Лабораторная работа № 4.
Базовая 'коммутация' и туннелирование используя язык программирования P4

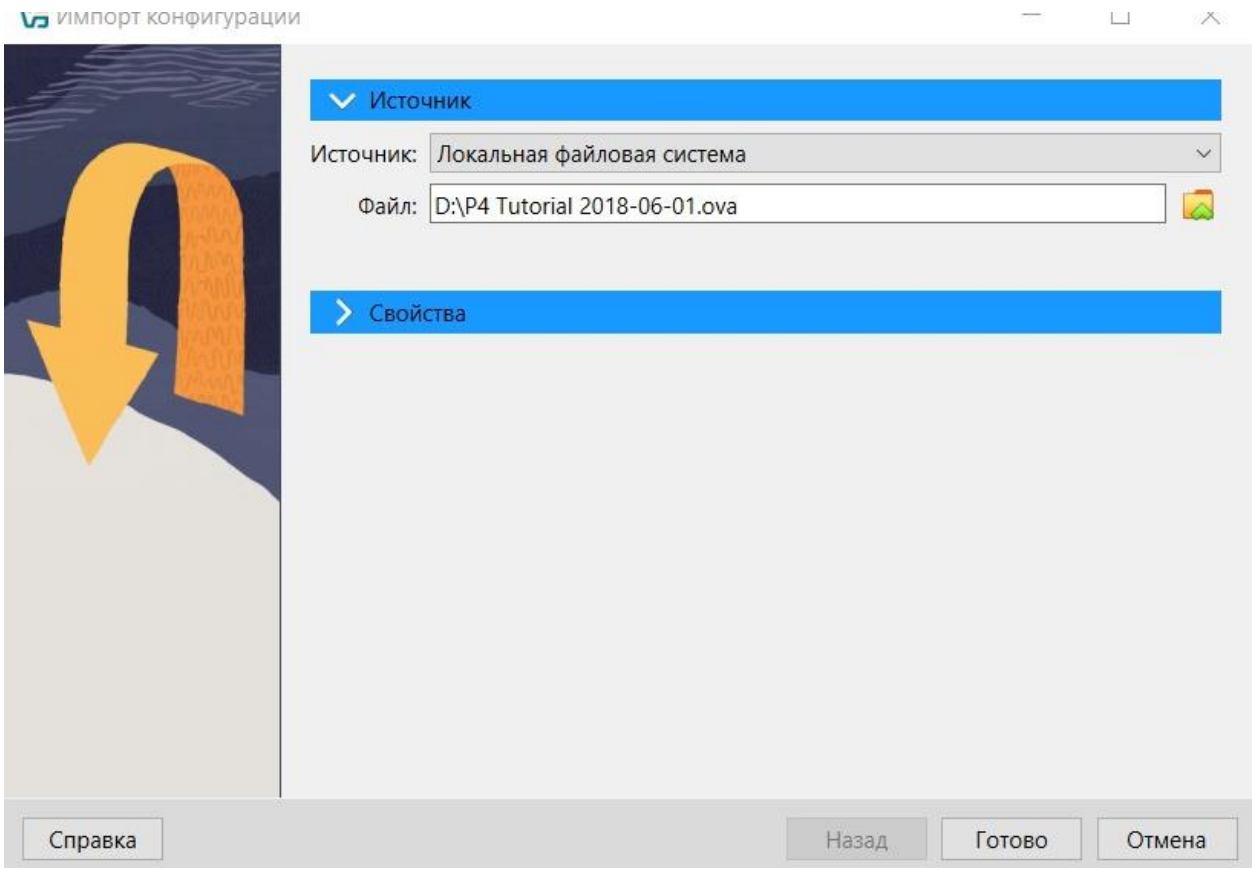
Выполнил: Морозов Матвей
Группа К3320

Цель работы

Изучить синтаксис языка программирования P4 и выполнить 2 задания обучающих задания от Open network foundation для ознакомления на практике с P4

