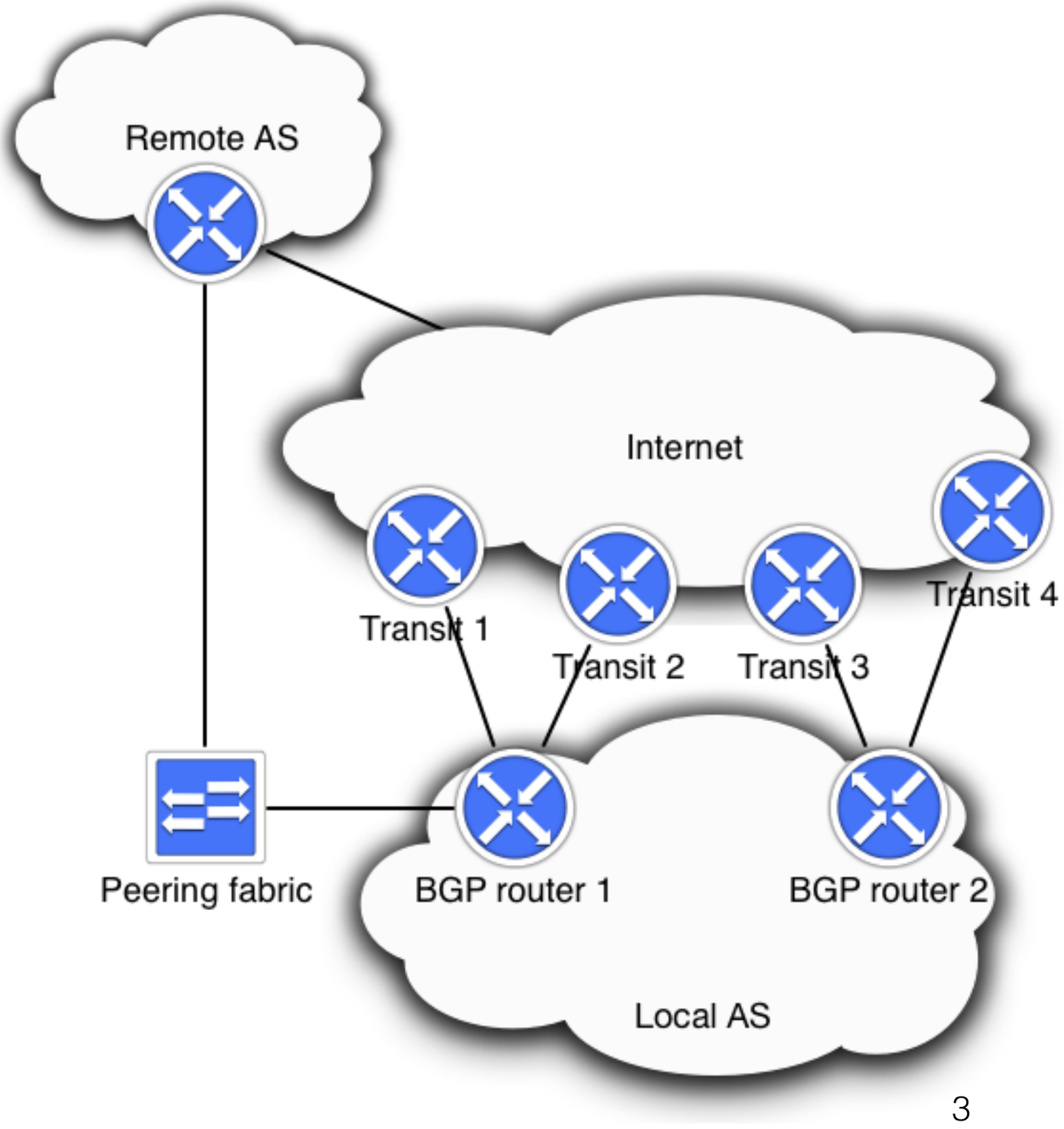
Internet Probing for Multi-homed AS

Wenqin SHAO

Summary

- The problem, motivation and challenges
- Mini tutorial to OpenFlow, rules of the game
- The design of flow table in POC, master the game
- Demo
- Future works

The problem and motivation



Inter-domain next-hop performance evaluation

RTT { packet loss
availability

modify

a pair of Path

partial description

next-hops

The problem (cont.)

BGP path

fixes the egress next-hop

Reference path

fixes both egress and ingress next-hop

Why?

Incoming path diversity->potential performance gains
Xavier's work on path diversity

Challenges

Outgoing

how to distribute probe traffic on all next-hops?

