Computação Distribuída - Introdução

Licenciatura em Engenharia Informática

Universidade Lusófona

Prof. Paulo Guedes

Prof. José Rogado

Prof. José Faísca

Programa da Cadeira (I)

1. Introdução
   - Caracterização dos Sistemas Distribuídos (SD)
   - Requisitos, arquitectura e funcionalidades
   - Exemplos de SDs
2. Arquitecturas e Modelos de Comunicação Distribuída
   - Arquitecturas de sistemas
   - Níveis de software: papel do Middleware
   - Interfaces e objectos
   - Modelos de interacção, de falhas e de segurança
3. Comunicação entre Processos Distribuídos
   - Características da comunicação entre processos
   - Invocação, parâmetros e heterogeneidade de dados
   - Técnicas de representação de dados e serialização
   - Comunicação cliente / servidor e multi-ponto

Programa da Cadeira (II)

4. Invocação Remota e Objectos Distribuídos
   - RPC: Modelo de execução
   - Linguagem de definição de interfaces e passagem de parâmetros.
   - Registo e descoberta de interfaces
   - Plataforma de execução: Sun RPC
   - Objectos Distribuídos: Modelo de Execução
   - Plataformas de Execução (Java RMI, Corba)
   - Nomeação e Serviços de Directório
5. Sistemas de Ficheiros Distribuídos
   - Problemática e caracterização
   - Arquitecturas de SGF distribuídos
   - Problemática do caching: performance e consistência
   - Implementações: NFS e AFS

Programa da Cadeira (III)

6. Arquitectura Orientada aos Serviços (SOA)
   - Modelo de Execução Web Services (WS)
   - O protocolo SOAP
   - Integração com Java
   - WS e CORBA
   - Linguagens de definição de Serviços (WSDL)
   - Serviço de Directório de WS
7. Contextos de Segurança Distribuídos
   - Canais de comunicação seguros
   - Sistemas de autenticação centralizados, distribuídos e federativos
   - Assinatura e Encriptação em XML.
   - SAML e Web Services Security

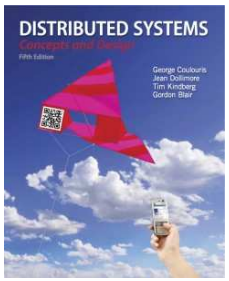
Metodologia de Trabalho

- Bolonha: as Aulas Teóricas duram 1h30 !
  - O professor tem mais um papel de orientador
    - Apresentação de temas e orientação do trabalho
  - Os alunos têm de realizar mais trabalho individual
    - Pesquisa e apresentação de temas da cadeira
- Aulas Laboratoriais – valorização pessoal
  - Serão realizados os projectos da cadeira
    - Pelo menos dois projectos
  - Propostos a partir da 2ª semana de aulas
  - Preparam para o trabalho final de curso

Avaliação da Cadeira

- Componentes da Avaliação
  - A Assiduidade é Fundamental
    - Regulamento obriga a assistir a 75% das aulas
  - Prática 50%
    - Mínimo de 2 trabalhos com entrega faseada
  - Teórica 50%
    - Frequência Final
  - Notas inferiores a **8** (sem arredondamento!) são eliminatórias
- Aprovação na Cadeira: nota final de 10 valores
  - As notas parcelares **não** são arredondadas
    - Notas dos trabalhos, nota prática, nota teórica
  - Só a nota final é arredondada
    - Arredondamento só é feito depois de calculada a média

Bibliografia



- "Distributed Systems: Concepts and Design" (5th Edition), by Coulouris, Dollimore & Kindberg, Ed. Addison-Wiley, 2011 ISBN 0132143011
  - Livro muito completo sobre Sistemas Distribuídos
  - São seguidos os capítulos 1, 2, 4, 5, parte do 6 e 8 (4ª edição)
  - Restantes podem ser tema de trabalho individual ou abordados no 2º ciclo
- "Distributed Systems: Principles and Paradigms" (2nd Edition) by Tanenbaum & van Steen, Prentice Hall, 2006
  - Mais orientado para os Sistemas Operativos distribuídos
  - Constitui uma boa referência sobre o tema
- "Web Services: Principles and Technology", by Michael Papazoglou, Prentice Hall, 2007
  - Mais orientado para os Web Services, mas muito completo
- "Tecnologia dos Sistema Distribuídos", by J. Marques e P. Guedes; FCA Editora; Maio 1998
  - Livro em língua portuguesa, já um pouco desactualizado

Bibliografia Complementar

- Outras referências para as aulas práticas e projectos
  - Fornecidas nas aulas específicas e nos enunciados dos trabalhos
  - Site Netlab
    - http://netlab.ulusofona.pt/cd
- Tutoriais Importantes
  - SUN RPC
    - http://docs.sun.com/app/docs/doc/816-1435/oncintro-fig-1?a=view
  - Java RMI
    - http://java.sun.com/docs/books/tutorial/rmi/index.html
  - JAX-WS Web Services
    - http://www.netbeans.org/kb/docs/websvc/jax-ws.html
  - Etc…

