Modelos de Programação Distribuída (parte 3)

CAPÍTULO III

Ricardo Costa – rcosta@estg.ipp.pt

Modelos de Programação Distribuída

Modelos de comunicação por mensagens

- Exemplo: Comunicação por mensagens através de Sockets (em Java)

Modelos Arquitecturais

- Cliente / Servidor
- Múltiplos Servidores
- Proxies
- Peer processes

Modelos Fundamentais

- Interação
- Falhas
- Segurança

Modelos de Programação Distribuída

Modelos fundamentais

- Os sistemas distribuídos podem ainda ser analisados segundo 3 aspectos transversais a todos os sistemas:
  - Modelo de Interação (ou de sincronismo)
  - Modelo de Falhas (ou Avarias)
  - Modelo de Segurança

Modelos de Programação Distribuída

Modelos fundamentais

- Modelo de interação:
  - Interação é a ação (comunicação e sincronização) entre as partes para realizar um qualquer trabalho.
  - É afetada por dois aspetos:
    - Performance dos canais de comunicação;
    - Inexistência de um tempo global.

Modelos de Programação Distribuída

Modelos fundamentais

- Modelo de interação:
  - 1. Performance dos canais de comunicação:
    - 1.a) Latência
      - Intervalo de tempo que medeia entre o início da transmissão de uma mensagem por um processo e o início da sua recepção pelo outro processo.
      - Depende de:
        - Demora (“delay”) de transmissão pela rede
        - Tempo requerido pelo sistema operativo em ambos os lados da comunicação
        - Demora no acesso aos recursos da rede

Modelos de Programação Distribuída

Modelos fundamentais

- Modelo de interação:
  - 1. Performance dos canais de comunicação:
    - 1.b) Largura de banda (“bandwidth”)
      - Total de informação que pode ser transmitida pela rede num dado intervalo de tempo;
    - 1.c). Jitter
      - Variação no tempo necessário para enviar grupos de mensagens consecutivos constituintes de uma informação transmitida de um ponto para outro na rede.
      - Importante na transmissão de som e imagem.

Modelos de Programação Distribuída

Modelos fundamentais

- Modelo de interação:
  - 2. Inexistência de um tempo global:
    - Cada computador tem um relógio “clock” interno.
      - => Cada relógio tem um “drift” (um desvio) do tempo de referência
      - => Os “drifts” de dois relógios distintos são também distintos
      - (o que significa que entre eles o tempo será sempre divergente)
      - Uma solução passa por obter o tempo fornecido por GPS – Global Positioning System, e enviar aos participantes do sistema distribuído.
      - **Problema:** Delays no envio dessa mensagem!

Modelos de Programação Distribuída

Modelos fundamentais

- Modelo de interação:
  - Duas variantes no modelo de interacção:
    - 1 – Sistemas distribuídos síncronos
      - Sistemas onde podem existir limites máximos de tempo conhecidos para:
        - Tempos de execução dos processos,
        - Atrasos na comunicação,
        - Variações no tempo (relógio) de referência

Modelos de Programação Distribuída

Modelos fundamentais

- Modelo de interação:
  - Duas variantes no modelo de interacção:
    - 1 – Sistemas distribuídos <u>síncronos (cont)</u>
      - Se:
        - o tempo necessário para executar cada passo de um processo tem um limite inferior e um limite superior conhecidos
        - cada mensagem transmitida por um canal é recebida dentro de um limite de tempo conhecido
        - cada processo tem um relógio cujo desvio máximo para o tempo de referência é conhecido
      - Podem definir-se "timeouts" para detectar falhas.
      - Dificuldade em encontrar os limites para os tempos, mais difícil ainda, provar a sua correcção.

Modelos de Programação Distribuída

Modelos fundamentais

- Modelo de interação:
  - Duas variantes no modelo de interacção:
    - 1 – Sistemas distribuídos assíncronos
      - Não possui limites para:
        - Tempo de execução dos processos – cada passo de execução pode levar um tempo arbitrariamente longo
      - Tempo de transmissão de mensagens - uma mensagem pode chegar rapidamente ou demorar dias
      - O desvio para o tempo de referência pode ser qualquer

Modelos de Programação Distribuída

Modelos fundamentais

- Modelo de interação:
  - Duas variantes no modelo de interacção:
    - 1 – Sistemas distribuídos assíncronos (cont)
      - Exemplo de um sistema assíncrono: Internet;
      - Como lidar com longos tempos de espera:
        - O sistema pode avisar o utilizador que o tempo de espera pode ser longo e solicitar uma alternativa;
        - O sistema pode dar oportunidade ao utilizador para fazer outras coisas;
        - ...

