Universidade do Minho

Mestrado Integrado em Engenharia Informática

Comunicações por Computador

Trabalho Prático Nº.1 – Protocolos da Camada de Transporte

PL5 - Grupo 8

Ana Rita Rosendo
A84475

Gonçalo Esteves
A85731

Rui Oliveira
A83610

3 de Março de 2020

Conteúdo

1 Questões e Respostas

1.1 Questão 1

Comando usado (aplicação)	Protocolo de Aplicação(se aplicável)	Protocolo de transporte(se aplicável)	Porta de atendimento(se aplicável)	Overhead de transporte em bytes (se aplicável)
Ping	-	-	-	-
traceroute	DNS	UDP	53	8/(75-20)*100 = 14.5
telnet	telnet	TCP	23	20/(52-20)*100 = 62.5
ftp	ftp	TCP	21	20/(74-20)*100 = 37.0
Tftp	tftp	UDP	69	8/(72-20)*100 = 15.4
browser/http	http	TCP	80	20/(40-20)*100 = 100.0
nslookup	DNS	UDP	53	8/(70-20)*100 = 16.0
ssh	ssh	TCP	22	20/(81-20)*100 = 32.9
Outras:	-	-	-	-

O comando *ping*, ao invês dos restantes comandos, utiliza um protocolo ICMP. Como tal, não analisamos as informações relativas a este.

Os comandos cujo protocolo de transporte é o UDP, possuem um cabeçalho de tamanho fixo de 8 *bytes*, ao contrário do protocolo TCP, que apesar de possuir um *overhead* de 20 *bytes* fixos, este valor acaba por poder variar, pois podem ser alocados mais *bytes* para opções, ou seja, este valor pode variar conforme seja usado o campo *options* ou não.

1.2 Questão 2

Uma representação num diagrama temporal das transferências da file1 por FTP e TFTP respetivamente. Se for caso disso, identifique as fases de estabelecimento de conexão, transferência de dados e fim de conexão. Identifica também claramente os tipos de segmentos trocados e os números de sequência usados quer nos dados como nas confirmações.

Por forma a responder a esta pergunta, tivemos de recorrer ao *Wireshark*, utilizado em simultâneo enquanto eram executadas as transferências a partir dos comandos da bash.

Inicialmente fizemos a transferência via FTP, obtendo os seguintes pacotes de dados:

60	81.073276	10.1.1.1	10.3.3.1	FTP	78 Request: RETR file1
61	81.073457	10.3.3.1	10.1.1.1	TCP	74 ftp-data > 53249 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=1285834 TSecr=0 WS=16
62	81.073610	10.1.1.1	10.3.3.1	TCP	74 53249 > ftp-data [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=1285834 TSecr=1285834 WS=16
63	81.073744	10.3.3.1	10.1.1.1	TCP	66 ftp-data > 53249 [ACK] Seq=1 Ack=1 Win=14608 Len=0 TSval=1285834 TSecr=1285834
64	81.073790	10.3.3.1	10.1.1.1	FTP	130 Response: 150 Opening BINARY mode data connection for file1 (193 bytes).
65	81.073793	10.3.3.1	10.1.1.1	FTP-DAT	259 FTP Data: 193 bytes
66	81.073833	10.3.3.1	10.1.1.1	TCP	66 ftp-data > 53249 [FIN, ACK] Seq=194 Ack=1 Win=14608 Len=0 TSval=1285834 TSecr=1285834
67	81.074182	10.1.1.1	10.3.3.1	TCP	66 53249 > ftp-data [ACK] Seq=1 Ack=194 Win=15552 Len=0 TSval=1285834 TSecr=1285834
68	81.074182	10.1.1.1	10.3.3.1	TCP	66 53249 > ftp-data [FIN, ACK] Seq=1 Ack=195 Win=15552 Len=0 TSval=1285834 TSecr=1285834
69	81.074301	10.3.3.1	10.1.1.1	TCP	66 ftp-data > 53249 [ACK] Seq=195 Ack=2 Win=14608 Len=0 TSval=1285834 TSecr=1285834

Figura 1: Captura da transferência via FTP

Através da análise dos vários pacotes de dados, concluímos que o diagrama temporal da transferência seria o da figura que se segue. No entanto, é de realçar que devemos ter atenção ao facto de que a captura dos pacotes capturou também pacotes não relevantes para a transferência (que pertenciam à parte do controlo). Logo após a transferência o cliente enviou um [ACK] para o servidor, mas o cliente recebeu um [FIN, ACK] do servidor antes que este tenha recebido o [ACK] dos dados.

