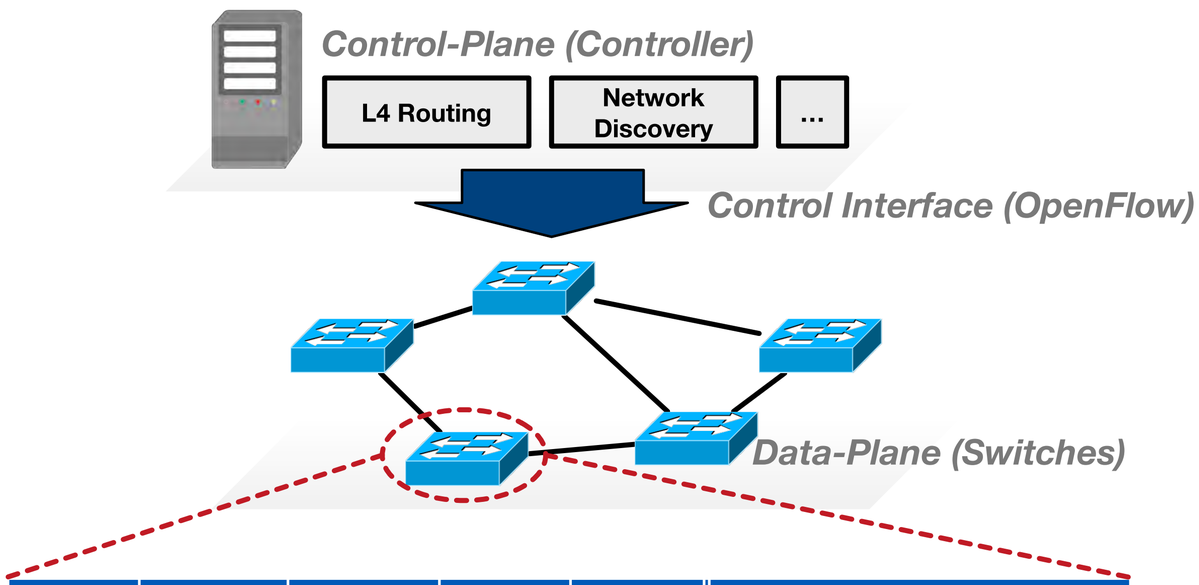
NS² Network and System Security Laboratory KAIST

# DPX:
## Data-Plane eXtensions for SDN Security Service Instantiation

**Taejune Park**[1], Yeonkeun Kim[1],
Vinod Yegneswaran[2], Phillip Porras[2],
Zhaoyan Xu[3],
KyoungSoo Park[1], and Seungwon Shin[1]

*1) KAIST, Korea  2) SRI International, USA  3) Palo Alto Networks, USA*

# Software-Defined Networking




| in_port | ip_src | ip_dst | tcp_src | tcp_dst | actions |
|---|---|---|---|---|---|
| 1 | 10.0.0.1 | 10.0.0.2 | * | * | output(2) |
| * | * | * | * | 80 | drop |
| 2 | * | 20.0.0.1 | 22 | * | set_ip_dst(20.0.0.2),output(1) |

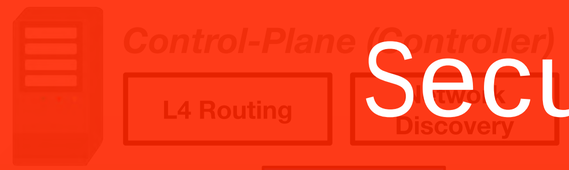
- Decouple control-plane from data-plane
  - Centralized controller
  - *SDN Switches*
- Centralized operation with standard protocol *(e.g., OpenFlow)*

