

# HEX Switch:

Hardware-assisted security extensions of OpenFlow

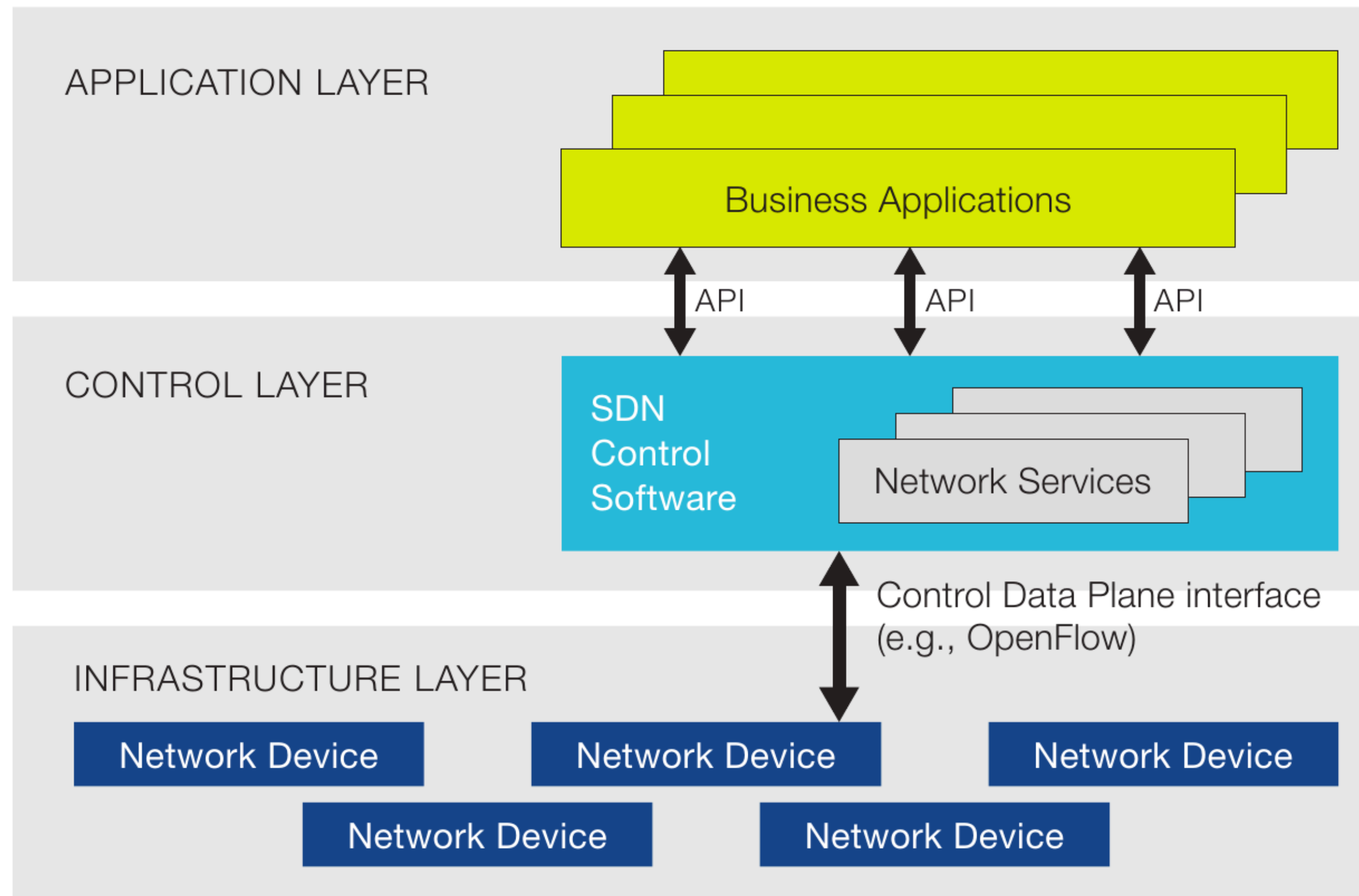
---

**Taejune Park / KAIST / [taejune.park@kaist.ac.kr](mailto:taejune.park@kaist.ac.kr)**

Zhaoyan Xu / StackRox Inc. / [z@stackrox.com](mailto:z@stackrox.com)

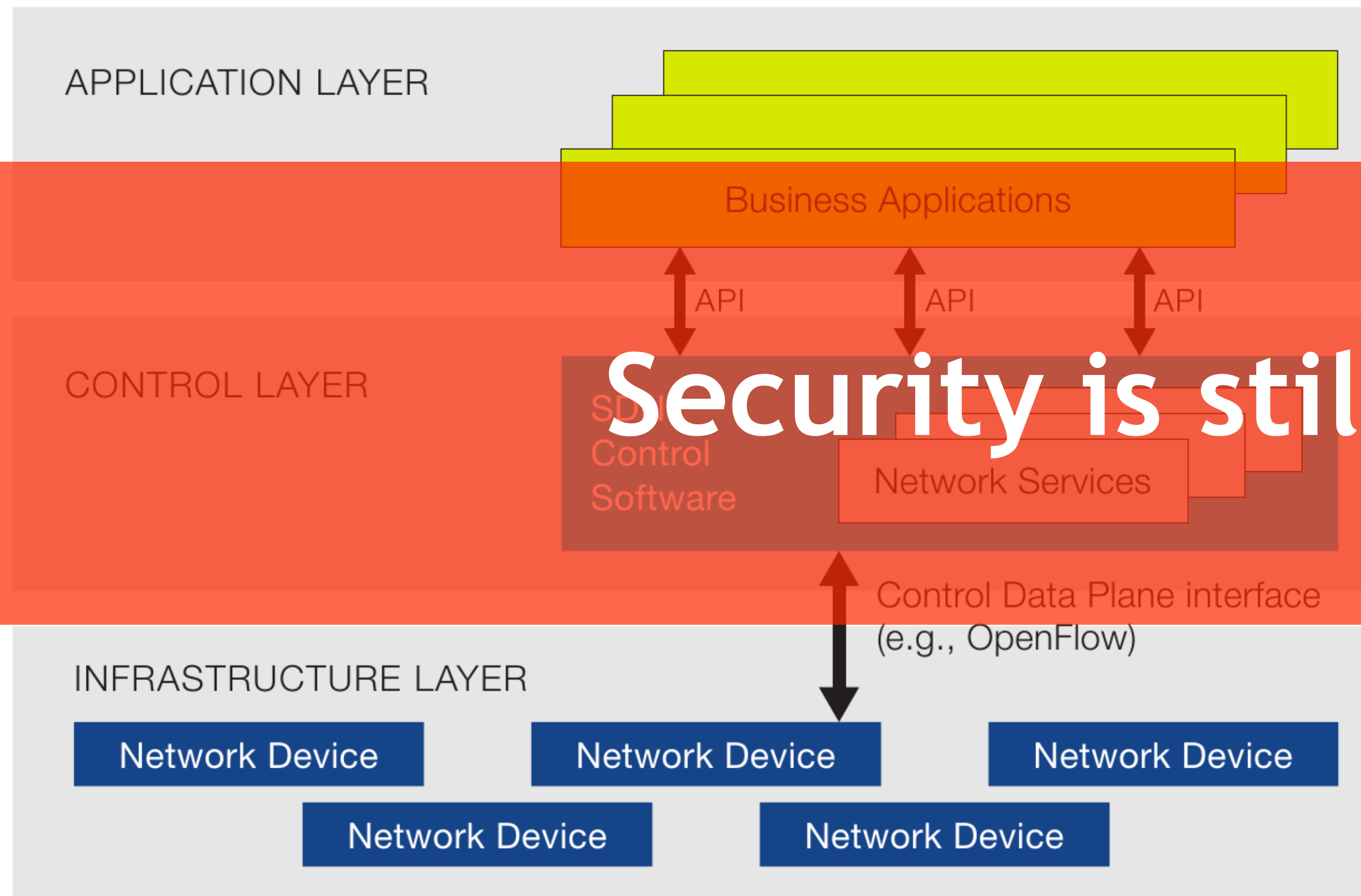
Seungwon Shin / KAIST / [claude@kaist.ac.kr](mailto:claude@kaist.ac.kr)

# Software-Defined Networking



- Centralized management
- Dynamic traffic engineering
- Programmable network operation
  
- **High-compatibility with virtualized environments**

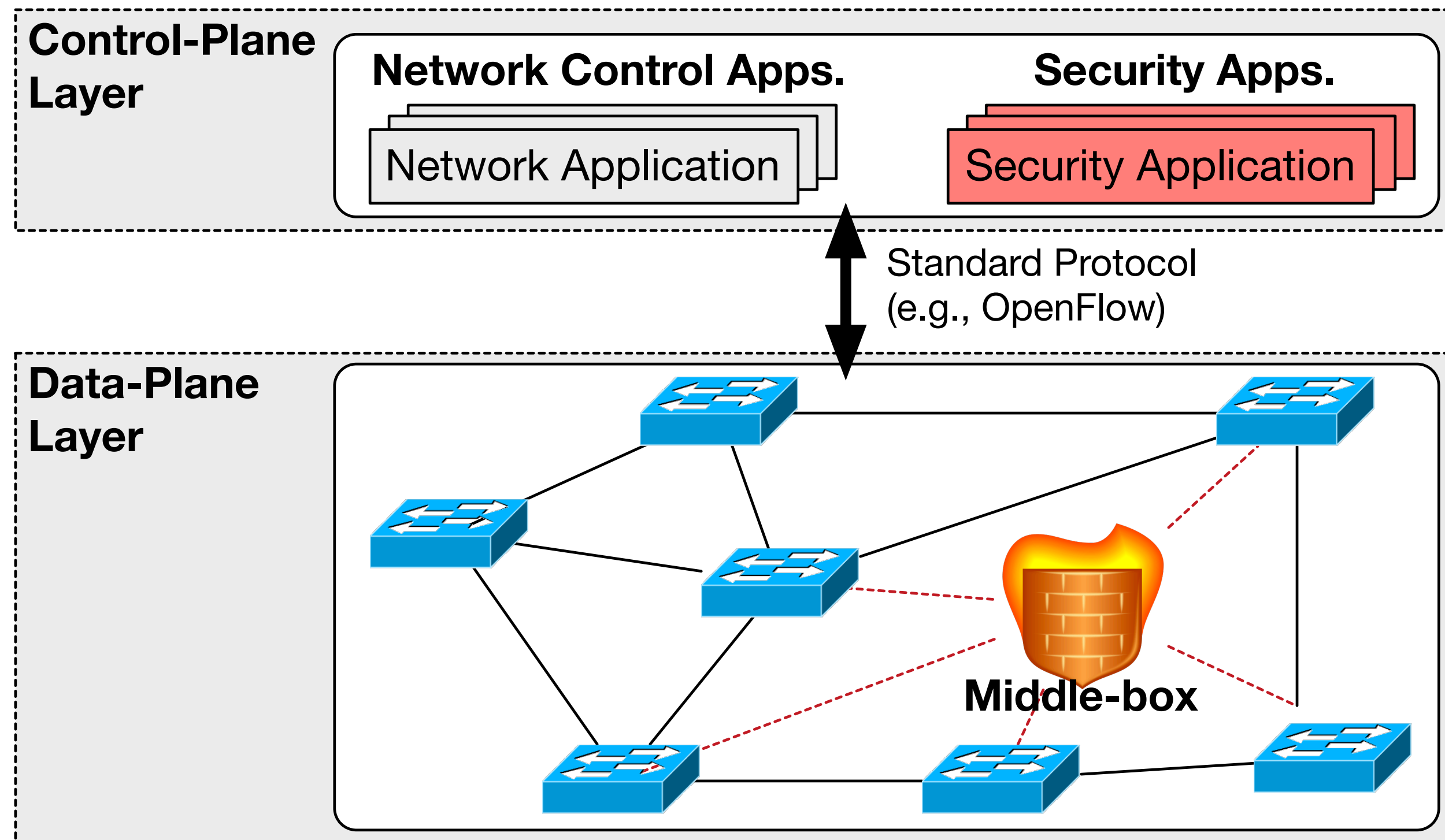
# Software-Defined Networking



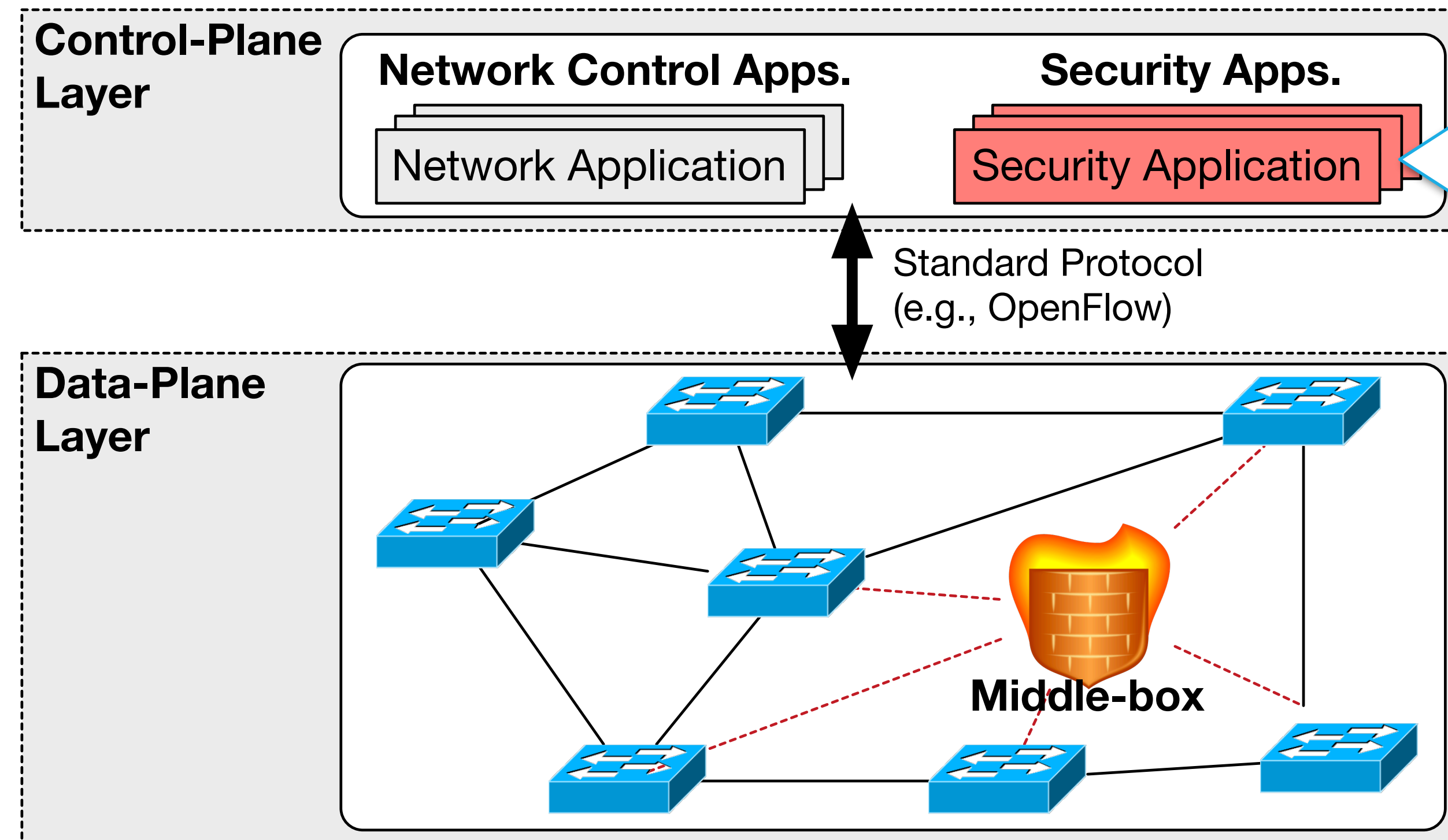
**Security is still required**

- Centralized management
- Dynamic traffic engineering
- Programmable network operation
- High-compatibility with virtualized environments

