In-band Network Telemetry(INT) using P4

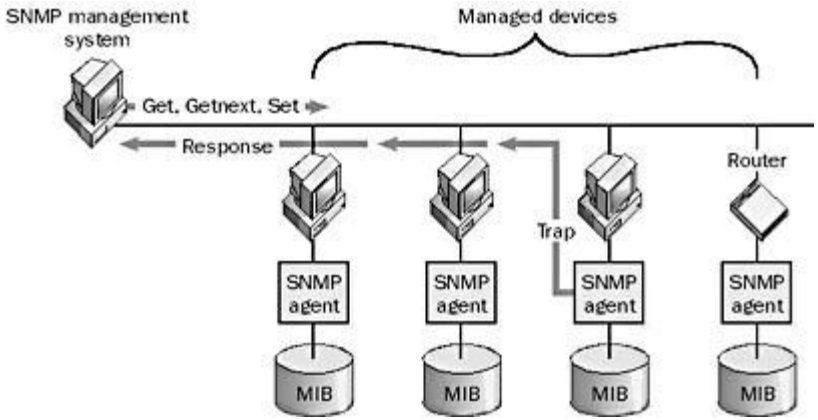
Vikas Kumar
08/05/2018
Under the guidance of
Prof Mythili Vutukuru

RnD outline

- Why INT?
- INT Introduction
- Telemetry Modes
- INT header
- Problem Statement
- Design and Implementation
- INT using P4
- Evaluation
- Summary

Why INT?

- Classical network monitoring : SNMP
  - Real time monitoring not possible
  - Control plane CPU overheads
  - Limited statistics available
  - Vendor specific support
  - Memory overhead MIB
- How INT can help?
  - Data plane implementation
  - Packet granularity
  - Configurable export information
  - No OS/CPU involvement



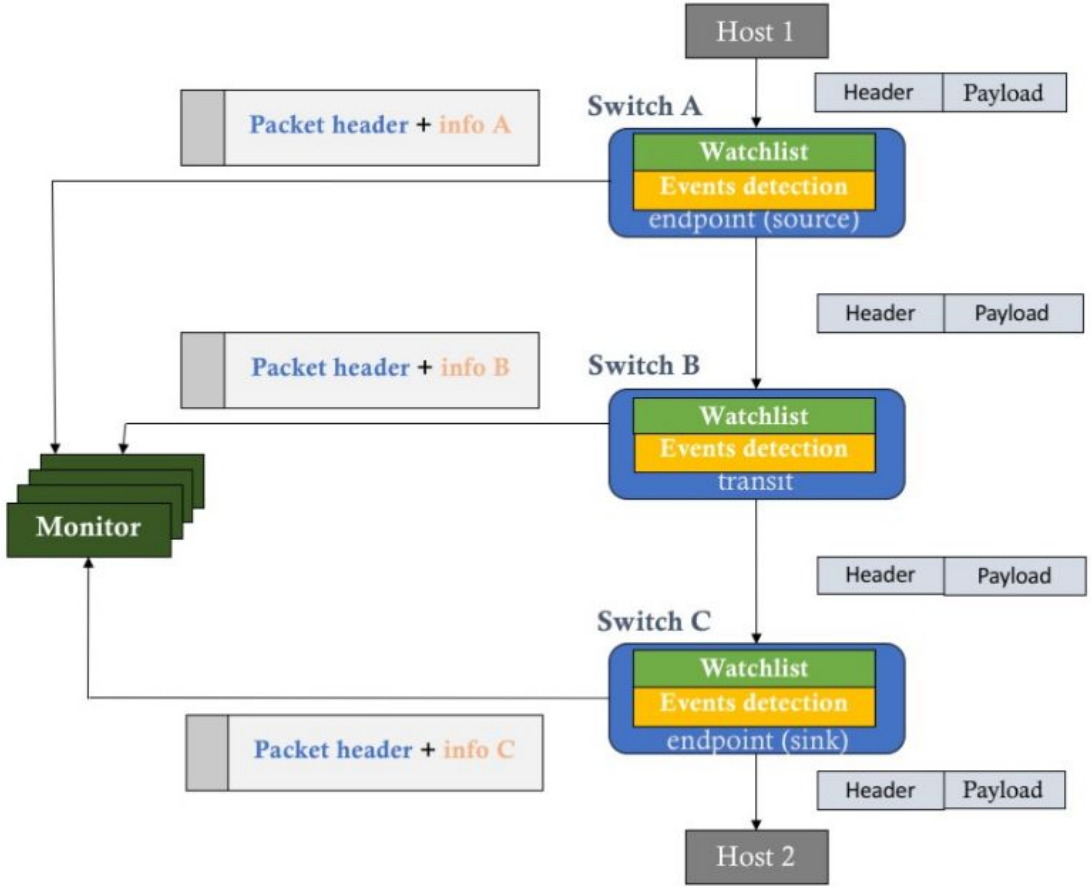
Ref:http://www.thenetworkencyclopedia.com/entry/simple-network-management-protocol-snmp/

INT Introduction

- “Inband Network Telemetry (“INT”) is a framework designed to allow the collection and reporting of network state, by the data plane, without requiring intervention or work by the control plane.” [1]
- What statistics to collect?
  - path,queue size, hop latency and link utilization
- What questions INT can answer?
  - Which path did the packet take?
  - How long did the packet queue at each switch?
  - What was the queue depth at each switch egress port?

