

Terceiro trabalho prático (Wireshark)

Rafael Begnini de Castilhos

19 de fevereiro de 2022

Resumo

O presente relatório possui como objetivo estudar, demonstrar, monitorar e aplicar na prática conceitos de gerenciamento de rede com o Wireshark, com o fito de prover ao leitor a base necessária para entendimento do uso desse *softwares* e replicar os experimentos realizados. Utilizando os conhecimentos adquiridos em aula e em pesquisas externas para monitorar o tráfego de uma rede local, com ênfase na captura de pacotes do protocolo HTTP.

Sumário

1	Introdução	2
2	Funcionamento do protocolo HTTP	2
2.1	Começando a conexão	2
2.2	Transferindo dados	3
2.3	Finalizando a conexão	3
3	Desenvolvimento do experimento com protocolo HTTP	3
3.1	Início da conexão HTTP	4
3.2	Tráfego da conexão	4
3.3	Encerrando conexão HTTP	5
4	Funcionamento do protocolo UDP e experimento	5
4.1	Detalhamento protocolo UDP	5
4.2	Desenvolvimento do experimento com protocolo UDP	6

1 Introdução

A ferramenta escolhida para realizar o trabalho é o Wireshark, devido à maior familiaridade com a ferramenta desde a elaboração do segundo trabalho prático da disciplina, além de possuir filtros que facilitam a gerência e visualização. O Wireshark é gratuito e permite monitorar, filtrar e identificar o tráfego de pacotes na rede onde está instalado. Além de possuir um formato próprio de arquivos que nos permite exportar as operações de monitoramento para serem carregadas em outras instâncias do programa [5].

No decorrer do relatório serão apresentadas as telas capturas, realizando a identificação dos pacotes relacionados à criação, tráfego e liberação de conexões dos protocolos HTTP que é a base de qualquer troca de dados na Web e um protocolo cliente-servidor, e também UDP que provê serviço de entrega não orientado a conexão.

Uma conexão é controlada na camada de transporte, e portanto fundamentalmente fora do controle do HTTP. Dentre os dois protocolos de transporte mais comuns na internet, o TCP é confiável e o UDP não. Portanto, o HTTP utiliza o padrão TCP, que é baseado em conexão, mesmo que nem sempre seja obrigatório o uso de uma conexão [3].

2 Funcionamento do protocolo HTTP

2.1 Começando a conexão

Adentrando os detalhes protocolo TCP, o estabelecimento da conexão é feito com um processo chamado *three-way-handshake* onde são enviadas 3 mensagens de sincronização em ambos os sentidos, configurando um "aperto de mão de três vias". Durante esse processo, são trocadas informações importantes para o funcionamento da comunicação, como o *checksum*, o *timestamp* e outros.

Os 3 passos são: O cliente envia um pacote com a flag SYN ativa; Posteriormente o servidor responde com um pacote com as flags SYN somado com ACK; E por fim o cliente responde com um pacote com a flag ACK [1].

2.2 Transferindo dados

[4] O TCP/IP é o principal protocolo de envio e recebimento de dados. TCP significa *Transmission Control Protocol* (Protocolo de Controle de Transmissão) e o IP, *Internet Protocol* (Protocolo de Internet). Na realidade, o TCP/IP é um conjunto de protocolos. Esse grupo é dividido em quatro camadas: aplicação, transporte, rede e interface. Cada uma delas é responsável pela execução de tarefas distintas. Essa divisão em camadas é uma forma de garantir a integridade dos dados que trafegam pela rede.

A camada de aplicação é utilizada pelos programas para enviar e receber informações de outros programas através da rede. Utilizando protocolos como SMTP (para email), FTP (transferência de arquivos) e também o HTTP.

A camada de transporte é responsável por receber os dados enviados pelo grupo acima, verificar a integridade deles e dividi-los em pacotes. Depois essas informações são enviadas a camada de rede.

Na camada de rede, os dados empacotados são recebidos e anexados ao endereço virtual (IP) do computador remetente e do destinatário. Agora é a vez dos pacotes serem, enfim, enviados pela internet. Para isso, são passados para a camada interface.

A tarefa da camada de interface é receber e enviar pacotes pela rede. Os protocolos utilizados nessa camada dependem do tipo de rede.