Introdução

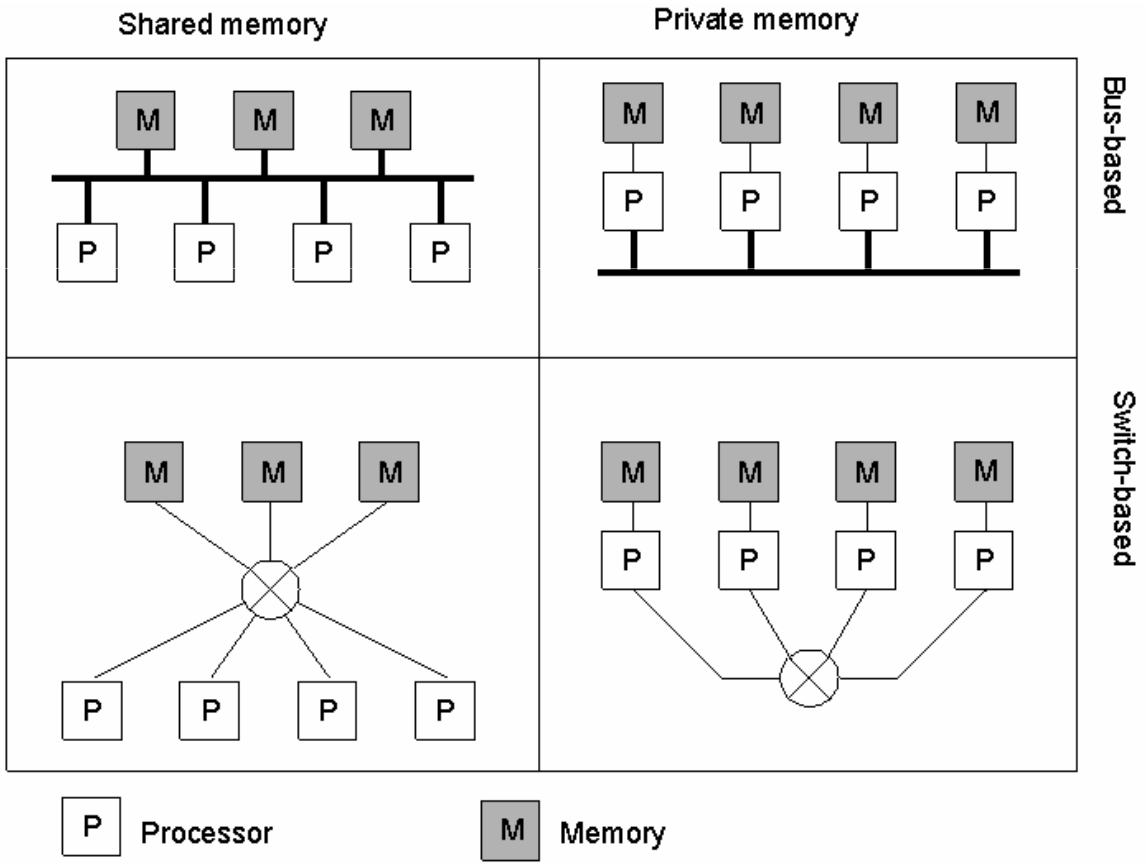
- Caracterização dos Sistemas Distribuídos
- Requisitos, arquitectura e funcionalidades
- Exemplos de SDs

Definição de Sistema Distribuído

- Um **Sistema Distribuído** é constituído por um conjunto de componentes de software e hardware ligados por uma infra-estrutura de comunicação
  - Bus, malha, rede, etc.
- Os componentes cooperam e coordenam-se para realizar um **objectivo comum**
  - Sistema Operativo Distribuído
  - Aplicação Distribuída

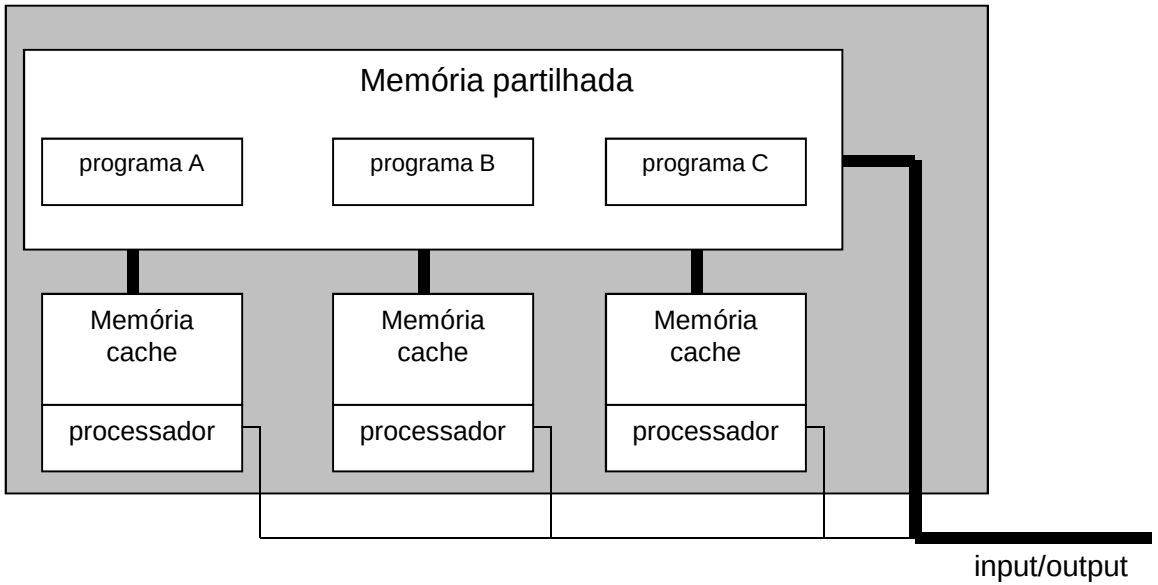
Suporte da Distribuição: Hardware

- Existem várias topologias de interligação dos componentes
  - Limiar entre sistemas paralelos e sistemas distribuídos



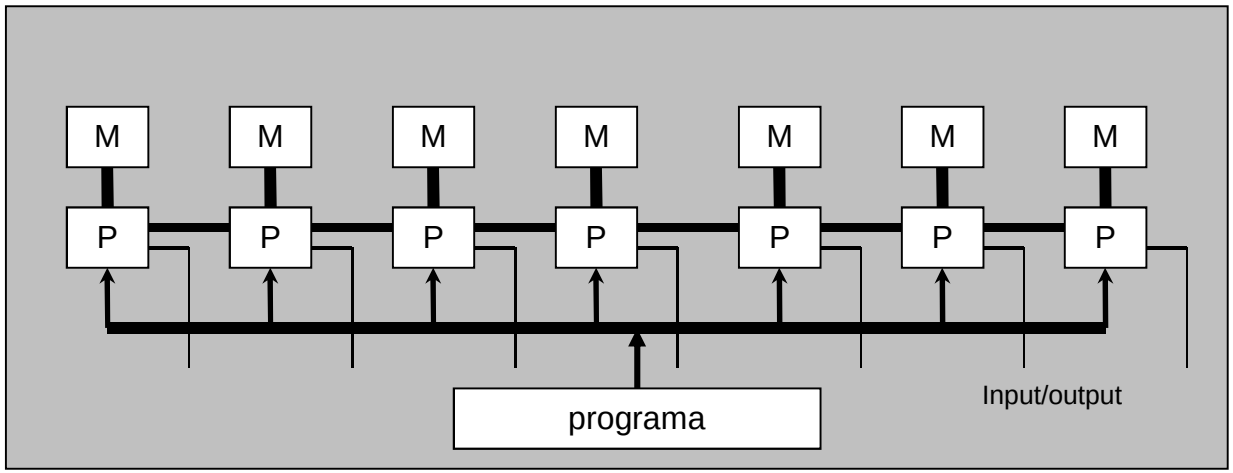
Processadores com Memória Partilhada

- Multiprocessador
  - Executa uma única instância do SO residente na MC
  - A distribuição (paralelização) é gerida pelo SO ao nível da linguagem de programação com suporte do hardware
    - threads, locks, semáforos
    - instruções indivisíveis