Modelos de Programação Distribuída

Modelos fundamentais

- Modelo de interação:
  - Duas variantes no modelo de interacção:
    - O modelo de interacção e o problema da ordenação de eventos
      - Por vezes é importante conhecer a ordem pela qual ocorreu um conjunto de eventos.
      - Ex.lo
        - Sejam os utilizadores X,Y, Z e A que trocam mails para marcar uma reunião:
        - X envia uma mensagem, com o assunto: Meeting, para Y, Z e A
        - Y e Z respondem para os outros com o assunto: Re: Meeting

Modelos de Programação Distribuída

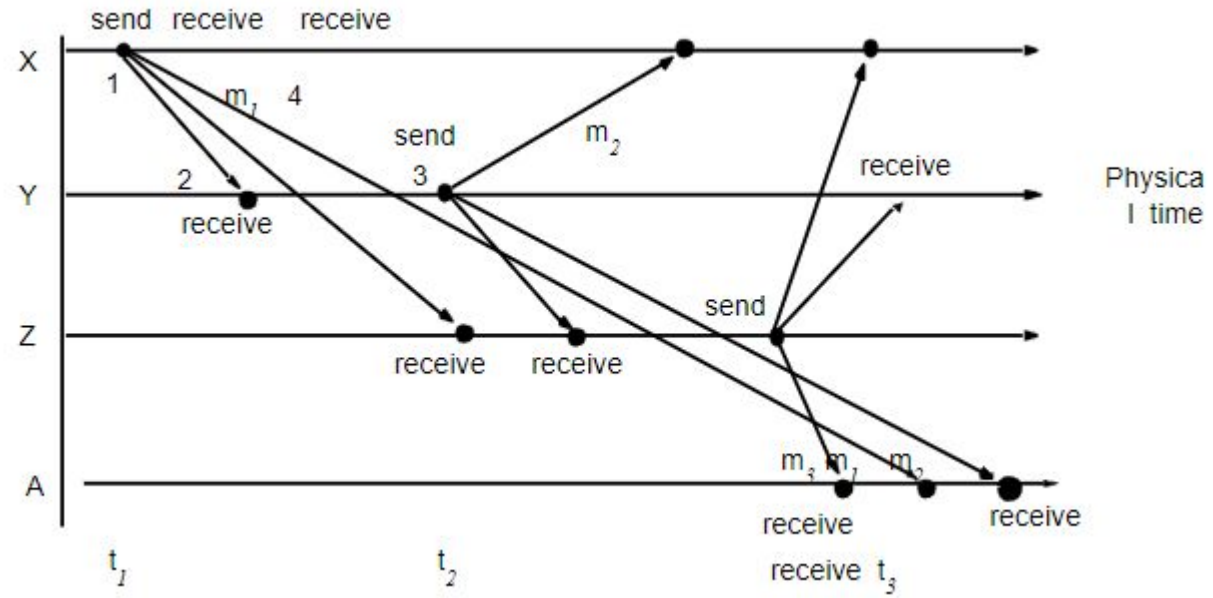
Modelos fundamentais

- Modelo de interação:
  - Duas variantes no modelo de interacção:
    - O modelo de interacção e o problema da ordenação de eventos
      - Uma vez que não há limites no tempo de comunicação as mensagens podem ser entregues de tal forma que o utilizador A receba as mensagens pela ordem:
      - De: Subject:
        Z Re:Meeting
        X Meeting
        Y Re:Meeting

Modelos de Programação Distribuída

Modelos fundamentais

- Modelo de interação:
  - Duas variantes no modelo de interacção:
    - O modelo de interacção e o problema da ordenação de eventos (cont)



Modelos de Programação Distribuída

Modelos fundamentais

- Modelo de interação:
  - Duas variantes no modelo de interacção:
    - O modelo de interacção e o problema da ordenação de eventos (cont)
      - Solução proposta por Lamport [1978]
        - Criar um tempo lógico para marcar a sequência de eventos e determinar a ordem correcta em que eles aparecem no tempo.

