

CC-TP1-PL1-G4

David Figueiredo^[a104360], Diogo Ferreira^[a104266] e Gustavo Paiva^[e12032]

¹ Universidade do Minho, Largo do Paço 4704-553 Braga

Abstract. Este relatório relata a resolução da ficha teórico-prática 1, apresentando imagens alusivas à resolução dos problemas, respostas e questões relativas aos exercícios e situações propostas para utilização do software de design de redes *Core*. Ainda é provido de uma secção onde são fornecidas conclusões retiradas do conhecimento aplicado às questões do enunciado.

Keywords: Core, TCP, Protocolo,FTP,TFTP,HTTP..

1 Instalação, configuração e validação da rede de testes

1.1 Instalação

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.

Inicie a simulação e responda às questões seguintes:

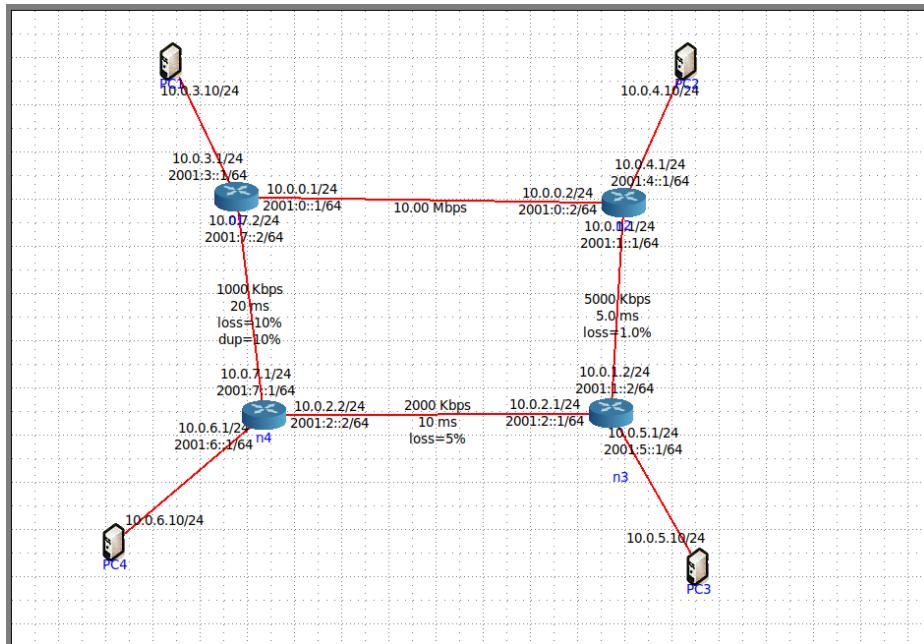


Fig. 1 - Topologia da rede

1.2 Verificação da conectividade

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.

Depois de estar a topologia construída, inicia-se a simulação de maneira a ser possível avaliar se a rede foi bem desenhada de modo a providenciar conectividade entre os hosts.

Em vista a esse objetivo, testamos a conectividade do host 1 a todos os outros, através de comandos ping e traceroute. Partindo do pressuposto de que se um traceroute do n1 para n3 tem de passar por n2 ou n3, existe conectividade entre os intermediários, se se obtiver resposta do traceroute. Isto é evidenciado pela figura 2.

```

root@PC1:/tmp/pycore.39991/PC1.conf# traceroute 10.0.4.10
traceroute to 10.0.4.10 (10.0.4.10), 30 hops max, 60 byte packets
 1  10.0.3.1 (10.0.3.1)  0.056 ms  0.011 ms  0.009 ms
 2  10.0.0.2 (10.0.0.2)  0.264 ms  0.337 ms  0.472 ms
 3  10.0.4.10 (10.0.4.10)  0.453 ms  0.398 ms  0.385 ms
root@PC1:/tmp/pycore.39991/PC1.conf# traceroute 10.0.5.10
traceroute to 10.0.5.10 (10.0.5.10), 30 hops max, 60 byte packets
 1  10.0.3.1 (10.0.3.1)  0.057 ms  0.010 ms  0.008 ms
 2  10.0.0.2 (10.0.0.2)  0.304 ms  0.449 ms  0.428 ms
 3  10.0.1.2 (10.0.1.2)  17.422 ms  17.274 ms *
 4  10.0.5.10 (10.0.5.10)  17.726 ms  17.573 ms *
root@PC1:/tmp/pycore.39991/PC1.conf# traceroute 10.0.6.10
traceroute to 10.0.6.10 (10.0.6.10), 30 hops max, 60 byte packets
 1  10.0.3.1 (10.0.3.1)  0.049 ms  0.009 ms  0.007 ms
 2  10.0.7.1 (10.0.7.1)  45.721 ms * 50.755 ms
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * 10.0.6.10 (10.0.6.10)  47.679 ms  42.804 ms
root@PC1:/tmp/pycore.39991/PC1.conf# iperf 10.0.6.10
iperf: ignoring extra argument -- 10.0.6.10
Usage: iperf [-s|-c host] [options]
Try 'iperf --help' for more information.
root@PC1:/tmp/pycore.39991/PC1.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data.
64 bytes from 10.0.4.10: icmp_seq=1 ttl=62 time=0.406 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=62 time=9.67 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=62 time=15.1 ms
^C
--- 10.0.4.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2026ms
rtt min/avg/max/mdev = 0.406/8.406/15.147/6.083 ms
root@PC1:/tmp/pycore.39991/PC1.conf# ping 10.0.5.10
PING 10.0.5.10 (10.0.5.10) 56(84) bytes of data.
64 bytes from 10.0.5.10: icmp_seq=2 ttl=61 time=31.6 ms
64 bytes from 10.0.5.10: icmp_seq=3 ttl=61 time=38.3 ms
^C
--- 10.0.5.10 ping statistics ---
3 packets transmitted, 2 received, 33.333% packet loss, time 2023ms
rtt min/avg/max/mdev = 31.576/34.913/38.250/3.337 ms
root@PC1:/tmp/pycore.39991/PC1.conf# ping 10.0.6.10
PING 10.0.6.10 (10.0.6.10) 56(84) bytes of data.
64 bytes from 10.0.6.10: icmp_seq=1 ttl=62 time=43.8 ms
64 bytes from 10.0.6.10: icmp_seq=3 ttl=62 time=53.7 ms
64 bytes from 10.0.6.10: icmp_seq=4 ttl=62 time=61.8 ms
64 bytes from 10.0.6.10: icmp_seq=5 ttl=62 time=56.3 ms
^C
--- 10.0.6.10 ping statistics ---
5 packets transmitted, 4 received, 20% packet loss, time 4070ms
rtt min/avg/max/mdev = 43.817/53.905/61.759/6.506 ms
root@PC1:/tmp/pycore.39991/PC1.conf# ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defau
lt qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
21: eth0@if22: <BRIDGE,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state U
P group default qlen 1000
    link/ether 00:00:00:aa:00:07 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.3.10/24 scope global eth0
        valid_lft forever preferred_lft forever
        inet6 fe80::200:ff:fea:7/64 scope link
            valid_lft forever preferred_lft forever
root@PC1:/tmp/pycore.39991/PC1.conf#

```

Fig. 1 - Teste da conectividade entre os hosts

1.3 Otimização das rotas

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?**

OBS: Salve a topologia para uso na Parte III deste TP

Partindo das restrições do enunciado, algumas rotas são mais limitadas, levando a que algumas conexões tenham perdas ou não sejam eficientes. As limitações com que estamos a lidar são as seguintes:

Entre os nós 1 e 3: 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.

Analisando discriminadamente, a rota entre os nós 1 e 3 apresenta um débito alto, com atraso e perdas nulas ou próximo de nulo. Esta rota deve ser uma prioridade por não representar um bottleneck da rede.