Processadores com Memória Privada

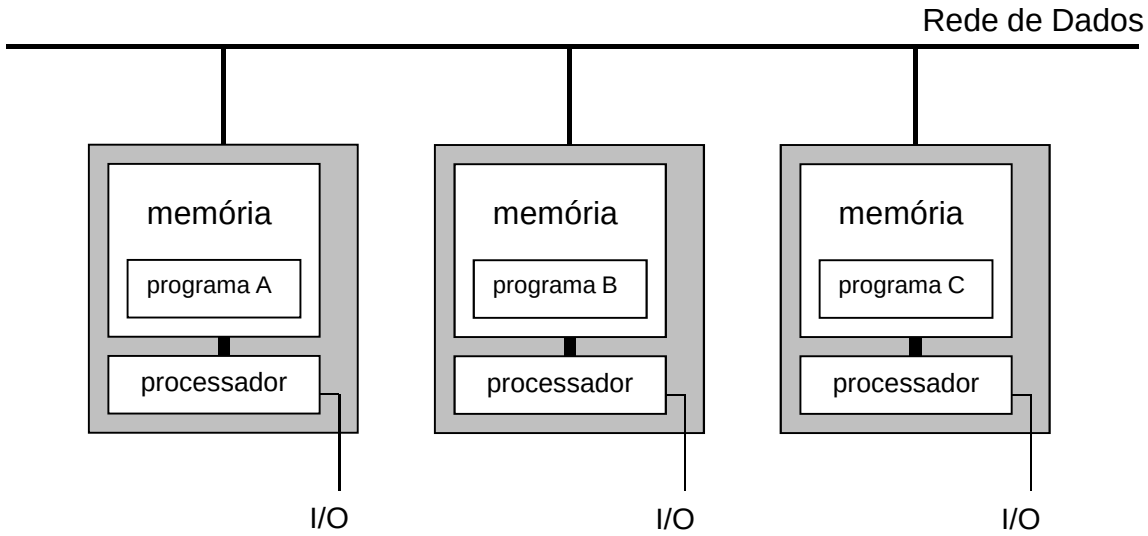
- Multicomputador
  - Cada processador executa uma instância distinta do SO mas oferece uma imagem única às aplicações
  - Os programas podem ser executados em paralelo por todos os processadores
  - A sincronização é gerida ao nível do Hardware e de cada SO
    - Mecanismos de sincronização específicos (encadeamento de instruções,…)
    - *Single System Image*



M – memória
P – processador

Processadores em Rede

- Processadores desacoplados
  - Cada processador executa uma instância distinta do SO, podendo ou não oferecer uma imagem de sistema único às aplicações
  - A distribuição é gerida exclusivamente por software
    - Tradicionalmente gerida pelo SO
    - Tem vindo a evoluir para o software
  - Tema da cadeira de CD



Evolução dos Sistemas Distribuídos (i)

- Anos 70 – 80
  - Sistemas Operativos
    - VMS da DEC
    - AOS/VS da Data General
    - Unix (ATT, Berkeley, OSF)
  - Primeiros Sistemas Operativos Distribuídos
    - Amoeba
    - Mach3.0
    - Chorus
- A distribuição é suportada a nível do sistema operativo
  - Mecanismos complexos
  - Difícil evolução e suporte de novas plataformas
  - Falta de heterogeneidade
- A utilização da informática de forma generalizada impõe:
  - Simplificação e uniformização da interface com o sistema
  - Extrair o máximo rendimento da infra-estrutura hardware

Evolução dos Sistemas Distribuídos (ii)

- Anos 90 – 00
  - Enorme evolução tecnológica das TIC
    - Redes de Computadores: aumento da largura de banda
    - Capacidade de processamento: PCs, arquitecturas multiprocessador
  - Evolução dos Sistemas Operativos
    - MAC/OS, Unix, Windows retomam conceitos de grandes sistemas (VMS…)
  - Desenvolvimento da computação na Web
    - Sistemas abertos e ubíquos
  - Aparecimento de Plataformas de Middleware
    - PVM, MQ Series, Tuxedo, Corba, Java, WebServices
    - Webmethods, Vitria, Tibco
  - =► O suporte da distribuição migra para as plataformas
- A **computação** torna-se **distribuída**
  - Os SO continuam a fornecer funcionalidades essenciais

Razões para a Distribuição

- Melhoria do desempenho
  - Repartição do processamento por vários computadores
- Partilha de Recursos
  - CPU, armazenamento, periféricos
- Maior tolerância a falhas
  - Aumento do factor de redundância
- Maior modularidade
  - Cada sistema individual é gerido independentemente
- Acesso generalizado
  - Mesmos serviços acessíveis em todo o lado

Vantagens Efectivas dos SD

- Melhoria de desempenho
  - Execução efectivamente paralela de secções independentes de aplicações
  - São necessárias optimizações para tirar real partido da distribuição
- Maior disponibilidade
  - Aumento da fiabilidade através da redundância
- Melhor escalabilidade
  - Facilidade de adicionar elementos de forma incremental
- Melhor suporte da mobilidade
  - Independentes da localização geográfica
- Maior modularidade
  - A distribuição pressupõe a componentização das aplicações
- Custos inferiores
  - Preço e manutenção de servidores inferior ao de *mainframes*

Porém…

A distribuição introduz problemas de resolução não trivial

- Execução concorrente de tarefas dependentes
  - Necessidade de coordenar entre si os múltiplos fluxos de execução
- Modelo de falhas complexo
  - Falhas de componentes e/ou de comunicações ?
- Inexistência de uma base temporal comum
  - É impossível coordenar exactamente os relógios dos vários processadores
- Problemas de segurança
  - A característica distribuída dos sistemas aumenta a vulnerabilidade aos ataques
- Impossibilidade de determinar o estado global de um sistema
  - A comunicação por mensagens com tempos de propagação não desprezáveis impossibilita o conhecimento instantâneo e global do sistema
  - A resolução de falhas pode ser muito complexa ou impossível

Requisitos dos Sistemas Distribuídos

- No sentido de permitir uma melhor resolução dos problemas apontados, os SD devem satisfazer um conjunto de requisitos:
  1. Heterogeneidade
  2. Abertura
  3. Segurança
  4. Escalabilidade
  5. Gestão de falhas
  6. Concorrência
  7. Transparência

