Lab 4 : Introdução a planos de dados programáveis com a linguagem P4

Rodrigo Seiji Piubeli Hirao (186837)

16 de dezembro de 2021

Conteúdo

1 Introdução

Neste laboratório será estudada a linguagem de programação **P4**, usada para desenvolver protocolos. Assim será também criado um simples protocolo para comunicação entre alunos de diferentes planetas.

2 Metodologia

2.1 Análise do programa P4

Primeiramente foi analizado o que ocorria em um programa pré existente em p4 que cria a topologia da Figura 01 configurando os switches.

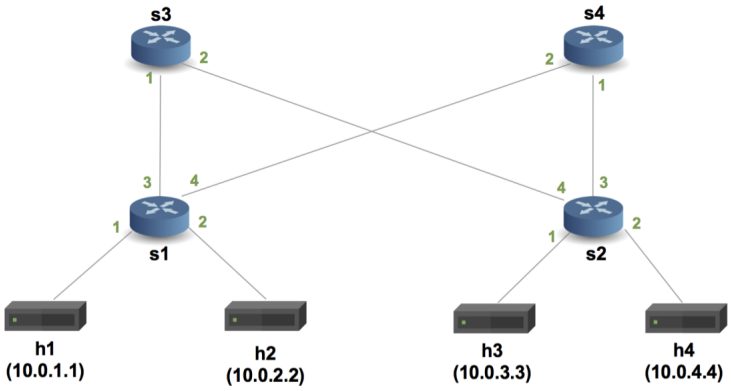

Figura 1: Topologia Experimental

2.2 Compilação e execução do programa P4

Então foi compilada a topologia para ser analizado o comportamento dos switches.

2.3 Programando em P4: Comunicação entre alunos de diferentes planetas

Foi então usado um script em p4, e adicionado as seguintes linhas para definir o cabeçalho do protocolo

```
13 header_type planet_t {
14     fields {
15         source : 24;
16         destination : 24;
17         seq : 32;
18     }
19 }
```

Permitinho endereços de 24 bits, ou seja 2^{24} endereços diferentes (2^{24} alunos diferentes).
Então foi compilado e rodado o código com a topologia da Figura 02

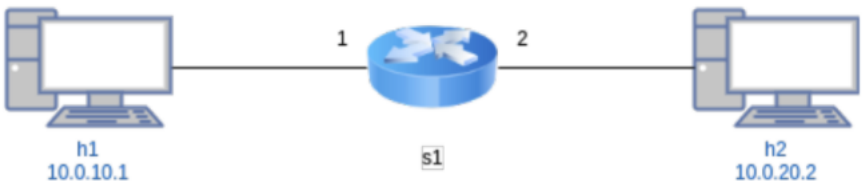

Figura 2: Topologia Experimental Planet

E enviado uma mensagem para o destino **186837**

3 Resultados e Discussão

3.1 Análise do programa P4

Pode ser visto as definições de cabeçalho no seguinte pedaço de código

```
15 header ethernet_t {
16     macAddr_t dstAddr;
17     macAddr_t srcAddr;
18     bit<16>   etherType;
19 }
20
21 header ipv4_t {
22     bit<4>    version;
23     bit<4>    ihl;
24     bit<8>    diffserv;
25     bit<16>   totalLen;
26     bit<16>   identification;
27     bit<3>    flags;
28     bit<13>   fragOffset;
29     bit<8>    ttl;
30     bit<8>    protocol;
31     bit<16>   hdrChecksum;
32     ip4Addr_t srcAddr;
33     ip4Addr_t dstAddr;
34 }
```

Que serão então parseados usando o parser definido na classe **MyParser**, onde será extraído os dados de ethernet e, dentro deste, do ipv4.

```
49 parser MyParser(packet_in packet,
50                  out headers hdr,
51                  inout metadata meta,
52                  inout standard_metadata_t standard_metadata) {
53
54     state start {
55         transition parse_ethernet;
56     }
57
58     state parse_ethernet {
59         packet.extract(hdr.ethernet);
60         transition select(hdr.ethernet.etherType) {
61             TYPE_IPV4: parse_ipv4;
62             default: accept;
63         }
64     }
65
66     state parse_ipv4 {
67         packet.extract(hdr.ipv4);
68         transition accept;
69     }
70
71 }
```

