EA080 - O — Análise Forense em Redes de Computadores

Professor: Christian Esteve Rothenberg

Leonardo Rodrigues Marques `RA: 178610`

---

1 Atividade 1

1.1 Análise do Programa P4

- Definição dos cabeçalhos Ethernet e IPv4 de pacotes: linhas 15 e 21
- Parser dos cabeçalhos Ethernet e IPv4 de pacotes: 58 e 66
- Procura de destino IPv4 na tabela de roteamento: 100
- Atualização de endereços MAC fonte e destino: 95 e 96
- Decremento do campo time-to-live (TTL) do pacote: 29
- Definição da porta de saída (egress) do pacote: 124
- Cálculo do checksum: 134

1.2 Execução do Programa P4.

Os campos Ethernet são diferentes porque ambos os hosts estão conectados através de switches, portanto eles devem enviar mensagens anteriormente para esses dispositivos, que definitivamente possuem MACs address diferentes. No caso de h1, o dst é uma mensagem do tipo broadcast para todos os componentes da rede a fim encontrar o destino final para o pacote e o src, o próprio endereço MAC do dispositivo. Para o host h4, o endereço src e do switch adjacente e dst, o próprio endereço MAC em que o pacote será destinado. Além das mudanças nos endereços dos campos src e dst de Ethernet, no campo TTL, houve uma diminuição de 64 para 61 (ms?). Os valores estão corretos. No host h1, onde o pacote foi gerado, o campo TTL foi setado para 64ms. Ao ser enviado, o pacote gastou um tempo de 3 ms (64-61) até atingir o host h4 (tempo é subtraído). Portanto lá, restavam ainda, 61ms até o pacote ser extinto.

```
root@p4:~/tutorials/exercises/basic# ./send.py 10.0.4.4 "Ola"
WARNING: No route found for IPv6 destination :: (no default route?)
sending on interface eth0 to 10.0.4.4
###[ Ethernet ]###
  dst       = ff:ff:ff:ff:ff:ff
  src       = 08:00:00:00:01:11
  type      = 0x800
###[ IP ]###
     version   = 4L
     ihl       = 5L
     tos       = 0x0
     len       = 43
     id        = 1
     flags     =
     frag      = 0L
     ttl       = 64
     proto     = tcp
     chksum    = 0x61c8
     src       = 10.0.1.1
     dst       = 10.0.4.4
     \options   \
###[ TCP ]###
        sport     = 53337
        dport     = 1234
        seq       = 0
        ack       = 0
        dataofs   = 5L
        reserved  = 0L
        flags     = S
        window    = 8192
        chksum    = 0xf142
        urgptr    = 0
        options   = []
###[ Raw ]###
           load      = 'Ola'
root@p4:~/tutorials/exercises/basic#
```

```
root@p4:~/tutorials/exercises/basic# ./receive.py
WARNING: No route found for IPv6 destination :: (no default route?)
sniffing on eth0
got a packet
###[ Ethernet ]###
  dst       = 08:00:00:00:04:44
  src       = 08:00:00:00:02:00
  type      = 0x800
###[ IP ]###
     version   = 4L
     ihl       = 5L
     tos       = 0x0
     len       = 43
     id        = 1
     flags     =
     frag      = 0L
     ttl       = 61
     proto     = tcp
     chksum    = 0x64c8
     src       = 10.0.1.1
     dst       = 10.0.4.4
     \options   \
###[ TCP ]###
        sport     = 53337
        dport     = 1234
        seq       = 0
        ack       = 0
        dataofs   = 5L
        reserved  = 0L
        flags     = S
        window    = 8192
        chksum    = 0xf142
        urgptr    = 0
        options   = []
###[ Raw ]###
           load      = 'Ola'
```

1.3 Programando em P4: Comunicação de alunos em diferentes planetas.

Considerando que há 24 bits, o número de planetas suportados seriam:

$$\text{Planetas Suportados} = 2^{24} = 16777216$$

Considerando que há 32 bits e que é necessário 20 bits para representar um RA:

$$\text{Alunos suportdos} = \frac{2^{32}}{20} = 214748364$$

A comunicação poderia ser feito da seguinte forma: em primeiro lugar, identificaríamos o planeta, ou seja, os switches, e depois identificaríamos a pessoa, o host, através do RA.

Os pacotes não foram recebidos no host 2, entretanto ao executar o mesmo comando no xterm de h2, o host 1 recebe os pacotes.

```
- planet_fib_match: planet.seq=178610 => fib_hit_nexthop(dmac=08:00:00:00:01:11, port=1)
- planet_fib_match: planet.seq=222222 => fib_hit_nexthop(dmac=08:00:00:00:02:00, port=2)
```

As duas linhas indicam ao P4 que a chave 178610 deve ser usada para buscar dentro da tabela planeta. Caso ocorra um match, ele realiza ações seguindo a tabela:

```
action on_miss() {
}

action fib_hit_nexthop(dmac, port) {
  modify_field(ethernet.dstAddr, dmac);
  modify_field(standard_metadata.egress_spec, port);
}

table planet_fib_match {
  reads {
    planet.seq : exact;
  }
  actions {
    fib_hit_nexthop;
    on_miss;
  }
  size : 512;
}
```

O comando `./send.sh` identifica um pacote com seq=178610 e o envia automaticamente para port=1, portanto ao executar o comando em h1, os pacotes não chegam a h2. Entretanto ao executar o comando em h2, os pacotes chegam a h1 passando por port=1 (configuração setada).