Modelos de Programação Distribuída

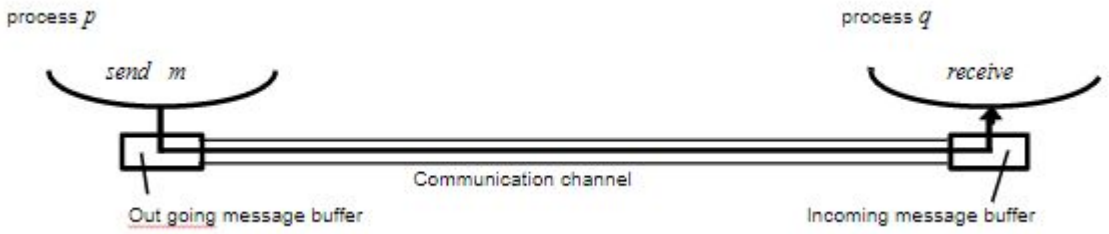
Modelos fundamentais

- Modelo de Avarias (Failure Model):
  - Uma avaria é qualquer alteração do comportamento do sistema em relação ao esperado (i.é, em relação à sua especificação).
  - Define de que maneira as avarias podem ocorrer
    - Podem atingir os processos ou os canais de comunicação
  - Tipos de Avarias:
    - Avarias por omissão;
    - Avarias arbitrárias;
    - Avarias em tempo.

Modelos de Programação Distribuída

Modelos fundamentais

- Modelo de Avarias (Failure Model):
  - Avarias por omissão:
    - quando um processo “deixa de funcionar” em algum ponto do sistema distribuído;
    - quando o canal de comunicação falha;



Modelos de Programação Distribuída

Modelos fundamentais

- Modelo de Avarias (Failure Model):
  - Tipos de Avarias por omissão:
    - Fail-stop – o processo bloqueou (crashed) e esse facto pôde ser detectado por outros processos.
    - Crash – o processo aparentemente bloqueou mas não é possível garantir que apenas deixou de responder por estar muito lento, ou porque as mensagens que enviou não chegaram.
    - Omission – uma mensagem colocada no buffer de emissão nunca chega ao buffer de recepção
      - (pode ocorrer por falta de espaço no buffer).
    - Send-omission – uma mensagem perde-se entre o emissor e o buffer de emissão
    - Receive-omission – uma mensagem perde-se entre o buffer de recepção e o receptor

Modelos de Programação Distribuída

Modelos fundamentais

- Modelo de Avarias (Failure Model):
  - Avarias arbitrárias (ou Bizantinas):
    - Qualquer tipo de erro pode acontecer:
    - Nos processos:
      - O processo não responde;
      - O estado do processo é corrompido;
      - Responde de forma errada;
      - Responde fora de tempo.

Modelos de Programação Distribuída

Modelos fundamentais

- Modelo de Avarias (Failure Model):
  - Avarias arbitrárias (ou Bizantinas) (cont):
    - Nos canais de comunicação:
      - mensagens corrompidas;
      - mensagens não entregues;
      - mensagens duplicadas;
      - mensagens inexistentes são entregues;
      - (São raras de ocorrer nos canais de comunicação porque o software de comunicação protege as mensagens com somas de verificação ("checksums"), números de sequenciamento, etc)

Modelos de Programação Distribuída

Class of failure	*Affects*	*Description*
Fail-stop	Process	Process halts and remains halted. Other processes may detect this state.
Crash	Process	Process halts and remains halted. Other processes may not be able to detect this state.
Omission	Channel	A message inserted in an outgoing message buffer never arrives at the other end's incoming message buffer.
Send-omission	Process	A process completes a *send*, but the message is not put in its outgoing message buffer.
Receive-omission	Process	A message is put in a process's incoming message buffer, but that process does not receive it.
Arbitrary (Byzantine)	Process or channel	Process/channel exhibits arbitrary behaviour: it may send/transmit arbitrary messages at arbitrary times, commit omissions; a process may stop or take an incorrect step.