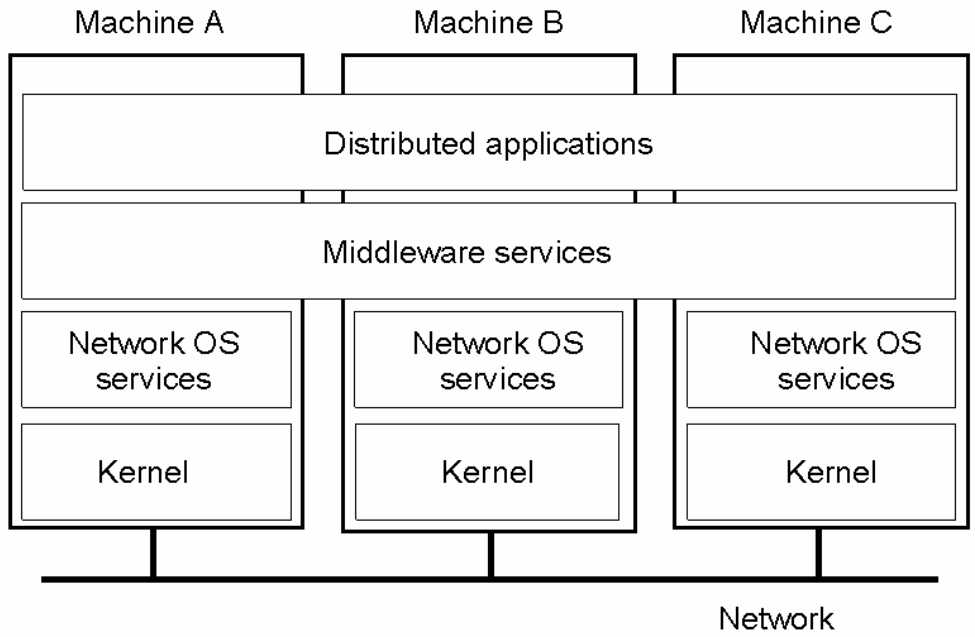
Heterogeneidade

Capacidade para suportar a variedade e a diferença

- Aplica-se a:
  - Redes e protocolos de comunicação
    - Enorme diversificação com a explosão das telecomunicações e Internet
  - Hardware
    - A diversidade dos dispositivos com capacidades de programação é ilimitada
  - Sistemas Operativos
    - Embora o Windows constitua a plataforma dominante, existem outsiders (OS/X, Linux, BSD, etc…) em número cada vez mais significativo
  - Linguagens de programação
  - Etc…
- É impossível um mesmo sistema suportar toda a diversidade fornecendo uma imagem única e integrada. Soluções:
  - *Middleware*
    - Camada intermédia de software que fornece um conjunto de serviços específicos e esconde a disparidade dos sistemas existentes
  - Máquina Virtual
    - Nível de abstracção que simula um processador uniforme que permite executar os mesmos programas em plataformas distintas =► mobilidade do código

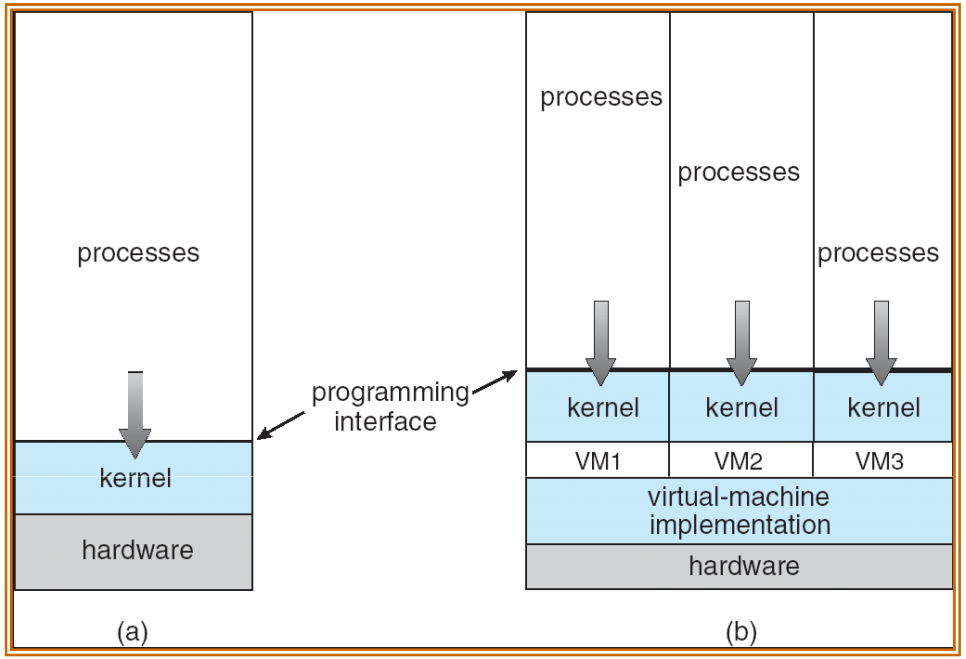
Posicionamento do Middleware

- A camada de *Middleware* é introduzida entre as aplicações e os sistemas operativos
  - Abstrai as interfaces de programação de sistemas diferentes
  - Fornece uma interface de programação homogénea
  - Permite a interoperabilidade entre componentes de aplicações distribuídas de forma independente da plataforma
    - Infra-estrutura: Corba, Java, WebServices, MPI, MQ Series, Tuxedo,
    - Integração: Webmethods, Vitria, Tibco



Máquinas Virtuais

- A noção de Máquina Virtual permite a execução das mesmas aplicações em sistemas diferentes
  - Cria um ambiente de execução idêntico e uniforme independente da plataforma
  - Pode ser orientado para uma linguagem (JVM), ou simular um sistema operativo completo (VMware, VirtualBox)
- Permite mobilidade de código, uma vez que uma aplicação pode migrar de um sistema para outro (*applets*, agentes)



