Universidade do Minho
Escola de Engenharia

Trabalho Prático 1

Streaming de Áudio e Vídeo a Pedido e em Tempo Real

Engenharia de Serviços em Rede
Mestrado em Engenharia Informática

PL7 - Grupo 3

Gabriela Santos Ferreira da Cunha - pg53829

Millena de Freitas Santos - pg54107

Nuno Guilherme Cruz Varela - pg54117



outubro, 2023

Conteúdo

1 Streaming HTTP simples sem adaptação dinâmica de débito

1.1 Questão 1

Capture três pequenas amostras de trágefo no link de saída do servidor, respetivamente com 1 cliente (VLC), com 2 clientes (VLC e Firefox) e com 3 clientes (VLC, Firefox e ffplay). Identifique a taxa em bps necessária (usando o ffmpeg -i videoA.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).

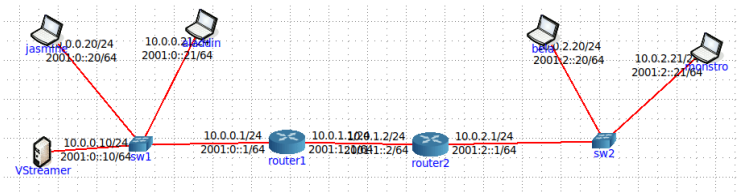


Figura 1: Topologia CORE utilizada.

Neste exercício, o servidor "VStreamer" foi utilizado para fazer *stream* por HTTP, através do VLC, de um ficheiro de vídeo previamente gravado. Três pequenas amostras foram retiradas do *link* de saída do "streamer" com a ajuda de 3 clientes (VLC, Firefox e ffplay).

Taxa de *bps* necessária

Através do *ffmpeg*, verificamos que a largura de banda necessária para o *streaming* do vídeo com áudio é 26 *kbps*, ou seja, 26000 *bps*, enquanto que apenas para o vídeo é 24 *kbps*.

```
core@xubuncore:~$ ffmpeg -i videoA.mp4 -hide_banner
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'videoA.mp4':
  Metadata:
    major_brand     : isom
    minor_version   : 512
    compatible_brands: isomiso2avc1mp41
    encoder         : Lavf58.29.100
  Duration: 00:00:09.55, start: 0.000000, bitrate: 26 kb/s
    Stream #0:0(und): Video: h264 (High) (avc1 / 0x31637661), yuv420p, 200x150, 24 kb/s, 20 fps, 20 tbr, 1
240 tbn, 40 tbc (default)
    Metadata:
      handler_name    : VideoHandler
```

Figura 2: Taxa de transmissão do vídeo A utilizando *ffmpeg*.

Em alternativa, recorremos ao *wireshark* e fizemos uso das funcionalidades "Conversations" e "Protocol Hierarchy" que nos permitem visualizar estatísticas detalhadas sobre as conversas e os protocolos presentes na captura, respetivamente.

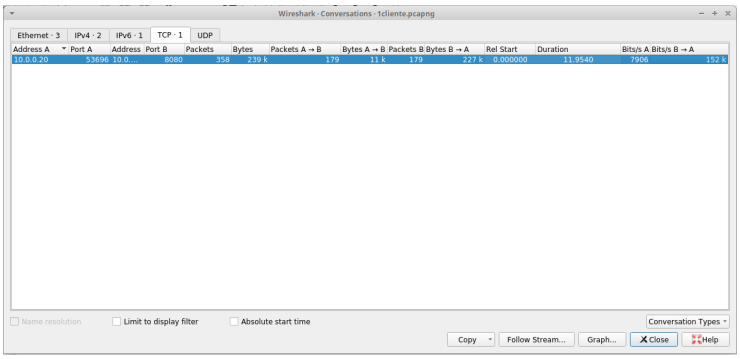


Figura 3: Conversas presentes no *streaming* com 1 cliente.

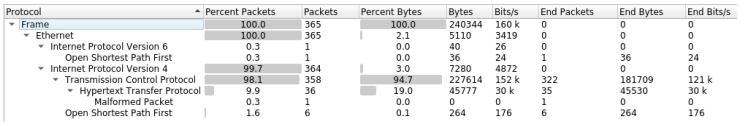


Figura 4: Protocolos presentes no *streaming* com 1 cliente.

