Índice

1. Introdução

1.1. Sistema Distribuído

Coleção de computadores independentes, conectados em rede e que se mostra aos utilizadores como um sistema único e coerente.

Independentes: em termos de arquitetura, SO e etc.

Sistema Único: interação com um único sistema.

Sistemas distribuídos atuais:

- Smartphone
- Computadores em todo o lado
- Armazenamento remoto
- Apps
- Toneladas de dados

1.2. Sistema Distribuído Características

- Operam concorrentemente
- Estão Fisicamente Distribuídos
- Conectados por uma rede

1.3. Consequências

Execução concorrente de processos:

- Não determinismo, race-conditions, sincronização, deadlocks…

Inexistente de estado global:

- Nenhum processo tem conhecimento total sobre o estado global do sistema
- Coordenação é feita por troca de mensagem

Unidades podem falhar independentemente:

- Falhas na rede podem isolar computadores que estão a funcionar corretamente.
- Falhas nos sistemas poderão não ser visíveis imediatamente.

1.4. As 8 Falácias da Computação distribuída

- A rede é fiável
- A latência é zero (Tempo para chegar a informação)
- A largura de banda é finita (Quantidade de dados enviados)
- A rede é segura (Privilégios, segurança nas aplicações)
- A topologia não se altera
- Existe um administrador
- Custo de transporte é zero (custos da rede, largura de banda comprada)
- A rede é homogênea (Aplicações programadas em diferentes linguagens)

2. Arquiteturas de Sistemas Distribuídos

2.1. Background

A arquitetura está relacionada com a forma que os componentes (**Software**) dispersos por múltiplas máquinas o que requerem organização apropriada. As arquiteturas podem **centralizadas**, **descentralizadas** e **híbridas**.

2.2. Tipos de arquitetura

- **Arquiteturas em camada**

  Componentes são organizados em camadas onde as camadas de baixo fornecem serviços para as de cima.

- **Arquiteturas baseada em objetos**

  Organizados como objetos que comunicam entre si usando **RPC**.

- **Arquiteturas em eventos**

  Pensar em **CBD publisher-subscriber!** Os componentes registrados são notificados.

- **Arquiteturas centrada em dados**

  Uma mesma **BD** acedida por múltiplos componentes!

2.3. Arquiteturas centralizadas

Modelo **Cliente-Servidor:** Cliente faz um pedido ao servidor, o servidor recebe, processa (cliente espera) e retorna à informação. Há disposição de camadas e está da seguinte forma:

- Camada Interface com o utilizador
- Camada de processamento
- Camada de Dados

Exemplo: Motor de busca.

É possível espalhar essas camadas pelas máquinas e de várias formas.

2.4. Arquitetura descentralizadas

Modelo **peer-to-peer:** Computadores podem servir como cliente e servidor ao mesmo tempo. **Todos têm o mesmo direito!**

Modelo **Server-Based:** Mais usados, assimétricas e podem conter hierarquias.

2.5. Híbrida

Combinação da arquitetura centralizada e descentraliza.

- Sistemas Edge Server (Servidores na “beira” da internet)

  Clientes finais comunicam através dos **ISP’s** (**NOS, VODAFONE, MEO…**)

- Sistemas colaborativos distribuídos

  BitTorrent

3. Remote Procedure Calls

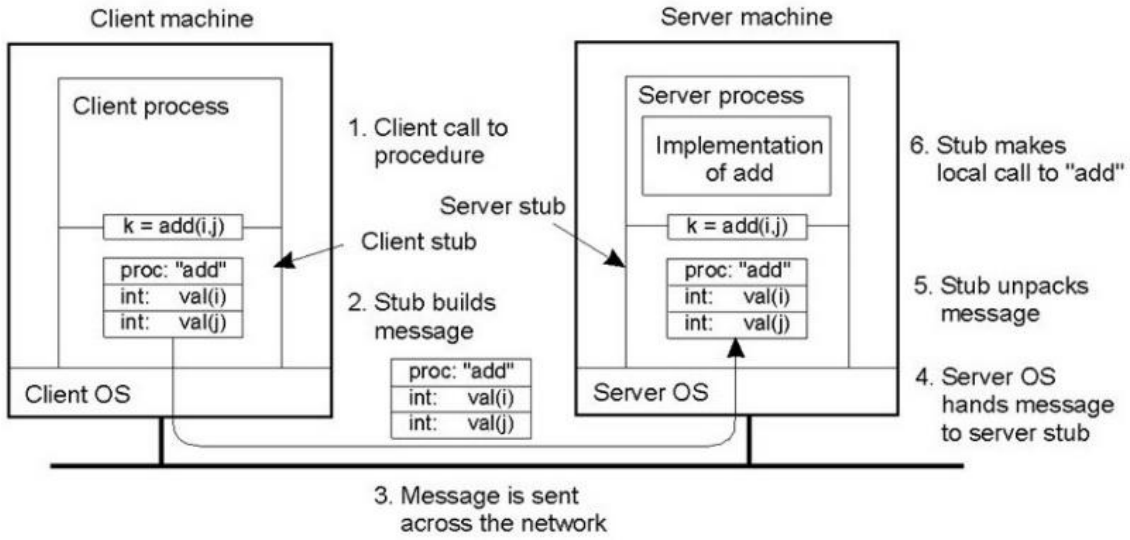
Providenciam acesso transparente aos recursos de uma outra máquina como se fosse na máquina local.