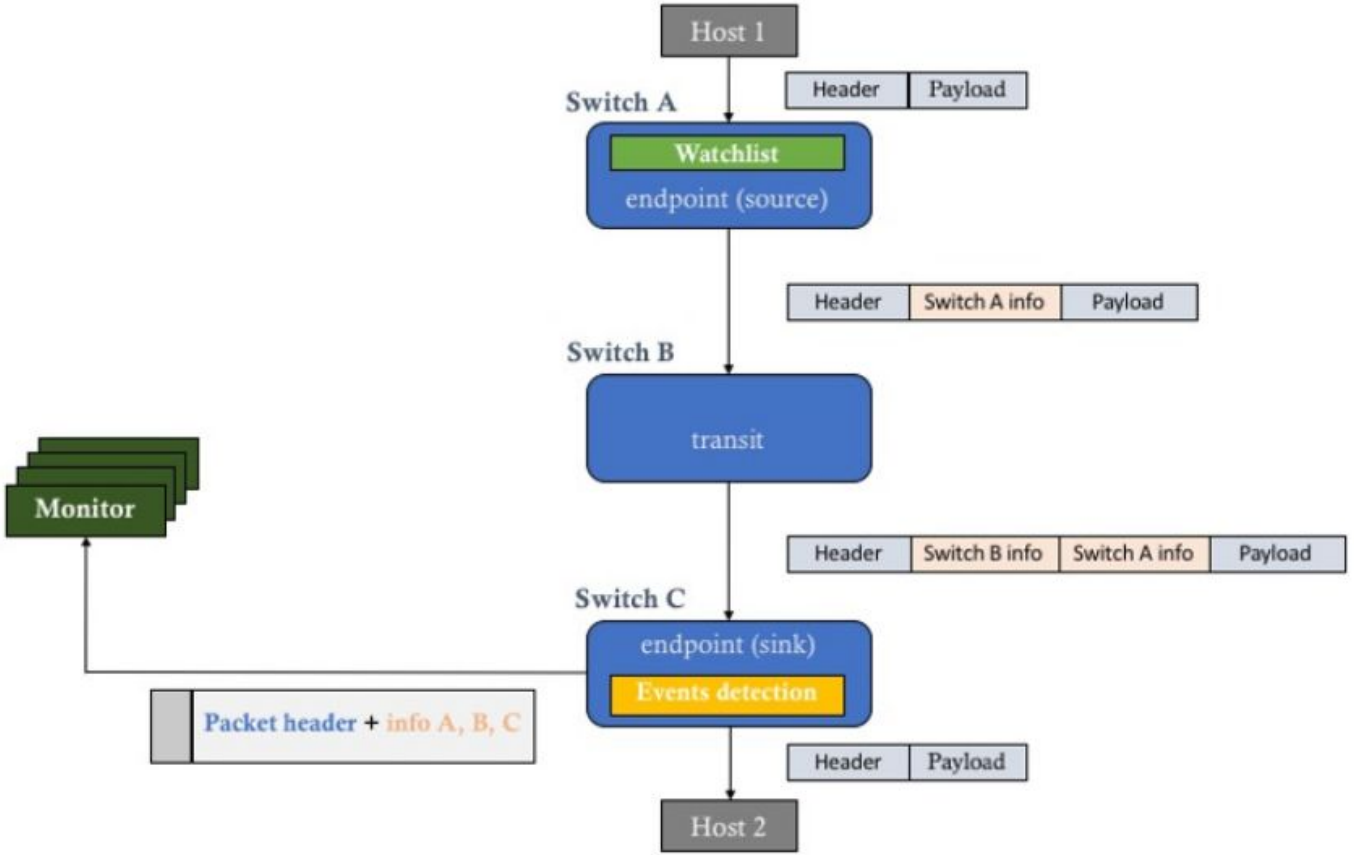
[1] : In-band Network Telemetry (INT) Dataplane Specification