Entre os nós 2 e 3, há um débito menor, um pequeno atraso e perdas. Esta rota deve ser tida em consideração por ser um bottleneck menor para a rede.

Entre os nós 3 e 4, já há um débito ainda menor, um atraso relativamente alto e alguma perda de pacotes, representando uma rota a evitar se possível.

Por fim, entre os nós 1 e 4, há o bottleneck maior, a utilizar só em caso de necessidade ou se a rede não for utilizada para tráfego sensível a perdas, por esta apresentar o débito menor, o maior atraso e perdas da rota. Para além disso, possui ainda duplicação de pacotes.

Feita a análise dos valores de cada rota, é evidente a necessidade de evitar a rota entre n1 e n4 para diminuir o bottleneck a que o tráfego está sujeito. Como as outras rotas não representam um bottleneck tão significativo como esta, essas rotas devem ser utilizadas pela prioridade n1 a n2; n2 a n3 e n3 a n4.

Se se mantiver o protocolo OSPF, sendo que, por defeito, este utiliza um grafo sem pesos, o que leva a que o cálculo não tenha em conta os bottlenecks que alguns caminhos geram na rede. Se nada for configurado, o cálculo é feito com base no número de saltos. Por isso, para melhorar o protocolo OSPF, é necessário definir a métrica para o cálculo das rotas a utilizar, utilizando os valores de débito, atraso,

perda e duplicações. Por isso, o algoritmo passa a conseguir diferenciar os caminhos com bottleneck e o cálculo seja mais efetivo e a rede mais eficiente.

2 Uso de camada de transporte por parte das aplicações

2.1 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

Nesta fase do trabalho, é pedido que se utilize um software de captura de pacotes (Wireshark), para capturar o tráfego em que se utiliza alguns protocolos, determinando assim o protocolo de aplicação utilizado no comando que se utiliza, assim como o protocolo de transporte, porta de atendimento do serviço e overhead de transporte em bytes.

Para isso, as próximas figuras demonstram os pacotes que foram capturados para serem utilizados como referência para obter os valores. Depois segue-se uma figura onde se apresenta os valores retirados dos pacotes.

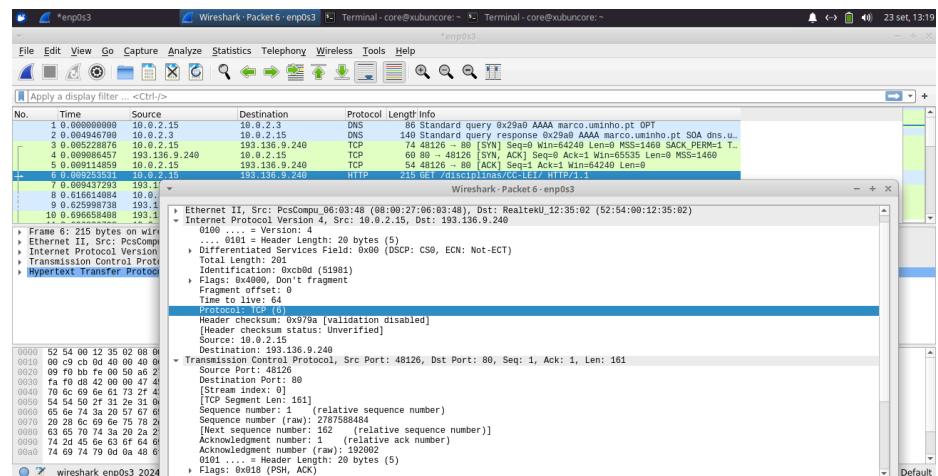


Fig. 3 - Captura de pacote obtido com comando wget

Neste pacote, existe um overhead de 40 bytes, que derivam 20 bytes do cabeçalho do segmento IP, e 20 bytes do segmento TCP.

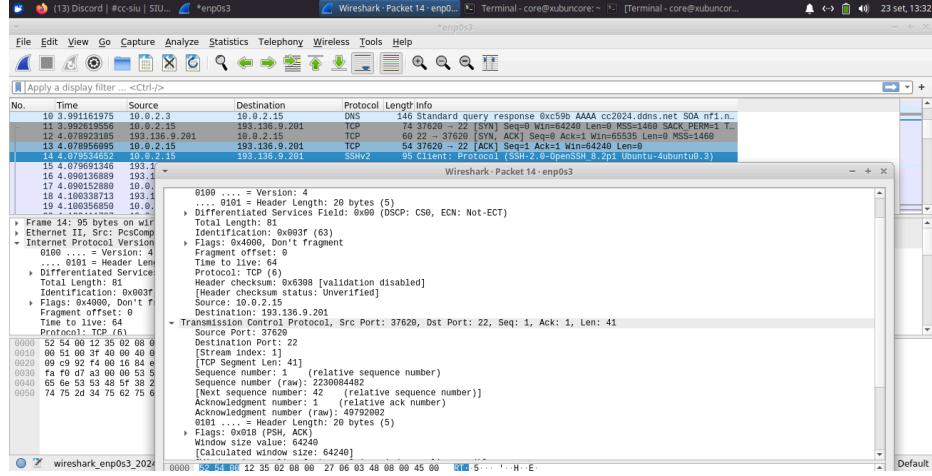


Fig. 4 - Captura de pacote obtido com comando ssh

Neste pacote, existe um 40 bytes, sendo que o datagrama IP tem um cabeçalho de 20 bytes, e o datagrama TCP necessita de 20 bytes.

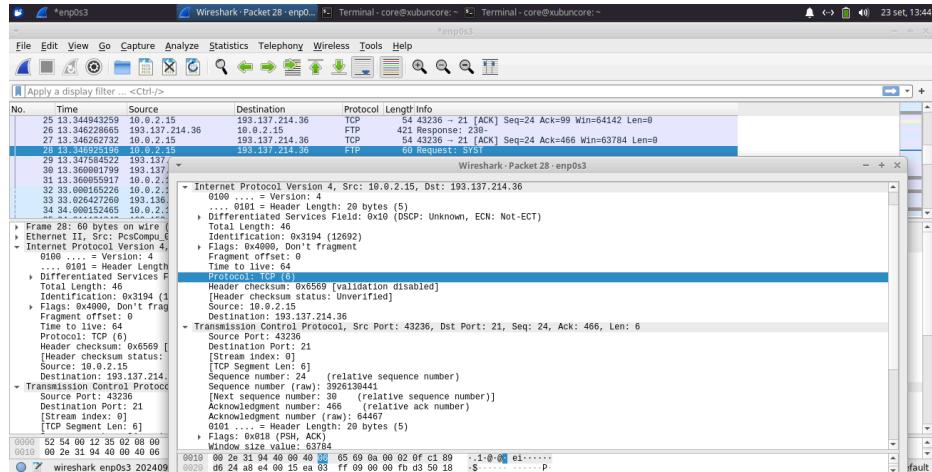


Fig. 5 - Captura de pacote obtido com comando ftp

Neste pacote, existe um 40 bytes, sendo que o datagrama IP tem um cabeçalho de 20 bytes, e o datagrama TCP necessita de 20 bytes.

