SISTEMAS DISTRIBUÍDOS

Capítulo 3 - Comunicação em Sistemas Distribuídos – Comunicação Ponto a Ponto

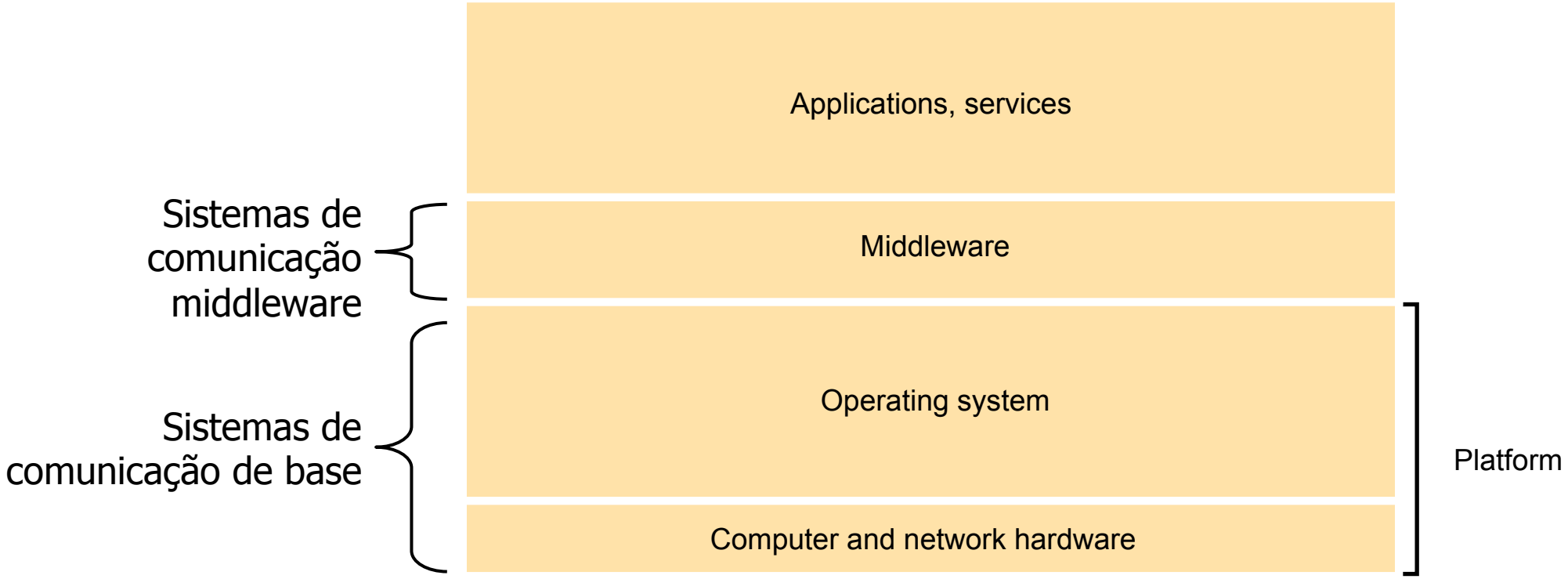
Parte 1

Nota prévia

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

G. Coulouris, J. Dollimore and T. Kindberg,
Distributed Systems - Concepts and Design,
Addison-Wesley, 4th Edition, 2005

COMUNICAÇÃO NUM SISTEMA DISTRIBUÍDOS



SISTEMAS DE COMUNICAÇÃO DE BASE

Os sistemas de operação suportam a comunicação de dados entre os diferentes computadores envolvidos num sistema distribuído

- Interfaces geralmente de baixo nível
- Protocolo mais popular: TPC/IP
  - Oferece os seguintes protocolos de comunicação base:
  - **TCP – *streams***
    - Dados transmitidos como fluxo contínuo.
    - Dados chegam de forma fiável a menos que o stream seja quebrado.
  - **UDP – datagrama**
    - Comunicação por mensagens.
    - Mensagens podem-se perder, duplicar e chegar fora de ordem.
  - **IP multicast**
    - Comunicação por mensagens com múltiplos receptores.
    - Semântica semelhante ao UDP.