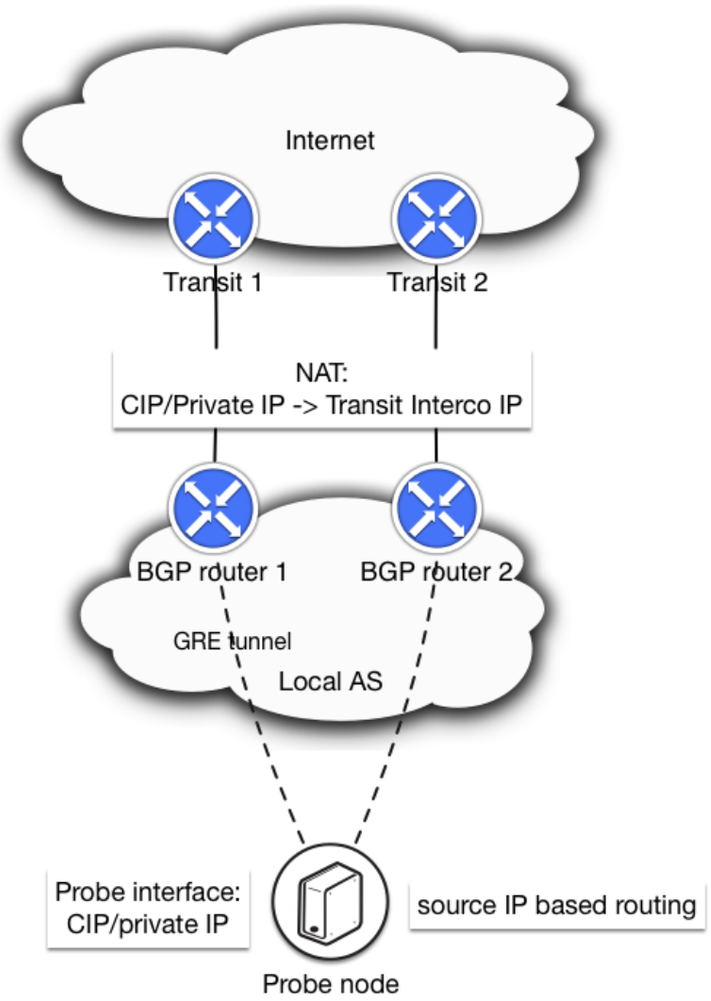
Incoming

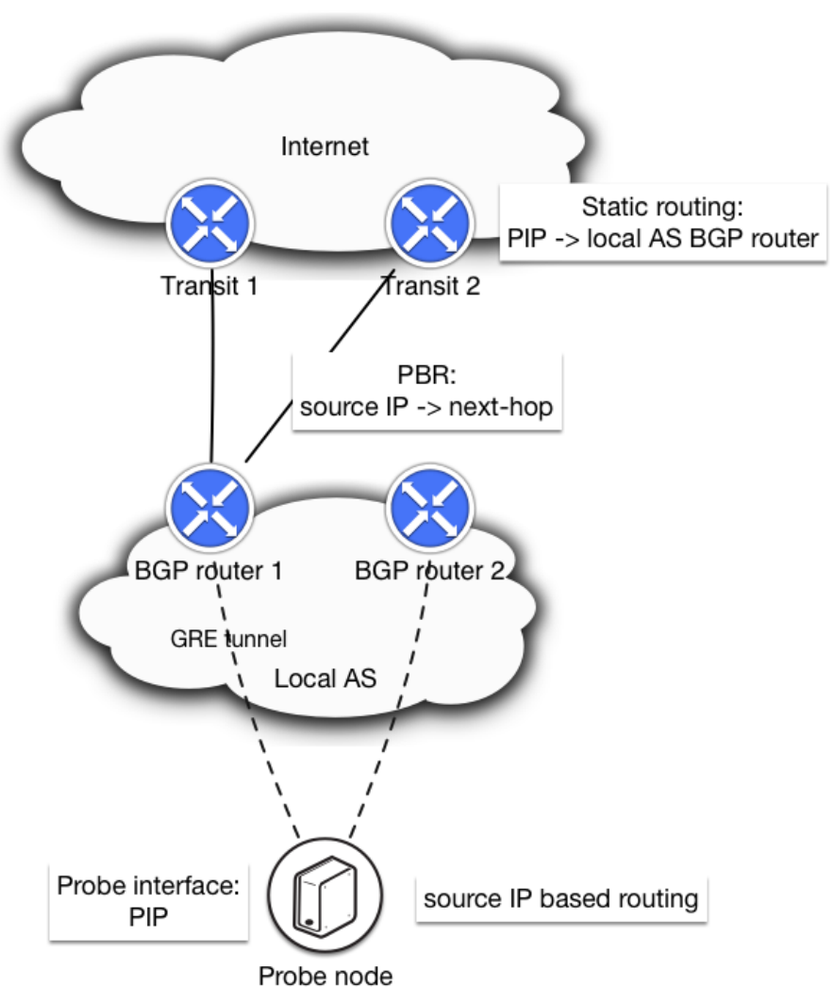
how to fixes probe traffic on certain ingress points?

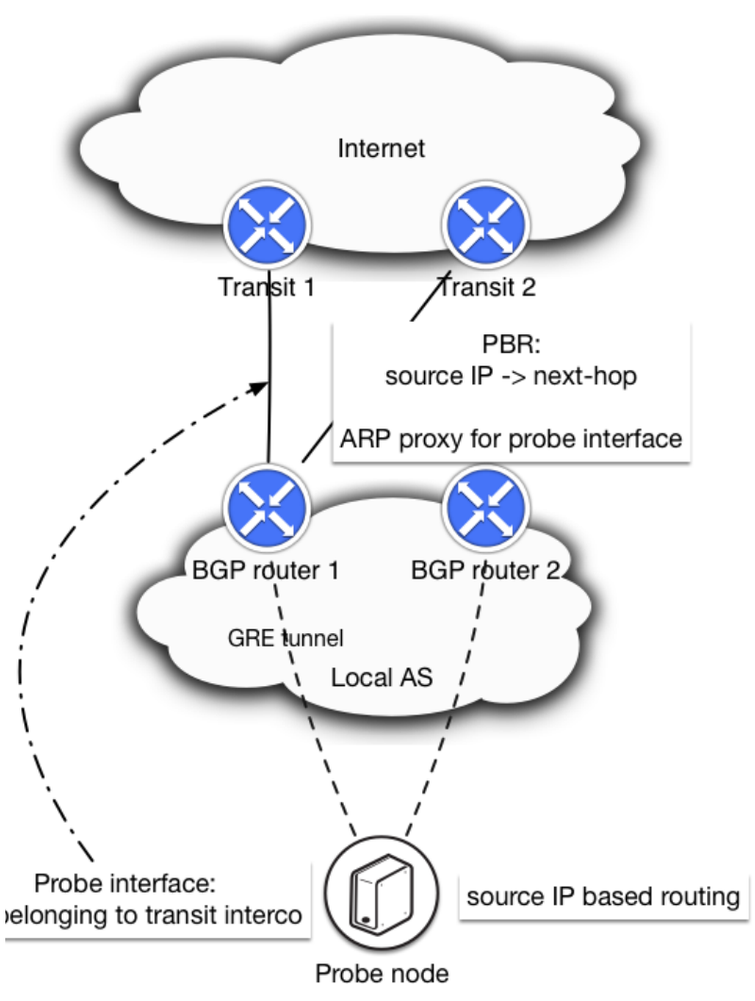
BGP path->CIP (Customer IP)

Reference path->PIP (Provider IP)

Challenges (cont.)