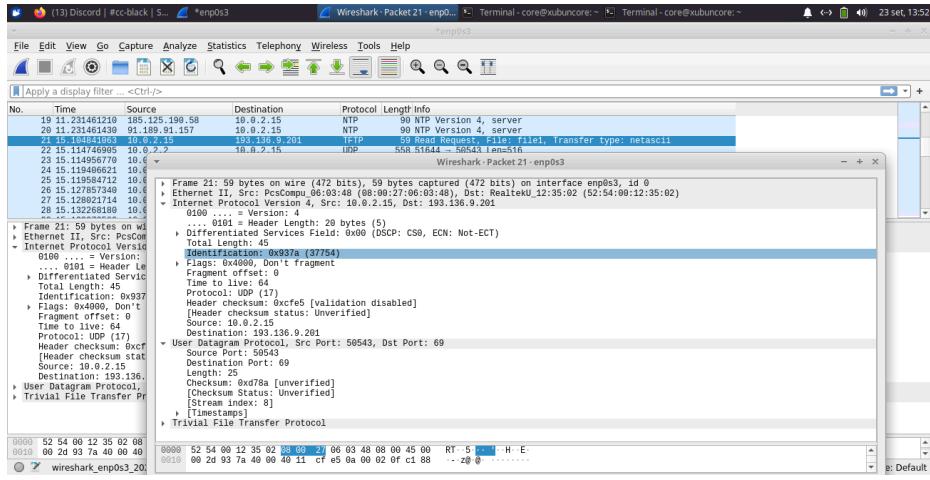


Fig. 6 - Captura de pacote obtido com comando tftp

Neste pacote, há um overhead de 28 bytes, sendo 20 bytes relativos ao datagrama IP, e 8 bytes relativos ao datagrama UDP.

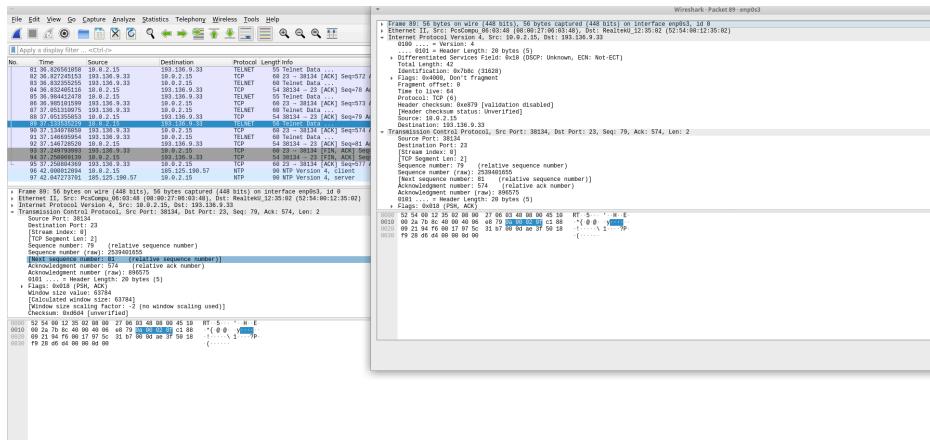


Fig. 7 - Captura de pacote obtido com comando telnet

Neste pacote, há um overhead de 40 bytes, sendo a soma de 20 bytes do datagrama IP com 20 bytes do datagrama TCP.

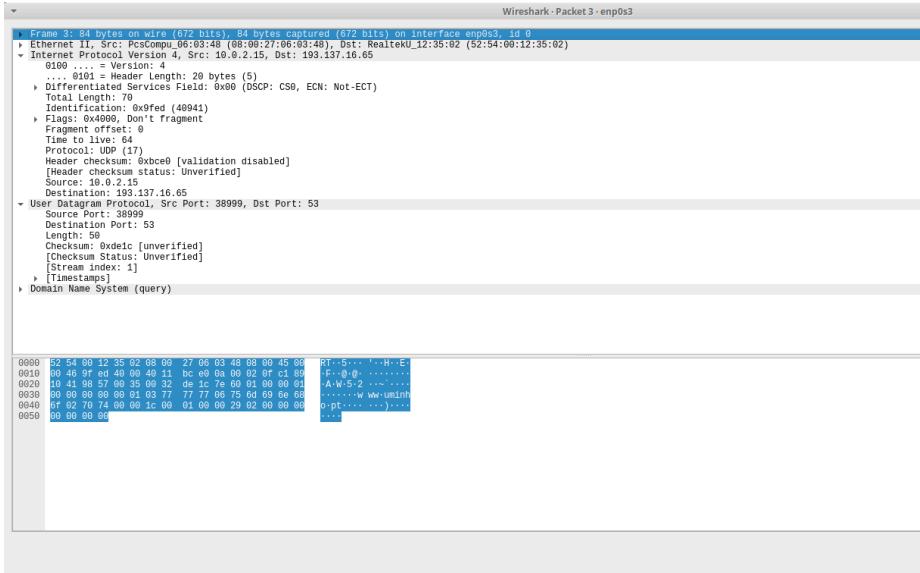


Fig. 8 - Captura de pacote obtido com comando nslookup

Neste pacote, existe um pacote com um overhead de 28 bytes, sendo 20 bytes para o datagrama IP e os restantes 8 bytes para o datagrama UDP.

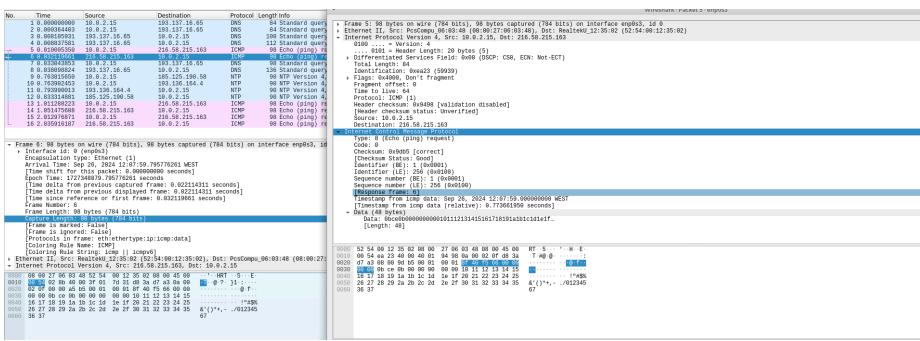


Fig. 9 - Captura de pacote obtido com comando ping

Neste pacote, o overhead é de 36 bytes. Destes 20 pertencem ao datagrama IP. Os restantes 16 bytes são a diferença do tamanho dos dados do datagrama ICMP (48 bytes) para um datagrama com 64 bytes.

Comando usado	Protocolo de Aplicação	Protocolo de transporte	Porta de atendimento	Overhead de transporte em bytes
wget	HTTP	TCP	80	40
ssh	SSHv2	TCP	22	40
ftp	FTP	TCP	21	40
tftp	TFTP	UDP	69	28
telnet	TELNET	TCP	23	40
nslookup	N/A	UDP	53	28
ping	ICMP	N/A	N/A	36
traceroute	UDP	N/A	N/A	28

Fig. 10 Tabela Serviços de transferências

3 Utilização dos serviços de transferência de ficheiros no ambiente Core

Neste exercício pretende-se transferir ficheiros utilizando os protocolos TFTP, FTP e HTTP no ambiente do CORE. Para tal, deve-se criar os ficheiros (file1 e file2) para as transferências e correr os servidores/clientes, conforme as instruções no Anexo I. Utilize a topologia criada na Parte I onde o PC1 deverá ser utilizado como servidor -- o host que possui o ficheiro a ser partilhado com os demais.

3.1 TFTP e FTP

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.

No caso do FTP, a perda de pacotes tem a gestão feita pelo TCP, retransmitindo os pacotes perdidos e ajustando a janela de controlo de congestionamento. Logo, é a camada de transporte que lida com perdas.

No TFTP, por este utilizar UDP, a gestão é feita na aplicação. O UDP retransmite blocos de dados que não são correspondidos por um ACK. Assim, a camada de aplicação é responsável pela gestão das perdas.

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.