COMUNICAÇÃO NO NÍVEL MIDDLEWARE

Implementa sistema de comunicação recorrendo às primitvas de comunicação base

Fornece propriedades adicionais, atrasando a entrega das mensagens

- Def: Entrega de um mensagem num sistema de comunicação representa a accção do sistema disponibilizar a mensagem para ser lida pelas aplicações

FACETAS DA COMUNICAÇÃO PONTO-A-PONTO

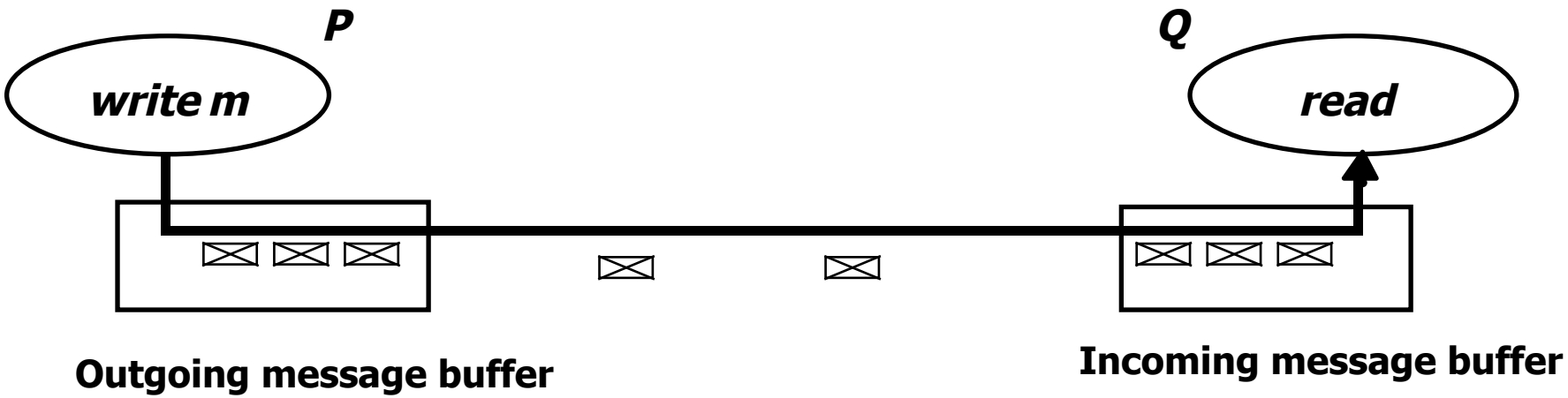
- Forma da interacção
  - Streams
  - Mensagens
    - Ordenação das mensagens
- Número de destinatários
  - Ponto-a-ponto
  - Multi-ponto (estudado mais tarde)
- Direcção de interacção
  - Uni-direccional
  - Bi-direccional
- Tipo de sincronização
  - Comunicação síncrona
  - Comunicação assíncrona
- Persistência
  - Comunicação persistente
  - Comunicação volátil
- Fiabilidade (modelo de falhas)

Forma de Interacção: streams

Emissor e receptor estabelecem um fluxo contínuo de dados

- Ordem dos dados enviados é mantida;
- Dimensão (fronteira) dos dados/mensagens não é mantida.

Exemplos de situações em que é apropriado?

