Lecture 10: Programmable Network Hardware

CS 234 / NetSys 210: Advanced Computer Networks

Sangeetha Abdu Jyothi

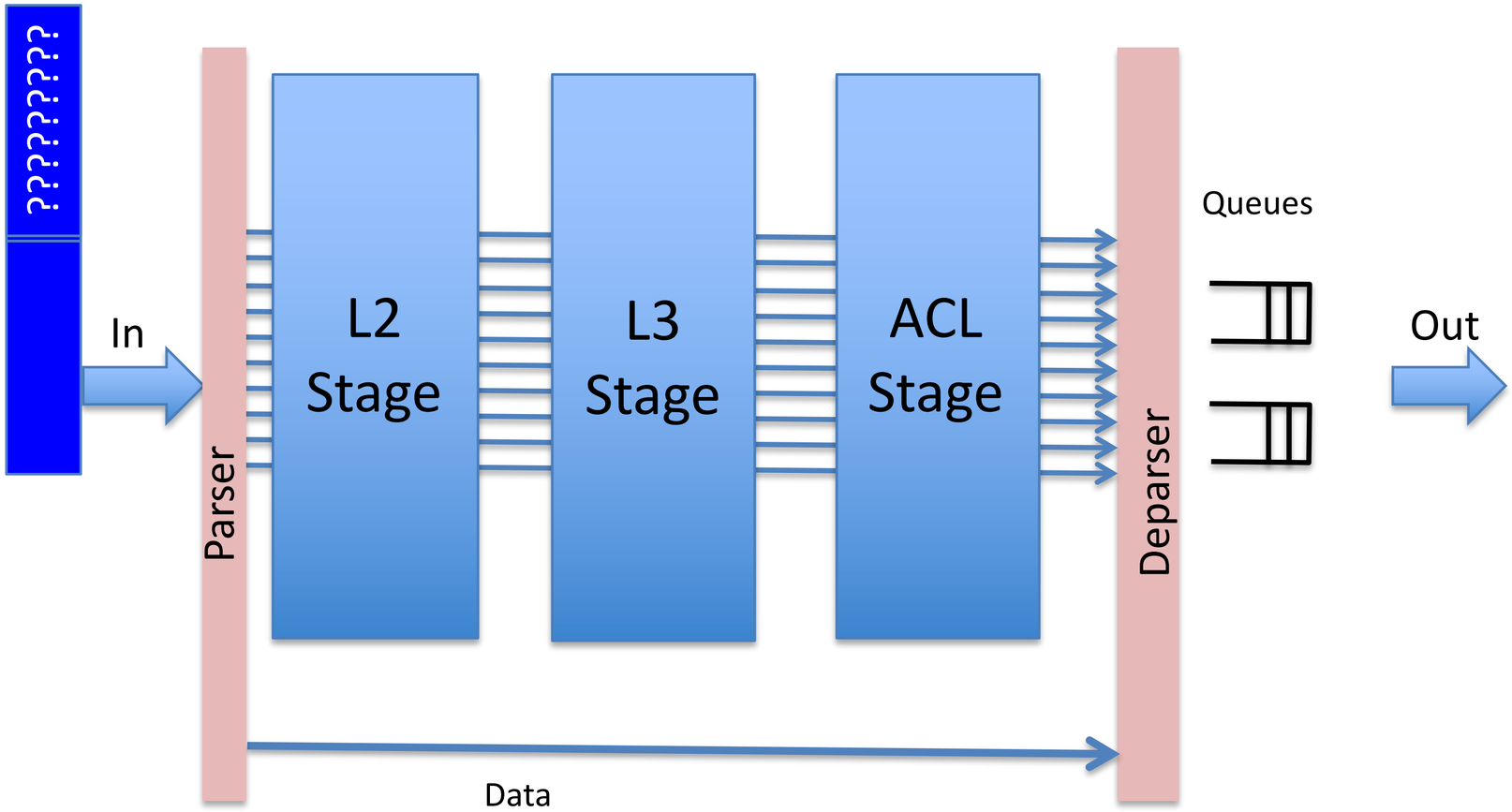
UCIrvine

This lecture uses material from Pat Bosshart's SIGCOMM'13 talk and Radhika Mittal's CS598HPN