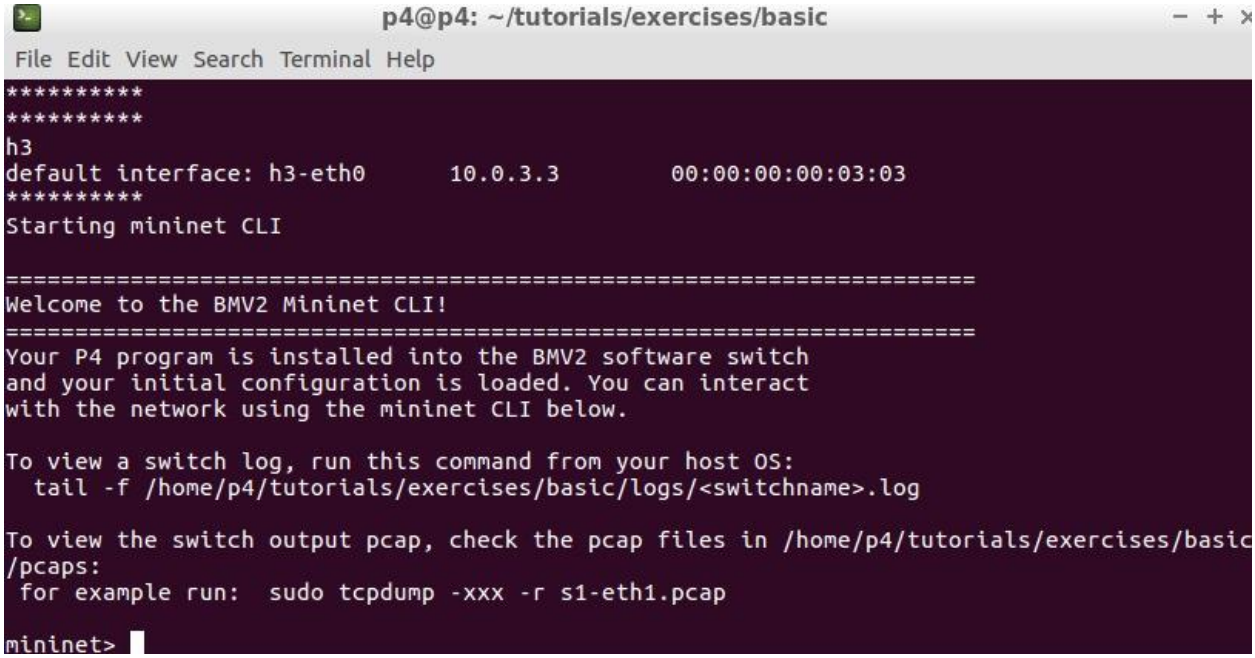
Ход работы

Был скачан образ виртуальной машины P4 Tutorial, далее образ был импортирован через импорт конфигураций в VirtualBox



Реализация базовой переадресации

В полученной виртуальной машине после перехода в директорию tutorials/exercices/basic была выполнена команда make run для поднятия Mininet



Выполнение команды pingall демонстрирует, что коммутаторы отбрасывают все входящие пакеты

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X
h2 -> X X
h3 -> X X
*** Results: 100% dropped (0/6 received)
mininet>
```

Чтобы это исправить, нужно внести изменения в файл конфигурации basic.p4

Добавим логику парсинга для пакетов ethernet и ipv4:

```
state parse_ethernet {
    packet.extract(hdr.ethernet);
    transition select(hdr.ethernet.etherType) {
        TYPE_IPV4 : parse_ipv4;
        default : accept;
```

```
state parse_ipv4 {
    packet.extract(hdr.ipv4);
    transition accept;
```

Проверка

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet>
```

Реализация базового туннелирования

Для реализации базового туннелирования нужно перейти в директорию

tutorials/exercices/basic_tunnel/ и внести изменения в файл basic_tunnel.p4.