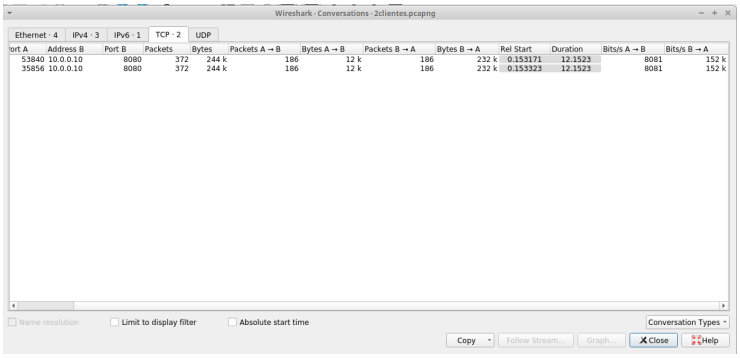


Figura 5: Conversas presentes no *streaming* com 2 clientes.

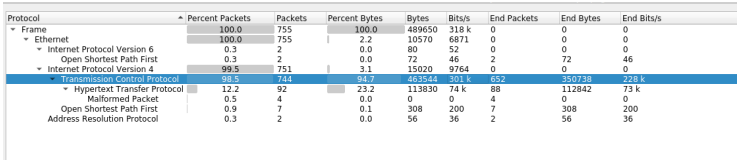


Figura 6: Protocolos presentes no *streaming* com 2 clientes.

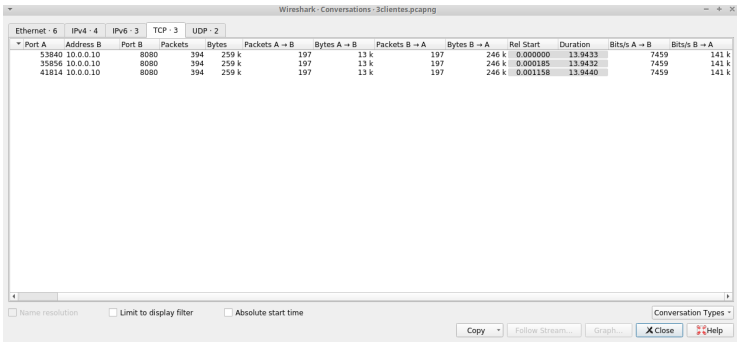


Figura 7: Conversas presentes no *streaming* com 3 clientes.

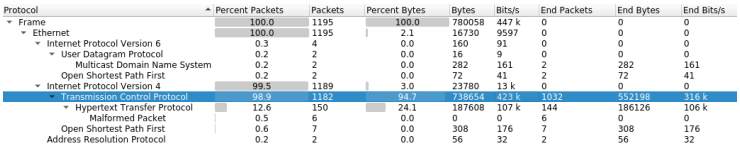


Figura 8: Protocolos presentes no *streaming* com 3 clientes.

As informações recolhidas encontram-se na tabela seguinte:

Número de clientes	Taxa p/ cliente	Taxa total
1	152 *kbps*	152 *kbps*
2	152 *kbps*	304 *kbps*
3	141 *kbps*	423 *kbps*

Tabela 1: Taxas de transmissão recolhidas utilizando o *wireshark*.

Tendo em conta a taxa obtida pelo *ffmpeg* e a taxa por cliente, na tabela 1, verificamos que o valor obtido pela primeira ferramenta é disperso dos restantes. Ao utilizarmos o *ffmpeg* obtemos uma taxa teórica de transmissão, presente nos metadados do ficheiro a ser transferido. Por outro lado, ao capturarmos o tráfego no *wireshark*, obtemos uma taxa real que resulta da média do débito capturado, este que, naturalmente, não é constante ao longo do tempo, daí a taxa não ser exatamente igual nas diferentes capturas.

Para além disso, observamos também que o primeiro valor é menor do que os que são obtidos nas capturas. Isto pode dever-se ao facto de o *wireshark* medir o tráfego na rede ao nível do pacote, capturando *overheads* adicionais dos protocolos de transporte nestes mesmos pacotes. Estes protocolos adicionam cabeçalhos que, ainda que não estejam diretamente relacionados com o conteúdo que é enviado, contribuem para a taxa de *bits* observada, resultando num valor maior.

