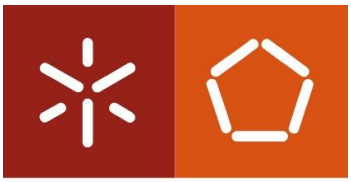


Universidade do Minho

Departamento de Informática

Redes de Computadores
Trabalho Prático 2 - Protocolo IPv4
(1ªParte)

Grupo Nº 18

Ariana Lousada (A87998)
Carlos Gomes (A77185)
Pedro Pereira (A80627)

22 de maio de 2021

Resumo

Para este projeto preparou-se uma topologia CORE para verificar o comportamento do traceroute. Para isso criámos um cenário com um Cliente1, que se ligou a um router R2; o router R2 a um router R3, que por sua vez, se ligou a um host (servidor) Servidor1.

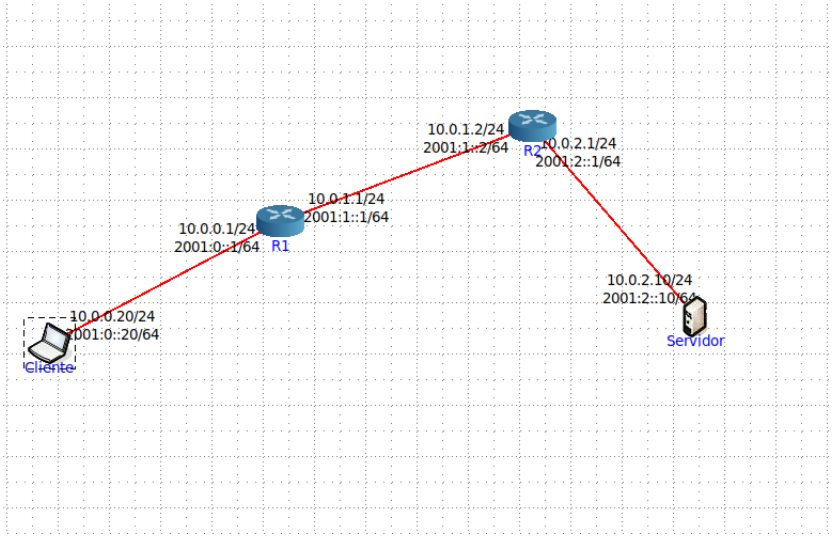


Figura 1: Cenário proposto

Conteúdo

Capítulo 1

Objetivos

Um dos principais objetivos deste trabalhos é o estudo do IP (Internet Protocol) e para isso iremos estudar as suas principais vertentes, como por exemplo, a fragmentação de pacotes IP, endereçamento IP, entre outros. Nesta primeira parte deste projeto foi realizado o registo de datagramas IP enviados e recebidos através da execução do programa *traceroute*. Com isto, analisámos os vários campos de um datagrama IP e detalhámos o processo de fragmentação realizado pelo IP.

Capítulo 2

Desenvolvimento

Para a resolução deste projeto, foram-nos propostas várias questões, as quais vamos passar a responder neste capítulo:

2.1 Captura de tráfego IP

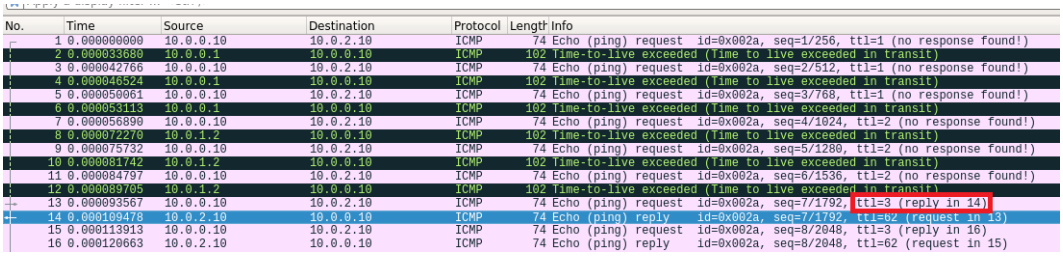
2.1.1 Exercício 1

1.a) **Active o wireshark ou o tcpdump no Cliente1. Numa shell do Cliente1, execute o comando traceroute -I para o endereço IP do Servidor1.**

```
traceroute to 10.0.2.10 (10.0.2.10), 30 hops max, 60 byte packets
 1  10.0.0.1 (10.0.0.1)  0.078 ms  0.023 ms  0.019 ms
 2  10.0.1.2 (10.0.1.2)  0.045 ms  0.031 ms  0.026 ms
 3  10.0.2.10 (10.0.2.10)  0.096 ms  0.036 ms  0.033 ms
root@Cliente:/tmp/pycore.40263/Cliente.conf# traceroute -I 10.0.2.10
traceroute to 10.0.2.10 (10.0.2.10), 30 hops max, 60 byte packets
 1  10.0.0.1 (10.0.0.1)  0.071 ms  0.022 ms  0.016 ms
 2  10.0.1.2 (10.0.1.2)  0.035 ms  0.024 ms  0.022 ms
 3  10.0.2.10 (10.0.2.10)  0.040 ms  0.032 ms  0.028 ms
root@Cliente:/tmp/pycore.40263/Cliente.conf# traceroute -I 10.0.2.10
traceroute to 10.0.2.10 (10.0.2.10), 30 hops max, 60 byte packets
 1  10.0.0.1 (10.0.0.1)  0.072 ms  0.022 ms  0.017 ms
 2  10.0.1.2 (10.0.1.2)  0.050 ms  0.025 ms  0.023 ms
 3  10.0.2.10 (10.0.2.10)  0.040 ms  0.030 ms  0.029 ms
root@Cliente:/tmp/pycore.40263/Cliente.conf#
```

Figura 2.1: Comando traceroute -I

1.b) **Registe e analise o tráfego ICMP enviado pelo Cliente1 e o tráfego ICMP recebido como resposta. Comente os resultados face ao comportamento esperado.**



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x002a, seq=1/256, ttl=1 (no response found!)
2	0.000033680	10.0.0.1	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
3	0.000042766	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x002a, seq=2/512, ttl=1 (no response found!)
4	0.000046524	10.0.0.1	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
5	0.000050061	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x002a, seq=3/768, ttl=1 (no response found!)
6	0.000053113	10.0.0.1	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
7	0.000056890	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x002a, seq=4/1024, ttl=2 (no response found!)
8	0.000072270	10.0.1.2	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
9	0.000075732	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x002a, seq=5/1280, ttl=2 (no response found!)
10	0.000081742	10.0.1.2	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
11	0.000084797	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x002a, seq=6/1536, ttl=2 (no response found!)
12	0.000089705	10.0.1.2	10.0.0.10	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
13	0.000093567	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x002a, seq=7/1792, ttl=3 (reply in 14)
14	0.000109478	10.0.2.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x002a, seq=7/1792, ttl=62 (request in 13)
15	0.000113913	10.0.0.10	10.0.2.10	ICMP	74	Echo (ping) request id=0x002a, seq=8/2048, ttl=3 (reply in 16)
16	0.000120663	10.0.2.10	10.0.0.10	ICMP	74	Echo (ping) reply id=0x002a, seq=8/2048, ttl=62 (request in 15)

Figura 2.2: TTL

Ao analisarmos os resultados obtidos, podemos constatar que o envio de pacotes teve duas fases: i) pacotes com TTL abaixo de 3 e ii) pacotes com TTL acima de 3.

Na primeira fase, podemos observar que os pacotes com TTL=1 e TTL=2 foram descartados pelos routers e para cada um destes foi recebido um pacote Time-to-live exceed. Já na segunda fase, nenhum deles foi descartado tendo como resposta pacotes Echo(ping) reply.

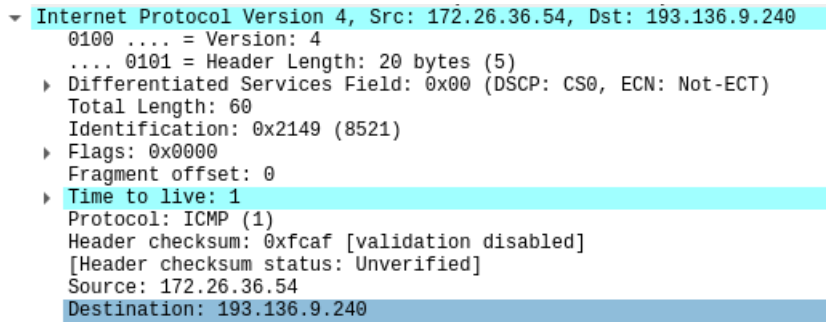
1.c) **Qual deve ser o valor inicial mínimo do campo TTL para alcançar o Servidor1? Verifique na prática que a sua resposta está correta.**

Na elaboração deste projeto, obtivemos um valor mínimo do campo TTL equivalente a 3.