Некоторые изменения были уже внесены, например, был добавлен заголовок myTunnel_t, содержащий в себе id протокола и источника

```
...

header myTunnel_t {
    bit<16> proto_id;
    bit<16> dst_id;
}
...
```

В секции PARSER требуется описать логику обработки заголовков myTunnel

```
/*************************************************************************
**********************  P A R S E R  ****************************************
*************************************************************************/
...

state parse_myTunnel {
        packet.extract(hdr.myTunnel);
        transition select(hdr.myTunnel.proto_id) {
            TYPE_IPV4  : parse_ipv4;
            default : accept;
        }
    }
...
```

В секции INGRESS PROCESSING нужно объявить новое действие myTunnel_forward и новую таблицу myTunnel_exact, а также добавить применение таблицы в секции apply

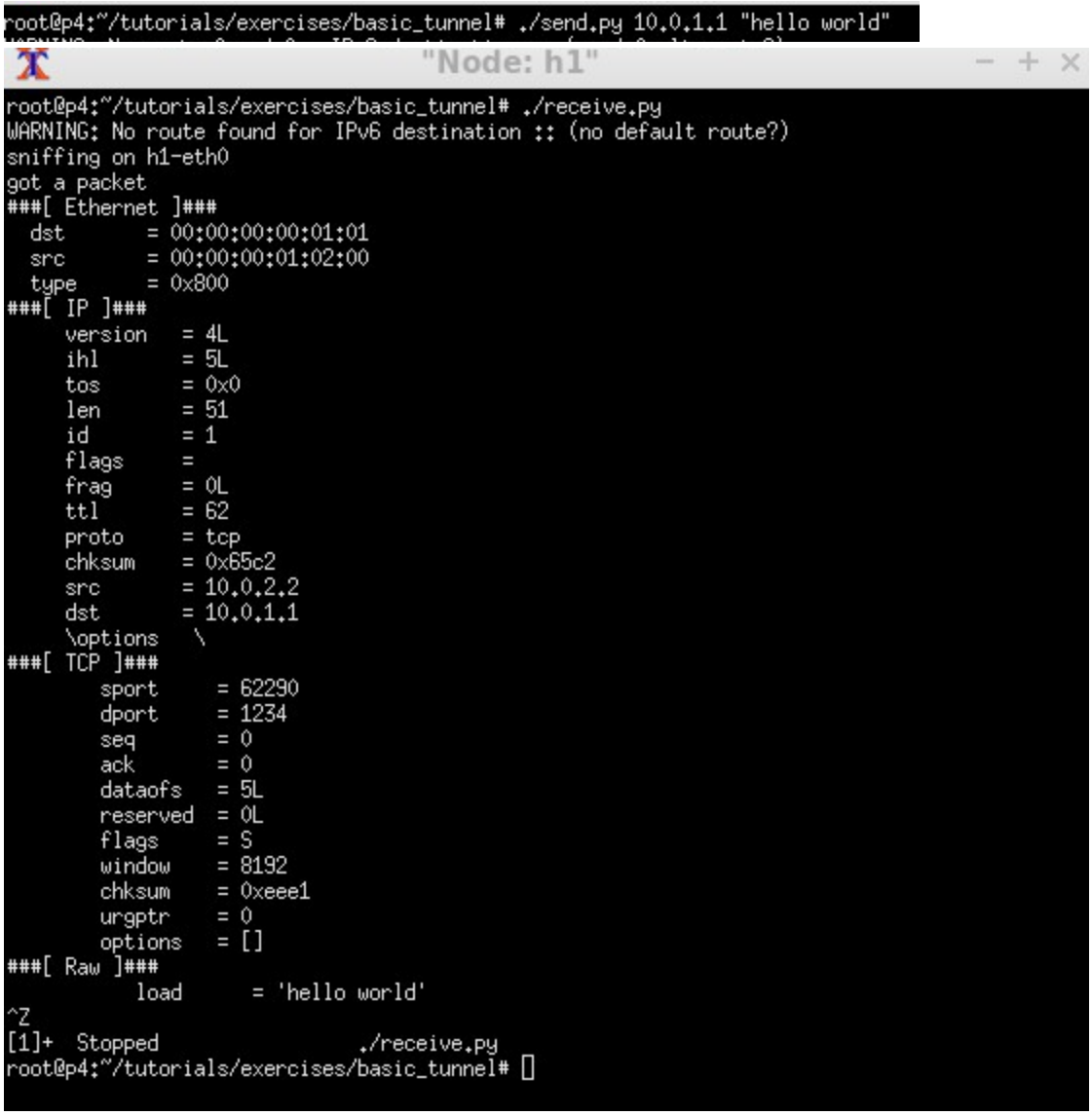
```
/*************************************************************************
*************  I N G R E S S   P R O C E S S I N G   *******************
*************************************************************************/
...

 action myTunnel_forward(egressSpec_t port) {
        standard_metadata.egress_spec = port;
    }

    table myTunnel_exact {
        key = {
            hdr.myTunnel.dst_id: exact;
        }
        actions = {
            myTunnel_forward;
            drop;
            NoAction;
        }
        size = 1024;
        default_action = NoAction();
    }

    apply {
        if (hdr.ipv4.isValid() && !hdr.myTunnel.isValid()) {
            ipv4_lpm.apply();
        }
        if (hdr.myTunnel.isValid()) {
            myTunnel_exact.apply();
```

