Comunicações Por Computador
Trabalho Prático 1º - Protocolos da Camada de Transporte

Enzo Vieira (a98352), Gustavo Barros (a100656), Pedro Ferreira (a97646)
Universidade do Minho 2024/2025

Parte I - Instalação, Configuração e Validação da Rede de Testes - 1.1 Definição da Topologia

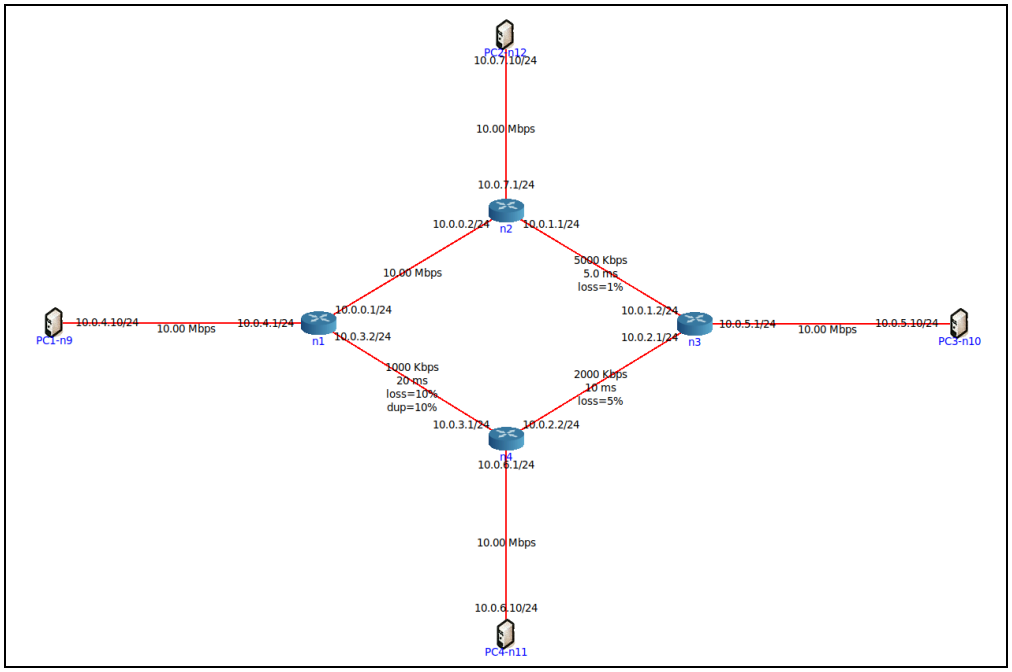
Questão: Defina em modo de edição uma topologia com quatro roteadores. Faça uma ligação do nó n1 para o nó n2, deste para o nó n3, e deste para o nó n4, resultando numa topologia em anel. Em cada um desses roteadores, ligue um host. Renomeie os hosts como PCx, onde x é o mesmo dígito que identifica o roteador a que está ligado. Por exemplo, PC1 é o host ligado ao roteador n1.

Verifique que são atribuídos automaticamente endereços de rede IPv4 e IPv6 aos vários nós. Apague os endereços IPv6 e deixe apenas os IPv4.

Inspecione as ligações que interligam os nós. Configure o débito das ligações entre os roteadores e hosts a 10 Mbps. Configure as demais ligações, entre os roteadores, da seguinte maneira:

- Entre os nós 1 e 2: Utilize um débito de 10 Mbps, atraso de 0 ms e perdas de 0%.
- Entre os nós 2 e 3: Utilize um débito de 5 Mbps, atraso de 5 ms e perdas de 1%.
- Entre os nós 3 e 4: Utilize um débito de 2 Mbps, atraso de 10 ms e perdas de 5%.
- Entre os nós 4 e 1: Utilize um débito de 1 Mbps, atraso de 20 ms, perdas de 10% e 10% de duplicações.

Resposta:

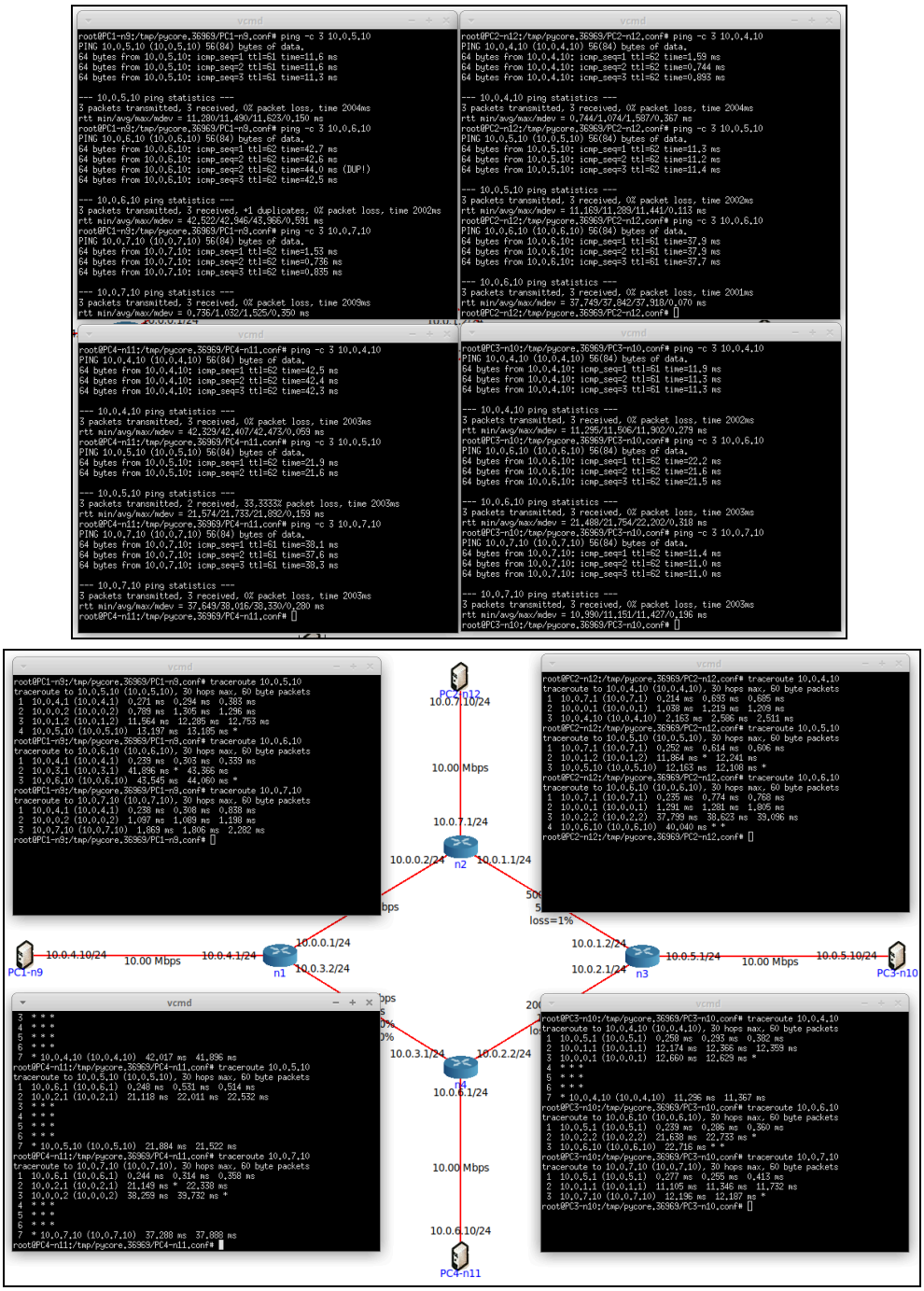


Topologia

Parte I - 1.2 Demonstração da Interligação dos Hosts

Questão: Verifique que todas as rotas foram configuradas com sucesso e demonstre que os hosts possuem ligação entre si. Utilize-se das ferramentas traceroute, ping e iperf para verificar as rotas entre hosts, as estimativas de perdas de pacotes, atrasos e débito fim-a-fim.

