Hands-on: Análise de Tráfego em Redes TCP/IP

WTR 2020 do PoP-BA/RNP

Instrutores

- Adriana Viriato
- Gildásio Júnior

Pré-requisitos

- Ferramentas para uso:
  - `tcpdump / windump / wireshark`
  - `netcat`
  - `traceroute / mtr`
  - `hping3`
  - `nmap`

Como alternativa, pode fazer download da máquina virtual disponibilizada no curso:

- https://filesender.rnp.br/?s=download&token=aed60b3f-e934-42c4-af63-32420f8e13b0

Prazo para download: **18/10/2020**

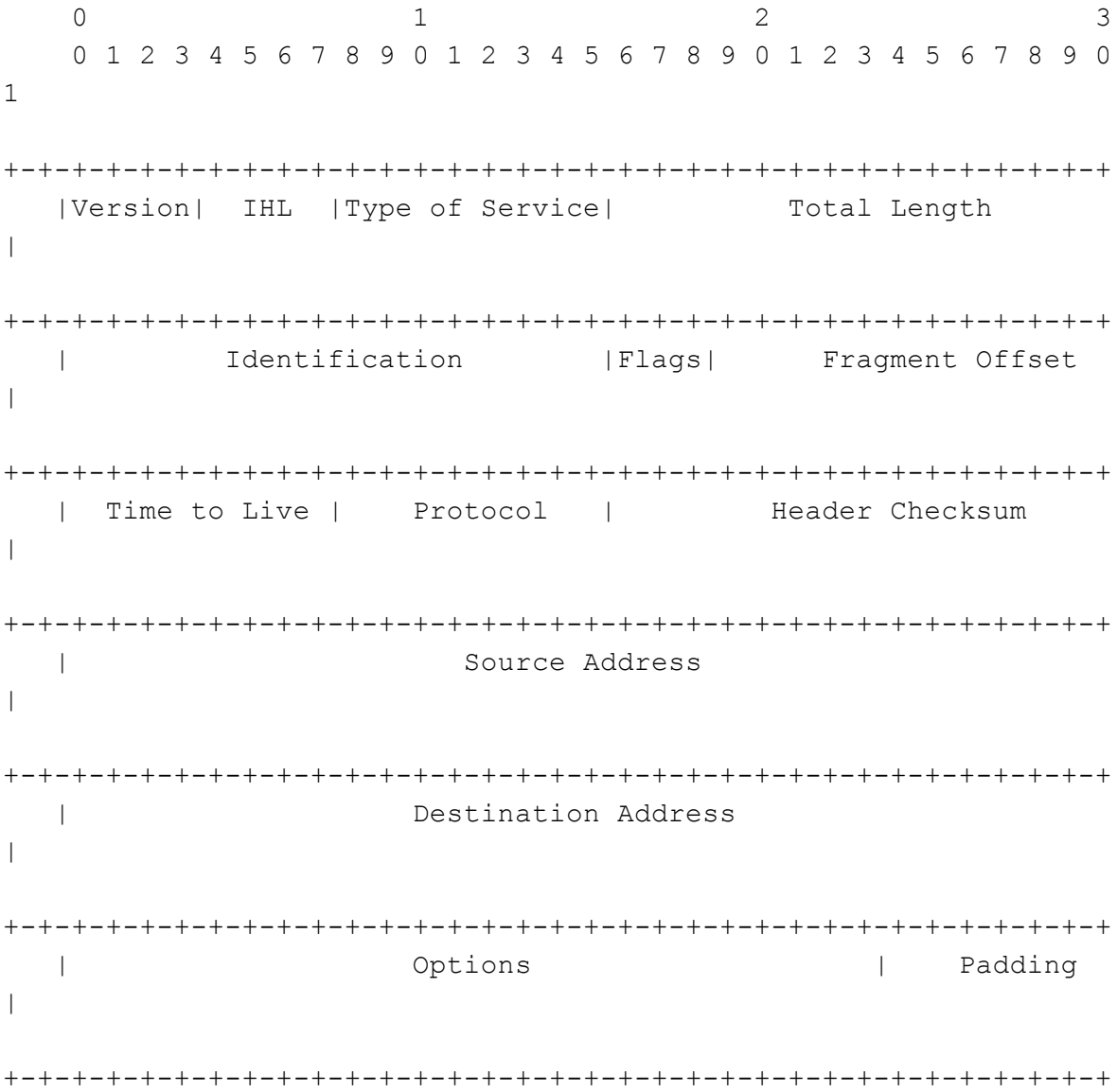
Gravações dos encontros:

Os encontros foram gravados e estão disponíveis nos links abaixo:

- Dia 1: https://video.rnp.br/portal/video/wtr2020-popba-curso-analise-trafego-pt1
- Dia 2: https://video.rnp.br/portal/video/wtr2020-popba-curso-analise-trafego-pt2

1º dia: IP | ICMP

IPv4 / RFC 791



Normalmente o local que os campos estão indicam sua motivação de uso:

- 1ª linha: como tratar os dados
- 2ª linha: identificação do pacote
- 3ª linha: controles
- 4ª / 5ª linhas: endereçamento
- 6ª / nª linhas: extras

Descrição dos campos:

- Version: poderia ter até 16 valores (0 a 15, em decimal), mas é usado 4 ou 6, referenciando o IPv4 e o IPv6, respectivamente.
- IHL (Internet Header Length): de 0 a 15 linhas (4 bits, 2^4). As primeiras cinco linhas são obrigatórias. Por isso o tamanho do cabeçalho do protocolo pode variar de 20 bytes (5 linhas de 4 bytes cada uma) a 60 bytes (15 linhas de 4 bytes cada uma).
- ToS ou DiffServ: é um campo pouco utilizado, pois pode causar atrasos, por conta da necessidade de processamento extra. Além disso, é ignorado pela maioria dos equipamentos.
- Total Length: o tamanho máximo do IPv4 é de 65535 bytes (equivale a 16 bits 1). O tamanho mínimo é de 21 bytes: 20 de cabeçalho + 1 payload. O IP serve para carregar algo e 20 bytes é o tamanho mínimo do cabeçalho, portanto, 20 não pode ser o tamanho mínimo do pacote. Obs¹: Utilizar o `tcpdump` com o parâmetro `-X` auxilia na análise desse campo. Obs²: Comumente, a sua placa de rede vai conseguir transmitir 1500 bytes (MTU para Ethernet e Wi-Fi). Então o pacote será fragmentado e todos os pedaços tem que ter cabeçalho.
- Identification: identifica a ordem dos pacotes em caso de fragmentação, o valor pode variar de 0 a 65535 bytes (equivale a 16 bits 1).
- Flags:
    - 0: guardado para uso futuro
    - DF (Don't fragment):
        - 0: pode fragmentar
        - 1: não pode fragmentar
    - MF (More fragments):
        - 0: não tem mais fragmentos
        - 1: tem mais fragmentos
- Fragment Offset: medido em unidades de 13 bits. Representa a quantidade de dados já enviados nos fragmentos anteriores.
- Time to live: varia de 0 a 255 e a cada salto na rede, o campo TTL é decrementado em 1. Como dificilmente se usa mais de 30 saltos para chegar a um destino na Internet, alguns sistemas operacionais utilizam valores inferiores a 255 ao criar um pacote. Desta forma, ao avaliar o valor do campo TTL, é possível ter indício do sistema operacional de um host:
    - Unix = 255
    - Windows atuais = 128
    - Linux = 64 (esse valor pode ser observado acessando o arquivo virtual `/proc/sys/net/ipv4_ip_default_ttl`)
- Protocol: o IP serve para carregar outros dados, então ele encapsula protocolos. Esse campo pode ter um valor variando de 0 a 255 (8 bits). A IANA é a instituição responsável por fazer a atribuição entre número e protocolo. No Linux, por exemplo, é possível verificar esses valores no arquivo `/etc/protocols`.
- Header checksum: cálculo realizado para verificação da integridade do cabeçalho. Como esse cálculo inclui o TTL, que é alterado a cada salto, isso significa que a cada salto é preciso realizar um novo cálculo para geração do *Header checksum*, além do cálculo para verificação de integridade. Se fosse incluído o payload, a quantidade de dados para os equipamentos processarem seria maior.
- Source IP Address: endereço IP de origem do pacote.