В секции DEPARSER также нужно добавить логику обработки для исходящих пакетов туннелирования

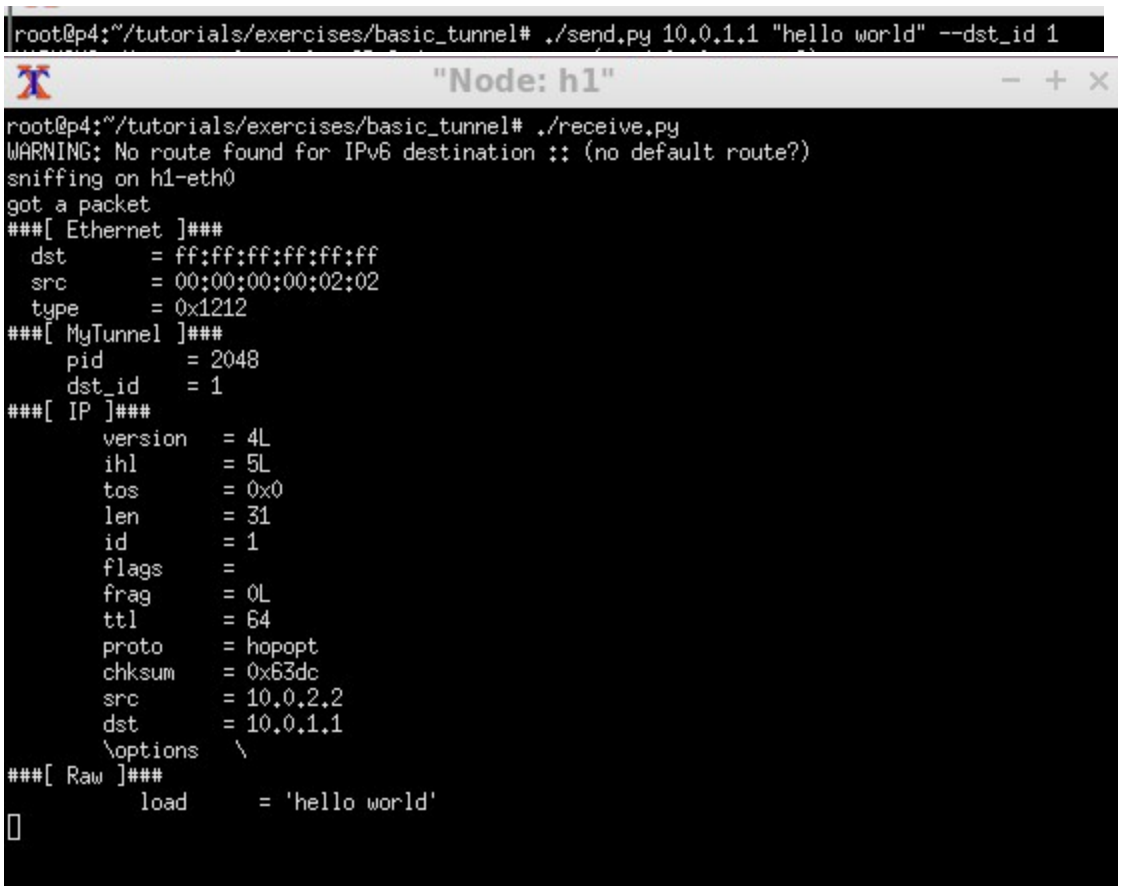
```
/*************************************************************************
**********************  D E P A R S E R  ***************************
*************************************************************************/

control MyDeparser(packet_out packet, in headers hdr) {
    apply {
        packet.emit(hdr.ethernet);
        packet.emit(hdr.myTunnel);
        packet.emit(hdr.ipv4);
    }
}
```