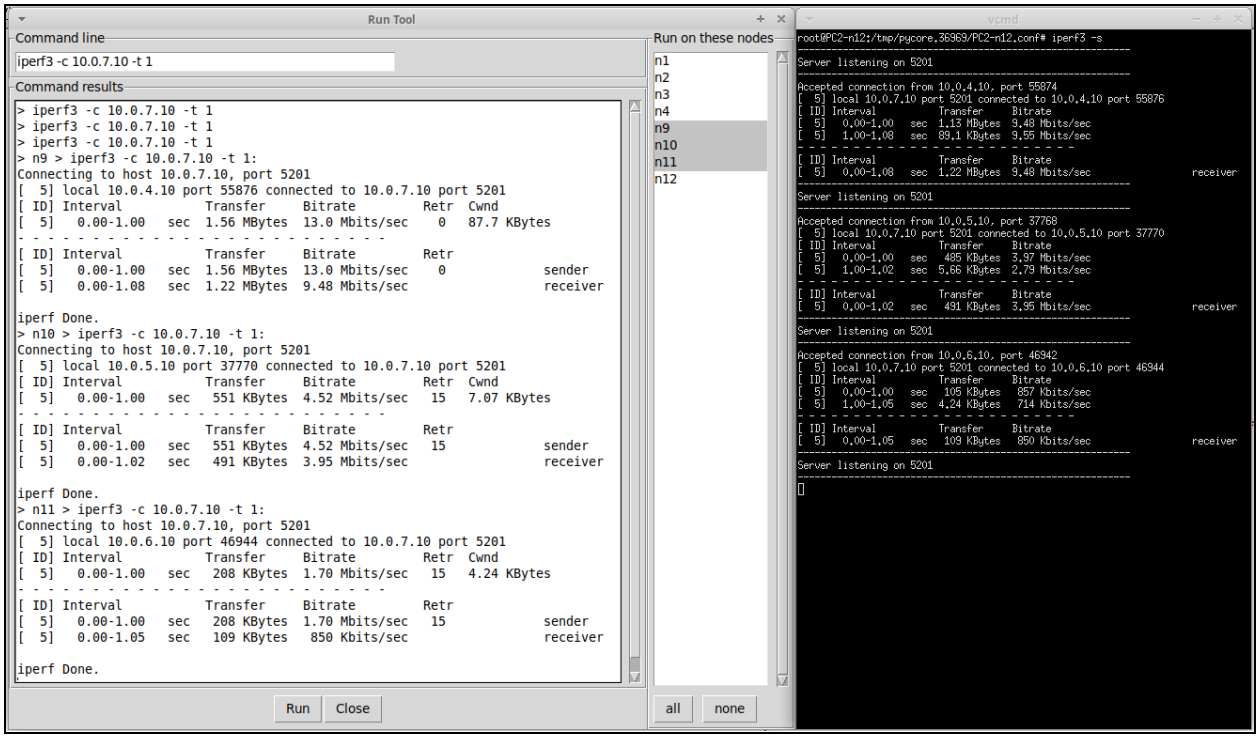
Resposta: Ao enviar, a partir de cada host, um *ping* a cada um dos restantes, é possível confirmar a interligação de todos eles. O processo é análogo nos *traceroute*, acrescenta-se apenas o feedback da exacta rota tomada pelos pacotes. Nas imagens, sentido horário desde janela superior esquerda: PC1, PC2, PC3,PC4.



Prova de interligação via *ping* (cima) e *traceroute* (baixo)

Parte I - 1.2 - Demonstração da Interligação dos Hosts (cont.)

O *iperf* permite o estabelecimento dum host como servidor ou cliente. Segundo este método, prova-se uma ligação entre dois hosts - um servidor, outro cliente - quando o servidor acusa "Accepted Connection".
Na imagem, demonstra-se o PC2 como servidor conectado com sucesso aos PC1, PC3 e PC4. Repetiu-se o processo nos outros hosts alternando o cargo de servidor, como prova de redundância, obtendo sempre igual resultado favorável.



Prova de ligações a PC2 via *iperf*

Parte I - 1.3 Configuração de Rotas por Open Shortest Path First (OSPF)

Questão: As configurações das rotas foram realizadas dinamicamente pelo protocolo OSPF.

- **a)** Para obter melhores resultados de débito, atraso e perdas de pacotes, quais rotas alteraria? Justifique.
- **b)** Caso se desejasse manter o uso do OSPF, seria possível melhorar as rotas definidas dinamicamente? Como?

Resposta:

- **a)** Seriam descartadas todas as rotas com a ligação n1-n4 (portas 10.0.3.2 - 10.0.3.1), já que, para qualquer situação, é mais lenta, mais estreita e mais sujeita a perdas que a alternativa. Vê-se então só útil em último recurso.
  Considere-se o caso mais evidente, uma transferência partida do PC1 destinada ao PC4:

  - **Rota n1-n4 (OSPF):** Limitada a 1000Kb/s com 20ms de atraso e 10% chance de perda de pacote.
  - **Rota n1-n2-n3-n4:** Limitada a 2000Kb/s, com 5+10=15ms de delay e 0.01*0.05≈0.05% chance de perda de pacote.

- **b)** O protocolo OSPF prioriza a ligação que lhe custa menos. Maior é o custo quanto mais estreita a banda da ligação. Há um valor-referência que define o menor custo possível (=1) atribuído a qualquer ligação de igual ou maior largura.

  No caso da transferência PC1-PC4 anterior, o valor-referência predefinido pode estar a impedir o reconhecimento da porta 10.0.0.1 como preferível - dado auferir 10Mb/s contra 1Mb/s - antes tomando as duas ligações como equivalentes. Assim, a referência deve ser aumentada.

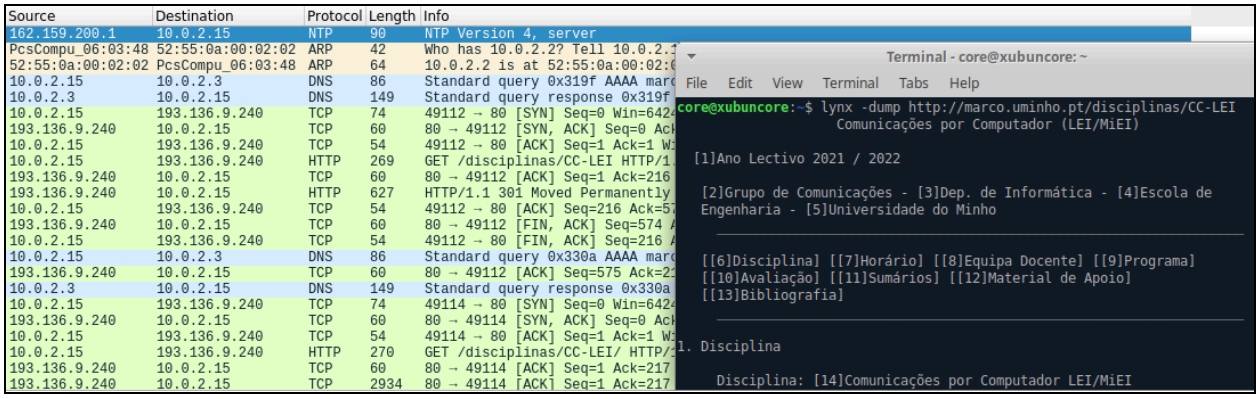
Parte II - Uso da Camada de Transporte por parte das Aplicações

Questão: Capture o tráfego em determinados instantes que considere adequados, observe atentamente como as várias aplicações utilizam os serviços da camada inferior. Com base no trabalho realizado, identifique para cada aplicação executada, qual o protocolo de aplicação, o protocolo de transporte, porta de atendimento e overhead de transporte.

Resposta:

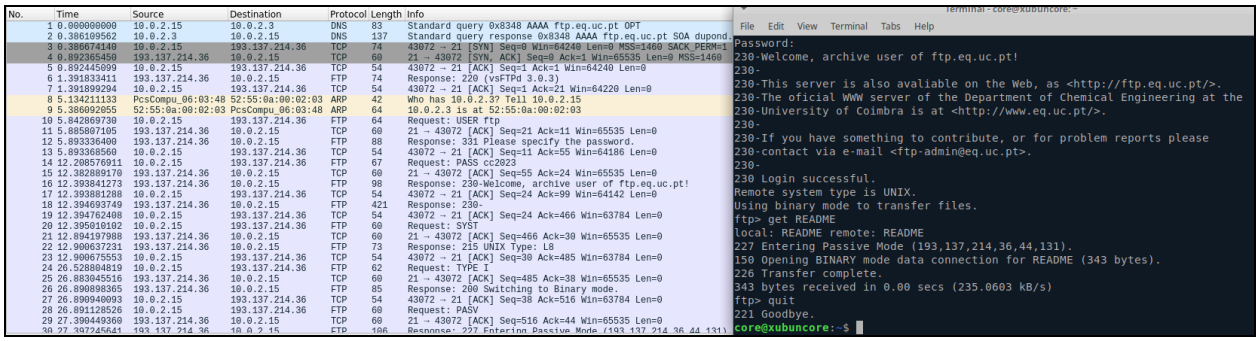
Comando Utilizado	Protocolo de Aplicação	Protocolo de Transporte	Porta de Atendimento	Overhead de Transporte (Bytes)
`lynx -dump http://marco.uminho.pt/disciplinas/CC-LEI`	HTTP	TCP	80	40 (SYN) 24 (SYNACK) 20 (restantes)
`ftp -p ftp.eq.uc.pt`	FTP	TCP	21	
`telnet 193.136.9.33`	Telnet	TCP	23	
`ssh cc@cc2024.ddns.net`	SSH-2	TCP	22	
`curl -v tftp://cc2024.ddns.net/file1`	TFTP	UDP	69	8
`nslookup www.uminho.pt`	DNS	UDP	53	8
`ping -c 5 www.google.pt`	DNS	UDP	53	8
`traceroute www.fccn.pt`	DNS	UDP	33434-33534	8

Considere-se que o *traceroute* foi mal sucedido para ambos os destinos sugeridos, acusando time-out em todos os pacotes enviados apesar de ter sido testado sob múltiplas redes. No entanto, bastou a tentativa de conexão para conhecer as portas para as quais os pacotes se dirigiam.