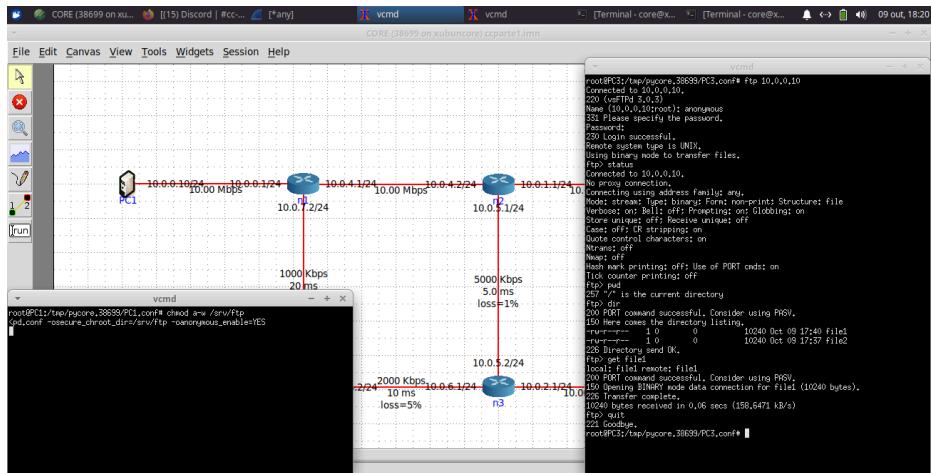
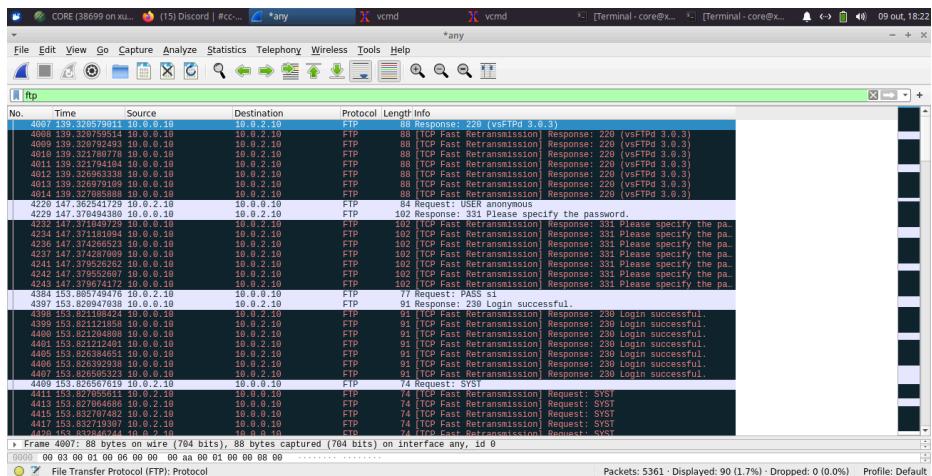


Fig. 11 - Output do terminal na utilização dos serviços



S

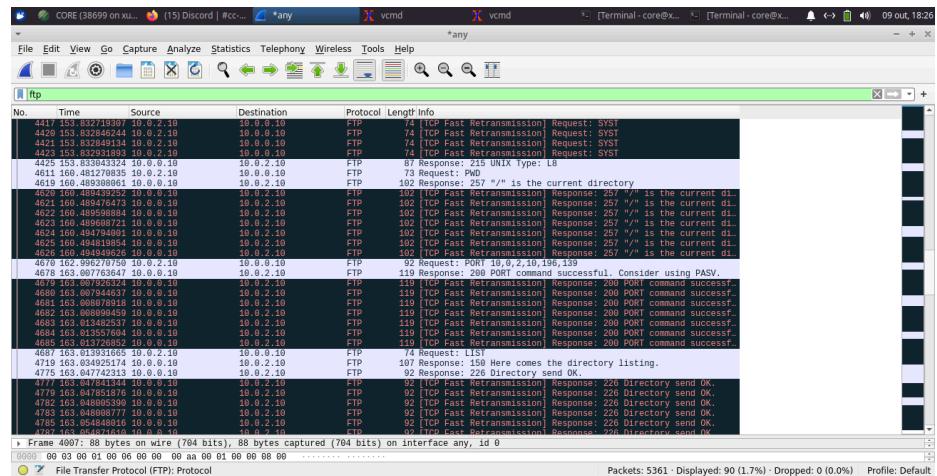


Fig.13 - Segunda fase de Conexão

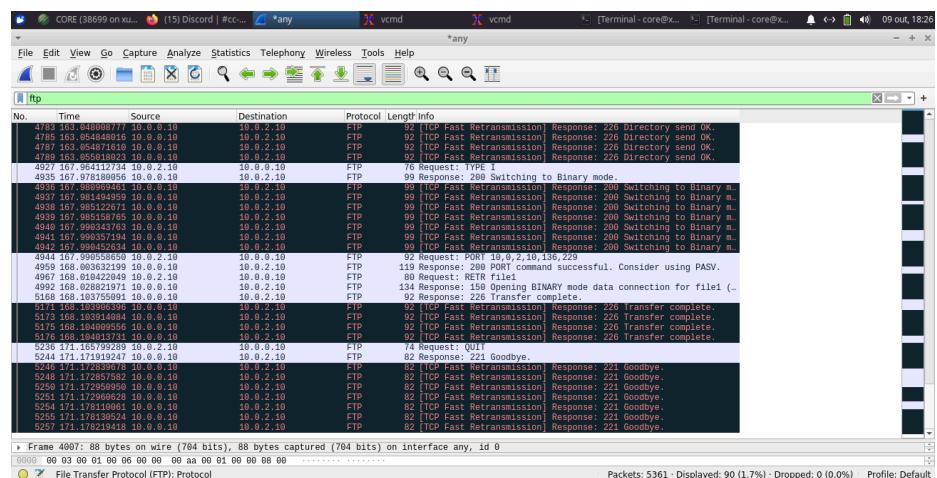


Fig.14 - Transferências de dados e término de conexão

Apartir da figura 12 e 13, respectivamente, é possível identificar que os pacotes apresentados na captura evidenciam tratam do início da conexão, autenticando quem usa o serviço através do user, enviado como PSH (request) para o servidor, e da pass, pedida como ACK para o PC1 e devolvida posteriormente como PSH para o servidor. É enviado um ACK para informar que a conexão foi aceite. É de ressaltar que esta captura apanha os pacotes da porta 21, sendo utilizada para conexão.

Depois do login ser feito, o PC1 envia PSH para o servidor para pedir o nome do Sistema Operativo que está a ser usado, que é respondido com ACK com UNIX. Seguidamente, o cliente envia um request para obter o output do comando PWD, sendo devolvido um ACK com a resposta.

A seguinte troca de request responses fazem parte da parte de conexão, no entanto, como utilizamos a filtragem pela porta da conexão os pacotes com a efetiva transferência de dados é feita pela porta 20, utilizando FTP-DATA. Apenas a seguinte captura apresenta os segmentos com transferência de dados efetivos.

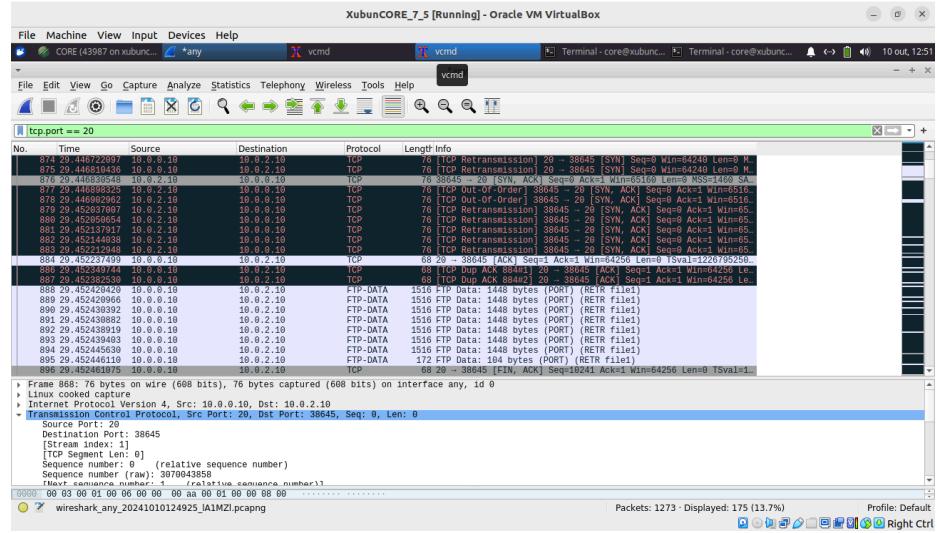


Fig. 15 - Transferência de dados por FTP

É possível confirmar que, filtrando os pacotes que utilizam a porta 20, obtém-se os pacotes com a transferência dos dados.