Проверка корректности работы повторно поднятого Mininet:

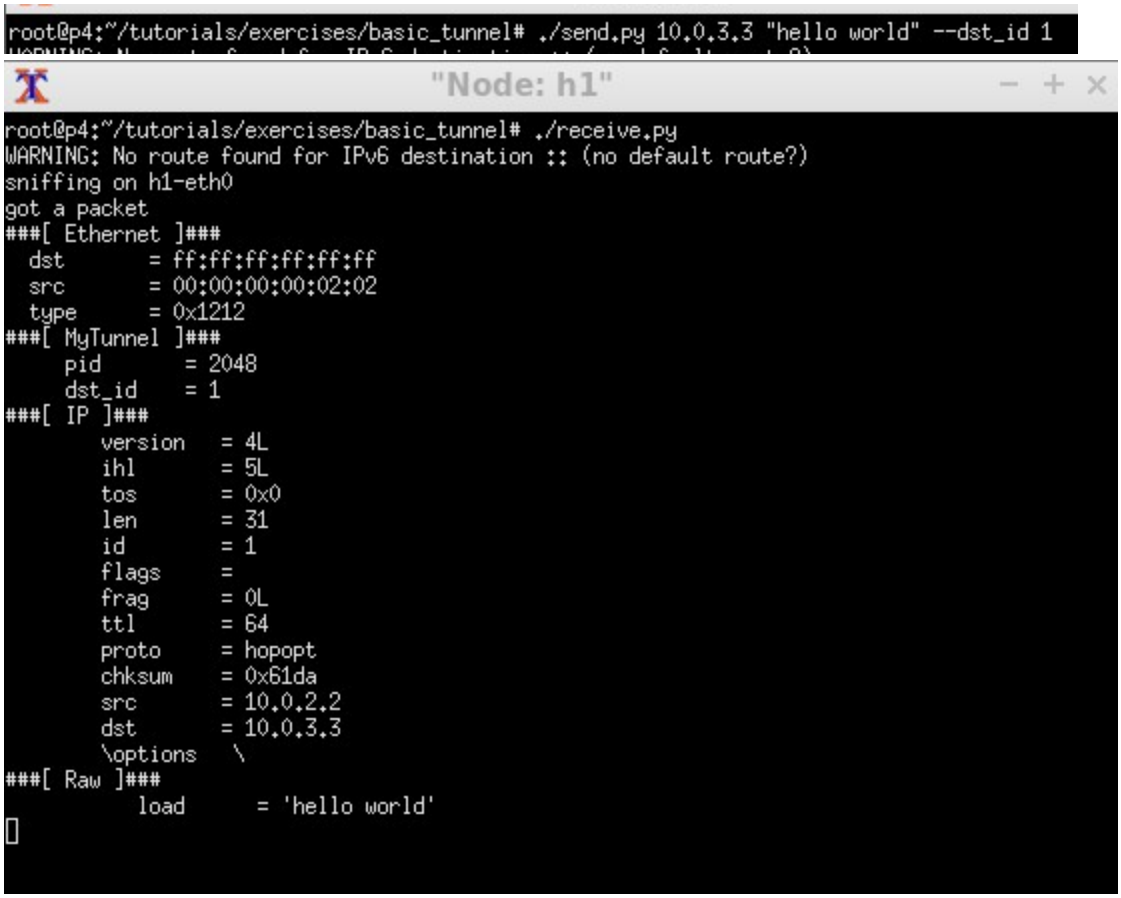
В терминале коммутатора H1 был запущен сервер ./receive.py, после чего в терминале H2 была выполнена команда ./send.py 10.0.1.1 "hello world". Отправка сообщения без туннелирования прошла успешно



