Computação Distribuída

Cap I - Introdução

Licenciatura em Engenharia Informática

Universidade Lusófona

Prof. Paulo Guedes (paulo.guedes@ulusofona.pt)

Introdução

- Caracterização dos Sistemas Distribuídos
- Desafios dos Sistemas Distribuídos

Sistemas Distribuídos: Definições …

"Conjunto de computadores independentes que se comportam perante os utilizadores como um único computador"

A. Tanenbaum

"Conjunto de computadores ligados por uma rede de comunicação, sem partilharem entre si memória nem um relógio único"

A. Silberschatz

"Sistema em que a falha de um computador que eu nem sabia que existia me impede de trabalhar"

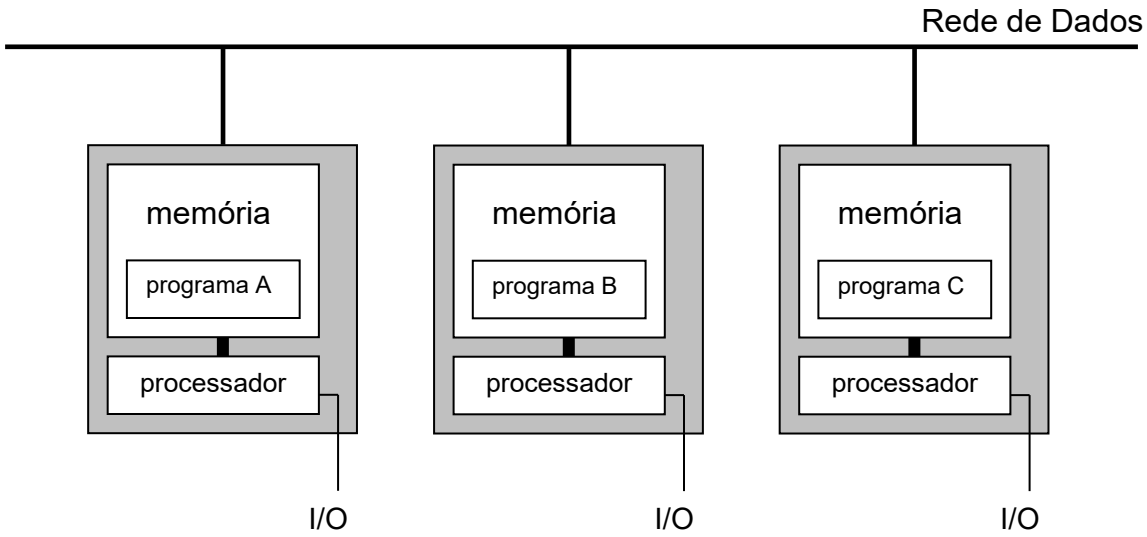
Leslie Lamport

Definição de Sistema Distribuído

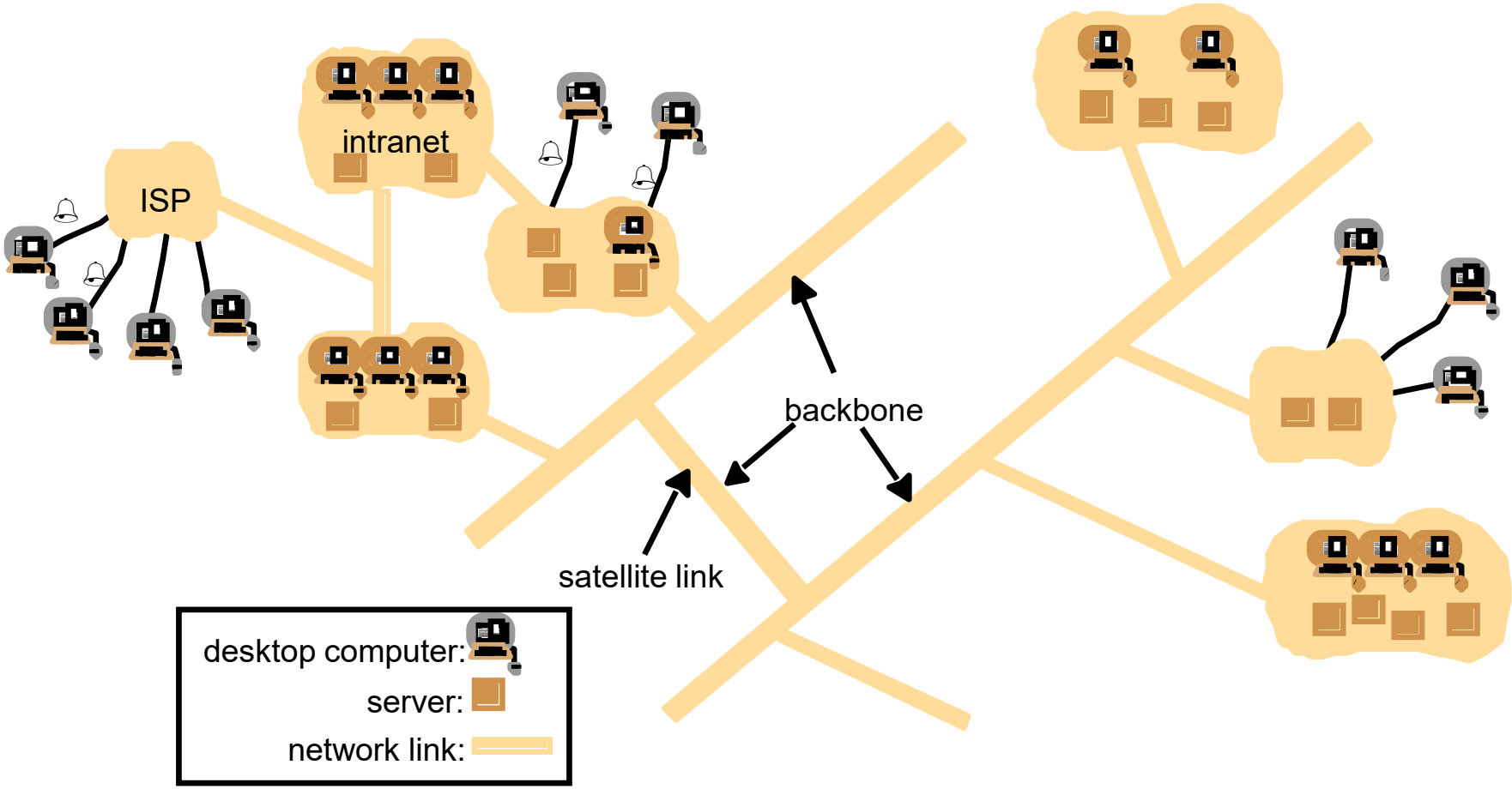
- Um **Sistema Distribuído** é constituído por um conjunto de componentes de software e hardware ligados por uma infra-estrutura de comunicação
  - Bus, malha, rede, etc.
- Os componentes cooperam e coordenam-se para realizar um **objectivo comum**
  - Aplicação Distribuída
  - Sistema Operativo Distribuído