Também é possível perceber que entre as linhas 888 e 896, se dá a resposta do servidor ao cliente segundo algum request que tenha sido feito.

Ainda se pode verificar que há um término da conexão, na linha 896, requisitado pelo pacote da figura 14.

c) Apresente um diagrama temporal para a transferência de 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.

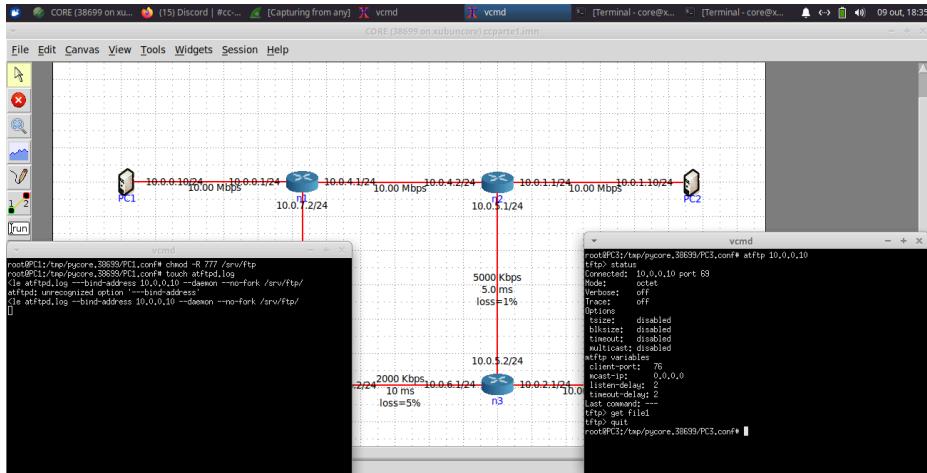


Fig. 16 - Output da conexão e request ao servidor

Para contexto, a figura 16 demonstra a conexão feita com o servidor e evidencia que se fez um request para download de um ficheiro.

Esta conexão é iniciada com o primeiro pacote enviado da figura 17.