- *Programable network management*
- *Dynamic traffic engineering*

# Software-Defined Networking

## Security is still required

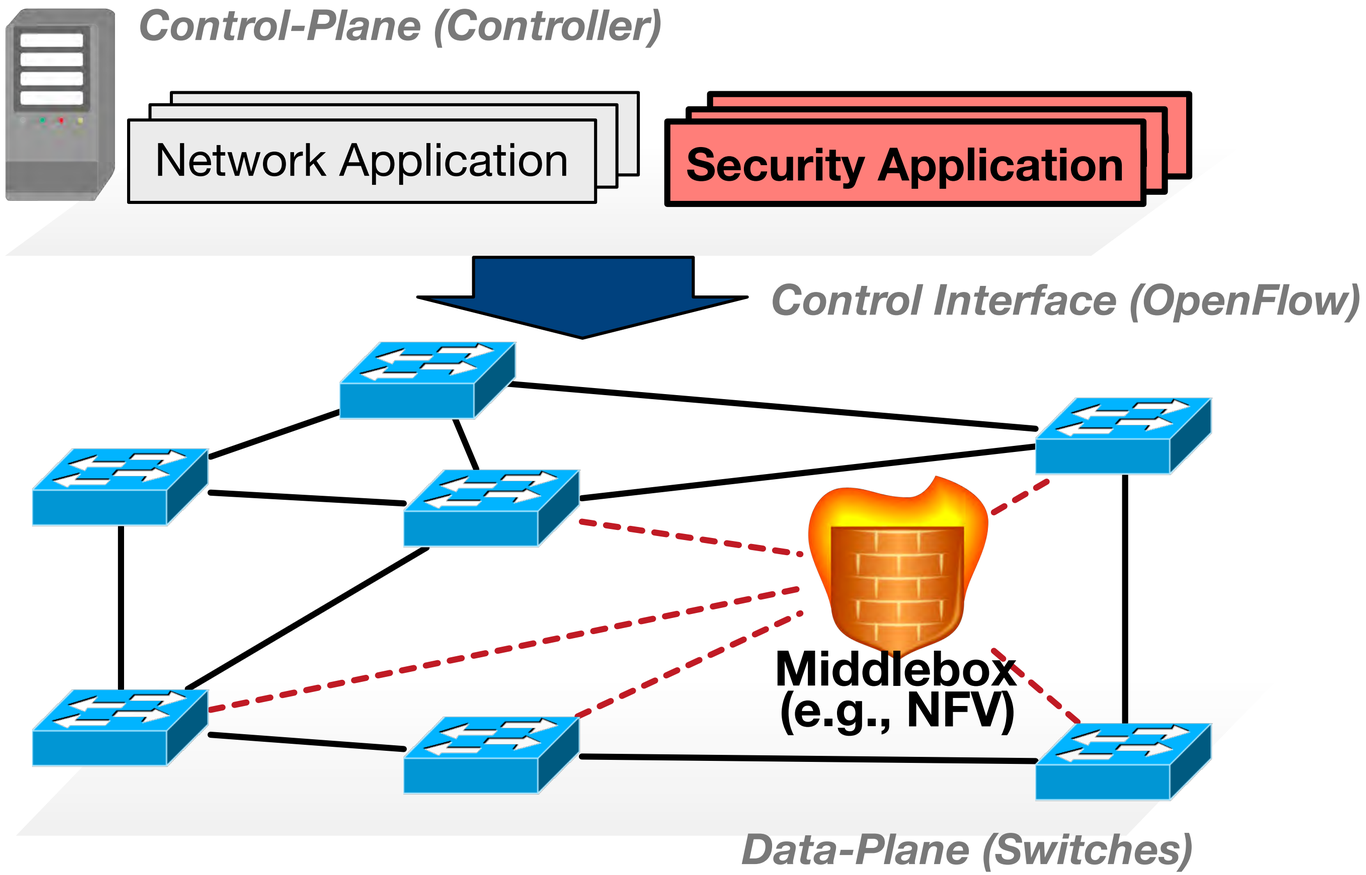
- Shin, Seung Won, et al. "Fresco: Modular composable security services for software-defined networks."
- Shin, Seung Won, et al. ”Cloudwatcher: Network security monitoring using openflow in dynamic cloud networks."
- Braga, Rodrigo, et al. "Lightweight DDoS flooding attack detection using NOX/OpenFlow."
- Yoon, Changhoon, et al. "Enabling security functions with SDN: A feasibility study."
- S. K. Fayazbakhsh, et al. “Enforcing network-wide policies in the presence of dynamic middlebox actions using flowtags”
- Z. A. Qazi, et al. “SIMPLE-fying Middlebox Policy Enforcement Using SDN.”
- And so on...