Um processo invocado pelo cliente no servidor é executado uma função onde os argumentos são enviados para o servidor pela rede, depois o servidor devolve esse valor e o cliente recebe o resultado. Exemplo: função **add(4, 8) value = 12.**

Processo Síncrono: Pedido -> Servidor Processa e Retorna -> Cliente obtém informação.

Processo Assíncrono: Pedido -> Cliente Continua tarefas -> Servidor Recebe Processa e Retorna -> Cliente é interrompido pelo servidor para receber Dados -> Cliente vê e confirma que recebeu.

Stub: Esboço de uma função, em desenvolvimento de software, é um pedaço de código usado para substituir algumas outras funcionalidades de programação.



- Processos entre diferentes computadores necessitam comunicar:
  - combinar formato de mensagens
  - ter em atenção as diferenças entre máquinas
- ***Stubs*** dos clientes e servidores são implementados
- Interfaces podem ser definidas através de um IDL

4. Protocolos TCP/IP

Fornece a abstração de uma rede fiável sobre um canal não-fiável.

Abstrai aplicações da complexidade da rede:

- Dados perdidos são reenviados
- Ordem de chegada dos dados é respeitada
- Garante estabilidade na rede e integridade dos dados

Este processo tem limitações:

- Cliente só pode enviar dados após enviar ***ACK***
- Servidor só pode enviar dados apos ***ACK*** do cliente
- Cada conexão implica uma ida-e-volta

4.1. Controle de Fluxo

Previne que o cliente sobrecarregue o servidor:

- Pode estar ocupado sobrecarregado com pedidos;
- Ter um buffer pequeno ou cheio;
- Utilizar variável para definir o tamanho dos dados enviados;

4.2. Receive Window (cwnd)

Número de dados que um nó pode receber.

- Cada nó inicializa o seu ***rwnd***. Em cada ***ACK,*** o nó receptor envia o seu ***rwnd*** onde o outro nó não envia mais que ***rwnd*** **bytes.**
- Se um dos lados não aguentar, manda um ***rwnd*** mais pequeno chegando a zero caso não pode receber mais.

4.3. Congestion Window (rwnd)

É mantido no lado do emissor, o envio do próximo número de pacotes está dependente se o ***ACK*** anterior foi totalmente recebido, caso contrário o próximo envio será mais baixo. O **cwnd** é incrementado gradualmente.

Limitações: Pode prejudicar a performance, depende de ***ACKS*** para crescer, logo demorará mais tempo.

4.4. Otimizações

Latência inicial (three-way handshake): Reutilizar conexões, mover servidor para mais perto do cliente.

Slow-Start: Aumentar o ***cwnd*** inicial, mover servidor para mais perto do cliente

Como programadores: enviar poucos dados enviar o máximo de dados agrupados

Em streaming de vídeo utilizar protocolo **UDP.**

5. Networking

5.1. Modelo OSI

- Open Systems Interconnection
- Modelo de referência (não necessariamente implementado na totalidade)
- Constituído por 7 camadas

5.1.1. Camada Física

Circuitos e hardware da rede (Layout de pinos, tensões e impedâncias)

Transmissão de sequências de dados binários (Circuitos elétricos, sinais de luz->fibra ótica, sinais eletromagnéticos-> rádio)

5.1.2. Ligação de dados

Deteção e correção de erros que possam ocorrer ao nível físico (Ligação ponto a ponto, Ligação ***broadcast***

5.1.3. Rede

- Fornece meios para controlar a operação na rede
- Roteamento de pacotes
- Controlo de sequência de pacotes
- Controlo de congestionamento
- Controlo de fluxo

5.1.4. Transporte

Recebe mensagens das camadas superior e segmenta-os para envio na camada de rede

- Segmentação e reassemblagem de pacotes
- Ordenação de pacotes
- Correção de erros e pacotes perdidos

5.1.5. Sessão

Responsável pela troca de dados entre hosts

5.1.6. Apresentação

Converte dados da camada de aplicação (#7) para a camada abaixo (#5)

- compressão dos dados da camada de aplicação
- conversão entre formatos de caracteres (ASCII > UTF8, etc)

5.1.7. Aplicação

Corresponde às aplicações que querem usar a rede:

-HTTP – IMAP – POP3 – SMTP – FTP – Telnet …

6. Naming

Um recurso pode estar associado a um ou mais nomes, mas um nome só está associado a um recurso.

