
Engenharia de Serviços em Rede

TP1

TRABALHO REALIZADO POR:

CARLOS MIGUEL LUZIA DE CARVALHO

PAULO SILVA SOUSA

RUI EMANUEL GOMES VIEIRA



PG47092
Carlos Carvalho



PG47556
Paulo Sousa



PG47635
Rui Vieira

Índice

1	Questões e Resposta	1
1.	Pergunta 1	1
a.	Explique em que se diferenciam ambos os modelos, salientando o papel das principais entidades envolvidas.	1
b.	Enuncia vantagens e desvantagens de cada paradigma e casos de aplicação.	1
2.	Pergunta 2	3
a.	Web Browsing	3
b.	Multimedia Streaming	3
c.	IP Telephony	4
d.	File transfer/sharing	4
e.	Interactive Games	5
f.	Video Conferencing	5
3.	Pergunta 3	6
a.	Tabelas comparativas	6
b.	Processo de cálculo	7
c.	Conclusões	10
2	Conclusão	11

1 Questões e Resposta

1. Pergunta 1

As aplicações em rede assentam normalmente em paradigmas cliente-servidor ou *peer-to-peer*.

- a. Explique em que se diferenciam ambos os modelos, salientando o papel das principais entidades envolvidas.

No caso do modelo cliente-servidor os clientes e o servidor são entidades que estão presentes e são diferenciáveis. Neste caso os clientes não comunicam entre si diretamente, estando todos conectados a um servidor centralizado que possui todos os dados. Em relação a conectividade o servidor está sempre acessível com um *IP address* permanente, no contexto dos clientes a ligação pode ser interrompida e os *IP addresses* podem ser dinâmicos.

Relativamente ao modelo *peer-to-peer* todos os nodos comunicam diretamente e não é necessário existir um servidor sempre conectado. Neste caso tanto o servidor e os clientes não são distinguíveis sendo interpretados como nodos de tal maneira que todos podem pedir e fornecer serviços. No que concerne ao *IP addresses*, estes não possuem um *IP addresses* permanente e estão conectados de forma interrupta.

Confrontando a escalabilidade de ambos os modelos podemos concluir que o modelo Cliente-Servidor irá ter uma menor capacidade de suportar o aumento de clientes enquanto que o modelo *peer-to-peer* irá beneficiar com o aumento dos nodos do sistema.

- b. Enuncia vantagens e desvantagens de cada paradigma e casos de aplicação.

Cliente-Servidor

Vantagens ^[2]

- A segurança é bastante grande por conseguir distinguir os vários clientes deixando estes apenas aceder aos serviços que lhe são permitidos
- Servidor funciona como uma central de controlo visto que todos os serviços neste modelo estão centralizados no mesmo

Desvantagens ^[2]

- O modelo tem um custo associado bastante elevado
- Se o servidor falhar deixa de ser possível aos clientes usufruir dos serviços
- No caso de vários clientes tentarem aceder ao mesmo serviço, o servidor aumenta o tempo de resposta

Casos de aplicação ^[3]

- **Servidores Web**

Este tipo de servidor possui vários websites utilizados pelos diferentes clientes.

- **Servidores de Mail**

Com este exemplo é possível gerir todos os processos necessários para o envio e receção de e-mail de todos os emissores e destinatários.

- **Serviços que usam autenticação e/ou requerem uma segurança elevada**

Este modelo é bastante efetivo neste tipo de serviços por usar uma arquitetura centralizada permitindo assim a distinção dos vários clientes para estes terem o acesso personalizado e seguro.

Peer-to-peer

Vantagens ^[2]

- Fácil implementação relativamente à conexão dos vários nodos
- Em caso de falha de um dos nodos, os restantes continuam conectados e capazes de usufruir dos serviços disponíveis
- Modelo simples de fácil implementação com um custo reduzido

Desvantagens ^[2]

- Os serviços não estão centralizados sendo por vezes complicado aceder a um certo serviço
- Não é possível manter um backup dos ficheiros
- Os nodos são responsáveis pela própria defesa contra vírus e/ou malware introduzidos na rede por algum nodo

Casos de aplicação ^[4]

- **Mensagens direta**

Nesta caso usando este modelo foi possível partilhar a largura de banda pelos utilizadores permitindo uma comunicação mais rápida e segura.

