Counteracting UDP Flooding Attacks in SDN

Yung-Hao Tung, Hung-Chuan Wei, Chia-Mu Yu

Yuan Ze University

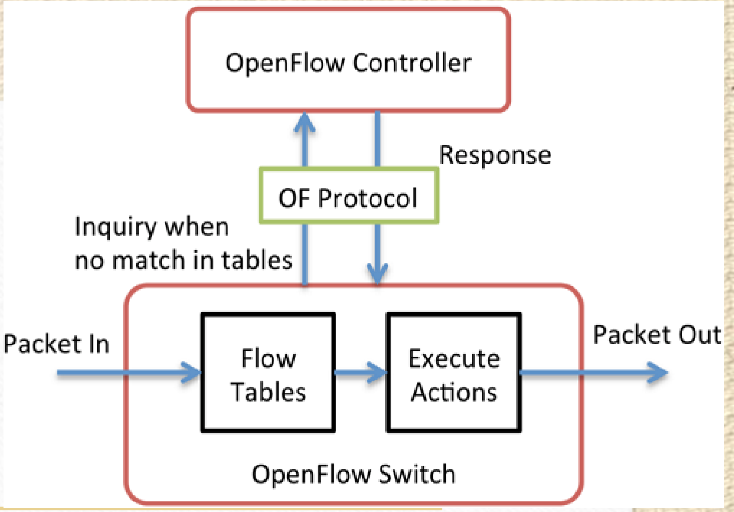
Outline

- **SDN** overview
- Problem statement
- Proposed method
- Experiments

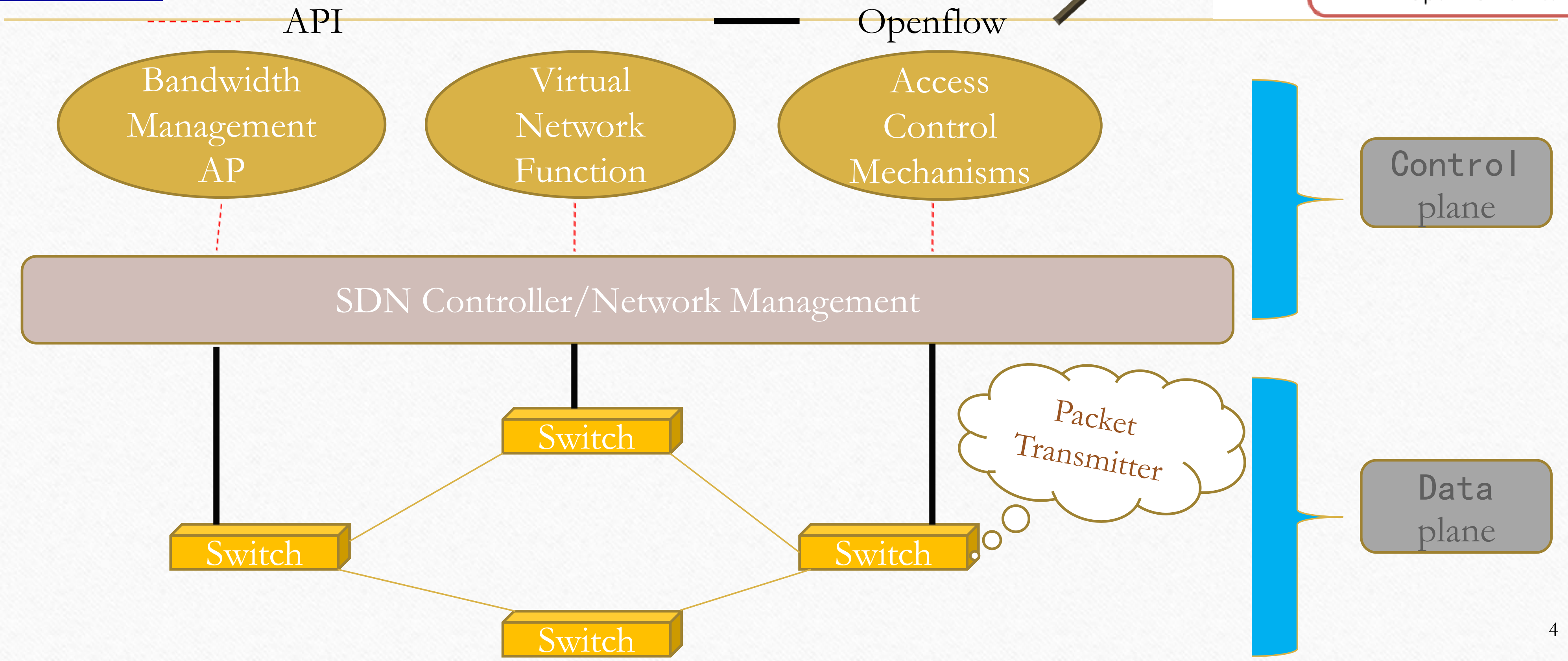
SDN Introduction

- Centralized approach
- SDN mainly divided into control plane and data plane
