UofSC South Carolina

Parser

Jorge Crichigno

College of Engineering and Computing, University of South Carolina

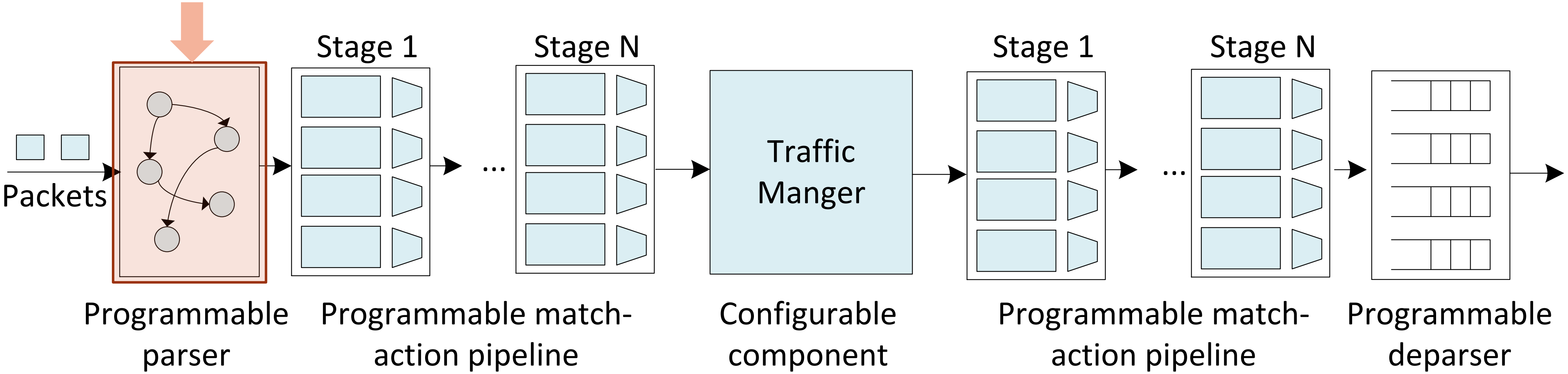
A Hands-on Tutorial on P4 Programmable Data Planes

Monday March 6, 2023

Parser

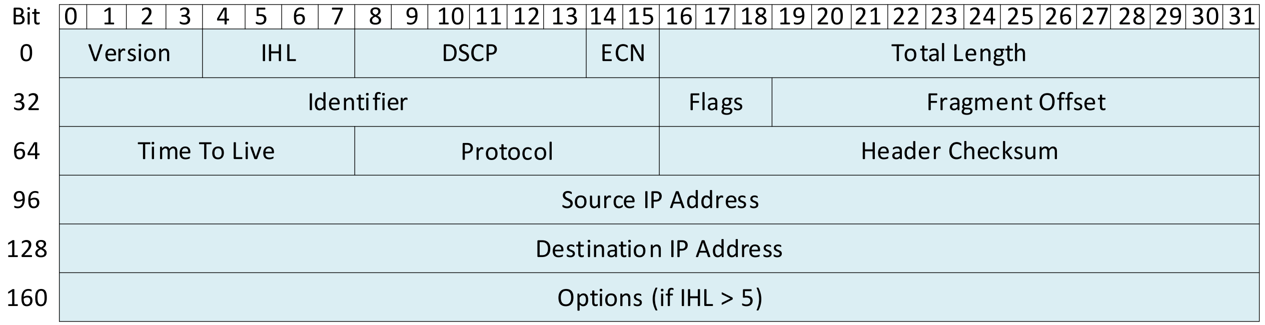
Programmable Parser

- The parser enables parsing arbitrary headers with a finite state machine
- The state machine follows the order of the headers within the packets
- The packet is split into the defined headers and the remaining is treated as the payload