- **Partilha de ficheiros**

Para este exemplo o modelo elimina a necessidade do uso de um servidor intermediário para se realizar a troca do ficheiro.

2. Pergunta 2

A Tabela 1 identifica tipos de aplicações amplamente usadas na Internet. Essas aplicações ou serviços apresentam diferente sensibilidade ao comportamento e desempenho da rede em si. Para cada tipo de aplicação (ou serviço), identifique qualitativamente os seus requisitos em termos de débito (throughput) necessário, atraso e suas variações (time sensitive) e perda de dados (loss sensitive). Dê exemplo concreto de aplicações da sua preferência que encaixem em cada tipo. Complemente a resposta quantificando os parâmetros em análise (referencie as suas fontes de informação).

a. Web Browsing

- Débito - Elastic
- Atraso (Time Sensitive) - Não
- Perda de Dados (Loss Sensitive) - Não aceita perdas
- Aplicações - Chrome, Firefox, Brave

Utilizando o exemplo em específico do Chrome

- Débito - [0.2Mbps-0.5Mbps](#)

No caso deste browser, é recomendada uma latência inferior a 100ms para uma experiência suave.

b. Multimedia Streaming

- Débito (audio) - [5kbps-1Mbps](#)
- Débito (vídeo) - [10kbps-5Mbps](#)
- Atraso (Time Sensitive) - Sim, alguns milissegundos
- Perda de Dados (Loss Sensitive) - Tolerante a perdas
- Aplicações - Spotify, Netflix, Youtube

Utilizando o exemplo do Spotify, em qualidade normal, para áudio e da Netflix, também em qualidade normal, para vídeo.

- Débito Spotify- [96kbps](#)
- Débito Netflix- [1.56Mbps](#)

Como podemos observar através dos dados recolhidos, a transmissão de vídeo exige muito mais débito do que a transmissão de áudio.

c. IP Telephony

- **Débito** - 20.8kbps-87.2kbps
- **Atraso (Time Sensitive)** - Sim, cerca de 10ms
- **Perda de Dados (Loss Sensitive)** - Tolerante a perdas
- **Aplicações** - WhatsApp, Viber, Messenger

Utilizando o exemplo em específico do WhatsApp

- **Débito** - 8-64kbps

Se compararmos os resultados obtidos para o WhatsApp com os resultados obtidos no ponto anterior, podemos verificar que voice over IP exige muito menos débito que streaming de vídeo/áudio.

d. File transfer/sharing

- **Débito** - Elastico
- **Atraso (Time Sensitive)** - Não
- **Perda de Dados (Loss Sensitive)** - Não aceita perdas
- **Aplicações** - BitTorrent, WeTransfer

Em relação a transferência de dados, o valor é elástico, ou seja, se o débito for baixo, o tempo de transferência/partilha dos ficheiros será muito elevado.

No entanto, deverá haver um valor mínimo de débito, mas como é demasiado baixo para a velocidade de internet nos dias de hoje, este não é anunciado por nenhuma aplicação de transferência/partilha de ficheiros.

e. Interactive Games

- **Débito** - Variável
- **Atraso (Time Sensitive)** - Sim, cerca de 10ms
- **Perda de Dados (Loss Sensitive)** - Não aceita perdas
- **Aplicações** - CS:GO, LOL, World of Warcraft

Utilizando os exemplos do CS:GO e do World of Warcraft

- **Débito CS:GO** - 560kbps
- **Débito World of Warcraft** - 90kbps

Como podemos observar com os resultados encontrados acima, o débito em jogos interativos é muito variável. Por exemplo, o CS:GO que é um jogo competitivo que exige um elevado tráfego de dados requer 560kbps de débito. Por outro lado, o WOW que é um MMO-RPG, exige muito menos tráfego de dados, sendo assim o débito apenas 90kbps.

