Engenharia de Serviços em Rede
Trabalho Prático 1: Nível Aplicacional: Conceitos Introdutórios

José Santos, Leonardo Marreiros, and Pedro Fernandes

University of Minho, Department of Informatics, 4710-057 Braga, Portugal
e-mail: {a84288, pg47398, pg}@alunos.uminho.pt

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.

O modelo cliente-servidor destinge-se por ter um servidor sempre ligado o qual trata de pedidos de clientes. Por outro lado, na arquitetura *peer-to-peer* existe mínima ou nenhuma utilização de servidores, a comunicação é feita entre pares de *hosts* que se conectam intermitentemente (*peers*), daí o nome desta arquitetura *peer-to-peer* uma vez que os utilizadores comunicam sem passar por um servidor dedicado.

Enquanto que numa arquitetura *peer-to-peer* vão sendo trocados pedaços de informação entre o par envolvido, isto é, cada nodo pode pedir e prover serviços, na arquitetura cliente-servidor quando um servidor recebe o pedido de um objeto por parte de um cliente, responde com o objeto requerido, isto é, não há comunicação direta entre utilizadores/clientes.

Outra característica desta aquitetura é o facto de que o servidor tem um endereço de IP fixo, conhecido por todos, o que significa que um cliente pode sempre realizar pedidos ao servidor enviando *packets* para esse IP. Pelo contrário, na arquitetura *peer-to-peer* não há um IP fixo uma vez que os sistemas finais com quem comunicam são arbitrários e os seus IP's mudam.

Alguns dos aplicativos mais conhecidos com uma arquitetura cliente-servidor incluem os *browsers* Web, FTP, Telnet e e-mail.

Quanto ao *peer-to-peer*, alguns dos aplicativos que utilizam esta arquitetura são: BitTorrent para partilha de ficheiros, e serviços de telefone e vídeo pela internet como o Skype.

Para concluir, a diferença entre *peer-to-peer* e cliente-servidor é que na primeira arquitetura, cada nodo pode solicitar e fornecer serviços (descentralizado), enquanto na segunda, os clientes solicitam serviços e o servidor responde com serviços (centralizado).

b. Enuncie vantagens e desvantagens de cada paradigma e casos de aplicação.

É comum que numa aplicação cliente-servidor, um único servidor é incapaz de conseguir lidar com todos os pedidos dos clientes (pouco escalável). Por esta razão são criados centros de dados. Um *data center* pode ter centenas de milhares de servidores, que devem ser alimentados e mantidos. Além disso, os provedores de serviço devem pagar custos recorrentes de interconexão e largura de banda para o envio de dados dos centros de dados.

No caso de apenas existir um servidor, a arquitetura cliente-servidor é de pouca confiança pois os clientes dependem desse servidor, isto é, uma falha/queda do servidor implica uma interrupção do seu funcionamento para todos os clientes.

Além do mais, nesta arquitetura, múltiplos clientes requisitam serviços do servidor pelo que o seu tempo de acesso a serviços é maior do que na arquitetura *peer-to-peer*. Quanto a custos, a arquitetura cliente-servidor é mais dispendiosa de implementar que a sua contrapartida em análise.

Finalmente, esta arquitetura é mais estvável e segura do que a arquitetura *peer-to-peer*.

A arquitetura *peer-to-peer* tem como principal vantagem a escalabilidade própria. Por exemplo, num aplicativo de partilha de arquivos P2P, embora cada par gere carga de trabalho solicitando arquivos, cada par também adiciona capacidade de serviço ao sistema, distribuindo arquivos para outros pares.

Além disso, este tipo de arquiteturas tem um bom custo-benefício uma vez que não se tem de preocupar com questões como infraestrutura e largura de banda de servidores e com a implementação de *data centers* como o que acontece no modelo cliente-servidor.

No entanto, devido à sua elevada descentralização, estas arquiteturas debruçam-se com problemas como a segurança, performance e confiança.

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).