Captura HTTP

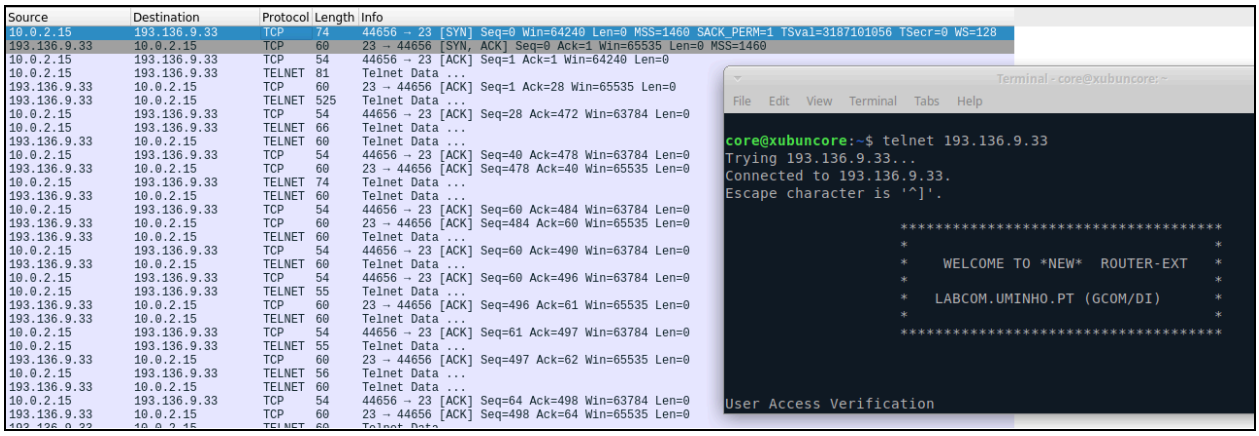
Parte II - Uso da Camada de Transporte por parte das Aplicações (cont.)



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	10.0.2.3	DNS	83	Standard query 0x8348 AAAA ftp.eq.uc.pt OPT
2	0.386109562	10.0.2.3	10.0.2.15	DNS	137	Standard query response 0x8348 AAAA ftp.eq.uc.pt SOA dupond.
3	0.386674148	10.0.2.15	193.137.214.36	TCP	74	43072 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
4	0.892365450	193.137.214.36	10.0.2.15	TCP	60	21 → 43072 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
5	0.892445099	10.0.2.15	193.137.214.36	TCP	54	43072 → 21 [ACK] Seq=1 Ack=1 Win=64240 Len=0
6	1.391833411	193.137.214.36	10.0.2.15	FTP	74	Response: 220 (vsFTPd 3.0.3)
7	1.391899294	10.0.2.15	193.137.214.36	TCP	54	43072 → 21 [ACK] Seq=1 Ack=21 Win=64220 Len=0
8	5.134211133	PcsCompu_06:03:48	52:55:0a:00:02:03	ARP	42	Who has 10.0.2.3? Tell 10.0.2.15
9	5.386092055	52:55:0a:00:02:03	PcsCompu_06:03:48	ARP	64	10.0.2.3 is at 52:55:0a:00:02:03
10	5.842869730	10.0.2.15	193.137.214.36	FTP	64	Request: USER ftp
11	5.885807105	193.137.214.36	10.0.2.15	TCP	60	21 → 43072 [ACK] Seq=21 Ack=11 Win=65535 Len=0
12	5.893336400	193.137.214.36	10.0.2.15	FTP	88	Response: 331 Please specify the password.
13	5.893368560	10.0.2.15	193.137.214.36	TCP	54	43072 → 21 [ACK] Seq=11 Ack=55 Win=64186 Len=0
14	12.200576911	10.0.2.15	193.137.214.36	FTP	67	Request: PASS cc2023
15	12.382889170	193.137.214.36	10.0.2.15	TCP	60	21 → 43072 [ACK] Seq=55 Ack=24 Win=65535 Len=0
16	12.393841273	193.137.214.36	10.0.2.15	FTP	98	Response: 230-Welcome, archive user of ftp.eq.uc.pt!
17	12.393881288	10.0.2.15	193.137.214.36	TCP	54	43072 → 21 [ACK] Seq=24 Ack=99 Win=64142 Len=0
18	12.394693749	193.137.214.36	10.0.2.15	FTP	421	Response: 230-
19	12.394762408	10.0.2.15	193.137.214.36	TCP	54	43072 → 21 [ACK] Seq=24 Ack=466 Win=63784 Len=0
20	12.395010102	10.0.2.15	193.137.214.36	FTP	60	Request: SYST
21	12.894197988	193.137.214.36	10.0.2.15	TCP	60	21 → 43072 [ACK] Seq=466 Ack=30 Win=65535 Len=0
22	12.900637231	193.137.214.36	10.0.2.15	FTP	73	Response: 215 UNIX Type: L8
23	12.900675553	10.0.2.15	193.137.214.36	TCP	54	43072 → 21 [ACK] Seq=30 Ack=485 Win=63784 Len=0
24	26.528804819	10.0.2.15	193.137.214.36	FTP	62	Request: TYPE I
25	26.883045516	193.137.214.36	10.0.2.15	TCP	60	21 → 43072 [ACK] Seq=485 Ack=38 Win=65535 Len=0
26	26.890898365	193.137.214.36	10.0.2.15	FTP	85	Response: 200 Switching to Binary mode.
27	26.890940093	10.0.2.15	193.137.214.36	TCP	54	43072 → 21 [ACK] Seq=38 Ack=516 Win=63784 Len=0
28	26.891128526	10.0.2.15	193.137.214.36	FTP	60	Request: PASV
29	27.390449360	193.137.214.36	10.0.2.15	TCP	60	21 → 43072 [ACK] Seq=516 Ack=44 Win=65535 Len=0
30	27.397245641	193.137.214.36	10.0.2.15	FTP	106	Response: 227 Entering Passive Mode (193,137,214,36,44,131)

```
Terminal - core@xubuncore: ~
File  Edit  View  Terminal  Tabs  Help
Password:
230-Welcome, archive user of ftp.eq.uc.pt!
230-
230-This server is also avaliable on the Web, as <http://ftp.eq.uc.pt/>.
230-The oficial WWW server of the Department of Chemical Engineering at the
230-University of Coimbra is at <http://www.eq.uc.pt/>.
230-
230-If you have something to contribute, or for problem reports please
230-contact via e-mail <ftp-admin@eq.uc.pt>.
230-
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> get README
local: README remote: README
227 Entering Passive Mode (193,137,214,36,44,131).
150 Opening BINARY mode data connection for README (343 bytes).
226 Transfer complete.
343 bytes received in 0.00 secs (235.0603 kB/s)
ftp> quit
221 Goodbye.
core@xubuncore:~$
```