# Security in Software-Defined Networking

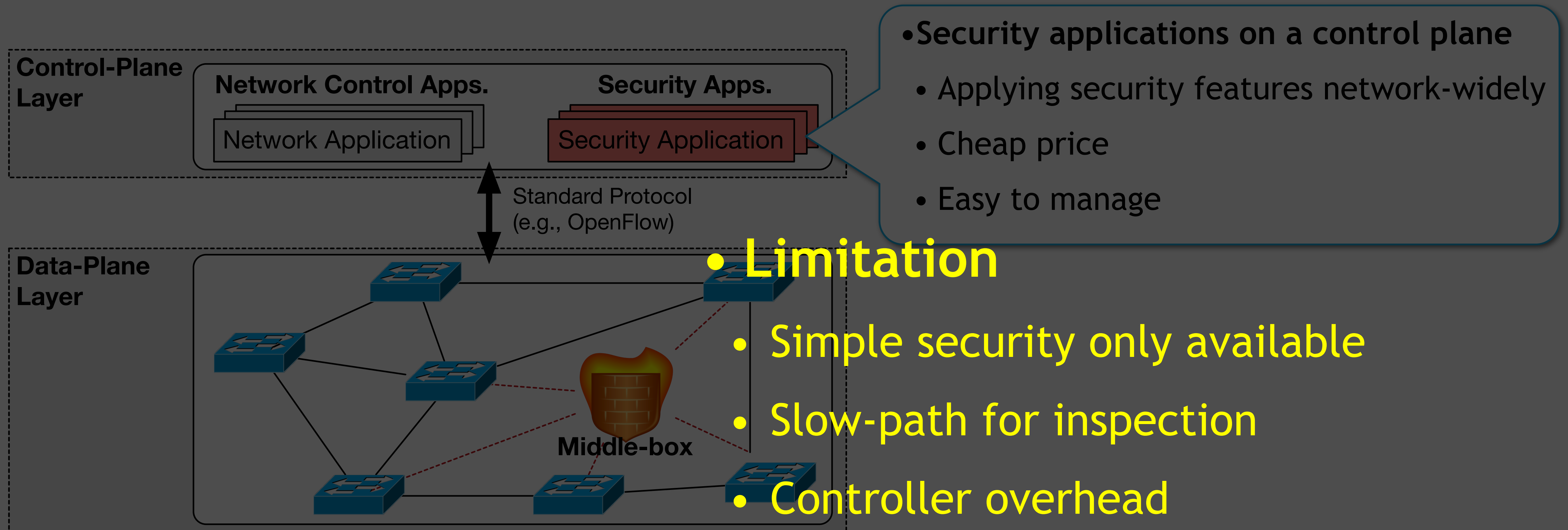


# Security in Software-Defined Networking

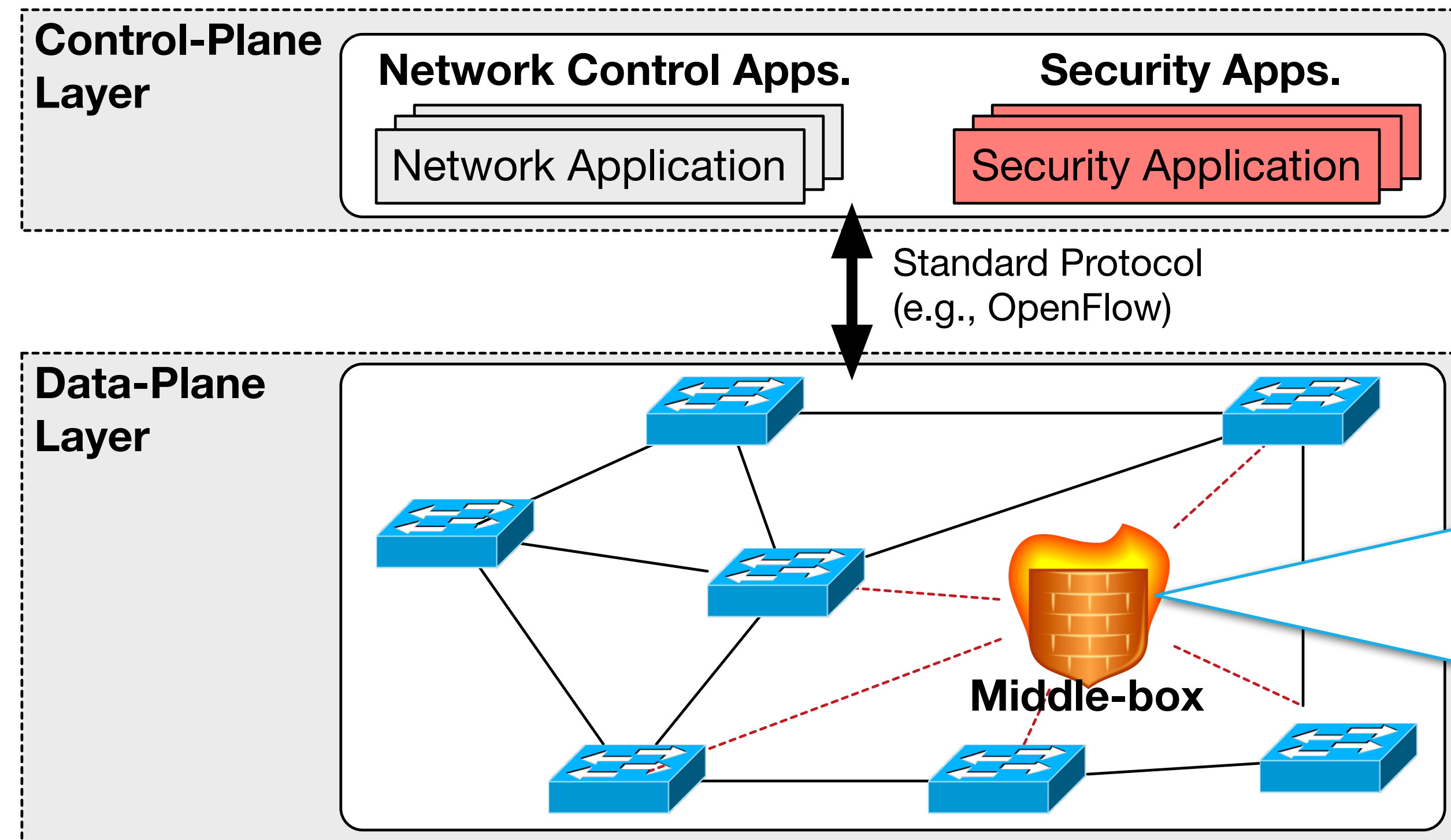


- Security applications on a control plane
  - Applying security features network-widely
  - Cheap price
  - Easy to manage

# Security in Software-Defined Networking

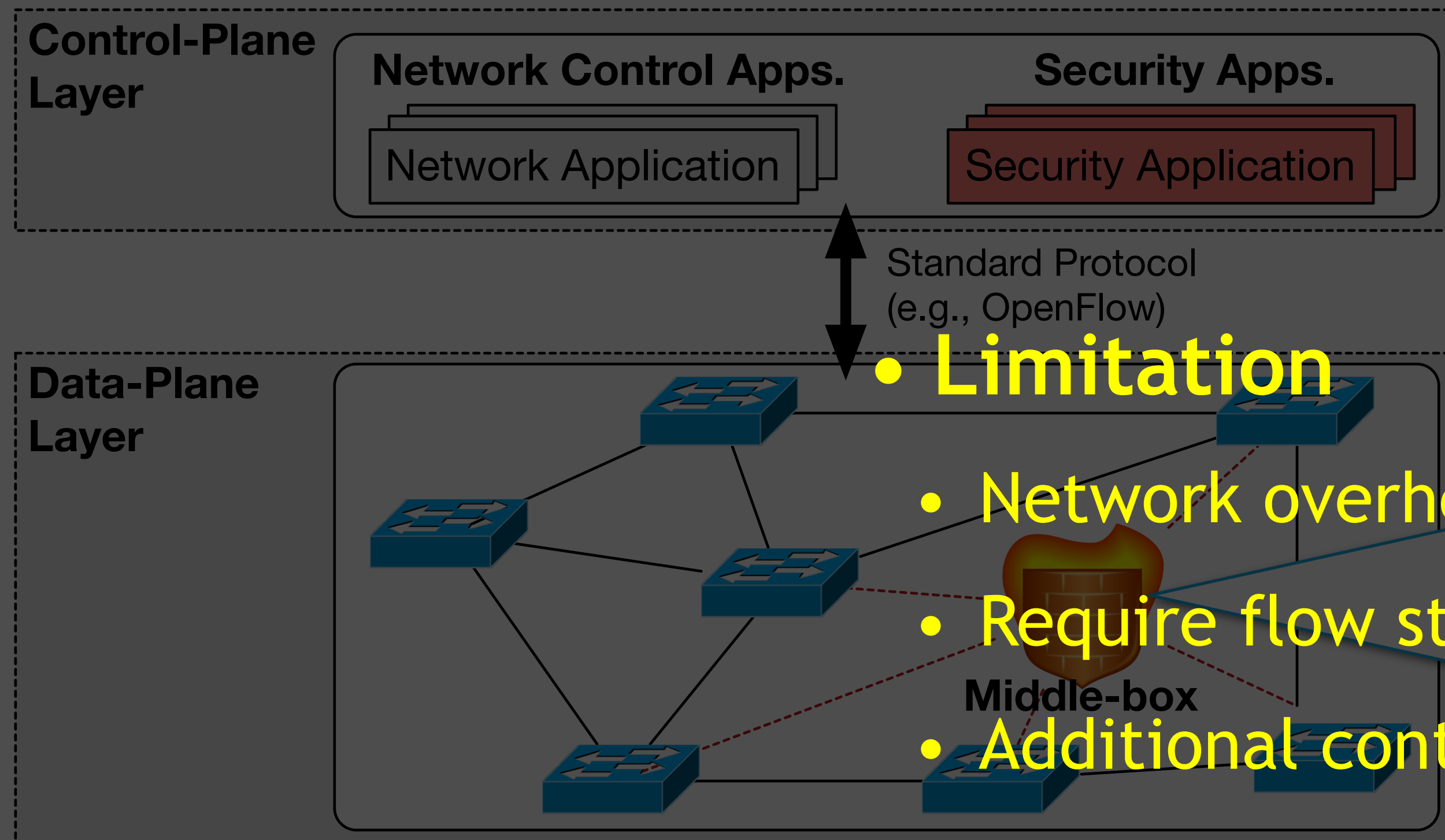


# Security in Software-Defined Networking



- **Middle-boxes on a data plane**
  - Better performance
  - Rich features such as payload inspection
  - No controller overhead

# Security in Software-Defined Networking



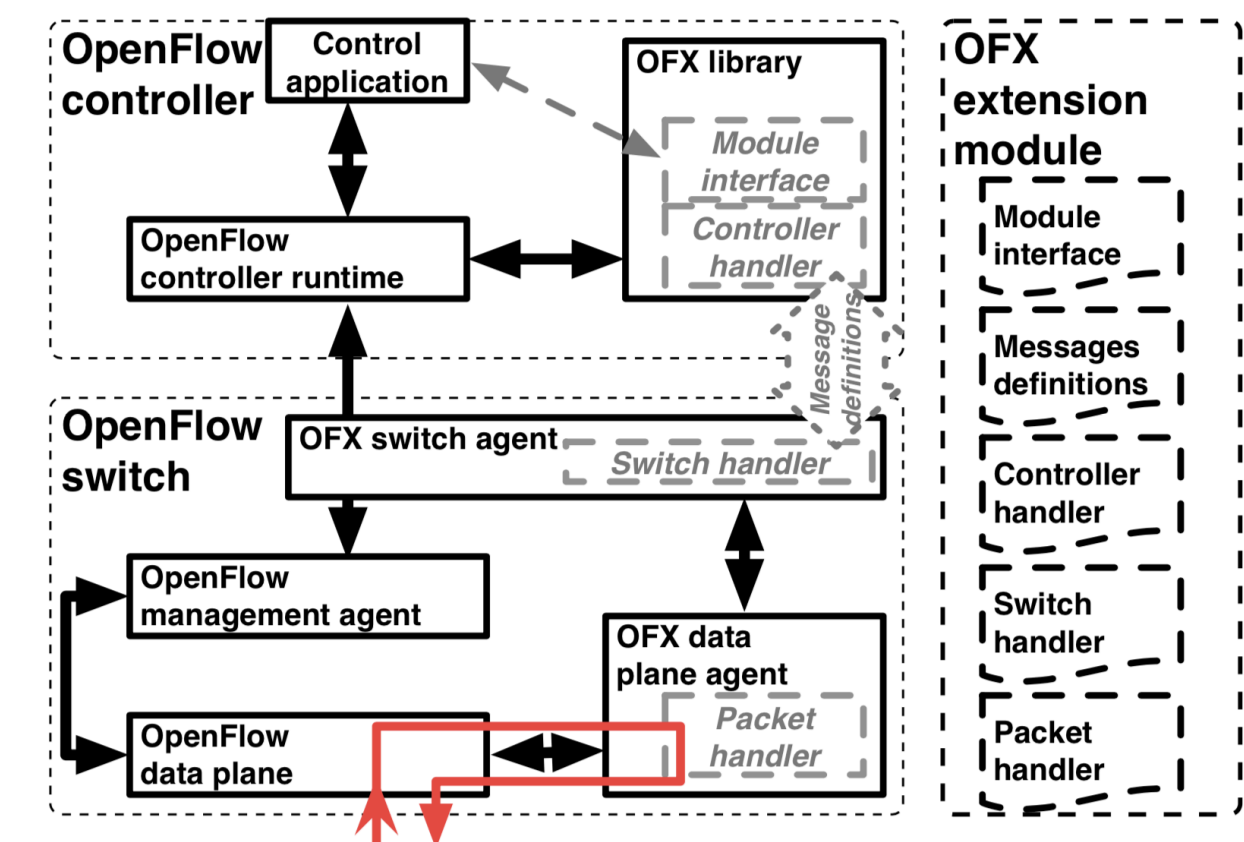
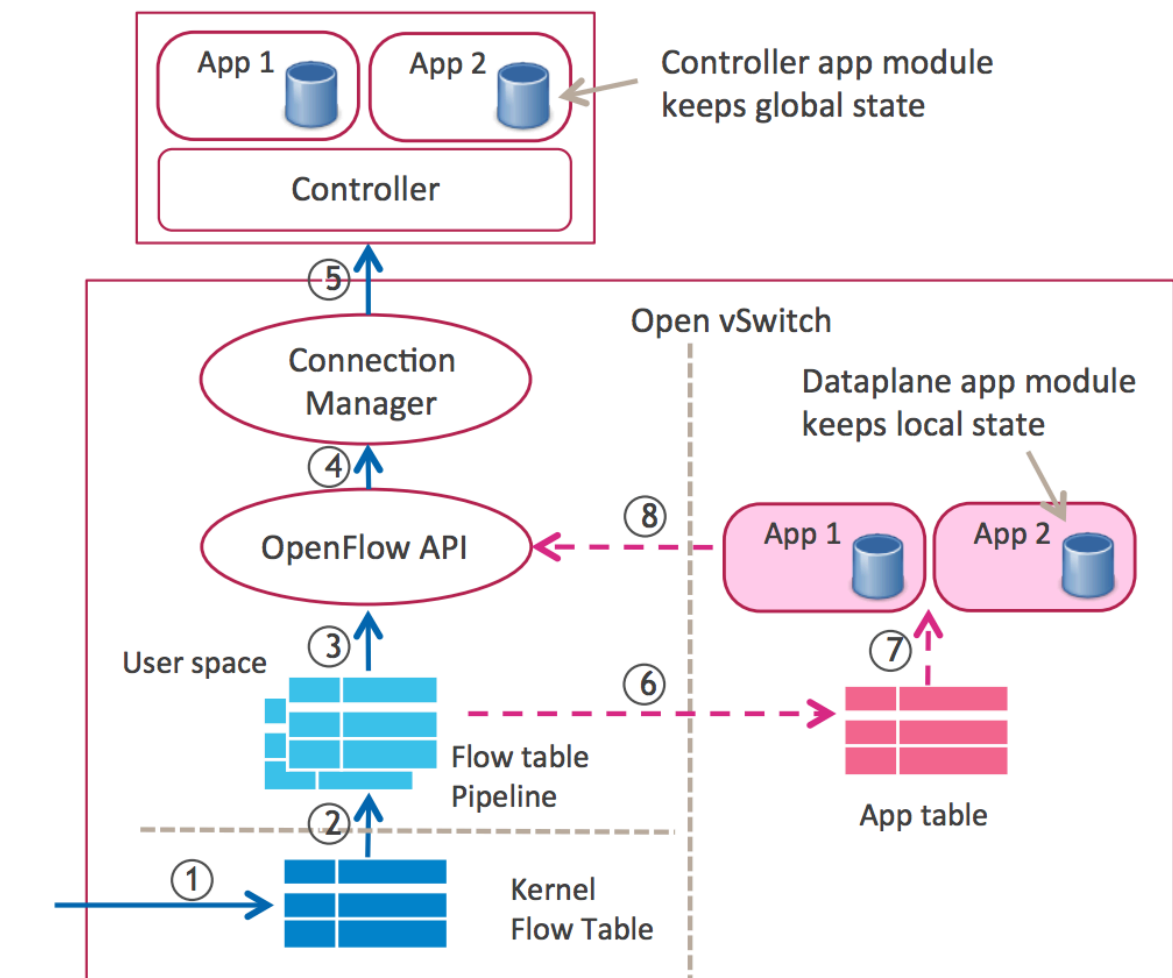
- Middle-boxes on a data plane
  - Better performance
  - Rich features such as payload inspection
  - No controller overhead
- Network overhead by traffic detouring (Taking extra hops)
- Require flow steering for NFs
- Additional control channels for NFs