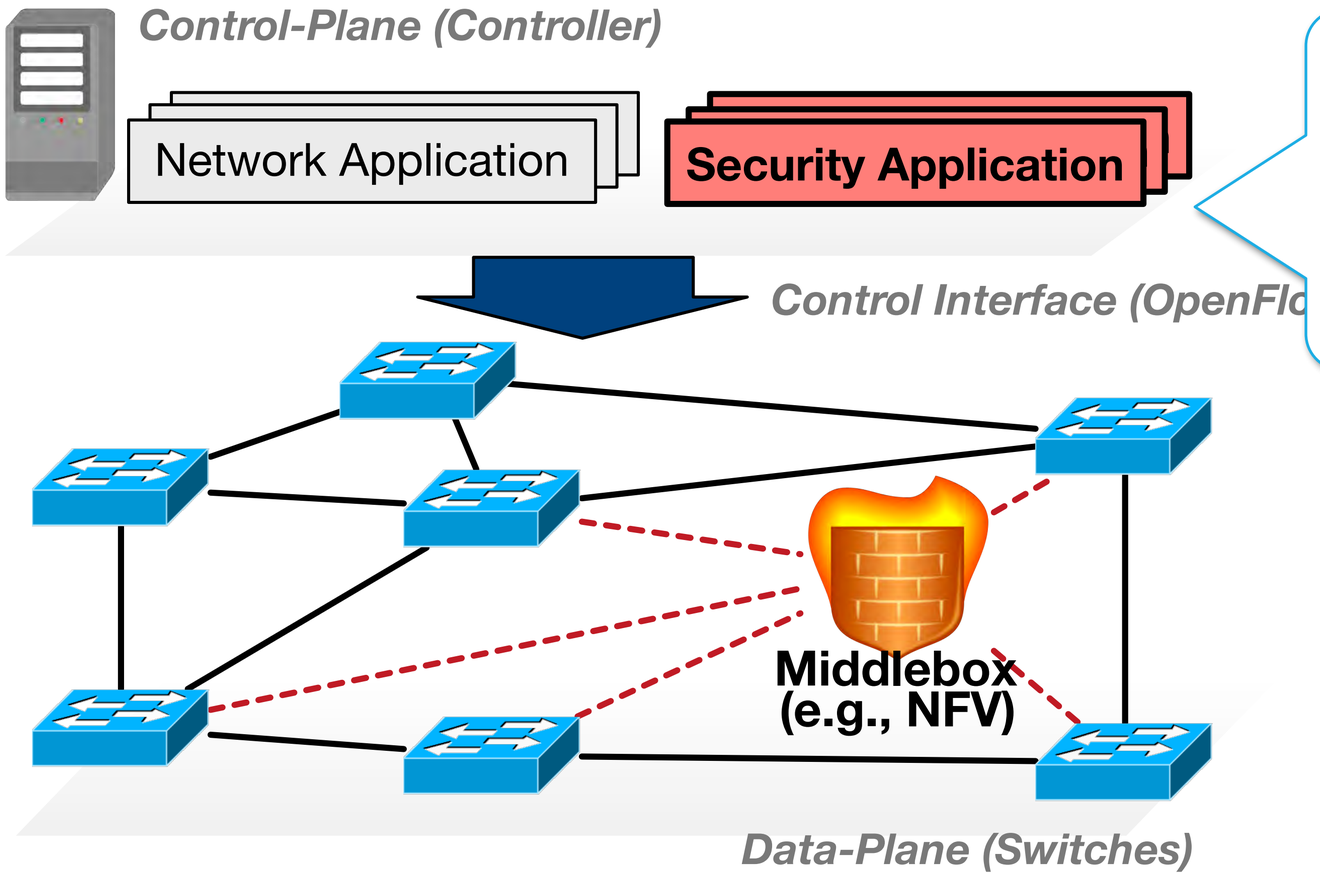

| in_port | ip_src | ip_dst | tcp_src | tcp_dst | actions |
|---|---|---|---|---|---|
| 1 | 10.0.0.1 | 10.0.0.2 | * | * | output(2) |
| * | * | * | * | 80 | drop |
| 2 | * | 20.0.0.1 | 22 | * | set_ip_dst(20.0.0.2),output(1) |