Telemetry modes - Postcard



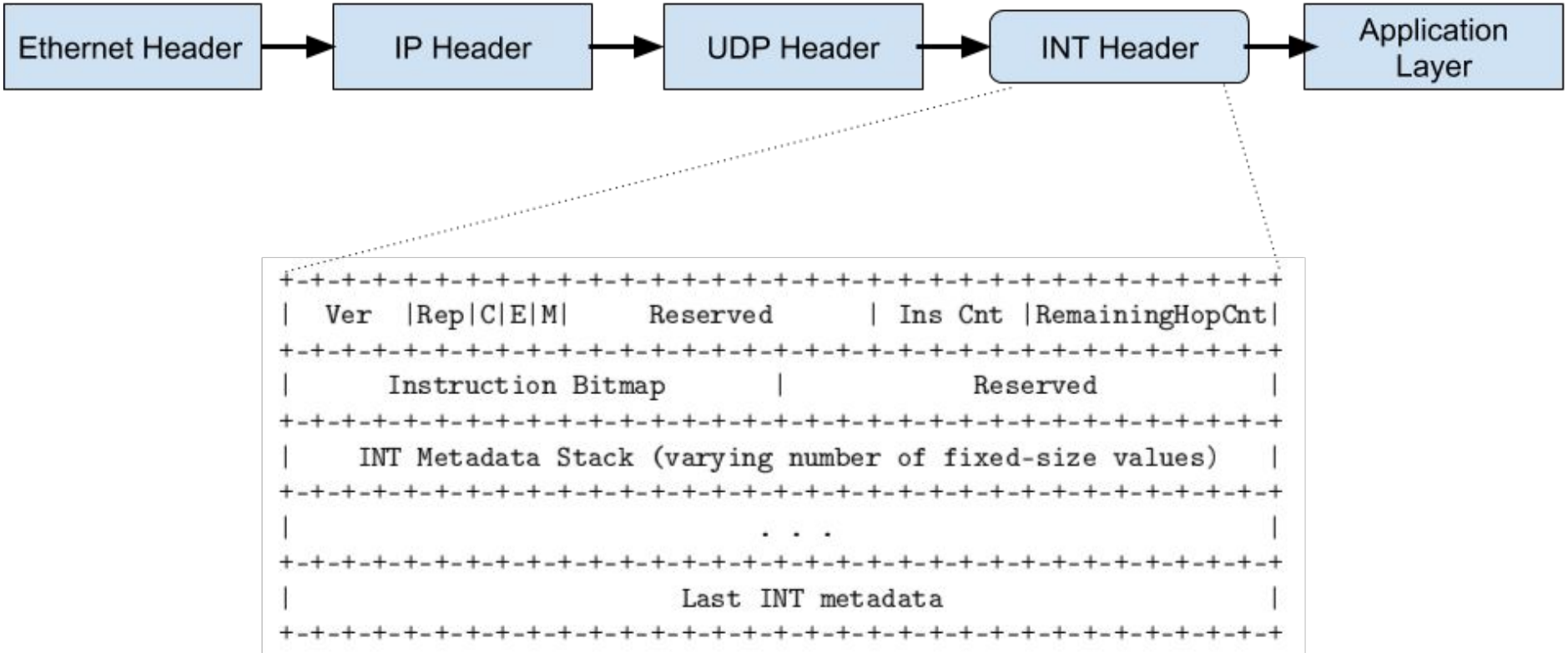
Ref : https://github.com/p4lang/p4-applications/blob/master/docs/INT.pdf

Telemetry modes - INT



Ref : https://github.com/p4lang/p4-applications/blob/master/docs/INT.pdf

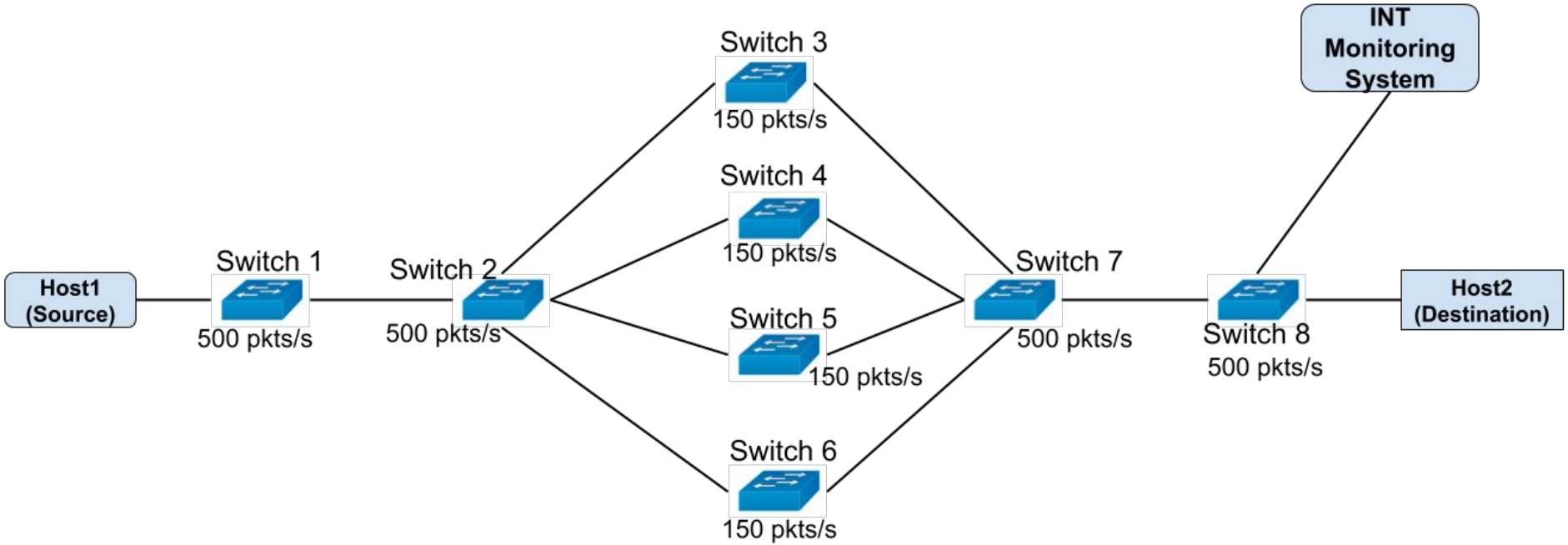
INT Header


Ethernet Header
IP Header
UDP Header
INT Header
Application Layer
+-+
| Ver |Rep|C|E|M| Reserved | Ins Cnt |RemainingHopCnt|
+-+
| Instruction Bitmap | Reserved |
+-+
| INT Metadata Stack (varying number of fixed-size values) |
+-+
| . . . |
+-+
| Last INT metadata |
+-+