Processadores em Rede

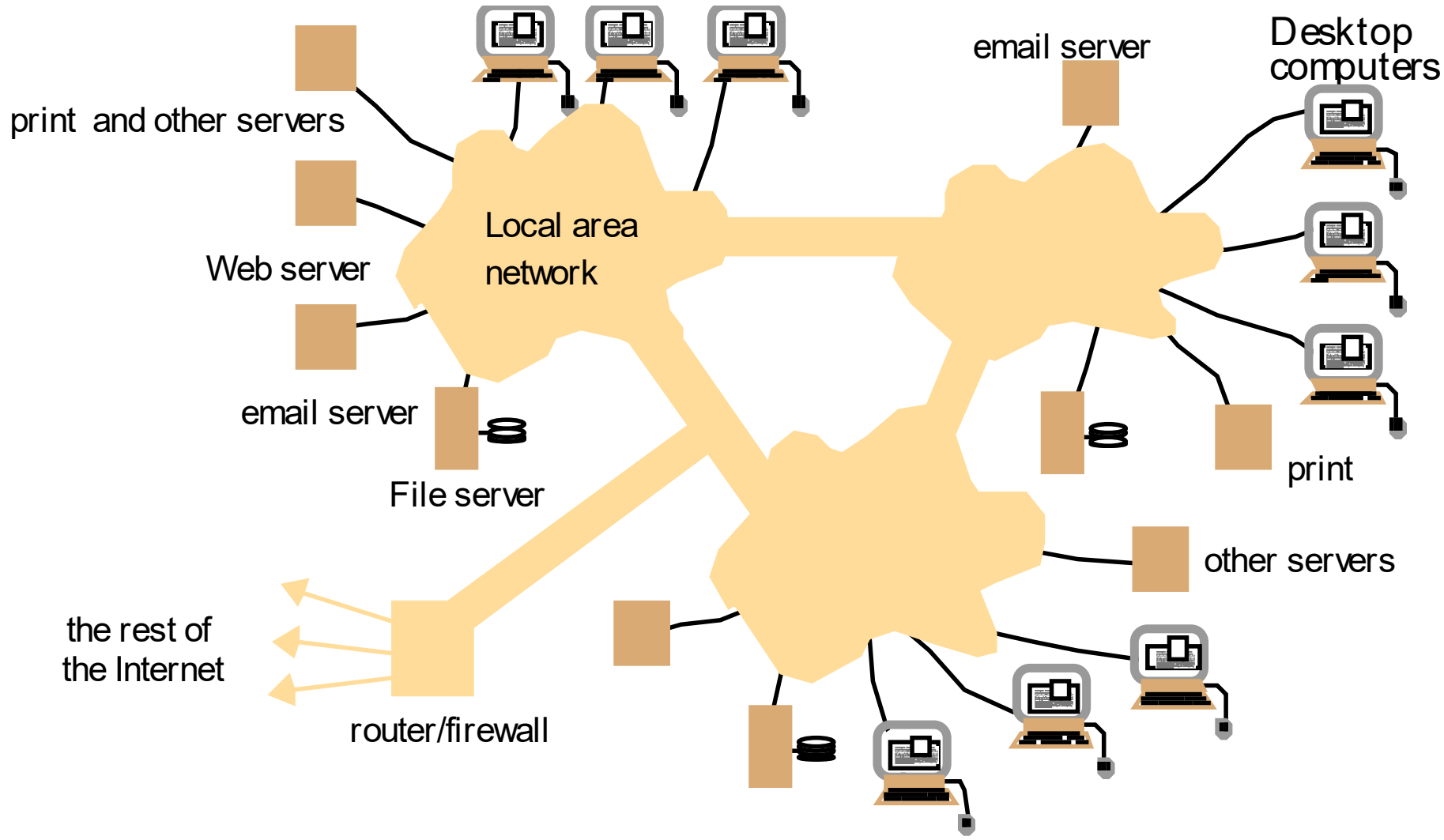
- Cada processador executa uma instância distinta do SO, podendo ou não oferecer uma imagem de sistema único às aplicações
- A distribuição é gerida exclusivamente por software