- Destination IP Address: endereço IP de destino do pacote.
- Options: campos opcionais para extensão do IP padrão.
- Padding: usado para garantir que o cabeçalho IP e os dados encapsulados estejam alinhados a 32 bits, composto de zeros. O padding é necessário quando os campos opcionais são utilizados e não alcançam o tamanho de 32 bits.

Análise de Tráfego IPv4

- Além de visualizar o cabeçalho do IP, é interessante também entender o uso da ferramenta, verificando a utilização de filtros e parâmetros para melhor visualização do tráfego. Para isso, veja a gravação das aulas disponíveis. Nas seções de análise, lembre-se de fazer as mudanças necessárias nos IPs e nas interfaces.

```
sudo tcpdump -i eno1 -n host 10.1.0.38

nc 10.1.0.38 80
```

```
11:13:48.627195 IP 10.1.0.114.36520 > 10.1.0.38.80: Flags [S], seq
3424060001, win 64240, options [mss 1460,sackOK,TS val 500434377 ecr
0,nop,wscale 7], length 0
```

- Quando o tcpdump utiliza “IP” ele está se referindo à versão 4. Ao utilizar IPv6, ele sinaliza com “IP6”.

```
sudo tcpdump -i eno1 -n host 10.1.0.38 or host fe80::c4aa:2dff:fe74:4dc3

ping -c 1 10.1.0.38
ping -c 1 fe80::c4aa:2dff:fe74:4dc3
```

```
11:27:09.227398 IP 10.1.0.114 > 10.1.0.38: ICMP echo request, id 31795,
seq 1, length 64
11:27:09.253030 IP 10.1.0.38 > 10.1.0.114: ICMP echo reply, id 31795,
seq 1, length 64

11:27:11.680654          IP6          fe80::87a3:4fca:e840:9748          >
fe80::c4aa:2dff:fe74:4dc3: ICMP6, echo request, seq 1, length 64
11:27:11.713113          IP6          fe80::c4aa:2dff:fe74:4dc3          >
fe80::87a3:4fca:e840:9748: ICMP6, echo reply, seq 1, length 64
```

- O TTL 64, indica o sistema operacional do host. Veja que está com a flag DF ativada e que o protocolo utilizado é o TCP, com o tamanho do pacote de 60 bytes. Note que ainda não mostra o tamanho do cabeçalho (campo IHL).

```
sudo tcpdump -i eno1 -n host 10.1.0.38 -vv

nc 10.1.0.38 80
```

11:14:13.406988 **IP** (**tos** 0x0, **ttl** 64, **id** 16480, **offset** 0, **flags** [DF], **proto** TCP (6), **length** 60) **10.1.0.114**.36594 > **10.1.0.38**.80: Flags [S], cksum 0x14c8 (incorrect -> 0x2cd9), seq 3136765178, win 64240, options [mss 1460,sackOK,TS val 500459156 ecr 0,nop,wscale 7], length 0

- Veja que embora o tamanho máximo de um pacote IP possa alcançar o valor de 65535, a camada de enlace comumente tem uma limitação de 1500 bytes. Com isso, o pacote precisa ser fragmentado. Perceba as flags [+], o offset, e que o campo identification não muda. Podemos testar essa questão da fragmentação utilizando a flag de não fragmentação com o comando ping.

```
sudo tcpdump -i eno1 -n -vvv icmp

ping -c 1 -s 4000 10.1.0.38
```

11:32:15.798311 IP (tos 0x0, ttl 64, **id 5432, offset 0, flags [+]**, proto ICMP (1), length **1500**)
    10.1.0.114 > 10.1.0.38: ICMP echo request, id 32010, seq 1, length **1480**

11:32:15.798321 IP (tos 0x0, ttl 64, **id 5432, offset 1480, flags [+]**, proto ICMP (1), length **1500**)
    10.1.0.114 > 10.1.0.38: ip-proto-1

11:32:15.798325 IP (tos 0x0, ttl 64, **id 5432, offset 2960, flags [none]**, proto ICMP (1), length **1068**)
    10.1.0.114 > 10.1.0.38: ip-proto-1

```
sudo tcpdump -i eno1 -n -v host 10.1.0.38

ping -c 1 -M do 10.1.0.38 -s 1472
```

11:43:05.266393 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ICMP (1), length 1500)
    10.1.0.114 > 10.1.0.38: ICMP echo request, id 32371, seq 1, length 1480
11:43:05.287230 IP (tos 0x0, ttl 64, id 42271, offset 0, flags [none],

```
proto ICMP (1), length 1500)
     10.1.0.38 > 10.1.0.114: ICMP echo reply, id 32371, seq 1, length
1480
```

```
ping -M do 10.1.0.38 -s 4000 -c 1
```

```
PING 10.1.0.38 (10.1.0.38) 4000(4028) bytes of data.
ping: local error: Message too long, mtu=1500

--- 10.1.0.38 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss,
time 0ms
```

- Sobre os protocolos IP, a referência para o mapeamento número/nome está disponível no site da IANA e pode também ser consultado no Linux no arquivo `/etc/protocols`:
  - http://www.iana.org/assignments/protocol-numbers

```
$ less /etc/protocols
$ getent protocols tcp
tcp             6 TCP
$ getent protocols udp
udp             17 UDP
$ getent protocols icmp
icmp            1 ICMP
```

```
sudo tcpdump -i eno1 -n -v ip dst 10.1.0.38 and not arp
nc 10.1.0.38 80
nc -u 10.1.0.38 23
ping -c 1 10.1.0.38
```

```
15:27:51.020143 IP (tos 0x0, ttl 64, id 33997, offset 0, flags [DF], proto
UDP (17), length 34)
    10.1.0.114.36278 > 10.1.0.38.23: UDP, length 6
15:28:09.906333 IP (tos 0x0, ttl 64, id 43653, offset 0, flags [DF], proto
ICMP (1), length 84)
    10.1.0.114 > 10.1.0.38: ICMP echo request, id 1788, seq 1, length 64
15:28:18.224347 IP (tos 0x0, ttl 64, id 27312, offset 0, flags [DF], proto
TCP (6), length 60)
    10.1.0.114.41278 > 10.1.0.38.80: Flags [S], cksum 0x14c8 (incorrect ->
0x225a), seq 4121385857, win 64240, options [mss 1460,sackOK,TS val
515703975 ecr 0,nop,wscale 7], length 0
```