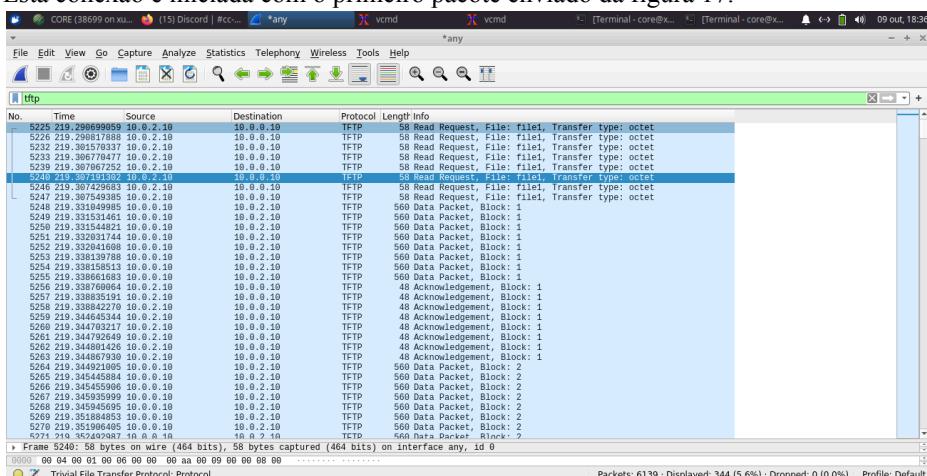


Fig. 17 - Pacotes do Request ao servidor e consequentes ACK

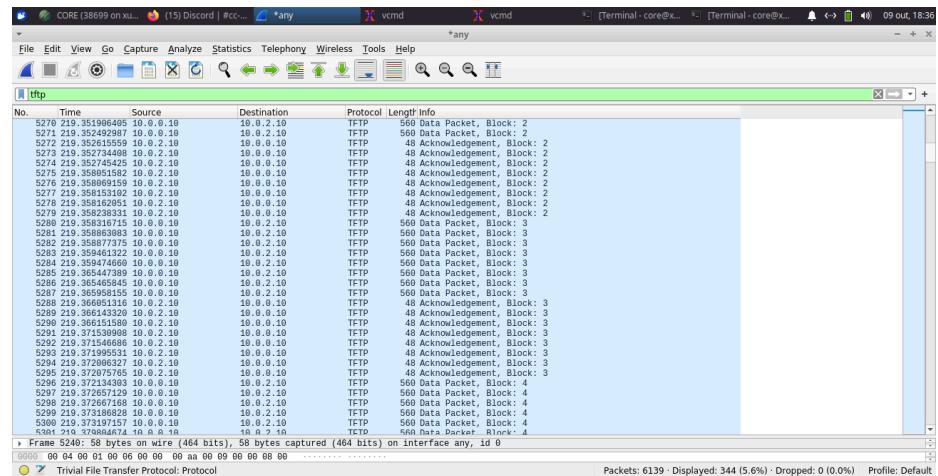


Fig. 18

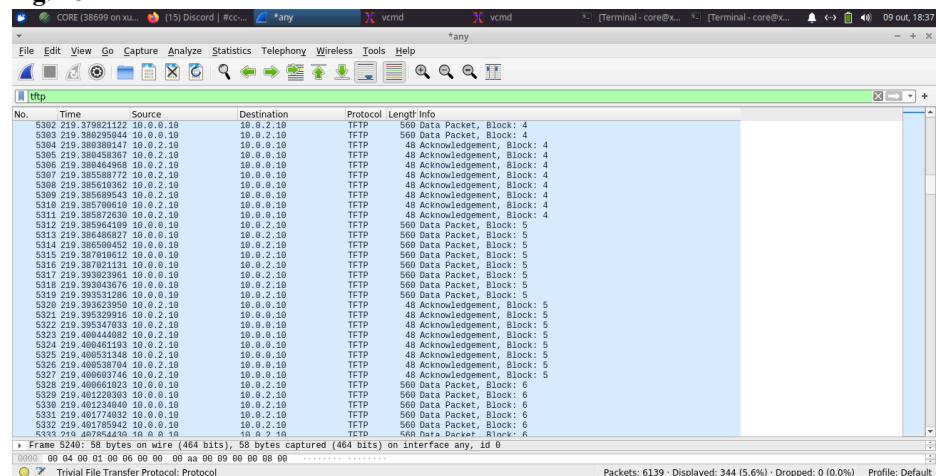


Fig. 19

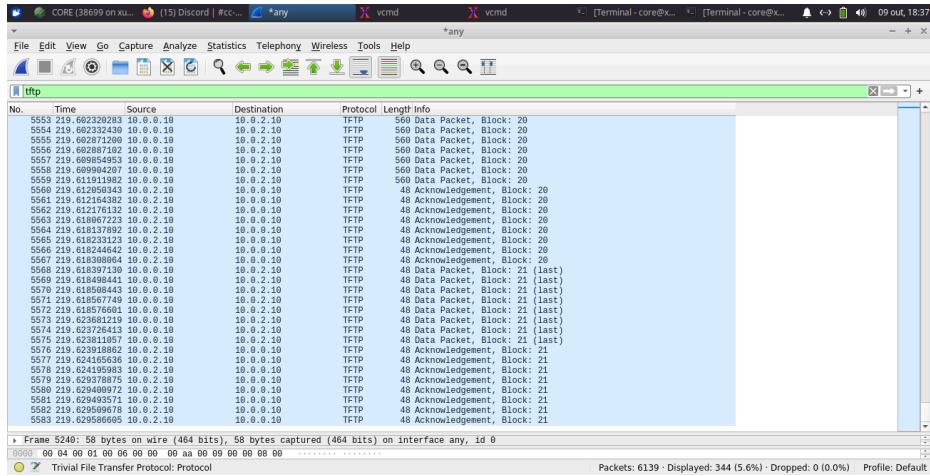


Fig. 20

Observando a totalidade da captura (figuras 17,18,19,20), verifica-se que a transferência de dados é realizada em blocos de pacotes, sendo enviados ao cliente também blocos de acknowledgements.

O fim da conexão é dado pelo bloco de pacotes com o comprimento igual ao dos pacotes de acknowledgement, indicando que o término da conexão (penúltimo bloco de pacotes enviados ao servidor).

3.2 Ao se descarregar os ficheiros a partir do PC2 e responda:

3.2.1 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 como ele é incrementado com cada pacote tanto pelo cliente quanto pelo servidor.

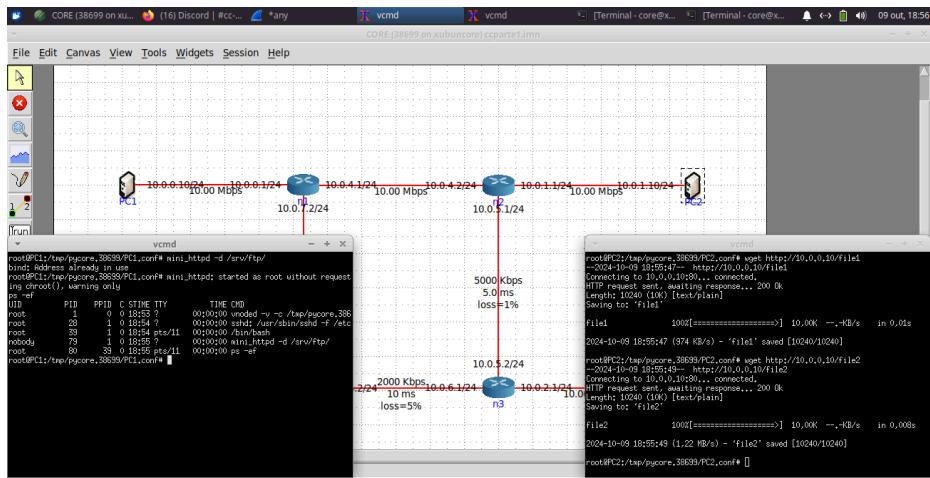


Fig. 21 - Pedido de transferência HTTP

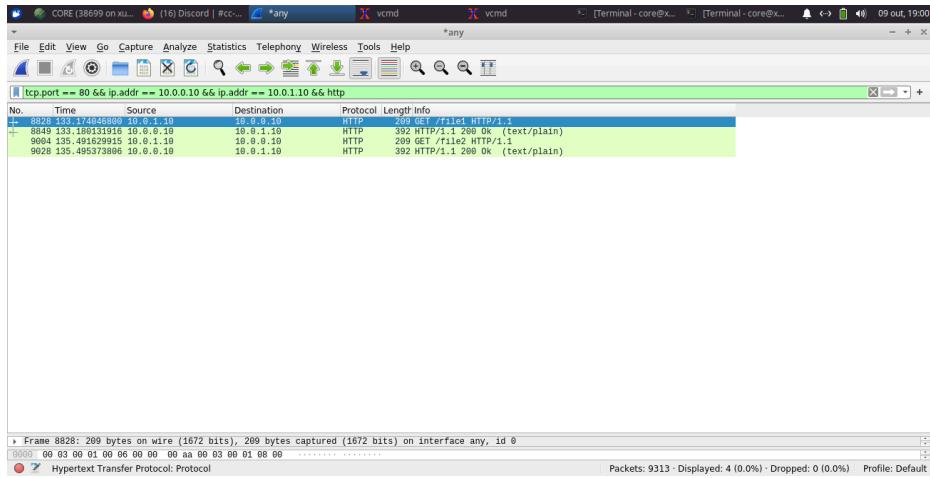


Fig.22 - Pacotes HTTP

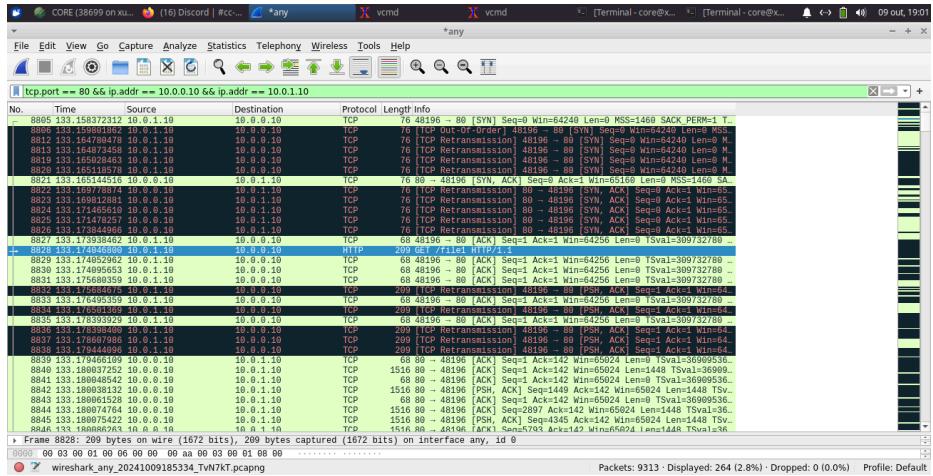


Fig. 23 - Captura do início da sessão TCP

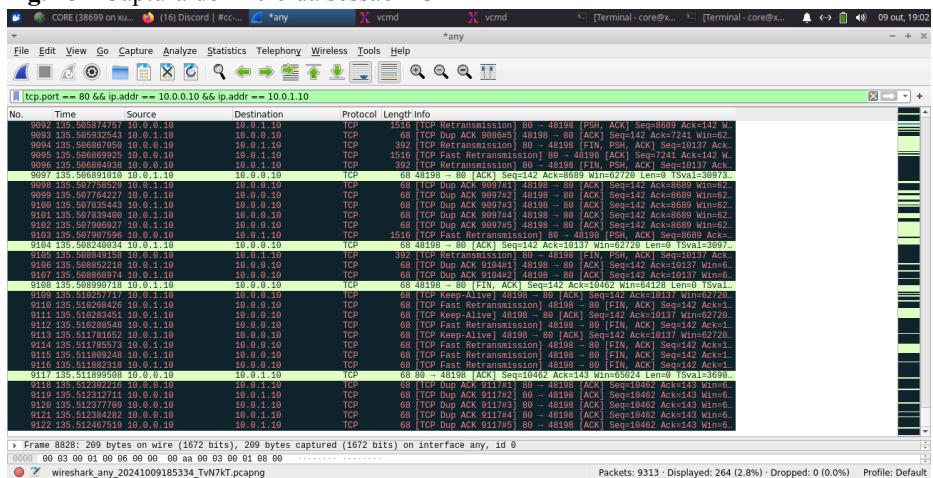


Fig. 24 - Captura com de um pacote com sucesso por parte do servidor

O início da sessão TCP é dado por 3 passos:

- O host envia um pacote para o servidor para solicitar a conexão;
 - O servidor devolve um pacote para confirmar a recepção do pacote do host e informar o número de sequência inicial do servidor;
 - O host envia um pacote de acknowledge de volta ao servidor, confirmando a recepção do pacote do mesmo. Nesse momento, a conexão TCP está estabelecida.