6.1. Endereço

Um recurso pode ter um ou mais pontos de acesso designados por endereços

Endereços podem ser, IP, MAC ADDREES, MEMÓRIA.

6.2. Nomeação Plana

Nomes não estruturados. Ex: PCSALA1, IMPRESSORA 21…

Não contem informação sobre a geolocalização do recurso.

Broadcast: Pedidos são enviados para todos presente na rede. Consome muitos recursos na rede.

Multicast: Pedidos só são enviados para um determinado grupo (ex: sala de chat)

Solução Caseira: É quando um nó define um endereço "casa" quando move para outra localização este informa o seu novo endereço para a "casa". **Desvantagens:** Sempre terá de comunicar com a casa, está sempre fixo e deve sempre existir, caso o recurso mude para uma localização fixa, a casa irá impor latência desnecessária.

6.3. Tabelas de Dispersão Distribuídas

Estrutura chave-valor. Numa tabela de dispersão, cada nó tem apenas uma pequena parte da tabela total.

Cada nó tem um sucessor e um predecessor e conhece ambos. Para achar um nome, o nó verifica se conhece um endereço do destinatário, caso não tenha, pergunta aos predecessores.

6.4. Nomeação Estruturada

Compostos por nomes simples e legíveis organizados num espaço de nomes.

Os espaços de nomes estão organizados hierarquicamente em camadas lógicas.

Camada Global: Nós de alto nível (google.com)

Camada Administrativa: Nós intermédios (**grh**.google.com)

Camada Final: Nós que representam recursos finais (**antonio.martins**.fct.iul.pt)

Formas de obter o endereço: Iterativamente e Recursivamente (mais usado, consome latência). Interativo é desvantajoso porque ocupa sempre um servidor a processar uma resposta. Consome recursos do servidor.

DNS: Domain Name Server.

Hierarquia

- O nome mais a direita é o domínio de topo (**.com**, .**org**, .**pt**)
- Domínios de países são geridos por países (.**pt**, .**br**, .**uk**), nomes genéricos geridos pela **IANA** (.**com**, .**net**, .**biz**).

6.5. Nomeação Baseada em Atributos

Pode-se querer organizar os nomes dos recursos por atributos: serviços que oferecem, capacidade que possuam, características que possuam. Ao pesquisar por atributos reduz-se efetivamente o espaço de pesquisa.

6.6. Diretórios de Serviços

Diretórios em que as entidades descrevem os atributos. **Exemplo de uma impressora:** Tipo de papel, tecnologia (laser, lnkjet), Cores ou Preto e Branco.

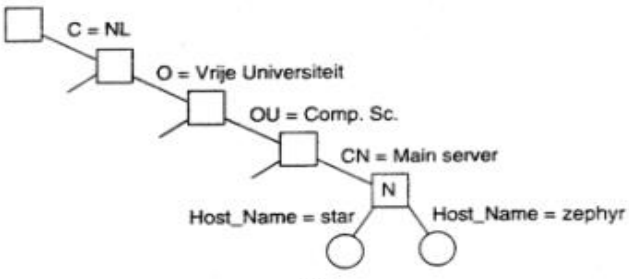
6.7. LDAP

Lightweight Directory Access Protocol

Mistura conceitos de nomes estruturados com atributos. Consiste num número de registos com uma coleção de atributos –> valor.

Attribute	Abbr.	Value
Country	C	NL
Locality	L	Amsterdam
Organization	O	Vrije Universiteit
OrganizationalUnit	OU	Comp. Sc.
CommonName	CN	Main server
Mail_Servers	—	137.37.20.3, 130.37.24.6, 137.37.20.10
FTP_Server	—	130.37.20.20
WWW_Server	—	130.37.20.20

Hierarquização dos vários atributos como se fossem domínios. Esta estrutura chama-se DIT (Directory Information Tree).



(a)

Attribute	Value
Country	NL
Locality	Amsterdam
Organization	Vrije Universiteit
OrganizationalUnit	Comp. Sc.
CommonName	Main server
Host_Name	star
Host_Address	192.31.231.42

Attribute	Value
Country	NL
Locality	Amsterdam
Organization	Vrije Universiteit
OrganizationalUnit	Comp. Sc.
CommonName	Main server
Host_Name	zephyr
Host_Address	137.37.20.10

(b)