- SDN uses the OpenFlow protocol
- SDN switch has a flow table, trying to have a rule match against the received packets

SDN Introduction



Framework:



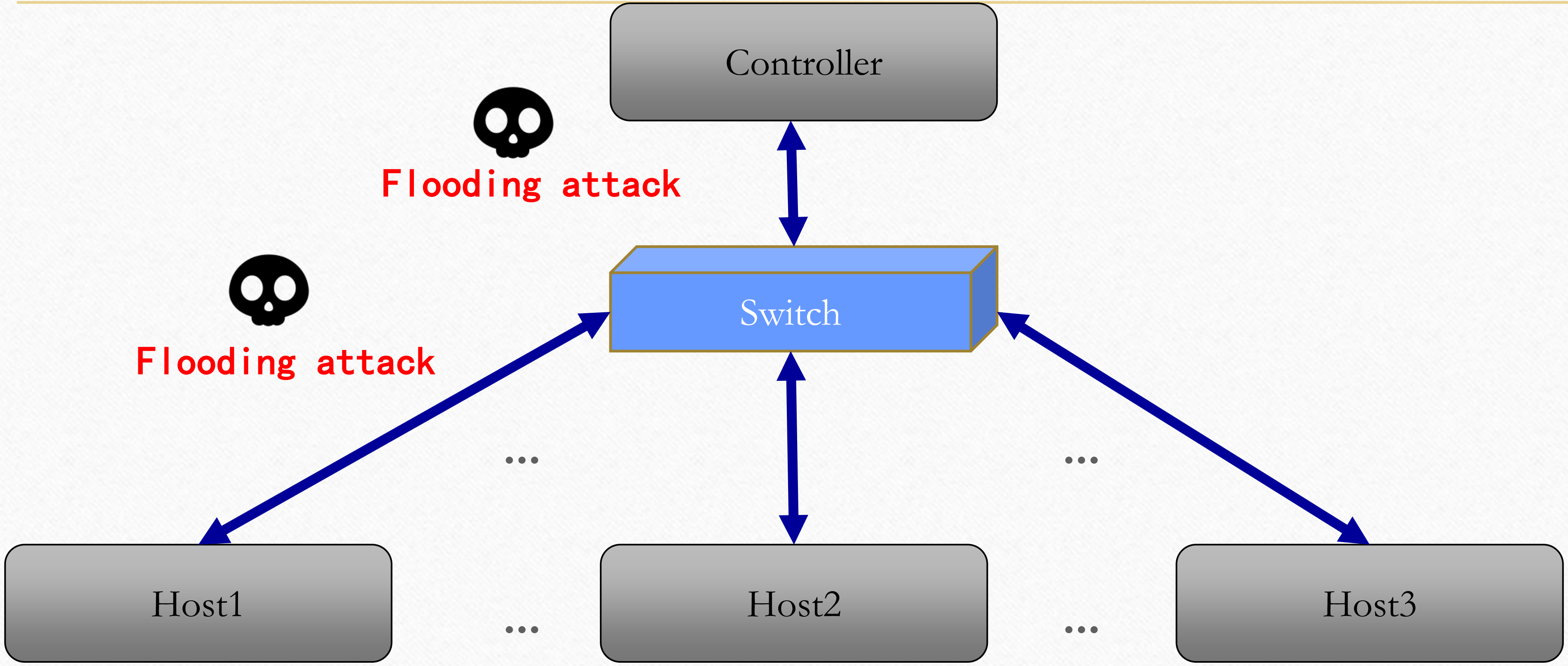
Problem Statement

- **Network Security**
  - The easiest way of compromising a network is to launch a flooding attack (ex: TCP SYN flooding, UDP flooding etc).

