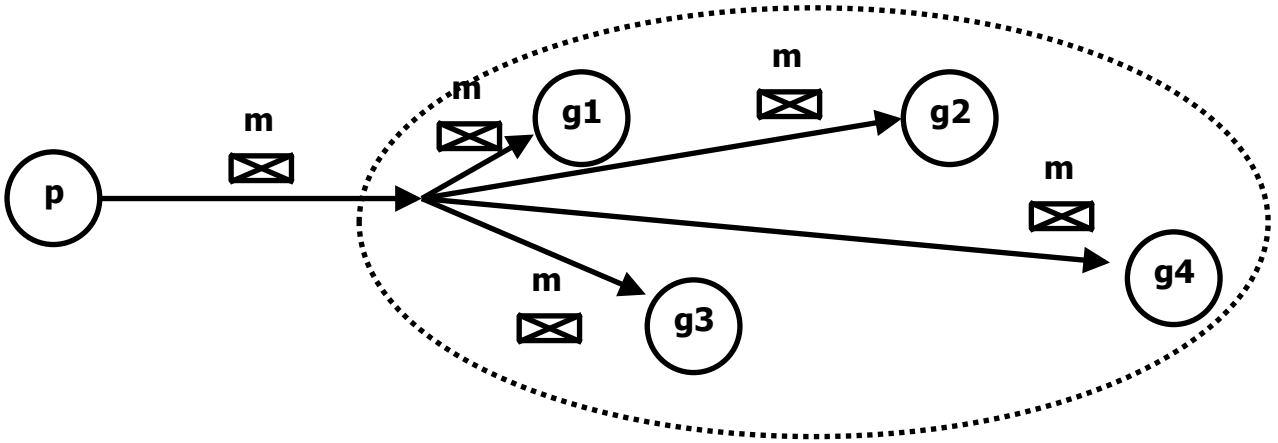
Organização do capítulo

Comunicação em grupo

COMUNICAÇÃO EM GRUPO

Um grupo é um conjunto dinâmico de *communication end-points* (portas, processos, servidores, etc.) que é tratado pelo sistema como uma entidade única do ponto vista da comunicação

Operação *multicast*: envio de uma mensagem para todos os elementos do grupo, usualmente de forma que a filiação (*membership*) do grupo é transparente para o emissor



ALGUMAS DEFINIÇÕES

Comunicação ponto a ponto (ou *unicast*) [1x1] – envio de uma mensagem de um único emissor para um único receptor

Formas de comunicação multi-ponto

- Comunicação *multicast* ou em grupo [1xN] – envio de uma mensagem de um emissor para um grupo de receptores
  - Pode exigir que o emissor pertença ao grupo ou não
  - Forma de identificar o grupo
- Comunicação broadcast ou por difusão [1xAll] – envio de uma mensagem de um emissor para todos as máquinas do sistema
- Comunicação funcional ou anycasting [1x1 among N] – envio de uma mensagem de um emissor para um de um grupo de receptores

CARACTERÍSTICAS DO *MULTICAST*

Multicasting pressupõe:

- endereço / identificador único do grupo
- composição dinâmica não necessariamente visível
- a modificação da composição é feita de forma independente dos emissores
- gestão, localização, comunicação, etc. com ou dos membros do grupo é assegurada pelo suporte de materialização da comunicação multi-ponto

Interface

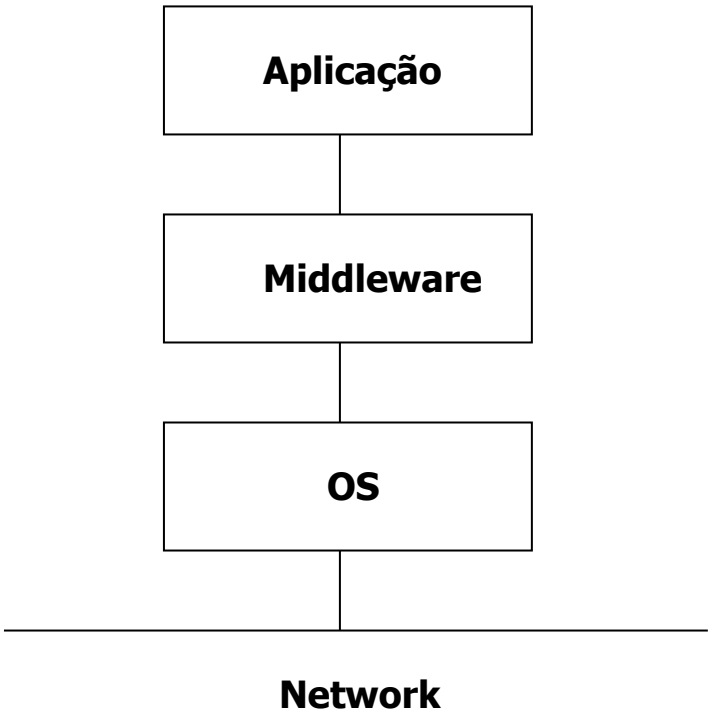
- createGroup, destroyGroup, joinGroup, leaveGroup, send, receive
- (disseminação de eventos) open, destroy, subscribe, unsubscribe, publish, receive (call-back)