2.3 Finalizando a conexão

O término da comunicação bem sucedida, que seguiu corretamente os passos do TCP/IP, termina da seguinte forma: O cliente envia um segmento com a flag FIN; O servidor responde com um segmento de confirmação, perguntando se o cliente realmente deseja finalizar a conexão; Caso sim, o servidor envia uma mensagem com o segmento SYN/ACK para o cliente; Por fim, o cliente envia ACK para o servidor [1].

3 Desenvolvimento do experimento com protocolo HTTP

No desenvolvimento do experimento, foi optado por utilizar como alvo um website, com objetivo de obter uma visualização melhor do passo a passo na comunicação entre o computador local e o servidor de aplicação que executa

o website. O site utilizado foi o Youtube, que além de utilizar HTTPS, com protocolo TLS na porta 433, também usa o websocket e outros protocolos para streaming de dados. Aqui iremos focar apenas na transmissão HTTP via TCP.

3.1 Início da conexão HTTP

Ao acessar o website, é iniciado o processo do three-way-handshake. No caso abaixo os endereços IPs estão na versão 6. O cliente (192.168.0.80) envia um SYN para o servidor (172.217.173.99). O servidor responde com SYN/ACK e por fim o cliente responde com ACK. Essa troca de mensagens pode ser vista na figura 1.

14.340322296	192.168.0.80	172.217.173.99	TCP	74 37786 - 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1535222553 TSecr=0 WS=128
14.356204016	172.217.173.99	192.168.0.80	TCP	74 443 - 37786 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1430 SACK_PERM=1 TSval=1347042177 TSecr=1535222553 WS=256
14.356234360	192.168.0.80	172.217.173.99	TCP	66 37786 - 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1535222570 TSecr=1347042177

Figura 1: Estabelecimento de conexão.

3.2 Tráfego da conexão

Após ter a comunicação estabelecida pelo protocolo HTTP, a transmissão de dados pelas duas fontes pode ser realizada por diversos protocolos, aqui foi escolhido o HTTP, por ser o protocolo utilizado pelo Youtube e outras plataformas. O TLS em si possui uma camada de criptografia e um handshake separado além do que já foi realizado pelo TCP.

Após finalizados os procedimentos de segurança os dados podem ser trafegados no mesmo modelo do protocolo HTTP comum, com *headers* e um *body*, além de outros metadados. O TLS possui um *keep-alive*, então múltiplas transmissões de dados podem ser realizadas a partir de um mesmo handshake. Na figura 2 podemos ver os pacotes que foram utilizados desde o início do handshake TLS até as transmissões de dados realizadas.