- Veja em hexadecimal:
    - Versão: **4**500
    - IHL: 4**5**00 (= 5 linhas = 20 bytes)
    - ToS: 45**00** (não está sendo usado. Valor: 0)
    - Total Length: **0054** (84 bytes)
    - Identification: **ce48** (52808)
    - Flags (3 bits): **4**000 -> **010**0 **(binário)**
        - 0: **0**10
        - DF: 0**1**0
        - MF: 01**0**
    - Fragment offset: 0 (binário) 000 (hexadecimal) = 0
    - Time to Live: **40**01 (64)
    - Protocol: 40**01** (1 = ICMP)
    - Header checksum: **57c7** (22471)
    - Source Address: **0a01 0072** (10.1.0.114)
    - Destination Address: **0a01 0026** (10.1.0.38)

```
sudo tcpdump -i eno1 -n -vvv -X icmp
ping -c 1 10.1.0.38
```

```
11:30:10.889735 IP (tos 0x0, ttl 64, id 52808, offset 0, flags [DF], proto ICMP (1), length 84)
    10.1.0.114 > 10.1.0.38: ICMP echo request, id 31927, seq 1, length 64
        0x0000:  4500 0054 ce48 4000 4001 57c7 0a01 0072  E..T.H@.@.W....r
        0x0010:  0a01 0026 0800 bbec 7cb7 0001 7295 575f  ...&....|...r.W_
        0x0020:  0000 0000 2993 0d00 0000 0000 1011 1213  ....).........
        0x0030:  1415 1617 1819 1a1b 1c1d 1e1f 2021 2223  .............!"#
        0x0040:  2425 2627 2829 2a2b 2c2d 2e2f 3031 3233  $%&'()*+,-./0123
        0x0050:  3435 3637                                4567
```

IPv6 / RFC 8200

Vamos fazer uma analogia entre campos do IPv4 e do IPv6:

Campos comuns:
- Version
- Source Address
- Destination Address

Campos renomeados:
- Traffic Class -> Type of Service ou DiffServ
- Payload Length -> Total Length
- Next Header -> Protocol
- Hop Limit -> Time To Live

Campos diferentes:
- Flow Label

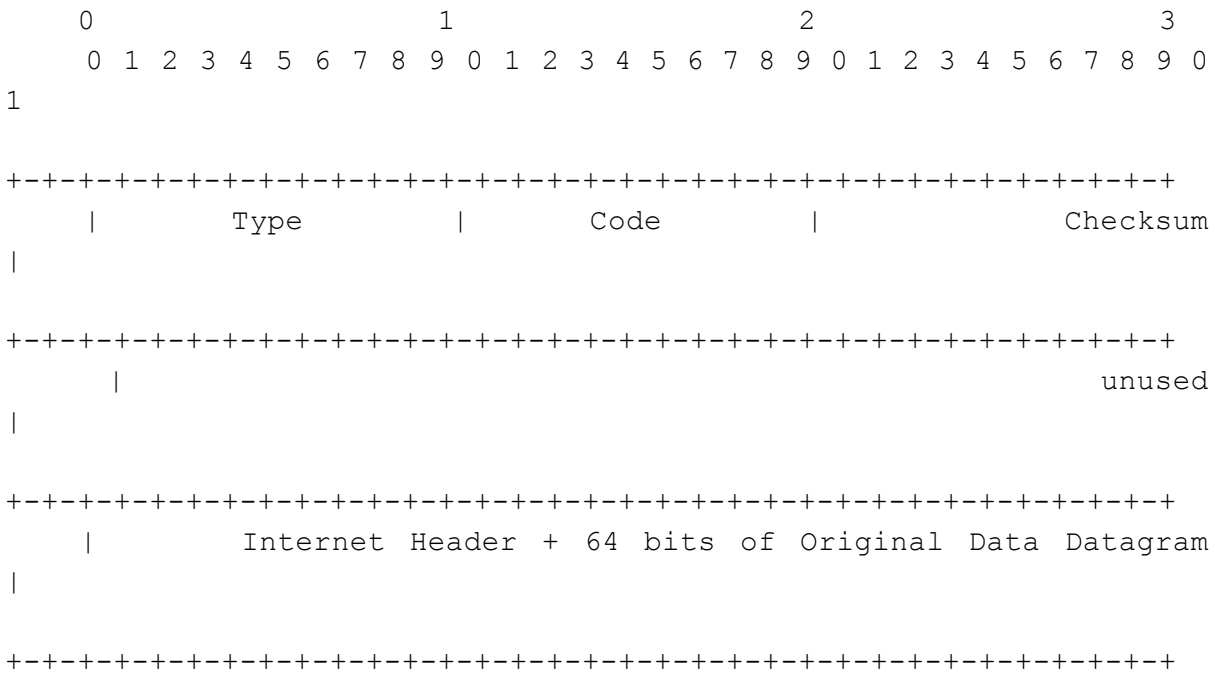
Exercício: Faça uma verificação do cabeçalho de pacotes IPv6 utilizando a saída do tcpdump que exibe o hexadecimal do pacote.

ICMP / RFC 792

O cabeçalho depende do tipo de mensagem ICMP:

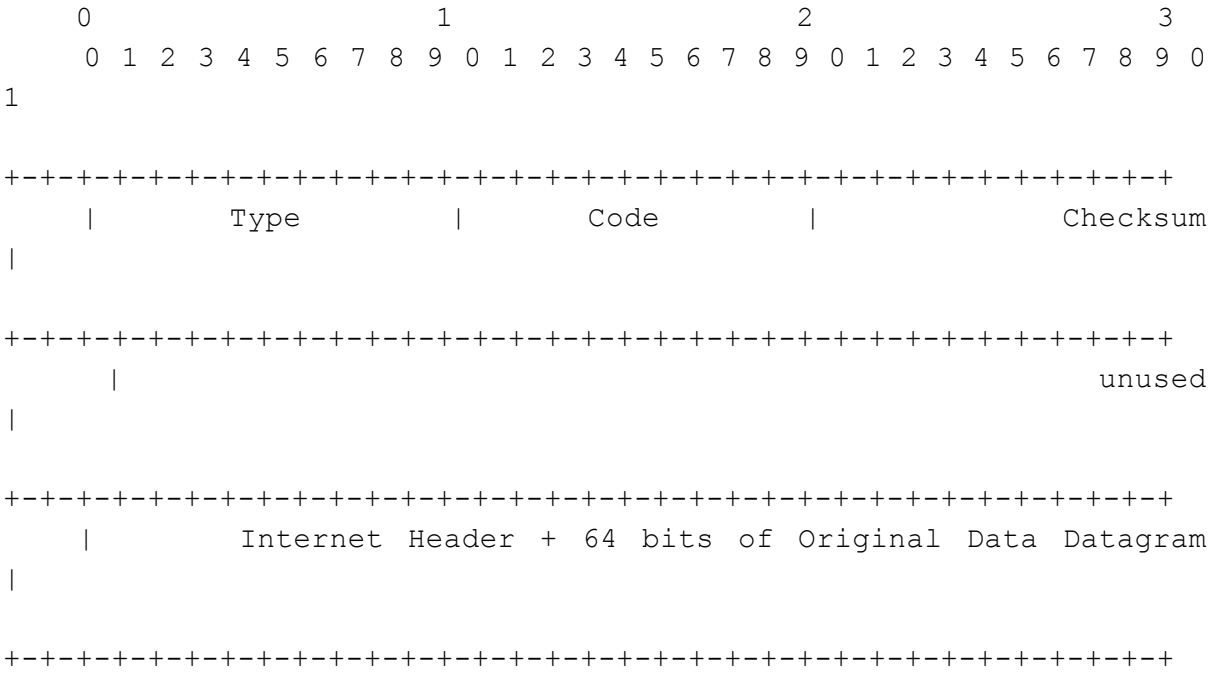
- Type:
    - 3: Destination Unreachable
    - 11: Time Exceeded
    - 8: Echo Request
    - 0: Echo Reply
- Code:
    - Destination Unreachable
        - 0 = net unreachable
        - 1 = host unreachable
        - 2 = protocol unreachable
        - 3 = port unreachable
        - 4 = fragmentation needed and DF set
        - 5 = source route failed.
    - Time Exceeded:
        - 0 = time to live exceeded in transit
        - 1 = fragment reassembly time exceeded
    - Echo Request / Reply:
        - 0 = default code
- Checksum: Integridade do pacote