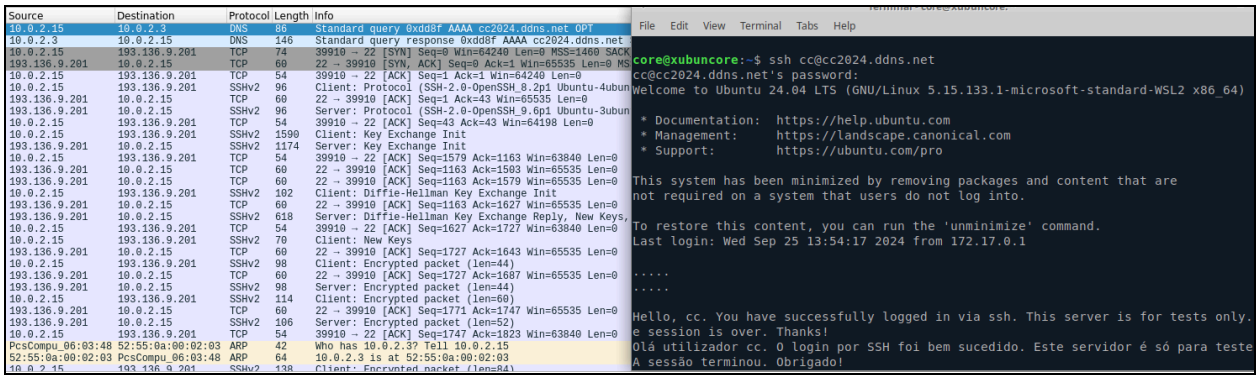
Captura FTP



Source	Destination	Protocol	Length	Info
10.0.2.15	193.136.9.33	TCP	74	44656 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3187101056 TSecr=0 WS=128
193.136.9.33	10.0.2.15	TCP	60	23 → 44656 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
10.0.2.15	193.136.9.33	TCP	54	44656 → 23 [ACK] Seq=1 Ack=1 Win=64240 Len=0
10.0.2.15	193.136.9.33	TELNET	81	Telnet Data ...
193.136.9.33	10.0.2.15	TCP	60	23 → 44656 [ACK] Seq=1 Ack=28 Win=65535 Len=0
193.136.9.33	10.0.2.15	TELNET	525	Telnet Data ...
10.0.2.15	193.136.9.33	TCP	54	44656 → 23 [ACK] Seq=28 Ack=472 Win=63784 Len=0
10.0.2.15	193.136.9.33	TELNET	66	Telnet Data ...
193.136.9.33	10.0.2.15	TELNET	60	Telnet Data ...
10.0.2.15	193.136.9.33	TCP	54	44656 → 23 [ACK] Seq=40 Ack=478 Win=63784 Len=0
193.136.9.33	10.0.2.15	TCP	60	23 → 44656 [ACK] Seq=478 Ack=40 Win=65535 Len=0
10.0.2.15	193.136.9.33	TELNET	74	Telnet Data ...
193.136.9.33	10.0.2.15	TELNET	60	Telnet Data ...
10.0.2.15	193.136.9.33	TCP	54	44656 → 23 [ACK] Seq=60 Ack=484 Win=63784 Len=0
193.136.9.33	10.0.2.15	TCP	60	23 → 44656 [ACK] Seq=484 Ack=60 Win=65535 Len=0
193.136.9.33	10.0.2.15	TELNET	60	Telnet Data ...
10.0.2.15	193.136.9.33	TCP	54	44656 → 23 [ACK] Seq=60 Ack=490 Win=63784 Len=0
193.136.9.33	10.0.2.15	TELNET	60	Telnet Data ...
10.0.2.15	193.136.9.33	TCP	54	44656 → 23 [ACK] Seq=60 Ack=496 Win=63784 Len=0
10.0.2.15	193.136.9.33	TELNET	55	Telnet Data ...
193.136.9.33	10.0.2.15	TCP	60	23 → 44656 [ACK] Seq=496 Ack=61 Win=65535 Len=0
193.136.9.33	10.0.2.15	TELNET	60	Telnet Data ...
10.0.2.15	193.136.9.33	TCP	54	44656 → 23 [ACK] Seq=61 Ack=497 Win=63784 Len=0
10.0.2.15	193.136.9.33	TELNET	55	Telnet Data ...
193.136.9.33	10.0.2.15	TCP	60	23 → 44656 [ACK] Seq=497 Ack=62 Win=65535 Len=0
10.0.2.15	193.136.9.33	TELNET	56	Telnet Data ...
193.136.9.33	10.0.2.15	TELNET	60	Telnet Data ...
10.0.2.15	193.136.9.33	TCP	54	44656 → 23 [ACK] Seq=64 Ack=498 Win=63784 Len=0
193.136.9.33	10.0.2.15	TCP	60	23 → 44656 [ACK] Seq=498 Ack=64 Win=65535 Len=0
193.136.9.33	10.0.2.15	TELNET	60	Telnet Data ...

```
Terminal - core@xubuncore: ~
File  Edit  View  Terminal  Tabs  Help
core@xubuncore:~$ telnet 193.136.9.33
Trying 193.136.9.33...
Connected to 193.136.9.33.
Escape character is '^]'.

                    ************************************
                    *                                  *
                    *    WELCOME TO *NEW*  ROUTER-EXT  *
                    *                                  *
                    *   LABCOM.UMINHO.PT (GCOM/DI)     *
                    *                                  *
                    ************************************


User Access Verification
```

Captura Telnet

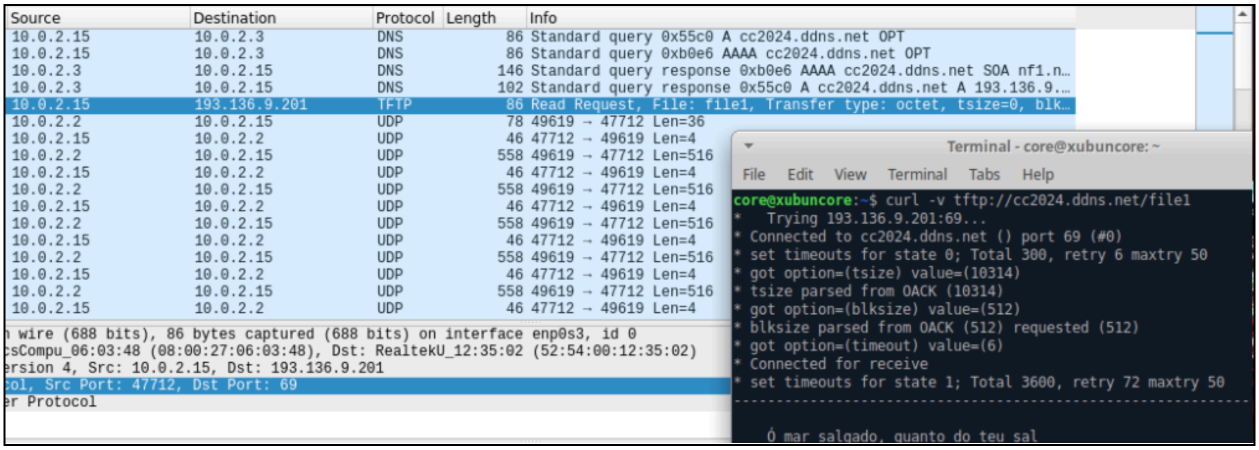


Source	Destination	Protocol	Length	Info
10.0.2.15	10.0.2.3	DNS	80	Standard query 0xdd8f AAAA cc2024.ddns.net OPT
10.0.2.3	10.0.2.15	DNS	146	Standard query response 0xdd8f AAAA cc2024.ddns.net
10.0.2.15	193.136.9.201	TCP	74	39910 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK
193.136.9.201	10.0.2.15	TCP	60	22 → 39910 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MS
10.0.2.15	193.136.9.201	TCP	54	39910 → 22 [ACK] Seq=1 Ack=1 Win=64240 Len=0
10.0.2.15	193.136.9.201	SSHv2	96	Client: Protocol (SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubun
193.136.9.201	10.0.2.15	TCP	60	22 → 39910 [ACK] Seq=1 Ack=43 Win=65535 Len=0
193.136.9.201	10.0.2.15	SSHv2	96	Server: Protocol (SSH-2.0-OpenSSH_9.6p1 Ubuntu-3ubun
10.0.2.15	193.136.9.201	TCP	54	39910 → 22 [ACK] Seq=43 Ack=43 Win=64198 Len=0
10.0.2.15	193.136.9.201	SSHv2	1590	Client: Key Exchange Init
193.136.9.201	10.0.2.15	SSHv2	1174	Server: Key Exchange Init
10.0.2.15	193.136.9.201	TCP	54	39910 → 22 [ACK] Seq=1579 Ack=1163 Win=63840 Len=0
193.136.9.201	10.0.2.15	TCP	60	22 → 39910 [ACK] Seq=1163 Ack=1503 Win=65535 Len=0
193.136.9.201	10.0.2.15	TCP	60	22 → 39910 [ACK] Seq=1163 Ack=1579 Win=65535 Len=0
10.0.2.15	193.136.9.201	SSHv2	102	Client: Diffie-Hellman Key Exchange Init
193.136.9.201	10.0.2.15	TCP	60	22 → 39910 [ACK] Seq=1163 Ack=1627 Win=65535 Len=0
193.136.9.201	10.0.2.15	SSHv2	618	Server: Diffie-Hellman Key Exchange Reply, New Keys,
10.0.2.15	193.136.9.201	TCP	54	39910 → 22 [ACK] Seq=1627 Ack=1727 Win=63840 Len=0
10.0.2.15	193.136.9.201	SSHv2	70	Client: New Keys
193.136.9.201	10.0.2.15	TCP	60	22 → 39910 [ACK] Seq=1727 Ack=1643 Win=65535 Len=0
10.0.2.15	193.136.9.201	SSHv2	98	Client: Encrypted packet (len=44)
193.136.9.201	10.0.2.15	TCP	60	22 → 39910 [ACK] Seq=1727 Ack=1687 Win=65535 Len=0
193.136.9.201	10.0.2.15	SSHv2	98	Server: Encrypted packet (len=44)
10.0.2.15	193.136.9.201	SSHv2	114	Client: Encrypted packet (len=60)
193.136.9.201	10.0.2.15	TCP	60	22 → 39910 [ACK] Seq=1771 Ack=1747 Win=65535 Len=0
193.136.9.201	10.0.2.15	SSHv2	106	Server: Encrypted packet (len=52)
10.0.2.15	193.136.9.201	TCP	54	39910 → 22 [ACK] Seq=1747 Ack=1823 Win=63840 Len=0
PcsCompu_06:03:48	52:55:0a:00:02:03	ARP	42	Who has 10.0.2.3? Tell 10.0.2.15
52:55:0a:00:02:03	PcsCompu_06:03:48	ARP	64	10.0.2.3 is at 52:55:0a:00:02:03
10.0.2.15	193.136.9.201	SSHv2	138	Client: Encrypted packet (len=84)

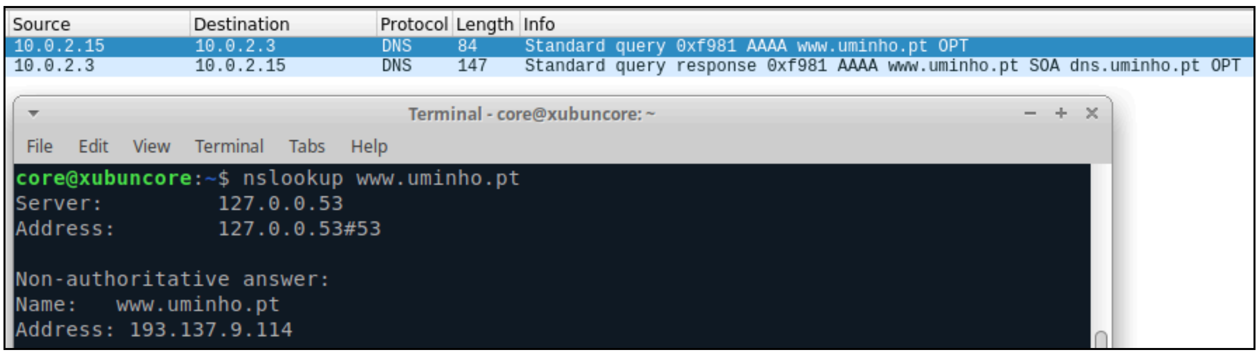
```
File  Edit  View  Terminal  Tabs  Help
core@xubuncore:~$ ssh cc@cc2024.ddns.net
cc@cc2024.ddns.net's password:
Welcome to Ubuntu 24.04 LTS (GNU/Linux 5.15.133.1-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Wed Sep 25 13:54:17 2024 from 172.17.0.1

.....
.....

Hello, cc. You have successfully logged in via ssh. This server is for tests only.
e session is over. Thanks!
Olá utilizador cc. O login por SSH foi bem sucedido. Este servidor é só para teste
A sessão terminou. Obrigado!
```