# Summary

Category	SDN Applications	Middle-boxes
Flexibility	✓	✗
Management	✓	✗
Deployability	✓	✗
Performance	✗	✓
Functionality	✗	✓

# Related works: Extending SDN architecture to support security

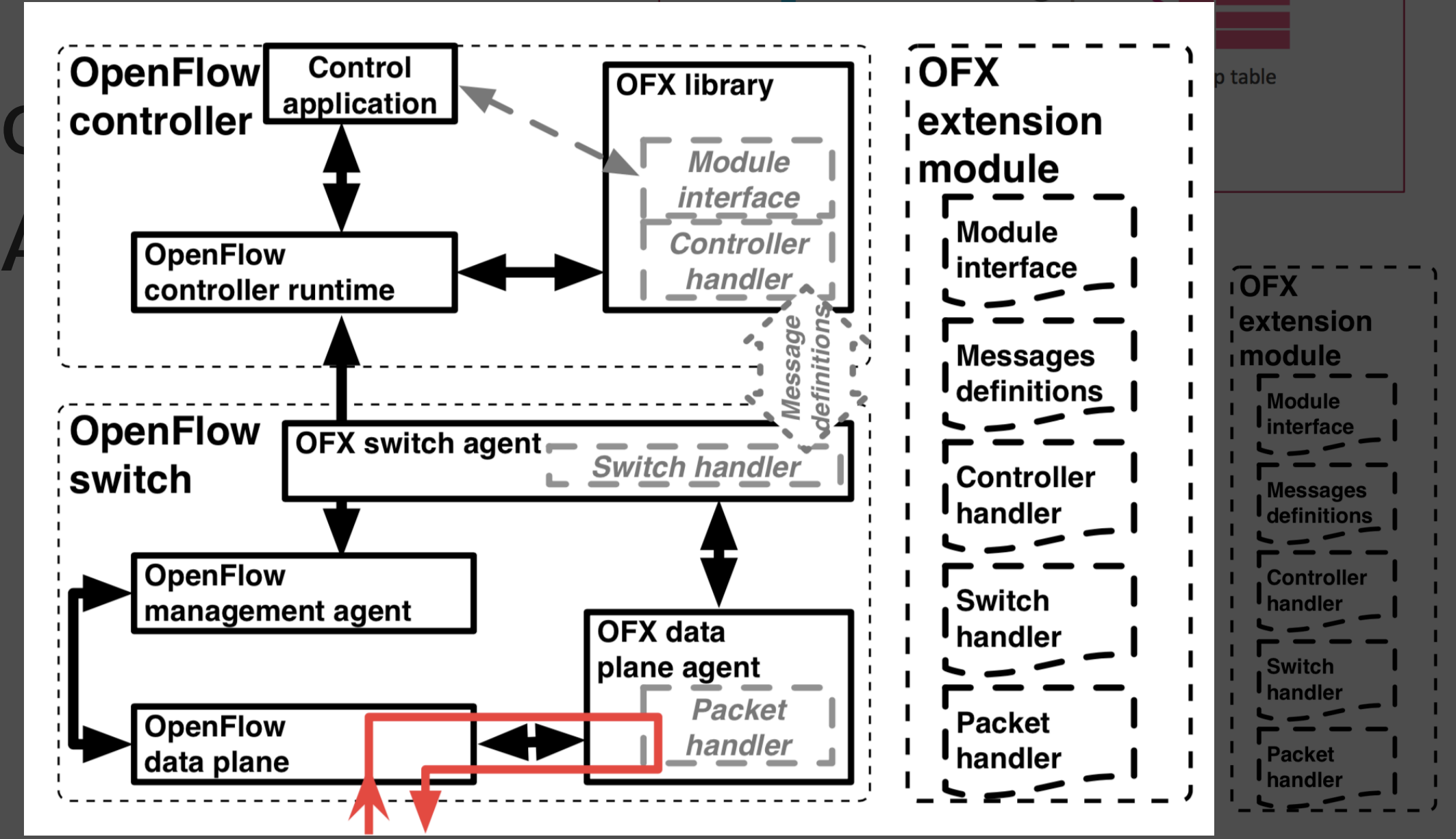
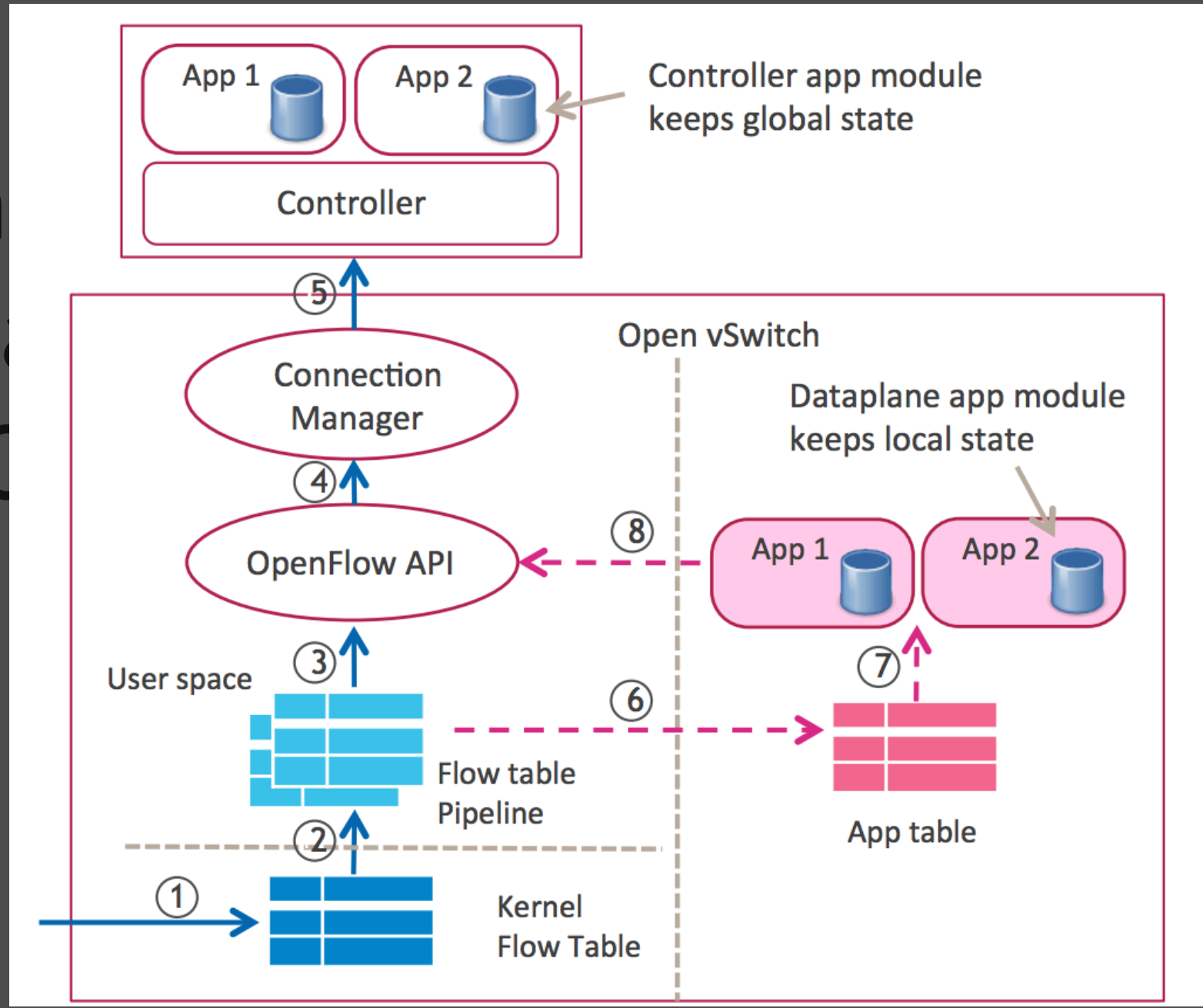
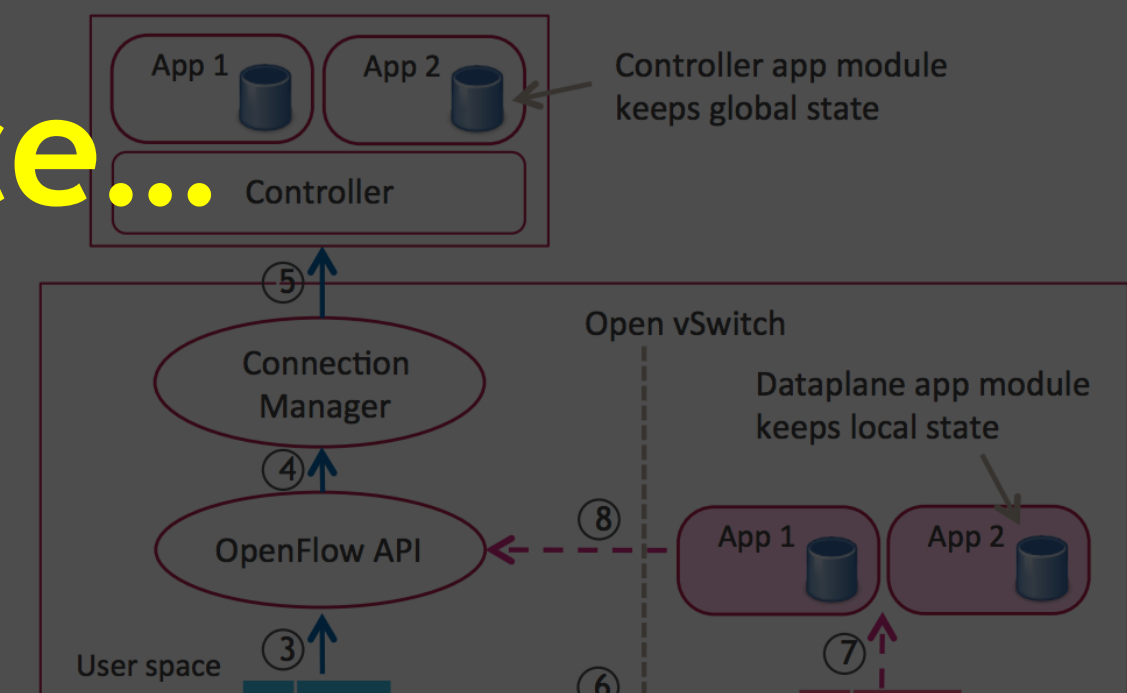
- Mekky, Hesham, et al. "Network function virtualization enablement within SDN data plane." *IEEE INFOCOM 2017 (Also, HotSDN 2014)*
- Sonchack, John, et al. "Enabling Practical Software-defined Networking Security Applications with OFX." *NDSS 2016*.



Related works:

• Their security functions are not fully consolidated into a data plane  
**Extending SDN architecture to support security**

- Mekky, Hesham, et al. "Network function virtualization enablement within SDN data plane." *IEEE INFOCOM 2017 (Also, HotSDN 2014)*
- Application module, Tap-based interface...



- Sonch...
- Softw...
- with C...

Practic...  
 Security A...

Related works:

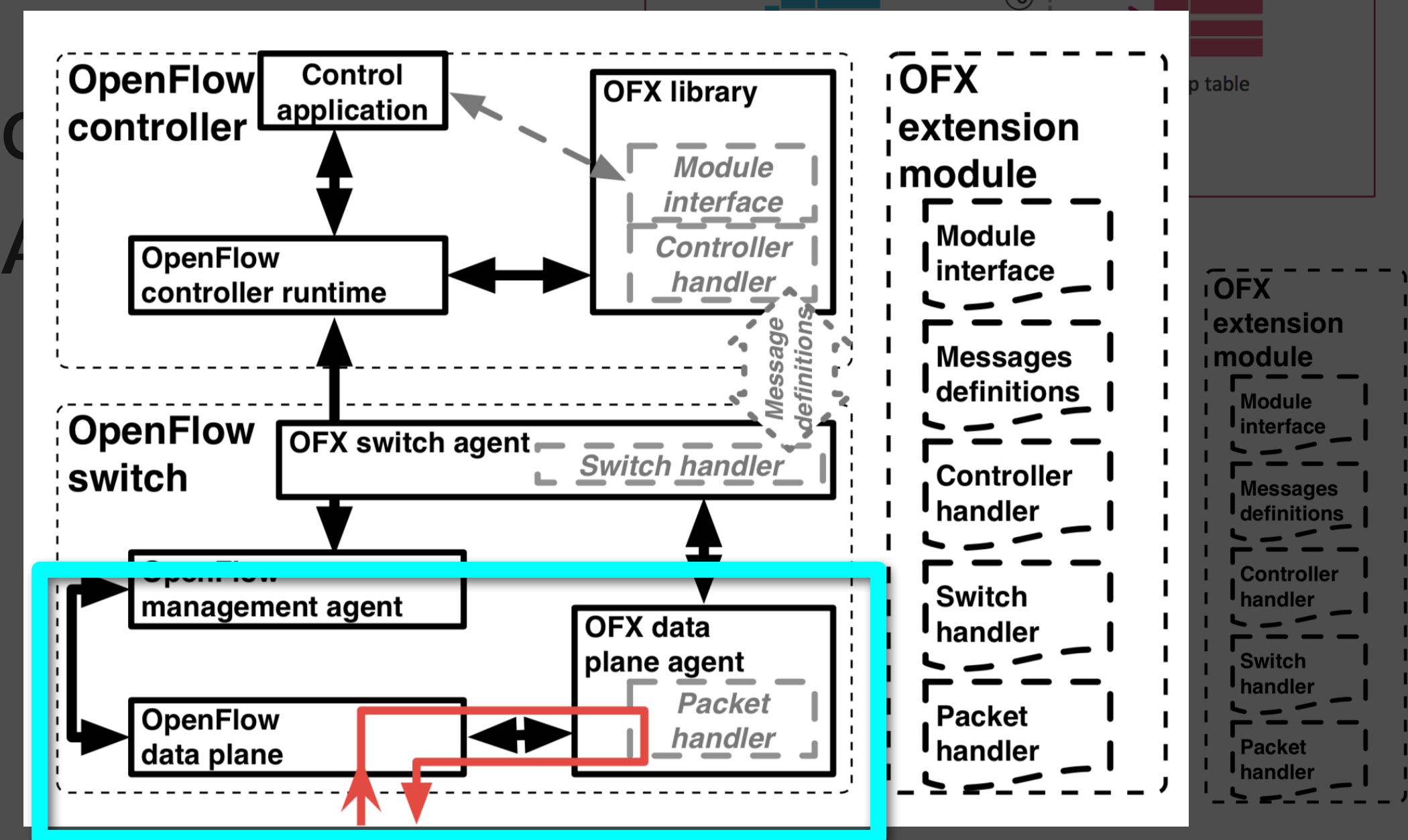
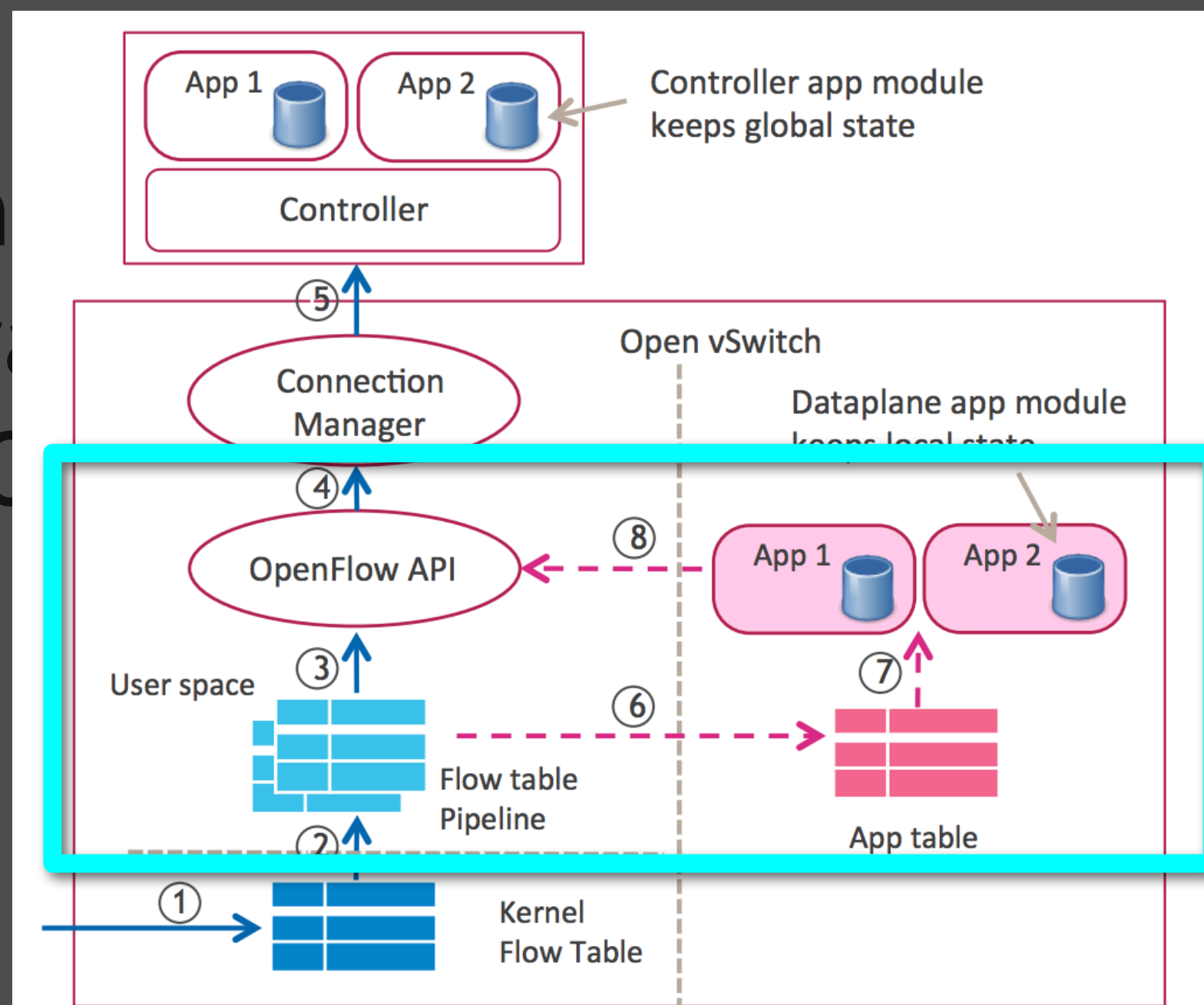
# Extending SDN architecture to support security

• In essence, they are NOT different from the middle-box structure!