Destination Unreachable Message



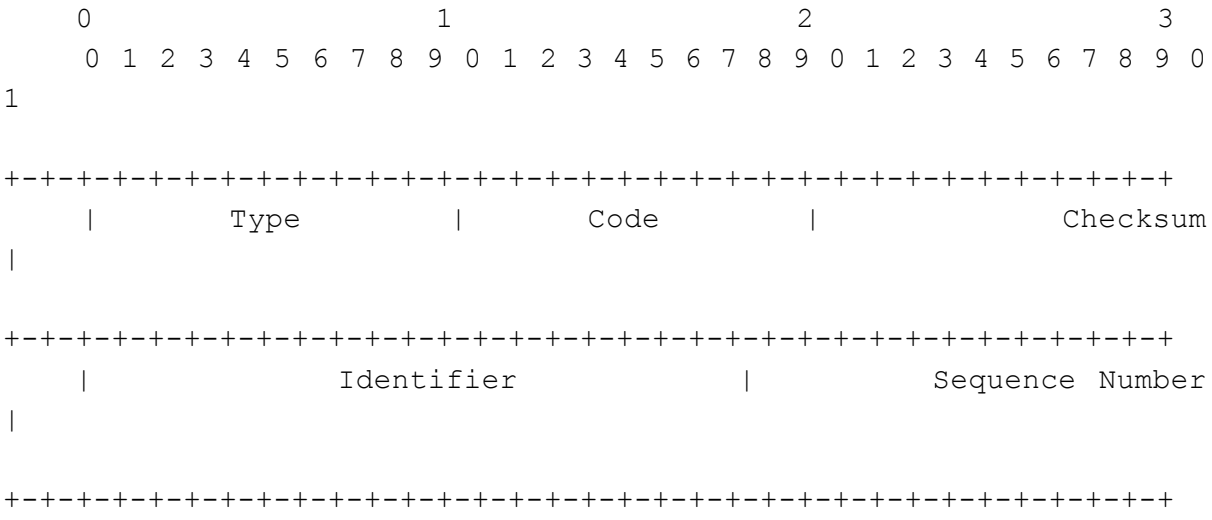
- Ping para um destino inalcançável. Ex: sem rota para uma rede X
- Usar o netcat para tentar acessar uma porta UDP que não está tendo acesso

Time Exceeded Message



- Observe que o traceroute/mtr utilizando esse tipo de mensagem ICMP
- Utilizando `ping -t 1 pop-ba.rnp.br` podemos fazer um ‘traceroute na mão’, variando o valor do TTL do pacote (parâmetro `-t`)
- Lembre que alguns equipamentos são configurados para não responderem tráfego com ICMP Time Exceeded e por isso não é possível identificar conectividade dessa maneira

Echo request/reply



- Ping tradicional

- Demonstração de conectividade com `hping` (bypass para quem não responde ICMP)

Análise de Tráfego ICMP

- Observe que com o tcpdump é possível verificar o tipo de mensagem ICMP que está sendo trafegada.

```
sudo tcpdump -i eno1 -n -v host 10.1.0.38 and icmp

ping -c 1 10.1.0.38
```

```
tcpdump: listening on eno1, link-type EN10MB (Ethernet), capture
size 262144 bytes
16:01:16.464365 IP (tos 0x0, ttl 64, id 40816, offset 0, flags
[DF], proto ICMP (1), length 84)
    10.1.0.114 > 10.1.0.38: ICMP echo request, id 2254, seq 1,
length 64
16:01:16.492272 IP (tos 0x0, ttl 64, id 46883, offset 0, flags
[none], proto ICMP (1), length 84)
    10.1.0.38 > 10.1.0.114: ICMP echo reply, id 2254, seq 1, length
64
```

- Caso um destino não seja alcançável, uma resposta ICMP é gerada pelo próprio comando ping. No caso abaixo, observe que no tráfego há envio de ARP request mas não houve nenhuma resposta. Em decorrência disso, a mensagem obtida pelo comando ping é de que o host está inalcançável.

```
sudo tcpdump -i eno1 -n -v host 10.1.0.200

ping -c 1 10.1.0.200
```

```
15:56:46.212251 ARP, Ethernet (len 6), IPv4 (len 4), Request
who-has 10.1.0.200 tell 10.1.0.38, length 28
15:56:47.238372 ARP, Ethernet (len 6), IPv4 (len 4), Request
who-has 10.1.0.200 tell 10.1.0.38, length 28
15:56:48.258376 ARP, Ethernet (len 6), IPv4 (len 4), Request
who-has 10.1.0.200 tell 10.1.0.38, length 28

PING 10.1.0.200 (10.1.0.200) 56(84) bytes of data.
From 10.1.0.38 icmp_seq=1 Destination Host Unreachable

--- 10.1.0.200 ping statistics ---
```

```
1 packets transmitted, 0 received, +1 errors, 100% packet loss,
time 0ms
```

- Além da mensagem de host inalcançável, podemos receber informações de rede, protocolo ou porta inalcançáveis. No caso a seguir, observe a resposta de porta inalcançável ao tentar comunicação com a porta 80/UDP.

```
sudo tcpdump -i eno1 -n -v host 10.1.0.38 and icmp

nc -u 10.1.0.38 80
```

```
15:52:54.456291 IP (tos 0xc0, ttl 64, id 63743, offset 0, flags
[none], proto ICMP (1), length 60)
    10.1.0.38 > 10.1.0.114: ICMP 10.1.0.38 udp port 80 unreachable,
length 40
        IP (tos 0x0, ttl 64, id 1818, offset 0, flags [DF], proto
UDP (17), length 32)
    10.1.0.114.56723 > 10.1.0.38.80: UDP, length 4
```