Problem statement

- Given a linear network topology of n nodes with m nodes being the bottleneck nodes (m<n),
  - Identify the flows causing saturation of the bottleneck switches
  - Dynamically re-route the flows to avoid congestion at the bottleneck switches

Design of network topology



Implementation of INT

- Source switch S1 will calculate and update hash of the 4-tuple flow (Src IP, Dst IP, UDP Src port, UDP Dst Port)
- INT module -
  - Functionality?
    - Identify elephant flows
  - Where?
    - On INT Monitoring System
  - How?
    - Sample every X packets hash
      - Check if hash is repeated in more than 1/3rd of packets
    - Check if avg packet size of the flow > T_p
      - Identify the flow entry to be modified to reroute the flow at S2
    - Use switch control plane API (simple_switch_CLI) to modify the entry

How to define INT header in P4?

```
/* INT headers */
header int_header_t {
        bit<4> ver;
        bit<2> rep;
        bit<1> c;
        bit<1> e;
        bit<1> m;
        bit<7> rsvd1;
        bit<3> rsvd2;
        bit<5> ins_cnt;
        bit<8> remaining_hop_cnt;
        bit<4> instruction_mask_0003;
        bit<4> instruction_mask_0407;
        bit<4> instruction_mask_0811;
        bit<4> instruction_mask_1215;
        bit<16> rsvd3;
}
```

How to add INT header?

```
action int_source(bit<5> ins_cnt, bit<4> ins_mask0003,bit<4> ins_mask0407) {
    // insert INT shim header
    hdr.intl4_shim.setValid();
    // int_type: Hop-by-hop type (1) , destination type (2)
    hdr.intl4_shim.int_type = 1;
    hdr.intl4_shim.len = INT_HEADER_LEN_WORD;

    // insert INT header
    hdr.int_header.setValid();
    hdr.int_header.ver = 0;
    hdr.int_header.rep = 0;
    hdr.int_header.c = 0;  // copy bit
    hdr.int_header.e = 0;  // Max hop count exceeded
    hdr.int_header.m = 0;  // MTU exceeded
    hdr.int_header.rsvd1 = 0; // to track header stack
    hdr.int_header.rsvd2 = 0;
    hdr.int_header.ins_cnt = ins_cnt; // Number of instructions that are set in instruction bitmap
    hdr.int_header.remaining_hop_cnt = REMAINING_HOP_CNT;
    hdr.int_header.instruction_mask_0003 = ins_mask0003;
    hdr.int_header.instruction_mask_0407 = ins_mask0407;
    hdr.int_header.instruction_mask_0811 = 0; // not supported
    hdr.int_header.instruction_mask_1215 = 0; // not supported
    hdr.int_header.rsvd3 = 0;

    // insert INT tail header
    hdr.intl4_tail.setValid();
    hdr.intl4_tail.next_proto = hdr.ipv4.protocol;
    hdr.intl4_tail.dest_port = hdr.udp.dport;
    hdr.intl4_tail.dscp = (bit<8>) hdr.ipv4.dscp;
```

INT at Transit switches

```
action int_set_header_0() { // switch id
    //hdr.int_switch_id.setValid();
    hdr.int_switch_id.push_front(1);
    hdr.int_switch_id[0].switch_id = int_metadata.switch_id;

}
```

```
action int_set_header_2() { // hop latency
    //hdr.int_hop_latency.setValid();
    hdr.int_hop_latency.push_front(1);
    hdr.int_hop_latency[0].hop_latency = (bit<32>) standard_metadata.deq_timedelta;

}
```

```
table int_inst_0003 {
    key = {
        hdr.int_header.instruction_mask_0003 : exact;
    }
    actions = {
        int_set_header_0003_i0;
        int_set_header_0003_i1;
        int_set_header_0003_i2;
        int_set_header_0003_i3;
        int_set_header_0003_i4;
        int_set_header_0003_i5;
        int_set_header_0003_i6;
        int_set_header_0003_i7;
        int_set_header_0003_i8;
        int_set_header_0003_i9;
        int_set_header_0003_i10;
        int_set_header_0003_i11;
        int_set_header_0003_i12;
        int_set_header_0003_i13;
        int_set_header_0003_i14;
        int_set_header_0003_i15;
    }
    default_action = int_set_header_0003_i0();
    size = 16;
}
```