Passando pela tabela de roteamento com sua procura do ipv4

```
100     table ipv4_lpm {
101         key = {
102             hdr.ipv4.dstAddr: lpm;
103         }
104         actions = {
105             ipv4_forward;
106             drop;
107             NoAction;
108         }
109         size = 1024;
110         default_action = drop();
111     }
112
113     apply {
114         if (hdr.ipv4.isValid()) {
115             ipv4_lpm.apply();
116         }
117     }
```

Então podemos ver a definição da porta de saída, seguido pela atualização dos endereços MAC e do decremento do TTL no pedaço de código a seguir

```
93      action ipv4_forward(macAddr_t dstAddr, egressSpec_t port) {
94          standard_metadata.egress_spec = port;
95          hdr.ethernet.srcAddr = hdr.ethernet.dstAddr;
96          hdr.ethernet.dstAddr = dstAddr;
97          hdr.ipv4.ttl = hdr.ipv4.ttl - 1;
98      }
```

E o cálculo do checksum na classe **MyComputeChecksum**

```
134 control MyComputeChecksum(inout headers  hdr, inout metadata meta) {
135      apply {
136  update_checksum(
137      hdr.ipv4.isValid(),
138            { hdr.ipv4.version,
139        hdr.ipv4.ihl,
140              hdr.ipv4.diffserv,
141              hdr.ipv4.totalLen,
142              hdr.ipv4.identification,
143              hdr.ipv4.flags,
144              hdr.ipv4.fragOffset,
145              hdr.ipv4.ttl,
146              hdr.ipv4.protocol,
147              hdr.ipv4.srcAddr,
148              hdr.ipv4.dstAddr },
149            hdr.ipv4.hdrChecksum,
150            HashAlgorithm.csum16);
151      }
152 }
```

3.2 Compilação e execução do programa P4

Foi então enviado o seguinte pacote por h1

```
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       = 50
     id        = 1
     flags     =
     frag      = 0L
     ttl       = 64
     proto     = tcp
     chksum    = 0x61c1
     src       = 10.0.1.1
     dst       = 10.0.4.4
###[ TCP ]###
        sport     = 62424
        dport     = 1234
        seq       = 0
        ack       = 0
        dataofs   = 5L
        reserved  = 0L
        flags     = S
        window    = 8192
        chksum    = 0xc78f
        urgptr    = 0
```

```
        options   = []
###[ Raw ]###
           load      = 'P4 is cool'
```

Que foi recebido de h4 com os mesmos cabeçalhos IP e TCP, mas diferentes MAC no ethernet. Isso se deve ao fato do thernet se comunicar com o switch, que traduz para o próximo no caminho, procurando o caminho na tabela de roteamento pelo ip e traduzindo para um novo MAC

```
got a packet
###[ Ethernet ]###
  dst       = 08:00:00:00:04:44
  src       = 08:00:00:00:02:00
  type      = 0x800
###[ IP ]###
     (...)
     ttl       = 61
     (...)
(...)
```

Vê-se também que o ttl indica que o pacote passou por 3 switches

3.3 Programando em P4: Comunicação entre alunos de diferentes planetas

Esse protocolo pode ser usado como o IP, de tal forma que os switches não tem endereços, mas possuem tabelas de redirecionamento, para encaminhar o pacote ao destino (nota-se que esse protocolo possui apenas 24 bits de tamanho, menor o ipv4 com 32 bits) E usando, além do endereço para localização, a sequência para temporização, assim definindo a ordem e perda dos pacotes.

Ao analizar a topologia no pedaço de código (s1-runtime.json) a seguir vemos que o switch irá sempre encaminhas destinos **186837** para a porta 1, que está ligado apenas no host1, o que significa que pacotes enviados de h2 para 186837 serão recebidos por h1, mas pacotes enviados de h1 para 186837 serão devolvidos para h1, como pode ser comprovado pelo wireshark (arquivos pcap em anexo, nota que o destino está em hexadecimal 2D9D5, que é o equivalente a 186837 em decimal).

```
12      {
13        "table": "planet_fib_match",
14        "match": {
15          "planet.seq": 186837
16        },
17        "action_name": "fib_hit_nexthop",
18        "action_params": {
19          "dmac": "08:00:00:00:01:11",
20          "port": 1
21        }
22      },
```

4 Conclusão

Pôde ser visto no laboratório o funcionamento do P4, que funciona de maneira intuitiva e prática para criar um protocolo de rede. O caso do protocolo planets criado mostrou o poder da linguagem, criando uma camada de rede similiar ao protocolo IP, mas simplificado.