- O campo TTL é utilizado por ferramentas como traceroute e mtr para identificar os saltos no caminho de uma origem a um destino. Essas ferramentas se baseiam no fato de que a maioria dos equipamentos envia uma mensagem do tipo ICMP time exceeded quando o TTL alcança o valor 0. Essa mensagem contém o IP de origem do pacote, possibilitando a identificação dos hosts ao longo do caminho.

```
sudo tcpdump -i any -n -v icmp

ping -4 -c 1 -t 1 rnp.br
```

```
16:05:54.725935 IP (tos 0x0, ttl 1, id 1972, offset 0, flags [DF],
proto ICMP (1), length 84)
    10.1.0.114 > 104.22.9.95: ICMP echo request, id 2570, seq 1,
length 64

16:05:54.726138 IP (tos 0xc0, ttl 64, id 29906, offset 0, flags
[none], proto ICMP (1), length 112)
    10.1.0.1 > 10.1.0.114: ICMP time exceeded in-transit, length 92
        IP (tos 0x0, ttl 1, id 1972, offset 0, flags [DF], proto
ICMP (1), length 84)
    10.1.0.114 > 104.22.9.95: ICMP echo request, id 2570, seq 1,
length 64
```

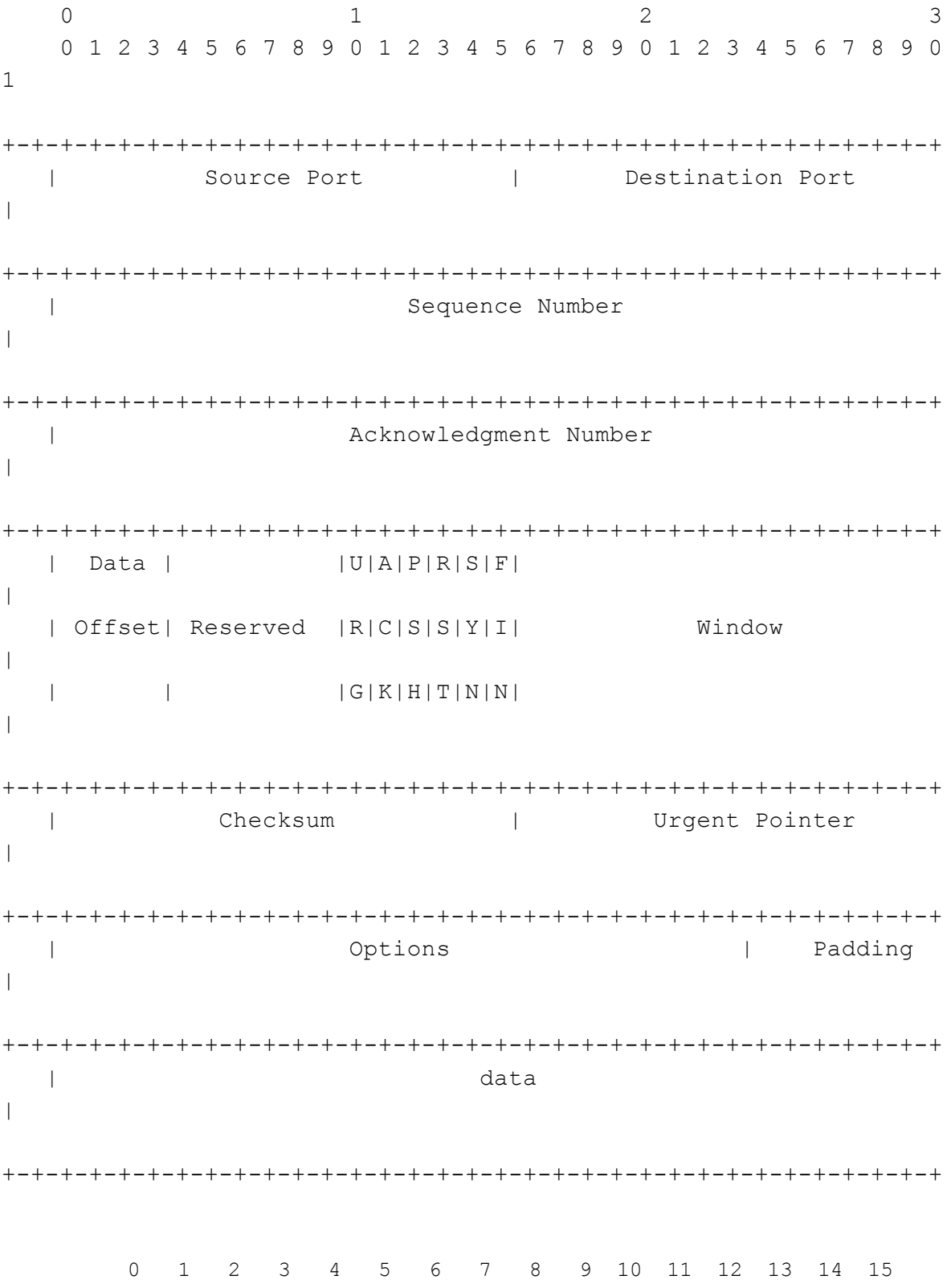
```
sudo tcpdump -i any -n -v icmp

ping -4 -c 1 -t 2 rnp.br
```

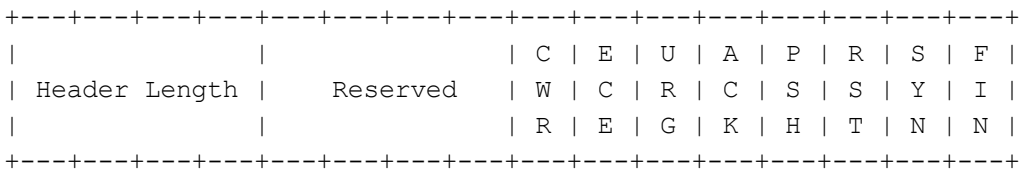
```
16:07:28.922857 IP (tos 0x0, ttl 2, id 26013, offset 0, flags
[DF], proto ICMP (1), length 84)
    10.1.0.114 > 104.22.8.95: ICMP echo request, id 2578, seq 1,
length 64
16:07:28.923270 IP (tos 0x0, ttl 254, id 0, offset 0, flags
[none], proto ICMP (1), length 56)

    200.128.6.148 > 10.1.0.114: ICMP time exceeded in-transit,
length 36
        IP (tos 0x0, ttl 1, id 26013, offset 0, flags [DF], proto
ICMP (1), length 84)
    10.1.0.114 > 104.22.8.95: ICMP echo request, id 2578, seq 1,
length 64
```

Exercício: Pensem em uma alternativa para verificar conectividade e identificar hosts quando algum equipamento não responder mensagens ICMP.