- Mekky, Hesham, et al. "Network function virtualization enablement within SDN data plane."
- It's just a scale down!

IEEE INFOCOM 2017 (Also, HotSDN 2014)

- Sonch...
- Softw...
- with C...

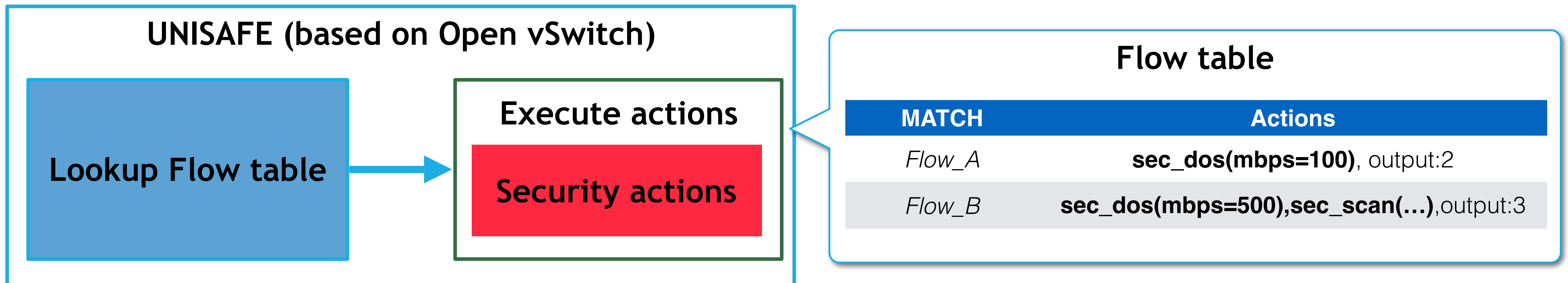


Related works:

## UNISAFE: A union of security actions for software switches

Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization. ACM, 2016

- Fully integrated security functions into a data plane, *not modular one*
- Security functions as a set of OpenFlow actions



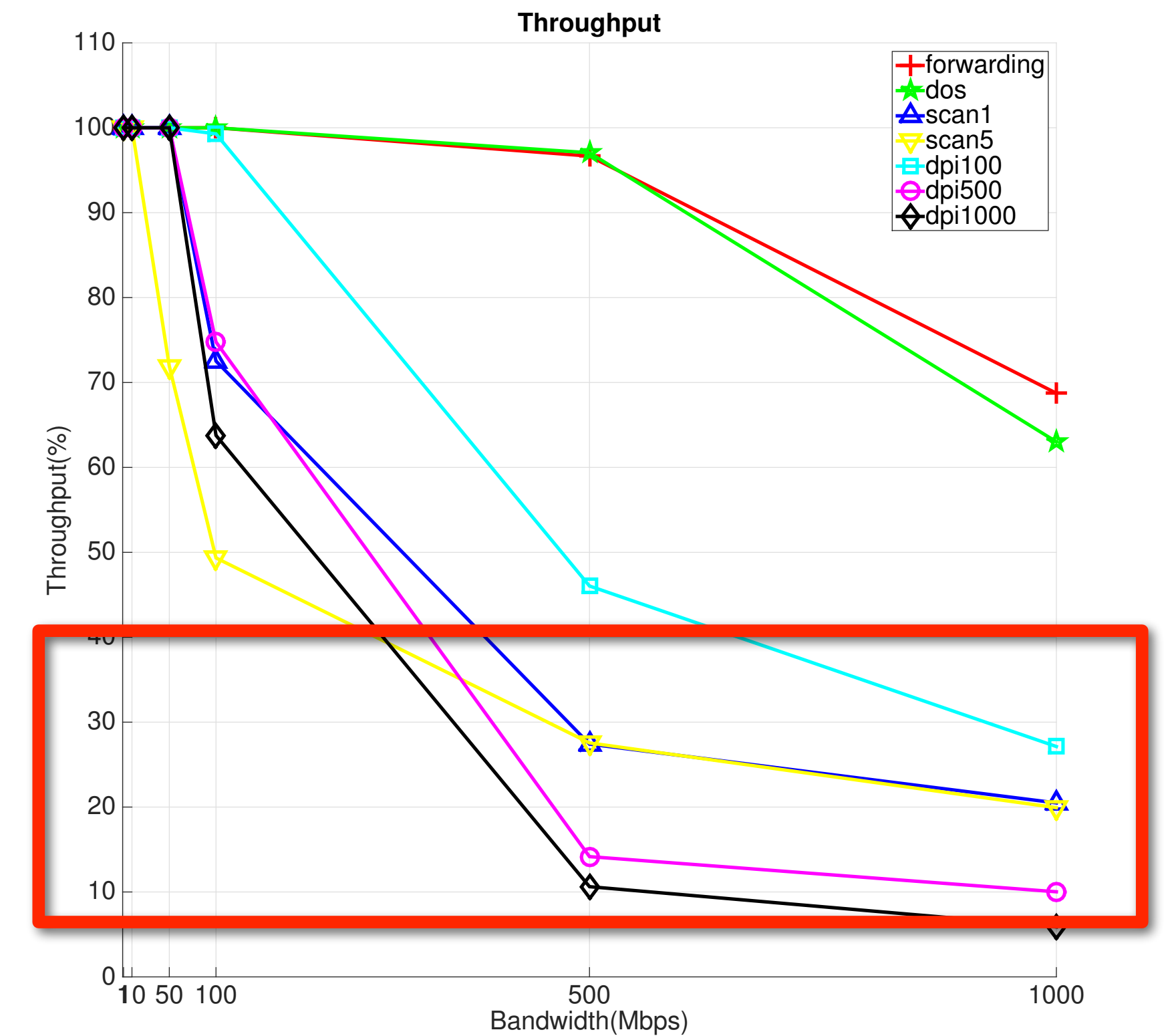
# Security actions of UNISAFE

---

- High-compatibility with common OpenFlow actions
  - *actions=sec\_dos(mbps=1000),set\_nw\_src(...),output:2*
- Fine-grained security enforcement per a flow
  - *in\_port=1,nw\_src=10.0.0.1,tp\_dst=80,actions=sec\_dos(...),...*
  - *in\_port=2,nw\_dst=10.0.1.2,actions=sec\_dpi(...),...*
- Easy configuration for a security service chaining
  - *actions=sec\_dos(...),sec\_scan(...),sec\_dpi(...),...*

# Performance in UNISAFE

- Achieve line-rate latency for all security
- But, lack of throughput in some actions
  - Payload Inspection (DPI) throughput
    - Throughput less than 100Mbps on 1Gbps



# Performance in UNISAFE

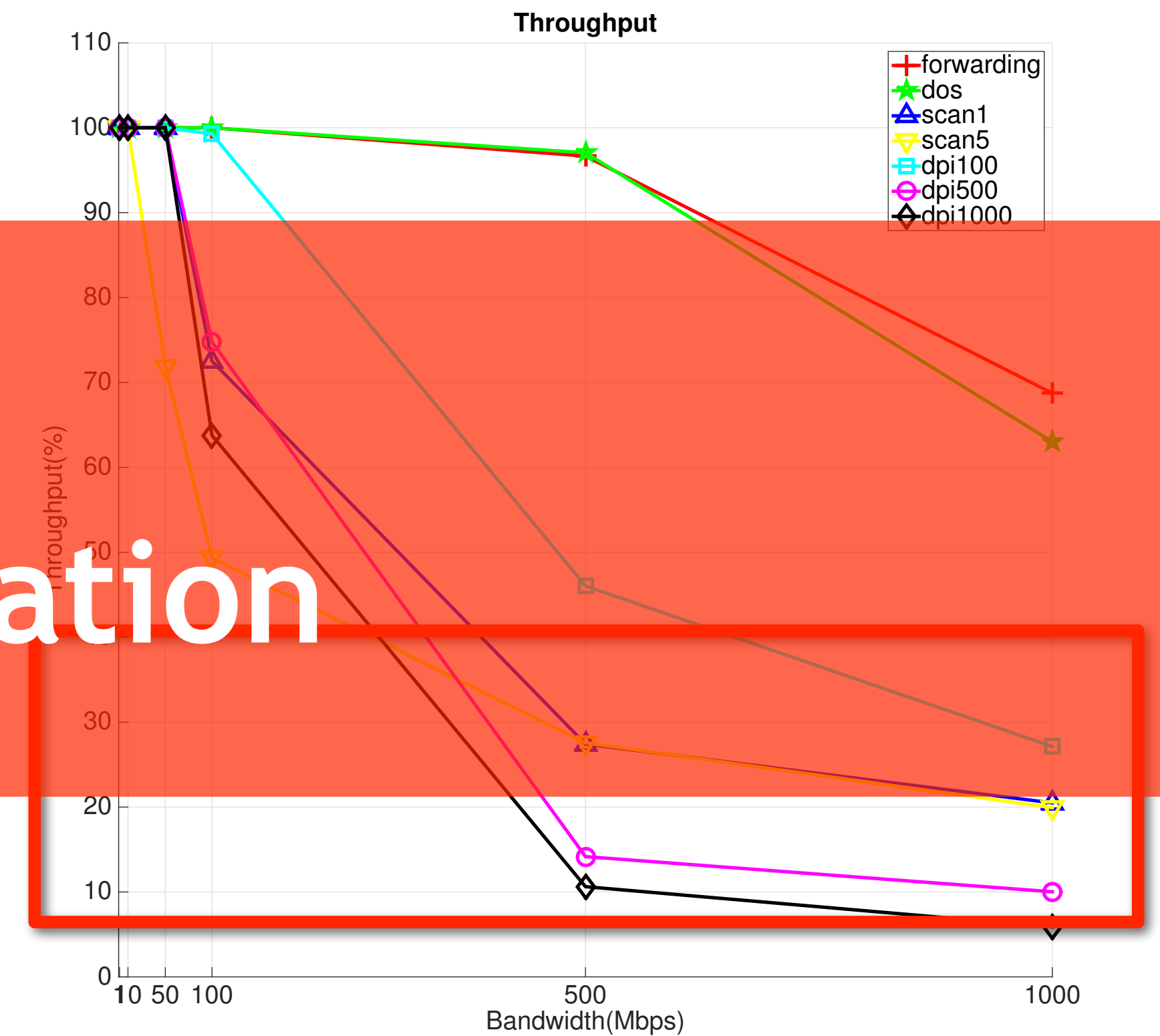
- Achieve line-rate latency for all security

- But, lack of throughput in some actions

- Payload Inspection (DPI) throughput

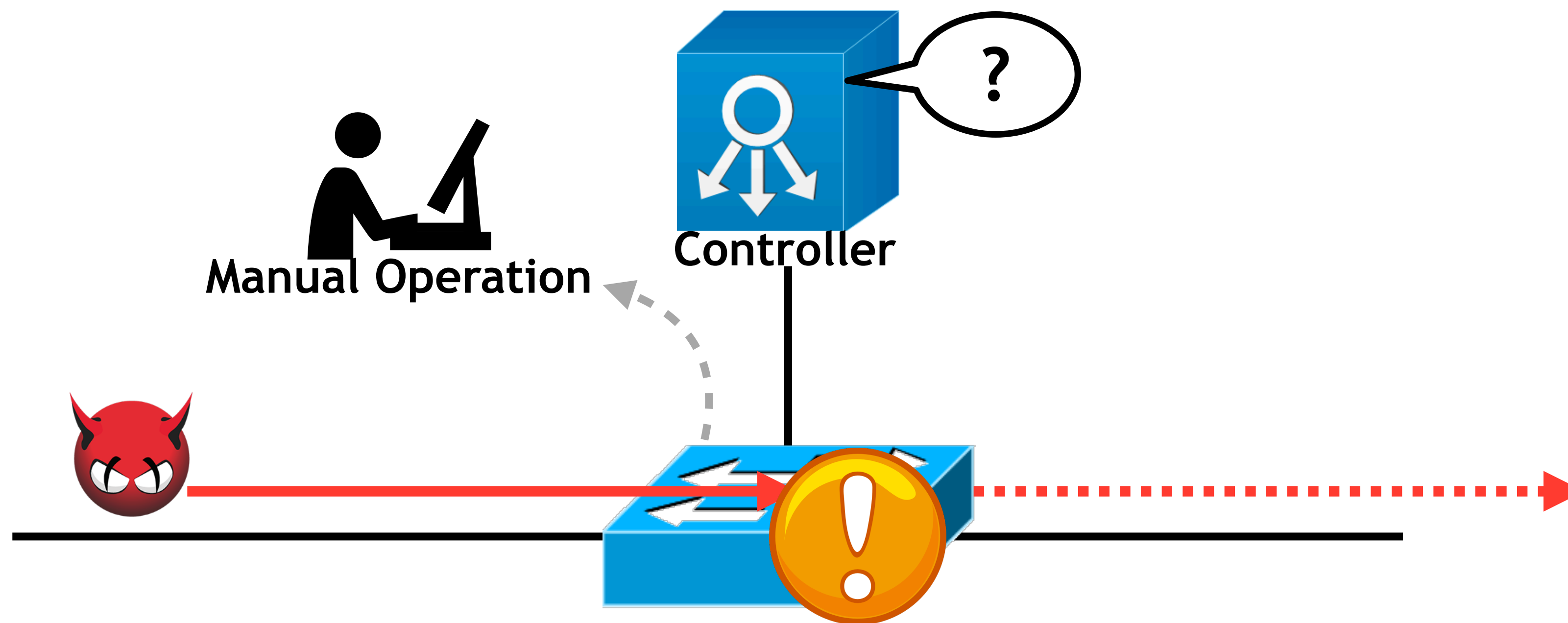
- Throughput less than 100Mbps or 1 Gbps