Tipos de Aplicações	Débito	Atrasos	Perda de Dados	Aplicações
Web Browsing	Elástico	Não	Sem perdas	Google Chrome [1]
Multimedia streaming	Elástico	Sim	Tolerante a perdas	Youtube [2]
IP Telephony(VoIP)	Elástico	Sim	Tolerante a perdas	Skype [3] [4]
File transfer/sharing	Elástico	Não	Sem perdas	BitTorrent [5]
Interactive Games	Elástico	Sim	Tolerante a perdas	League of Legends [6] [7]
Video Conferencing	Elástico	Sim	Tolerante a perdas	Zoom [8] [9]

Tabela 1: Requisitos dos diferentes tipos de aplicações

[1] https://support.google.com/chrome/a/answer/3339263?hl=en
[2] https://support.google.com/youtube/answer/78358?hl=en
[3] https://support.skype.com/en/faq/FA1417/how-much-bandwidth-does-skype-need
[4] https://docs.microsoft.com/en-us/skypeforbusiness/optimizing-your-network/media-quality-and-network-connectivity-performance
[5] https://computer.howstuffworks.com/bittorrent2.htm
[6] https://oce.learnwithleague.com/knowledgebase/what-is-lols-typical-data-and-bandwidth-usage/
[7] http://web.cs.wpi.edu/ claypool/iqp/league-lag/iqp-final-project.pdf
[8] https://support.zoom.us/hc/en-us/articles/204003179-System-requirements-for-Zoom-Rooms
[9] https://support.zoom.us/hc/en-us/articles/202920719-Meeting-and-phone-statistics

O *web browser* mais utilizado no mundo é o *google chrome*. Segundo a referência 1 o tempo de atraso desta aplicação deve ser inferior a 100ms e, dependendo do tipo de serviço a usar nesta aplicação o débito pode oscilar entre 0.2 a 5 Mbps.

O *Youtube* por sua vez é uma aplicação do tipo de *streaming* multimédia. Esta aplicação necessita de débitos diferentes dependendo da qualidade do vídeo que o utilizador está a ver. Em qualidades de vídeo mais baixas como 360p é necessário um débito de apenas 0.7 Mbps enquanto que, para qualidades superiores como 4k, são já necessários 20 Mbps, segundo a referência 2.

Uma aplicação muito conhecida do tipo IP *Telephony(VoIP)* é o *Skype*. Esta aplicação, dependendo do tipo de serviço que está a fornecer, pode mudar imenso os seus débitos. Fatores como fazer chamadas de apenas voz ou com vídeo, ou fazer chamadas com diversas pessoas impacta bastante o débito necessário para o efeito. Assim uma simples chamada, em débitos recomendados, necessita de apenas 100 kbps, enquanto que vídeo chamadas com grupos com mais de sete pessoas pode exceder aos 8 Mbps (Referência 3). No que toca à latência (Referência 4) de um *Round-Trip Time* (RTT) não deve ser superior aos 100 ms.

Para realizar partilha de ficheiros, o *BitTorrent* usa um método chamado *Tit for tat*. Este método favorece as pessoas que mais partilham ficheiros em detrimento das outras mais egoístas. Assim as pessoas que mais enviam dados, irão também receber dados a um ritmo maior que as outras que apenas se preocupam em ter o seu ficheiro descarregado. (Referência 5)

Jogos *online* são um outro tipo de aplicações que necessitam da Internet para funcionarem, Um exemplo disso é o *League of Legends* (LOL). Segundo (Referência 6) durante uma partida (de duração aproximadamente 30 min) são usados 14 MB de *download* e 3 MB de *upload* e os respetivos débitos devem rondar os 64 e 14 kbps respetivamente. Sendo o LOL um jogo jogado a uma escala global, este está dividido em diversas regiões, e estas têm diferenças significativas a níveis de latência. Na Coreia do Sul a latência média é de apenas 8 ms, enquanto que a grande parte das regiões têm uma latência média entre 60 e 70 ms. Em regiões de maiores dimensões estas latências aumentam, como é o caso da América do Norte cuja latência média é de cerca de 90 ms.(Referência 7)