2º dia: TCP | UDP

TCP / RFC 793





- Source Port: campo com 16 bits, indicando 65536 possibilidades. As portas variam de 0 até 65535, mas a porta 0 é reservada pela IANA e não deve ser usada. Na visão do cliente, essa porta comumente é uma porta alta e associada de forma aleatória.
- Destination Port: campo com 16 bits, indicando 65536 possibilidades. As portas variam de 0 até 65535, mas a porta 0 é reservada pela IANA e não deve ser usada. Na visão do cliente, essa porta costuma ter serviços definidos, padronizados pela IANA (ver `/etc/services`). No entanto, é possível configurar serviços em portas diferentes. Por exemplo, por questões de segurança, você pode configurar o seu serviço SSH em uma porta diferente da porta padrão (22). Isso evita que, caso seja realizado um mapeamento da rede, o serviço seja identificado através do número da porta que está aberta. Apesar disso, esse processo não é recomendado para alguns serviços, como HTTP, DNS e NTP, visto que são serviços comumente abertos ao público externo.
- Sequence Number: identifica o segmento TCP. É semelhante ao campo *Identification* do IP. No entanto, no caso do TCP, o primeiro número será aleatório e o restante será sequencial de acordo com os dados já enviados. O número de sequência é um campo importante pois auxilia no processo de confiabilidade de entrega do protocolo TCP.
- Acknowledgment Number: o host envia um valor informando qual o próximo segmento que espera receber (Sequence Number + Length). Para saber disso, o host precisa ter recebido um segmento anterior e calculado o próximo a ser recebido, portanto, esse campo funciona como uma confirmação de recebimento dos pacotes recebidos. Uma vez estabelecida a conexão, sempre será enviado ACK nos pacotes. Além disso, é possível observar que os campos sequence number e ACK tem o mesmo tamanho (32 bits), já que o ACK mantém um sequence number.
- Data Offset: De acordo com a RFC, o data offset indica onde os dados começam. Alguns autores consideram que esse campo informa a quantidade de linhas do cabeçalho TCP, ou seja, tem a mesma função do *IHL* do IP. O campo tem 4 bits, então o valor pode variar de 0 a 15. As cinco primeiras linhas do cabeçalho são obrigatórias (assim como no IP). Dessa forma, o tamanho mínimo do cabeçalho TCP é de 20 bytes e o tamanho máximo é de 60 bytes. Dado encapsulamento TCP/IP, podemos ter um mínimo de 40 bytes (20 TCP + 20 IP) de cabeçalho e um máximo de 120 bytes (60 TCP + 60 IP).
- Reserved: Reservado para uso futuro. Alguns bits já foram usados e documentados em RFCs mais recentes (exemplo RFC 3168) atualizando a informação da que foi utilizada como referência nesse treinamento.
- Flags: é um campo composto por 12 bits. Atualmente, 8 dos 12 bits são utilizados para as flags. As flags CWR e ECE foram adicionadas na RFC 3168.

    - URG: indica que o pacote contém dados importantes. Essa flag não é muito utilizada.
    - ACK: indica uso do campo de reconhecimento. Só não será usada no início do three-way handshake.
    - PSH: utilizada para sinalizar que há dados no payload do segmento TCP.
    - RST: indica que o host não entendeu o segmento recebido.
    - SYN: utilizada para iniciar conexão. Obs: A conexão deve ser aberta nos dois sentidos.
    - FIN: utilizada para informar a finalização da conexão
        - Há duas formas de fechamento de conexão: meio fechamento e fechamento completo. No meio fechamento, um lado fecha completamente sua conexão, enquanto o outro permanece aberto. Já o fechamento completo ocorre de forma semelhante ao three-way handshake.
        - Meio fechamento: FIN | ACK | FIN | ACK
        - Fechamento completo: FIN | FIN/ACK | ACK
    - CWR e ECE: estão relacionadas à Qualidade de Serviço (QoS). Como são flags mais novas, nem todos os equipamentos dão suporte.
- Window: Quantidade de octetos que o host está apto a aceitar em um segmento. Associado ao campo opcional *wscale*, possibilita o uso de janelas deslizantes, que permite o envio de vários segmentos em série, sem a necessidade de confirmação individual por pacote.
- Checksum: campo similar ao *Checksum* do IP, a diferença é que no caso do TCP, o conteúdo também é verificado. Como o TCP é um protocolo fim a fim, o checksum é calculado apenas na origem e no destino. Desta forma, não há adição de processamento nos hosts intermediários.
- Urgent Pointer: é utilizado em conjunto com a flag URG e tem o objetivo de enviar informações que sejam urgentes para o tráfego na rede. O campo contém a posição final, dentro do payload, dos dados que são urgentes.
- Options:
    - Tipo 0: indica que foi inserido o último elemento do campo Options. Só será usado se o fim do campo Options não coincidir com o fim do cabeçalho TCP. Seu código é 00h
    - Tipo 1: utilizado como alinhamento de informações, fazendo com que o próximo dado comece em um bit que seja o início de um byte. Seu código é 01h
    - **Tipo 2:** *maximum segment size* (*mss*), indica a maior quantidade de dados que o payload de um segmento TCP poderá conter. Esse campo só poderá aparecer com a flag SYN. Seu código é 02h.
    - **Tipo 3:** *window scale* (*wscale*), é uma técnica de expansão do campo *Window Size*. Seu código é 03h.
    - Tipo 4: *selective ack ok* ou *sack-permitted* (*sackOK*), é mais uma forma de avisar o host oposto sobre uma possível ocorrência de segmentos chegando fora de ordem ou sendo perdidos. No entanto, ele é mais detalhista do que a flag ACK, pois informa exatamente quais segmentos foram recebidos. Esse campo só poderá aparecer com a flag SYN. Seu código é 04h.

    - Tipo 5: *selective ack* (*sack*). Esse campo só poderá aparecer com a flag SYN. Seu código é 05h.
    - Tipo 8: *timestamp*, é utilizado para calcular o RTT.
- Padding: usado para garantir que o cabeçalho IP e os dados encapsulados estejam alinhados a 32 bits, composto de zeros. O padding é necessário quando os campos opcionais são utilizados e não alcançam o tamanho de 32 bits.
- Data: campo onde são colocados os dados enviados no segmento TCP

Análise de Tráfego TCP

- Veja as portas de origem e de destino. Perceba que comumente a porta do servidor é uma porta conhecida e a porta do cliente é aleatória. Em alguns serviços não funcionam dessa maneira, como NTP e DHCP (mas esses rodam sobre UDP). Em muitos casos, o NTP usa a mesma porta para servidor e cliente (porta 123).

```
sudo tcpdump -i eno1 -n -v host 10.1.0.38 and tcp

nc 10.1.0.38 80
```

```
14:12:07.770326 IP 10.1.0.114.38504 > 10.1.0.38.80: Flags [S], seq
2231480575, win 64240, options [mss 1460,sackOK,TS val 597533526
ecr 0,nop,wscale 7], length 0
14:12:07.793391 IP 10.1.0.38.80 > 10.1.0.114.38504: Flags [S.], seq
3213945376, ack 2231480576, win 65160, options [mss 1324,sackOK,TS
val 728386932 ecr 597533526,nop,wscale 7], length 0
14:12:07.793456 IP 10.1.0.114.38504 > 10.1.0.38.80: Flags [.], ack
1, win 502, options [nop,nop,TS val 597533549 ecr 728386932],
length 0
14:12:08.632724 IP 10.1.0.114.38504 > 10.1.0.38.80: Flags [F.], seq
1, ack 1, win 502, options [nop,nop,TS val 597534388 ecr
728386932], length 0
14:12:08.662462 IP 10.1.0.38.80 > 10.1.0.114.38504: Flags [F.], seq
1, ack 2, win 510, options [nop,nop,TS val 728387801 ecr
597534388], length 0
14:12:08.662513 IP 10.1.0.114.38504 > 10.1.0.38.80: Flags [.], ack
2, win 502, options [nop,nop,TS val 597534418 ecr 728387801],
length 0
```

Exercício: Façam uma nova conexão na porta 80 de algum servidor web e observem a mudança da porta cliente.

