Trabalho Prático Nº2 –

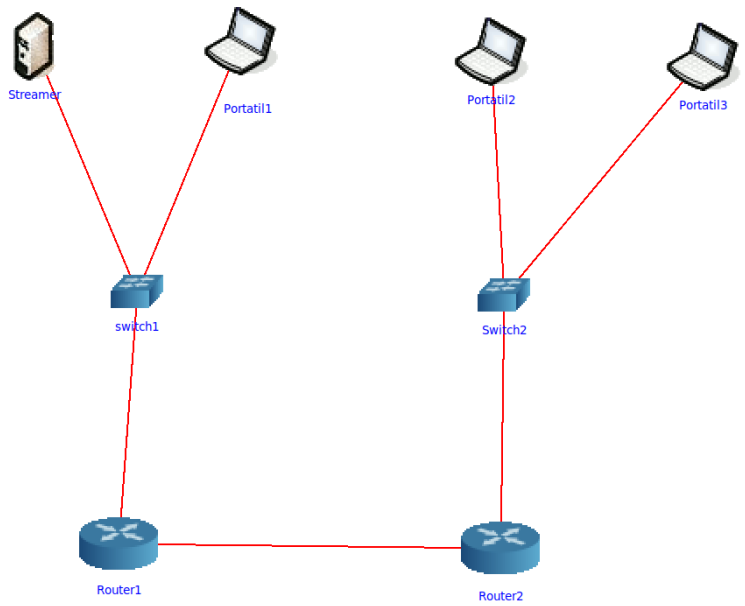
Eduardo Benjamim Lopes Coelho, Henrique Gabriel dos Santos Neto e Irenel Lopo da Silva
{pg47164, pg47238, pg42644}@alunos.uminho.pt

Universidade do Minho

Resumo Neste trabalho pretende-se experimentar várias soluções de streaming a pedido e em tempo real usando o emulador CORE como bancada experimental. O objetivo é em primeiro lugar perceber as opções disponíveis em termos de pilha protocolar e as diferenças conceptuais entre elas. Por outro lado, espera-se também alguma familiarização com os formatos multimédia e como se faz o seu empacotamento (apenas de forma superficial, dada a imensidão de possibilidades), bem como com as ferramentas open source disponíveis para uso.

Perguntas e Respostas

Topologia em Estudo



Questão 1

Capture três pequenas amostras de tráfego no link de saída do servidor, respectivamente com 1 cliente (*VLC*), com 2 clientes (*VLC* e *Firefox*) e com 3 clientes (*VLC*, *Firefox*

e *ffmeg*). Identifique a taxa em *bps* necessária (usando o $ffmpeg\ -i\ video1.mp4$ e/ou o próprio *wireshark*), o encapsulamento usado e o número total de fluxos gerados. Comente a escalabilidade da solução. Ilustre com evidências da realização prática do exercício (ex: capturas de ecrã)

A partir do comando $ffmpeg\ -i\ video1.mp4$ registamos que a *bitrate* para o nosso video de 49 segundos é de 62 kb/s. Para cada cliente, independente do software de recepção é aberto um fluxo de dados $HTTP$, sendo que desta forma cada *stream* ocorre sobre TCP, sendo que desta forma cada *chunk* (pedaço do vídeo) é encapsulado em OGG pelo *vlc*, seguida pelo cabeçalho $HTTP$ em que cada trama é por sua vez encapsulada num cabeçalho TCP. Com base nisto podemos concluir que embora seja prático implementar o *streaming* sobre $HTTP$ (visto que é um protocolo aplicacional bastante popular) não é fácil escalar esta solução, visto que para além de estarmos perante um design cliente-servidor que pode provocar quebras de serviços quando estamos perante um grande volume de utilizadores, os mecanismos de controlo de fluxo e congestão do protocolo TCP podem comprometer o serviço por parte do utilizador se este estiver sempre em situações de $buffering$ (o cliente está a aguardar pelos chunks de video).