Com a pandemia houve uma aplicação de vídeo conferência cuja popularidade aumentou exponencialmente, o *Zoom*. Dependendo do tipo de funcionalidades em uso durante uma reunião, o débito necessário vai ser diferente. Assim são necessários 2 Mbps de *upload* e de 2 a 6 Mbps de *download* consoante o número de monitores em uso. A partilha de ecrã necessita entre 150 e 300 kbps (Referência 8). Segundo (Referência 9) a latência média de quem envia conteúdo é de 12 ms para áudio, 11 ms para vídeo e 7 ms para partilha de ecrã. No lado de quem recebe conteúdo a latência média é de 16 ms para áudio e 10 ms para vídeo.

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 di e ui . 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 ni . O valor de X deve ser indexado ao identificador de cada grupo de trabalho, i.e., X=IDGrupo/10.

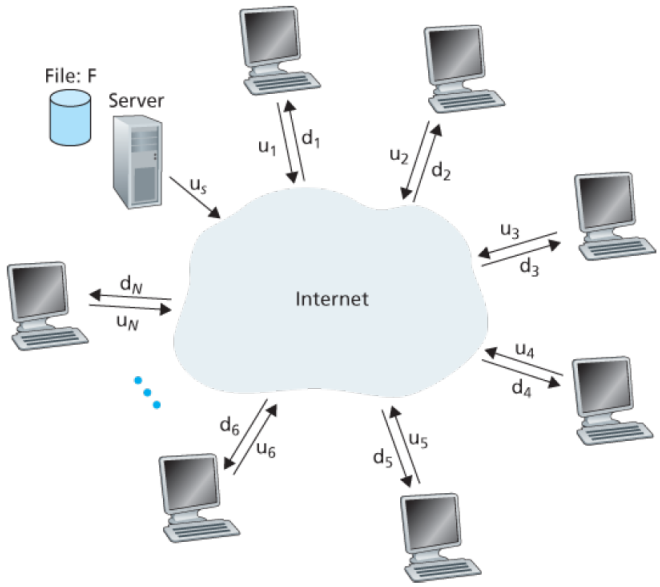

Figura 1: Distribuição do ficheiro F [Kurose, and Ross, 2016].

Sabendo que o servidor tem um débito de upload us=1Gbps, e que di =100Mbps, 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 ui 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.

3.1. Cliente-Servidor

Vamos primeiro determinar o tempo de distribuição para a arquitetura cliente-servidor, que denotamos por D_{cs} . Na arquitetura cliente-servidor, nenhum dos pares ajuda na distribuição do ficheiro.

Nesta arquitetura, o servidor deve enviar uma copia do ficheiro para cada um dos N nodos. Isto significa que o servidor transmite NF bits e, uma vez que a velocidade de upload é u_s, o tempo para distribuir o ficheiro deve ser pelo menos NF/u_s.

Sabendo que d_i é o débito de download de cada nodo, então para obter todos os bits do ficheiro F, necessita de pelo menos F/d_i segundos.

Assim sendo, temos: $D_{cs} = max\{NF/u_s, F/d_i\}$.

Sabendo que: $F = 5Gb = 5368709120\ bits, u_s = 1Gbps = 1000000000\ bps, d_i = 100\ Mbps = 104857600\ bps$, podemos calcular a seguinte tabela:

Número de nodos	NF/u_s	F/d_i	F/u_s	D_{cs}
10	53.6s	51s	5.37s	53.6s
100	536.8s	51s	5.37s	536.8s
1000	5368.7s	51s	5.37s	5368.7s

Tabela 2: Tempo de distribuição de F

Pela expressão de D_{cs} anterior podemos verificar que o tempo de distribuição aumenta linearmente com o número de nodos N na arquitetura cliente-servidor. É possível verificar através dos resultados da tabela que com um aumento do número de nodos em 10 vezes, o tempo de distribuição necessário para distribuir o ficheiro pelos nodos aumenta também em 10 vezes.