f. Video Conferencing

- **Débito** - 384kbps-7Mbps
- **Atraso (Time Sensitive)** - Sim, cerca de 10ms
- **Perda de Dados (Loss Sensitive)** - Tolerante a perdas
- **Aplicações** - Zoom, Google Meets, Microsoft Teams

Utilizando o exemplo em específico do Zoom para chamadas de grupo em 1080p

- **Débito** - 3.0Mbps-3.8Mbps
- **Jitter** - 125-200ms

Comparando estes resultados com os resultados de VOIP, concluímos que a transmissão de vídeo exige muito mais débito do que apenas transmissão de áudio, sendo este cerca de 3Mbps em vez de 8-64kbps.

3. Pergunta 3

Considere a topologia da Figura 1 onde será distribuído um ficheiro de tamanho X Gbits entre N nodos (hosts), Assuma que os débitos de download e upload do nodo i . são respetivamente d_i e u_i . Assuma ainda que: (i) os hosts estão dedicados à distribuição do ficheiro, i.e. não realizam outras tarefas; e (ii) o núcleo da rede (core) não apresenta qualquer estrangulamento (bottleneck) em termos de largura de banda, i.e., qualquer eventual limitação existe nas redes de acesso dos vários n_i . O valor de X deve ser indexado ao identificador de cada grupo de trabalho, i.e., $X = ID_{Grupo}/10$.

Sabendo que o servidor tem um débito de upload $u_s = 1\text{Gbps}$, e que $d_i = 100\text{Mbps}$, calcule, justificando, o tempo mínimo de distribuição de F pelos N nodos quando $N=10$, $N=100$ e $N=1000$, e para débitos de upload u_i de: a) 1Mbps ; b) 5Mbps e c) 10Mbps , usando os modelos de distribuição: (i) cliente-servidor e (ii) *peer-to-peer*. Apresente os resultados numa tabela comparativa, bem como o processo de cálculo. Que conclusões pode tirar? Note que: 1kbits de dados a transmitir são 1024 bits e um débito de 1kbps são 1000 bits por segundo.

a. Tabelas comparativas

Client-Server

Client-Server	U-server (segundos)
N 10	26.8435456 s
N 100	268.435456 s
N 1000	2684.35456 s

Peer to Peer

Client-Server	1Mbps	5Mbps	10Mbps
N 10	26.8435456 s	26.8435456 s	26.8435456 s
N 100	244.0322327272727 s	178.95697066666668 s	134.217728 s
N 1000	1342.17728 s	447.39242666666667 s	244.0322327272727 s

b. Processo de cálculo

Os valores descritos nestas tabelas são resultado de duas formulas diferentes.

Client-Server

O tempo de distribuição de um ficheiro de tamanho F para N clientes numa abordagem Client-Server é dado pela formula:

$$D(s - c) \geq \max(NF/u(s), F/d(\min)) \quad (1)$$

Para calcular todos estes tempos mais rapidamente decidimos criar um script em python tornando estes cálculos mais rápidos.

```
def client_server(N):  
    u = 1 * 1000**3  
    x = 2.5 * 1024**3  
    d = 100 * 1000**2  
  
    print("N == " + str(N))  
  
    primeiro = N*x/u  
    print("p " + str(primeiro))  
    segundo = x/d  
    print("s " + str(segundo))  
  
    res = max(primeiro, segundo)  
    print(str(res) + "\n\n")  
  
    return 0;  
  
client_server(10)  
client_server(100)  
client_server(1000)
```

Figure 1: Client-Server script

```
N == 10  
p 26.8435456  
s 26.8435456  
26.8435456  
  
N == 100  
p 268.435456  
s 26.8435456  
268.435456  
  
N == 1000  
p 2684.35456  
s 26.8435456  
2684.35456
```

Figure 2: Client-Server script result

Peer-to-peer

Já para calcular o tempo de distribuição numa abordagem Peer to peer temos já também de ter em conta para além do client download rate minimo também o somatorio de upload rates.