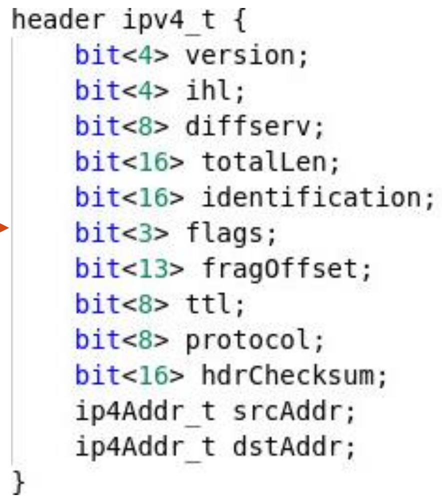


Packet Headers

- The packet headers are specified by the programmer
- The programmer has the flexibility of defining custom/non-standardized headers
- Such capability is not available in non-programmable devices



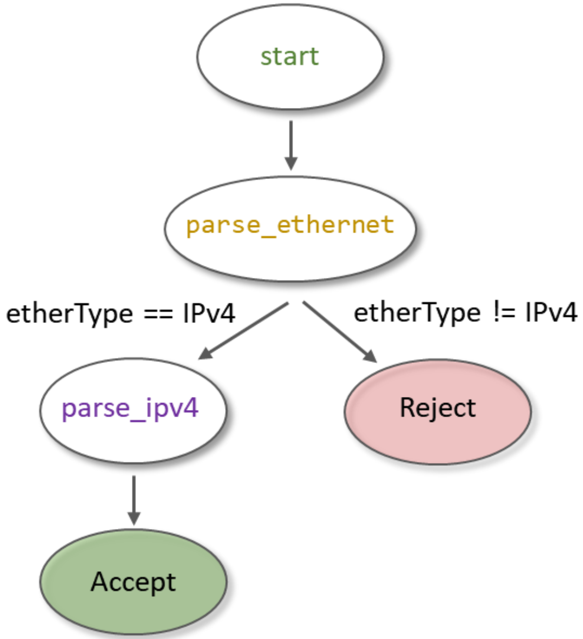
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Version				IHL				DSCP						ECN		Total Length															
32	Identifier																Flags			Fragment Offset												
64	Time To Live								Protocol								Header Checksum															
96	Source IP Address																															
128	Destination IP Address																															
160	Options (if IHL > 5)																															

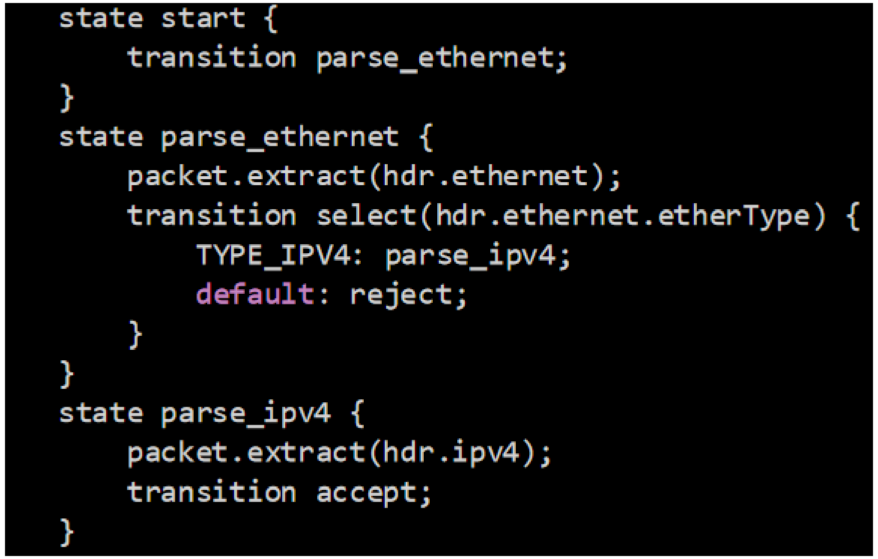


```
header ipv4_t {
    bit<4> version;
    bit<4> ihl;
    bit<8> diffserv;
    bit<16> totalLen;
    bit<16> identification;
    bit<3> flags;
    bit<13> fragOffset;
    bit<8> ttl;
    bit<8> protocol;
    bit<16> hdrChecksum;
    ip4Addr_t srcAddr;
    ip4Addr_t dstAddr;
}
```

Programmable Parser

- Every parser has three predefined states: start, accept, and reject
- Other states may be defined by the programmer
- In each state, the parser executes statements and then transitions to another state