14.358413634	192.168.0.80	172.217.173.99	TLSv1.3	583 Client Hello
14.37637572	172.217.173.99	192.168.0.80	TCP	66 443 - 37786 [ACK] Seq=1 Ack=518 Win=66816 Len=0 TSval=1347042196 TSecr=1535222571
14.438363899	172.217.173.99	192.168.0.80	TLSv1.3	1484 Server Hello, Change Cipher Spec
14.438393860	192.168.0.80	172.217.173.99	TCP	66 37786 - 443 [ACK] Seq=518 Ack=519 Win=64128 Len=0 TSval=1535222651 TSecr=1347042347
14.438618763	172.217.173.99	192.168.0.80	TCP	1484 443 - 37786 [PSH, ACK] Seq=1419 Ack=518 Win=66816 Len=1418 TSval=1347042247 TSecr=1535222571 [TCP segment of a reassembled PDU]
14.438630872	192.168.0.80	172.217.173.99	TCP	66 37786 - 443 [ACK] Seq=518 Ack=2037 Win=64128 Len=0 TSval=1535222651 TSecr=1347042247
14.438694917	172.217.173.99	192.168.0.80	TCP	1484 443 - 37786 [ACK] Seq=2037 Ack=518 Win=66816 Len=1418 TSval=1347042247 TSecr=1535222571 [TCP segment of a reassembled PDU]
14.438695996	192.168.0.80	172.217.173.99	TCP	66 37786 - 443 [ACK] Seq=518 Ack=2595 Win=64128 Len=0 TSval=1535222651 TSecr=1347042247
14.439124348	172.217.173.99	192.168.0.80	TLSv1.3	404 Application Data
14.439129748	192.168.0.80	172.217.173.99	TCP	66 37786 - 443 [ACK] Seq=518 Ack=4683 Win=63744 Len=0 TSval=1535222651 TSecr=1347042247
14.440360143	192.168.0.80	172.217.173.99	TLSv1.3	139 Change Cipher Spec, Application Data
14.44141708	192.168.0.80	172.217.173.99	TLSv1.3	358 Application Data
14.444397617	192.168.0.80	172.217.173.99	TLSv1.3	1484 Application Data
14.444369096	192.168.0.80	172.217.173.99	TCP	1484 37786 - 443 [ACK] Seq=2002 Ack=4683 Win=64128 Len=1418 TSval=1535222657 TSecr=1347042247 [TCP segment of a reassembled PDU]
14.444367926	192.168.0.80	172.217.173.99	TCP	1484 37786 - 443 [ACK] Seq=518 Ack=4683 Win=64128 Len=1418 TSval=1535222657 TSecr=1347042247 [TCP segment of a reassembled PDU]
14.444369175	192.168.0.80	172.217.173.99	TCP	1484 37786 - 443 [ACK] Seq=4928 Ack=4683 Win=64128 Len=1418 TSval=1535222657 TSecr=1347042247 [TCP segment of a reassembled PDU]
14.444370394	192.168.0.80	172.217.173.99	TCP	1484 37786 - 443 [PSH, ACK] Seq=3346 Ack=4683 Win=64128 Len=1418 TSval=1535222657 TSecr=1347042247 [TCP segment of a reassembled PDU]
14.451319163	192.168.0.80	172.217.173.99	TCP	1484 37786 - 443 [ACK] Seq=7764 Ack=4683 Win=64128 Len=1418 TSval=1535222665 TSecr=1347042247 [TCP segment of a reassembled PDU]
14.45146277	192.168.0.80	172.217.173.99	TLSv1.3	234 Application Data
14.459388091	172.217.173.99	192.168.0.80	TCP	66 443 - 37786 [ACK] Seq=4683 Ack=582 Win=66816 Len=0 TSval=1347042281 TSecr=1535222656

Figura 2: Protocolo HTTP transmitindo dados.

3.3 Encerrando conexão HTTP

O final de uma conexão do protocolo HTTP necessita de uma série procedimentos e acordos, conforme descrito previamente. Na figura 3 podemos ver isso acontecendo diversas vezes em conexões diferentes. Perceba o envio de um FIN/ACK que depois é respondido com FIN/ACK para finalizar completamente a conexão.

21.153682395	192.168.0.80	172.217.173.99	TCP	66 [TCP Previous segment not captured] 27714 - 443 [FIN, ACK] Seq=2 Ack=1 Win=501 Len=0 TSval=1535229306 TSecr=2875788958
21.158792480	192.168.0.80	172.217.173.99	TCP	66 37786 - 443 [FIN, ACK] Seq=18912 Ack=9688 Win=64128 Len=0 TSval=1535229379 TSecr=1347842782
21.176166354	192.168.0.80	172.217.173.99	TCP	66 37714 - 443 [ACK] Seq=3 Ack=2 Win=501 Len=0 TSval=1535229386 TSecr=2075809864
21.189532769	192.168.0.80	172.217.173.99	TCP	66 37786 - 443 [ACK] Seq=18913 Ack=9689 Win=64128 Len=0 TSval=1535229393 TSecr=1347848998

Figura 3: Finalizando conexão HTTP.

4 Funcionamento do protocolo UDP e experimento

4.1 Detalhamento protocolo UDP

A razão básica é que o UDP é um protocolo sem conexão ao contrário do TCP. O protocolo de datagramas do usuário (UDP) opera sobre o protocolo da Internet (IP) para transmitir datagramas em uma rede. O UDP não exige que a origem e o destino estabeleçam um handshake triplo antes que a transmissão ocorra. Além disso, não há necessidade de uma conexão de ponta a ponta.