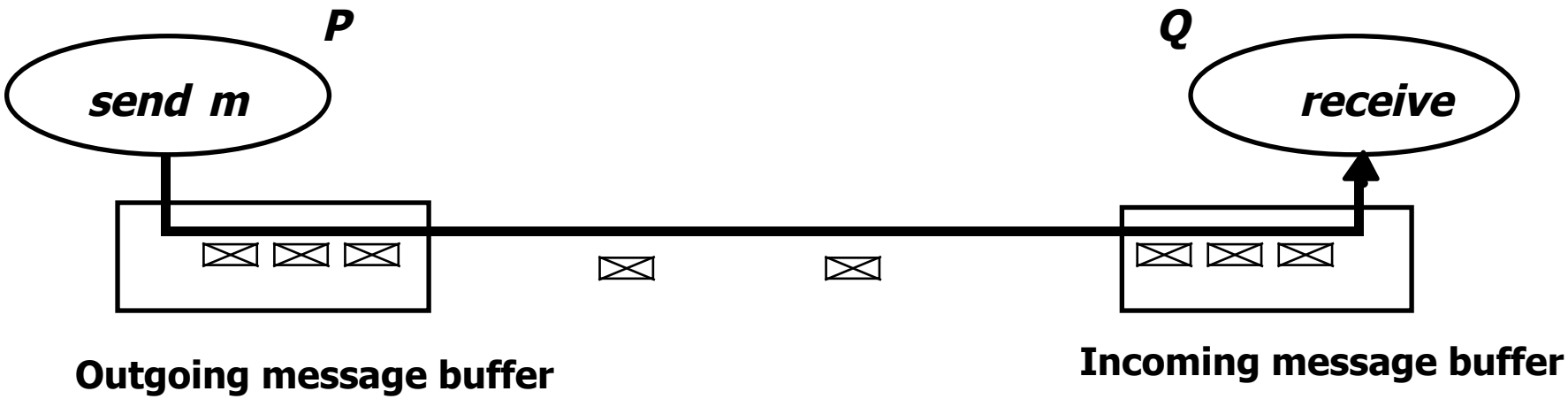


FORMA DE INTERACÇÃO: MENSAGENS

Emissor e receptor comunicam trocando mensagens

Cada mensagem tem um limite (e dimensão) bem-definida.

Exemplos de situações em que é apropriado?



Forma de Interação: ordenação das mensagens

Sem garantias de ordem

Sistema não garante que as mensagens são entregues pela ordem que foram enviadas

Entrega pela mesma ordem da emissão – FIFO (first in first out)

Sistema garante que as mensagens dum emissor são entregues pela mesma ordem que foram enviadas. Como implementar em TCP/UDP?

Número de destinatários

Comunicação ponto-a-ponto

Comunicação entre um emissor e um receptor

Comunicação multi-ponto

Comunicação entre um emissor e um conjunto de receptores
(detalhado na segunda parte deste capítulo)

DIRECÇÃO DE INTERACÇÃO

Comunicação uni-direccional:

Comunicação apenas num sentido: emissor->receptor

Comunicação bi-direccional:

Comunicação nos dois sentidos

SINCRONIZAÇÃO

Comunicação assíncrona:

- o emissor só fica bloqueado até o seu pedido de envio ser tomado em consideração
- o receptor fica bloqueado até ser possível receber dados
  - Em geral, o sistema de comunicação do receptor armazena (algumas) mensagens caso não exista nenhum receptor bloqueado no momento da sua recepção. Assim, funciona como um *buffer* entre o emissor e o receptor
  - É possível variante em que o receptor não fica bloqueado e devolve erro ou a recepção é efectuada em background

Comunicação síncrona:

- o emissor fica bloqueado até:
  - o receptor "receber" os dados – comunicação síncrona unidireccional
  - receber a resposta do receptor – comunicação pedido / resposta ou cliente / servidor
- o receptor fica bloqueado até ser possível consumir dados

PERSISTÊNCIA

Comunicação volátil: mensagens apenas são encaminhadas se o receptor existir e estiver a executar, caso contrário são destruídas.

Exemplo: ???

Comunicação persistente: mensagens são guardadas pelo sistema de comunicação até serem consumidas pelos destinatários, que podem não estar a executar. Mensagens são guardadas num receptáculo independente do receptor – mailbox, canal, porta persistente, etc.

Exemplo: ???

FIABILIDADE

Comunicação fiável: o sistema garante a entrega das mensagens em caso de falha temporária. Como implementar?