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.0.0.20	42466	10.0.0.10	8080	3 887	3553k	1 502	99k	2 385	3454k	0.000000	61.6793	12k	448k

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	3932	100.0	3557290	461k	0	0	0
Ethernet	100.0	3932	1.5	55048	7138	0	0	0
Internet Protocol Version 6	0.3	10	0.0	400	51	0	0	0
User Datagram Protocol	0.1	2	0.0	16	2	0	0	0
Multicast Domain Name System	0.1	2	0.0	90	11	2	90	11
Open Shortest Path First	0.2	6	0.0	216	28	6	216	28
Internet Control Message Protocol v6	0.1	2	0.0	32	4	2	32	4
Internet Protocol Version 4	99.6	3918	2.2	78360	10k	0	0	0
Transmission Control Protocol	98.9	3887	96.2	3421652	443k	3460	2809294	364k
Hypertext Transfer Protocol	10.9	427	16.9	599610	77k	416	587575	76k
Malformed Packet	0.3	11	0.0	0	0	11	0	0
Open Shortest Path First	0.8	31	0.0	1364	176	31	1364	176
Address Resolution Protocol	0.1	4	0.0	112	14	4	112	14

Figura 1. Fluxos e Hierarquias - 1 cliente (VLC)

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.0.0.20	42466	10.0.0.10	8080	3 251	2880k	1 313	86k	1 938	2793k	1.559354	59.6667	11k	374k
10.0.1.20	56072	10.0.0.10	8080	3 206	2877k	1 266	83k	1 940	2793k	1.559456	59.6664	11k	374k

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	6501	100.0	5760518	752k	0	0	0
Ethernet	100.0	6501	1.6	91014	11k	0	0	0
Internet Protocol Version 6	0.1	7	0.0	280	36	0	0	0
Open Shortest Path First	0.1	7	0.0	252	32	7	252	32
Internet Protocol Version 4	99.8	6488	2.3	129760	16k	0	0	0
Transmission Control Protocol	99.3	6457	96.1	5537680	723k	5780	4566954	596k
Hypertext Transfer Protocol	10.4	677	16.5	951472	124k	657	928212	121k
Malformed Packet	0.3	20	0.0	0	0	20	0	0
Open Shortest Path First	0.5	31	0.0	1364	178	31	1364	178
Address Resolution Protocol	0.1	6	0.0	168	21	6	168	21

Figura 2. Fluxos e Hierarquias - 2 clientes (VLC e $Firefox$)

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.0.0.20	42466	10.0.0.10	8080	3 473	3114k	1 378	90k	2 095	3023k	0.000000	60.9581	11k	396k
10.0.1.20	56072	10.0.0.10	8080	3 401	3109k	1 305	86k	2 096	3023k	0.000271	60.9581	11k	396k
10.0.1.21	37300	10.0.0.10	8080	3 415	3113k	1 316	87k	2 099	3026k	0.002242	60.9567	11k	397k

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	10334	100.0	9341093	1209k	0	0	0
Ethernet	100.0	10334	1.5	144676	18k	0	0	0
Internet Protocol Version 6	0.1	6	0.0	240	31	0	0	0
Open Shortest Path First	0.1	6	0.0	216	27	6	216	27
Internet Protocol Version 4	99.9	10320	2.2	206400	26k	0	0	0
Transmission Control Protocol	99.6	10289	96.2	8987973	1163k	9187	7411437	959k
Hypertext Transfer Protocol	10.7	1102	16.5	1545646	200k	1063	1501219	194k
Malformed Packet	0.4	39	0.0	0	0	39	0	0
Open Shortest Path First	0.3	31	0.0	1364	176	31	1364	176
Address Resolution Protocol	0.1	8	0.0	224	29	8	224	29

Figura 3. Fluxos e Hierarquias - 3 clientes (*VLC*, *Firefox* e *ffplay*)

Todas as tramas foram encapsuladas no formato OGG de modo a haver maior compatibilidade entre browsers e dispositivos. O cliente *VLC* encontra-se no Laptop1 que possui o endereço 10.0.0.20, o cliente *firefox* encontra-se no Laptop2 que possui o endereço 10.0.1.20 e o cliente *ffplay* encontra-se no Laptop3 que possui o endereço 10.0.1.21.

Questão 2

Diga qual a largura de banda necessária, em bits por segundo, para que o cliente de streaming consiga receber o vídeo no firefox e qual a pilha protocolar usada neste cenário.

Tendo em conta a estrutura do protocolo DASH no caso em estudo, a largura de banda necessária para que o cliente assista o vídeo varia conforme a qualidade que este está a ver, ou seja conforme o ficheiro que esteja a receber. Para o tamanho de 180×120 pixeis, o vídeo está codificado para ter um débito de $207kbps$, para o tamanho de 360×240 pixeis o video apresenta uma *bitrate* de $521kbps$ enquanto que para o tamanho maior (540×360 pixeis) o bitrate necessário será aproximadamente $1018kbps$.

Adicionalmente, existe um *overhead* de dados e um possível atraso que é apresentado em cada trama de dados que é provocado pelo protocolo aplicacional ($HTTP$) e consequentemente pelo protocolo de transporte correspondente (TCP) que impacta bastante a continuidade de serviço devido aos mecanismos de controlo de fluxo e congestão que pode atrasar bastante os dados, e provocar *jitter* no serviço.