INT at Sink switch

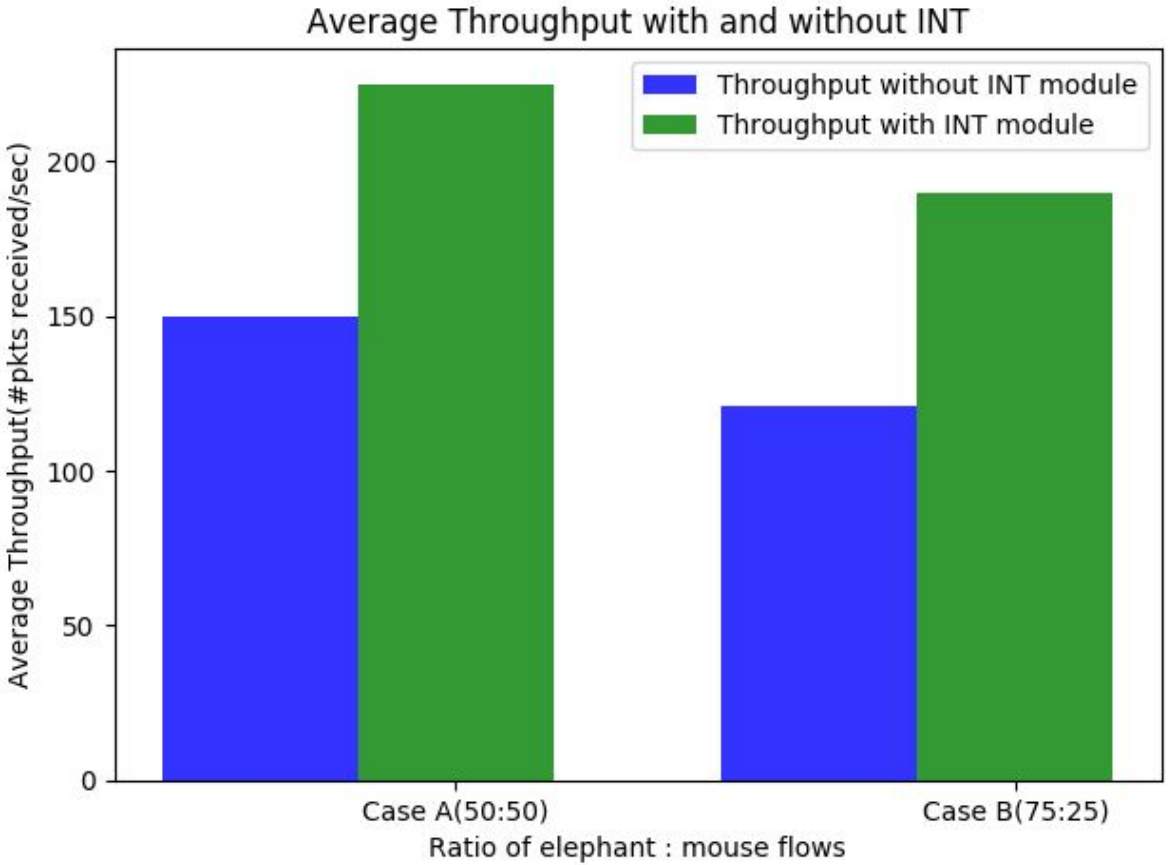
```
action int_sink() {
    // restore length fields of IPv4 header and UDP header
    hdr.ipv4.totalLen = hdr.ipv4.totalLen - (bit<16>)((hdr.intl4_shim.len - (bit<8>)hdr.int_header.ins_cnt) << 2);
    hdr.ipv4.totalLen = hdr.ipv4.totalLen - 12;
    hdr.udp.len = hdr.udp.len - (bit<16>)((hdr.intl4_shim.len - (bit<8>)hdr.int_header.ins_cnt) << 2);
    hdr.udp.len = hdr.udp.len -12;
    // remove all the INT information from the packet
    hdr.intl4_shim.setInvalid();
    hdr.int_header.setInvalid();
    hdr.int_switch_id.pop_front(REMAINING_HOP_CNT); // pop REMAINING_HOP_CNT stack
    hdr.int_hop_latency.pop_front(REMAINING_HOP_CNT);
    hdr.int_q_occupancy.pop_front(REMAINING_HOP_CNT);
    hdr.intl4_tail.setInvalid();
}
```

Evaluation

- Setup
  - Mininet setup with simple_switc_CLI target
  - Simple_switch_cli
    - Software switch from p4lang repo [2]
  - INT monitoring System
    - VM with 4vcpus, 4GB RAM
- Loads
  - 10B short flows for 0.1 sec
  - 1370B long flows for 10sec
  - Normal distribution with mean and 0.008 and 0.02 for 50:50 and 75:35 long:short flows ratio respectively