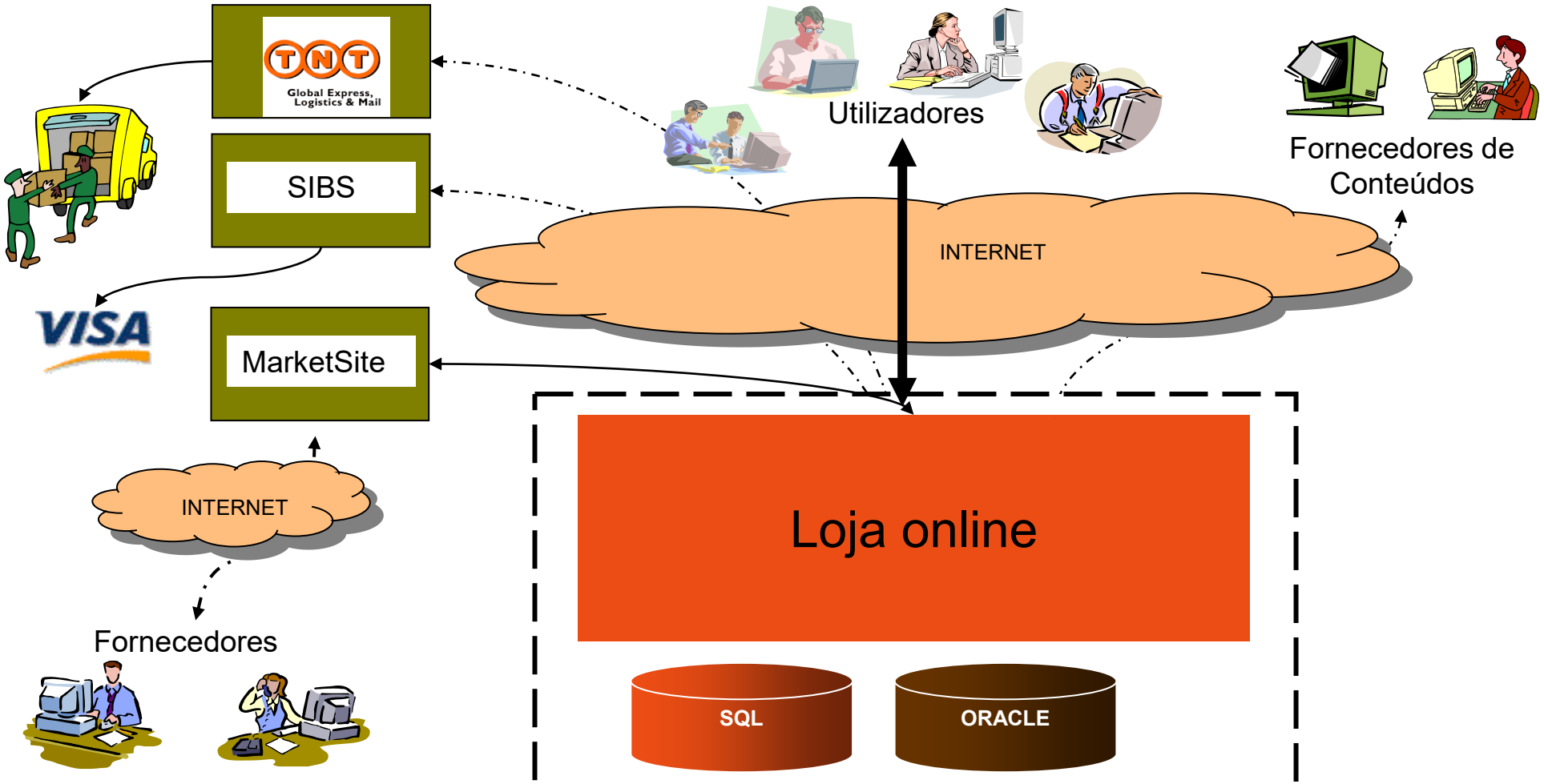
Exemplo: Computação Distribuída na Internet