(a) Sistema clássico (b) máquina virtual

Abertura

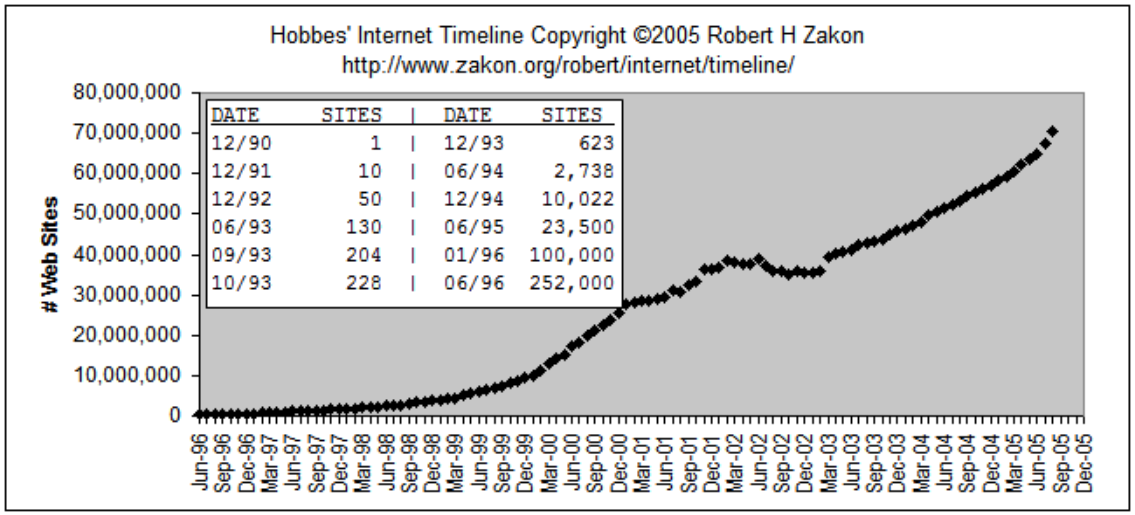
- Característica de um sistema que determina a sua capacidade de extensão e modificação
  - Num SD indica a possibilidade de adicionar novos serviços e de os tornar acessíveis às aplicações
- A abertura pressupõe a especificação e documentação das interfaces de programação
  - Publicação e/ou estandardização de APIs
    - Protocolos Internet alvo de RFCs da IETF, ISO e ITU
    - Interfaces de objectos especificadas pela OMG
- Sistemas abertos
  - Interfaces públicas e evolução controlada por organismos de normalização independentes
- Sistemas proprietários
  - Interfaces desconhecidas e evolução dependendo das leis do mercado e da estratégia comercial do fabricante

Segurança

- A natureza aberta dos Sistemas Distribuídos torna-os mais permeáveis a ataques
- A utilização de sistemas distribuídos para realizar tarefas com valor acrescentado sobre dados sensíveis implica a necessidade de protecção eficaz
- Os SD têm portanto de fornecer
  - Autenticação dos interlocutores
  - Confidencialidade e integridade dos dados enviados
  - Disponibilidade dos serviços e canais de comunicação
- Os requisitos de segurança mais difíceis de atingir:
  - Protecção contra ataques de negação de serviço (*Denial of Service*)
  - Certificação do código móvel na Internet

Escalabilidade

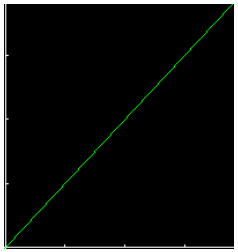
- A escalabilidade é a propriedade de um sistema que indica a sua capacidade em responder de forma linear a aumentos significativos da carga, número de utilizadores ou área de abrangência
- A escala de um SD pode ser muito variável
  - Pequena ou média escala no seio de uma Intranet
  - Larga escala quando abrange grandes áreas geográficas ou múltiplas organizações



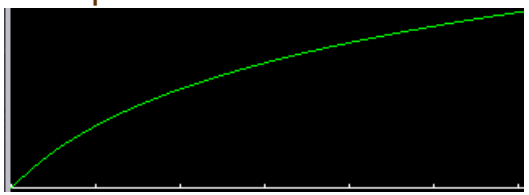
Evolução do factor de escala na Internet

Requisitos para a Escalabilidade

- Para garantir a escalabilidade de um SD
  - Limitar a necessidade do aumento do número de recursos
    - O aumento da dimensão do sistema deve traduzir-se num aumento linear do número de recursos necessários => *O(n)*
    - Ex: aumento de acessos a uma site Web vs. aumento de servidores necessários



  - Limitar a degradação de performance
    - O aumento das necessidades de computação não deve conduzir a uma degradação de desempenho => *O (log n)*
    - Ex: utilização de estruturas hierárquicas de consulta para reduzir os tempos de acesso (DNS, LDAP)



  - Evitar pontos de estrangulamento
    - Utilizar algoritmos descentralizados
    - Reduzir o número de mensagens através de caching e replicação
  - Evitar limitações na previsão do número de recursos
    - As definições básicas do sistema devem prever a escalabilidade
      - Ex: IPv4, bug ano 2000
- Garantir a escalabilidade é um dos desafios dominantes no desenvolvimento de um SD

Falhas

- Definições
  - **Falta (*Fault*)**: acontecimento que altera o funcionamento de um componente do sistema.
  - **Erro (*Error*)**: manifestação de uma falta no estado do sistema
  - **Falha (*Failure*)**: incapacidade do sistema em oferecer determinado serviço
- Propagação
  - Falta =► Erro =► Falha
- As causas podem ser múltiplas
  - Erros de transmissão ou de software
  - Avarias do material
- Num sistema distribuído as falhas não implicam uma completa paragem
  - Geralmente são parciais e independentes
  - Os efeitos podem propagar-se através de relações de dependência
- A capacidade de continuar a fornecer serviços em presença de falhas parciais é uma característica essencial de um SD =► disponibilidade do sistema
  - Gerir as falhas:
    - Detectar
    - Mascarar
    - Tolerar
    - Recuperar

Gestão de Falhas

- Detecção de falhas
  - Algumas falhas podem ser detectadas
    - Ex: mecanismos de checksum detectam erros de transmissão
  - Outras não são detectáveis
    - Ex: crash de um servidor ou congestão na rede de dados?
- Mascarar falhas
  - Depois de detectadas, **algumas** falhas podem ser mascaradas
    - Ex: retransmissão de pacotes, utilizar cópia local dos dados
- Tolerância a falhas
  - Capacidade de "tratar" um erro previamente identificado, continuando a oferecer parte do serviço
  - Especificação do comportamento na ocorrência de falhas
    - Ex: continuar com funcionalidade reduzida, utilizar redundância
- Recuperação de falhas
  - Envolve a capacidade de recuperar o estado do sistema antes da falha e reconstitui-lo (*roll-back*)
  - Mecanismos complexos que envolvem a noção de **transacção**

Falhas em Sistemas Distribuídos

- Papel da Falha *ou "como a distribuição pode piorar o seu problema ..."*
  - Falhas independentes ignoradas produzem:
    - Decréscimo da disponibilidade;
    - Decréscimo da fiabilidade;
    - Decréscimo da autonomia;
  - A detecção das falhas independentes ocorridas produz:
    - Aumento da disponibilidade;
    - Aumento da fiabilidade;
    - Aumento da autonomia;
  - Uma comunicação não fiável:
    - Não permite distinguir as situações de "em baixo" e de "desligado"
    - Origina a detecção de partições na comunicação.
    - Provoca um aumento do atraso na transmissão de mensagens.
    - Impõe que se assumam valores relativos ao número e duração das falhas e dos atrasos das mensagens.