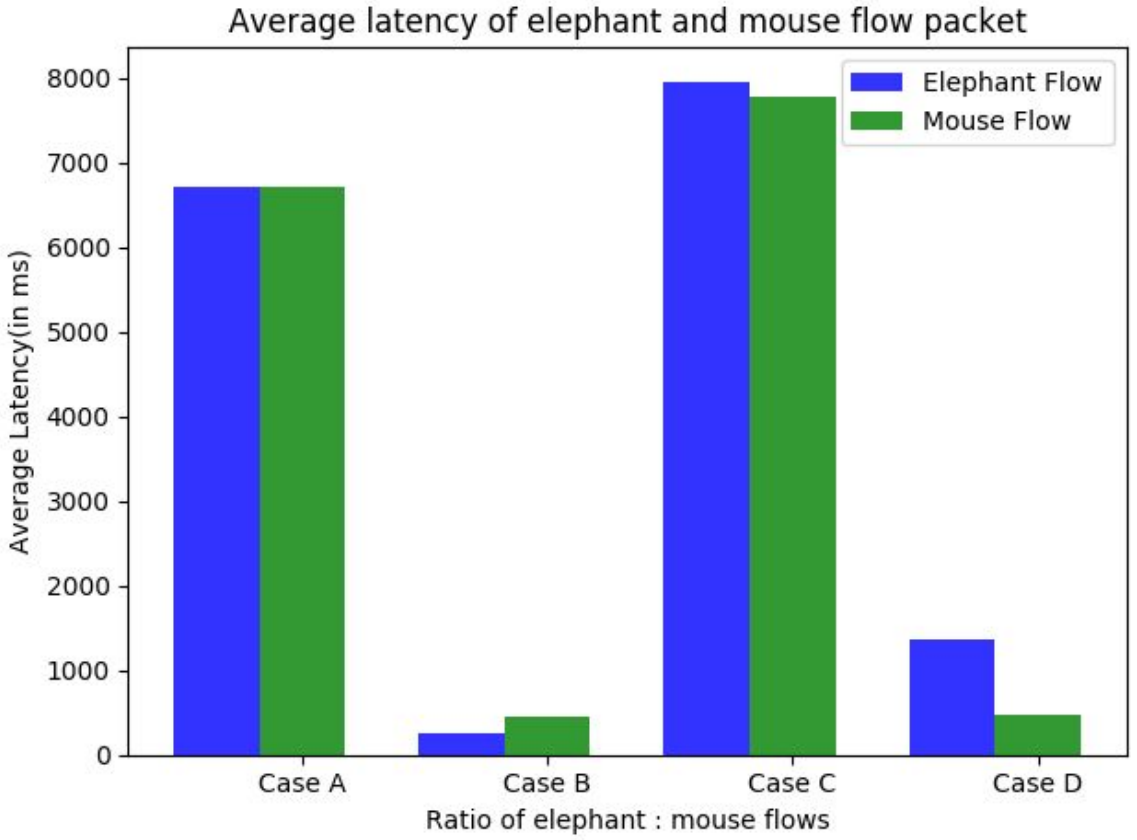
2 : https://github.com/p4lang/behavioral-model/tree/master/targets/simple_switch