$$D(p2p) \geq \max(NF/u(s), F/d(\min), NF/(u(s) + N * u(i))) \quad (2)$$

Também para efetuar mais rapidamente estes calculos criamos um script para a abordagem Peer to peer

```
def p2p(N, u):  
    u = 1 * 1000**3  
    x = 2.5 * 1024**3  
    d = 100 * 1000**2  
  
    print("N == " + str(N))  
    primeiro = x/u  
    print("p " + str(primeiro))  
    segundo = x/d  
    print("s " + str(segundo))  
    terceiro = N * x / (u + N * u)  
    print("t " + str(terceiro))  
  
    res = max(primeiro, segundo, terceiro)  
    print(str(res) + "\n" )  
  
    return 0;  
  
print("1MBPS\n")  
p2p(10, 1 * 1000**2)  
p2p(100, 1 * 1000**2)  
p2p(1000, 1 * 1000**2)  
  
print("\n\n5MBPS\n")  
p2p(10, 5 * 1000**2)  
p2p(100, 5 * 1000**2)  
p2p(1000, 5 * 1000**2)  
  
print("\n\n10MBPS")  
p2p(10, 10 * 1000**2)  
p2p(100, 10 * 1000**2)  
p2p(1000, 10 * 1000**2)
```

Figure 3: *Peer-to-peer* script

```
1MBPS
N == 10
p 2.68435456
s 26.8435456
t 26.57776792079208
26.8435456

N == 100
p 2.68435456
s 26.8435456
t 244.0322327272727
244.0322327272727

N == 1000
p 2.68435456
s 26.8435456
t 1342.17728
1342.17728
```

Figure 4: *Peer-to-peer* script result 1Mbps

```
5MBPS
N == 10
p 2.68435456
s 26.8435456
t 25.565281523809524
26.8435456

N == 100
p 2.68435456
s 26.8435456
t 178.95697066666668
178.95697066666668

N == 1000
p 2.68435456
s 26.8435456
t 447.39242666666667
447.39242666666667
```

Figure 5: *Peer-to-peer* script result 5Mbps

```
10MBPS
N == 10
p 2.68435456
s 26.8435456
t 24.403223272727274
26.8435456

N == 100
p 2.68435456
s 26.8435456
t 134.217728
134.217728

N == 1000
p 2.68435456
s 26.8435456
t 244.0322327272727
244.0322327272727
```

Figure 6: *Peer-to-peer* script result 10Mbps

c. Conclusões

Como podemos observar nas tabelas comparativas na alínea a) e como já referido anteriormente nas alíneas a) e b) da pergunta 1, o modelo **Peer-to-Peer** escala melhor a nível do número de nodos em relação ao modelo **Client-Server**. Uma vez que, com a análise da tabela, verifica-se que apesar de depender do débito de upload, o tempo decorrido na transferência de um ficheiro F com 2.5Gb, tem tendência a aumentar com o aumento do número de utilizadores porém é um aumento logarítmico, ao contrário do modelo oposto que tem tendência a aumentar linearmente com o aumento do número de utilizadores.

Numa arquitetura **Client-Server** cada servidor precisa de ser pensado para a quantidade específica de clientes a que dará suporte. Quando o número de clientes aumenta, o CPU do servidor, a memória, a rede e o desempenho do disco também precisam de crescer e podem, eventualmente, chegar a um ponto em que o servidor interrompe a operação. Se existirem mais clientes do que um único servidor pode suportar, provavelmente será necessário implementar vários servidores.

Já num modelo **Peer-to-peer**, quantos mais dispositivos existirem numa rede, mais nodos poderão participar na entrega de dados. Dessa forma, os sistemas *Peer-to-peer* podem ser considerados organicamente escaláveis, o que significa que o aumento da escala vem intrínseco com o aumento de nodos.

Em resumo, os sistemas P2P são organicamente escaláveis. Mais procura significa mais oferta, tornando-os ideais para aplicativos que envolvem grandes volumes de dados e/ou muitos clientes.

2 Conclusão

Neste trabalho foi-nos possível expandir os nossos conhecimentos sobre os modelos cliente-servidor e *peer-to-peer* permitindo-nos desta forma compreender e consolidar o conhecimento adquirido no início desta unidade curricular de Engenharia de Serviços em Rede.