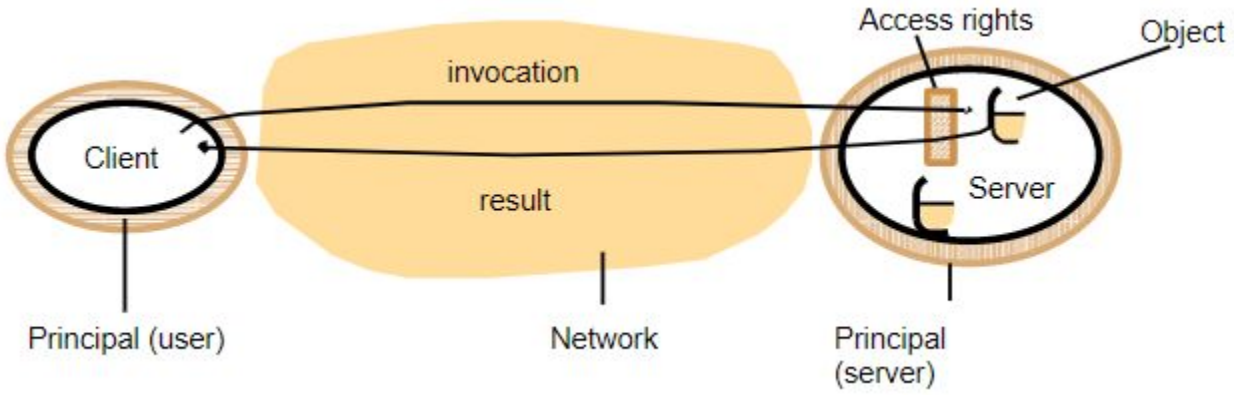
Class of Failure	*Affects*	*Description*
Clock	Process	Process's local clock exceeds the bounds on its rate of drift from real time.
Performance	Process	Process exceeds the bounds on the interval between two steps.
Performance	Channel	A message's transmission takes longer than the stated bound.

2 (parte 2) /23

Modelos de Programação Distribuída

Modelos fundamentais

- Modelo de Segurança:
  - Proteção das entidades do sistema, processo/utilizador (“principal”)
  - Direitos de acesso especificam que entidades podem aceder, e de que forma, a que recursos.
  - Ex. Que entidade pode executar que operações num dado objecto.



Modelos de Programação Distribuída

Modelos fundamentais

- Modelo de Segurança:
  - O servidor é responsável por verificar
    - a identidade de quem (entidade) fez o pedido, e verificar se essa entidade tem direitos de acesso para realizar a operação pretendida.
  - O cliente deverá verificar
    - a identidade de quem lhe enviou a resposta, para ver se a resposta veio da entidade esperada.

Modelos de Programação Distribuída

Modelos fundamentais

- Modelo de Segurança:
  - Ameaças:
    - Supondo que existe um processo inimigo (adversário) capaz de:
      - enviar qualquer mensagem para qualquer processo
      - interceptar (ler/copiar) qualquer mensagem trocada entre 2 processos
    -
    - Classificação das ameaças:
      - aos processos
      - à comunicação
      - negação de serviço

Modelos de Programação Distribuída

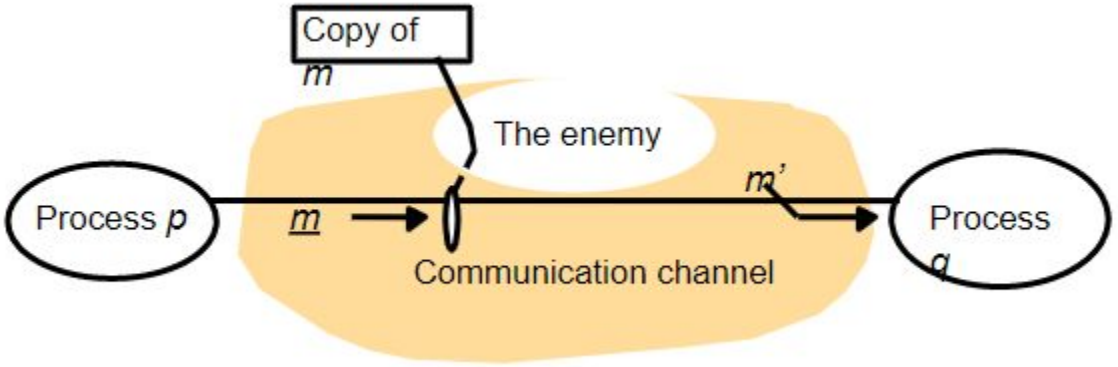
Modelos fundamentais

- Modelo de Segurança:
  - Ameaças:
    - i) Ataques a processos
      - Ao projectar um servidor, ter consciência de que:
        - Os protocolos de rede não oferecem protecção para que o servidor saiba a identidade do emissor
          - IP inclui o endereço do computador origem da mensagem mas um processo inimigo pode forjar esse endereço
        - Um cliente também não dispõe de métodos para validar as respostas de um servidor

Modelos de Programação Distribuída

Modelos fundamentais

- Modelo de Segurança:
  - Ameaças:
    - i) Ataques a processos
      - Em ambos os casos um processo inimigo pode fazer-se passar pela entidade (cliente /servidor) e enviar a mensagem solicitada