Durante o streaming em si o débito pode variar entre valores próximos aos três valores anteriores visto que o cliente indica qual dos formatos pretende ver conforme a sua conexão de forma a garantir a disponibilidade e continuidade do serviço ao longo do tempo.

Questão 3

Ajuste o débito dos links da topologia de modo que o cliente no portátil 2 exiba o vídeo de menor resolução e o cliente no portátil 1 exiba o vídeo com mais resolução. Mostre evidências.

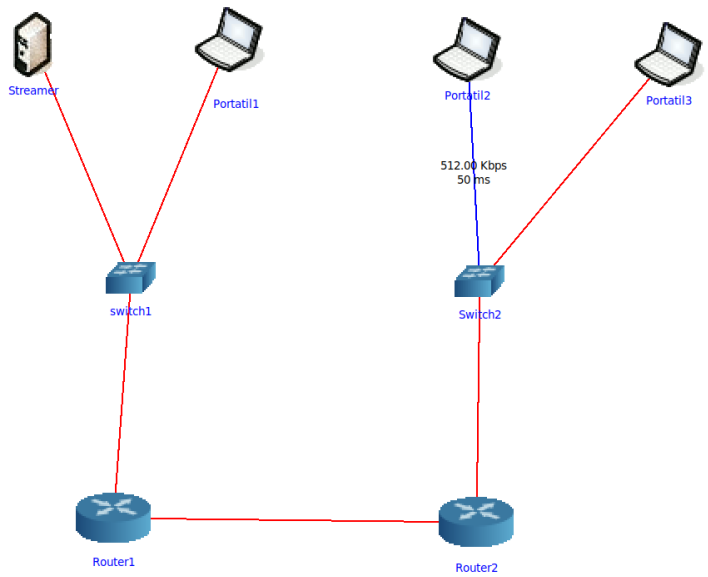


Figura 4. Topologia da rede com débito reduzido

Figura 5. Transição de formatos no browser do Laptop2

Questão 4

Descreva o funcionamento do DASH neste caso concreto, referindo o papel do ficheiro MPD criado.

O protocolo *DASH* sobre *http* tem o objetivo de permitir a um utilizador um acesso constante a um serviço de *streaming* mesmo em redes sujeitas a vários erros e atrasos a partir de vários dados e métodos de software. Para exemplificar melhor o funcionamento deste protocolo, a largura de banda do Laptop 2 foi limitada a 512kps de forma a visualizar melhor o padrão do protocolo nas amostras realizadas como é visivel na Figura 4.

Para garantir o serviço, o protocolo *DASH* consiste em disponibilizar um serviço com *adaptive bitrate* (*ABR*), ou seja um cliente poderá assistir ao mesmo fluxo de conteúdo a partir de vários débitos diferentes, o que resulta em conteúdo com qualidades diferentes. Um conteúdo destinado a este serviço é primeiramente sujeito a uma codificação de várias formas diferentes, que permitirão ao

sistema disponibilizar o conteúdo em vários *bitrates* diferentes, a partir de vários fatores, tais como: resoluções diferentes e compressões com perdas. No nosso caso, estes ficheiros correspondem aos vídeos $video2\_180\_120\_200k\_dash.mp4$, $video2\_360\_240\_500k\_dash.mp4$ e $video2\_540\_360\_1000k\_dash.mp4$ que originaram de uma recodificação do ficheiro $video2.mp4$ em várias resoluções diferentes.

De seguida os ficheiros anteriores são mapeados num ficheiro $MPEG-DASH$ (MPD), maioritariamente conhecido como *manifest*. O *manifest* consiste num ficheiro de texto com formato semelhante a um *XML*, que define e mapeia as respetivas divisões entre os vários formatos codificados. Este ficheiro é transferido pelo cliente antes de este aceder ao fluxo de dados e permite a este escolher um dado *chunk* (pedaço do vídeo) do ficheiro codificado conforme o bitrate desejado, a partir de um índice corresponde ao instante atual do vídeo no cliente. Como sugerido anteriormente, esta escolha do ficheiro e consequentemente da largura de banda (*bitrate*) usada ocorre conforme o estado da sessão do cliente (ou seja, o *delay*, o *jitter*, as perdas de informação, entre outros).