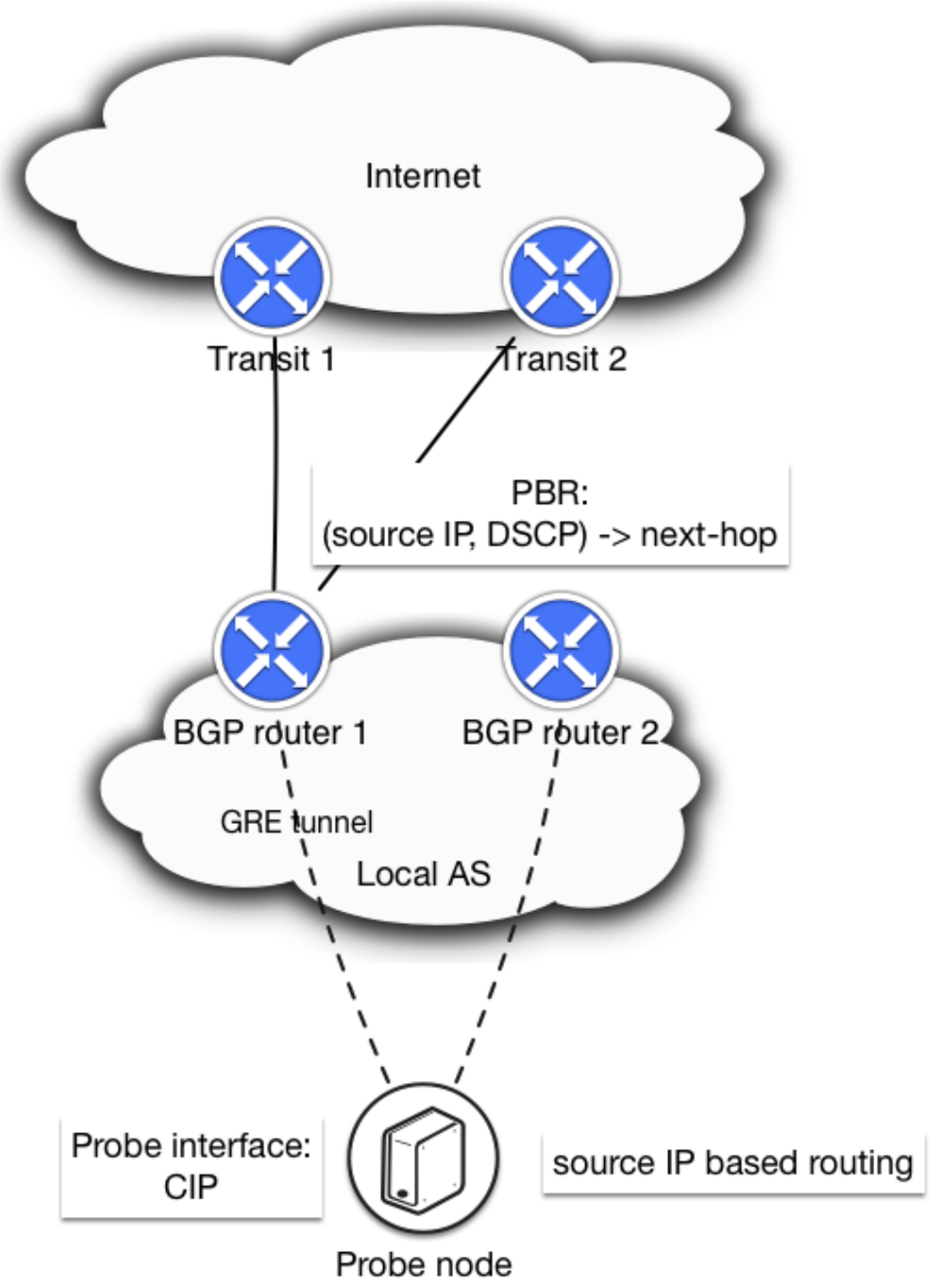






PIP probing

Challenges (cont.)



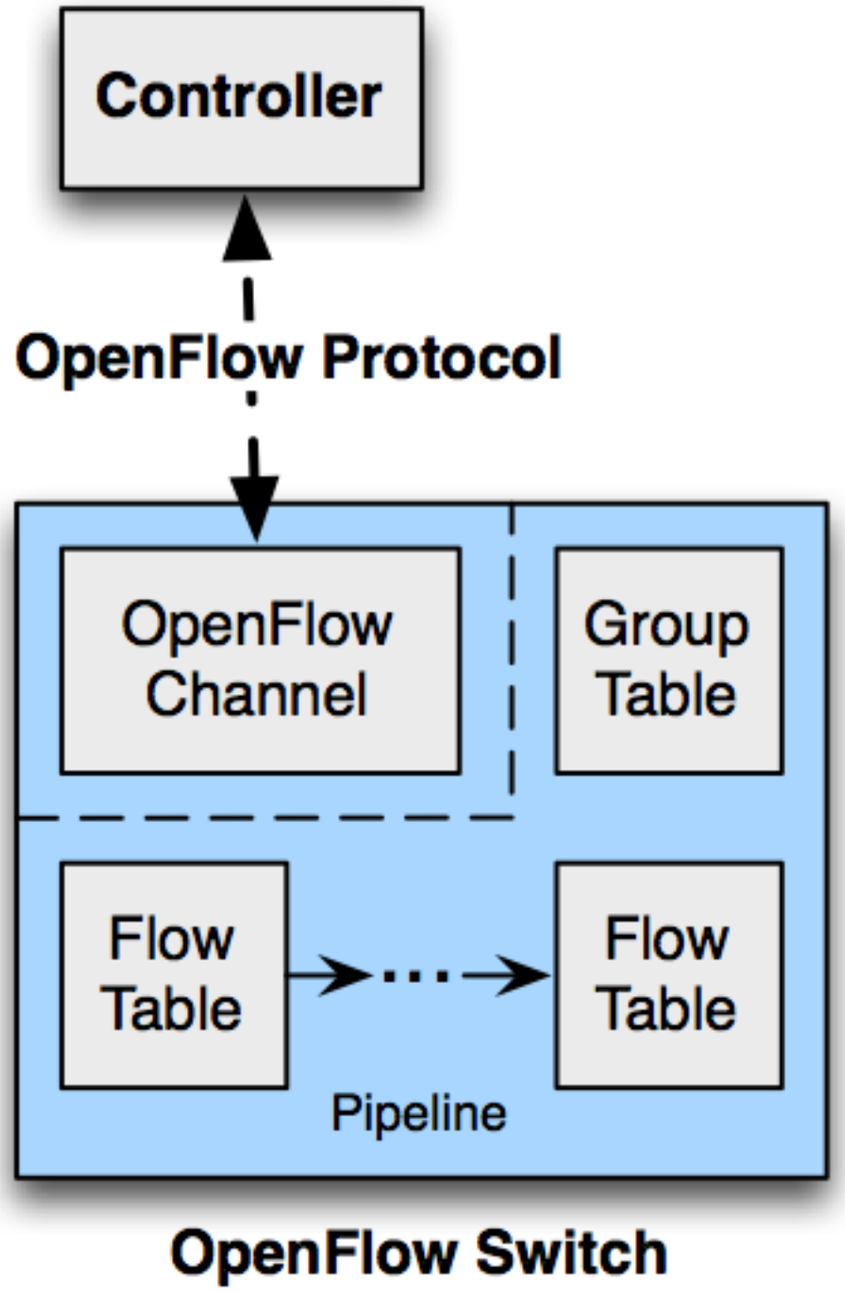
CIP probing

Why SDN? Why OpenFlow?

Configurability
Maintenance time
Evolution
Scalability
etc.