Modelos de Programação Distribuída

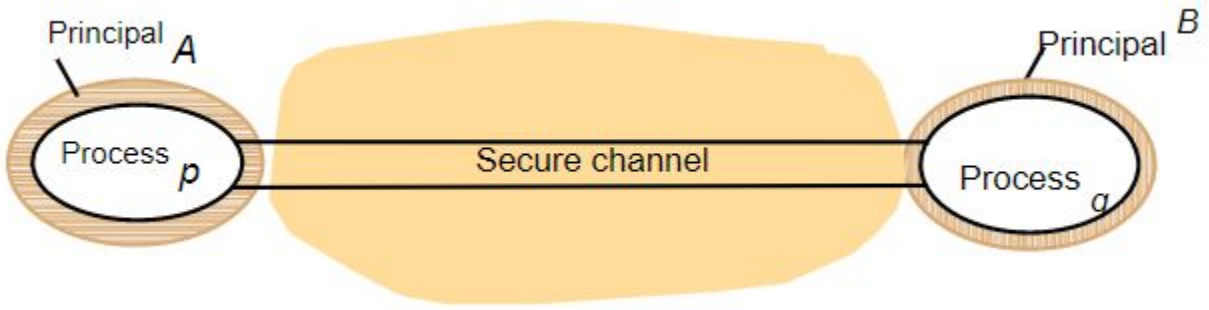
Modelos fundamentais

- Modelo de Segurança:
  - Ameaças:
    - ii) Ataques a canais de comunicação
      - Um processo inimigo pode copiar, alterar ou injetar mensagens na rede
      - A comunicação pode ser violada por processos que observam a rede à procura de mensagens significativas
      - (essas mensagens podem posteriormente ser reveladas a terceiros)

Modelos de Programação Distribuída

Modelos fundamentais

- Modelo de Segurança:
  - Ameaças:
    - iii) Negação de serviço
      - Um processo intruso captura uma mensagem de solicitação de serviço e retransmite-a inúmeras vezes ao destinatário, fazendo-o executar sistematicamente o mesmo serviço e ultrapassando a sua capacidade de resposta
      - Como lidar com estas ameaças? - utilização de canais seguros



Modelos de Programação Distribuída

Modelos fundamentais

- Modelo de Segurança:
  - Definição de canal seguro:
    - Canal utilizado para comunicação entre dois processos com as seguintes características:
      - Cada processo pode identificar com 100% de confiança a entidade
      - responsável pela execução do outro processo;
      - As mensagens que são transferidas de um processo para outro são garantidas do ponto de vista da integridade e da privacidade;
      - As mensagens têm garantia de não repetibilidade ou reenvio por ordem distinta
      - (cada mensagem inclui um tempo físico ou lógico);

Modelos de Programação Distribuída

Modelos fundamentais

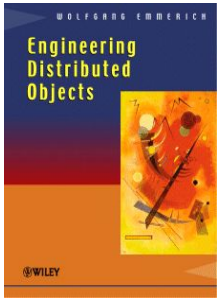
- Modelo de Segurança:
  - Criptografia:
    - Técnica de codificar o conteúdo de uma mensagem de forma a "esconder" o seu conteúdo.
    - É necessário que ambos os processos possuam a chave de codificação/descodificação
  - Autenticação:
    - Incluir na mensagem uma porção (encriptada) que contenha informação suficiente para identificar a entidade e verificar os seus direitos de acesso

Modelos de Programação Distribuída

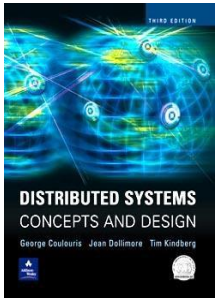
Modelos fundamentais

- Modelo de Segurança:
  - Criar um modelo de segurança:
    - Analisar as principais ameaças:
      - riscos envolvidos /possíveis consequências;
    - Fazer o balanço entre o custo de proteger o sistema e o risco que de facto as ameaças representam.

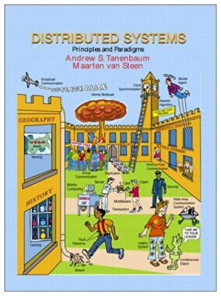
Bibliografia



From: Wolfgang Emmerich
Engineering Distributed Objects
John Wiley & Sons, Ltd 2000



From: Coulouris, Dollimore and Kindberg
Distributed Systems: Concepts and Design
Edition 4 © Addison-Wesley 2005



From: Andrew S., Tanembaum and Van Steen, Maarten
Distributed Systems: Principles and Paradigms
Edition 2 © Pearson 2013

Questões?