Comunicação não-fiável: em caso de falha, as mensagens podem-se perder

SISTEMAS DE COMUNICAÇÃO MIDDLEWARE

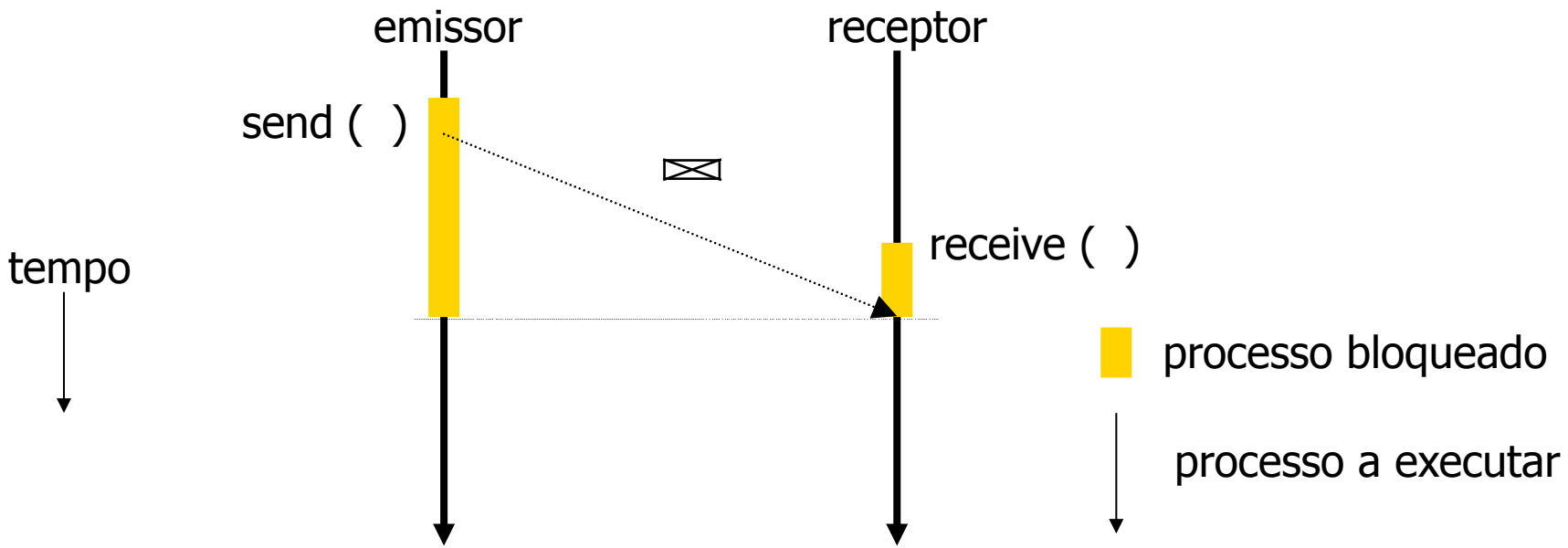
Podem ter diferentes modelos

- Simples por mensagens
- Simples por streams
- Pedido/resposta
- Baseado no paradigma de código móvel

Fornecem diferentes propriedades relativas às facetas abordadas

- Forma da interacção
- Número de destinatários
- Direcção de interacção
- Tipo de sincronização
- Persistência
- Fiabilidade (modelo de falhas)

COMUNICAÇÃO SÍNCRONA UNIDIRECCIONAL



O emissor fica bloqueado à espera de o receptor estar disposto a receber a mensagem

Implementação exige envio de mensagem com informação de recepção para o emissor

COMUNICAÇÃO SÍNCRONA UNIDIRECCIONAL

Concorrência:

Limita a concorrência, porque o emissor se bloqueia até o receptor receber a mensagem enviada.

Sincronização:

Após o envio o emissor sabe que o receptor acabou de receber a mensagem;

Após a recepção, o receptor sabe que o emissor esteve bloqueado até esse momento.

Ordenação:

Em geral, garante a ordem das mensagens do mesmo emissor.

Modelo de falhas:

Em caso de sucesso o emissor sabe que o receptor recebeu a mensagem.

Em caso de insucesso não se sabe exactamente o que se passou (problemas de rede ou do receptor ?).

Variações: é possível o emissor só ficar bloqueado até a mensagem chegar ao site do receptor mesmo que este não a consuma logo.