How to benchmark
these properties
is something that
we should think
about in evaluation part

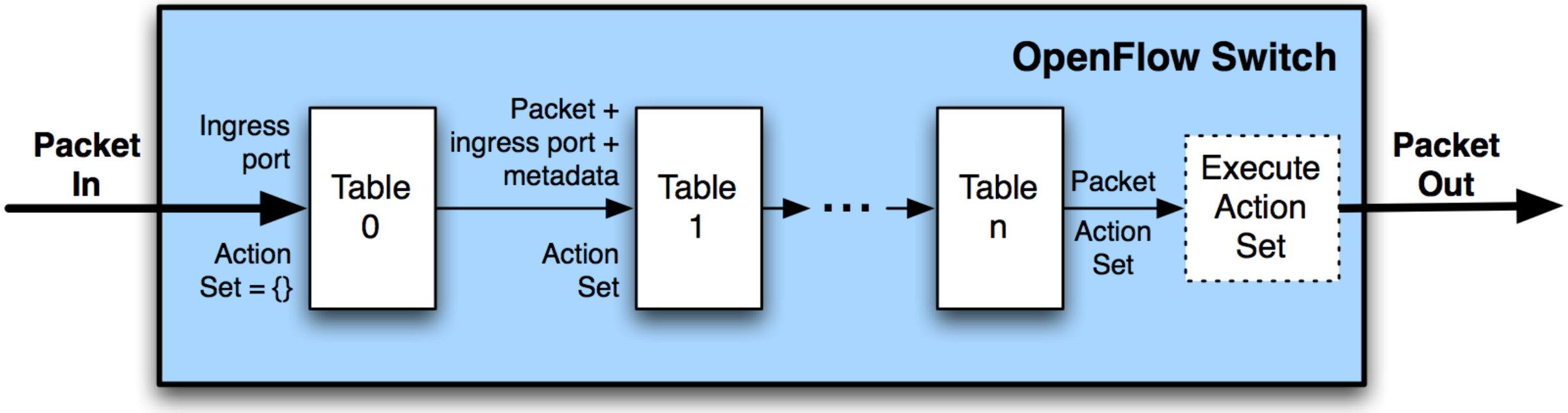
Mini-tutorial to OpenFlow



What is an OpenFlow Switch capable of?
How to design each flow table?
How to design the pipeline?
so that….

good performance
extensibility
debug
maintenance
isolation between network functions
etc.

Mini-tutorial to OpenFlow (cont.)



Flow table and pipeline

Mini-tutorial to OpenFlow (cont.)

Pipeline : [flow_table,] # instructions decides table seq

flow_table : [flow_entry,] # priority order

flow_entry : {match: [match_field,]
instructions: [instructions([actions,]),]}

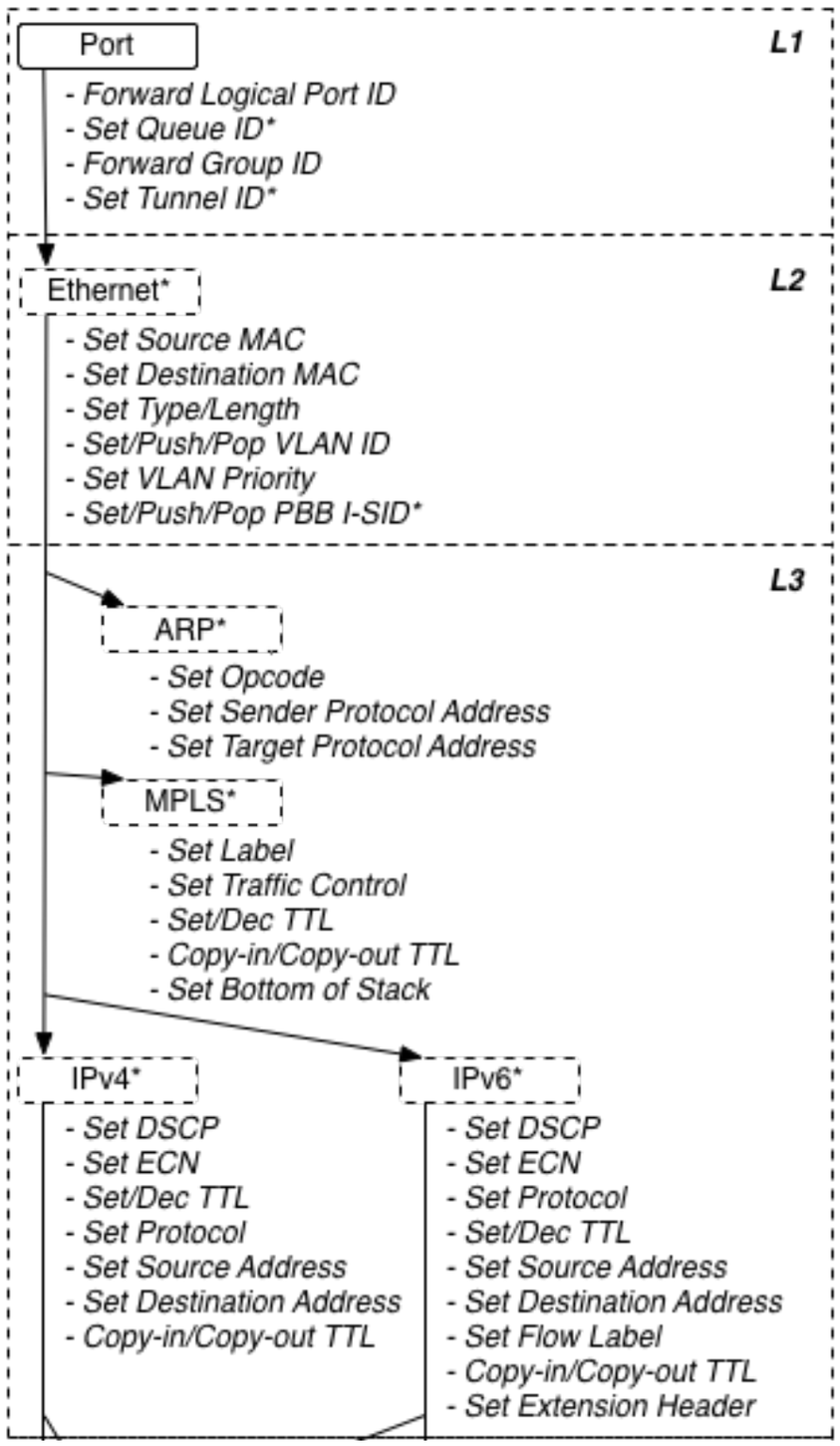
all possible match fields
all possible actions and their orders
all possible instructions and their orders

Mini-tutorial to OpenFlow (cont.)