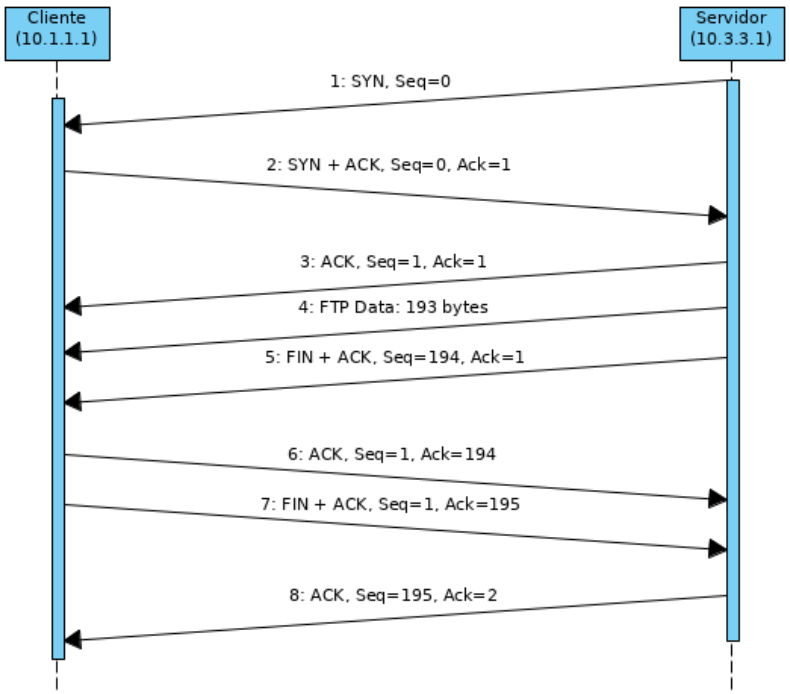

Figura 2: Linha temporal da transferência via FTP

Relativamente ao TFTP, o processo foi semelhante, sendo também capturado através do *Wireshark* os pacotes de dados:

31	145.902896	10.1.1.1	10.3.3.1	TFTP	56	Read Request, File: file1, Transfer type: octet
32	145.907178	00:00:00_aa:00:14	Broadcast	ARP	42	Who has 10.3.3.254? Tell 10.3.3.1
33	145.907178	00:00:00_aa:00:10	00:00:00_aa:00:14	ARP	42	10.3.3.254 is at 00:00:00:aa:00:10
34	145.907294	10.3.3.1	10.1.1.1	TFTP	239	Data Packet, Block: 1 (last)
35	145.908410	10.1.1.1	10.3.3.1	TFTP	46	Acknowledgement, Block: 1

Figura 3: Captura da transferência via TFTP

Mais uma vez, analisando o tráfego de pacotes, facilmente concluímos que o processo de transferência foi mais simples que o anterior:

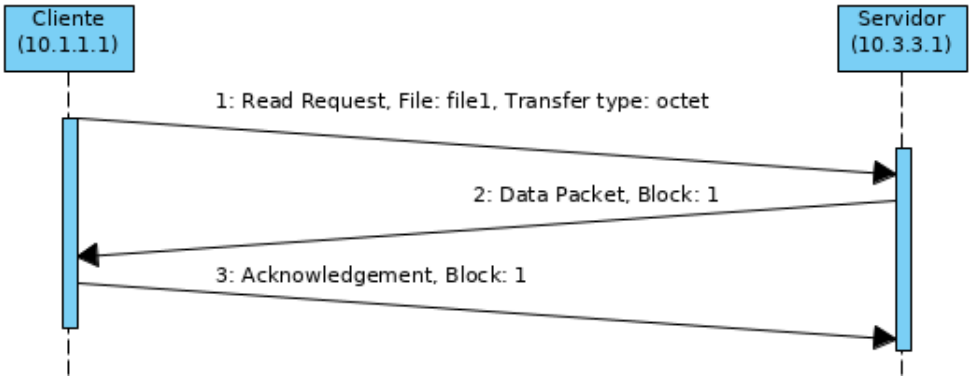

Figura 4: Linha temporal da transferência via TFTP

1.3 Questão 3

Com base nas experiências realizadas, distinga e compare sucintamente as quatro aplicações de transferência de ficheiros que usou nos seguintes pontos (i) uso da camada de transporte; (ii) eficiência na transferência; (iii) complexidade; (iv) segurança;

(i) **Uso da camada de transporte**

- **SFTP:** Utiliza o protocolo TCP.
- **FTP:** Utiliza o protocolo TCP.
- **TFTP:** Utiliza o protocolo UDP.
- **HTTP:** Utiliza o protocolo TCP.

(ii) **Eficiência na tranferência**

- **SFTP:** Garante a transferência de dados porém é mais lento, exceto do FTP. É semelhante ao FTP.
- **FTP:** O FTP faz uso do protocolo TCP que garante uma troca de dados fiável e ordenada uma vez que este protocolo efetua controlo de erros, fluxo e congestão. Todavia há uma perda de eficiência uma vez que todos os segmentos TCP têm de ser confirmados (*acknowledges*) e enquanto essa confirmação não for efetuada, não se pode continuar. Portanto, o FTP garante a transferência de dados, no entanto, é mais lento.
- **TFTP:** O TFTP faz uso do protocolo UDP que garante uma troca de dados não fiável e desordenada. Como o protocolo UDP não usa *acknowledge* não é possível confirmar se o pacote foi entregue com sucesso. Tal torna o TFTP mais rápido.
- **HTTP:** É uma escolha ideial visto que é fiável e permite que vários HTTP requests sejam enviados numa única ligação TCP tornando-o mais rápido.