Já o fim da sessão é dado pelo envio do pacote FIN ACK do cliente ao servidor, terminando efetivamente com a resposta deste com um ACK a esse mesmo pacote.

Avaliando as figuras 23 e 24, estas mostram que o início da conexão com um pacote SYN enviado pelo cliente. Depois o servidor responde com um SYN ACK, respondido de volta por um ACK. Como vimos, fica a conexão estabelecida.

Ao longo dos restantes pacotes é possível verificar que existem muitos erros causados pela má qualidade das conexões, levando a muitos ACK, assim como os SEQ, que tem o mesmo número ao longo das capturas. Isto deve-se ao facto de essas variáveis serem utilizadas para determinar a presença de erros, sendo que o pedido das mesmas, mais de uma vez indica que falta o pacote com o byte presente no valor do ACK.

O término da conexão é dado pelos pacotes das linhas, 9108, 9117. Nestes pacotes o cliente envia o FIN ACK para o servidor, e o servidor responde com o ACK, encerrando a conexão.

Por existir muita perda/duplicação de pacotes, é possível perceber que o SEQ inicial é 0, sendo utilizado caso a ordem de chegada dos blocos não seja feita em ordem.

b) Qual o protocolo mais adequado para o PC2 obter o ficheiro? Justifique.

Para o PC2 obter o ficheiro, o melhor protocolo seria FTP ou HTTP. O débito direto de 10Mbps e a fiabilidade do TCP permitem uma troca de pacotes com entrega correta. A robustez da autenticação do FTP tornam-no numa melhor opção comparando com o TFTP, em caso de falha ocasional.

Por fim, terá mais vantagens com o uso do FTP, pois o ambiente controlado do CORE não indica vantagens ao uso do HTTP em prol do FTP.

3.3 – 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.**

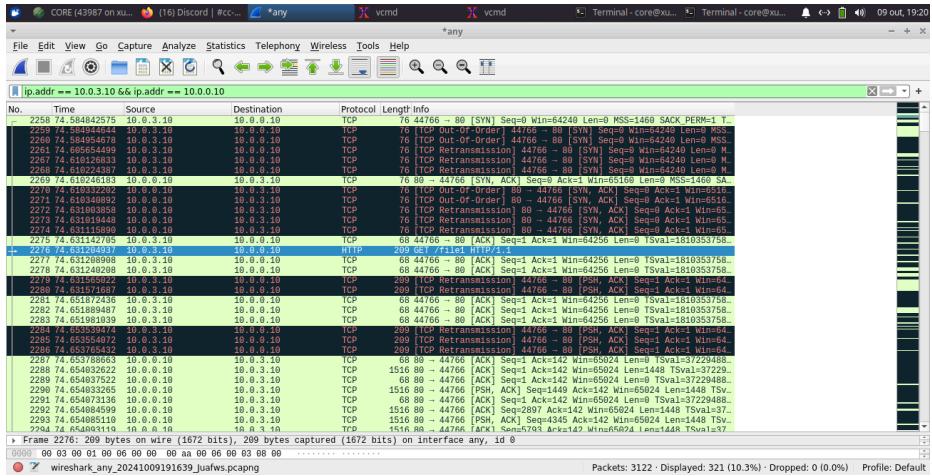


Fig. 25 - Início da conexão

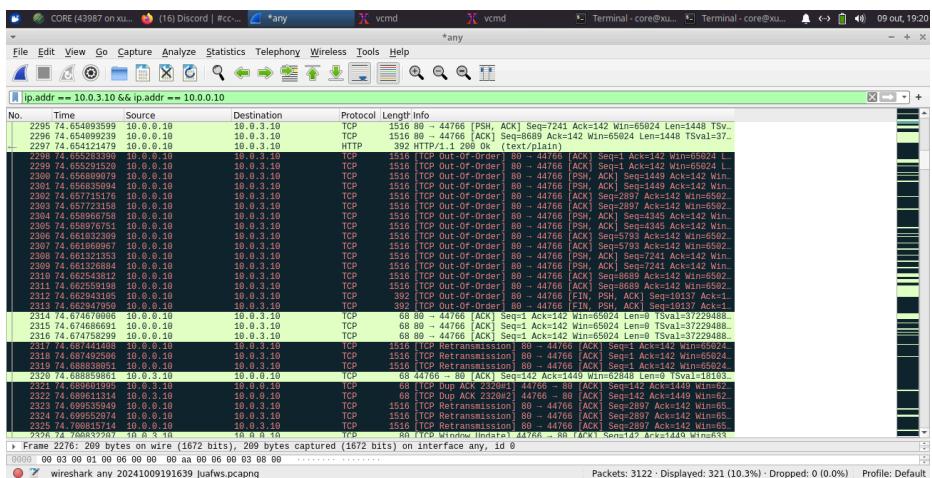
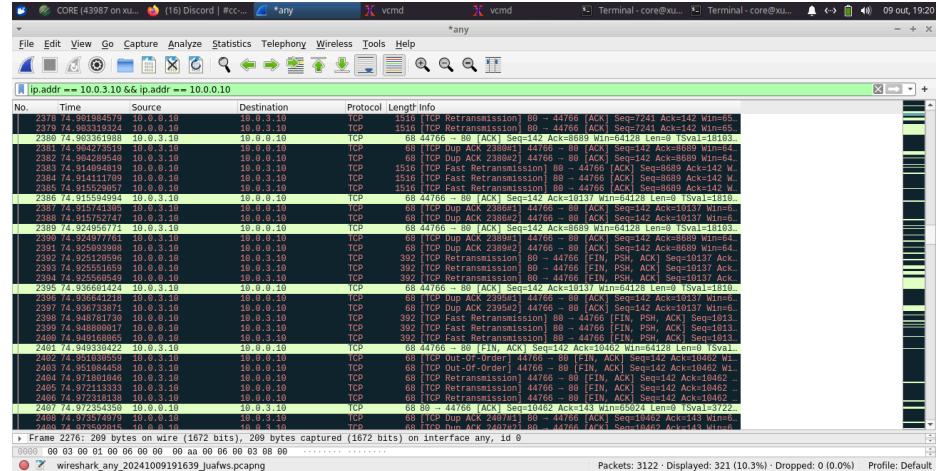


Fig. 26 - Transferência de dados

**Fig. 27 - Término da conexão**

A perda de pacotes aumenta a latência e reduz a eficiência da conexão devido às retransmissões necessárias. Já a duplicação de pacote impacta a largura de banda, e o processamento no envio de dados já recebidos.

É possível verificar a perda e a duplicação de pacotes no tráfego através das figuras anteriores, caracterizadas pelos pacotes que indicam [TCP Retransmission] e [TCP Dup ACK]. Estes são evidentes na figura 26, linhas 2317 e 2322, respectivamente.

Para lidar com estes problemas, o TCP implementa vários mecanismos. São exemplos a utilização de variáveis de controlo como ACK cumulativo, SEQ, retransmissão, timeout, e ainda implementa um controlo de congestionamento com uma *sliding window*.

O ACK cumulativo utiliza o número de sequência esperado. Se um pacote duplicado for recebido, o receptor simplesmente ignora o pacote duplicado, uma vez que já enviou o ACK para o pacote original.

O SEQ é um número único dado a cada byte transmitido, sendo que quando um pacote que tem um SEQ que já tenha sido recebido, o pacote é descartado.