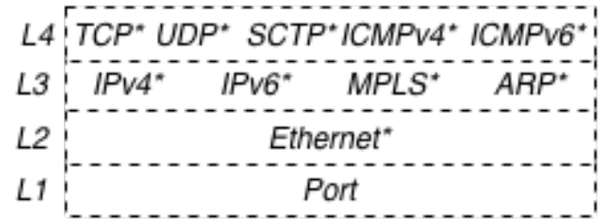
OFPXMT_OFB_IN_PORT
OFPXMT_OFB_IN_PHY_PORT
OFPXMT_OFB_METADATA
OFPXMT_OFB_ETH_DST
OFPXMT_OFB_ETH_SRC
OFPXMT_OFB_ETH_TYPE
OFPXMT_OFB_VLAN_VID
OFPXMT_OFB_VLAN_PCP
OFPXMT_OFB_IP_DSCP
OFPXMT_OFB_IP_ECN
OFPXMT_OFB_IP_PROTO
OFPXMT_OFB_IPV4_SRC
OFPXMT_OFB_IPV4_DST
OFPXMT_OFB_TCP_SRC
OFPXMT_OFB_TCP_DST
OFPXMT_OFB_UDP_SRC
OFPXMT_OFB_UDP_DST
OFPXMT_OFB_SCTP_SRC
OFPXMT_OFB_SCTP_DST
OFPXMT_OFB_ICMPV4_TYPE
OFPXMT_OFB_ICMPV4_CODE
OFPXMT_OFB_ARP_OP
OFPXMT_OFB_ARP_SPA
OFPXMT_OFB_ARP_TPA
OFPXMT_OFB_ARP_SHA
OFPXMT_OFB_ARP_THA
OFPXMT_OFB_IPV6_SRC
OFPXMT_OFB_IPV6_DST
OFPXMT_OFB_IPV6_FLABEL
OFPXMT_OFB_ICMPV6_TYPE
OFPXMT_OFB_ICMPV6_CODE
OFPXMT_OFB_IPV6_ND_TARGET = 31
OFPXMT_OFB_IPV6_ND_SLL
OFPXMT_OFB_IPV6_ND_TLL
OFPXMT_OFB_MPLS_LABEL
OFPXMT_OFB_MPLS_TC
OFPXMT_OFP_MPLS_BOS
OFPXMT_OFB_PBB_ISID
OFPXMT_OFB_TUNNEL_ID
OFPXMT_OFB_IPV6_EXTHDR

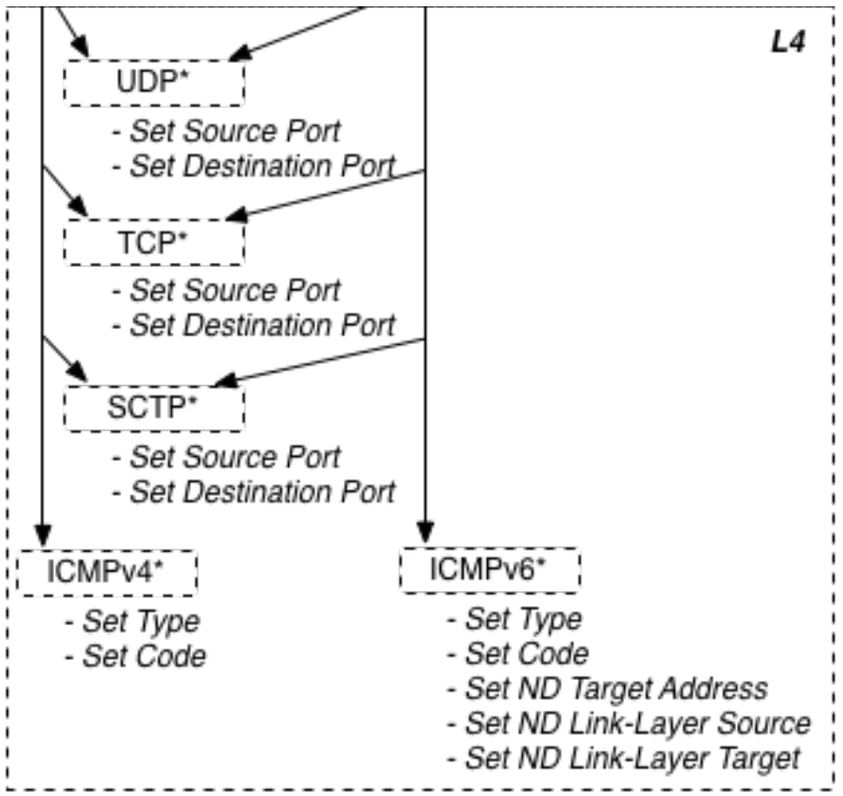
Complete list of match fields in OF v1.3 works with mask

Mini-tutorial to OpenFlow (cont.)