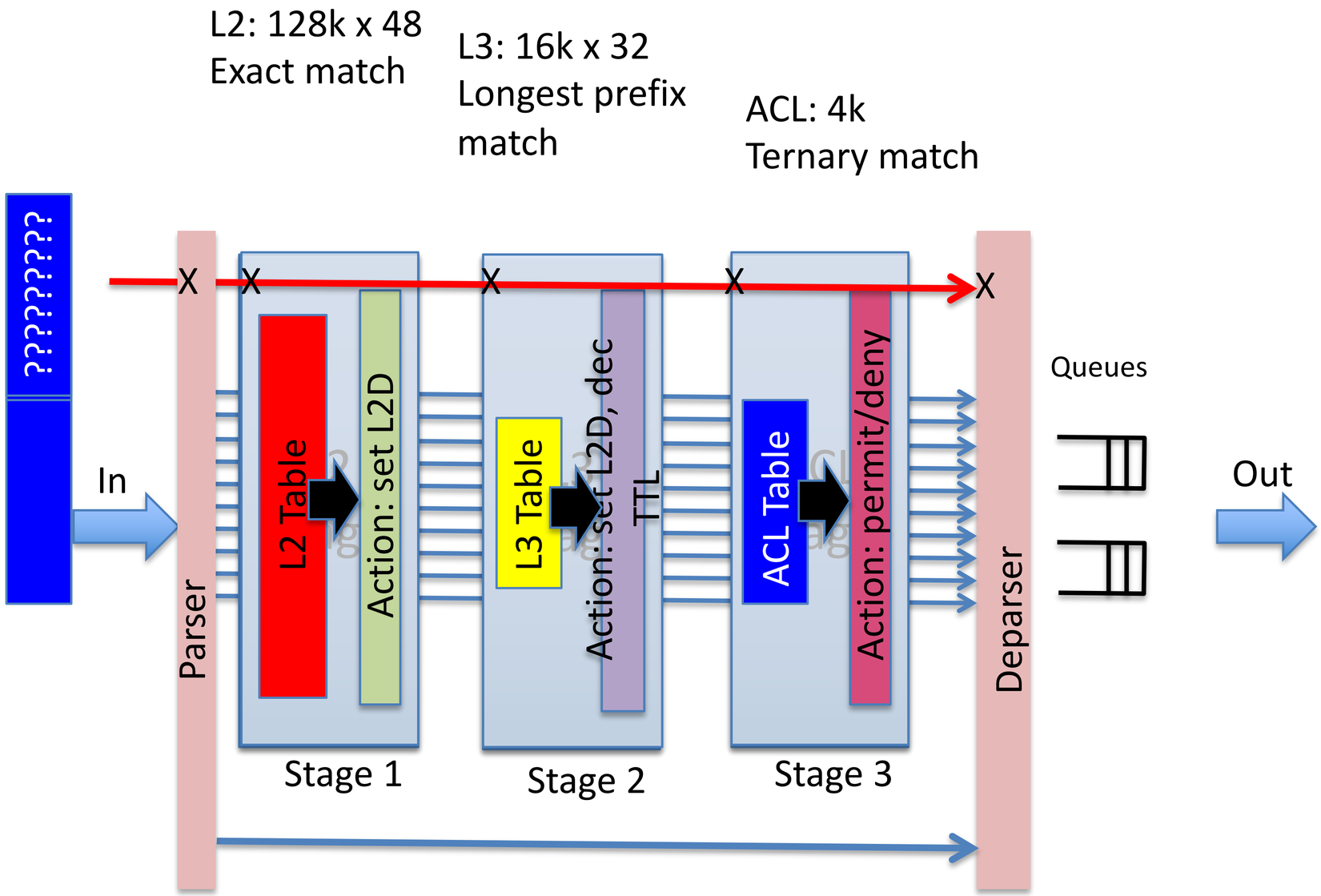
Conventional SDN

- Very flexible control plane in software
- Interacts with dataplane through OpenFlow
- Dataplane flexibility limited by
    - what OpenFlow supports
    - what the underlying hardware can support

Fixed Function Switch



Fixed Function Switch



What if you need flexibility?