1.d) **Calcule o valor médio do tempo de ida-e-volta (Round-TripTime) obtido.**

2.1.2 Exercício 2

2.a) **Qual é o endereço IP da interface ativa do seu computador?**



```
Internet Protocol Version 4, Src: 172.26.36.54, Dst: 193.136.9.240
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0x2149 (8521)
  Flags: 0x0000
    Fragment offset: 0
  Time to live: 1
    Protocol: ICMP (1)
    Header checksum: 0xfcaf [validation disabled]
    [Header checksum status: Unverified]
    Source: 172.26.36.54
    Destination: 193.136.9.240
```

Figura 2.3: Cabeçalho IP

Como se pode observar na imagem, a máquina utilizada para teste possui o endereço IP de 172.26.36.54.

2.b) **Qual é o valor do campo protocolo? O que identifica?**

Verificámos que o valor equivale a 1, que corresponde ao ICMP.

2.c) **Quantos bytes tem o cabeçalho IP(v4)? Quantos bytes tem o campo de dados (payload) do datagrama? Como se calcula o tamanho do payload?**

Ao analisarmos o campo Header length na figura 2.3, podemos concluir que o cabeçalho IP(v4) tem 20 bytes (5). O payload do datagrama corresponde a 40. O tamanho do payload pode ser obtido através da diferença entre o número de bytes e o tamanho do cabeçalho do datagrama (*Total length - header length*).

2.d) **O datagrama IP foi fragmentado? Justifique.**

Não, uma vez que ambos o *fragment offset* e o *flag more fragments* são 0 (na primeira mensagem capturada).

2.e) **Ordene os pacotes capturados de acordo com o endereço IP fonte (e.g., selecionando o cabeçalho da coluna Source), e analise a sequência de tráfego ICMP gerado a partir do endereço IP atribuído à interface da sua máquina. Para a sequência de mensagens ICMP enviadas pelo seu computador, indique que campos do cabeçalho IP variam de pacote para pacote.**

Os únicos campos do cabeçalho que vão variando são a *Identificação*, *time-to-live* e a *header checksum*.

```
  5 0.016977368   172.26.36.54        193.136.9.240       ICMP       74 Echo (ping) request  id=0x78bc, seq=1/256, ttl=1 (no response found!)
  6 0.017000712   172.26.36.54        193.136.9.240       ICMP       74 Echo (ping) request  id=0x78bc, seq=2/512, ttl=1 (no response found!)
  7 0.017011963   172.26.36.54        193.136.9.240       ICMP       74 Echo (ping) request  id=0x78bc, seq=3/768, ttl=1 (no response found!)
  8 0.017024492   172.26.36.54        193.136.9.240       ICMP       74 Echo (ping) request  id=0x78bc, seq=4/1024, ttl=2 (no response found!)
  9 0.017072692   172.26.36.54        193.136.9.240       ICMP       74 Echo (ping) request  id=0x78bc, seq=5/1280, ttl=2 (no response found!)
 10 0.017116761   172.26.36.54        193.136.9.240       ICMP       74 Echo (ping) request  id=0x78bc, seq=6/1536, ttl=2 (no response found!)
 11 0.017159265   172.26.36.54        193.136.9.240       ICMP       74 Echo (ping) request  id=0x78bc, seq=7/1792, ttl=3 (no response found!)
 12 0.017200262   172.26.36.54        193.136.9.240       ICMP       74 Echo (ping) request  id=0x78bc, seq=8/2048, ttl=3 (no response found!)
 13 0.017208326   172.26.36.54        193.136.9.240       ICMP       74 Echo (ping) request  id=0x78bc, seq=9/2304, ttl=3 (no response found!)
 14 0.017251794   172.26.36.54        193.136.9.240       ICMP       74 Echo (ping) request  id=0x78bc, seq=10/2560, ttl=4 (reply in 29)
 15 0.017259945   172.26.36.54        193.136.9.240       ICMP       74 Echo (ping) request  id=0x78bc, seq=11/2816, ttl=4 (reply in 30)
 16 0.017297269   172.26.36.54        193.136.9.240       ICMP       74 Echo (ping) request  id=0x78bc, seq=12/3072, ttl=4 (reply in 31)
 17 0.017306270   172.26.36.54        193.136.9.240       ICMP       74 Echo (ping) request  id=0x78bc, seq=13/3328, ttl=5 (reply in 32)
 18 0.017340757   172.26.36.54        193.136.9.240       ICMP       74 Echo (ping) request  id=0x78bc, seq=14/3584, ttl=5 (reply in 34)
 19 0.017348816   172.26.36.54        193.136.9.240       ICMP       74 Echo (ping) request  id=0x78bc, seq=15/3840, ttl=5 (reply in 35)
 20 0.017386285   172.26.36.54        193.136.9.240       ICMP       74 Echo (ping) request  id=0x78bc, seq=16/4096, ttl=6 (reply in 36)
 22 0.019796688   172.26.36.54        193.136.9.240       ICMP       74 Echo (ping) request  id=0x78bc, seq=17/4352, ttl=6 (reply in 39)
```