Для проверки работы туннелирования на H2 была выполнена команда ./send.py 10.0.1.1 "hello world" --dst_ip 1. У полученного пакета есть заголовок MyTunnel



При выполнении команды ./send.py 10.0.3.3 "hello world" --dst_ip 1 сообщение будет получено первым коммутатором, несмотря на то, что указан IP-адрес третьего. Это объясняется тем, что при наличии заголовка MyTunnel в пакете именно он используется для маршрутизации, а не IP-заголовок



Полученные файлы basic.p4 и basic_tunnel.p4

basic.p4

```
/* -*- P4_16 -*- */
#include <core.p4>
#include <v1model.p4>

const bit<16> TYPE_IPV4 = 0x800;

/*************************************************************************
*********************** H E A D E R S  ***********************************
*************************************************************************/

typedef bit<9>  egressSpec_t;
typedef bit<48> macAddr_t;
typedef bit<32> ip4Addr_t;

header ethernet_t {
  macAddr_t dstAddr;
  macAddr_t srcAddr;
  bit<16>   etherType;
}

header ipv4_t {
  bit<4>   version;
```

```
    bit<4>   ihl;
    bit<8>   diffserv;
    bit<16>  totalLen;
    bit<16>  identification;
    bit<3>   flags;
    bit<13>  fragOffset;
    bit<8>   ttl;
    bit<8>   protocol;
    bit<16>  hdrChecksum;
    ip4Addr_t srcAddr;
    ip4Addr_t dstAddr;
}

struct metadata {
    /* empty */
}

struct headers {
    ethernet_t   ethernet;
    ipv4_t       ipv4;
}

/*************************************************************************
**********************  P A R S E R  ***********************************
*************************************************************************/

parser MyParser(packet_in packet,
                out headers hdr,
                inout metadata meta,
                inout standard_metadata_t standard_metadata) {

    state start {
        /* TODO: add parser logic */
        transition parse_ethernet;
    }

    state parse_ethernet {
        packet.extract(hdr.ethernet);
        transition select(hdr.ethernet.etherType) {
            TYPE_IPV4: parse_ipv4;
            default: accept;
        }
    }

    state parse_ipv4 {
        packet.extract(hdr.ipv4);
        transition accept;
    }
}


/*************************************************************************
************   C H E C K S U M    V E R I F I C A T I O N   *************
*************************************************************************/

control MyVerifyChecksum(inout headers hdr, inout metadata meta) {
    apply { }
}


/*************************************************************************
```

```
************** I N G R E S S  P R O C E S S I N G  *******************
*************************************************************************/

control MyIngress(inout headers hdr,
                  inout metadata meta,
                  inout standard_metadata_t standard_metadata) {
  action drop() {
    mark_to_drop();
  }

  action ipv4_forward(macAddr_t dstAddr, egressSpec_t port) {
    standard_metadata.egress_spec = port;
    hdr.ethernet.srcAddr = hdr.ethernet.dstAddr;
    hdr.ethernet.dstAddr = dstAddr;
    hdr.ipv4.ttl = hdr.ipv4.ttl - 1;
  }

  table ipv4_lpm {
    key = {
      hdr.ipv4.dstAddr: lpm;
    }
    actions = {
      ipv4_forward;
      drop;
      NoAction;
    }
    size = 1024;
    default_action = NoAction();
  }

  apply {
    if (hdr.ipv4.isValid()) {
      ipv4_lpm.apply();
    }
  }
}

/*************************************************************************
***************** E G R E S S  P R O C E S S I N G  *******************
*************************************************************************/

control MyEgress(inout headers hdr,
                 inout metadata meta,
                 inout standard_metadata_t standard_metadata) {
  apply { }
}
/*************************************************************************
*************** C H E C K S U M  C O M P U T A T I O N  **************
*************************************************************************/

control MyComputeChecksum(inout headers hdr, inout metadata meta) {
   apply {
    update_checksum(
      hdr.ipv4.isValid(),
      { hdr.ipv4.version,
        hdr.ipv4.ihl,
        hdr.ipv4.diffserv,
        hdr.ipv4.totalLen,
        hdr.ipv4.identification,
        hdr.ipv4.flags,
        hdr.ipv4.fragOffset,
```

```
          hdr.ipv4.ttl,
          hdr.ipv4.protocol,
          hdr.ipv4.srcAddr,
          hdr.ipv4.dstAddr },
        hdr.ipv4.hdrChecksum,
        HashAlgorithm.csum16);
  }
}


/*************************************************************************
*********************** D E P A R S E R  ***************************
*************************************************************************/

control MyDeparser(packet_out packet, in headers hdr) {
  apply {
    packet.emit(hdr.ethernet);
    packet.emit(hdr.ipv4);
  }
}

/*************************************************************************
*********************** S W I T C H  *******************************
*************************************************************************/

V1Switch(
MyParser(),
MyVerifyChecksum(),
MyIngress(),
MyEgress(),
MyComputeChecksum(),
MyDeparser()
) main;
```