## Challenge 1: Performance limitation



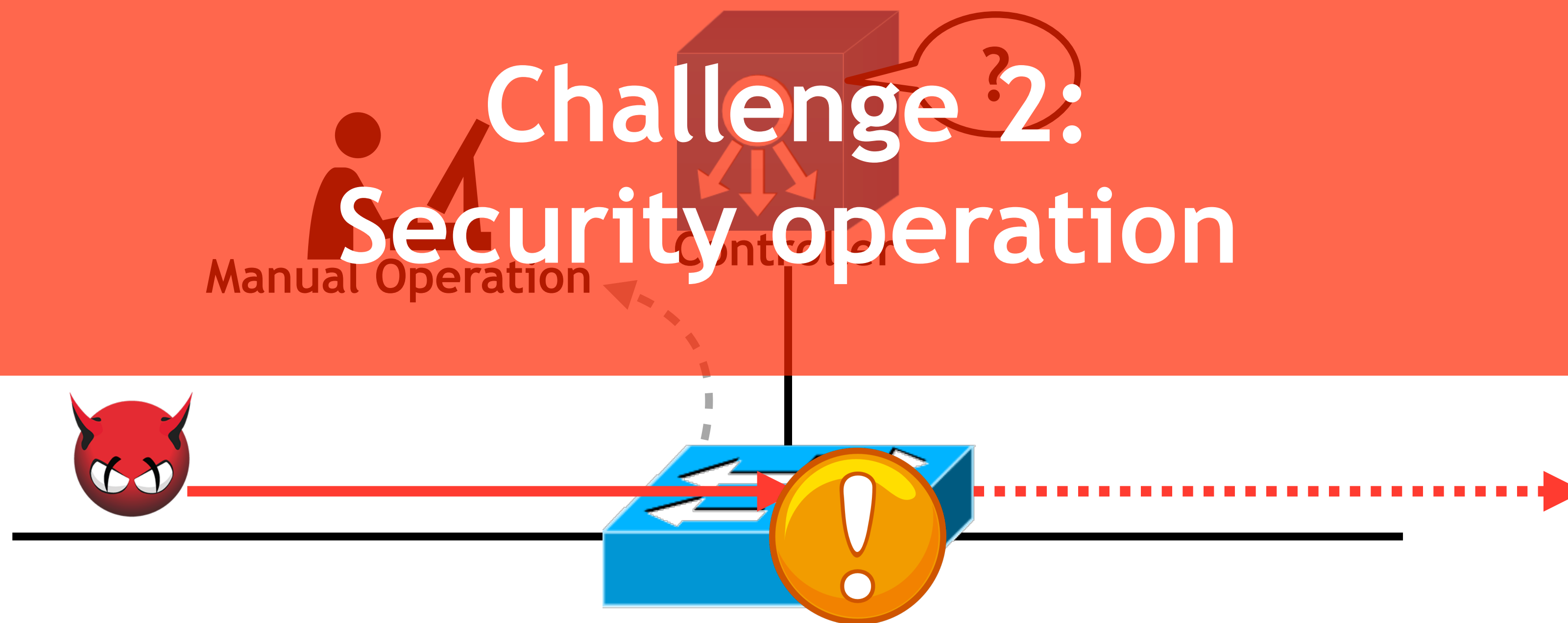
# Security operation in UNISAFE

- Manual operation for security violations by an administrator



# Security operation in UNISAFE

- Manual operation for security violations by an administrator

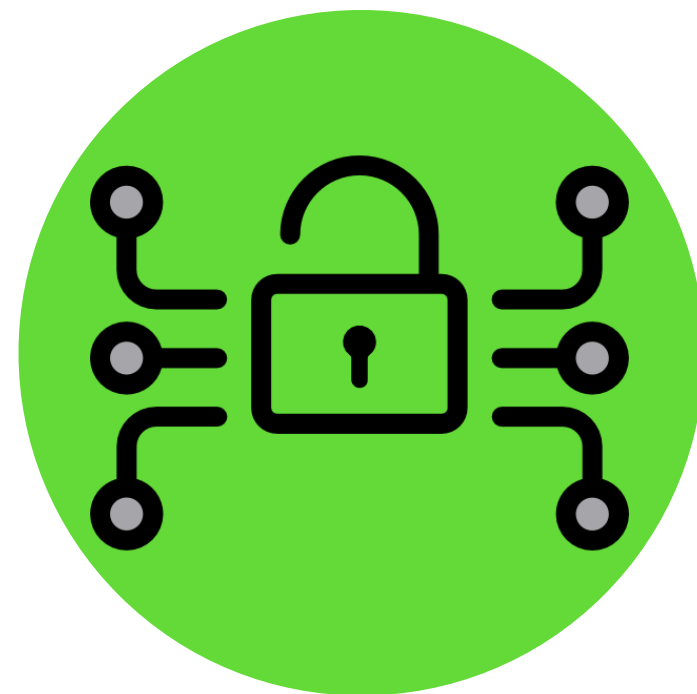


# HEX Switch:

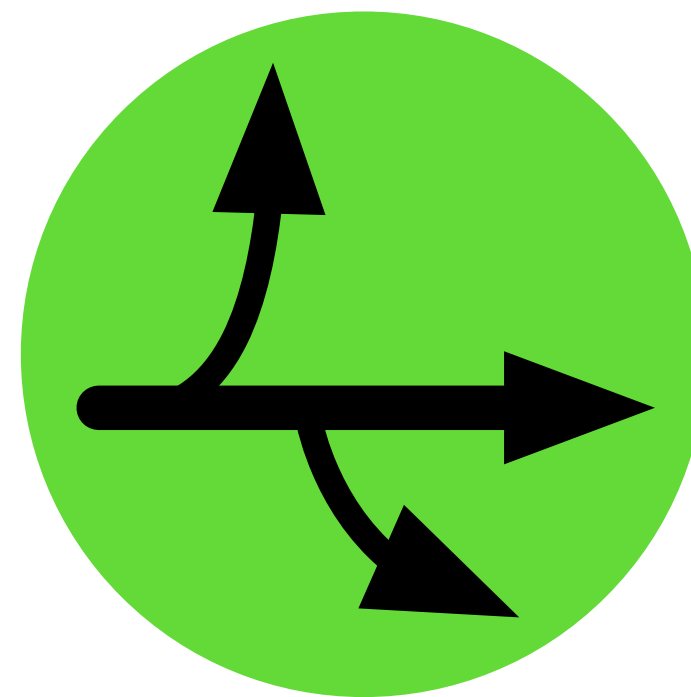
## Hardware-assisted security extensions of OpenFlow

---

- Hardware-based approach for UNISAFE
  - Using NetFPGA
  - Providing line-rate performance with configurability



Security Actions



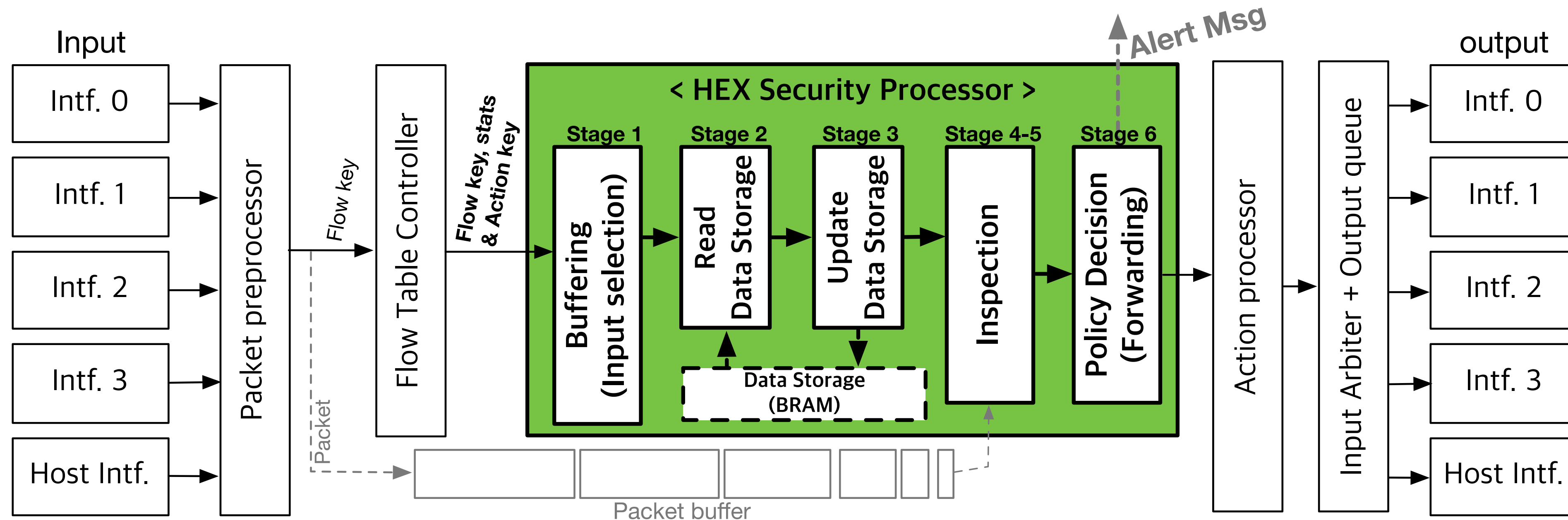
Security Policy



Controller  
communication

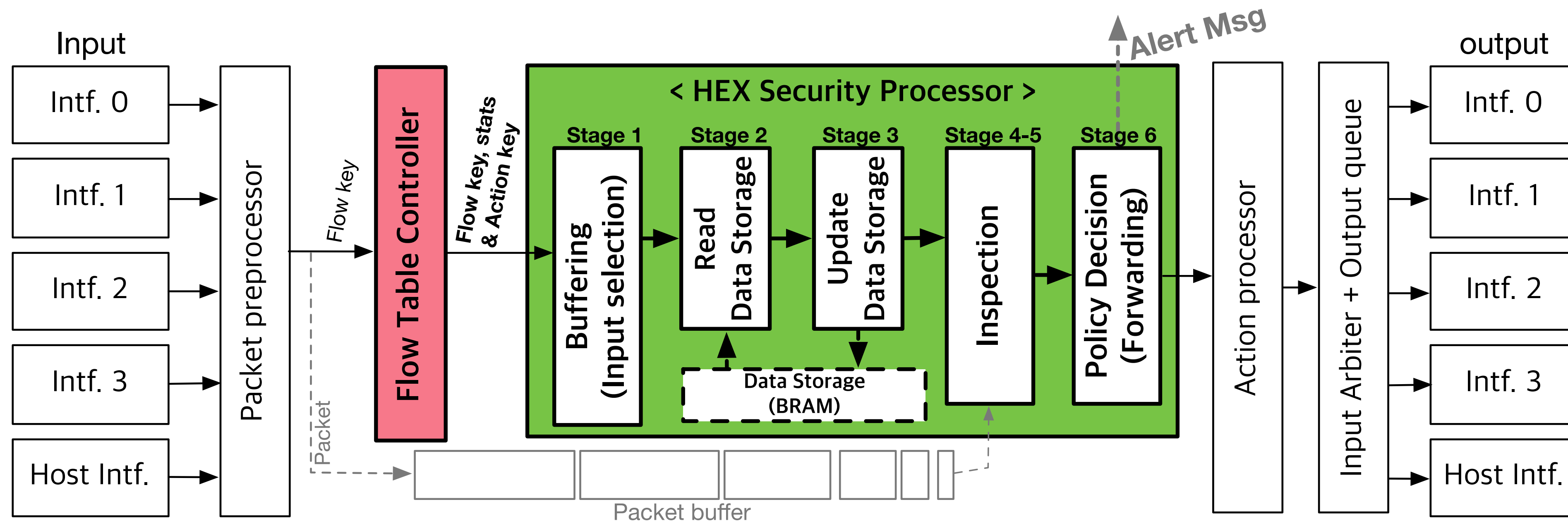
# Design

- Security Processor between the packet processing sequence.
  - Six-stages pipeline: Mainly consist of **data storage** and **inspection logic**
  - Flow table controller forwards *flow keys, stats and action key* after matching



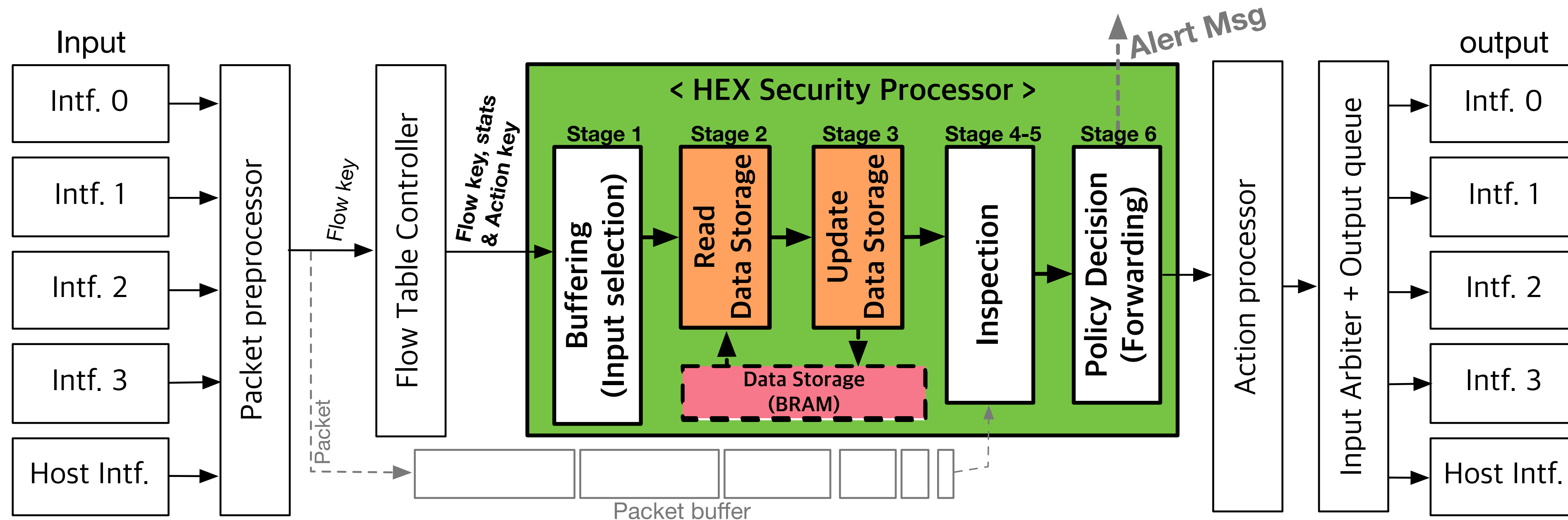
# Design

- Security Processor between the packet processing sequence.
  - Six-stages pipeline: Mainly consist of **data storage and inspection logic**
  - Flow table controller forwards *flow keys, stats and action key* after matching



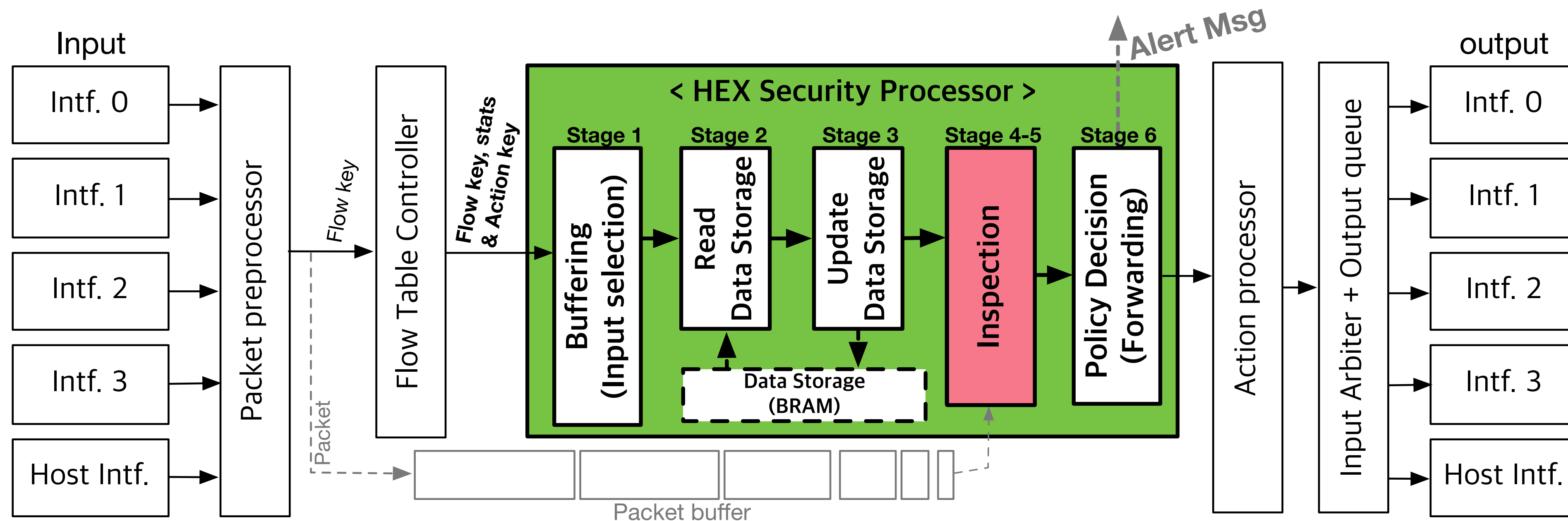
# Design

- Security Processor between the packet processing sequence.
  - Six-stages pipeline: Mainly consist of **data storage** and **inspection logic**
  - Flow table controller forwards *flow keys, stats and action key* after matching