3.2. Peer-to-peer

Na arquitetura peer-to-peer, cada nodo pode ajudar o servidor na distribuição do ficheiro. Isto é, quando um nodo recebe uma parte do ficheiro, pode usar a sua capacidade de upload para redistribuir essa parte do ficheiro para os outros nodos.

No inicio da distribuição apenas o servidor tem o ficheiro pelo que o servidor o deve enviar na sua totalidade para o link de acesso. Então, o tempo mínimo de distribuição é pelo menos F/u_s. A diferença entre esta arquitetura e a anterior encontra-se no facto de que um bit de informação enviado pelo servidor uma vez não necessita de ser enviado de novo pelo servidor uma vez que os nodos podem redistribuir esse bit entre si.

Assim como na arquitetura cliente-servidor, cada nodo obtém o ficheiro F em pelo menos F/d_i segundos.

Finalmente, a capacidade de *upload* total do sistema é igual à soma do débito de *upload* do servidor com o débito de *upload* de cada um dos nodos, isto é, $u_{total} = u_s + u_1 + ...u_n$. Isto significa que o débito máximo a que o sistema distribui os NF bits é u_{total}, pelo que o tempo de distribuição é, no mínimo, NF/u_{total}.

Assim sendo, temos: $D_{p2p} = max\{F/u_s, F/d_i, NF/u_{total}\}$

Sabendo que: $F = 5Gb = 5368709120\ bits, u_s = 1\ Gbps = 1000000000\ bps, d_i = 100\ Mbps = 104857600\ bps$, podemos calcular a seguinte tabela:

Número de nodos	u_i	F/u_s	F/d_i	NF/u_{total}	u_{total}	D_{p2p}
10	1Mbps	53.6s	51s	53.15s	1010000000 bps	53.6s
100	1Mbps	536.8s	51s	488s	1100000000 bps	536.8s
1000	1Mbps	5368.7s	51s	2684.35s	2000000000 bps	5368.7s
10	5Mbps	53.6s	51s	51.13s	1050000000 bps	53.6s
100	5Mbps	536.8s	51s	358s	1500000000 bps	536.8s
1000	5Mbps	5368.7s	51s	894.78s	6000000000 bps	5368.7s
10	10Mbps	53.6s	51s	48.8s	1100000000 bps	53.6s
100	10Mbps	536.8s	51s	268.43s	2000000000 bps	536.8s
1000	10Mbps	5368.7s	51s	488,06s	11000000000 bps	5368.7s

Tabela 3: Tempo de distribuição de F

Pela tabela acima podemos verificar que numa arquitetura *peer-to-peer*, o tempo de upload do ficheiro quando ele ainda não existe no link de acesso é igual à arquitetura cliente-servidor, no entanto esse caso apenas ocorre uma vez. No caso em que os nodos participam na redistribuição do ficheiro, podemos verificar que o tempo de distribuição é bastante inferior, quão mais inferior quanto o maior débito de upload dos nodos e o seu número.

3.3. Conclusões

Analisando os resultados, podemos concluir que na arquitetura cliente-servidor o tempo de distribuição aumenta linearmente e sem limite à medida que o número de *peers* N aumenta. Ao passo que na arquitetura *peer-to-peer*, não só o tempo mínimo de distribuição é sempre inferior à arquitetura cliente-servidor, como também quanto maior o número de nodos e maior o seu débito de upload, mais eficiente esta arquitetura se torna. Ainda assim, esta melhoria não é linear, isto é, a certa altura, o aumento de nodos e do seu débito de upload não tornam o tempo de distribuição menor, este valor tende para um limite.

Assim, podemos verificar que, de facto, arquiteturas *peer-to-peer* apresentam escalabilidade própria, escalabilidade essa que é resultado dos nodos serem distribuidores e consumidores de bits.