- Trade one memory size for another
- Add a new table
- Add a new header field
- Add a different action

Forwarding Metamorphosis: Fast Programmable Match-Action Processing in Hardware for SDN [SIGCOMM'13]

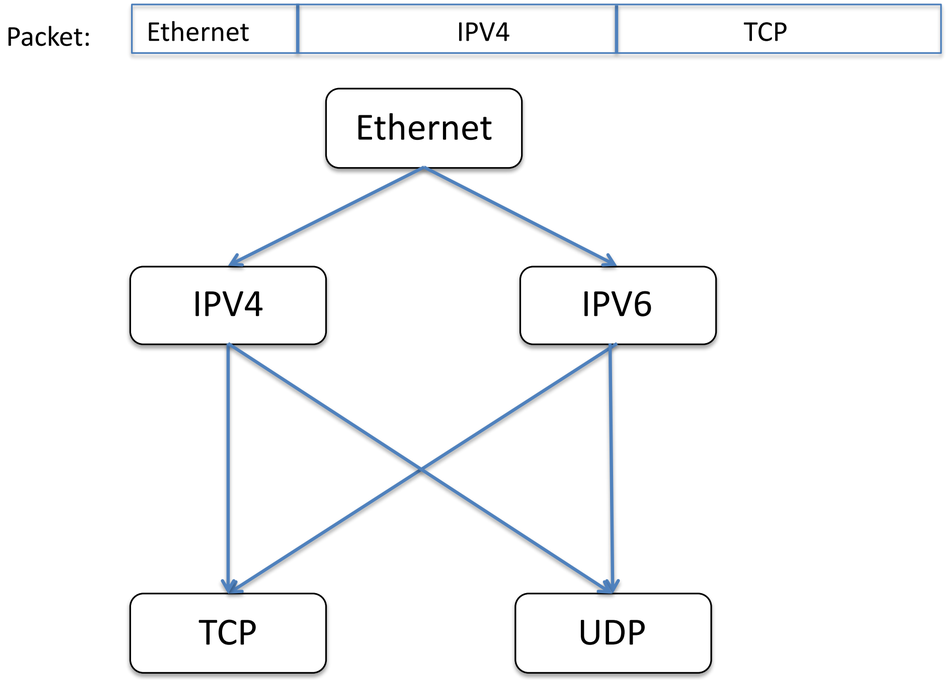
RMT Switch Model

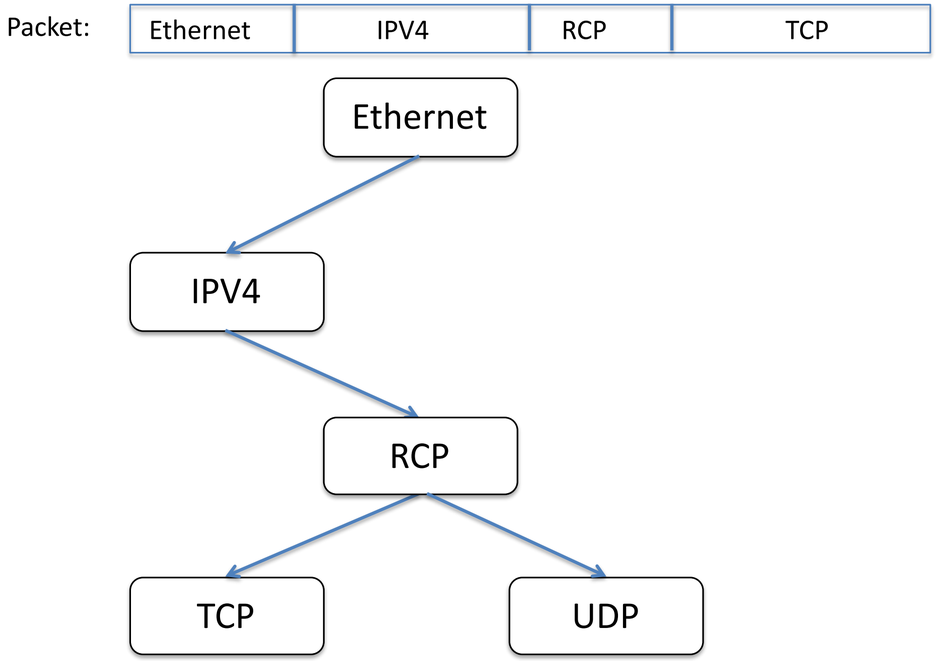
- Reconfigurable Match Tables model
- Enables flexibility through
  - Programmable parsing: support arbitrary header fields
  - Ability to configure number, topology, width, and depths of match-tables
  - Programmable actions: allow a flexible set of actions (including arbitrary packet modifications)