Concorrência

- A existência de vários fluxos de execução implica acessos simultâneos a recursos partilhados
  - Ex: acessos de vários clientes a registo de dados
- Torna-se portanto necessário garantir a sincronização de certas operações para assegurar a coerência de dados
  - Em cada componente, as instâncias de execução de objectos concorrentes utilizam os mecanismos clássicos de sincronização
    - Semáforos, monitores, secções críticas
  - A nível global, são necessárias técnicas de sincronização específicas para algoritmos distribuídos

Transparência

- A transparência é definida como a capacidade de esconder do utilizador e das aplicações a natureza distribuída do sistema
  - O sistema aparece como um todo (*Single System Image*) e não como uma conjunto de componentes
- O projecto ANSA (*Advanced Networked Systems Architecture* - 1989), um dos primeiros consórcios de empresas a abordar a normalização da arquitectura dos sistemas distribuídos define os tipos de transparência
- O modelo RM-ODP da ISO-ITU (*Reference Model for Open Distributed Processing*) introduz os seguintes 8 tipos de transparência:
  1. Acesso
  2. Localização
  3. Concorrência
  4. Replicação
  5. Falha
  6. Mobilidade
  7. Desempenho
  8. Escalabilidade

Tipos de Transparência

Definições extraídas do Reference Manual do projecto ANSA (Advanced Network Systems Architecture), Cambridge, 1989

1. *Access transparency*: enables local and remote resources to be accessed using identical operations.
2. *Location transparency*: enables resources to be accessed without knowledge of their physical or network location (for example, which building or IP address).
3. *Concurrency transparency*: enables several processes to operate concurrently using shared resources without interference between them.
4. *Replication transparency*: enables multiple instances of resources to be used to increase reliability and performance without knowledge of the replicas by users or application programmers.
5. *Failure transparency*: enables the concealment of faults, allowing users and application programs to complete their tasks despite the failure of hardware or software components.
6. *Mobility transparency*: allows the movement of resources and clients within a system without affecting the operation of users or programs.
7. *Performance transparency*: allows the system to be reconfigured to improve performance as loads vary.
8. *Scaling transparency*: allows the system and applications to expand in scale without change to the system structure or the application algorithms.

Tipos de Transparência

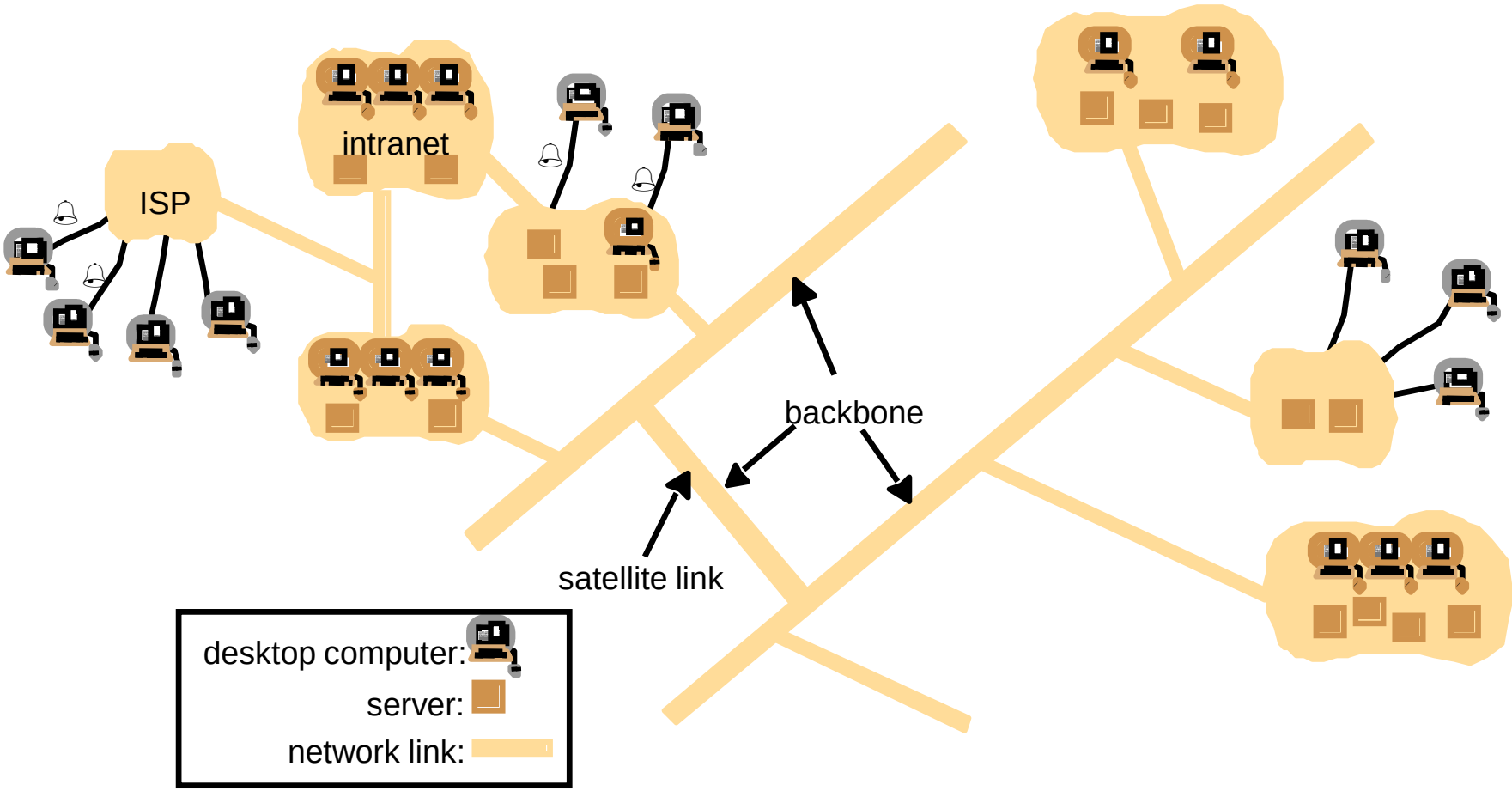
Definições extraídas do Reference Manual do projecto ANSA (Advanced Network Systems Architecture), Cambridge, 1989

1. *Transparência de Acesso*: Permite acessos locais e remotos através de operações idênticas.
2. *Transparência de Localização*: Permite acessos a recursos sem conhecimento da sua localização física ou do seu endereço de rede.
3. *Transparência de Concorrência*: Permite que vários processos funcionem simultaneamente sem interferências utilizando recursos partilhados.
4. *Transparência de Replicação*: Permite que múltiplas instâncias de recursos sejam utilizadas para aumentar a fiabilidade e o desempenho sem conhecimento dos utilizadores ou dos programadores das aplicações.
5. *Transparência às Falhas*: Permite o isolamento das falhas, fazendo com que os utilizadores e os programadores possam completar as suas tarefas em caso de mau funcionamento de componentes hardware ou software.
6. *Transparência de Mobilidade*: Permite a movimentação de recursos e componentes do sistema sem afectar o normal funcionamento dos programas e dos seus utilizadores.
7. *Transparência de Desempenho*: Permite que o sistema seja reconfigurado sem descontinuidade quando a sua carga varia.
8. *Transparência de Escalabilidade*: Permite que o sistema e as aplicações sejam expandidas sem modificação da sua arquitectura e dos seus algoritmos de funcionamento.