Outro aspeto que é de notar é o aumento linear da taxa de transmissão total consoante o número de conexões estabelecidas ao servidor.

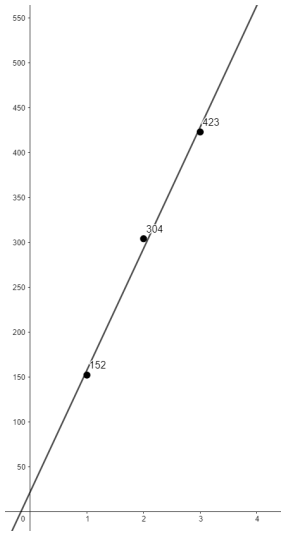


Figura 9: Aumento da taxa de transmissão consoante o número de conexões.

Encapsulamento usado

Através da funcionalidade "Protocol Hierarchy" não só retiramos informação relativa às taxas como também aos protocolos. Pelas figuras 4, 6 e 8, visualizamos os vários protocolos da pilha protocolar. Na camada de enlace é utilizado o protocolo *Ethernet II*, na camada de rede o IPv4 (*Internet Protocol Version 4*), na camada de transporte o TCP (*Transmission Control Protocol*) e na camada de aplicação o HTTP (*HyperText Transfer Protocol*).

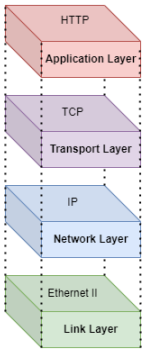


Figura 10: Representação da pilha protocolar.

Número total de fluxos gerados

De modo a comprovar o número total de fluxos gerados, utilizamos o filtro *tcp.stream eq*. Pela figura seguinte, observamos que o número de fluxos gerados é 3, o que era de esperar com 3 clientes.

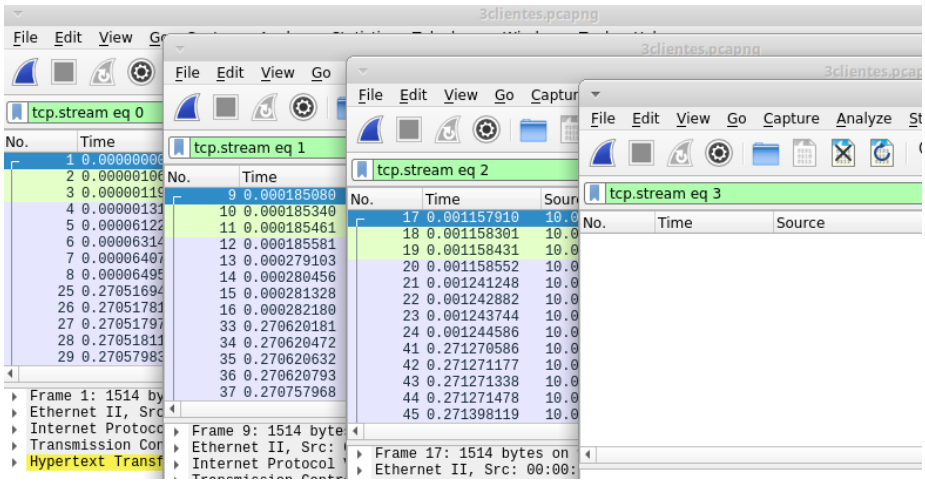


Figura 11: Número total de fluxos gerados com 3 clientes.

Escalabilidade da solução

Efetuado o estudo sobre o impacto da adição de clientes no tráfego da rede, concluímos que esta solução de *streaming* não é boa em termos de escalabidade, uma vez que, como podemos verificar pelos *screenshots* anexados acima, a taxa de transmissão aumenta linearmente com o número de conexões, visto que estamos perante uma conexão *unicast*. Desta forma, quantos mais clientes estabelecerem conexão com o servidor, maior terá de ser a largura de banda necessária para suportar os vários pedidos.

No modelo *unicast*, os dados são enviados de um único remetente para um único destinatário, pelo que os pacotes são endereçados explicitamente para um dispositivo. Desta maneira, ao utilizar esta forma de tráfego em *streaming* para vários clientes, iremos estar a enviar os mesmos dados em diferentes pacotes, o que originará atrasos no tráfego com o aumento de clientes.

2 Streaming adaptativo sobre HTTP (MPEG-DASH)

2.1 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.