Captura SSH

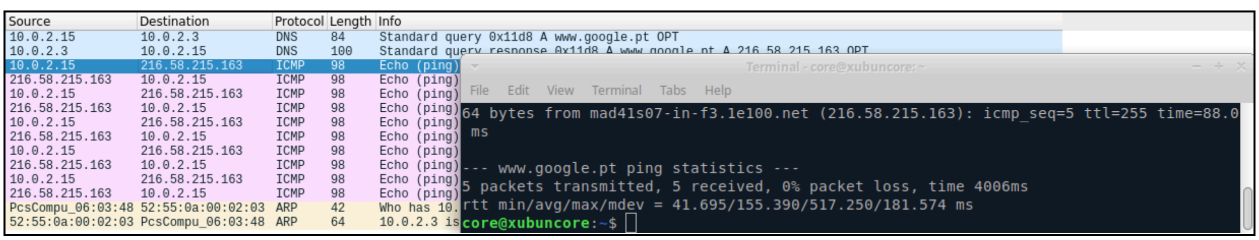
Parte II - Uso da Camada de Transporte por Parte das Aplicações (cont.)



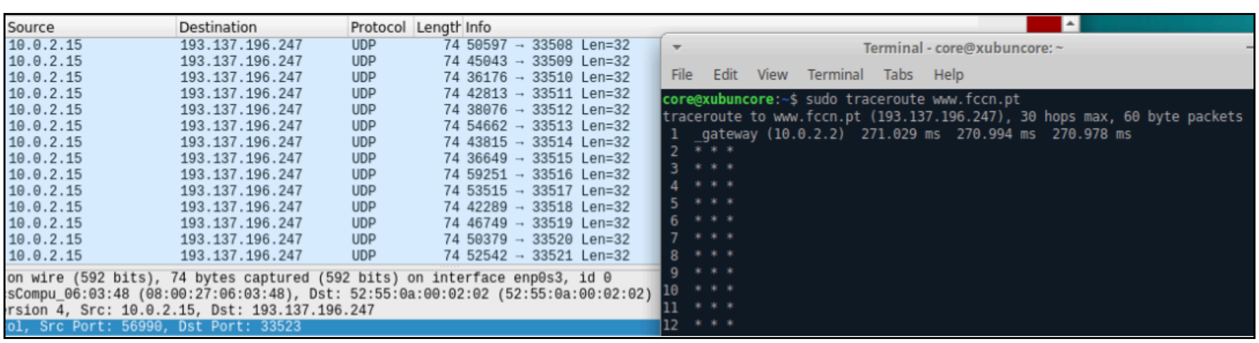
Captura TFTP



Captura *nslookup*



Captura *ping*



Captura *traceroute*

Parte III - Utilização de Serviços de Transferência de Ficheiro no Ambiente CORE - 3.1 Transferência para PC3

Questão: Descarregue os ficheiros a partir do PC3 com os protocolos TFTP e FTP e responda:

a) De que forma as perdas de pacotes afetaram o desempenho das aplicações? Que camada lidou com as perdas - transporte ou aplicação? Responda com base nas experiências feitas e nos resultados observados.

b) Apresente um diagrama temporal para a transferência do file1 por FTP. Foque-se apenas na transferência de dados [ftp-data] e não na conexão de controlo, pois o FTP usa mais que uma conexão em simultâneo. Identifique, se aplicável, as fases de início de conexão, transferência de dados e fim de conexão.

c) Apresente um diagrama temporal para a transferência do file1 por TFTP. Identifique, se aplicável, as fases de início de conexão, transferência de dados e fim de conexão. Identifique também os tipos de segmentos trocados e os números de sequência usados quer nos dados como nas confirmações.

Resposta:

a) Para o caso do TFTP, é a camada da aplicação a responsável por lidar com perdas de pacotes. Tendo por base UDP, que não é capaz de controlar fluxo, usa o método "stop and wait", que consiste na espera de confirmação por todos os blocos de dados enviados. Havendo timeout para uma confirmação, há reenvio, mas, ao contrário das aplicações baseadas em TCP, não tira partido da estratégia de Fast Retransmit para acelerar retransmissões. Ao lado, vê-se um exemplo do envio duplicado tanto de blocos de dados num sentido como de ACKs para o outro.

```
Data Packet, Block: 8
Acknowledgement, Block: 8
Data Packet, Block: 9
Acknowledgement, Block: 9
Hello Packet
Hello Packet
Acknowledgement, Block: 9
Data Packet, Block: 9
Acknowledgement, Block: 9
Data Packet, Block: 10
Acknowledgement, Block: 10
Data Packet, Block: 10
Acknowledgement, Block: 10
```

b) Foi aplicado um filtro *´ftp-data || ftp´* na captura para se conseguir analisar de modo isolado todos os pacotes FTP trocados entre as máquinas. Verifica-se assim que é dirigida para a porta 21 do servidor (a porta de controlo) toda a comunicação relacionada com inícios/conclusões de ligação, envio de comandos e reconhecimentos/acknowledgements recíprocos.

O início da conexão FTP estabelece-se quando o servidor responde ao cliente recém-ligado com o código 220, que indica que estar a postos para lhe comunicar.

Após o login e a colocação em modo binário - o mais apto para transferências - o comando "get file1" leva o cliente a pedir acesso à porta 20 (usa-se uma porta separada para transferências de dados) e subsequente pedido de transferência do file1.

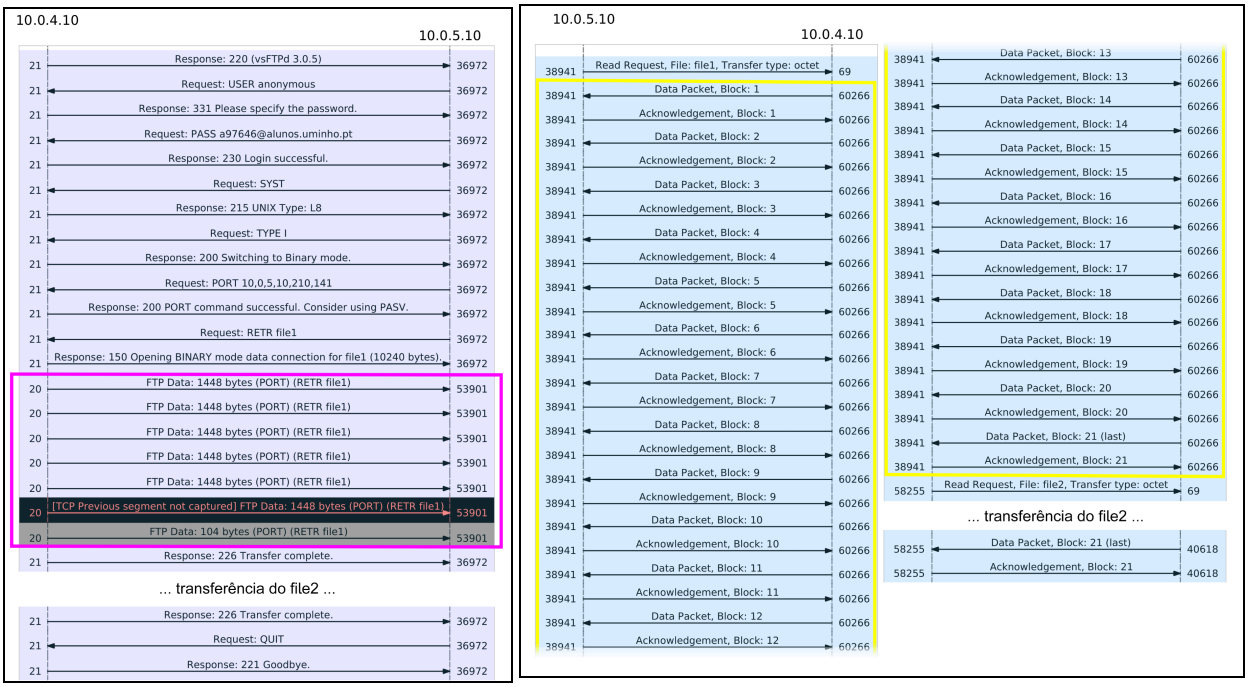
Ambas máquinas usam portas alheias às de controlo FTP para transferência de dados. No caso do servidor, é dedicada a 20 por protocolo, e, no lado do cliente, foi usada a 53901 neste caso. Assim, é o primeiro pacote da conexão de portas 20 a 53901 que marca o início da transferência. Mal se conclua, fecha-se a conexão de dados e o servidor volta a usar a de controlo para informar de tal como o código 226, que indica isso exatamente.