# Design

- Security Processor between the packet processing sequence.
  - Six-stages pipeline: Mainly consist of **data storage** and **inspection logic**
  - Flow table controller forwards *flow keys, stats and action key* after matching



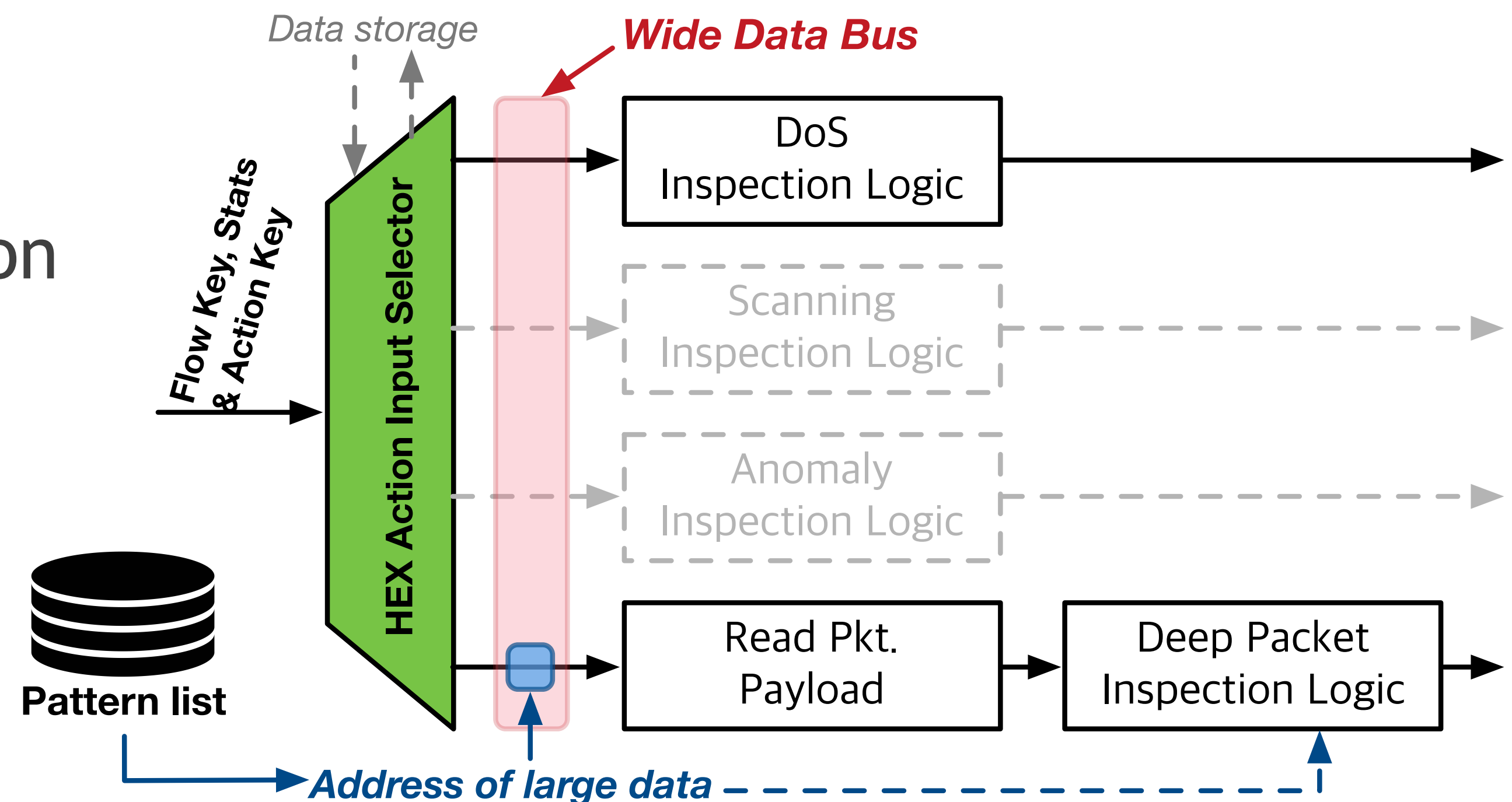
# Security Action Processing

- All security actions are performed in parallel
  - Forward the data storage data to inspection logic through the wide data bus.

- **Challenge**

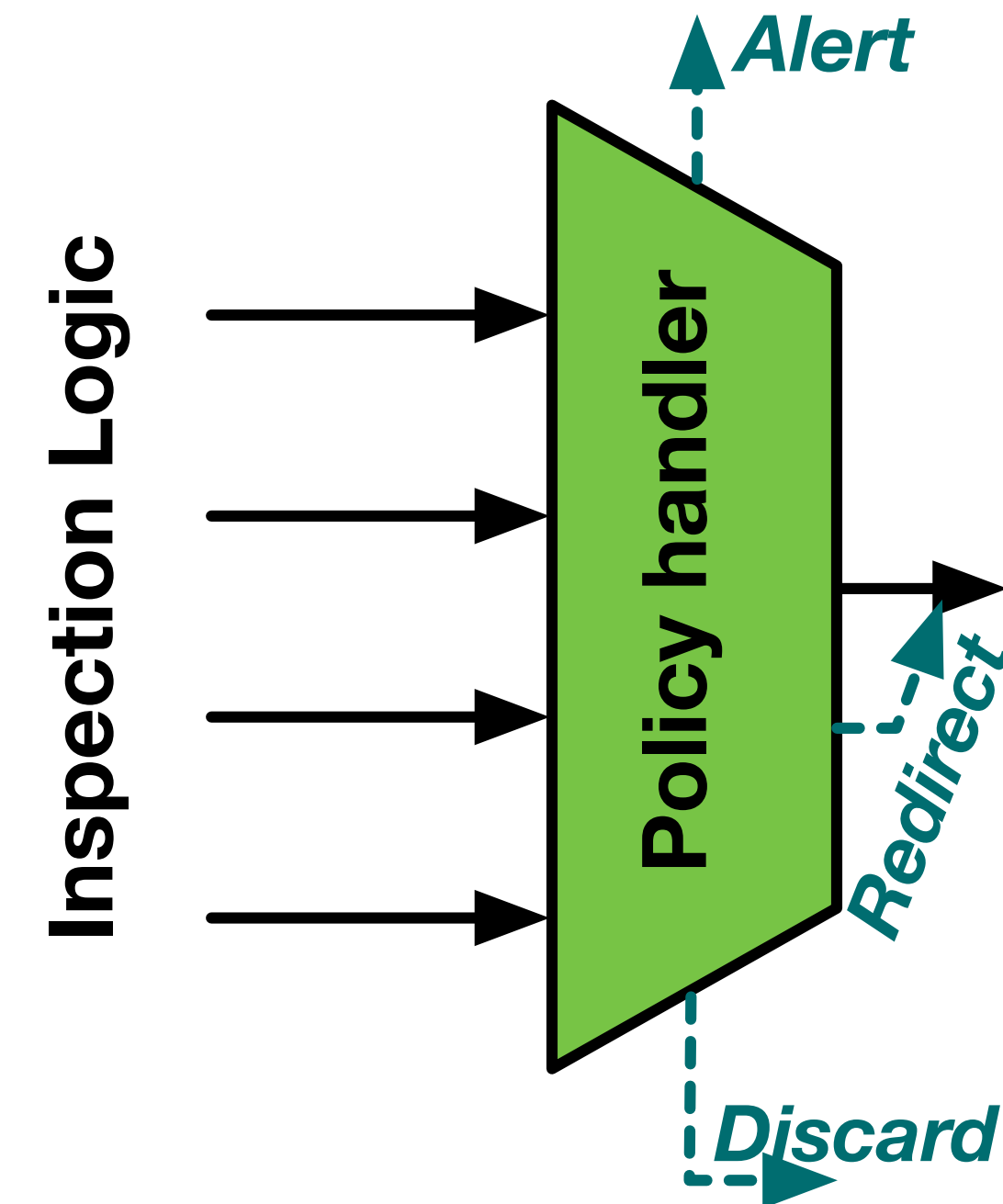
- Pattern list for payload inspection requires width bandwidth

=> *Transfer the address first and read directly memory*



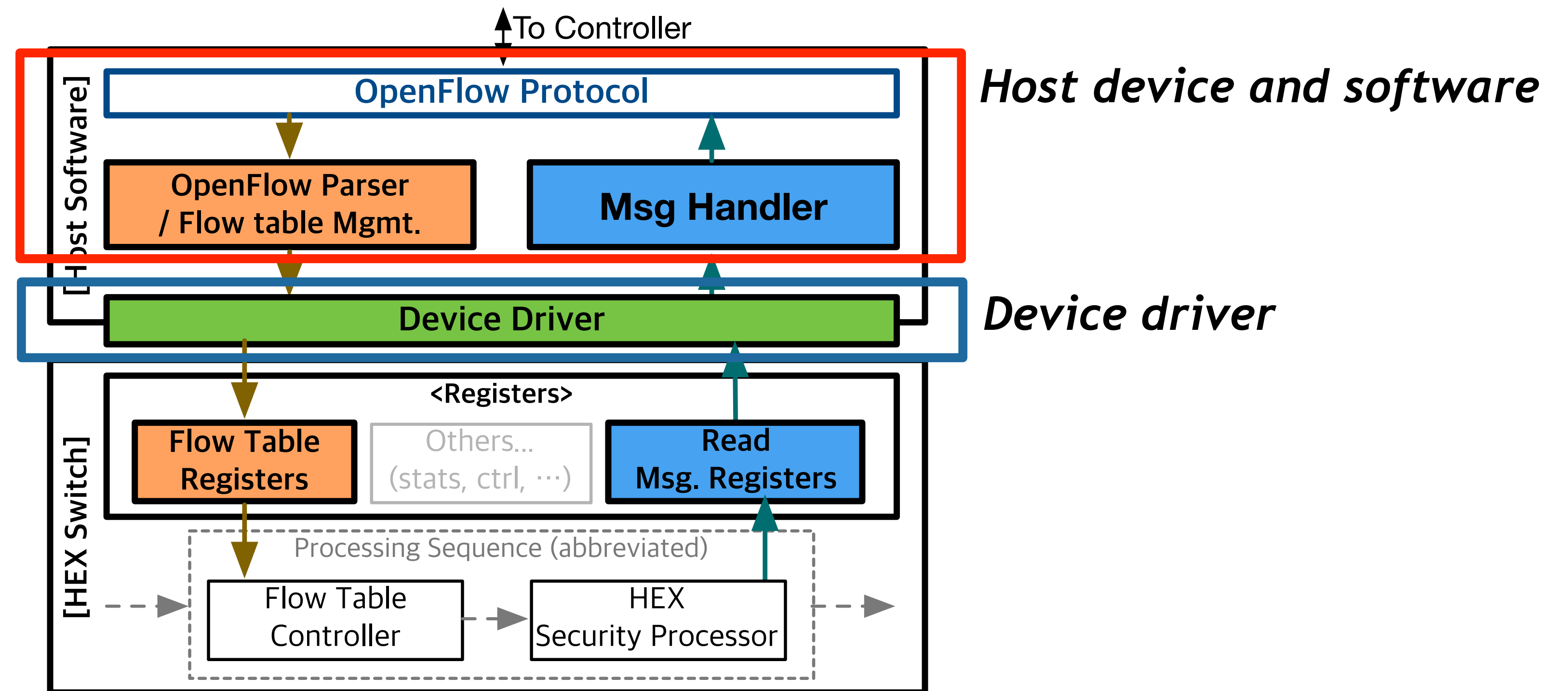
# After Processing: Applying Policy

- Actions can handle violated packets according to a policy
  - *e.g., `actions=sec_dos(mbpps=1000,policy=redirect:2)`*  
=> If the current bps exceeds 1000 Mbps, redirect the flow to port 2.
- Four polices
  - *Neglect: Ignores the violation*
  - *Alert: Send an alert msg to a controller*
  - *Discard: Terminates the packet processing and drop the packet*
  - *Redirect: Forward packets to an alternative port*



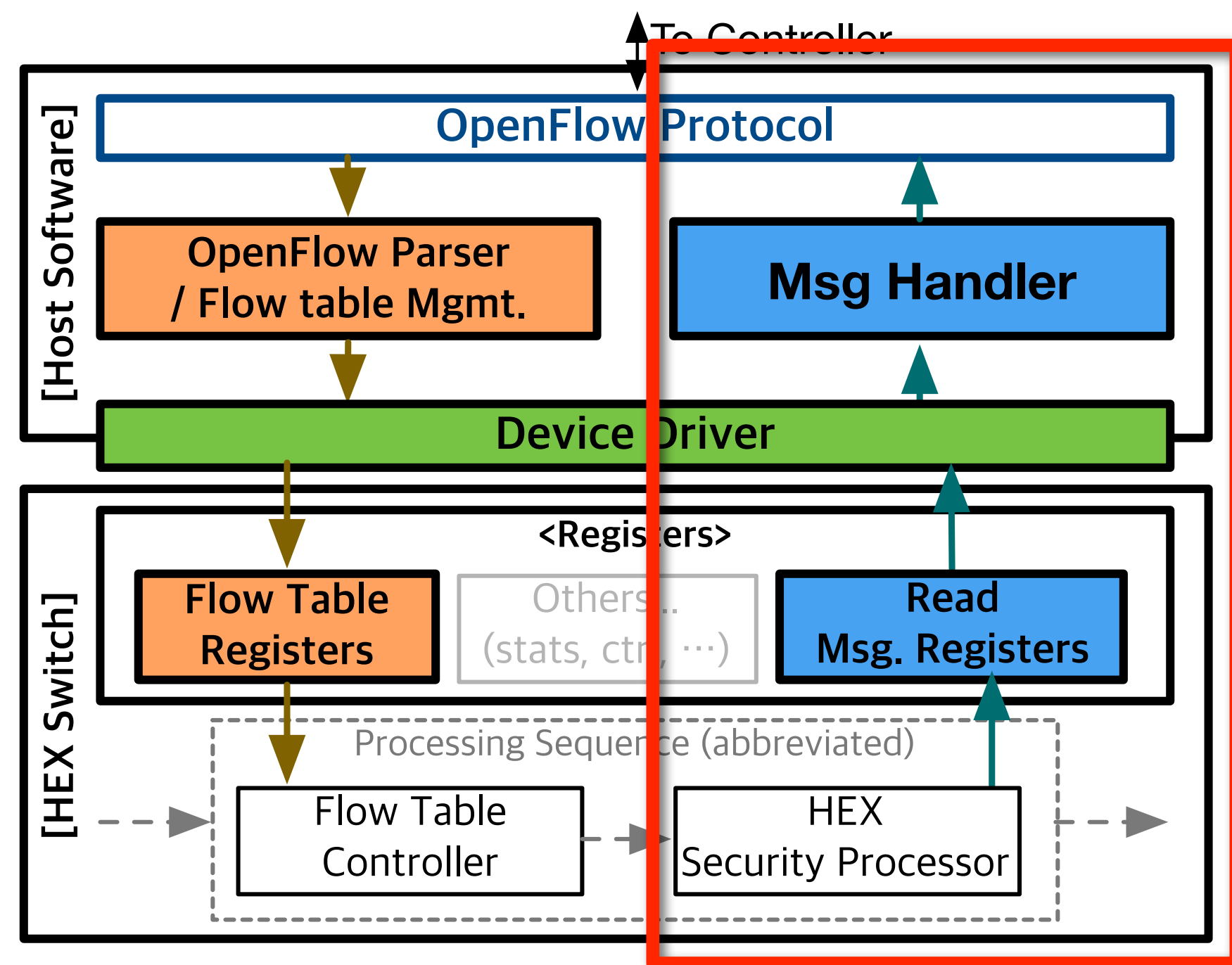
# Communication with a controller

- By the host device with its software
  - The host device and the HEX switch are bound by the device driver