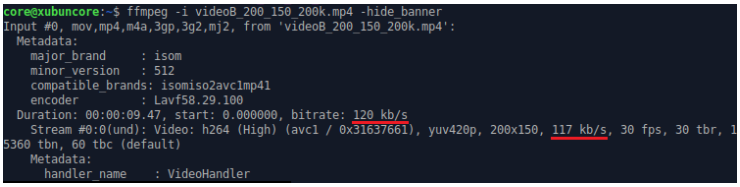


Figura 12: Taxa de transmissão do vídeo B utilizando *ffmpeg*.

A largura de banda necessária de acordo com o comando *ffmpeg* para o *streaming* do vídeo com áudio é 120 *kbps*, enquanto que apenas para o vídeo é 117 *kbps*. Este valor não leva em consideração todo o encapsulamento e a pilha protocolar que acaba por exigir uma maior largura de banda por conta de toda a informação adicional para fins de controlo, segurança, entre outros presentes em cada *header*.

Em relação à pilha protocolar, esta é idêntica à do primeiro vídeo, estando representada na figura 10.

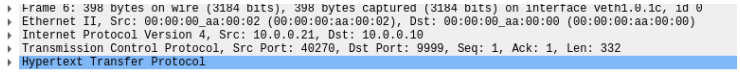


Figura 13: Pilha protocolar do vídeo B.

2.2 Questão 3

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

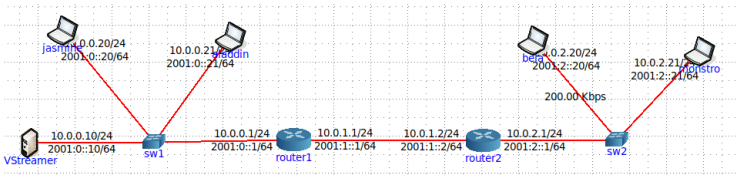


Figura 14: Topologia com o débito do *link* para o portátil Bela ajustado.

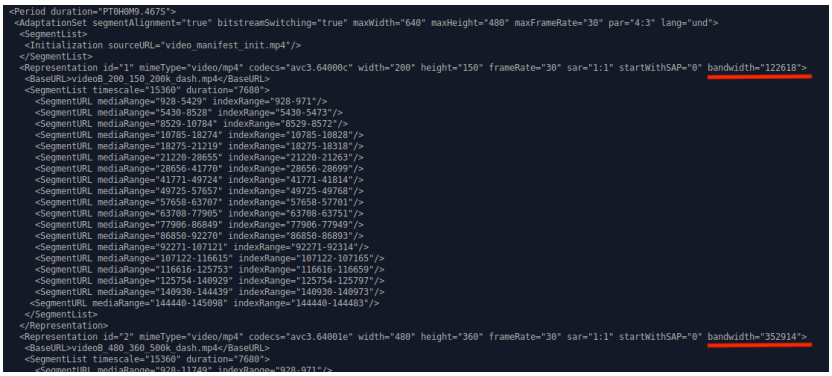


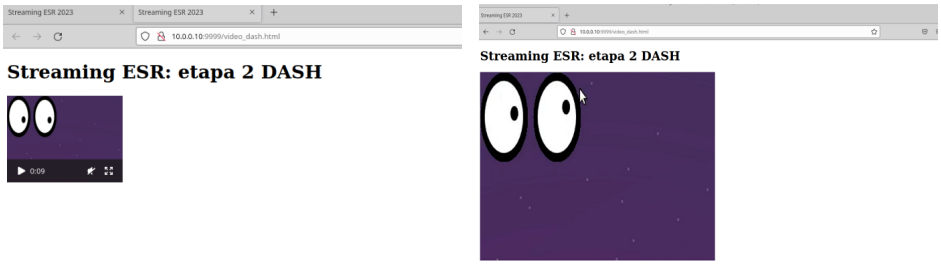
Figura 15: Larguras de banda do vídeo com menor e média resolução.