ENTREGA DAS MENSAGENS

Diz-se que sistema de comunicação entrega uma mensagem quando a mensagem é entregue à aplicação

O *middleware* pode reordenar as mensagens recebidas localmente 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



CARACTERIZAÇÃO DO MULTICAST: FIABILIDADE

Multicast não fiável (unreliable multicast) – em caso de falha, não existem garantias sobre a entrega das mensagem aos vários elementos do grupo

- e.g. IP mutlicast

Multicast fiável (reliable multicast) – uma mensagem enviada para um grupo ou é entregue por todos os membros correctos (que não falham) ou por nenhum

- Membros que falham podem ter entregue a mensagem ou não
- Implementação simples (e errada): o emissor emite para cada um dos elementos do grupo de forma fiável (acks+retransmissões)

Porquê errada?

CARACTERIZAÇÃO DO MULTICAST: FIABILIDADE

Multicast não fiável (unreliable multicast) – em caso de falha, não existem garantias sobre a entrega das mensagem aos vários elementos do grupo

- e.g. IP mutlicast

Multicast fiável (reliable multicast) – uma mensagem enviada para um grupo ou é entregue por todos os membros correctos (que não falham) ou por nenhum

- Membros que falham podem ter entregue a mensagem ou não
- Implementação simples (e errada): o emissor emite para cada um dos elementos do grupo de forma fiável (acks+retransmissões)
  - 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

Uniform agreement: se algum processo que falha entrega a mensagem, todos os processos correctos devem entregar a mensagem

CARACTERIZAÇÃO DO MULTICAST: ORDEM

Sem ordem – as mensagens podem ser entregues por diferentes ordens em diferentes processos

Ordem FIFO – as mensagens de um processo são entregues pela ordem de emissão em todos os processos

Ordem total – as mensagens m1, m2 são entregues por ordem total, se forem entregues pela mesma ordem em todos os processos

- ∀pi, receive(m1,pi) < receive(m2,pi) ou
- ∀pi, receive(m1,pi) > receive(m2,pi)

NOTA: a ordem de entrega das mensagens de um mesmo emissor pode ser diferente da ordem de emissão

Um sistema de multicast fiável no qual as mensagens são entregues por ordem total chama-se sistema de **multicast atómico**