# Communication with a controller: Transferring an alert message

- The device driver reads the registers and the HEX handler transfers it to a controller through an OpenFlow channel
- A controller provides a handler API to process the alert message



```
import pox.openflow.hexswitch as HEX

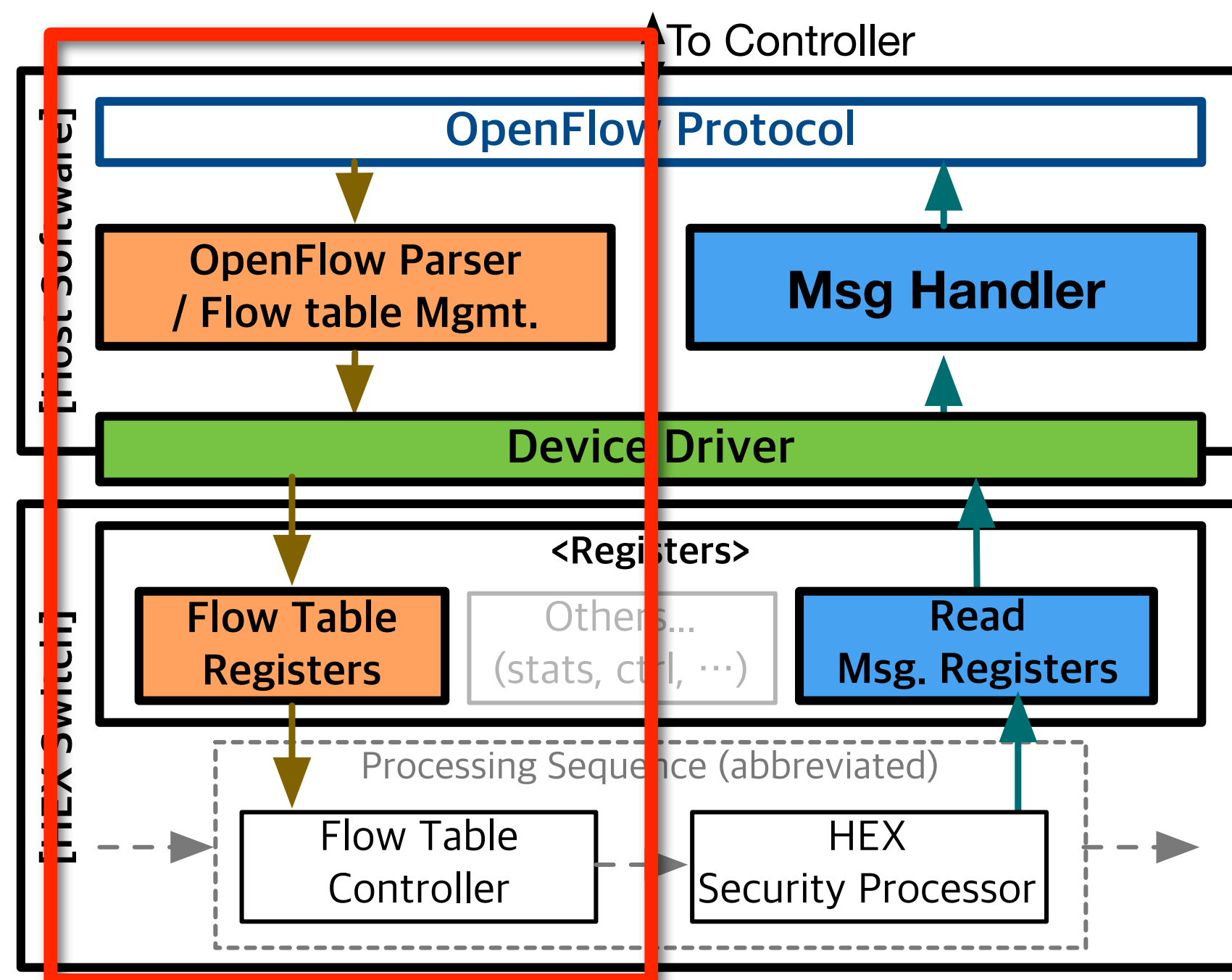
def install_DPI(self):
    msg = of.ofp_flow_mod()
    msg.match.in_port = 1
    msg.actions.append(HEX.dpi(rule="/patterns.txt", policy=HEX.POLICY_ALERT))
    msg.actions.append(of.ofp_action_output(port = 2))
    self.connection.send(msg)

def _handle_HEX_Alert (self, event):
    alert = event.ofp
    packet = event.parsed # This is the parsed packet data.

    print "[HEX ALERT]"
    print "in_port :", alert.in_port
    print "reason :", alert.reason
    print "cluster_id :", alert.cluster_id
    print "value :", alert.value
```

# Communication with a controller: Deploying security actions:

- The security actions are deployed by `flow_mod` messages
  - Security actions are compatible with common OpenFlow actions



```
import pox.openflow.hexswitch as HEX

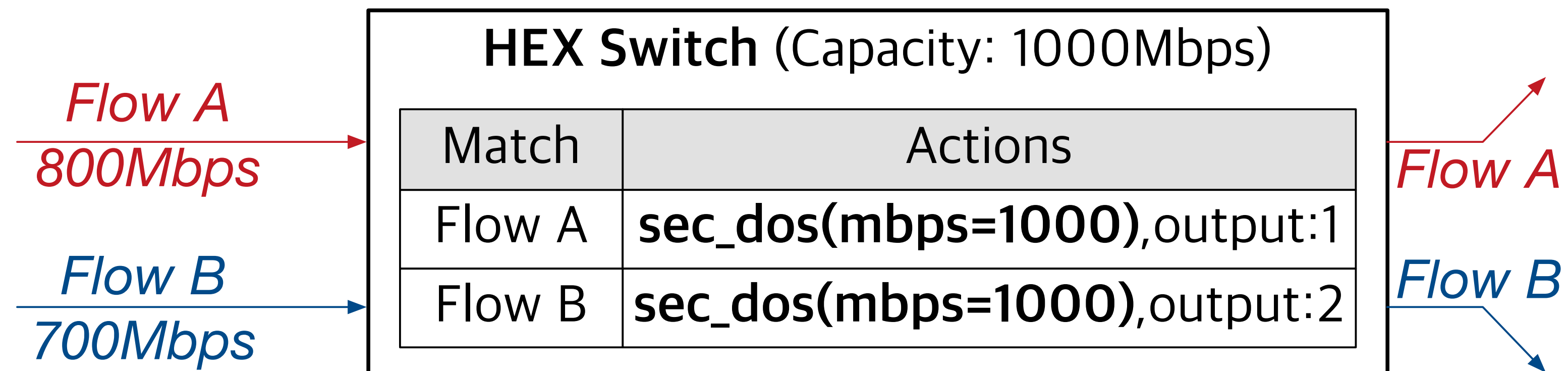
def install_DPI(self):
    msg = of.ofp_flow_mod()
    msg.match.in_port = 1
    msg.actions.append(HEX.dpi(rule="/patterns.txt", policy=HEX.POLICY_ALERT))
    msg.actions.append(of.ofp_action_output(port = 2))
    self.connection.send(msg)

def _handle_HEX_Alert (self, event):
    alert = event.ofp
    packet = event.parsed # This is the parsed packet data.

    print "[HEX ALERT]"
    print "in_port :", alert.in_port
    print "reason :", alert.reason
    print "cluster_id :", alert.cluster_id
    print "value :", alert.value
```

# Challenge in flow-level security deployment

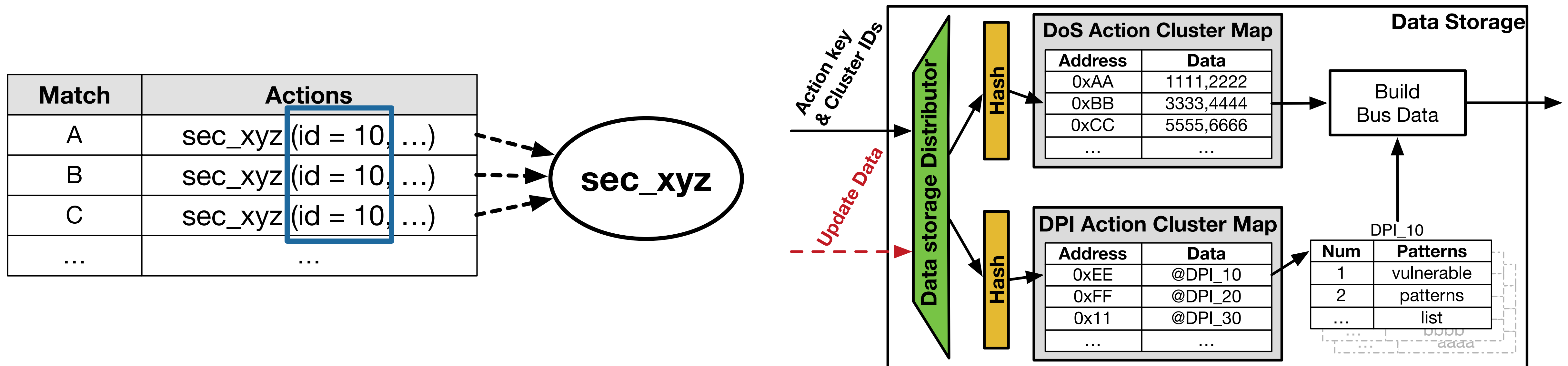
- The flow-level security cannot represent a security policy across multiple flows
- *Simple example:*



The total incoming bandwidth from Flow A/B evidently **exceeds 1000 Mbps**,  
but **the DoS detectors never trigger an alert!**

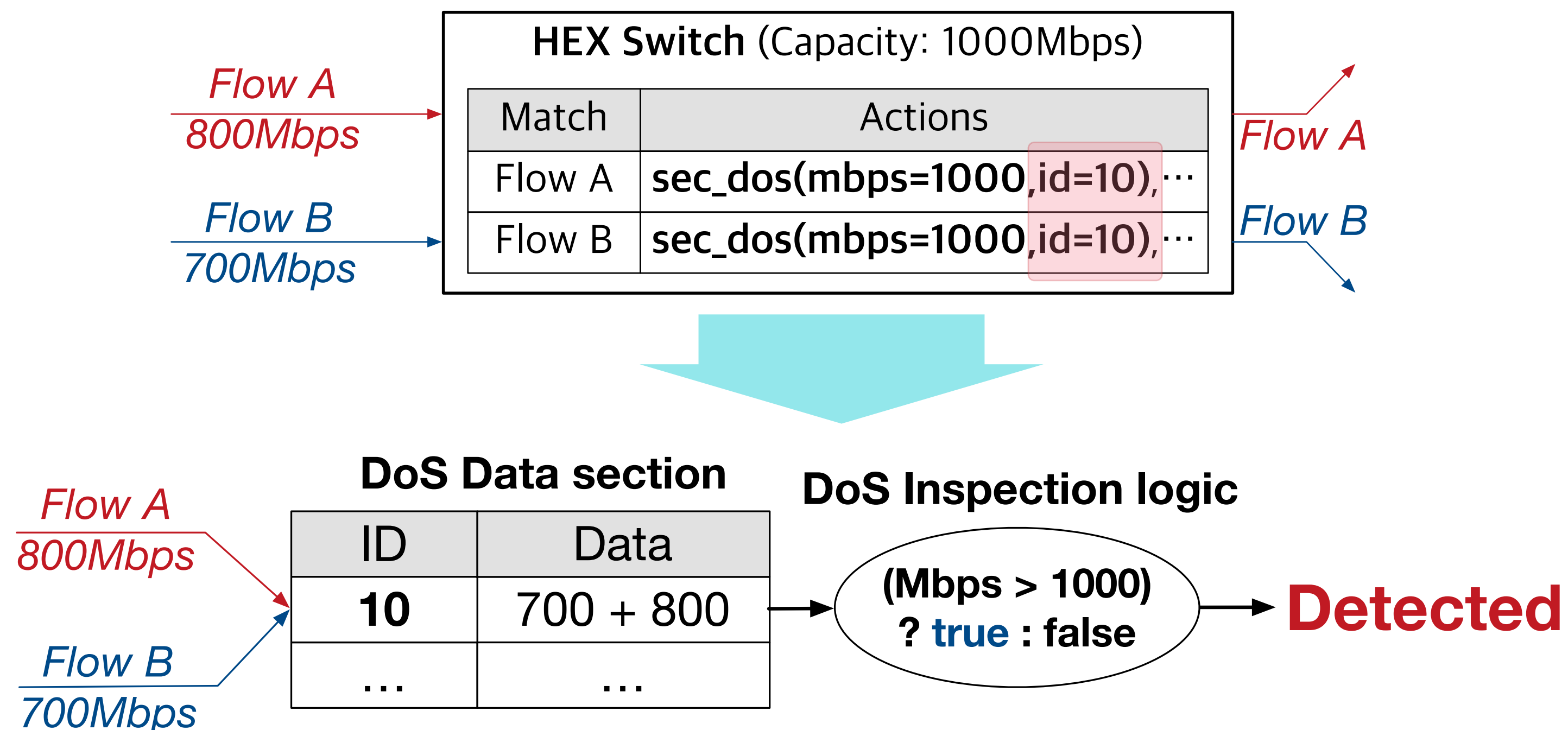
# Action Clustering

- Security actions have a cluster ID in their parameter
  - The actions that use the same cluster ID are considered to belong to the same cluster
  - The clustered action works as the integrated single action across different flow rules
- Implementing by sharing the data storage by the cluster map



# Applying Action Clustering

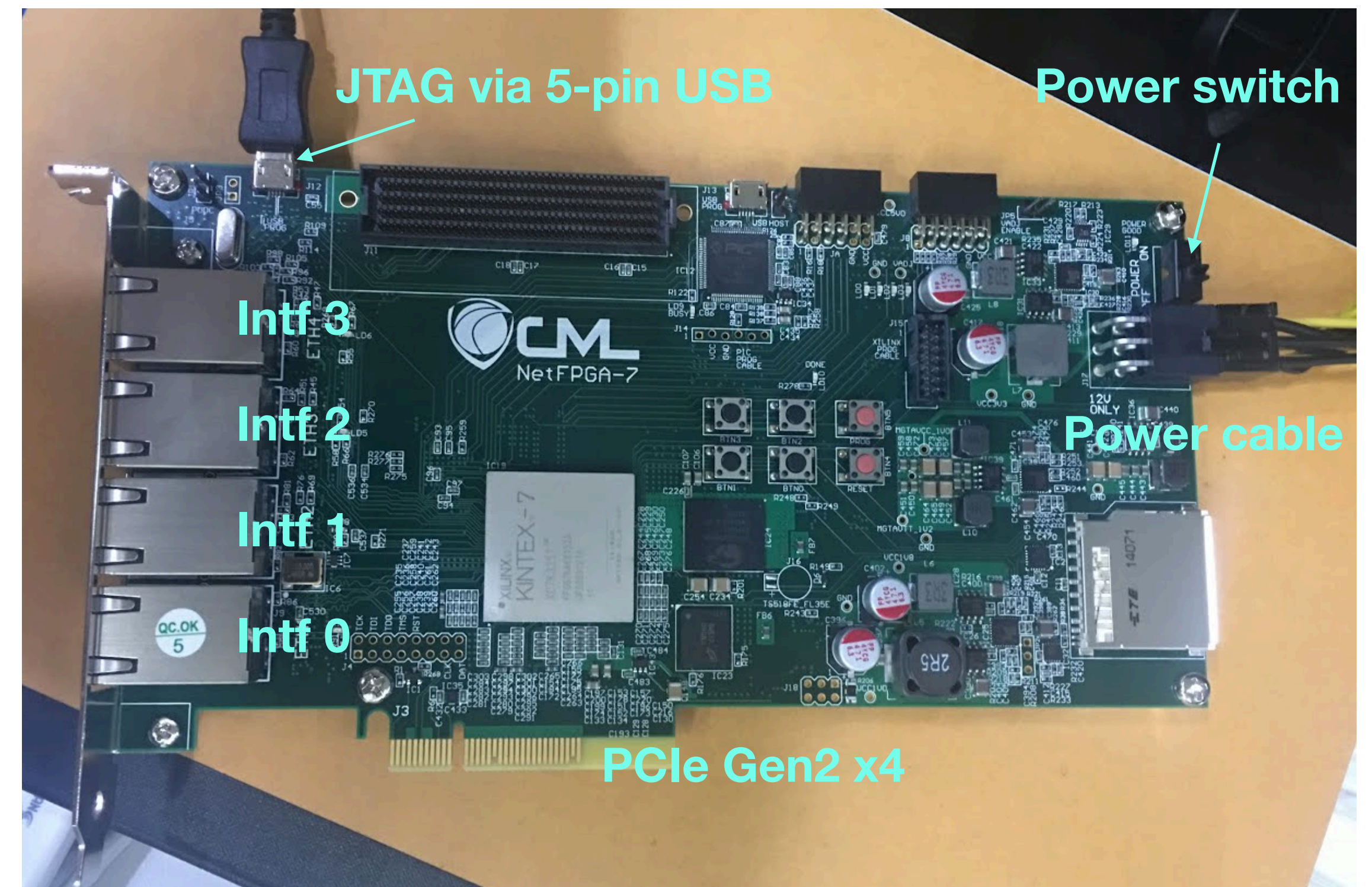
- Applying the action clustering to the previous example



DoS detector can successfully detect the bandwidth excess and alert this.