RMT Abstract Model

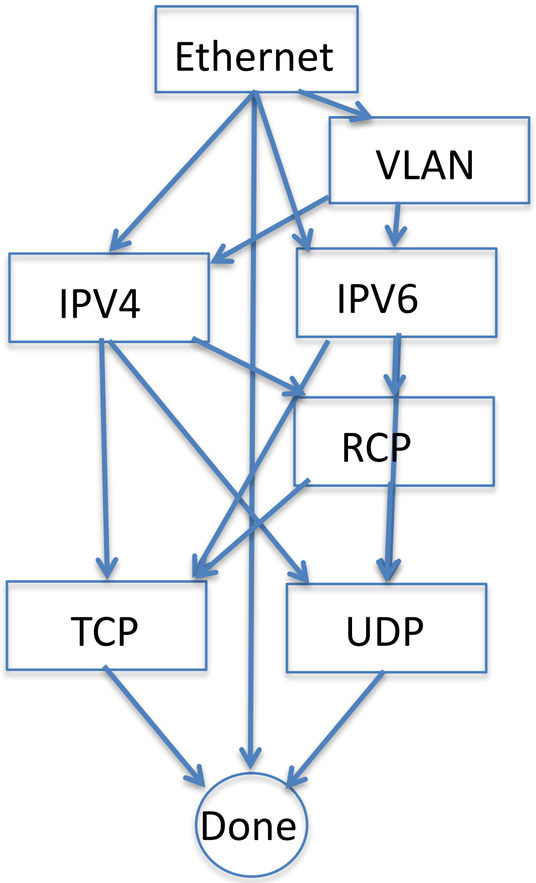
- Parse graph
    - expresses permissible header sequences
- Table flow graph
    - Expresses the set of match tables and the control flow between them