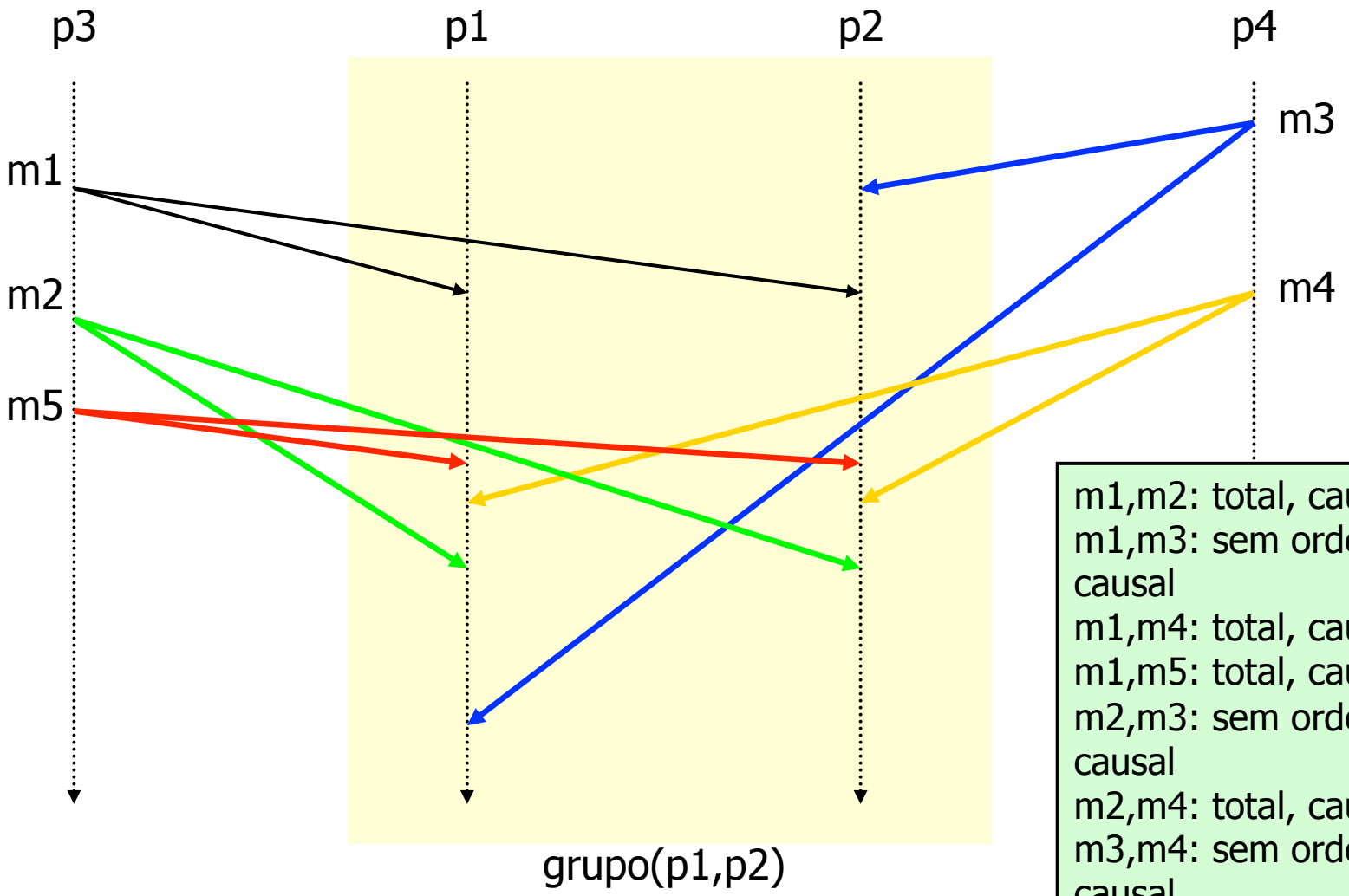
CARACTERIZAÇÃO DO MULTICAST: ORDEM

Ordem causal – se m1 pode causar m2, m1 deve ser entregue sempre antes de m2

- Se duas mensagens forem emitidas pelo mesmo processo, considera-se que o envio da primeira mensagem pode causar o envio da segunda
  - Neste caso, a recepção da primeira deve acontecer antes da recepção da segunda
- Se duas mensagens forem emitidas em processos diferentes, a primeira mensagem pode causar a segunda se for recebida antes do envio da segunda (ou transitivamente incluindo outras mensagens)
  - Se uma mensagem não pode causar a outra, qualquer ordem de entrega respeita a ordem causal

NOTA: a noção de causalidade estende-se a qualquer tipo de comunicação

ORDENS: EXEMPLO



m1,m2: total, causal
m1,m3: sem ordem, causal
m1,m4: total, causal
m1,m5: total, causal
m2,m3: sem ordem, causal
m2,m4: total, causal
m3,m4: sem ordem, não causal
m2,m5: total, não causal, não FIFO

CARACTERIZAÇÃO DO MULTICAST: VISTAS

Def: 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 efectuada 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

FACETAS DA COMUNICAÇÃO MULTICAST: RESUMO

- Fiabilidade
  - Multicast fiável
  - Multicast não-fiável
- Ordenação das mensagen
  - Sem ordem
  - Ordem FIFO
  - Ordem total
  - Ordem casual
- Relativamente às vistas
  - Sincronia virtual

Sistemas de Comunicação em Grupo e Replicação de dado

Pode-se usar um sistema de comunicação em grupo para propagar e executar operações num conjunto de réplicas. Que semântica usar nos seguintes cenários:

- Pretende-se manter um forúm de mensagens, em que uma mensagem pode ter uma resposta. Uma operação que cria um nova mensagem/resposta deve ser entregue com que semântica?
- Pretende-se manter o stock dum produto, garantindo que o seu valor não excede um dado limite. Uma operação para modificar o stock deve ser entregue com que semântica?