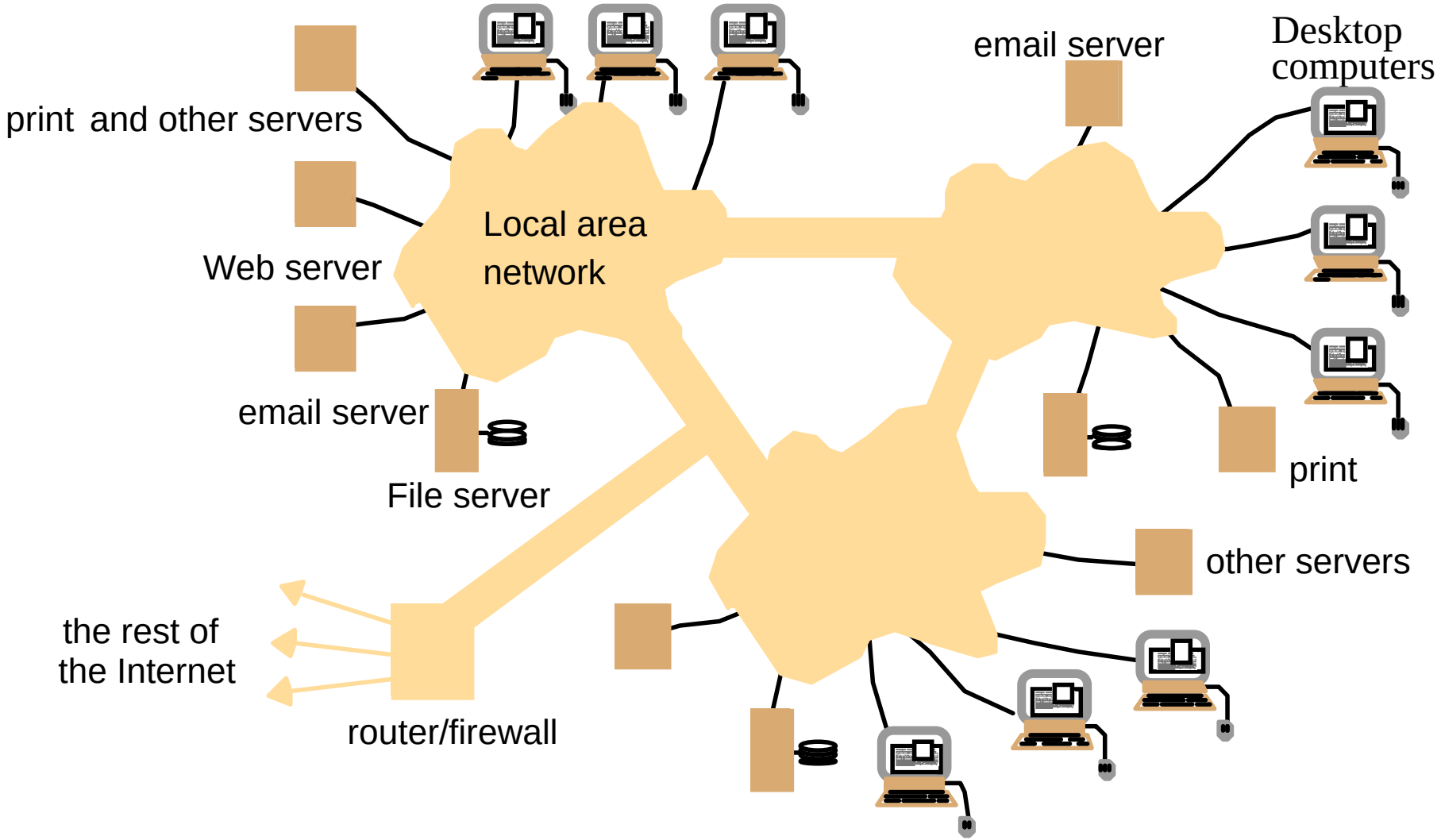
Trabalho Individual a realizar pelos alunos

- Ler o 1º capítulo do livro “Distributed Systems: Concepts and Design” (4th Edition), by Coulouris, Dollimore & Kindberg
- Fornecer exemplos de Sistemas Distribuídos

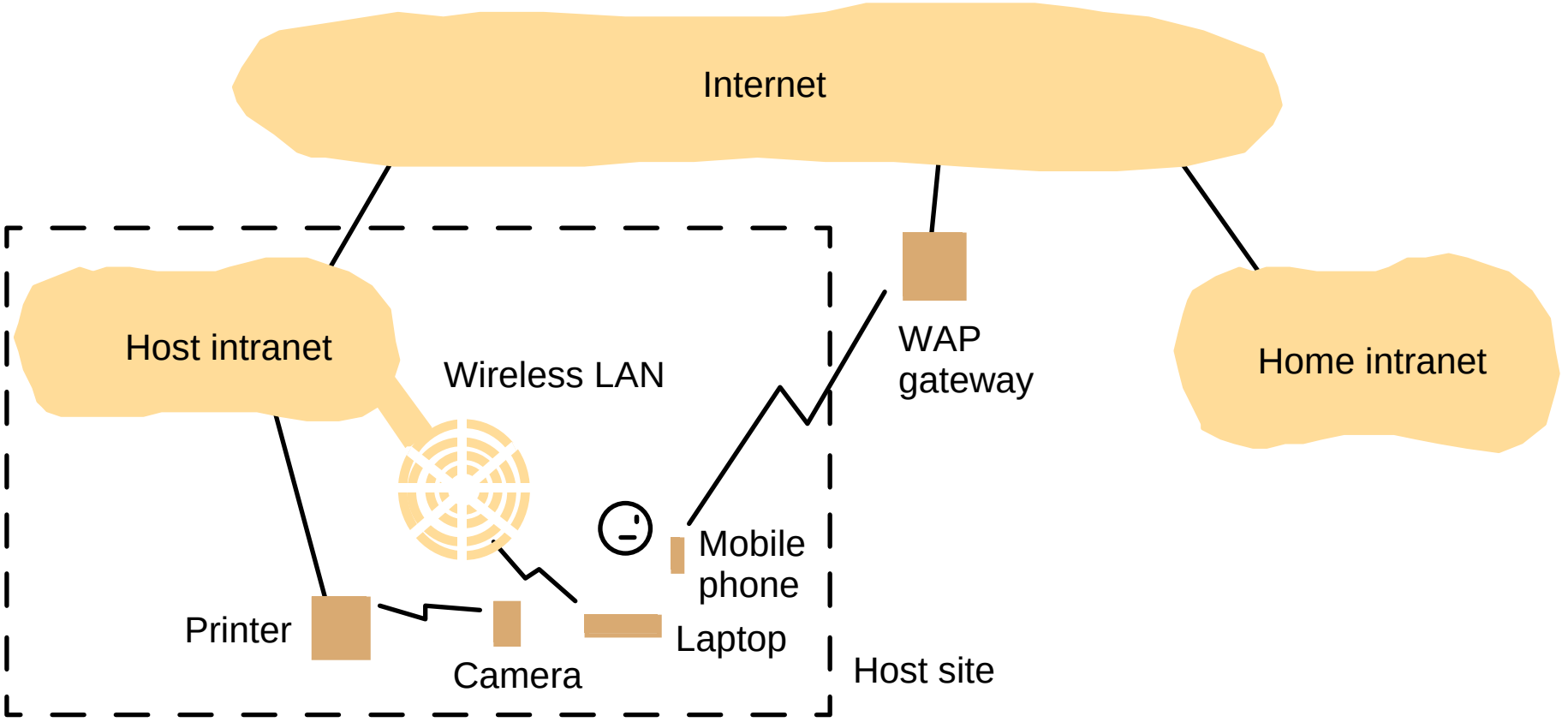
Exemplo 1: Computação Distribuída na Internet