Parse Graphs





Parse Graph to Table Graph



Parse Graph

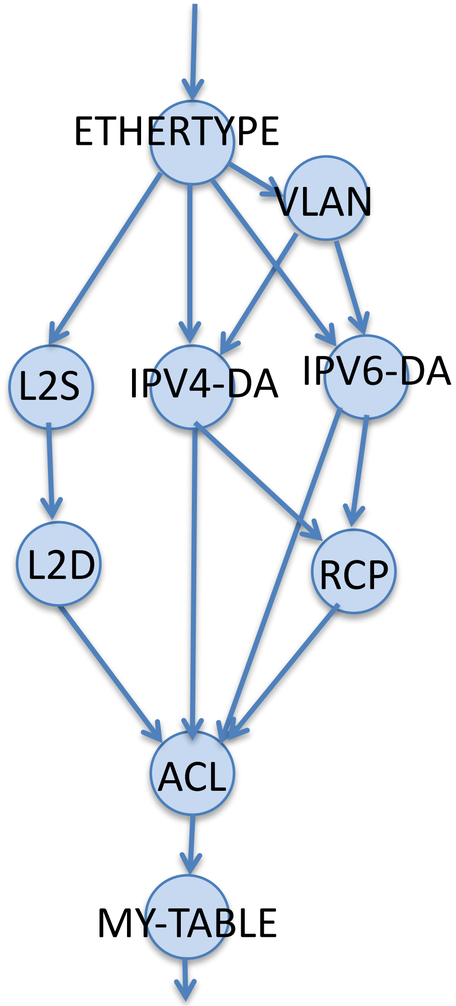
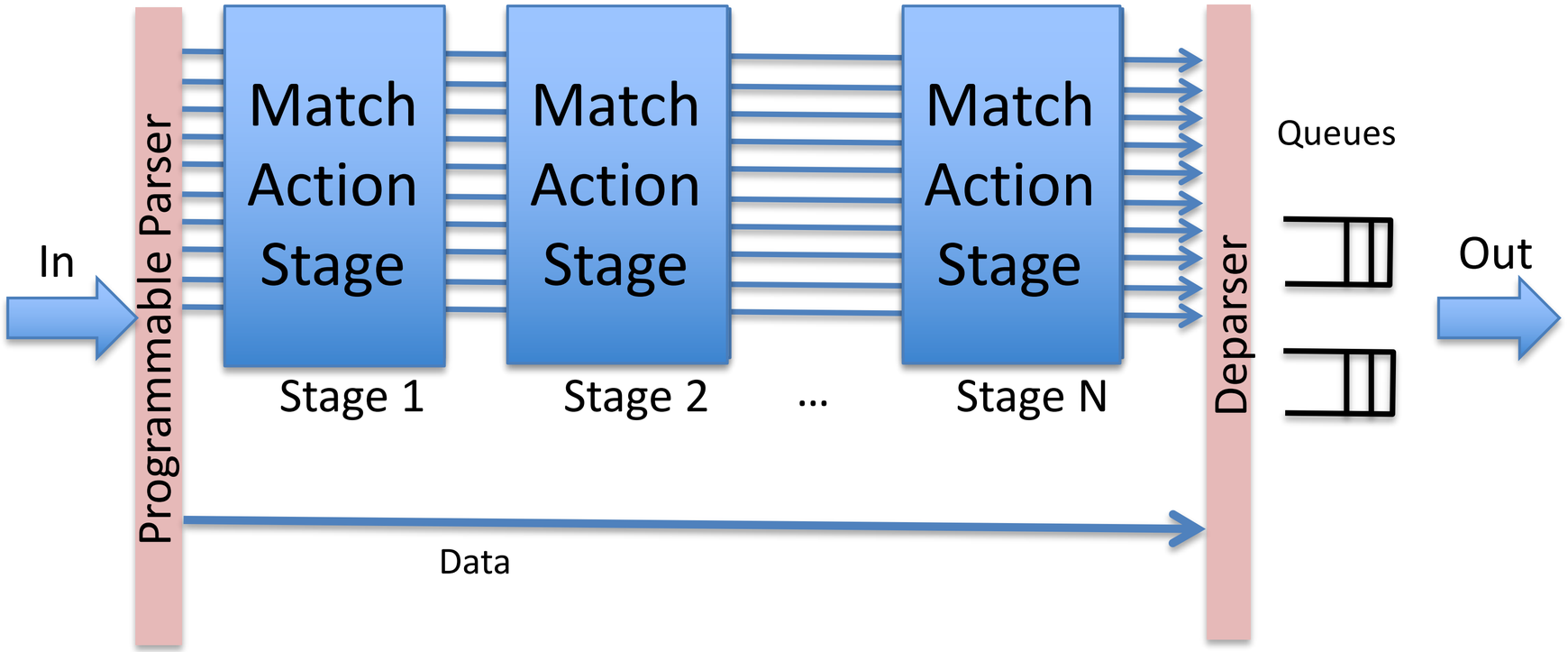


Table Graph

Match-Action Forwarding Model



Programmable Network Hardware Trade-Offs

- Flexibility
  - More flexible than conventional switch hardware
  - Less flexible than software switches
- Power and cost requirements
  - Slightly higher power and cost requirements than conventional switch hardware
  - Significantly lower than software switches

PISA: Protocol Independent Switch Architecture

- RMT
    - Programmable parsers
    - Reconfigurable match-action tables
- Intel FlexPipe
- Cavium Xpliant

What is missing?