- Decouple control-plane from data-plane
- Centralized controller
- SDN Switches
- Interface implementation with standard protocol (e.g., OpenFlow)
- Programable network management
- *Dynamic traffic engineering*

# Security in Software-Defined Networking

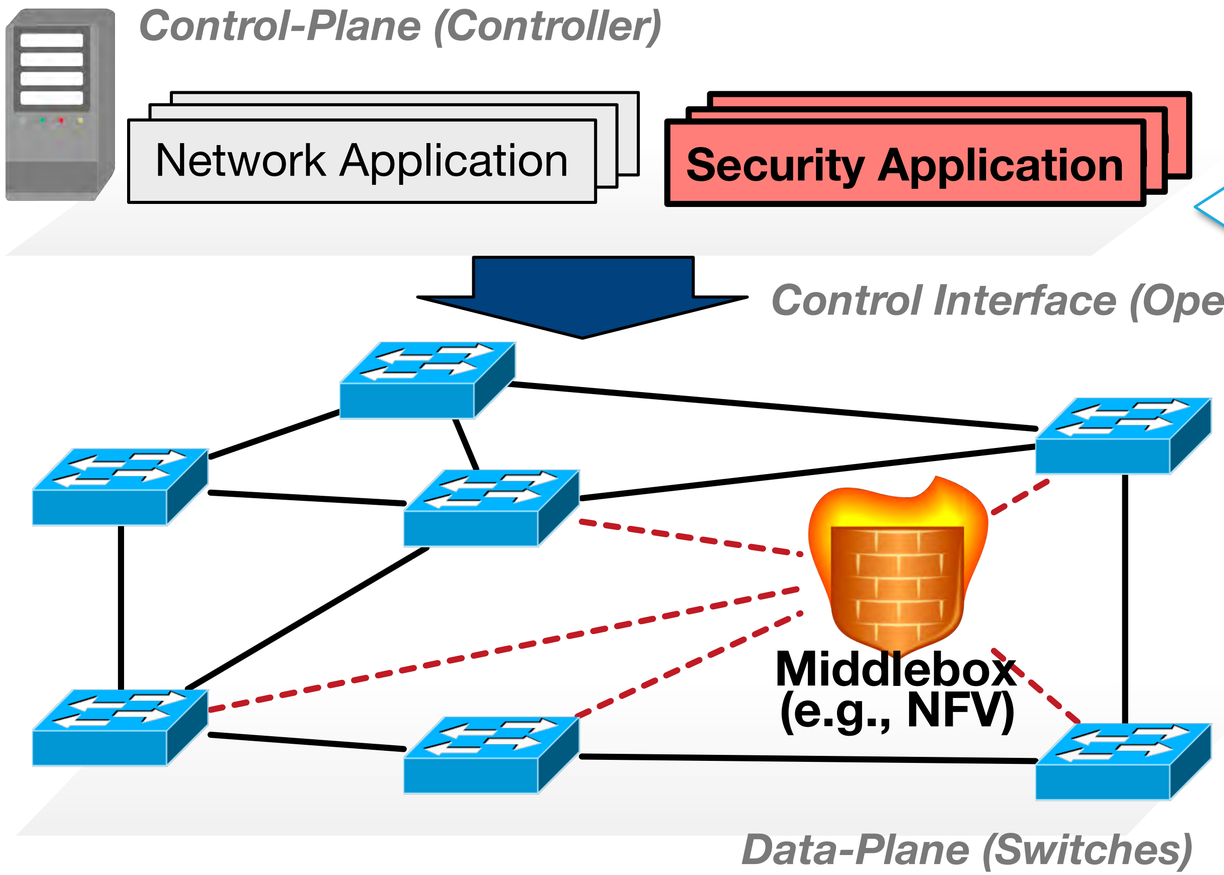