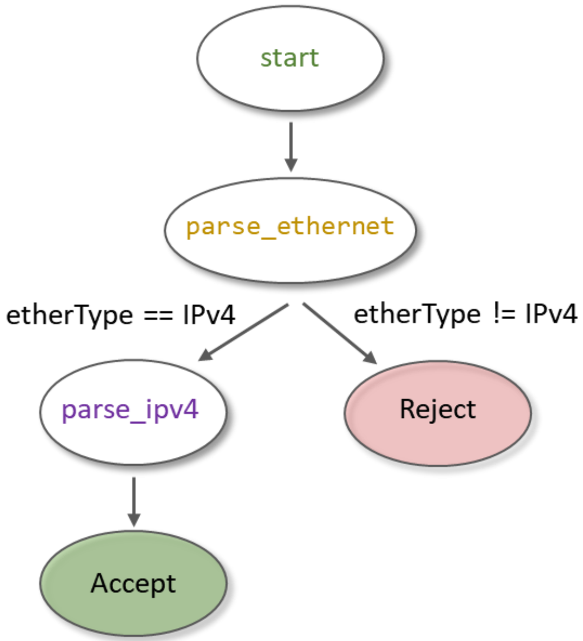


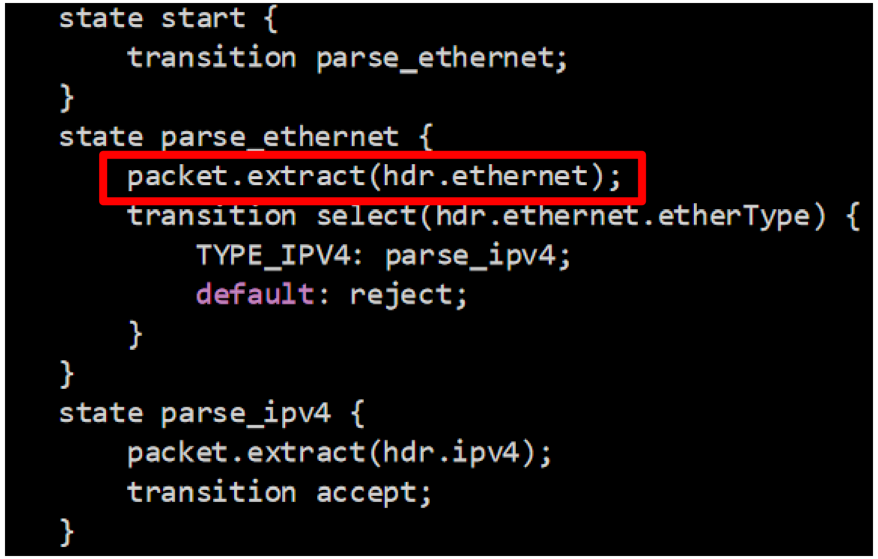
```
state start {
    transition parse_ethernet;
}
state parse_ethernet {
    packet.extract(hdr.ethernet);
    transition select(hdr.ethernet.etherType) {
        TYPE_IPV4: parse_ipv4;
        default: reject;
    }
}
state parse_ipv4 {
    packet.extract(hdr.ipv4);
    transition accept;
}
```

`packet` is an input parameter; `hdr` is an output parameter

Programmable Parser

- $P4_{16}$ has an extract method that can be used to “fill in” the fields of a header from the “raw” packet