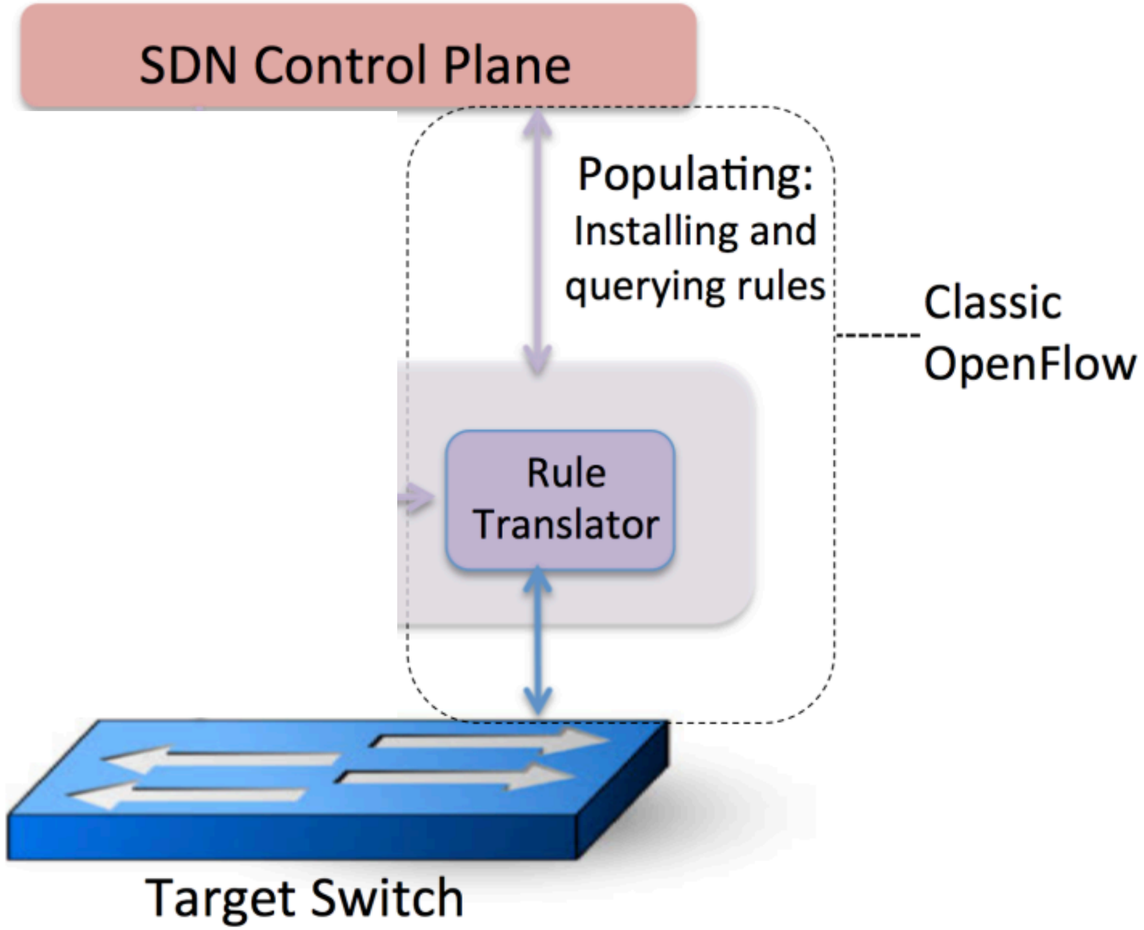
An interface to program the switches

P4: Programming Protocol-Independent Packet Processors

P4 Goals

- Protocol independence
  - Switches are not tied to specific packet formats
- Reconfigurability
  - Controller can redefine packet parsing and processing in the field
- Target Independence
  - User program need not be tied to a specific hardware
  - Compiler's job to do the mapping