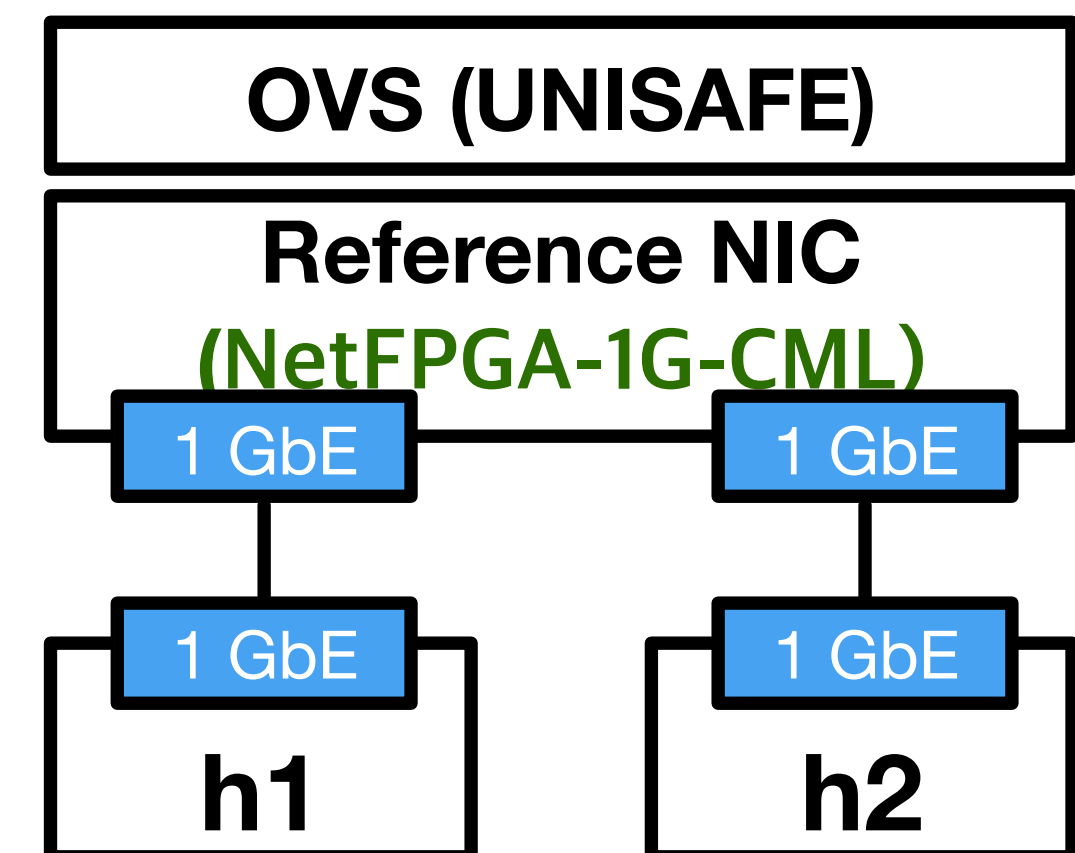
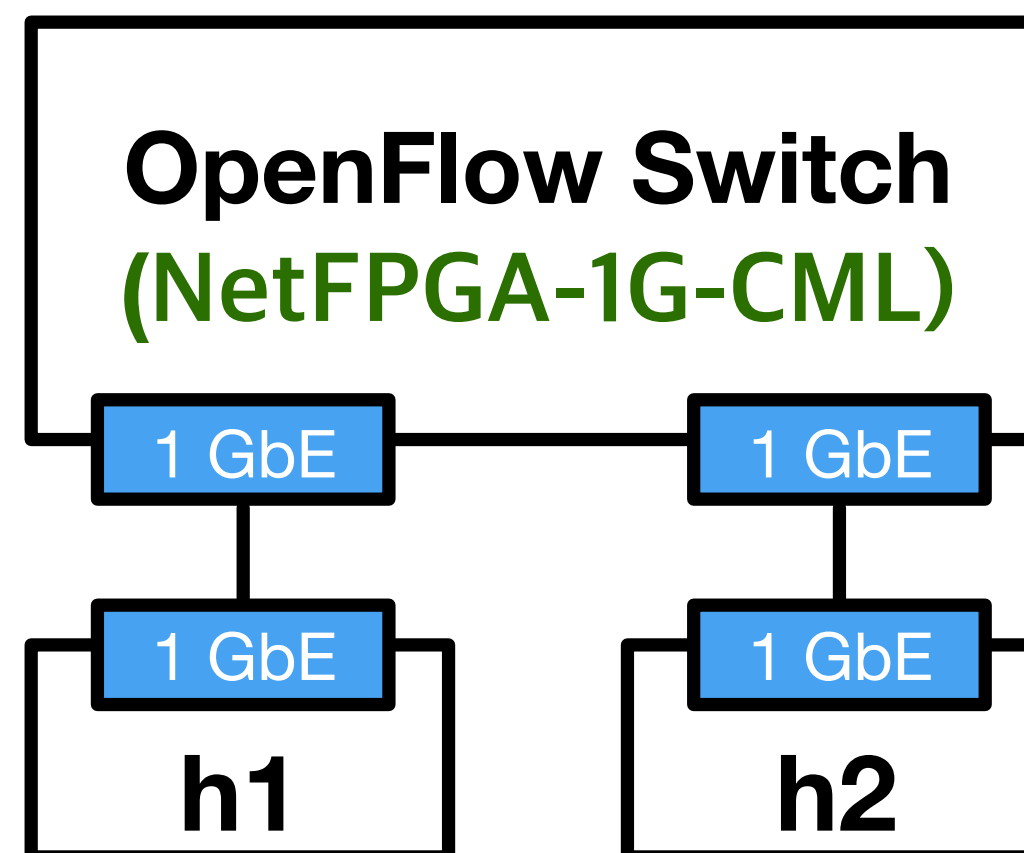
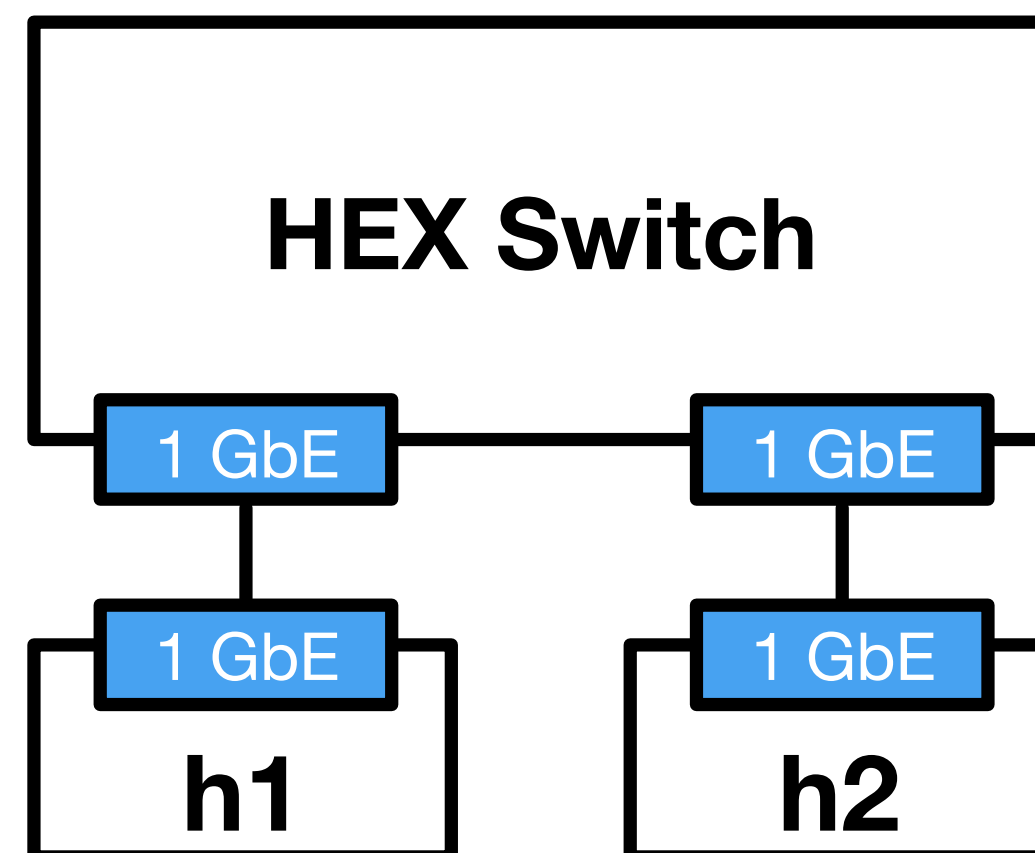
# Implementation

- NetFPGA-1G-CML
  - Based on Reference NIC and OpenFlow switch from the NetFPGA project (<https://github.com/NetFPGA>)
- Support DoS Detector and Deep Packet Inspector (*Payload inspector*)



# Evaluation

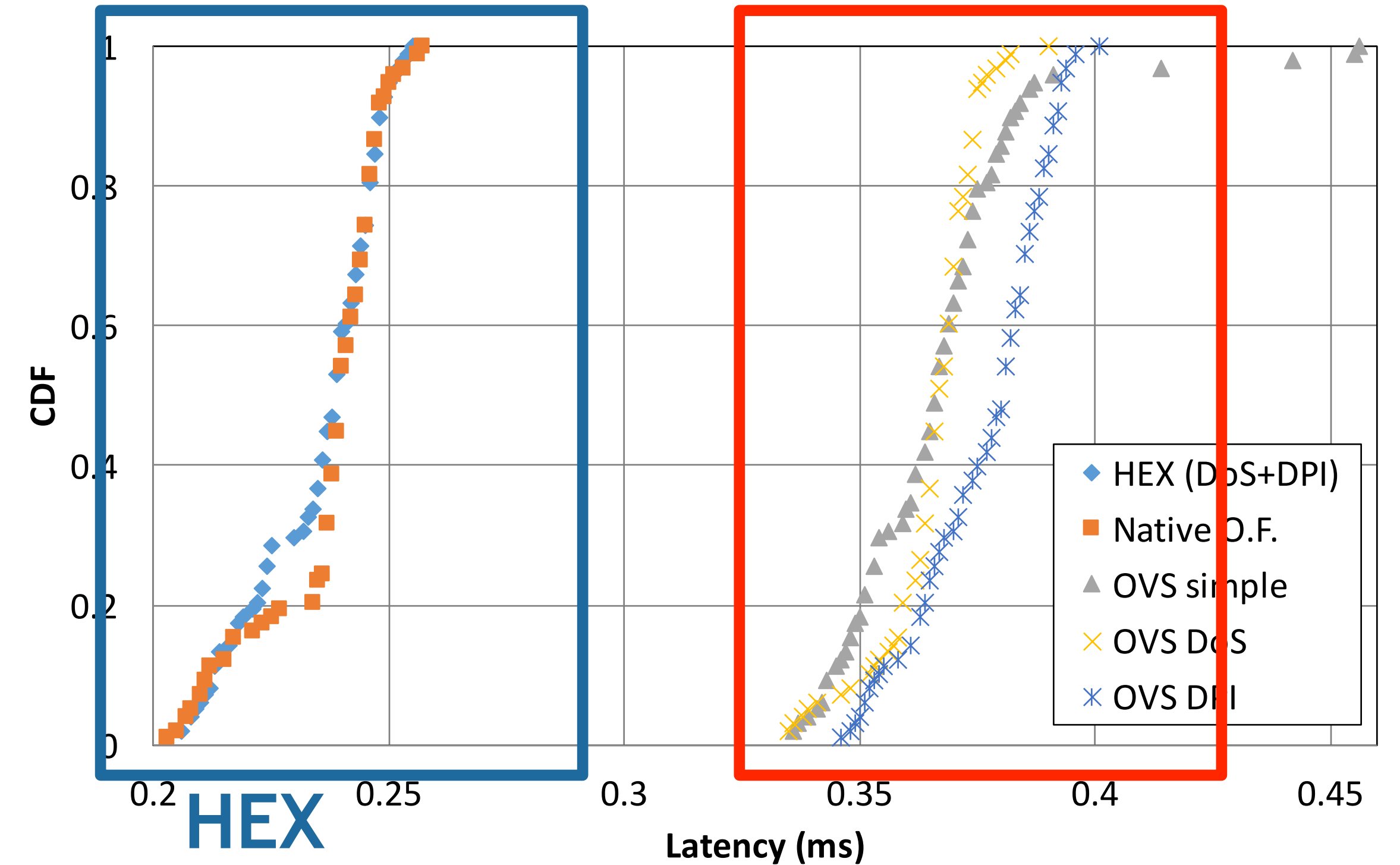
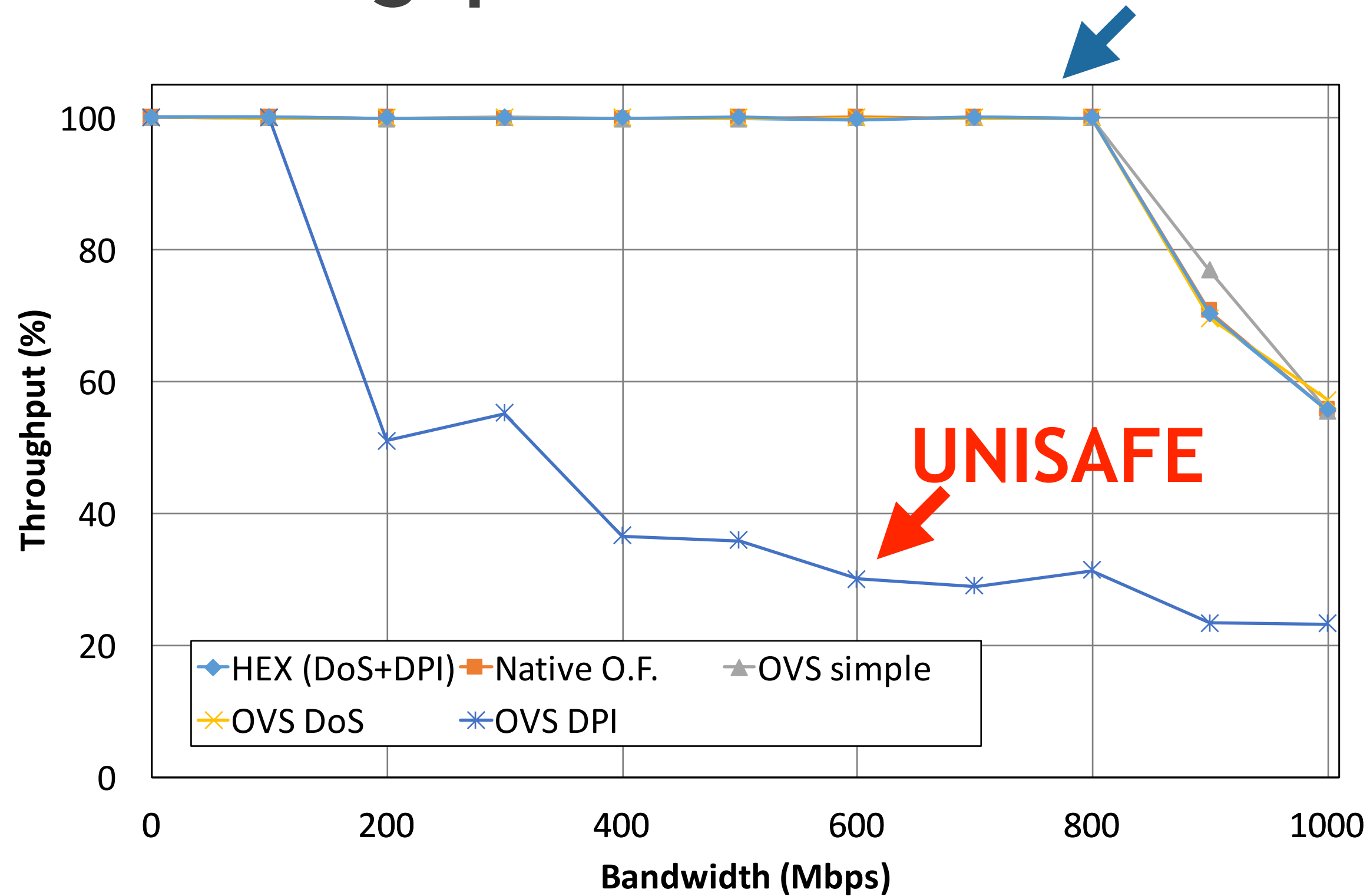
- Measure throughput and latency
  - 1) Performance of the HEX switch
  - 2) Performance of simple forwarding by the normal OpenFlow switch
  - 3) Performance of OVS based implementation (i.e., UNISAFE)



# Evaluation Result

## • Throughput

## HEX & Simple Fwd. • Latency



# Conclusion

---

- The HEX switch that embeds security functions
  - Using NetFPGA
  - As as a set of actions
  - Support security policy and controller APIs
- Achieves line-rate performance without overhead.

Thank you!  
Questions?