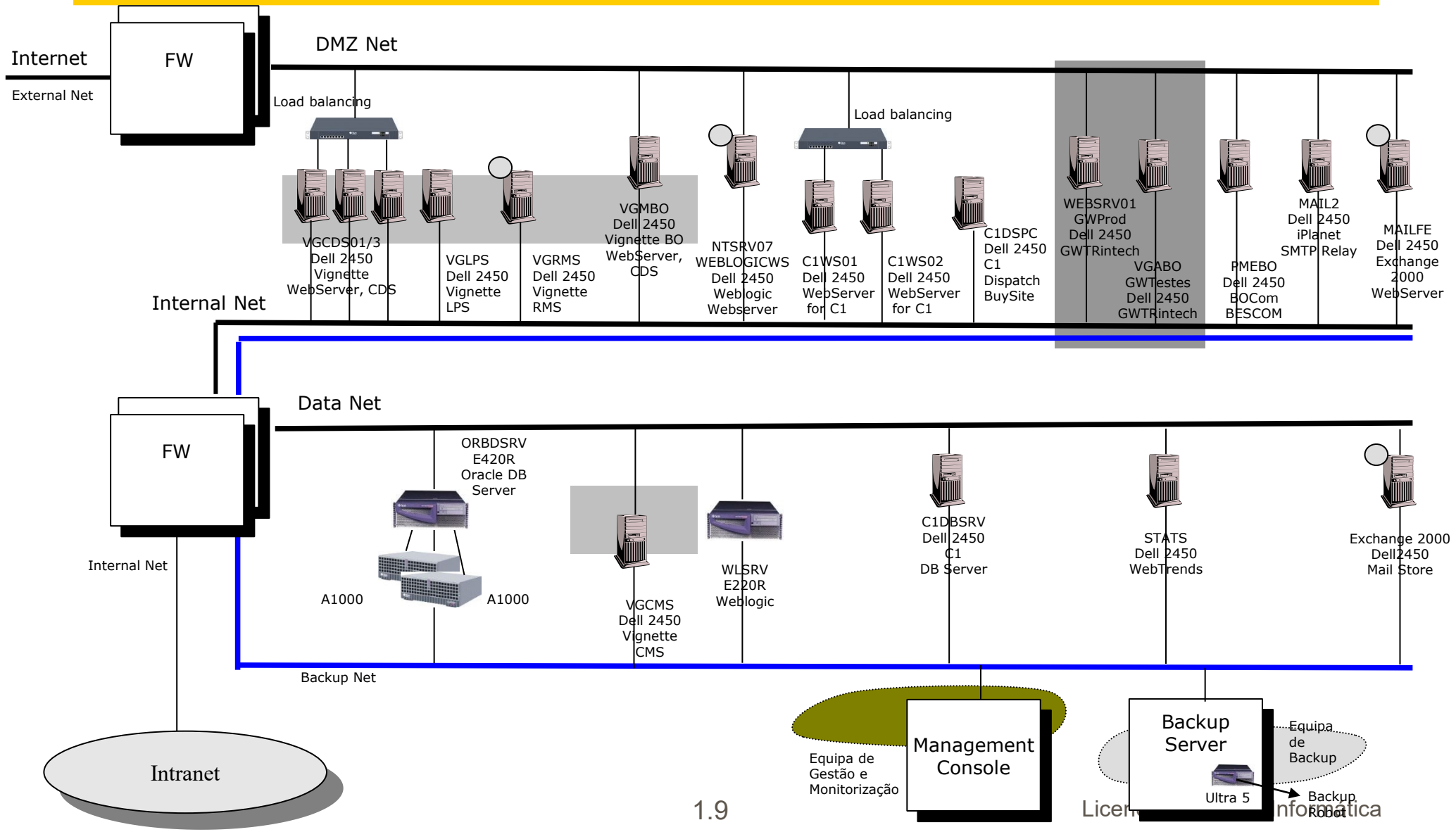
Exemplo: Computação Distribuída numa Intranet