(iii) **Complexidade**

- **SFTP:** É semelhante ao FTP, no entanto, é mais seguro e permite acesso, transferência e gestão dos ficheiros tornando-se num protocolo bastante complexo.
- **FTP:** O FTP suporta vários pedidos para transferir em paralelo revelando-se um protocolo bastante complexo.
- **TFTP:** É uma alternativa simplificada do protocolo FTP e como faz uso do protocolo UDP torna-se menos complexo.
- **HTTP:** O HTTP é um protocolo utilizado para sistemas de informação hipermedia e garante confiança e escalabilidade. Este possuí uma complexidade moderada.

(iv) **Segurança:**

- **SFTP:** Ao contrário do FTP, este protocolo garante elevada segurança uma vez que usa encriptação de dados através do SSH. Também possui autenticação e proteção da integridade dos dados.
- **FTP:** Este protocolo possui diversas falhas de segurança uma vez que não usa encriptação tornando a transmissão suscetível a interseções por parte de terceiros, ou seja, qualquer pessoa que realize captura de pacotes na rede pode ter acesso aos nomes de utilizadores, passwords, etc.

- **TFTP:** Não garante qualquer tipo de segurança, este não fornece autenticação.
- **HTTP:** Uma vez que toda a informação é representada em texto e este protocolo não usa encriptação, os dados, tanto do utilizador como do servidor, podem ser alterados por invasores. Sendo assim, a segurança deste protocolo é muito baixa.

1.4 Questão 4

As características das ligações de rede têm uma enorme influência nos níveis de Transporte e de Aplicação. Discuta, relacionando a resposta com as experiências realizadas, as influências das situações de perda ou duplicação de pacotes IP no desempenho global de Aplicações fiáveis (se possível, relacionando com alguns dos mecanismos de transporte envolvidos).

A perda e duplicação de pacotes IP nos níveis de Transporte e de Aplicação são bastante comuns e é cada vez mais complicado combater as mesmas. A LAN3, que se encontra na topologia de rede fornecida no enunciado, é um exemplo claro disso. Sendo assim, são os protocolos TCP e UDP os responsáveis por tentar resolver estas adversidades. Por um lado, o TCP faz a deteção e correção de erros, pelo que quando são encontrados pacotes com erros, ocorre uma tentativa de retransmissão do mesmo e, com isto, acontece também uma redução no débito de transmissão e, consequentemente, as filas dos routers vão sendo esvaziadas, sendo que depois o débito de transmissão vai sofrendo um aumento gradual. Por outro lado, o UDP apesar de detetar erros, não os corrige, estes são apenas descartados e, por isso, torna-se impossível detetar perdas de pacotes, surgindo, de modo a resolver este problema, uma dependência de um protocolo que se encontra acima do UDP, cuja função é identificar pacotes pelo seu ID ou nº de sequência, conseguindo saber assim se algum pacote se perdeu.

As figuras 1 e 2 mostram o download do ficheiro "file1", usando FTP e TFTP. Em ambas as situações, a maior parte dos envios foram bem sucedidos, porém, após algumas tentativas, concluimos que são, por vezes, encontrados erros. Quando usado o FTP existiam segmentos ACK que acabavam repetidos mas, mesmo assim, o ficheiro acabava por conseguir chegar ao seu destino final, enquanto que ao usar o TFTP o ficheiro não chegava ao seu destino final e o host acabava por perder muito tempo até se aperceber que tal tinha acontecido.

Nos casos em que as transferências eram finalizadas com sucesso, a transferência usando TFTP em relação a FTP era mais rápida. Concluimos então que caso o ficheiro a transmitir seja pequeno ou a velocidade a prioridade, o TFTP deverá ser a aplicação a seguir, no entanto, caso a prioridade seja o ficheiro chegar ao destino(grandes quantidades de dados) devemos escolher o FTP que assegura a transferência de todos os dados.

2 Conclusões

Neste trabalho foram abordados os diversos Protocolos da Camada de Transporte. Para além destes, foram também estudados múltiplos Protocolos da Camada de Aplicação.

Com o auxílio das ferramentas *Core* e *Wireshark*, foram realizados vários testes possibilitando a visualização dos processos de tranferência de dados. Assim, foi-nos permitido diferenciar os protocolos TCP e UDP.

Podemos concluir que o TCP permite uma transferência de dados fiável e garante que os dados enviados são recebidos, porém não podemos exigir a maior rapidez e eficiência. Por outro lado, concluimos que o UDP permite uma transferência de dados não fiável mas a uma velocidade maior. Ou seja, se a perda de pacotes não for um problema este deve ser o protocolo a usar.