basic_tunnel.p4

```
/* -*- P4_16 -*- */
#include <core.p4>
#include <v1model.p4>

// NOTE: new type added here
const bit<16> TYPE_MYTUNNEL  = 0x1212;
const bit<16> TYPE_IPV4  = 0x800;

/*************************************************************************
*********************** H E A D E R S  ***********************************
*************************************************************************/

typedef bit<9>  egressSpec_t;
typedef bit<48> macAddr_t;
typedef bit<32> ip4Addr_t;

header ethernet_t {
   macAddr_t dstAddr;
   macAddr_t srcAddr;
   bit<16>   etherType;
}

// NOTE: added new header type
header myTunnel_t {
   bit<16> proto_id;
```

```
    bit<16> dst_id;
}

header ipv4_t {
    bit<4>    version;
    bit<4>    ihl;
    bit<8>    diffserv;
    bit<16>   totalLen;
    bit<16>   identification;
    bit<3>    flags;
    bit<13>   fragOffset;
    bit<8>    ttl;
    bit<8>    protocol;
    bit<16>   hdrChecksum;
    ip4Addr_t srcAddr;
    ip4Addr_t dstAddr;
}

struct metadata {
    /* empty */
}

// NOTE: Added new header type to headers struct
struct headers {
    ethernet_t   ethernet;
    myTunnel_t   myTunnel;
    ipv4_t       ipv4;
}

/*************************************************************************
**********************  P A R S E R  ***********************************
*************************************************************************/

// TODO: Update the parser to parse the myTunnel header as well
parser MyParser(packet_in packet,
                out headers hdr,
                inout metadata meta,
                inout standard_metadata_t standard_metadata) {

    state start {
        transition parse_ethernet;
    }

    state parse_ethernet {
        packet.extract(hdr.ethernet);
        transition select(hdr.ethernet.etherType) {
            TYPE_IPV4  : parse_ipv4;
            TYPE_MYTUNNEL:  parse_myTunnel;
            default: accept;
        }
    }

    state parse_myTunnel {
        packet.extract(hdr.myTunnel);
        transition select(hdr.myTunnel.proto_id) {
            TYPE_IPV4:  parse_ipv4;
            default: accept;
        }
    }

    state parse_ipv4 {
        packet.extract(hdr.ipv4);
        transition accept;
    }
```

```
}

/*************************************************************************
************ C H E C K S U M  V E R I F I C A T I O N *************
*************************************************************************/

control MyVerifyChecksum(inout headers hdr, inout metadata meta) {
   apply {  }
}


/*************************************************************************
************** I N G R E S S  P R O C E S S I N G *******************
*************************************************************************/

control MyIngress(inout headers hdr,
                  inout metadata meta,
                  inout standard_metadata_t standard_metadata) {
   action drop() {