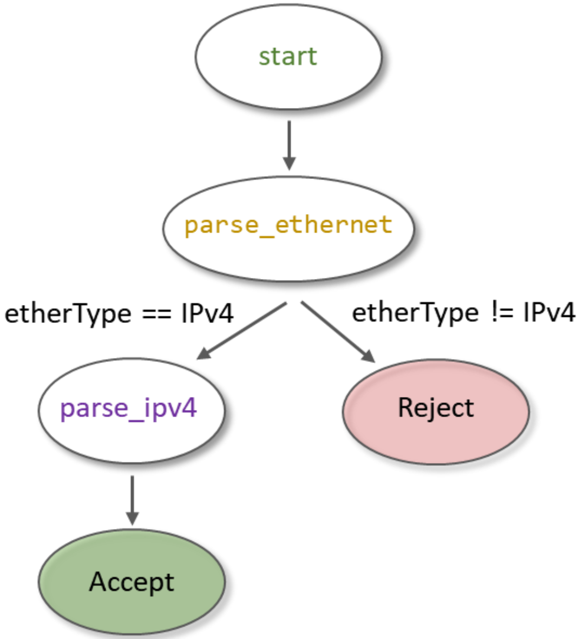


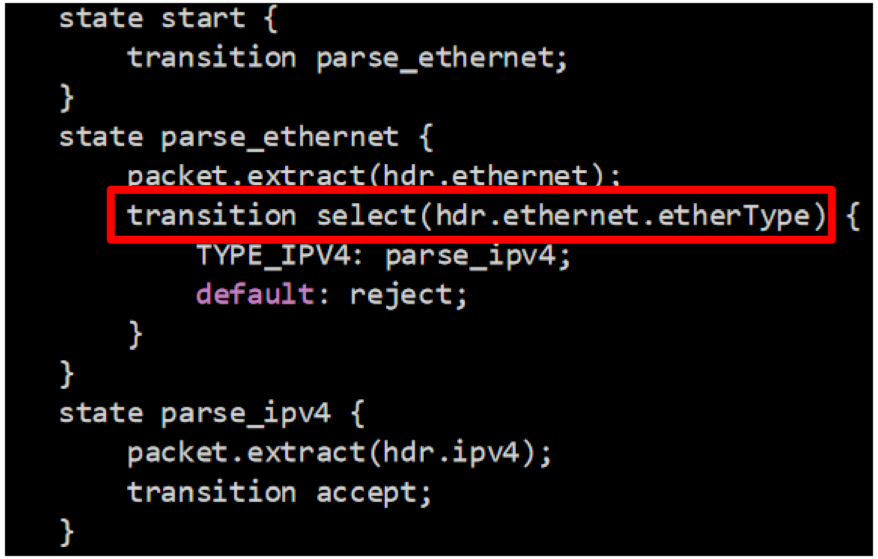
```
state start {
    transition parse_ethernet;
}
state parse_ethernet {
    packet.extract(hdr.ethernet);
    transition select(hdr.ethernet.etherType) {
        TYPE_IPV4: parse_ipv4;
        default: reject;
    }
}
state parse_ipv4 {
    packet.extract(hdr.ipv4);
    transition accept;
}
```

`packet` is an input parameter; `hdr` is an output parameter

Programmable Parser

- P4$_{16}$ has a select statement that can be used to branch in a parser



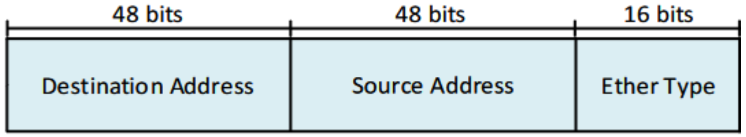


```
state start {
    transition parse_ethernet;
}
state parse_ethernet {
    packet.extract(hdr.ethernet);
    transition select(hdr.ethernet.etherType) {
        TYPE_IPV4: parse_ipv4;
        default: reject;
    }
}
state parse_ipv4 {
    packet.extract(hdr.ipv4);
    transition accept;
}
```

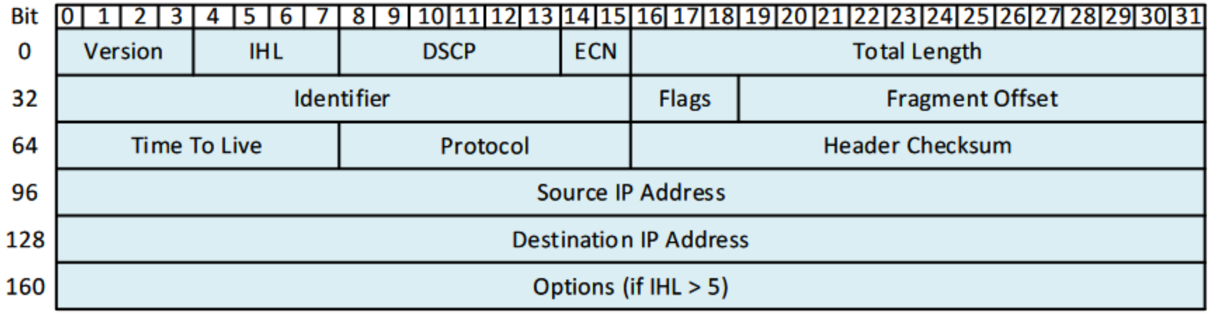
`packet` is an input parameter; `hdr` is an output parameter