- **SDN Security Problems**
  - When a new flow arrives, the SDN switch will send a packet-in message to the SDN controller.
  - However, intentional abusing the controller (or say packet-in message) may incur the security problem.

Problem Statement

Simulation SDN Network Attack Graph



PROTOCOL DESIGN

- **Our experiment can be divided into two phases**
  - First, consider a bunch of simple UDP packets transmitted to the switch.
  - Then, we began to do the code implementation on the simulated switch and controller, and evaluated the performance and the security of our defense mechanism.

PROTOCOL DESIGN

Attack Model:

- In the case of no match found, the controller will perform a broadcast to ask whether there is a match for the purpose of IP addresses.

- The attacker can assign a random value to the destination field in the packet.

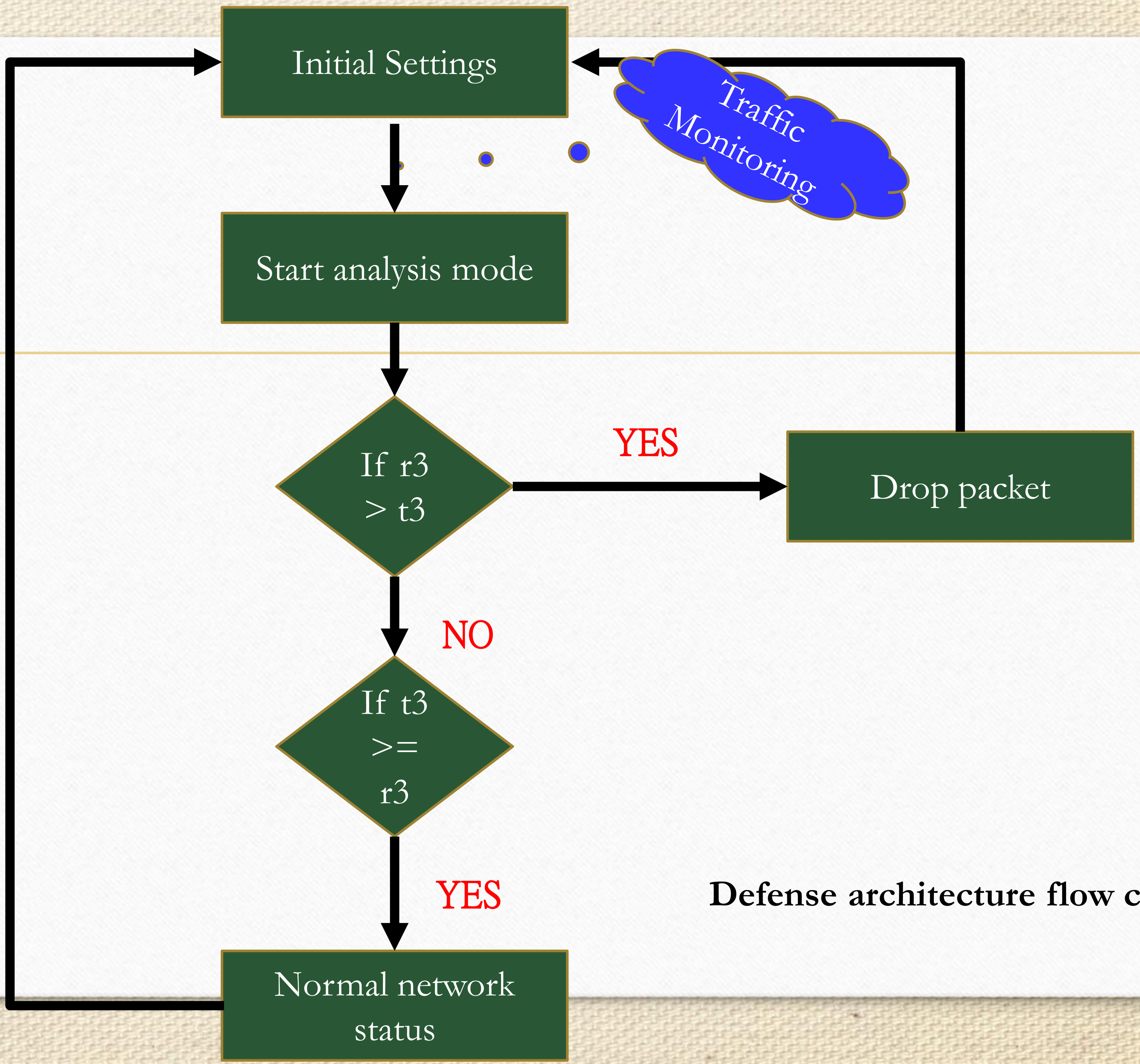
```
def generate_ip(): # Create random IP
return str(random.randint(0, 255)) + '.'\ + str(random.randint(0, 255)) + '.'\
+ str(random.randint(0, 255)) + '.'\ + str(random.randint(0, 255))
```

UDP Packet Section.