Throughput with and without INT module

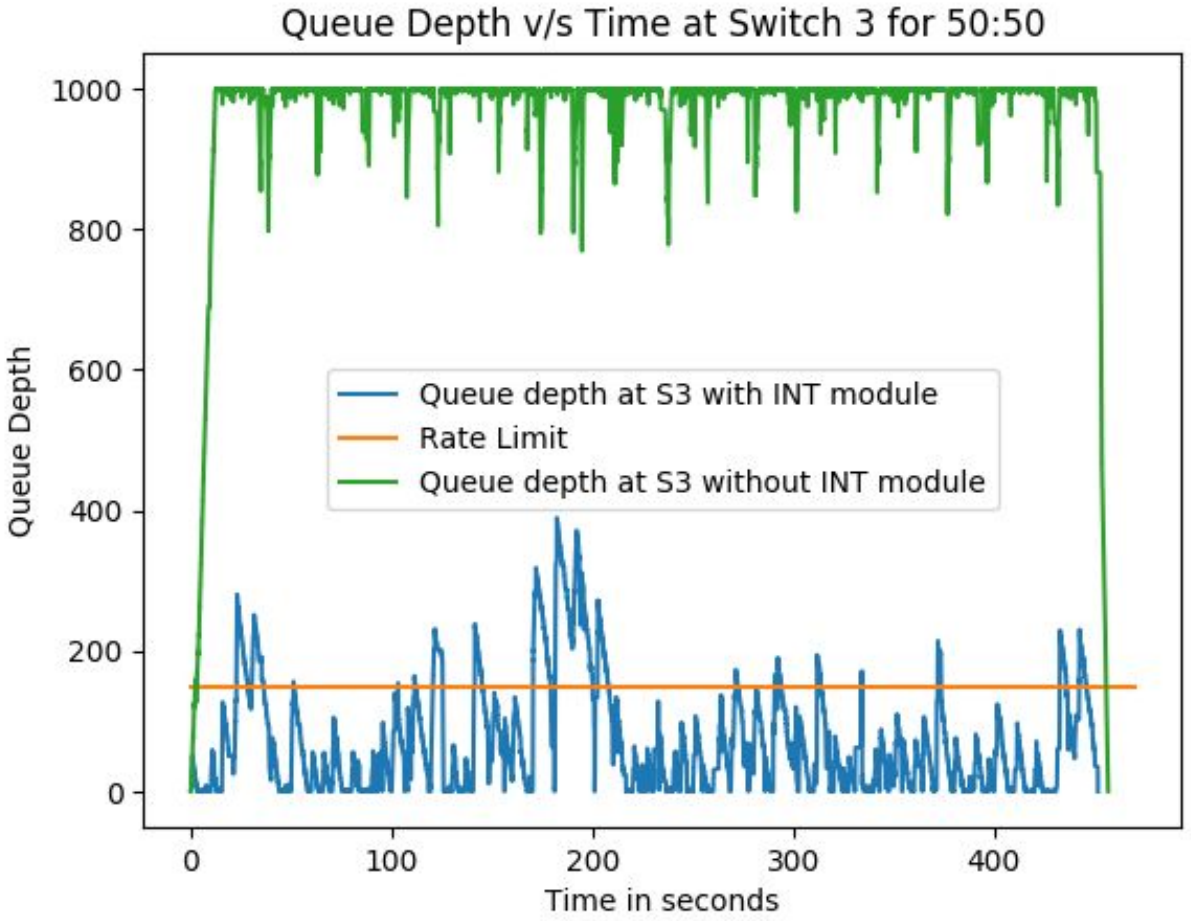


End to End avg latency with and without INT module

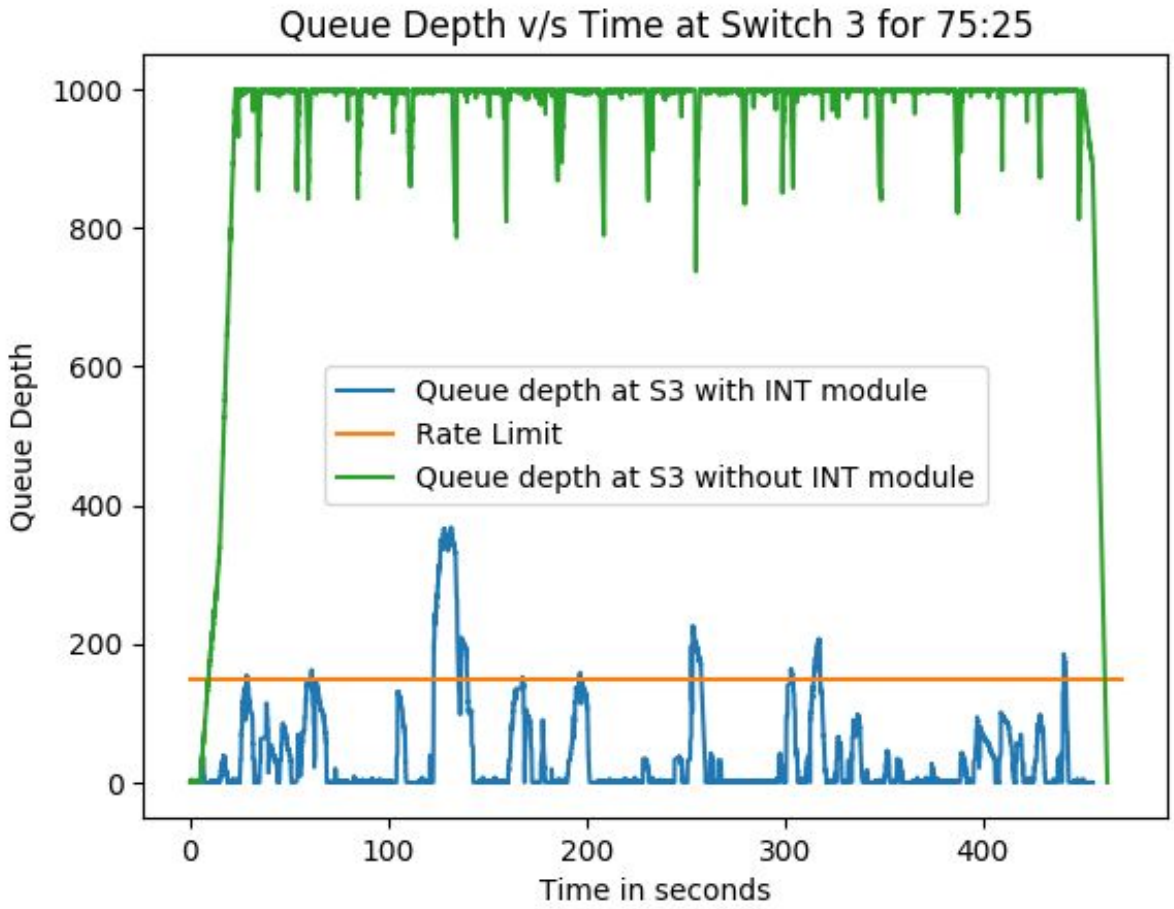
Case A - 50:50 without INT module
Case B - 50:50 with INT module
Case C - 75:25 without INT module
Case D - 75:25 with INT module



Qdepth at S3 with and without INT module(50:50)



Qdepth at S3 with and without INT module(75:25)