276	14.312524642	10.0.2.20	10.0.0.10	HTTP	398 GET /videoB_200_150_200k_dash.mp4 HTTP/1.1
744	23.798645413	10.0.0.10	10.0.2.20	MP4	570
867	27.375126142	10.0.0.10	10.0.2.20	MP4	862
876	27.647780613	10.0.2.20	10.0.0.10	HTTP	400 GET /videoB_200_150_200k_dash.mp4 HTTP/1.1
978	29.525759932	10.0.2.20	10.0.0.10	HTTP	399 GET /videoB_200_150_200k_dash.mp4 HTTP/1.1
1176	33.959162217	10.0.0.10	10.0.2.20	MP4	570
1190	34.250931796	10.0.2.20	10.0.0.10	HTTP	400 GET /videoB_200_150_200k_dash.mp4 HTTP/1.1
1272	35.822174516	10.0.2.20	10.0.0.10	HTTP	402 GET /videoB_200_150_200k_dash.mp4 HTTP/1.1

Figura 16: Pedidos HTTP GET para o vídeo de menor resolução.

27	28.568611033	10.0.0.21	10.0.0.10	HTTP	425 GET /video_dash.html HTTP/1.1
29	28.568837191	10.0.0.10	10.0.0.21	HTTP	556 HTTP/1.1 200 Ok (text/html)
35	29.014376025	10.0.0.21	10.0.0.10	HTTP	363 GET /dash.all.debug.js HTTP/1.1
2248	29.165488106	10.0.0.10	10.0.0.21	HTTP	254 HTTP/1.1 200 Ok (application/x-javascript)
2255	29.528333150	10.0.0.21	10.0.0.10	HTTP	379 GET /favicon.ico HTTP/1.1
2257	29.528456174	10.0.0.10	10.0.0.21	HTTP	741 HTTP/1.1 404 Not Found (text/html)
2264	29.557612619	10.0.0.21	10.0.0.10	HTTP	426 GET /video_manifest.mpd HTTP/1.1
2270	29.557819392	10.0.0.10	10.0.0.21	HTTP/X…	271 HTTP/1.1 200 Ok
2279	30.963966133	10.0.0.21	10.0.0.10	HTTP	369 GET /video_manifest_init.mp4 HTTP/1.1
2281	30.964196711	10.0.0.10	10.0.0.21	MP4	1060
2817	31.158918158	10.0.0.21	10.0.0.10	HTTP	401 GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
3346	31.168708459	10.0.0.10	10.0.0.21	MP4	1349
3373	31.425734711	10.0.0.21	10.0.0.10	HTTP	401 GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
3887	31.436072262	10.0.0.10	10.0.0.21	MP4	1349
3894	31.493262859	10.0.0.21	10.0.0.10	HTTP	401 GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
4375	31.510835416	10.0.0.10	10.0.0.21	MP4	1349
4382	31.550969198	10.0.0.21	10.0.0.10	HTTP	401 GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
4930	31.564100065	10.0.0.10	10.0.0.21	MP4	1349
4952	31.597494793	10.0.0.21	10.0.0.10	HTTP	402 GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1
5572	31.610819910	10.0.0.10	10.0.0.21	MP4	1349
5584	31.647189279	10.0.0.21	10.0.0.10	HTTP	403 GET /videoB_640_480_1000k_dash.mp4 HTTP/1.1

Figura 17: Pedidos HTTP GET para o vídeo de maior resolução.



(a) *Streaming* do vídeo de menor resolução. (b) *Streaming* do vídeo de maior resolução.

Figura 18: *Streamings* dos vídeos de menor e maior resolução.

De acordo com a figura 15, o ficheiro MPD correspondente ao vídeo, a largura de banda mínima estimada para o vídeo de menor resolução é de 122618 *bps*, enquanto que para o da próxima resolução disponível é de 352914 *bps*. Portanto, como estes valores são uma estimativa, limitou-se a largura de banda a 200 *kbps*, de modo a ser um valor superior ao de menor resolução e inferior ao da subsequente resolução disponível. Desta forma, o cliente opta pelo vídeo com menor resolução, como é possível observar na captura obtida no *wireshark* na figura 16.

Quanto ao Aladdin, a largura de banda por defeito no *core* consta como *unlimited*, pelo qual o cliente irá utilizar o de maior qualidade disponível, como se pode verificar na figura 17.

2.3 Questão 4

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

O DASH é uma tecnologia de *streaming* que permite a adaptação dinâmica da qualidade com base na largura de banda disponível. Para isso, utiliza um arquivo MPD que descreve as diferentes versões do conteúdo, incluindo detalhes como largura de banda necessária, resolução, *bitrate*, *codecs* e tipos MIME.