Exemplo: site de compras



Exemplo: arquitectura física de um site de compras



Razões para a Distribuição

- Distribuição geográfica
  - Organização com instalações em Lisboa, Porto, Paris, …
  - Ligação entre organizações geograficamente distantes
- Extensibilidade, modularidade
  - Crescimento gradual
- Partilha de recursos
  - Troca de informação entre departamentos, empresas
- Maior disponibilidade
  - Replicação
- Maior desempenho
  - Distribuição da carga

Requisitos dos Utilizadores

- Transparência
- Partilha de informação
- Melhoria da comunicação entre as pessoas
- Segurança e protecção
- Fiabilidade e disponibilidade

Requisitos dos Programadores

- Interfaces normalizadas
- Ambiente de programação independente das características do hardware e das redes
- Controlo sobre o desempenho, fiabilidade, disponibilidade, segurança

Requisitos dos Gestores de Recursos

- Investimento reduzido
- Modularidade e extensibilidade na evolução do sistema
  - Protecção do investimento anterior
- Adequação às equipas técnicas
- Optimização da gestão dos recursos
- Desempenho, disponibilidade, fiabilidade, segurança

Desafios dos Sistemas Distribuídos

- No sentido de permitir uma melhor resolução dos problemas apontados, os SD têm que responder a um conjunto de desafios:
  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

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
  - Intrusos podem ler mensagens em trânsito, injectar novas mensagens
  - Não existe controlo sobre o software sistema e aplicações remotas
- 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
  - Controlo de acessos
  - 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
- Aparente paradoxo
  - Distribuição de carga => Melhor desempenho
  - Acesso remoto e não local => Pior desempenho
- 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

Gestão de Falhas

- Modelo de faltas mais complexo
  - Máquinas falham independentemente
  - Redes podem perder pacotes, trocar a sua ordem, ...
- Conhecimento parcial do estado do sistema
  - Das outras máquinas, só se sabe realmente que uma mensagem chegou, ou não chegou
  - Uma mensagem não chegou porque
    - Se perdeu ?
    - O emissor falhou ?
    - O emissor está muito lento ?

    Podemos nunca saber ao certo!!
- Aparente Paradoxo
  - Replicação => Melhor disponibilidade
  - Mais máquinas que podem falhar => Pior disponibilidade

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**

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. *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.