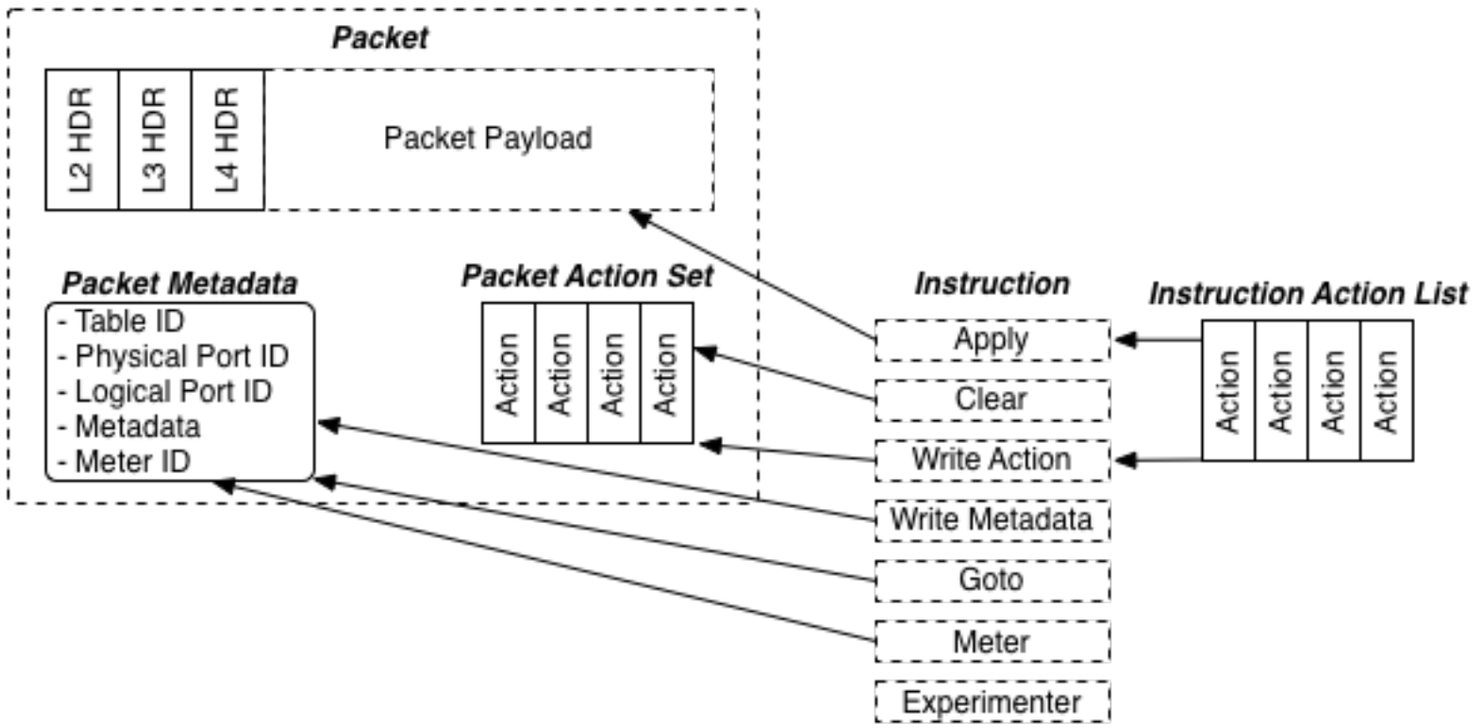
Actions as of OF v1.3





Mini-tutorial to OpenFlow (cont.)

Instructions as of OF v1.3



<u>order:</u>
meter
apply actions
clear actions (the whole set)
write actions
write metedata
goto table

Mini-tutorial to OpenFlow (cont.)

Difference between action set and action list

action set

one action each type

later one overwrites the former one

used with WRITE_ACTIONS inst

order:

copy TTL inwards

pop

puch-MPLS

push PBB

push-VLAN

copy TTL outwards

decrement TTL

set

qos

group

output

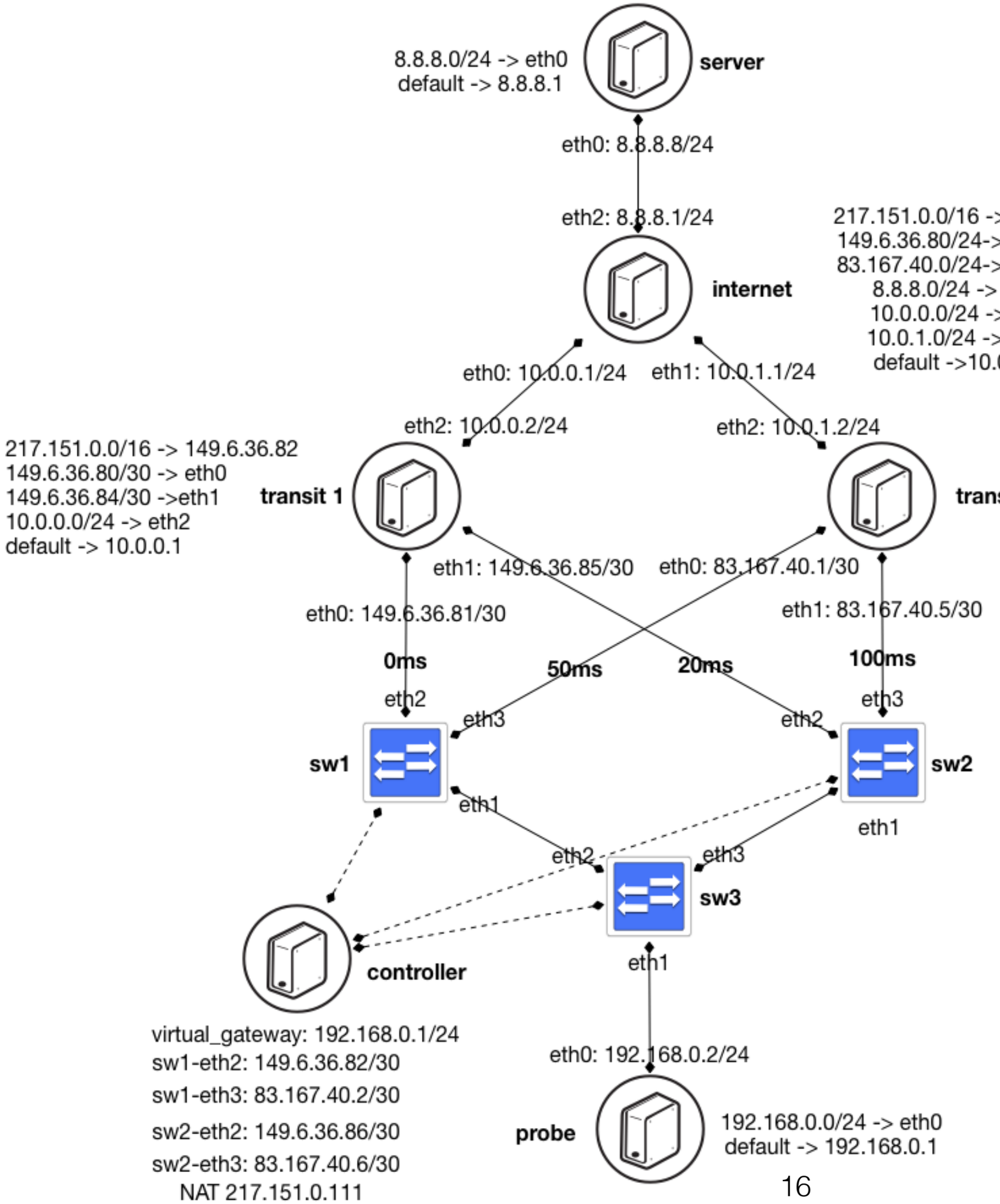
action list

multiple action each type

cumulative effect

used with APPLY_ACTIONS inst

order specified by the list itself



Probe interface config
192.168.0.11 PIP@149.6.36.81
192.168.0.12 PIP@149.6.36.85
192.168.0.13 PIP@83.167.40.1
192.168.0.14 PIP@83.167.40.5
192.168.0.21 CIP@149.6.36.81
192.168.0.22 CIP@149.6.36.85
192.168.0.23 CIP@83.167.40.1
192.168.0.24 CIP@83.167.40.5

Topology scripted in Mininet

Flow tables on BGP routers (sw1, sw2)

Catch ARP (table_id = 0)