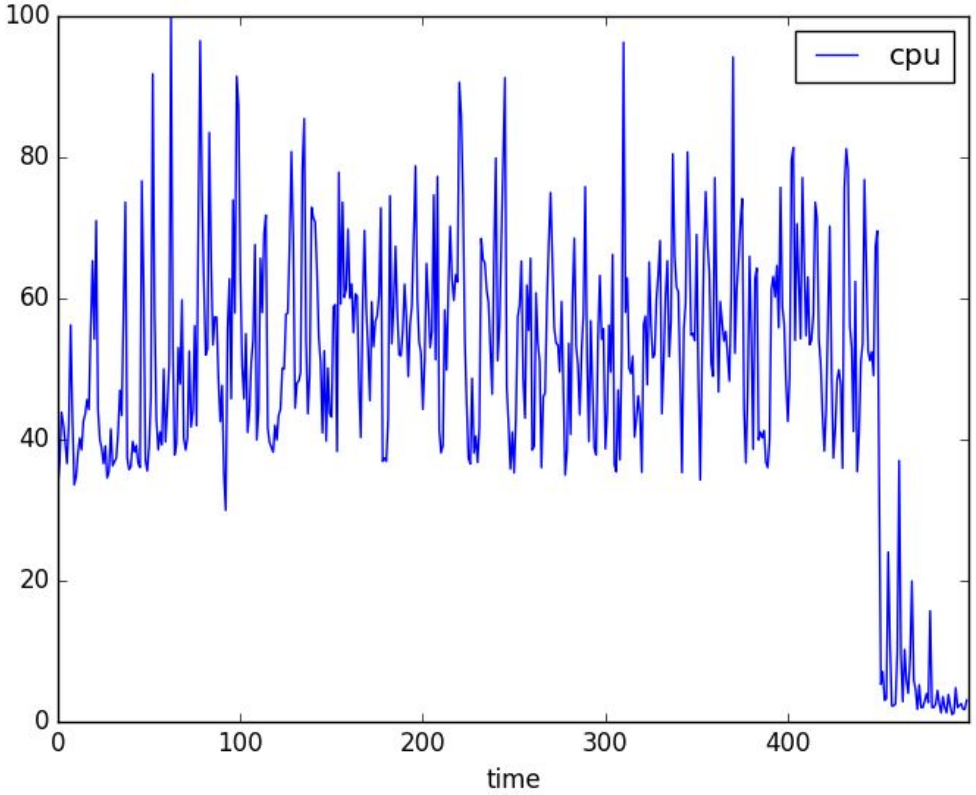


Summary

- INT provides better visibility at network core
- Response time
- Per packet granularity
- Limitations of INT specifications:
  - MTU can exceed if we have more application data with a complex topology
- Using INT module together with switch control plane flows are rerouted as excepted with 50% increase in throughput and 93% reduction in end to end latency.
- INT module is able to handle dynamic workload

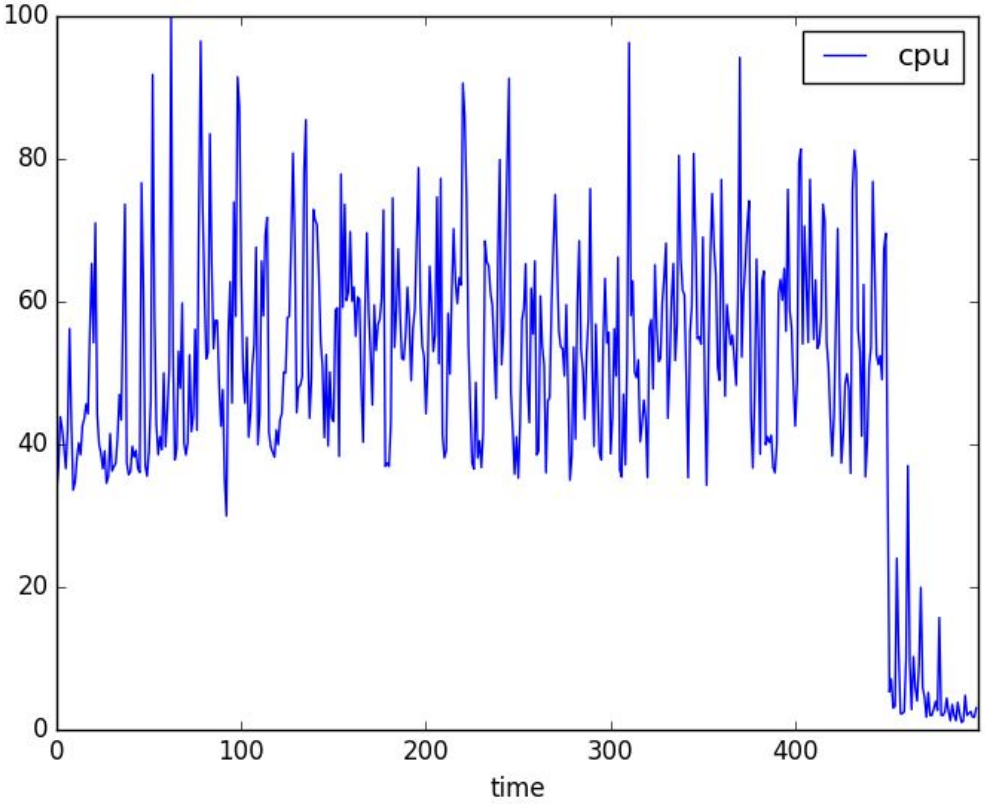
Thank You

Backup



Cpu utilization 50 50

TH



Cpu utilization 75 25