      mark_to_drop();
   }

   action ipv4_forward(macAddr_t dstAddr, egressSpec_t port) {
      standard_metadata.egress_spec = port;
      hdr.ethernet.srcAddr = hdr.ethernet.dstAddr;
      hdr.ethernet.dstAddr = dstAddr;
      hdr.ipv4.ttl = hdr.ipv4.ttl - 1;
   }

   table ipv4_lpm {
      key = {
         hdr.ipv4.dstAddr: lpm;
      }
      actions = {
         ipv4_forward;
         drop;
         NoAction;
      }
      size = 1024;
      default_action = drop();
   }

   action myTunnel_forward(egressSpec_t port) {
      standard_metadata.egress_spec = port;
   }


   table myTunnel_exact {
      key = {
         hdr.myTunnel.dst_id: exact;
      }
      actions = {
         myTunnel_forward;
         drop;
         NoAction;
      }
      size = 1024;
      default_action = NoAction();
   }


   apply {
      if (hdr.ipv4.isValid() && !hdr.myTunnel.isValid()) {
         ipv4_lpm.apply();
```

```
        }
        if (hdr.myTunnel.isValid()) {
            myTunnel_exact.apply();
        }
    }
}

/*************************************************************************
**************** E G R E S S  P R O C E S S I N G  *******************
*************************************************************************/

control MyEgress(inout headers hdr,
                 inout metadata meta,
                 inout standard_metadata_t standard_metadata) {
    apply {  }
}

/*************************************************************************
************* C H E C K S U M  C O M P U T A T I O N  **************
*************************************************************************/

control MyComputeChecksum(inout headers  hdr, inout metadata meta) {
     apply {
        update_checksum(
            hdr.ipv4.isValid(),
            { hdr.ipv4.version,
              hdr.ipv4.ihl,
              hdr.ipv4.diffserv,
              hdr.ipv4.totalLen,
              hdr.ipv4.identification,
              hdr.ipv4.flags,
              hdr.ipv4.fragOffset,
              hdr.ipv4.ttl,
              hdr.ipv4.protocol,
              hdr.ipv4.srcAddr,
              hdr.ipv4.dstAddr },
            hdr.ipv4.hdrChecksum,
            HashAlgorithm.csum16);
    }
}

/*************************************************************************
*********************** D E P A R S E R  ****************************
*************************************************************************/

control MyDeparser(packet_out packet, in headers hdr) {
    apply {
        packet.emit(hdr.ethernet);
        packet.emit(hdr.myTunnel);
        packet.emit(hdr.ipv4);
    }
}

/*************************************************************************
***********************  S W I T C H  *******************************
*************************************************************************/

V1Switch(
MyParser(),
MyVerifyChecksum(),
MyIngress(),
MyEgress(),
MyComputeChecksum(),
MyDeparser()
) main;
```