A sliding window, determina quantos pacotes podem ser enviados antes de ser necessário um ACK do receptor. Se o emissor não receber um ACK dentro de um tempo pré-determinado (timeout), ele assume que o pacote foi perdido e retransmite-o.

b) Qual o protocolo mais adequado para o PC2 obter o ficheiro? Justifique.

Já de análises anteriores, inferimos que a utilização do TFTP é a que traz menos benefícios de todos os protocolos. Com isso, e através da análise de outras situações

semelhantes onde se compara o HTTP com FTP, determina-se que a utilização do protocolo FTP para transferências de ficheiros, onde o meio seja mais propenso a situações de perda ou duplicação, demonstra ser mais vantajoso e adequado.

3.4 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

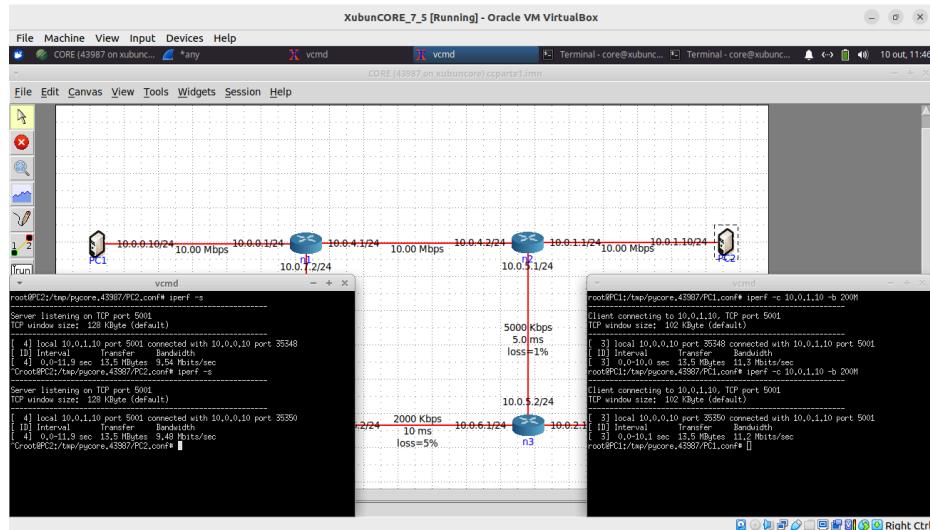


Fig. 28 - Uso do comando iperf entre o PC1 e PC2

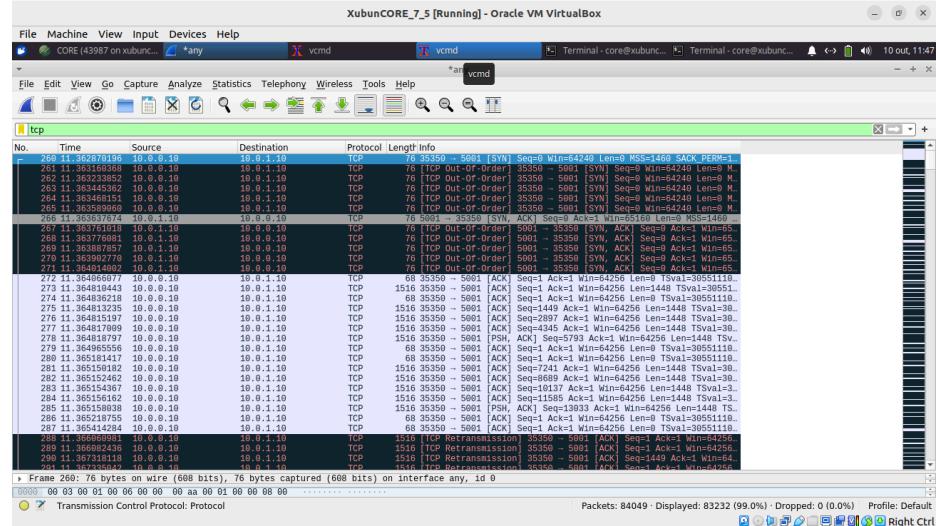


Fig. 29 - Análise do tráfego via Wireshark após uso iperf



Fig. 30 - Janela de congestão entre o PC1 e PC2 após uso do iperf

O congestionamento da rede foi simulado usando o iperf para gerar uma taxa de bits superior à largura de banda do canal. Na primeira imagem, vemos o iperf a ser executado com uma taxa de 200 Mbps, enquanto a largura de banda do canal entre o PC1 e o PC2 é de 10 Mbps. Isso cria uma situação de congestionamento, onde o tráfego gerado excede a capacidade do canal, levando a uma queda no desempenho.

O TCP utiliza mecanismos como o slow start e o congestion avoidance para ajustar o fluxo de dados. Quando uma conexão TCP é iniciada, ela começa com o slow start, onde a janela de congestionamento aumenta exponencialmente até atingir o limite. Após identificar perda de pacotes ou atraso, o TCP entra no modo congestion avoidance, onde a janela de congestionamento cresce de forma mais linear.

No evento de congestionamento, o TCP identifica a perda de pacotes (geralmente com base em timeouts ou recepção de ACKs duplicados). Quando isso acontece, o TCP assume que há congestionamento na rede e entra no Fast Recovery, onde a janela de congestão é reduzida para evitar mais perdas. O valor de ssthresh também é reduzido pela metade, o que significa que o TCP irá entrar em congestion avoidance mais cedo na próxima vez.

A Fig. 30 mostra a escala da janela do TCP (Window Scaling). Durante o slow start, a janela aumenta rapidamente, conforme mostrado no gráfico. Após certo ponto, a janela estabiliza-se e cresce de maneira mais moderada, representando o congestion avoidance.

$$\text{Fórmula Throughput} = \frac{\text{Tempo Total da Transferência}}{\text{Total de Dados Transferidos}}$$

A Fig. 29 mostra a análise do tráfego via Wireshark, onde vemos pacotes TCP com retransmissões e perdas (sinal de congestionamento). O throughput real pode ser calculado pela análise dos pacotes capturados. Na segunda execução do iperf, observamos uma transferência de 9,07 Mbps($13,5 * 8 / 11,9 = 9,07$), que está próxima à capacidade do canal (10 Mbps), demonstrando que o TCP ajusta o seu throughput para se adaptar à largura de banda disponível. Mesmo gerando uma carga de 200 Mbit/sec (bem acima dos 10 Mbps disponíveis), o TCP ajusta o fluxo de dados para a capacidade real do canal, o que demonstra o funcionamento dos mecanismos de controle de congestão.

As figuras de capturas de tráfego no Wireshark mostram claramente retransmissões TCP (indicadas pelas mensagens "Out-of-Order" e "Retransmission"), que confirmam o congestionamento. Isso também suporta a observação de como o TCP se ajusta conforme ocorre perda de pacotes.

Conclusão

Ao longo deste trabalho prático, foi possível explorar vários aspectos fundamentais da configuração de redes, o comportamento de diferentes protocolos de transporte e aplicação, e o impacto de condições adversas na comunicação. A partir das atividades desenvolvidas, destaco as seguintes aprendizagens:

- Configuração de Topologia de Rede e Análise de Tráfego

- Uso da Camada de Transporte pelas Aplicações
- Uso da Camada de Transporte pelas Aplicações
- Impacto das Perdas e Congestionamento
- Análise Comparativa dos Protocolos

Em resumo, o trabalho foi essencial para aprofundar o entendimento do comportamento dos protocolos de rede em diferentes condições. Além disso, a prática com captura de pacotes, análise de tráfego e simulações de congestionamento foram cruciais para reforçar conceitos teóricos e evidenciar na prática a importância de ajustar os parâmetros de rede conforme o ambiente e as necessidades de cada aplicação.