O processo repete-se para transferir file2. Mais tarde, o comando "quit" no cliente despoleta o pedido QUIT ao servidor, que o leva a desconectar a porta 21 logo após responder ao pedido com o código 221.

Observar, no fim das alíneas, no diagrama de fluxo da figura.

Parte III - 3.1 Transferência para PC3 (cont.)

c) Ao contrário do FTP, o TFTP é um serviço baseado em pacotes de protocolo UDP. Sob este, toda a comunicação se resume à transmissão de dados, não existindo troca de pacotes dedicados a reconhecimento de início/fim de ligação entre as duas máquinas dum estiloanálogo ao do TCP.



esquerda: Comunicação FTP pelas conexões de controlo e de dados (transferência *file1* a magenta)

direita: Comunicação TFTP (transferência *file1* a amarelo)

Parte III - 3.2 Transferência para PC2

Questão: Descarregue os ficheiros a partir do PC2 com os protocolos TFTP, FTP e HTTP e responda…

a) Na transferência HTTP…

- Identifique o início e o fim da sessão TCP e analise como os números de sequência e ACKs são usados na conexão.
- Identifique o número de sequência inicial e analise como ele é incrementado com cada pacote tanto pelo cliente quanto pelo servidor.

b) Qual dos protocolos seria o mais adequado para a obtenção dos ficheiros pelo PC2? Justifique.

Source	Destination	Protocol	Length	Info
00:00:00…	00:00:00_…	ARP	42	10.0.7.1 is at 00:00:00:aa:00:1f
10.0.7.10	10.0.4.10	TCP	74	39492 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1502153828 TSecr=0 WS=128
10.0.4.10	10.0.7.10	TCP	74	80 → 39492 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2518838269 TSecr=1502153828 WS=128
10.0.7.10	10.0.4.10	TCP	66	39492 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1502153830 TSecr=2518838269
10.0.7.10	10.0.4.10	HTTP	302	GET /file1 HTTP/1.0
10.0.4.10	10.0.7.10	TCP	66	80 → 39492 [ACK] Seq=1 Ack=237 Win=65024 Len=0 TSval=2518838271 TSecr=1502153831
10.0.4.10	10.0.7.10	TCP	15…	80 → 39492 [ACK] Seq=1 Ack=237 Win=65024 Len=1448 TSval=2518838273 TSecr=1502153831 [TCP segment of a reassembled PDU]
10.0.7.10	10.0.4.10	TCP	66	39492 → 80 [ACK] Seq=237 Ack=1449 Win=62848 Len=0 TSval=1502153837 TSecr=2518838273
10.0.4.10	10.0.7.10	TCP	15…	80 → 39492 [ACK] Seq=1449 Ack=237 Win=65024 Len=1448 TSval=2518838273 TSecr=1502153831 [TCP segment of a reassembled PDU]
10.0.7.10	10.0.4.10	TCP	66	39492 → 80 [ACK] Seq=237 Ack=2897 Win=62720 Len=0 TSval=1502153839 TSecr=2518838273
10.0.4.10	10.0.7.10	TCP	15…	80 → 39492 [ACK] Seq=2897 Ack=237 Win=65024 Len=1448 TSval=2518838273 TSecr=1502153831 [TCP segment of a reassembled PDU]
10.0.7.10	10.0.4.10	TCP	66	39492 → 80 [ACK] Seq=237 Ack=4345 Win=62720 Len=0 TSval=1502153840 TSecr=2518838273
10.0.4.10	10.0.7.10	TCP	15…	80 → 39492 [ACK] Seq=4345 Ack=237 Win=65024 Len=1448 TSval=2518838273 TSecr=1502153831 [TCP segment of a reassembled PDU]
10.0.7.10	10.0.4.10	TCP	66	39492 → 80 [ACK] Seq=237 Ack=5793 Win=62720 Len=0 TSval=1502153841 TSecr=2518838273
10.0.4.10	10.0.7.10	TCP	15…	80 → 39492 [PSH, ACK] Seq=5793 Ack=237 Win=65024 Len=1448 TSval=2518838273 TSecr=1502153831 [TCP segment of a reassembled
10.0.7.10	10.0.4.10	TCP	66	39492 → 80 [ACK] Seq=237 Ack=7241 Win=62720 Len=0 TSval=1502153843 TSecr=2518838273
10.0.4.10	10.0.7.10	TCP	15…	80 → 39492 [ACK] Seq=7241 Ack=237 Win=65024 Len=1448 TSval=2518838273 TSecr=1502153831 [TCP segment of a reassembled PDU]
10.0.7.10	10.0.4.10	TCP	66	39492 → 80 [ACK] Seq=237 Ack=8689 Win=62720 Len=0 TSval=1502153844 TSecr=2518838273
10.0.4.10	10.0.7.10	TCP	15…	80 → 39492 [PSH, ACK] Seq=8689 Ack=237 Win=65024 Len=1448 TSval=2518838273 TSecr=1502153831 [TCP segment of a reassembled
10.0.7.10	10.0.4.10	TCP	66	39492 → 80 [ACK] Seq=237 Ack=10137 Win=62720 Len=0 TSval=1502153845 TSecr=2518838273
10.0.4.10	10.0.7.10	HTTP	390	HTTP/1.0 200 Ok (text/plain)
10.0.7.10	10.0.4.10	TCP	66	39492 → 80 [FIN, ACK] Seq=237 Ack=10462 Win=64128 Len=0 TSval=1502153847 TSecr=2518838273
10.0.4.10	10.0.7.10	TCP	66	80 → 39492 [ACK] Seq=10462 Ack=238 Win=65024 Len=0 TSval=2518838287 TSecr=1502153847
10.0.7.1	224.0.0.5	OSPF	78	Hello Packet

Source	Destination	Protocol	Length	Info
10.0.7.1	224.0.0.5	OSPF	78	Hello Packet
10.0.7.10	10.0.4.10	TCP	74	49468 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1502176550 TSecr=0 WS=128
10.0.4.10	10.0.7.10	TCP	74	80 → 49468 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2518860990 TSecr=1502176550 WS=128
10.0.7.10	10.0.4.10	TCP	66	49468 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1502176551 TSecr=2518860990
10.0.7.10	10.0.4.10	HTTP	302	GET /file2 HTTP/1.0
10.0.4.10	10.0.7.10	TCP	66	80 → 49468 [ACK] Seq=1 Ack=237 Win=65024 Len=0 TSval=2518860993 TSecr=1502176553
10.0.4.10	10.0.7.10	TCP	15…	80 → 49468 [ACK] Seq=1 Ack=237 Win=65024 Len=1448 TSval=2518860995 TSecr=1502176553 [TCP segment of a reassembled PDU]
10.0.7.10	10.0.4.10	TCP	66	49468 → 80 [ACK] Seq=237 Ack=1449 Win=62848 Len=0 TSval=1502176560 TSecr=2518860995
10.0.4.10	10.0.7.10	TCP	15…	80 → 49468 [ACK] Seq=1449 Ack=237 Win=65024 Len=1448 TSval=2518860995 TSecr=1502176553 [TCP segment of a reassembled PDU]
10.0.7.10	10.0.4.10	TCP	66	49468 → 80 [ACK] Seq=237 Ack=2897 Win=62720 Len=0 TSval=1502176561 TSecr=2518860995
10.0.4.10	10.0.7.10	TCP	15…	80 → 49468 [ACK] Seq=2897 Ack=237 Win=65024 Len=1448 TSval=2518860995 TSecr=1502176553 [TCP segment of a reassembled PDU]
10.0.7.10	10.0.4.10	TCP	66	49468 → 80 [ACK] Seq=237 Ack=4345 Win=62720 Len=0 TSval=1502176563 TSecr=2518860995
10.0.4.10	10.0.7.10	TCP	15…	80 → 49468 [ACK] Seq=4345 Ack=237 Win=65024 Len=1448 TSval=2518860995 TSecr=1502176553 [TCP segment of a reassembled PDU]
10.0.7.10	10.0.4.10	TCP	66	49468 → 80 [ACK] Seq=237 Ack=5793 Win=62720 Len=0 TSval=1502176564 TSecr=2518860995
10.0.4.10	10.0.7.10	TCP	15…	80 → 49468 [PSH, ACK] Seq=5793 Ack=237 Win=65024 Len=1448 TSval=2518860995 TSecr=1502176553 [TCP segment of a reassembled
10.0.7.10	10.0.4.10	TCP	66	49468 → 80 [ACK] Seq=237 Ack=7241 Win=61312 Len=0 TSval=1502176565 TSecr=2518860995
10.0.4.10	10.0.7.10	TCP	15…	80 → 49468 [ACK] Seq=7241 Ack=237 Win=65024 Len=1448 TSval=2518860995 TSecr=1502176553 [TCP segment of a reassembled PDU]
10.0.7.10	10.0.4.10	TCP	66	49468 → 80 [ACK] Seq=237 Ack=8689 Win=59904 Len=0 TSval=1502176567 TSecr=2518860995
10.0.4.10	10.0.7.10	TCP	15…	80 → 49468 [PSH, ACK] Seq=8689 Ack=237 Win=65024 Len=1448 TSval=2518860995 TSecr=1502176553 [TCP segment of a reassembled
10.0.7.10	10.0.4.10	TCP	66	49468 → 80 [ACK] Seq=237 Ack=10137 Win=62720 Len=0 TSval=1502176568 TSecr=2518860995
10.0.4.10	10.0.7.10	HTTP	390	HTTP/1.0 200 Ok (text/plain)
10.0.7.10	10.0.4.10	TCP	66	49468 → 80 [FIN, ACK] Seq=237 Ack=10462 Win=64128 Len=0 TSval=1502176569 TSecr=2518860995
10.0.4.10	10.0.7.10	TCP	66	80 → 49468 [ACK] Seq=10462 Ack=238 Win=65024 Len=0 TSval=2518861009 TSecr=1502176569
fe80::20…	ff02::5	OSPF	90	Hello Packet