O processo de reprodução inicia com o cliente DASH a solicitar o arquivo MPD ao servidor de *streaming*. Após analisar o MPD, o cliente escolhe a versão do vídeo a ser reproduzida, considerando a largura de banda disponível. A transmissão do vídeo ocorre em segmentos, permitindo que o cliente se adapte à largura de banda em tempo real e mantenha um *buffer* adequado. Na página HTML é referenciado este ficheiro MPD e, com isto, o *browser*, tendo em conta a largura de banda que tem disponível, escolhe a *stream* com a melhor resolução que consegue reproduzir.

O MPD é fundamental neste processo pois fornece informações críticas para a escolha da qualidade do vídeo e garante uma experiência de *streaming* suave e de alta qualidade para o espectador. É importante notar que a especificação MPEG-DASH se concentra na definição do MPD e dos formatos de segmento, enquanto a entrega e o comportamento do cliente estão fora do seu alvo.

3 Streaming RTP/RTCP unicast sobre UDP e multicast com anúnicos SAP

3.1 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.

Quando estamos perante o *Internet Protocol*, são detetados três modos distintos sobre os quais os *hosts* podem comunicar entre si: *unicast*, *multicast* e *broadcast*. Neste exercício abordaremos especificamente as comunicações *unicast* e *multicast*.

Unicast

Assim como foi concluído na secção 1, no modelo *unicast* os dados são enviados de um único remetente para um único destinatário. Quando pretendemos servir múltiplos clientes, este método de tráfego não se revela ótimo em termos de escalabilidade, visto que consome uma elevada largura de banda pelo facto de os pacotes serem enviados separadamente para cada destinatário.

Multicast

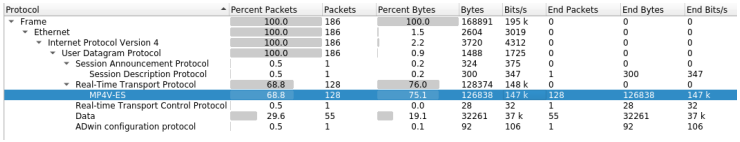
No método *multicast*, ao contrário do *unicast*, os dados são enviados de um único remetente para vários destinatários. O servidor envia os dados para um grupo *multicast* identificado por um endereço *multicast* especial. Os clientes necessitam de se juntar a este grupo, de modo a receberem os dados. Estes dados são enviados para um *switch*, pelo que o *streamer* descarta a responsabilidade para este dispositivo que irá enviar os dados para todo o grupo.

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	151	100.0	145391	242 k	0	0	0
Ethernet	100.0	151	1.5	2114	3521	0	0	0
Internet Protocol Version 4	100.0	151	2.1	3020	5030	0	0	0
User Datagram Protocol	100.0	151	0.8	1208	2012	0	0	0
Session Announcement Protocol	0.7	1	0.2	324	539	0	0	0
Session Description Protocol	0.7	1	0.2	300	499	1	300	499
Real-Time Transport Protocol	89.4	135	90.6	131726	219 k	0	0	0
MP4V-ES	89.4	135	89.5	130106	216 k	135	130106	216 k
Real-time Transport Control Protocol	0.7	1	0.0	28	46	1	28	46
Data	9.3	14	4.8	6971	11 k	14	6971	11 k

Figura 19: *Multicast* sem clientes.

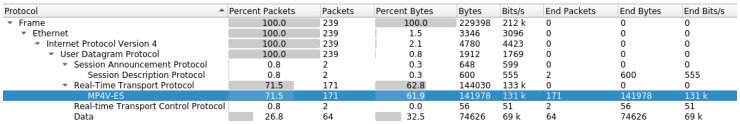
Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	183	100.0	155552	194 k	0	0	0
Ethernet	100.0	183	1.6	2562	3205	0	0	0
Internet Protocol Version 6	0.5	1	0.0	40	50	0	0	0
Internet Control Message Protocol v6	0.5	1	0.0	16	20	1	16	20
Internet Protocol Version 4	99.5	182	2.3	3648	4564	0	0	0
User Datagram Protocol	98.4	180	0.9	1440	1801	0	0	0
Session Announcement Protocol	1.1	2	0.4	648	810	0	0	0
Session Description Protocol	1.1	2	0.4	600	750	2	600	750
Real-Time Transport Protocol	85.2	156	83.2	129375	161 k	0	0	0
MP4V-ES	85.2	156	82.0	127503	159 k	156	127503	159 k
Real-time Transport Control Protocol	1.1	2	0.0	56	70	2	56	70
Data	10.9	20	11.4	17735	22 k	20	17735	22 k
Internet Group Management Protocol	1.1	2	0.0	32	40	2	32	40

Figura 20: *Multicast* com 1 cliente.