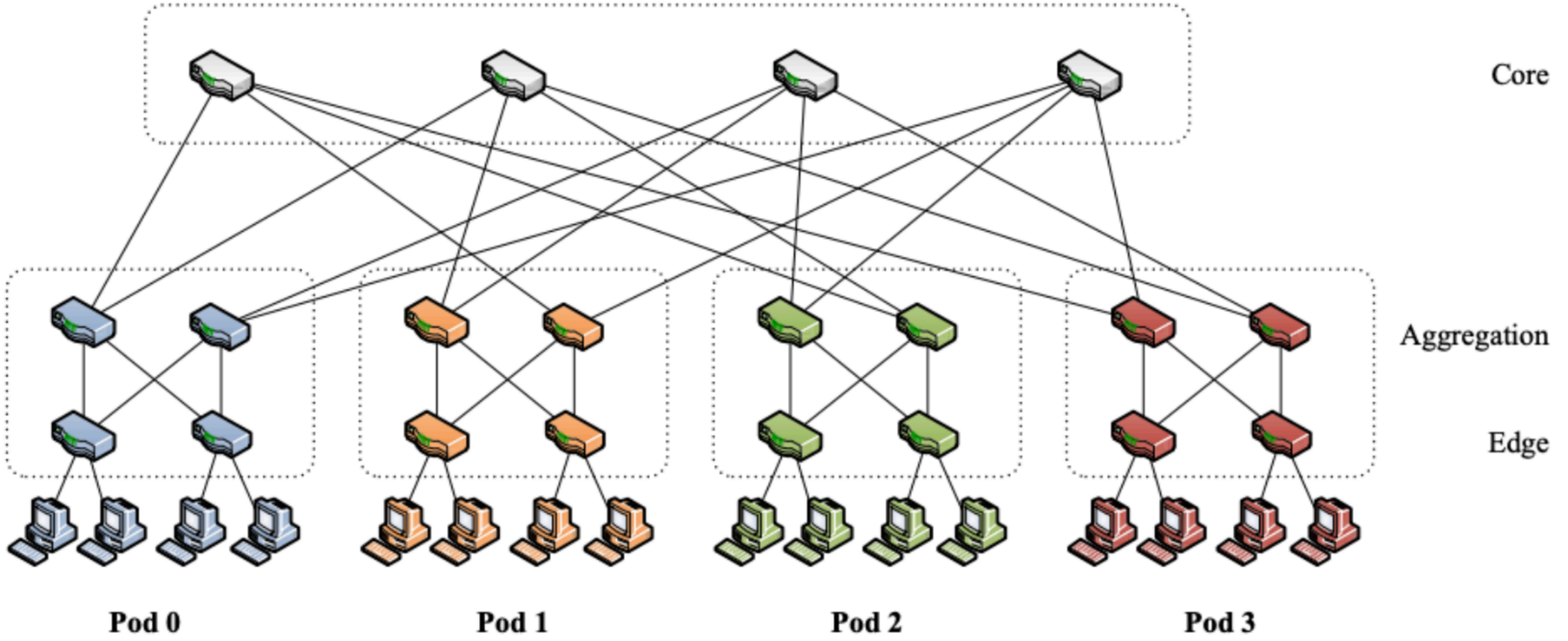
P4 vs. OpenFlow



Components of a P4 program

- Header definitions
- Parser definition
- Tables: what fields to match on, and which action to execute
- Action definition

Example



Example

```
header ethernet {
    fields {
        dst_addr : 48; // width in bits
        src_addr : 48;
        ethertype : 16;
    }
}
```

```
header vlan {
    fields {
        pcp : 3;
        cfi : 1;
        vid : 12;
        ethertype : 16;
    }
}
```

```
header mTag {
    fields {
        up1 : 8;
        up2 : 8;
        down1 : 8;
        down2 : 8;
        ethertype : 16;
    }
}
```