Estes comportamentos foram registados no caso em estudo. Como referido anteriormente, o Laptop 2 foi limitado de forma a não permitir larguras de banda superiores a $512kbps$. Desta forma foi feito um pedido $HTTP$ ao *streamer* para o ficheiro $video_dash.html$ que, por sua vez, efetuará outros pedidos $HTTP$. Estas comunicações foram capturadas com recurso ao *wireshark* e, por sua vez, foram filtradas as tramas $HTTP$ pois, para além de serem as únicas tramas com fácil interpretação humana, possuem a informação necessária para estudarmos o protocolo. Os resultados encontram-se na figura 6. O primeiro pedido relevante, correspondente ao $manifest$ presente na trama 22, ocorre, como referido anteriormente, antes da transmissão de conteúdo para permitir ao cliente escolher qual o ficheiro que possui o *bitrate* melhor para a sua situação. De seguida podemos ver que o cliente pediu uma trama de resolução média (360×180, trama 30), visto que em principio conseguiria com base na topologia atual cumprir os requisitos de $500kbps$ que este exige. Ora, isto não se verificou porque, rapidamente, o vídeo, no *browser*, entrou em *buffering*, visto que o segundo *chunk* não chegou rápido o suficiente para cumprir o débito de consumo do utilizador. Tendo isto em conta, o protocolo pediu pela resolução inferior à originalmente pedida (180×120, tramas 925, 1258 e 1591) e prosseguiu com esta até a transmissão terminar. Esta transição é visível na figura 5.

No.	Time	Source	Destination	Protocol	Length	Info
6	2.6080556…	10.0.1.20	10.0.0.10	HTTP	501	GET /video_dash.html HTTP/1.1
14	2.6824905…	10.0.1.20	10.0.0.10	HTTP	439	GET /dash.all.debug.js HTTP/1.1
22	2.7551432…	10.0.1.20	10.0.0.10	HTTP	502	GET /video_manifest.mpd HTTP/1.1
30	3.1394680…	10.0.1.20	10.0.0.10	HTTP	398	GET /video2_360_240_500k_dash.mp4 HTTP/1.1
925	18.405271…	10.0.1.20	10.0.0.10	HTTP	399	GET /video2_180_120_200k_dash.mp4 HTTP/1.1
1258	23.053328…	10.0.1.20	10.0.0.10	HTTP	400	GET /video2_180_120_200k_dash.mp4 HTTP/1.1
1591	27.711210…	10.0.1.20	10.0.0.10	HTTP	400	GET /video2_180_120_200k_dash.mp4 HTTP/1.1
8	2.6081904…	10.0.0.10	10.0.1.20	HTTP	272	HTTP/1.1 304 Not Modified
16	2.6826333…	10.0.0.10	10.0.1.20	HTTP	272	HTTP/1.1 304 Not Modified
24	2.7553244…	10.0.0.10	10.0.1.20	HTTP	273	HTTP/1.1 304 Not Modified
802	12.160989…	10.0.0.10	10.0.1.20	MP4	1496	
1233	22.121688…	10.0.0.10	10.0.1.20	MP4	841	
1566	26.769717…	10.0.0.10	10.0.1.20	MP4	841	

Figura 6. Captura de Tráfego HTTP DASH no Laptop 2

Questão 5

Compare o cenário unicast aplicado com o cenário multicast. Mostre vantagens e desvantagens na solução multicast ao nível da rede, no que diz respeito a escalabilidade (aumento do nº de clientes) e tráfego na rede. Tire as suas conclusões.

No cenário *multicast*, cada *frame* de vídeo e áudio do conteúdo a ser transmitido atravessa a rede uma vez, porque o servidor envia os dados para o *ipmulticast* da rede, que por sua vez envia os dados para cada cliente. Esta abordagem tem a desvantagem de, uma vez que os clientes estão a ver a mesma *stream* não têm acesso a opcões de controlo tais como pausar, avançar e retroceder. Por outro lado, tem a vantagem de utilizar menos débito e de gerar menos tráfego na rede.

O modelo de *streamingmulticast* não escala facilmente em redes heterogéneas nem na rede pública. O *multicast* assume que todos os clientes que estão a visualizar a *stream* têm a mesma capacidade de largura de banda e que estão a utilizar um dispositivo semelhante. Isto dificulta o escalamento para os clientes que têm uma largura de banda menor ou com muita variação, ou dispositivos diferentes do que a stream espera.

No cenário *unicast*, é enviada uma cópia do vídeo para cada cliente. Esta abordagem permite os utilizadores pausarem, retrocederem e avançarem no contéudo que estão a visualizar, no entanto, utiliza mais débito e gera bastante mais tráfego na rede podendo levar ao seu congestionamento e, eventualmente, perda de pacotes.

Deste modo, o modelo de *streamingunicast* é mais escalável, na medida em que permite que o conteúdo a ser transmitido se adapte à largura de banda e dispositivos dos clientes.