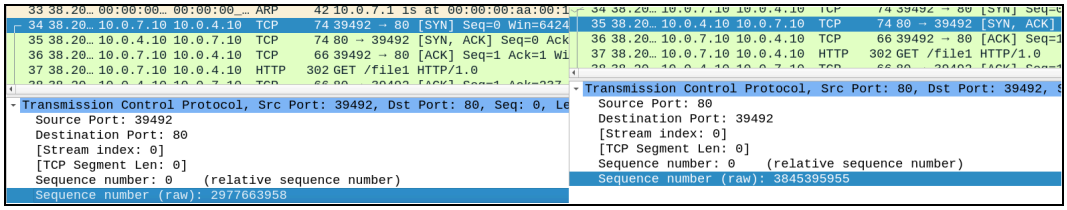
Comunicação HTTP com pedido de *file1* (cima) e *file2* (baixo)

Parte III - 3.2 Transferência para PC2 (cont.)

Resposta:

a) Tendo por base a transferência de *file1*:

O início da conexão consiste nos três primeiros segmentos, de flags SYN, SYNACK e ACK.



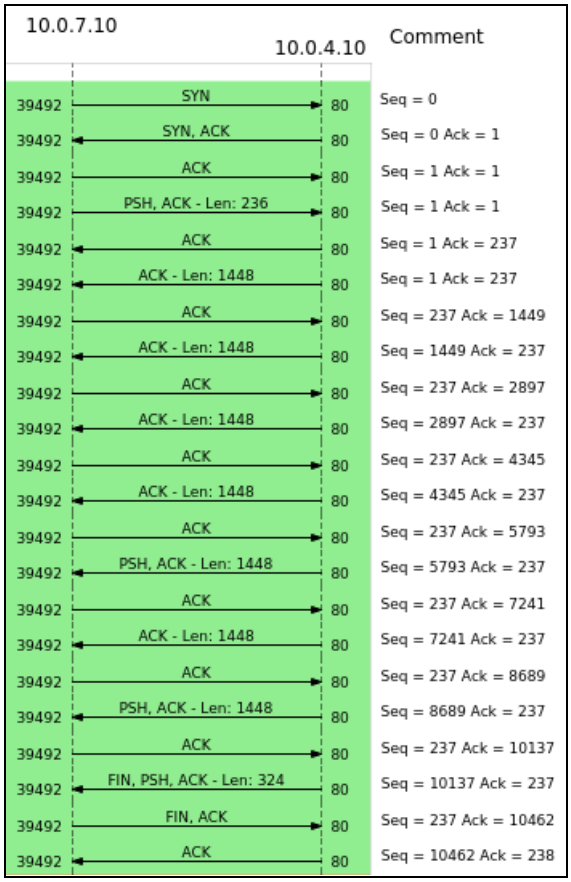
Cada uma das máquinas constituintes define um número de sequência inicial. No servidor PC1 (10.0.4.10), é 3845395955, enquanto que o do cliente PC2 (10.0.7.10) é 2977663958.

Se PC2 envia um SYN com Seq=2977663958, então o SYNACK de PC1 tem Ack=297766395+1, confirmando a última receção e esperando o segmento seguinte, de Seq=2977663958+1. Note-se que os valores na imagem lateral são relativos.

Seguindo a mesma lógica, se o SYNACK de PC1 é de Seq=3845395955, então PC2 responde com um ACK tal que Ack=3845395955+1.

Em modo geral, o Ack Number dos segmentos dum determinado host pode ser interpretado como o **próximo byte contíguo esperado** (todavia raramente partindo do valor inicial 0). Daí, a receção de N bytes faz com que o ACK recíproco tenha N bytes acrescidos comparado ao anterior.

Como o PC2 se limita a receber dados, os seus ACKs mantêm o constante Seq=237 pois corresponde a 1 byte enviado na fase inicial da conexão somado a 236 provenientes do comando *get file1*. Por outro lado, os Ack Numbers vão incrementando por 1448, o mesmo número de bytes de dados em cada das sequências recebidas.



Como o PC1 envia dados em segmentos de 1448 bytes, vêm daí as sucessivas incrementações desse valor ao Sequence Number e a estagnação do Ack Number em 237 pelo referido atrás.

Parte III - 3.2 Transferência para PC2 (cont.)

b) A transferência de PC1 a PC2 é feita em condições relativamente herméticas, num canal que não sofre perdas e com largura de banda 10^4 vezes o tamanho dos ficheiros. Nesta disposição, não se considera justificável o overhead TCP inerente aos protocolos HTTP e FTP, cuja utilidade se revela mais em canais suscetíveis a ruído e falhas de entrega. Assim, bastando o UDP como protocolo de transporte, o TFTP é preferível.

Parte III - 3.3 Transferência para PC4

Questão: Descarregue os ficheiros a partir do PC4 com os protocolos TFTP, FTP e HTTP e responda:

a) Na transferência HTTP:

- Identifique a perda e a duplicação de pacotes numa sessão TCP.
- Explique o impacto da perda e duplicação de pacotes numa sessão TCP, bem como os mecanismos usados pelo TCP para lidar com estas situações.

b) Qual dos protocolos seria o mais adequado para a obtenção dos ficheiros pelo PC4? Justifique.

Resposta:

a) Atente-se, abaixo, na transmissão do *file2* do PC1 (10.0.4.10) para o PC4 (10.0.6.10), em que os pacotes partidos do PC4 estão coloridos de cinza e os de PC1 coloridos de verde.

Src	Dest	Proto	In Flight	Seq#	Ack#	Calc. Window Size	TCP Seg Len	Info
10.0.6…	10.0.4…	TCP		0	0	64240	0	42756 → 80 [SYN] Seq=0 Win=64240 Len
10.0.6…	10.0.4…	TCP		0	0	64240	0	[TCP Retransmission] 42756 → 80 [SYN
10.0.4…	10.0.6…	TCP		0	1	65160	0	80 → 42756 [SYN, ACK] Seq=0 Ack=1 Wi
10.0.6…	10.0.4…	TCP		1	1	64256	0	42756 → 80 [ACK] Seq=1 Ack=1 Win=642
10.0.6…	10.0.4…	HTTP	236	1	1	64256	236	GET /file2 HTTP/1.0
10.0.4…	10.0.6…	TCP		1	237	65024	0	80 → 42756 [ACK] Seq=1 Ack=237 Win=6
10.0.4…	10.0.6…	TCP		1	237	65024	0	[TCP Dup ACK 57#1] 80 → 42756 [ACK]
10.0.4…	10.0.6…	TCP	1448	1	237	65024	1448	80 → 42756 [ACK] Seq=1 Ack=237 Win=6
10.0.6…	10.0.4…	TCP		237	1449	62848	0	42756 → 80 [ACK] Seq=237 Ack=1449 Wi
10.0.4…	10.0.6…	TCP		10137	237	65024	324	[TCP Previous segment not captured]
10.0.6…	10.0.4…	TCP		237	1449	64128	0	[TCP Window Update] 42756 → 80 [ACK]
10.0.4…	10.0.6…	TCP	9013	1449	237	65024	1448	[TCP Out-Of-Order] 80 → 42756 [ACK]
10.0.6…	10.0.4…	TCP		237	2897	63616	0	42756 → 80 [ACK] Seq=237 Ack=2897 Wi
10.0.4…	10.0.6…	TCP	7565	2897	237	65024	1448	[TCP Out-Of-Order] 80 → 42756 [ACK]
10.0.6…	10.0.4…	TCP		237	4345	62208	0	42756 → 80 [ACK] Seq=237 Ack=4345 Wi
10.0.4…	10.0.6…	TCP	6117	4345	237	65024	1448	[TCP Out-Of-Order] 80 → 42756 [PSH,
10.0.6…	10.0.4…	TCP		237	5793	62080	0	42756 → 80 [ACK] Seq=237 Ack=5793 Wi
10.0.4…	10.0.6…	TCP	4669	5793	237	65024	1448	[TCP Out-Of-Order] 80 → 42756 [ACK]
10.0.6…	10.0.4…	TCP		237	7241	62080	0	42756 → 80 [ACK] Seq=237 Ack=7241 Wi
10.0.4…	10.0.6…	TCP	3221	7241	237	65024	1448	[TCP Out-Of-Order] 80 → 42756 [PSH,
10.0.6…	10.0.4…	TCP		237	8689	62080	0	42756 → 80 [ACK] Seq=237 Ack=8689 Wi
10.0.4…	10.0.6…	TCP	1773	8689	237	65024	1448	[TCP Out-Of-Order] 80 → 42756 [ACK]
10.0.6…	10.0.4…	TCP		237	10462	64128	0	42756 → 80 [ACK] Seq=237 Ack=10462 W
10.0.6…	10.0.4…	TCP		237	10462	64128	0	42756 → 80 [FIN, ACK] Seq=237 Ack=10
10.0.6…	10.0.4…	TCP		237	10462	64128	0	[TCP Retransmission] 42756 → 80 [FIN
10.0.4…	10.0.6…	TCP		10462	238	65024	0	80 → 42756 [ACK] Seq=10462 Ack=238 W

Há várias ocorrências de interesse:

- Retransmissão no SYN e no FINACK.
- O Ack pós comando HTTP é duplicado.
- É entregue a PC4 um segmento de Seq=10137 enquanto este contava com Seq=1449, formando uma lacuna.