Sistemas de disseminação de eventos / Publish/ Subscribe

Extensão dos sistemas Message-Oriented Middleware (MOM) para grupos

- Pode ser visto como um sistema de multicast com interface diferente
- Comunicação por mensagens

Ideia base:

- Processos produzem mensagens que difundem
  - Produtores não conhecem subscritores
  - Interacção assíncrona
- Subscritores recebem subconjunto das mensagens emitidas

PUBLISH/SUBSCRIBE: BASEADO EM CANAIS/TÓPICOS

Existem canais dedicados a determinados tópicos

- Todos os subscritores recebem todas as mensagens

Interface (e.g.):

```
Channel c = open( "SD")// abre canal para publicar mensagens
c.publish( "msg 1")    // publica mensagem
c.close()

Channel c = open( "SD")
c.subscribe( new MsgListener() {
   public void newMessage( Message msg) {
          ... // processo mensagem
  }
})
```

PUBLISH/SUBSCRIBE: COM FILTRAGEM POR CONTEÚDO

As mensagens enviadas têm, geralmente, um conjunto de atributos definidos pelo emissor

- Os subscritors definem quais as mensagens que estão interessados em receber através de condições sobre os atributos

Interface (e.g.):

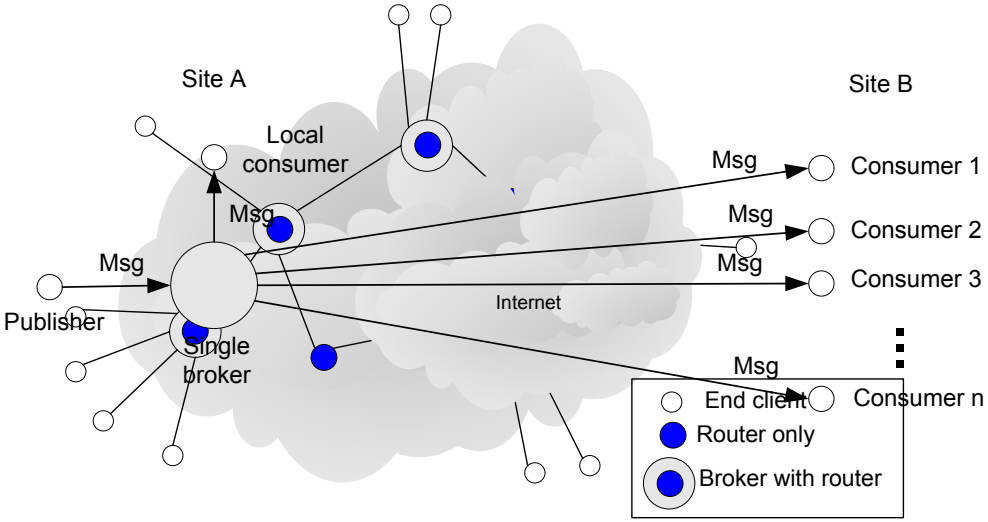
```
Channel c = open() // abre canal para publicar mensagens
c.publish( "msg 1", {"SD"})       // publica mensagem
c.close()

Channel c = open()
c.subscribe( new MsgFilter() {
    public boolean match( Message msg, AtributeSet set) {
            return set.contains( "SD");
  }
}, new MsgListener() {
    public void newMessage( Message msg) {
            ... // processo mensagem
  }
})
```

PUBLISH/SUBSCRIBE: ARQUITECTURAS

Cliente/servidor

Permite implementar funcionalidade adicional facilmente – e.g.: persistência, filtragem



Peer-to-peer

Sem ponto central de falhas, potencialmente mais escalável

Recorrendo a multicast tradicional ou baseado em comunicação epidémica

PUBLISH/SUBSCRIBE: PROPRIEDADES

- Desacoplamento entre componentes
  - Vantagens:
    - Emissor não necessita de conhecer receptor
    - Emissor e receptor não necessitam de estar a funcionar ao mesmo tempo
    - Emissor não necessita de esperar pelo receptor - assincronismo
    - Melhor desempenho
  - Desvantagens:
    - Mais complicado definir propriedades exactas do sistema
- Modelo de programação *event-driven*
  - Modelo reactivo - como nos GUIs

PUBLISH-SUBSCRIBE: SISTEMAS E *STANDARDS*

- Web-services
  - WS-Notification
    - Tem mais funcionalidade que WS-Eventing, mas são mutuamente incompatíveis
  - WS-Eventing
- Java: JMS
- Corba:
  - Event service
  - Notification service
- Sistemas:
  - TiBCO, IBM Gryphon, Apache pubscribe

SISTEMAS: COMO IMPLEMENTAR?

Suponham que têm um backbone de caches de internet, uma por ISP em cada país

(nota: por exemplo, a akamai tem esta infraestrutura)

Como actualizar as caches?

==========

Sistema de controlo de tráfego áereo

Radares monitorizam espaço aéreo

Quem está interessado na informação? Controlo de tráfego aérea, companhias, aeroportos

==========

Editor de texto cooperativo

SISTEMA: COMO IMPLEMENTAR?

Sistema de disseminação de informação da bolsa

==========

VC2

Para saber mais

G. Coulouris, J. Dollimore, T. Kindberg, G. Blair, Distributed Systems –Concepts and Design, Addison-Wesley, 5th Edition, 2011

capítulo 6.1-6.4, 15.4