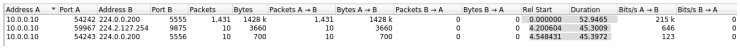
Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	186	100.0	168891	195 k	0	0	0
Ethernet	100.0	186	1.5	2604	3019	0	0	0
Internet Protocol Version 4	100.0	186	2.2	3720	4312	0	0	0
User Datagram Protocol	100.0	186	0.9	1488	1725	0	0	0
Session Announcement Protocol	0.5	1	0.2	324	375	0	0	0
Session Description Protocol	0.5	1	0.2	300	347	1	300	347
Real-Time Transport Protocol	68.8	128	76.0	128374	148 k	0	0	0
MP4V-ES	68.8	128	75.1	126838	147 k	128	126838	147 k
Real-time Transport Control Protocol	0.5	1	0.0	28	32	1	28	32
Data	29.6	55	19.1	32261	37 k	55	32261	37 k
ADwin configuration protocol	0.5	1	0.1	92	106	1	92	106

Figura 21: *Multicast* com 2 clientes.



Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	239	100.0	229398	212 k	0	0	0
Ethernet	100.0	239	1.5	3346	3096	0	0	0
Internet Protocol Version 4	100.0	239	2.1	4780	4423	0	0	0
User Datagram Protocol	100.0	239	0.8	1912	1769	0	0	0
Session Announcement Protocol	0.8	2	0.3	648	599	0	0	0
Session Description Protocol	0.8	2	0.3	600	555	2	600	555
Real-Time Transport Protocol	71.5	171	62.8	144030	133 k	0	0	0
MP4V-ES	71.5	171	61.9	141978	131 k	171	141978	131 k
Real-time Transport Control Protocol	0.8	2	0.0	56	51	2	56	51
Data	26.8	64	32.5	74626	69 k	64	74626	69 k

Figura 22: *Multicast* com 3 clientes.



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	54242	224.0.0.200	5555	1,431	1428 k	1,431	1428 k	0	0	0.000000	52.9465	215 k	0
10.0.0.10	59967	224.2.127.254	9875	10	3660	10	3660	0	0	4.200604	45.3009	646	0
10.0.0.10	54243	224.0.0.200	5556	10	700	10	700	0	0	4.548431	45.3972	123	0

Figura 23: Fluxos *Multicast*.

Número de clientes	Taxa de transmissão
0	216 *kbps*
1	159 *kbps*
2	147 *kbps*
3	131 *kbps*

Tabela 2: Taxas de transmissão recolhidas utilizando o *wireshark*.

Analisando a informação da tabela 2, podemos concluir que a taxa de transmissão se mantém praticamente constante, o que era expectável. Tal acontece porque no tráfego *multicast* os pacotes são enviados apenas uma vez na rede, ao contrário da forma *unicast* em que os pacotes são enviados tantas vezes quanto o número de clientes. Isto pode ser comprovado pela figura 23, onde existe apenas 1 fluxo no qual há transmissão de pacotes referentes ao vídeo. Disto podemos concluir que a comunicação *multicast* é mais eficiente em termos de largura de banda e de sobrecarga na rede.

Por outro lado, ao utilizarmos o UDP como protocolo de transporte, as comunicações não possuem mecanismo de correção de erros, o que torna a solução não fiável, deixando a camada aplicacional responsável por tal encargo. Ademais, o protocolo também não possui controlo de congestão na rede, algo que é assegurado pelo TCP. Justamente por isto, o UDP tem um *overhead* muito baixo em comparação com o TCP. Este também não estabelece conexão antes de enviar os dados, o que o torna mais rápido, conseguindo garantir o *streaming* para clientes com baixa largura de banda. O protocolo UDP é, assim, mais popular entre plataformas de *streaming* em HTTP, onde não é estritamente necessário que o vídeo carregue perfeitamente na primeira vez.