Apesar de não ter acontecido qualquer envio de dados duplicados, é relevante referir que estes são descartados via comparação de números de sequência. Lacunas e/ou perdas no envio são detectadas pelo emissor ao ser respondido com ACKs duplicados (ou seja, ACKs com um Ack# igual a outro antecedente), suscitando uma retransmissão dos dados esperados. Note-se também, pelo decréscimo no tamanho da janela, que os dados entregues em avanço causaram uma retenção em buffer até a contiguidade se restaurar, visto que não podem ser entregues fora de ordem.

Parte III - 3.3 Transferência para PC4 (cont.)

b) Como, ao contrário da questão anterior, se coloca o caso duma transmissão num canal não fiável (vistas as probabilidades de 10% perda e 10% duplicação), faz mais sensata a aposta numa aplicação que use TCP como protocolo de transporte, pois este garante a entrega dos dados.

Sendo que o objetivo é uma conexão curta para a transmissão de dois ficheiros leves em tamanho, é preferível HTTP quando comparado com FTP, pois tem fases de início e término de conexão com muito menos troca de pacotes inerente. A menor fiabilidade do canal acrescenta possível demora adicional a estas fases, sendo que os segmentos destas também estão sujeitos a falhas de entrega ou duplicações. Isto faz do FTP mais orientado a conexões de cariz mais prolongado e estável, que não têm cabimento nesta situação específica.

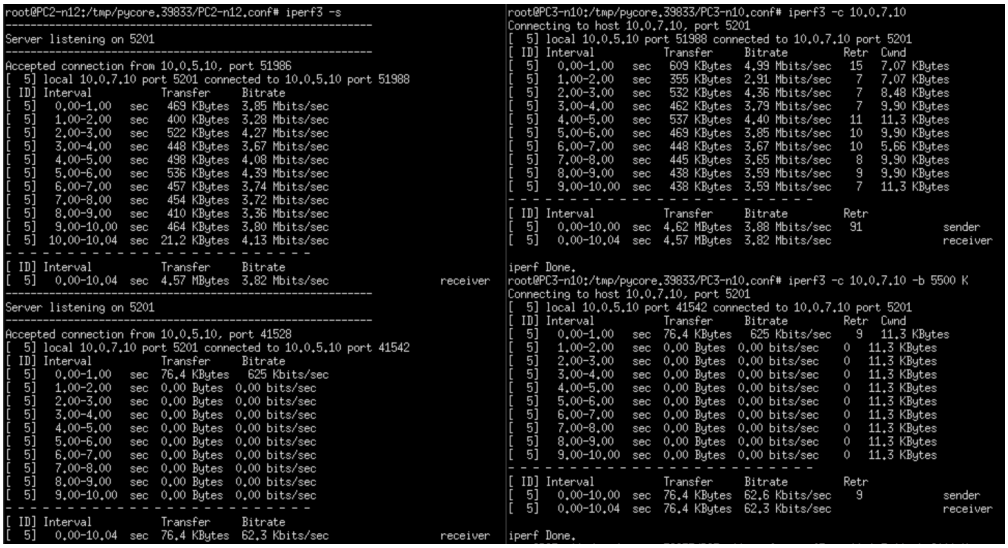
Parte III - 3.4 Congestão de Rede

Questão: Simule uma congestão de rede fazendo o iperf gerar uma taxa de bits por segundo superior à largura de banda do canal (conexão) a partir de um host. Investigue como a janela de congestão muda durante o evento de congestão.

- Explique como os mecanismos de controle de congestão do TCP (ex.: slow start, congestion avoidance) ajustam o fluxo de dados.
- Apresente a taxa de transferência (throughput) da conexão TCP e compare-a com o débito calculado na Parte I.
- Forneça imagens das capturas de tráfego para suportar suas observações.

Resposta:

Definiu-se PC2 (10.0.7.10) como servidor. PC3 (10.0.5.10) foi escolhido para cliente. Assim, o canal a ser estudado foi o de 5Mb/s e 1% chance de perda.



Ambas as conexões foram corridas para 10 segundos.

- Primeiro, correu-se uma conexão de controlo como na Parte I, ou seja, sem bitrate explícito, de modo a não congestionar o canal - Obteu-se um throughput médio de 3.82Mb/s no recetor e uma janela de congestão linearmente crescente até aos 5 segundos, onde começa a descer até metade possivelmente devido a uma perda.

- De seguida, configurou-se 5.5Mb/s. Nestas condições, a janela de congestão não chegou sequer a aumentar, e em média obteu-se um throughput de 62.3Kb/s fruto de só ter sido possível transferência durante o primeiro segundo de conexão.

- Como segunda tentativa (sem captura), configurou-se para 6Mb/s, que resultou no mesmo mas com uma maior CWND para o primeiro segundo, o que é expectável.

Uma conexão tem uma janela de congestão associada, derivada da janela de receção das duas máquinas ligadas (mas só relevante a que tem o papel de receção de dados, como é natural). Essa janela, inicialmente pequena por não ser conhecido o limite de throughput do canal (slow start), vai aumentando linearmente (Additive Increase) enquanto não houver indícios de congestão (por meio de ACKs duplicados). Nesta eventualidade, a implementação do TCP pode voltar ao slow start, deitando a perder todo o alargamento da janela por haver demasiado congestionamento, ou recorrer à Fast Recovery, uma alternativa que só decresce para metade do valor aquando da deteção de ACKs duplicados. Especialmente à conta do decréscimo da CWND durante o ensaio de controlo, conjecturamos que foi usada Fast Recovery.

Parte III - 3.4 Congestão de Rede (cont.)

Receive Window original do servidor:

Time	Src	Dest	Proto	SrcPort	DstPort	In Flight	Calc. Window Size	Window size value	TCP Seg Len	Info
10.0…	10.…	224…	OSPF							Hello Packet
11.9…	10.…	10.…	TCP	5201	51986		65160	65160	0	5201 → 51986
11.9…	10.…	10.…	TCP	5201	51986		65152	509	0	5201 → 51986
11.9…	10.…	10.…	TCP	5201	51986	1	65152	509	1	5201 → 51986
12.0…	10.…	224…	OSPF							Hello Packet
12.0…	10.…	10.…	TCP	5201	51986		65152	509	0	5201 → 51986
12.0…	10.…	10.…	TCP	5201	51986		65152	509	0	5201 → 51986
12.0…	10.…	10.…	TCP	5201	51986	1	65152	509	1	5201 → 51986

1º Ensaio: surge maior número de ACKs duplicados por volta dos 5 ou 6 segundos após início:

Time	Src	Dest	Proto	SrcPort	DstPort	In Flight	Calc. Window Size	Window size value	TCP Seg Len	Info
16.53…	10.0…	10.0.…	TCP	5201	51988		273152	2134	0	5201 → 51988
16.54…	10.0…	10.0.…	TCP	5201	51988		273152	2134	0	5201 → 51988
16.55…	10.0…	10.0.…	TCP	5201	51988		273152	2134	0	5201 → 51988
16.55…	10.0…	10.0.…	TCP	5201	51988		273152	2134	0	5201 → 51988
16.55…	10.0…	10.0.…	TCP	5201	51988		273152	2134	0	[TCP Dup ACK
16.56…	10.0…	10.0.…	TCP	5201	51988		273152	2134	0	[TCP Dup ACK
16.56…	10.0…	10.0.…	TCP	5201	51988		273152	2134	0	[TCP Dup ACK
16.57…	10.0…	10.0.…	TCP	5201	51988		273152	2134	0	[TCP Dup ACK
16.57…	10.0…	10.0.…	TCP	5201	51988		273152	2134	0	[TCP Dup ACK
16.57…	10.0…	10.0.…	TCP	5201	51988		273152	2134	0	[TCP Dup ACK
16.58…	10.0…	10.0.…	TCP	5201	51988		273152	2134	0	[TCP Dup ACK
16.58…	10.0…	10.0.…	TCP	5201	51988		273152	2134	0	[TCP Dup ACK
16.58…	10.0…	10.0.…	TCP	5201	51988		273152	2134	0	[TCP Dup ACK
16.61…	10.0…	10.0.…	TCP	5201	51988		273152	2134	0	[TCP Dup ACK
16.61…	10.0…	10.0.…	TCP	5201	51988		273152	2134	0	[TCP Dup ACK
16.61…	10.0…	10.0.…	TCP	5201	51988		273152	2134	0	[TCP Dup ACK
16.61…	10.0…	10.0.…	TCP	5201	51988		263040	2055	0	5201 → 51988

Conclusão

Em síntese, o trabalho consistiu tanto numa breve retoma de alguns conceitos primeiramente abordados em Redes de Computadores, como no aprofundar sobre o funcionamento básico das aplicações de comunicação e dos protocolos de transporte que lhes servem de alicerce.