Figura 2.4: Pacotes capturados

2.f) **Observa algum padrão nos valores do campo de Identificação do datagrama IP e TTL?**

O ID incrementa por 1 de pacote em pacote, enquanto que o TTL incrementa por 1 de 3 em 3 pacotes.

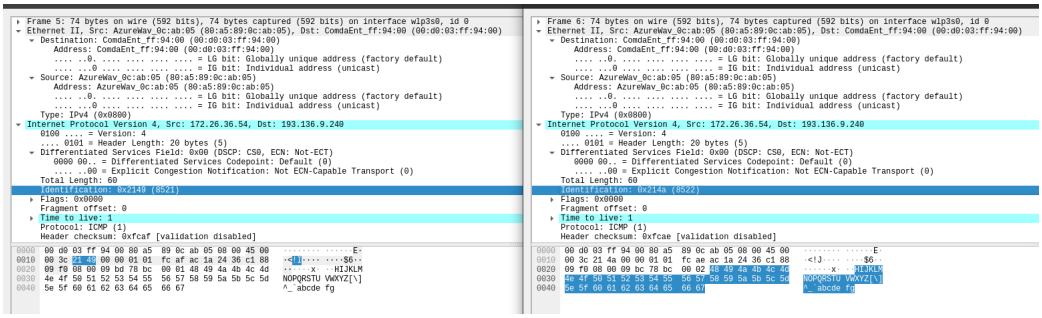


Figura 2.5

2.g) **Ordene o tráfego capturado por endereço destino e encontre a série de respostas ICMP TTL exceeded enviadas ao seu computador. Qual é o valor do campo TTL? Esse valor permanece constante para todas as mensagens de resposta ICMP TTL exceeded enviados ao seu host? Porquê?**

O ttl é de 1 e permance constante. O TTL é decrementado em cada salto e quando é 1 e não está no destino (sendo decrementado passaria a 0) é enviado para trás.

2.1.3 Exercício 3

3.a) **Localize a primeira mensagem ICMP. Porque é que houve necessidade de fragmentar o pacote inicial?**

Foi necessário fragmentar o pacote inicial, uma vez que o limite da camada IP é de 1500 enquanto que o tamanho do pacote enviado é de 3218.

3.b) **Imprima o primeiro fragmento do datagrama IP segmentado. Que informação no cabeçalho indica que o datagrama foi fragmentado? Que informação no cabeçalho IP indica que se trata do primeiro fragmento? Qual é o tamanho deste datagrama IP?**

A *flag more fragments* a 1 indica que foi fragmentado. O *fragment offset* a 0 indica que se trata do primeiro fragmento. Com isto, podemos portanto dizer que tamanho do datagrama IP é de 1500 bytes.

3.c) **Imprima o segundo fragmento do datagrama IP original. Que informação do cabeçalho IP indica que não se trata do 1º fragmento? Há mais fragmentos? O que nos permite afirmar isso?**

O *fragment offset* a 1480 indica que não se trata do primeiro fragmento. Há mais fragmentos pois a *flag more fragments* está a 1.

3.d) **Quantos fragmentos foram criados a partir do datagrama original? Como se detecta o último fragmento correspondente ao datagrama original?**

Foram criados 3 fragmentos a partir do datagrama original. O último fragmento é detetável pela *flag more fragments*, uma vez que está a 0.

3.e) **Indique, resumindo, os campos que mudam no cabeçalho IP entre os diferentes fragmentos, e explique a forma como essa informação permite reconstruir o datagrama original.**

Os campos que mudam são a *total length*, o *fragment offset* e as *flags* (e consequentemente a *header checksum*). Uma vez que os fragmentos têm a mesma identificação e sabendo o offset de cada um, é possível reconstruir o datagrama original.