Computação Distribuída

Odorico Machado Mendizabal

Universidade Federal de Santa Catarina – UFSC
Departamento de Informática e Estatística – INE

Características de Sistemas Distribuídos

Sistemas Distribuídos

Plataformas Computacionais

- Arquiteturas de computadores mais baratas e com grande poder computacional
- Grande interconectividade

Utilidade

- Diversidade de aplicações (Comércio eletrônico, sistemas transacionais, processamento de grandes volumes de dados)

Computação Distribuída

- Poder de processamento,
- Escalabilidade,
- Confiabilidade,
- Desempenho
- ...

Sistemas Distribuídos

Computação Distribuída

- Poder de processamento,
- Escalabilidade,
- Confiabilidade,
- Desempenho
- ...

Sistemas Distribuídos

Computação Distribuída

- Poder de processamento,
- Escalabilidade,
- Confiabilidade,
- Desempenho
- ...

Desafios

- Heterogeneidade
- Extensibilidade / Sistemas Abertos (*Openness*)
- Escalabilidade
- Segurança
- Tratamento de falhas
- Concorrência

Heterogeneidade

- Permite que nodos/componentes com diferentes propriedades façam parte de um mesmo sistema
- Esta diferenciação pode ser quanto a:
  - Redes (Ethernet, ATM, Myrinet, etc..)
  - *Hardware* de computador (PC/CISC – x32 ou x64, RISC,...)
  - Sistemas Operacionais (Linux, Windows, Android...)
  - Linguagens de programação (C, C++, Java, C#, Python,...)
  - Implementações de um subsistema (Ex. Roteador CISCO, USRobotics, ..., Banco de Dados: Oracle, MS SQL Server, etc...)

Heterogeneidade – Usando Camadas Intermediárias

- Camadas intermediárias criam interfaces comuns a todos elementos do sistema, mesmo que estes sejam computadores e redes heterogêneos
- Independente do SO e HW local, o acesso remoto é feito a partir de uma interface bem definida

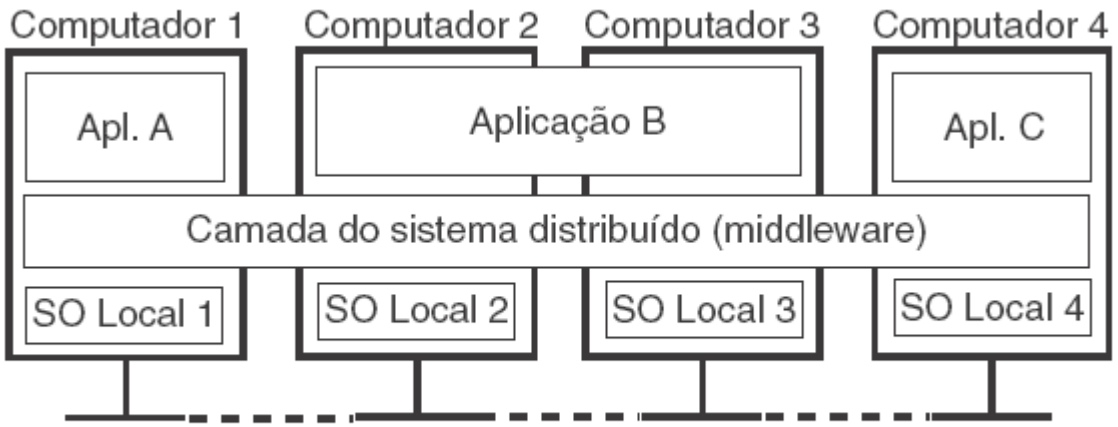

Figura 1.1 Sistema distribuído organizado como middleware.

Fonte: *Sistemas Distribuídos – Princípios e Paradigmas. Tanenbaum e Steen*

Extensibilidade

- Determina se novas funcionalidades podem ser incorporados ao sistema
- Como fomentar a extensibilidade:
  - Publicação de interfaces
  - Documentação e especificação
  - Uso de código aberto (*Open source*)
  - Uso de padrões de projeto e arquiteturas de SW bem definidas
- Exemplos:
  - Descrição pública na Internet através de RFCs (*Request For Comments*)
  - Conjunto de *system calls* de IPC no BSD UNIX (anos 80)

Exemplo: RFC 793 – TCP

TRANSMISSION CONTROL PROTOCOL DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION
September 1981
prepared for Defense Advanced Research Projects Agency Information Processing Techniques Office 1400 Wilson Boulevard Arlington, Virginia 22209
By Information Sciences Institute University of Southern California 4676 Admiralty Way Marina del Rey, California 90291

TABLE OF CONTENTS PREFACE

Flow Control: TCP provides a means for the receiver to govern the amount of data sent by the sender. This is achieved by returning a "window" with every ACK indicating a range of acceptable sequence numbers beyond the last segment successfully received. The window indicates an allowed number of octets that the sender may transmit before receiving further permission.