Por último, é importante considerar que configurar uma solução *multicast* é mais complexo que configurar uma solução *unicast*.

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.0.0.10	33388	224.0.0.100	5555	5 832	4399k	5 832	4399k	0	0	0.000000	162.2645	216k	0
10.0.0.10	56767	224.2.127.254	9875	33	12k	33	12k	0	0	1.864720	160.4318	602	0
10.0.0.10	33389	224.0.0.100	5556	32	2240	32	2240	0	0	1.864764	155.7509	115	0

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	5898	100.0	4413628	217k	0	0	0
Ethernet	100.0	5898	1.9	82572	4070	0	0	0
Internet Protocol Version 6	0.0	1	0.0	40	1	0	0	0
Internet Control Message Protocol v6	0.0	1	0.0	16	0	1	16	0
Internet Protocol Version 4	100.0	5897	2.7	117940	5813	0	0	0
User Datagram Protocol	100.0	5897	1.1	47176	2325	0	0	0
Session Announcement Protocol	0.6	33	0.2	10692	527	0	0	0
Session Description Protocol	0.6	33	0.2	9900	487	33	9900	487
Real-Time Transport Protocol	98.0	5779	93.5	4124711	203k	0	0	0
MP4V-ES	98.0	5779	91.9	4055363	199k	5779	4055363	199k
Real-time Transport Control Protocol	0.5	32	0.0	896	44	32	896	44
Data	0.9	53	0.7	29585	1458	53	29585	1458

Figura 7. Fluxos de Tramas UDP e Hierarquia de protocolos em Multicast

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.0.0.10	55051	10.0.1.21	5556	21	1470	21	1470	0	0	2.889710	100.4820	117	
10.0.0.10	55050	10.0.1.21	5555	3 903	3095k	3 903	3095k	0	0	2.889761	102.3939	241k	

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	4011	100.0	3108799	225k	0	0	0
Ethernet	100.0	4011	1.8	56154	4079	0	0	0
Internet Protocol Version 6	0.4	15	0.0	600	43	0	0	0
User Datagram Protocol	0.0	2	0.0	16	1	0	0	0
Multicast Domain Name System	0.0	2	0.0	282	20	2	282	20
Open Shortest Path First	0.3	11	0.0	396	28	11	396	28
Internet Control Message Protocol v6	0.0	2	0.0	32	2	2	32	2
Internet Protocol Version 4	99.5	3990	2.6	79800	5797	0	0	0
User Datagram Protocol	97.8	3924	1.0	31392	2280	0	0	0
Real-time Transport Control Protocol	0.5	21	0.0	588	42	21	588	42
Data	97.3	3903	94.3	2931839	212k	3903	2931839	212k
Open Shortest Path First	1.4	56	0.1	2464	179	56	2464	179
Internet Control Message Protocol	0.2	10	0.2	5068	368	0	0	0
Real-time Transport Control Protocol	0.0	1	0.0	28	2	1	28	2
Data	0.2	9	0.2	4680	340	9	4680	340
Address Resolution Protocol	0.1	6	0.0	168	12	6	168	12

Figura 8. Fluxos de Tramas UDP e Hierarquia de protocolos em Unicast

Conclusões

Ao experimentar várias soluções de *streaming*, conseguimos aprofundar o nosso conhecimento do funcionamento de aplicações de *streaming* de conteúdo multimédia.

Na verdade, ao utilizar diferentes aplicações tais como o VLC e o $ffmpeg$ conseguimos perceber as opções disponíveis em termos de pilha protocolar e as suas diferenças conceptuais. Para além disso, aprendemos novas funcionalidades da nossa conhecida aplicação VLC.

A primeira parte demonstrou-nos os problemas que podem ocorrer quando se faz streaming $HTTP$ simples sem adaptação dinâmica de débito.

Numa segunda parte, conseguimos verificar o funcionamento do protocolo $DASH$ e o modo como possibilita a adaptação do serviço de *streaming* $HTTP$ simples às condições do cliente, com recurso ao ficheiro $MPEG-DASH(MPD)$.

Por último, ao testar o streaming $RTP/RTCP unicast$ sobre UDP e *multicast* com anúnicos SAP fomos capazes de compreender muito melhor as diferenças entre estes dois cenários, tal como as várias vantagens e desvantagens de cada um.

Em suma, este trabalho foi uma excelente oportunidade de concretizar todos os conhecimentos teóricos sobre os serviços de *streaming* de multimédia, exemplificando as diferenças entre as várias opções da pilha protocolar e modos de *streaming*.