O UDP é considerado um protocolo sem conexão porque não exige que um circuito virtual seja estabelecido antes que qualquer transferência de dados ocorra. O protocolo de comunicação apenas envia os pacotes, o que significa que ele tem muito menos sobrecarga de largura de banda e latência.

Com isso, os pacotes podem seguir caminhos diferentes entre o emissor e o receptor e, como resultado, alguns pacotes podem ser perdidos ou recebidos fora de ordem. Desse modo, existem muitos protocolos que usam UDP, entre eles: DNS, DHCP, BOOTP, TFTP, RIP, Protocolo em tempo real que não tolera atrasos, *multicast* e etc.

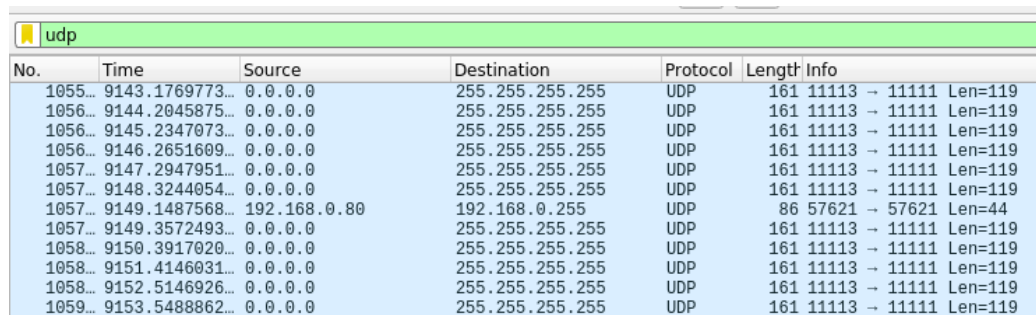
4.2 Desenvolvimento do experimento com protocolo UDP

De acordo com Bamdeb Ghosh [2], O UDP fornece dois serviços não fornecidos pela camada IP, um deles é o número da porta e outro é um recurso de soma de verificação para identificar se os dados estão intactos.

O cabeçalho UDP contém 8 bytes e possui:

- Porta de origem: O número da porta de origem do pacote. Exemplo: 4444.
- Porta de destino: O número da porta de destino do pacote. Exemplo: 51164.
- Comprimento: O comprimento dos dados UDP mais cabeçalho UDP.
- Checksum: Detecta o erro.

Para realizar a análise através do Wireshark, foi utilizado o serviço de streaming de vídeo Twitch, conforme a figura 4:



No.	Time	Source	Destination	Protocol	Length	Info
1055...	9143.1769773...	0.0.0.0	255.255.255.255	UDP	161	11113 → 11111 Len=119
1056...	9144.2045875...	0.0.0.0	255.255.255.255	UDP	161	11113 → 11111 Len=119
1056...	9145.2347073...	0.0.0.0	255.255.255.255	UDP	161	11113 → 11111 Len=119
1056...	9146.2651609...	0.0.0.0	255.255.255.255	UDP	161	11113 → 11111 Len=119
1057...	9147.2947951...	0.0.0.0	255.255.255.255	UDP	161	11113 → 11111 Len=119
1057...	9148.3244054...	0.0.0.0	255.255.255.255	UDP	161	11113 → 11111 Len=119
1057...	9149.1487568...	192.168.0.80	192.168.0.255	UDP	86	57621 → 57621 Len=44
1057...	9149.3572493...	0.0.0.0	255.255.255.255	UDP	161	11113 → 11111 Len=119
1058...	9150.3917020...	0.0.0.0	255.255.255.255	UDP	161	11113 → 11111 Len=119
1058...	9151.4146031...	0.0.0.0	255.255.255.255	UDP	161	11113 → 11111 Len=119
1058...	9152.5146926...	0.0.0.0	255.255.255.255	UDP	161	11113 → 11111 Len=119
1059...	9153.5488862...	0.0.0.0	255.255.255.255	UDP	161	11113 → 11111 Len=119

Figura 4: Protocolo UDP trafegando dados.