Precedence and Security: The users of TCP may indicate the security and precedence of their communication. Provision is made for default values to be used when these features are not needed.

Sequence Number: 32 bits The sequence number of the first data octet in this segment (except when SYN is present). If SYN is present the sequence number is the initial sequence number (ISN) and the first data octet is ISN+1.

Acknowledgment Number: 32 bits If the ACK control bit is set this field contains the value of the next sequence number the sender of the segment is expecting to receive. Once a connection is established this is always sent.

Escalabilidade

- Suporta o acréscimo de recursos e usuários mantendo o desempenho satisfatório e estabilidade do sistema
- Mudanças na dimensão do sistema computacional ou na carga de utilização não devem afetar o comportamento do sistema
- Desafios:
  - Custo dos recursos físicos (infra-estrutura)
  - Controlar a queda de desempenho inerente às limitações físicas dos dispositivos
  - Prevenir escassez de recursos
  - Evitar gargalos de desempenho

Escalabilidade – Exemplo

- Algumas técnicas podem ser usadas para aumentar a escalabilidade:

  (a) “Ocultar latência” - minimizar transferência de dados entre requisitante e servidor.

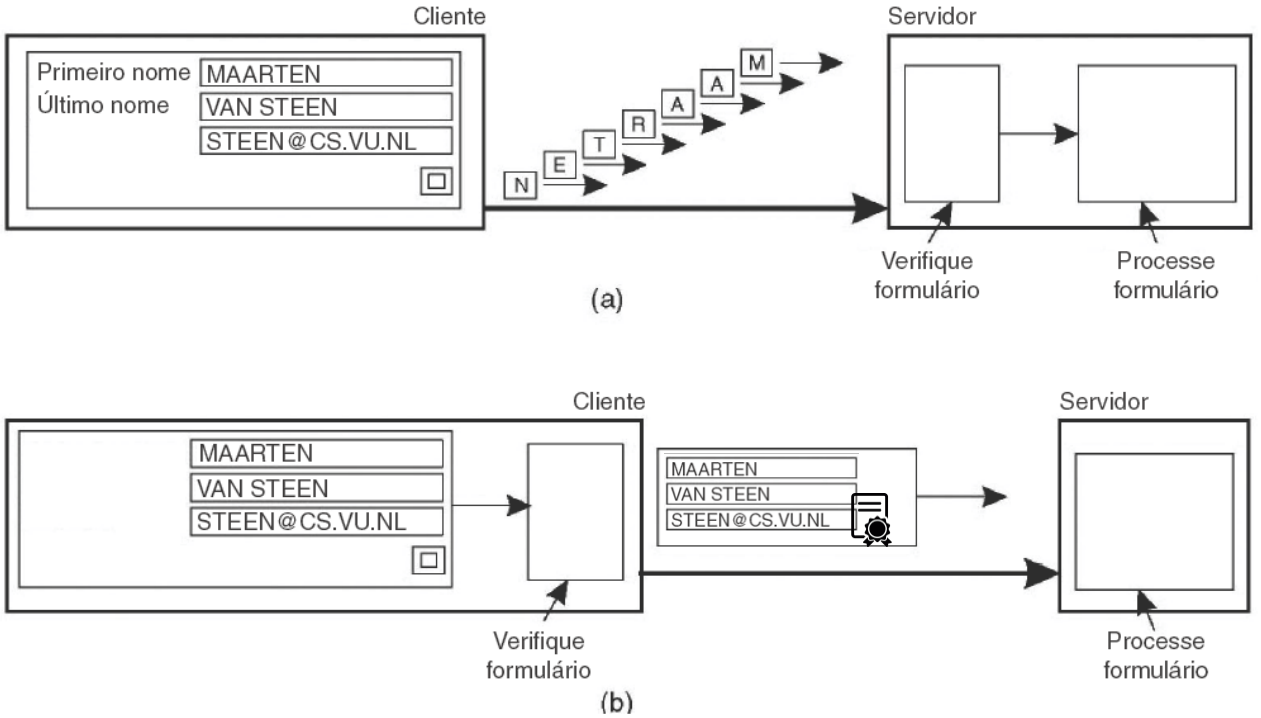

Figura 1.2 A diferença entre deixar (a) um servidor ou (b) um cliente verificar formulários à medida que são preenchidos.