Message-Oriented Middleware (MOM) ou Message-Queuing systems

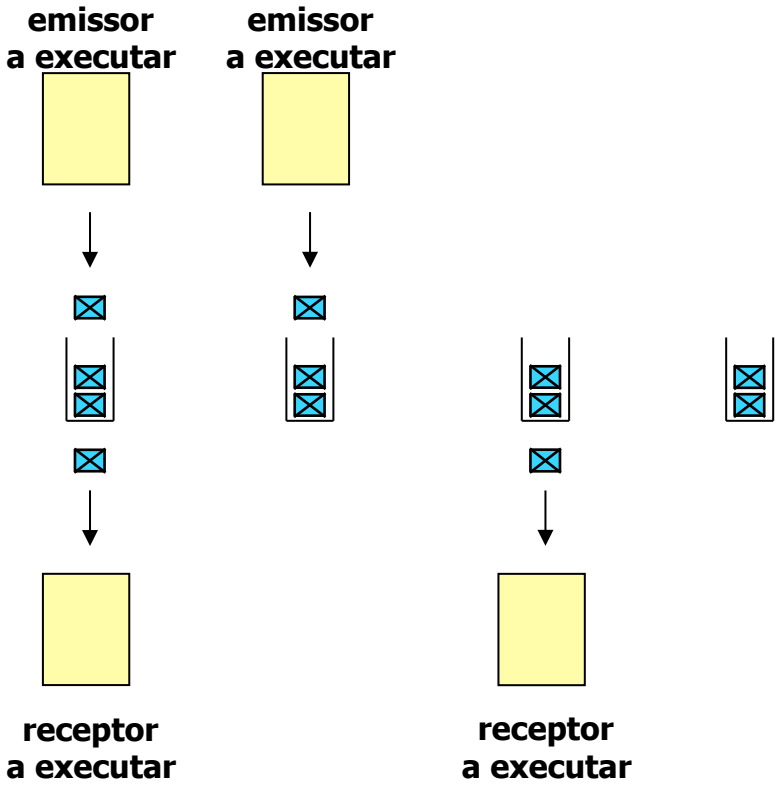
Os processos comunicam pela troca de mensagens usando um subsistema intermédio que assegura a persistência através de filas de mensagens (queues)

- Comunicação assíncrona
- Suporta transferência de mensagens que duram vários minutos

Uma mensagem é sempre endereçada por um processo para uma queue e só pode ser consumida da queue por um outro processo / aplicação (uma queue pode, no entanto, ser partilhada por vários processos / aplicações)