Priority	Match	Instruction
10	ARP	[APPLY_ACT([CONTROLLER,]),]
10	IP	[GOTO_TBL(NAT_TABLE),]
0	[]	DROP

NAT_TABLE (table_id = 1)

Priority	Match	Instruction
20	ip, tcp, ip_src=X, tcp_src=Y	[WRITE_ACT([set_field:X'->ip_src, set_field:Y'->tcp_src,]), GOTO(PBR_TABLE),]
20	ip, tcp, ip_dst=S, tcp_dst=T	[APPLY_ACT([set_field:S'->ip_src, set_field:T'->tcp_src,]), GOTP(PBR_TABLE),]
10	ip, ip_dst = interco/NAT	[APPLY_ACT([CONTROLLER,]),]
10	ip, src_ip = 192.168.0.0/24	[APPLY_ACT([CONTROLLER,]),]
0	[]	[GOTO_TBL(PBR_TABLE),]

PBR_TABLE (table_id = 2)

Priority	Match	Instruction
10	ip, ip_src = probe ip	[WRITE_METADATA(nh_ip), WRITE_ACT([Output(port_no),]), GOTO_TBL(MAC_TABLE),]
0	[]	[GOTO_TBL(ROUTING_TABLE)]

ROUTING_TABLE (table_id = 3)

Priority	Match	Instruction
20	ip, ip_dst = next_hop	[WRITE_METADATA(nh_ip), WRITE_ACT(Output(port_no)), GOTO_TBL(MAC_TABLE),]
10	ip, ip_dst = 192.168.0.0/24	[APPLY([Output(port_no),]),]
0	[]	[WRITE_METADATA(default_nh_ip), WRITE_ACT([Output(port_no),]), GOTO_TBL(MAC_TABLE),]

MAC_TABLE (table_id = 4)

Priority	Match	Instruction
10	metadate=nh_ip	[WRITE_ACT([set_field:MAC->eth_dst,]),]
0	[]	[APPLY_ACT([CONTROLLER,]), CLEAR_ACT,]

DROP

FORWARDED

CONTROLLER

Flow tables on local AS network (sw3)

Catch ARP (table_id = 0)

Priority	Match	Instruction
10	ARP	[APPLY_ACT([CONTROLLER,]),]
10	IP	[GOTO_TBL(PBR_TABLE),]
0	[]	DROP

PBR_TABLE (table_id = 1)

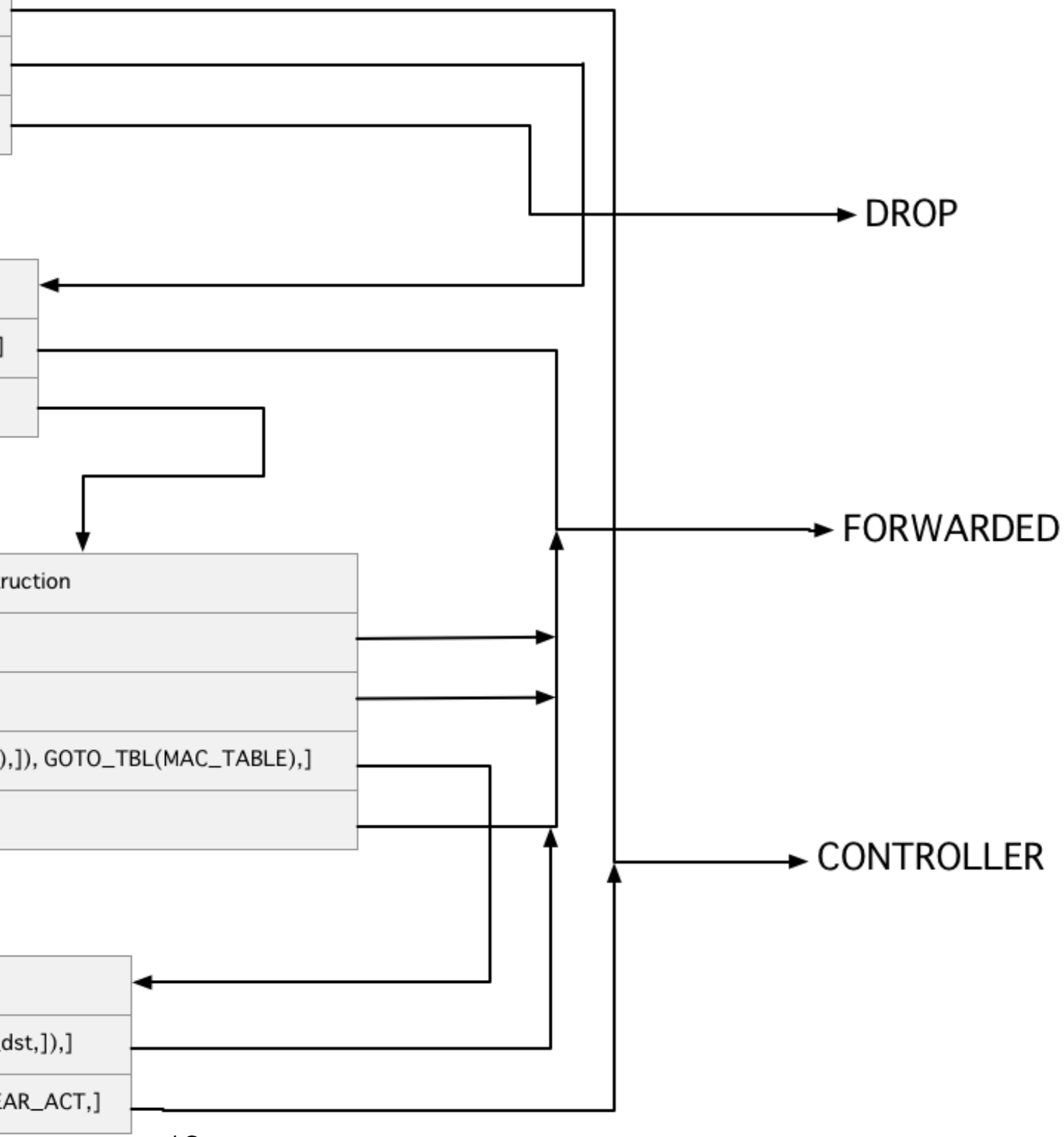
Priority	Match	Instruction
10	ip, ip_src = probe ip	[APPLY_ACT([Output(port_no),]),]
0	[]	[GOTO_TBL(ROUTING_TABLE)]

ROUTING_TABLE (table_id = 2)