Fonte: *Sistemas Distribuídos – Princípios e Paradigmas. Tanenbaum e Steen*

Escalabilidade – Exemplo

(b) Distribuição entre servidores

E.x.: DNS – Servidor de Resolução de Nomes Global

Diferentes servidores são usados para resolver nomes em diferentes níveis.

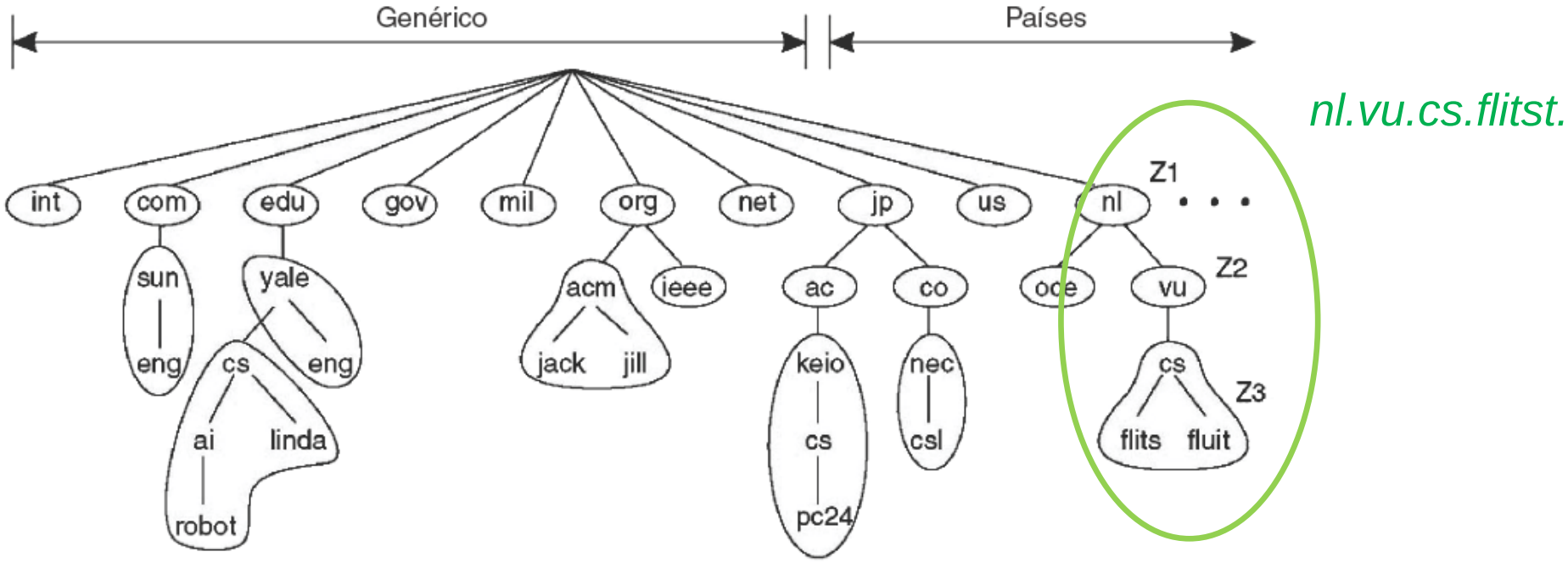

Figura 1.3 Exemplo de divisão do espaço de nomes do DNS em zonas.

Fonte: *Sistemas Distribuídos – Princípios e Paradigmas. Tanenbaum e Steen*

Escalabilidade – Exemplo

(c) Replicação

Diferentes servidores são usados para manter o mesmo conteúdo, sendo que:

- Servidores podem atender diferentes clientes por proximidade

- Servidores podem ser escolhidos para atender requisições aleatoriamente ou baseado em políticas de balanceamento de carga

- Servidores podem implementar sistemas de cache

Desafio: Garantir consistência da informação replicada

Escalabilidade vertical baseia-se na adição de mais componentes à capacidade do nodo em execução;
Escalabilidade horizontal envolve a adição de mais nodos

Segurança

- Características que precisam ser garantidas
  - Confidencialidade
  - Integridade
  - Disponibilidade

  outros: autenticidade, legalidade (responsabilidade)
- Desafios
  - Vulnerabilidades devido a falhas acidentais
  - Ataques intencionais
    - Negação de Serviço (DDOS – *Distributed Denial of Service*)
    - Código Móvel (Vírus, *worms*, *bots*, *keyloggers*, *spyware*, etc.)