PROTOCOL DESIGN

	Total Rate	CPU(s)	Load avg.
Normal state	5 kbits/sec	0.6 us	0.32
Attack state	6100 ↑ kbits/sec	27 ↑ us	0.87 ↑

Defense Architecture



Defense architecture flow chart

r3 : The number of packets receive by the port

t3 : The number of packets sent by the port

Defense Architecture

- Our analysis model has two conditions.
  - If the received packet (r3) ＞ send packets (t3):
    - This means that the destination of the sending packet does not exist in the current network, resulting in the controller constantly broadcasting.
  - If the packet is sent (t3) ＞＝ receive packets (r3):
    - The controller can handle the packet-in message and broadcast packets.

Defense Architecture

UDP Defense Section

```
@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def _packet_in_handler(self, ev):
if ev.msg.msg_len < ev.msg.total_len: self.logger.debug("packet truncated: only %s of %s bytes",
ev.msg.msg_len, ev.msg.total_len)
.
.
if(r3 > t3):
        actions = []
.
.
elif(t3 >= r3):
       flooding
```

Defense Architecture

Return packets on all ports

```
body = ev.msg.body
        self.logger.info('datapath         port      '
                'rx-pkts  rx-bytes rx-error '
                'tx-pkts  tx-bytes tx-error')
            self.logger.info('---------------- -------- '
                    '-------- -------- -------- '
                    '-------- -------- --------')
            for stat in sorted(body, key=attrgetter('port_no')):
        self.logger.info('%016x %8x %8d %8d %8d %8d %8d %8d',
            ev.msg.datapath.id, stat.port_no,
            stat.rx_packets, stat.rx_bytes, stat.rx_errors,
            stat.tx_packets, stat.tx_bytes, stat.tx_errors
```

EXPERIMENTS

- Experiment Setting
  - In the experiment. we use mininet to simulate the SDN OpenFlow switch, and use RYU to simulate the controller.
  - Moreover, IPerf, TOP, IPTRAF are used as monitoring tools.
  - For the network topology, we considered two physical hosts and a controller.
  - They are on different physical machines for ensuring more accurate measurement.

EXPERIMENTS

- Defense Achievements

  - In our experiment, we consider two cases (with and without attack) and observe the difference between these two cases.

EXPERIMENTS

Network bandwidth and controller performance comparison

	IPerf	Top	IPtraf
No Defense	TX bps:412 Bytes/s	CPU(s)： 27.2 us	Total rate: 6139.0 Kbits/sec 4846.4 packets/sec
Defense	TX bps: 33 Bytes/s	CPU(s)： 14.8 us	Total rate: 2790.7 Kbits/sec 1861.8 packets/sec

Related Work

- Comparison of Defense

	FloodGuard	UDP
No Defense	7 Mbps	6 Mbps
Defense	2 Mbps	2 Mbps

CONCLUSION

- The proposed defense resist against the UDP flooding with a minor modification in SDN module.

- The countermeasure particularly designed for only UDP flooding works with better performance

Let us know if you have any comments or questions.

Thank you for listening.

Mailbox:

s1036023@mail.yzu.edu.tw

s1036010@mail.yzu.edu.tw

chiamuyu@saturn.yzu.edu.tw

Question

- Given the operation flow chart probing the switches periodically, it would be a naturally raised question how much overhead this approach would introduce.
- Furthermore, this question extends to what is the parameters we should consider to trade off security and performance compromise.

Answer

- Using this method, we are only at the expense of request packet for some time. The following mechanisms to facilitate the analysis.
- Although this sacrifices some benign request, but in exchange for increased security.
- But in the time of the attack, a benign request to wait for a short time.

Qustion

- The conditions, 'If r3 > t3' or 't3 >= r3' over simplifies or ignores lots of other possibilities considering the nature of UDP traffic (eg. streaming applications).

Answer

- Perhaps while watching the movie, the flow slightly. But the normal traffic.

- This time we use to calculate packet per second to reduce false positives.