Priority	Match	Instruction
20	ip, ip_dst = gw	[APPLY([Output(port_no),]),]
20	ip, ip_dst = next_hop	[APPLY([Output(port_no),]),]
10	ip, ip_dst = 192.168.0.0/24	[WRITE_ACT([Output(port_no),]), GOTO_TBL(MAC_TABLE),]
0	[]	[APPLY([Output(port_no),]),]

MAC_TABLE (table_id = 3)

Priority	Match	Instruction
10	ip, ip_dst=nh_ip	[WRITE_ACT([set_field:MAC->eth_dst,]),]
0	[]	[APPLY_ACT([CONTROLLER,]), CLEAR_ACT,]



Controller logic

- Config phase
    - structure pipeline; pre-install static flow entries
- Main phase
    - ARP handler:
        - learn ip->mac mapping, write to MAC_TABLE
        - answer to ARP request to virtual IP
- NAT_TABLE miss
    - ICMP NAT: rephrase ICMP echo; reply to echo request
    - TCP NAT: install flows, in and out;
- Reply to ICMP echo request to virtual IP
- MAC_TABLE miss
    - Send ARP request using propre virtual IP

wenqin@wenqin-Tkp-Ub: ~/b6_probing 151x13

```
217.151.0.0/16 via 149.6.36.82
*** setting IP and routes for t2
t2-eth2 10.0.1.2/24
t2-eth1 83.167.40.5/30
t2-eth0 83.167.40.1/30
default via 10.0.1.1
217.151.0.0/16 via 83.167.40.6
*** setting probing source IP for probe
*** Starting SimpleHTTPServer on  server
*** Starting CLI:
mininet> xterm probe
mininet> xterm server
mininet>
```

wenqin@wenqin-Tkp-Ub: ~/ryu 151x21

```
INFO: Flow Install
    dpid 1; table_id: 4; priority: 10; idle_timeout: 0
    Match: OFPMatch(oxm_fields={'metadata': 2500207697})
    Instructions:[OFPInstructionActions(actions=[OFPActionSetField(eth_dst='46:e9:66:cf:aa:1c')],type=3)]
DEBUG: Outgoing probe packet NAT flow table miss
INFO: Flow Install
    dpid 1; table_id: 1; priority: 20; idle_timeout: 300
    Match: OFPMatch(oxm_fields={'ip_proto': 6, 'eth_type': 2048, 'ipv4_src': '192.168.0.11', 'in_port': 1, 'tcp_src': 5001})
    Instructions:[OFPInstructionActions(actions=[OFPActionSetField(ipv4_src='149.6.36.82'), OFPActionSetField(tcp_src=13968)],type=3), OFPInstructionGo
toTable(len=8,table_id=2,type=1)]
INFO: Flow Install
    dpid 1; table_id: 1; priority: 20; idle_timeout: 300
    Match: OFPMatch(oxm_fields={'ip_proto': 6, 'eth_type': 2048, 'ipv4_dst': '149.6.36.82', 'tcp_dst': 13968})
    Instructions:[OFPInstructionActions(actions=[OFPActionSetField(ipv4_dst='192.168.0.11'), OFPActionSetField(tcp_dst=5001)],type=4), OFPInstructionGo
toTable(len=8,table_id=2,type=1)]
INFO: MAC learnt 149.6.36.81@46:e9:66:cf:aa:1c at dpid 1
INFO: Flow Install
    dpid 1; table_id: 4; priority: 10; idle_timeout: 0
    Match: OFPMatch(oxm_fields={'metadata': 2500207697})
    Instructions:[OFPInstructionActions(actions=[OFPActionSetField(eth_dst='46:e9:66:cf:aa:1c')],type=3)]
```

wenqin@wenqin-Tkp-Ub: ~ 151x20

Node: probe

```
[  5] local 192.168.0.13 port 5001 connected with 8.8.8.8 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.1 sec   599 MBytes   500 Mbits/sec
root@wenqin-Tkp-Ub:/# arp -i probe-eth0 -d 192.168.0.1
root@wenqin-Tkp-Ub:/# iperf -c 8.8.8.8 -B 192.168.0.12
------------------------------------------------------------
Client connecting to 8.8.8.8, TCP port 5001
Binding to local address 192.168.0.12
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  5] local 192.168.0.12 port 5001 connected with 8.8.8.8 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.0 sec  1.30 GBytes  1.11 Gbits/sec
root@wenqin-Tkp-Ub:/# arp -i probe-eth0 -d 192.168.0.1
root@wenqin-Tkp-Ub:/# iperf -c 8.8.8.8 -B 192.168.0.11
------------------------------------------------------------
Client connecting to 8.8.8.8, TCP port 5001
Binding to local address 192.168.0.11
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  5] local 192.168.0.11 port 5001 connected with 8.8.8.8 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.0 sec  25.7 GBytes  22.1 Gbits/sec
root@wenqin-Tkp-Ub:/#
```

Node: server

```
root@wenqin-Tkp-Ub:/# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  6] local 8.8.8.8 port 5001 connected with 217.151.0.111 port 19023
[ ID] Interval       Transfer     Bandwidth
[  6]  0.0-10.1 sec   677 MBytes   565 Mbits/sec
[  7] local 8.8.8.8 port 5001 connected with 217.151.0.111 port 36922
[  7]  0.0-10.0 sec  19.3 GBytes  16.6 Gbits/sec
[  6] local 8.8.8.8 port 5001 connected with 217.151.0.111 port 56171
[  6]  0.0-10.2 sec  62.2 MBytes  51.3 Mbits/sec
[  7] local 8.8.8.8 port 5001 connected with 83.167.40.2 port 32886
[  7]  0.0-10.3 sec  48.8 MBytes  39.6 Mbits/sec
[  6] local 8.8.8.8 port 5001 connected with 83.167.40.6 port 32458
[  6]  0.0-10.4 sec   284 MBytes   229 Mbits/sec
[  7] local 8.8.8.8 port 5001 connected with 83.167.40.2 port 52602
[  7]  0.0-10.1 sec   599 MBytes   499 Mbits/sec
[  6] local 8.8.8.8 port 5001 connected with 149.6.36.86 port 3561
[  6]  0.0-10.1 sec  1.30 GBytes  1.11 Gbits/sec
[  7] local 8.8.8.8 port 5001 connected with 149.6.36.82 port 13968
[  7]  0.0-10.0 sec  25.7 GBytes  22.1 Gbits/sec
```

Future works

- How to evaluate this POC? Is OpenFlow really helpful in doing the job?
- Related works on active probing
- Combination of active and passive measuring (metering function in OpenFlow)
- Cross-analysis of probing results
    - probing results from multiple GP sites
    - external probing results: RIPE Atlas, Planet labs, etc.
- Probing result quality evaluation
- Monitoring system for intra-domain