- O emissor apenas tem garantias que a mensagem foi entregue na queue

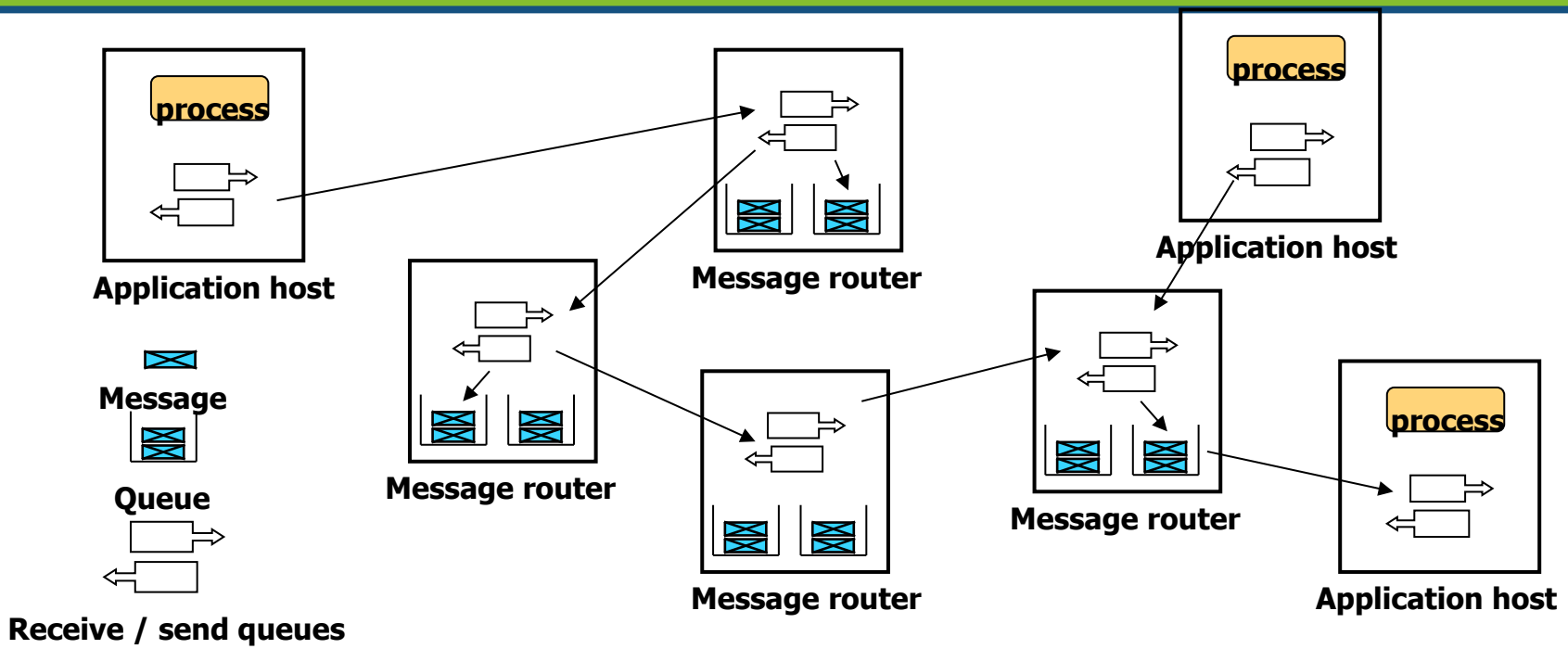
INTERACÇÃO EMISSOR / RECEPTOR



Primitivas:

- Put – adiciona uma mensagem a uma *queue*
- Get – bloqueia até que a *queue* não esteja vazia e remove a primeira mensagem
- Poll – verificar se existem mensagens na *queue*
- Notify – Instalar um *handler* para ser notificado se uma mensagem chegar

Estrutura geral de um MOM



Mensagens podem ser propagadas para o destino através de filas intermédias

- Algumas máquinas podem ter filas e actuar apenas como *routers*
  - Permite implementar funcionalidade adicional – ex.: log para segurança ou tolerância a falhas, conversão de formato de mensagens/gateway (message brokers)

Aplicações: e-mail, workflow, processamento em batch, systemas multidatabase (MQSeries)

PARA SABER MAIS

George Coulouris, Jean Dollimore, Tim Kindberg and Gordon Blair,
Distributed Systems – Concepts and Design,
Addison-Wesley, 5th Edition, 2011

Capítulo 4.1-4.3.

Heterogeneidade na representação dos dados

Diferentes arquitecturas (processadores), sistemas e linguagens podem ter:

- diferentes representações de números reais
- diferente ordem na representação de inteiros (big-endian, little-endian)
- diferentes representações de caracteres (ASCII, Unicode, EBCDIC)
- diferentes definições do que é um inteiro (2, 4, 8 bytes?), real

Problema: num sistema distribuídos, como garantir que dois processos a funcionar em diferentes arquitecturas, sistemas ou linguagens conseguem trocar dados?

- Definir formato no qual os dados devem passar na rede
  - Formato do emissor, formato do receptor, formato independente
- *Marshalling*: codificação do formato interno para o formato rede
- *Unmarshalling*: descodificação do formato rede para o formato interno