Parsers

```
parser start {
    ethernet;
}

parser ethernet {
    switch(ethertype) {
        case 0x8100: vlan;
        case 0x9100: vlan;
        case  0x800: ipv4;
        // Other cases
    }
}
```

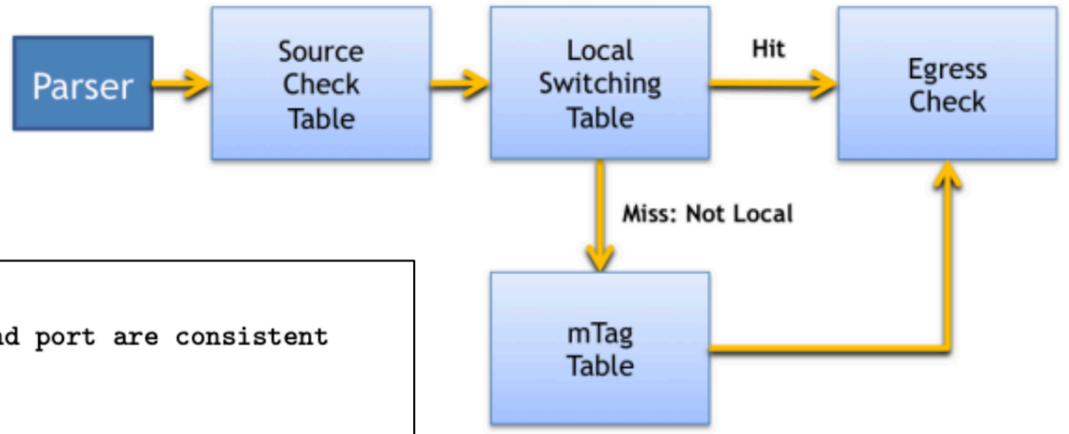
```
parser vlan {
    switch(ethertype) {
        case 0xaaaa: mTag;
        case  0x800: ipv4;
        // Other cases
    }
}

parser mTag {
    switch(ethertype) {
        case 0x800: ipv4;
        // Other cases
    }
}
```

Example

```
table mTag_table {
    reads {
        ethernet.dst_addr : exact;
        vlan.vid : exact;
    }
    actions {
        // At runtime, entries are programmed with params
        // for the mTag action.  See below.
        add_mTag;
    }
    max_size : 20000;
}
```

Example




```
control main() {
    // Verify mTag state and port are consistent
    table(source_check);

    // If no error from source_check, continue
    if (!defined(metadata.ingress_error)) {
        // Attempt to switch to end hosts
        table(local_switching);

        if (!defined(metadata.egress_spec)) {
            // Not a known local host; try mtagging
            table(mTag_table);
        }

        // Check for unknown egress state or
        // bad retagging with mTag.
        table(egress_check);
    }
}
```

Example: New Action Specification

```
action add_mTag(up1, up2, down1, down2, egr_spec) {
    add_header(mTag);
    // Copy VLAN ethertype to mTag
    copy_field(mTag.ethertype, vlan.ethertype);
    // Set VLAN's ethertype to signal mTag
    set_field(vlan.ethertype, 0xaaaa);
    set_field(mTag.up1, up1);
    set_field(mTag.up2, up2);
    set_field(mTag.down1, down1);
    set_field(mTag.down2, down2);

    // Set the destination egress port as well
    set_field(metadata.egress_spec, egr_spec);
}
```

P4 Compiler

- If the target is a fixed-function switch?
    - Check if specified parser and match-action tables are supported
    - If not, return error
- If target is a software switch?
    - Full flexibility to execute specified program
    - May use specific software data structures for optimizations
- If target is an RMT switch?
    - Figure out table layout
        - Mapping logical stages to physical ones
    - If tables don’t fit, an action not support, etc: return an error.

P4 Challenges

- Optimizing P4 code compilation
- Debugging
- Resource management at switches
- Consistent updates across the network
- ….

Thanks!