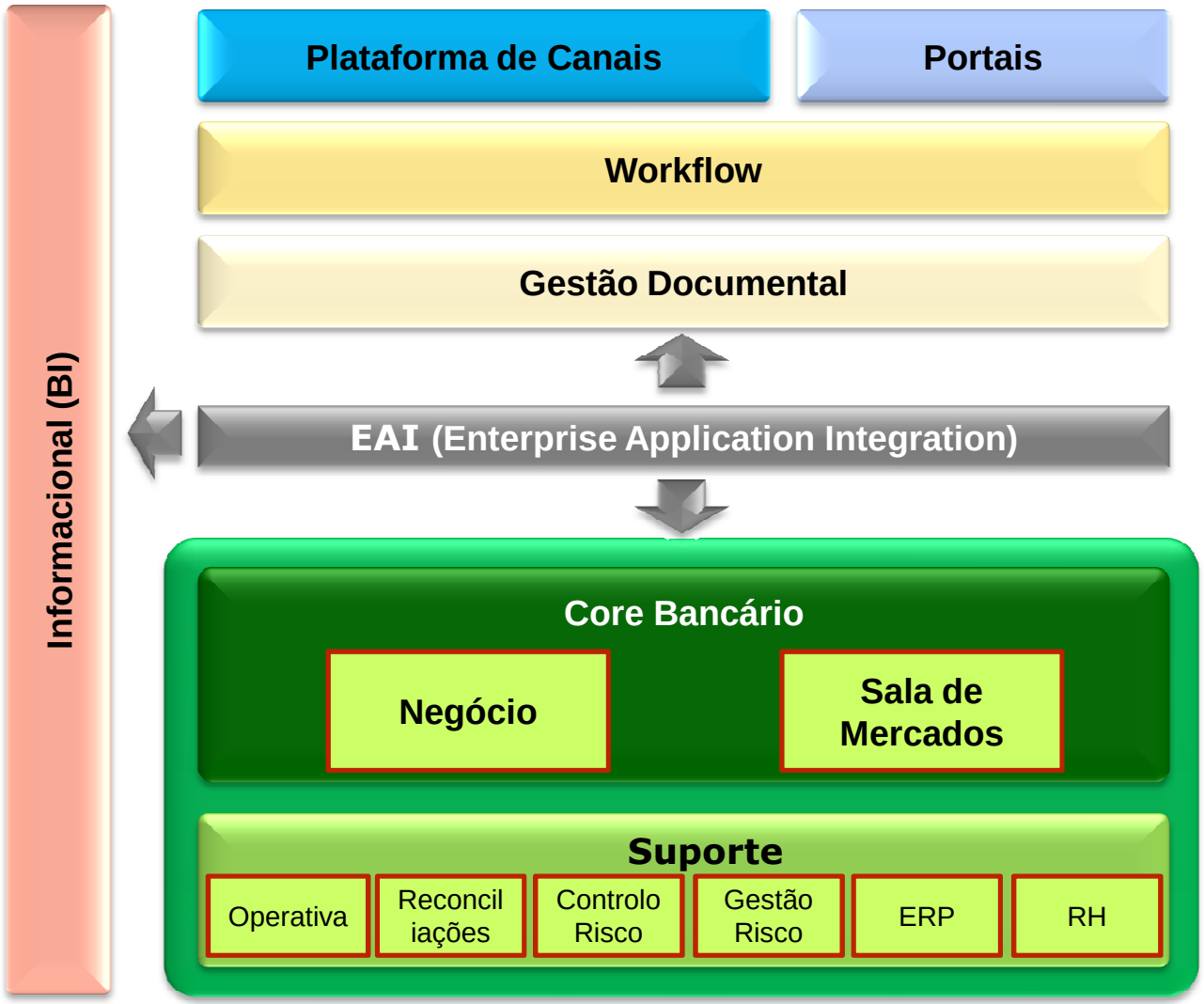
Exemplo 2: Computação Distribuída numa Intranet



Exemplo 3: Computação Móvel Distribuída



Exemplo 4: Computação Distribuída no SI de um Banco



Fim da Introdução

Objectivos:

- No final deste capítulo, os alunos devem ficar a conhecer:
  - O conceito de distribuição e as suas motivações
  - As características essenciais dos sistemas distribuídos
  - Os requisitos da distribuição
    - Heterogeneidade
    - Abertura
    - Segurança
    - Escalabilidade
    - Gestão de falhas
    - Concorrência
    - Transparência
- Trabalho individual complementar
  - Fornecer exemplos de Sistemas Distribuídos
  - Ler o 1º capítulo do livro “Distributed Systems: Concepts and Design” (4th Edition), by Coulouris, Dollimore & Kindberg

Referências

- Livro de Referência
  - http://www.cdk5.net
- ANSA Project - Introdução
  - http://www.ansa.co.uk/ANSATech/89/TR0302.pdf
- OMG - Object Management Group
  - http://www.omg.org
- Corba
  - http://www.omg.org/gettingstarted/corbafaq.htm