Headers Format

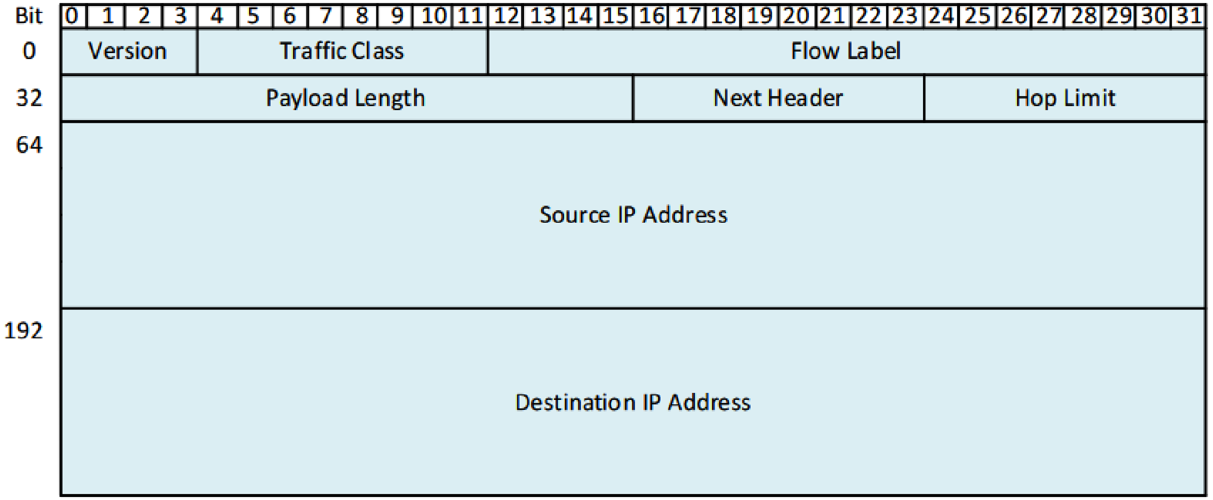
- Ethernet header:



- IPv4 header:

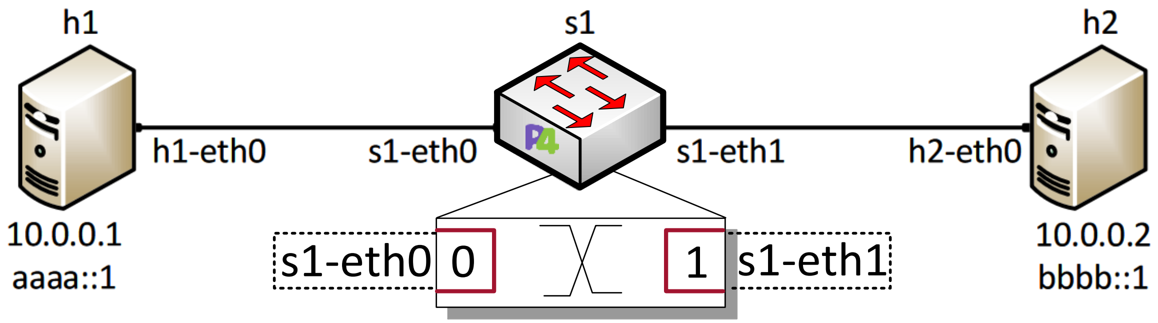


- IPv6 header:



Lab 4 Topology and Objectives

- The topology consists of two hosts: h1 and h2; one P4 switch: s1
- The objectives are:
  - Defining the headers for Ethernet, IPv4 and IPv6
  - Implementing the parser
  - Testing and verifying the switch behavior when IPv4 and IPv6 packets are received


h1
s1
h2
h1-eth0
s1-eth0
s1-eth1
h2-eth0
10.0.0.1
aaaa::1
s1-eth0
0
1
s1-eth1
10.0.0.2
bbbb::1