Ver a atribuição de portas a determinados serviços:

- https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

```
$ less /etc/services
$ getent services http
http            80/tcp
$ getent services 53
domain          53/tcp
$ getent services ftp
ftp             21/tcp
```

Listar as portas que estão em modo "*listening"* no SO:

```
lsof -i
netstat -lnvpt
ss -lnpt
```

- Observe a correspondência entre os campos ACK e Sequence Number, usados para validação e ordenação de mensagens.

```
sudo tcpdump -i eno1 -n host 10.1.0.38 and tcp -S

nc 10.1.0.38 80
```

```
14:17:46.392644 IP 10.1.0.114.39002 > 10.1.0.38.80: Flags [S], seq 3886319713, win 64240, options [mss 1460,sackOK,TS val 597872148 ecr 0,nop,wscale 7], length 0
14:17:46.443354 IP 10.1.0.38.80 > 10.1.0.114.39002: Flags [S.], seq 1108175147, ack 3886319714, win 65160, options [mss 1324,sackOK,TS val 728725561 ecr 597872148,nop,wscale 7], length 0
14:17:46.443402 IP 10.1.0.114.39002 > 10.1.0.38.80: Flags [.], ack 1108175148, win 502, options [nop,nop,TS val 597872199 ecr 728725561], length 0
14:17:47.691998 IP 10.1.0.114.39002 > 10.1.0.38.80: Flags [F.], seq 3886319714, ack 1108175148, win 502, options [nop,nop,TS val 597873447 ecr 728725561], length 0
14:17:47.717521 IP 10.1.0.38.80 > 10.1.0.114.39002: Flags [F.], seq 1108175148, ack 3886319715, win 510, options [nop,nop,TS val 728726855 ecr 597873447], length 0
14:17:47.717592 IP 10.1.0.114.39002 > 10.1.0.38.80: Flags [.], ack 1108175149, win 502, options [nop,nop,TS val 597873473 ecr 728726855], length 0
```

- Agora veja a relação entre os campos length, sequence number e ACK. Os campos indicam a quantidade de bytes recebidos e o que é esperado receber nos próximos

pacotes. Observem que apenas os pacotes com a flag PUSH habilitada têm tamanho superior a 0. Isso ocorre pois essa é a única flag que permite que sejam enviados dados utilizando TCP.

```
sudo tcpdump -i eno1 -n host 10.1.0.38 and tcp

nc 10.1.0.38 80
```

14:26:16.208894 IP 10.1.0.114.39010 > 10.1.0.38.80: Flags [S], **seq 2777450481**, win 64240, options [mss 1460,sackOK,TS val 598381964 ecr 0,nop,wscale 7], **length 0**
14:26:16.235235 IP 10.1.0.38.80 > 10.1.0.114.39010: Flags [S.], **seq 1470946849**, **ack 2777450482**, win 65160, options [mss 1324,sackOK,TS val 729235377 ecr 598381964,nop,wscale 7], **length 0**
14:26:16.235284 IP 10.1.0.114.39010 > 10.1.0.38.80: Flags [.], **ack 1**, win 502, options [nop,nop,TS val 598381991 ecr 729235377], **length 0**
14:26:20.975069 IP 10.1.0.114.39010 > 10.1.0.38.80: Flags [P.], **seq 1:14**, **ack 1**, win 502, options [nop,nop,TS val 598386730 ecr 729235377], **length 13**: HTTP
14:26:21.089951 IP 10.1.0.38.80 > 10.1.0.114.39010: Flags [.], **ack 14**, win 509, options [nop,nop,TS val 729240145 ecr 598386730], **length 0**
14:26:21.090004 IP 10.1.0.38.80 > 10.1.0.114.39010: Flags [P.], **seq 1:484**, **ack 14**, win 509, options [nop,nop,TS val 729240146 ecr 598386730], **length 483**: HTTP: HTTP/1.1 400 Bad Request
14:26:21.090033 IP 10.1.0.114.39010 > 10.1.0.38.80: Flags [.], **ack 484**, win 501, options [nop,nop,TS val 598386845 ecr 729240146], **length 0**
14:26:21.090050 IP 10.1.0.38.80 > 10.1.0.114.39010: Flags [F.], **seq 484**, **ack 14**, win 509, options [nop,nop,TS val 729240146 ecr 598386730], **length 0**
14:26:21.133130 IP 10.1.0.114.39010 > 10.1.0.38.80: Flags [.], **ack 485**, win 501, options [nop,nop,TS val 598386888 ecr 729240146], **length 0**
14:26:22.199244 IP 10.1.0.114.39010 > 10.1.0.38.80: Flags [F.], **seq 14**, **ack 485**, win 501, options [nop,nop,TS val 598387955 ecr 729240146], **length 0**
14:26:22.260712 IP 10.1.0.38.80 > 10.1.0.114.39010: Flags [.], **ack 15**, win 509, options [nop,nop,TS val 729241371 ecr 598387955], **length 0**

- Observe o tráfego TCP em hexadecimal. Com esse mecanismo de visualização é possível verificar o valor de campos como data offset.

```
sudo tcpdump -i tap0 -n host 10.1.0.38 and tcp -X

netcat 10.1.0.38 80
```

```
14:42:28.415951 IP 10.1.0.114.39016 > 10.1.0.38.80: Flags [S], seq
2570042444, win 64240, options [mss 1460,sackOK,TS val 599354171
ecr 0,nop,wscale 7], length 0
0x0000:  4500 003c 4986 4000 4006 dc9c 0a01 0072  E..<I.@.@......r
0x0010:  0a01 0026 9868 0050 992f bc4c 0000 0000  ...&.h.P./.L....
0x0020:  a002 faf0 14c8 0000 0204 05b4 0402 080a  ................
0x0030:  23b9 6b3b 0000 0000 0103 0307            #.k;........
```

- As flags TCP são bastante utilizadas. Veja a utilização da flag SYN para estabelecimento de conexão, da flag PUSH para envio de dados e da flag FIN para encerramento da conexão. Além disso, observe que o ACK é utilizado em todos os momentos, exceto no envio do primeiro SYN.

```
sudo tcpdump -i tap0 -n host 10.1.0.38 and tcp

netcat 10.1.0.38 80
```