Tratamento de Falhas

- O desvio do funcionamento esperado nas aplicações (ocasionado por falhas de HW ou SW) pode produzir resultados inesperados.

- Tipos de falhas
  - Física, *software* e humana

- Técnicas de Prevenção e/ou Recuperação:
  - Detecção de falhas
  - Ocultação de falhas
  - Tolerância a falhas (uso de replicação)
  - Recuperação de falhas

Concorrência

- Permitir que recursos compartilhados sejam utilizados por diversos processos concorrentemente
  - Usuários distintos
  - Independência de recursos
  - Consistência

- Resolução envolve desenvolvimento de mecanismos para:
  - sincronização entre processos
  - Exclusão mútua distribuída

Transparência

- Esconder do usuário/programador a separação de componentes de um sistema distribuído, tal que este seja visto como um sistema centralizado

- Formas de transparência: acesso, localização, concorrência, replicação, falha, mobilidade, desempenho e escala

Tipos de Transparência

- **Transparência de acesso**: recursos locais e remotos são acessados pelas mesmas operações
- **Transparência de localização**: recursos são acessados sem que sua localização seja determinada explicitamente
- **Transparência de concorrência**: processos executam concorrentemente, utilizando recursos compartilhados, sem interferirem na execução dos outros
- **Transparência de replicação**: múltiplas cópias de um recurso para aumentar o desempenho e disponibilidade dos seus serviços, sem o conhecimento das réplicas por usuários e programadores
- **Transparência a falhas**: ocultar e tratar as falhas (hardware ou software) permitindo que as aplicações ou usuários completem suas tarefas de forma correta

Tipos de Transparência

- **Transparência de mobilidade**: movimento de recursos ou clientes dentro do sistema não podem afetar a operação dos usuários ou programas
- **Transparência de desempenho**: sistema deve permitir ser reconfigurado para garantir bom desempenho a despeito da variação de carga
- **Transparência de escala**: as aplicações e o sistema devem permitir serem expandidas, sem modificar a estrutura ou algoritmos

Transparência – Exemplo

NFS (*Network File System*):

- Permite o compartilhamento de arquivos entre usuários Linux/Unix (desde 1980)
- Monta um sistema de arquivos remoto e viabiliza o acesso e manipulação dos arquivos pela rede

```
# arquivo /etc/exports
/users 192.168.0.10(rw,sync,no_root_squash)
```

Servidor

```
$ mount -t nfs 192.168.0.10:/users /mnt/usuarios

# arquivo /etc/fstab
192.168.0.10:/users /mnt/usuarios  nfs\
auto,nofail,noatime,nolock,intr,tcp,actimeo=1800 0 0
```

```
$ ls /mnt/usuarios
$ cd /mnt/usuarios
$ echo “alo” > test.txt
$ rm text.txt
```

Cliente

Transparência – observações

- Prover “transparência completa” pode ser muito custoso
  - Latências não negligenciáveis
  - Esconder falhas completamente é impossível
- Transparência completa impacta o desempenho
  - Ex. manter réplicas atualizadas atomicamente
  - Escritas síncronas em disco para tolerância a falhas

Transparência – observações

- Diferentes graus de transparência
- Expor distribuição pode ser desejável
  - Conhecer a localização de um recurso ou usuário pode ser benéfico
    - Ex. encontrar um recurso mais próximo (*mirror* para download)
  - Quando participantes estão em diferentes fusos
  - Ter clareza sobre o que está acontecendo
    - Se o servidor não responde por um longo período, pode ser preferível saber da falha e cancelar a operação

Referências

Parte destes slides são baseadas em material de aula dos livros:

- *Coulouris, George; Dollimore, Jean; Kindberg, Tim; Blair, Gordon. Sistemas Distribuídos: Conceitos e Projetos. Bookman; 5ª edição. 2013. ISBN: 8582600534*

- *Tanenbaum, Andrew S.; Van Steen, Maarten. Sistemas Distribuídos: Princípios e Paradigmas. 2007. Pearson Universidades; 2ª edição. ISBN: 8576051427*