Esmiuçando os segmentos, na figura 5 foi possível identificar os cabeçalhos e os dados do pacote:

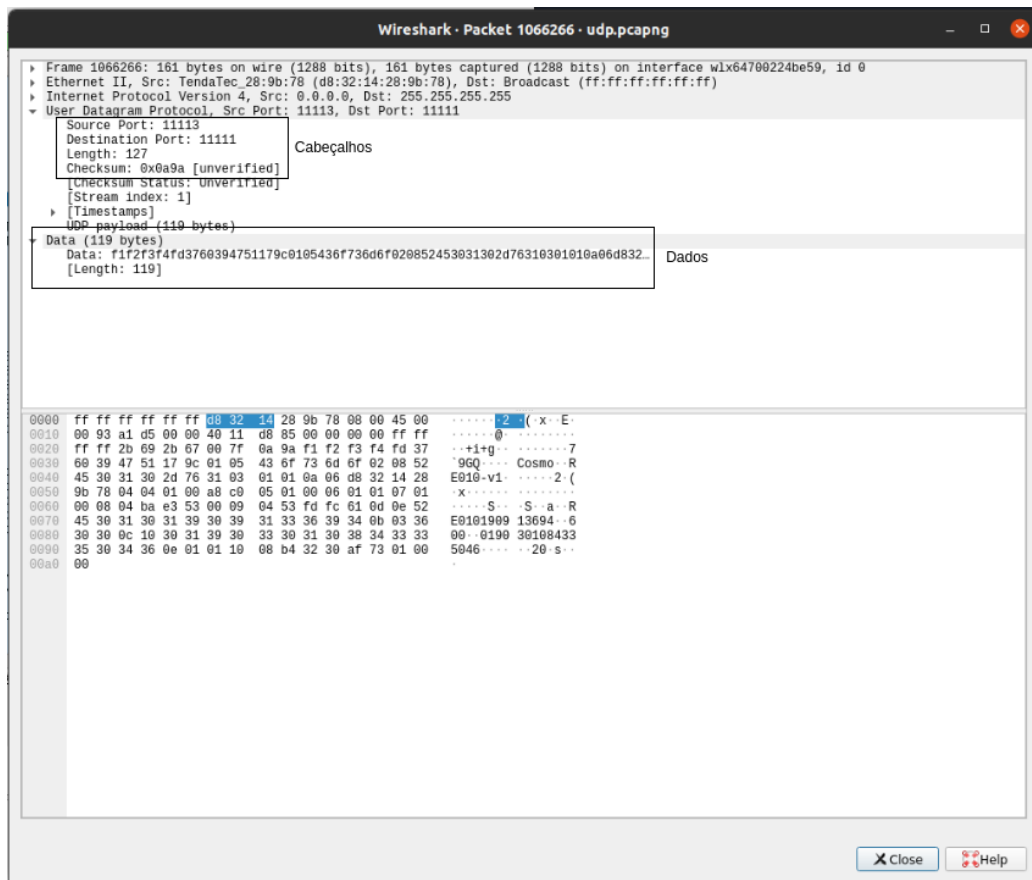


Figura 5: Protocolo UDP análise de cabeçalhos e dados.

5 Conclusão

Realizando esse trabalho foi possível perceber na prática o funcionamento de protocolos que fundamentam a rede mundial de computadores e a vida de todos nós. O uso da ferramenta Wireshark, bem como do protocolo HTTP, junto com o TCP permitiu a experiência prática e uma maior conhecimento da gerência de redes e desses protocolos utilizados. Em suma, o trabalho cumpriu seus requisitos pedagógicos e didáticos para a formação de um profissional da ciência da computação.

Referências

- [1] Gitbook. Three-way handshake. <https://gitbook.ganeshicmc.com/redes/three-way-handshake>, 2022. [Online; acessado em 05 de fevereiro].
- [2] linuxhint. Udp wireshark analysis. https://linuxhint.com/udp_wireshark_analysis, 2022. [Online; acessado em 05 de fevereiro].
- [3] Mozilla. Uma visão geral do http. <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview>, 2022. [Online; acessado em 04 de fevereiro].
- [4] Tecmundo. O que é tcp/ip. <https://www.tecmundo.com.br/o-que-e/780-o-que-e-tcp-ip-.htm>, 2022. [Online; acessado em 04 de fevereiro].
- [5] Wireshark. Wireshark docs. <https://www.wireshark.org/docs>, 2022. [Online; acessado em 04 de fevereiro].