14:45:49.524193 IP 10.1.0.114.39018 > 10.1.0.38.80: **Flags [S]**, seq 584169342, win 64240, options [mss 1460,sackOK,TS val 599555280 ecr 0,nop,wscale 7], **length 0**
14:45:49.539439 IP 10.1.0.38.80 > 10.1.0.114.39018: **Flags [S.]**, seq 356366604, ack 584169343, win 65160, options [mss 1324,sackOK,TS val 730408696 ecr 599555280,nop,wscale 7], **length 0**
14:45:49.539486 IP 10.1.0.114.39018 > 10.1.0.38.80: **Flags [.]**, ack 1, win 502, options [nop,nop,TS val 599555295 ecr 730408696], **length 0**
14:45:52.905342 IP 10.1.0.114.39018 > 10.1.0.38.80: **Flags [P.]**, seq 1:5, ack 1, win 502, options [nop,nop,TS val 599558661 ecr 730408696], **length 4**: HTTP
14:45:52.918869 IP 10.1.0.38.80 > 10.1.0.114.39018: **Flags [.]**, ack 5, win 510, options [nop,nop,TS val 730412075 ecr 599558661], **length 0**
14:45:52.931831 IP 10.1.0.38.80 > 10.1.0.114.39018: **Flags [P.]**, seq 1:484, ack 5, win 510, options [nop,nop,TS val 730412075 ecr 599558661], **length 483**: HTTP: HTTP/1.1 400 Bad Request
14:45:52.931895 IP 10.1.0.114.39018 > 10.1.0.38.80: **Flags [.]**, ack 484, win 501, options [nop,nop,TS val 599558687 ecr 730412075], **length 0**
14:45:52.931916 IP 10.1.0.38.80 > 10.1.0.114.39018: **Flags [F.]**, seq 484, ack 5, win 510, options [nop,nop,TS val 730412075 ecr 599558661], **length 0**
14:45:52.973122 IP 10.1.0.114.39018 > 10.1.0.38.80: **Flags [.]**, ack 485, win 501, options [nop,nop,TS val 599558729 ecr 730412075], **length 0**
14:45:54.113664 IP 10.1.0.114.39018 > 10.1.0.38.80: **Flags [F.]**, seq 5, ack 485, win 501, options [nop,nop,TS val 599559869 ecr 730412075], **length 0**

14:45:54.286325 IP 10.1.0.38.80 > 10.1.0.114.39018: **Flags [.]**, ack 6, win 510, options [nop,nop,TS val 730413403 ecr 599559869], **length 0**

Exercício: Analise a saída em hexadecimal do tráfego abaixo e tente descobrir quais flags TCP estão habilitadas:

```
0x0000:  4500 0034 ea30 4000 4006 3bfa 0a01 0026  E..4.0@.@.;....&
0x0010:  0a01 0072 0050 9868 acd9 56a0 992f bc54  ...r.P.h..V../.T
0x0020:  8011 01fe 53f8 0000 0101 080a 2b86 3ab4  ....S.......+.:.
0x0030:  23b9 9082                                #...
```

- Além das flags citadas anteriormente, a flag RESET também é comumente utilizada para sinalizar o não entendimento. No caso a seguir, a flag é utilizada para cancelar a abertura de uma conexão durante a varredura de portas TCP através da ferramenta nmap.

```
sudo tcpdump -i eno1 -n host 10.1.0.38 and tcp

sudo hping3 -p 80 -S 10.1.0.38 -c 1
sudo nmap -p80 --scanflags SYN 10.1.0.38
```

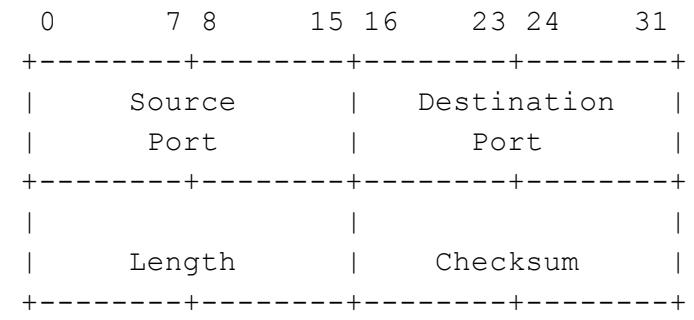
14:50:56.793485 IP 10.1.0.114.35101 > 10.1.0.38.80: **Flags [S]**, seq 842197359, win 1024, options [**mss 1460**], length 0
14:50:56.831311 IP 10.1.0.38.80 > 10.1.0.114.35101: **Flags [S.]**, seq 2319280844, ack 842197360, win 64240, options [**mss 1324**], length 0
14:50:56.831363 IP 10.1.0.114.35101 > 10.1.0.38.80: **Flags [R]**, seq 842197360, win 0, length 0

- Os campos wscale e window são usados para utilizar janelas deslizantes. Esses campos são definidos no início da conexão mas podem ser alterados posteriormente.

15:12:03.232303 IP 10.1.0.38.51948 > 10.1.0.114.80: Flags [S], seq 1657538111, **win 64240**, options [mss 1460,sackOK,TS val 731927289 ecr 0,nop,**wscale 7**], length 0
15:12:03.258416 IP 10.1.0.114.80 > 10.1.0.38.51948: Flags [S.], seq 3586172498, ack 1657538112, **win 65160**, options [mss 1324,sackOK,TS val 601073889 ecr 731927289,nop,**wscale 7**], length 0
15:12:03.258478 IP 10.1.0.38.51948 > 10.1.0.114.80: Flags [.], ack 1, **win 502**, options [nop,nop,TS val 731927316 ecr 601073889], length 0

15:12:04.909626 IP 10.1.0.38.51948 > 10.1.0.114.80: Flags [P.], seq 1:7, ack 1, **win 502**, options [nop,nop,TS val 731928967 ecr 601073889], length 6: HTTP: get /

UDP



- Source Port: campo com 16 bits, indicando 65536 possibilidades. As portas variam de 0 até 65535, mas a porta 0 é reservada pela IANA e não deve ser usada. Na visão do cliente, essa porta comumente é uma porta alta e associada de forma aleatória. Alguns protocolos, como o DHCP, tem uma porta de cliente predefinida: 67/UDP servidor e 68/UDP cliente. No protocolo UDP, esse campo **não é obrigatório**.
- Destination Port: campo com 16 bits, indicando 65536 possibilidades. As portas variam de 0 até 65535, mas a porta 0 é reservada pela IANA e não deve ser usada. Na visão do cliente, essa porta costuma-se ter serviços definidos, padronizados pela IANA (ver `/etc/services`).
- Length: indica o tamanho total do segmento UDP (cabeçalho + payload) medido em bytes. Como o cabeçalho é obrigatório, o valor mínimo desse campo é de 8 bytes.
- Checksum: é calculado de forma idêntica ao *Checksum* do TCP.

Análise de Tráfego UDP

- Compare os dois tráfegos a seguir e observe como o tráfego UDP é mais simples. O protocolo não é orientado a conexão e é mais simples do que o TCP.

```
sudo tcpdump -i eno1 -n host 10.1.0.38 and udp
nc -u 10.1.0.38 23

sudo tcpdump -i eno1 -n host 10.1.0.38 and tcp
nc 10.1.0.38 79
```

Referências

Análise de Tráfego:

- Livro: MOTA FILHO, João Eriberto. **Análise de Tráfego em Redes TCP/IP: Utilize tcpdump na análise de tráfegos em qualquer sistema operacional**. Novatec Editora, 2013.
- Minicurso Análise de Tráfego em Redes TCP/IP Parte 1: https://www.youtube.com/watch?v=gK3gl3Vh8L0
- Minicurso Análise de Tráfego em Redes TCP/IP Parte 2: https://www.youtube.com/watch?v=YFOBLyf2SG0

IPv4:

- https://tools.ietf.org/html/rfc791

IPv6:

- http://ipv6.br/
- https://tools.ietf.org/html/rfc8200

ICMP:

- https://tools.ietf.org/html/rfc792

TCP:

- https://tools.ietf.org/html/rfc793
- https://tools.ietf.org/html/rfc3168#section-23.2

UDP:

- https://tools.ietf.org/html/rfc768