Organização do capítulo

Níveis dos sistemas de comunicação

Formas de comunicação ao nível do middleware e sua caracterização

- Comunicação cliente/servidor

Heterogeneidade e representação dos dados

COMUNICAÇÃO NO NÍVEL MIDDLEWARE

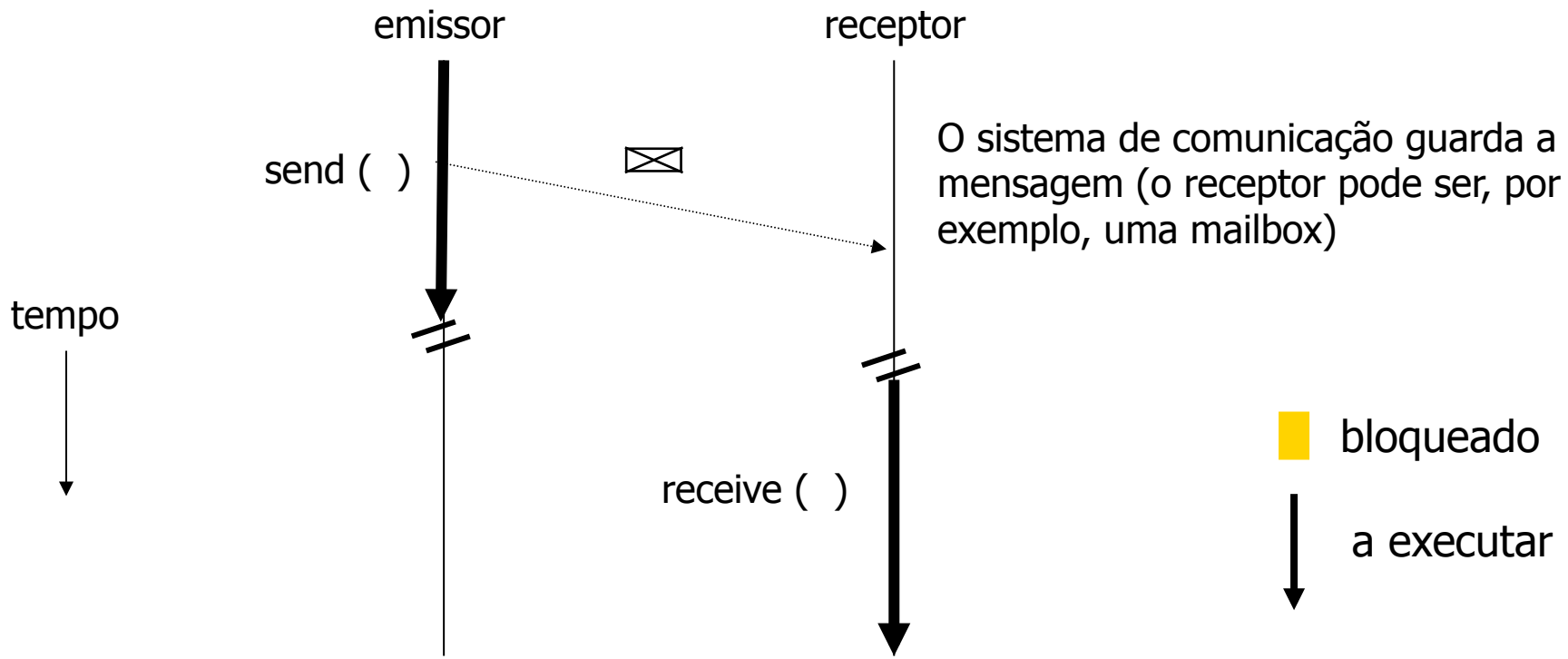
- Modelo de comunicação simples
  - Comunicação por streams (fluxos)
    - 1 ou N destinatários;
    - Uni-direccional ou bi-direccional.
  - Comunicação por mensagens.
    - 1 ou N destinatários;
    - Fiabilidade no envio de mensagens ou não;
    - Diferentes garantias de ordenação na propgação das mensagens.
- Comunicação baseada no paradigma pedido / resposta.
  - Invocação de métodos/procedimentos remotos (RMI / RPC)
- Comunicação baseada no paradigma do código móvel ("agentes")

SISTEMAS DE COMUNCAÇÃO

Alguns exemplos

- Streams assíncronos (TCP)
- Comunicação assíncrona por mensagens (UDP)
- Comunicação síncrona unidireccional
- Comunicação síncrona pedido / resposta
- Comunicação persistente síncrona
- Comunicação persistente assíncrona
- Message-Oriented Middleware (MOM)

COMUNICAÇÃO PERSISTENTE ASSÍNCRONA



COMUNICAÇÃO PERSISTENTE ASSÍNCRONA

Concorrência:

Emissor nunca se bloqueia. Receptor pode bloquear-se na recepção de mensagens.

Sincronização:

Não existe sincronização entre emissor e receptor.

Ordenação:

Geralmente garante a ordem das mensagens do mesmo emissor.

Modelo de falhas:

Emissor não tem garantia que mensagem foi armazenada. Se for armazenada, em geral, o sistema garante que a mensagem é guardada até o receptor a consumir.

Não existem garantias que algum receptor a consuma.