# Security in Software-Defined Networking




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

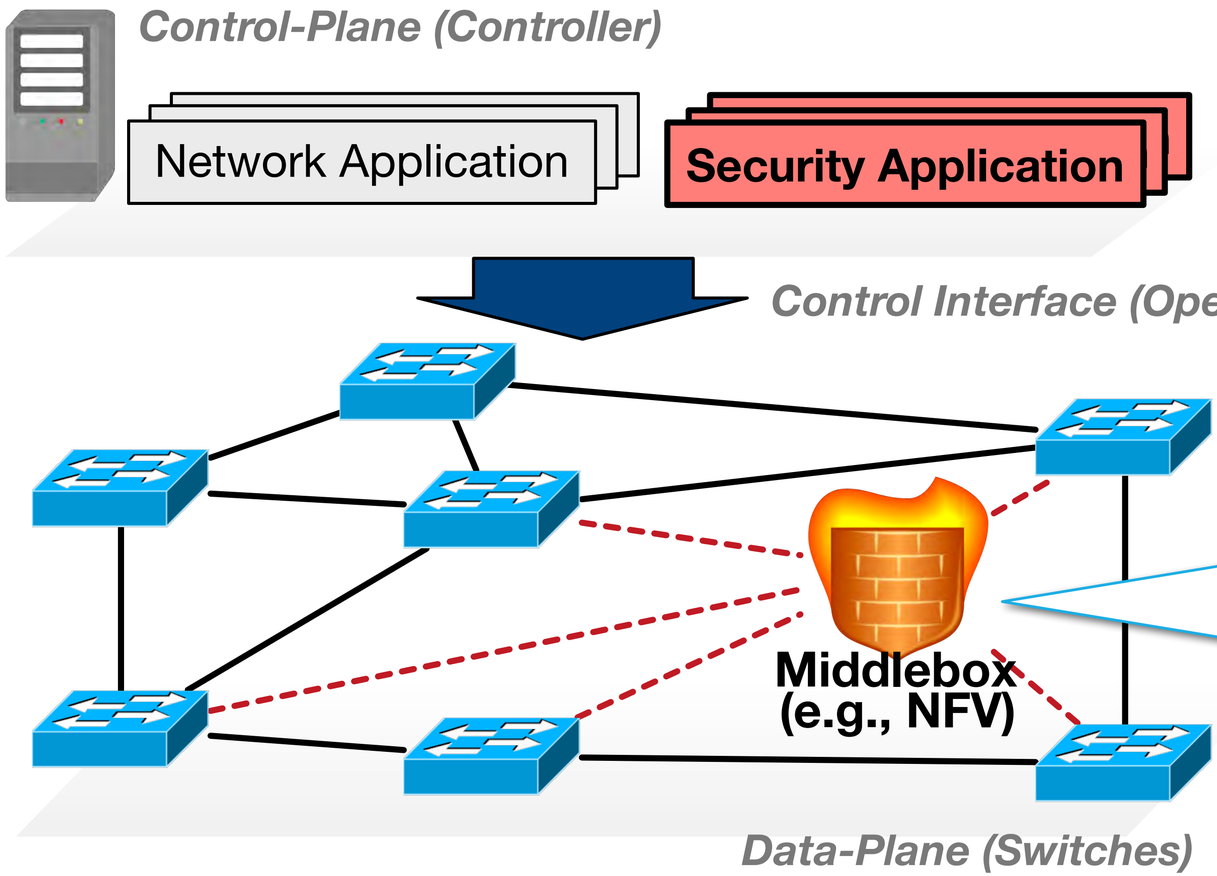
# Security in Software-Defined Networking




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

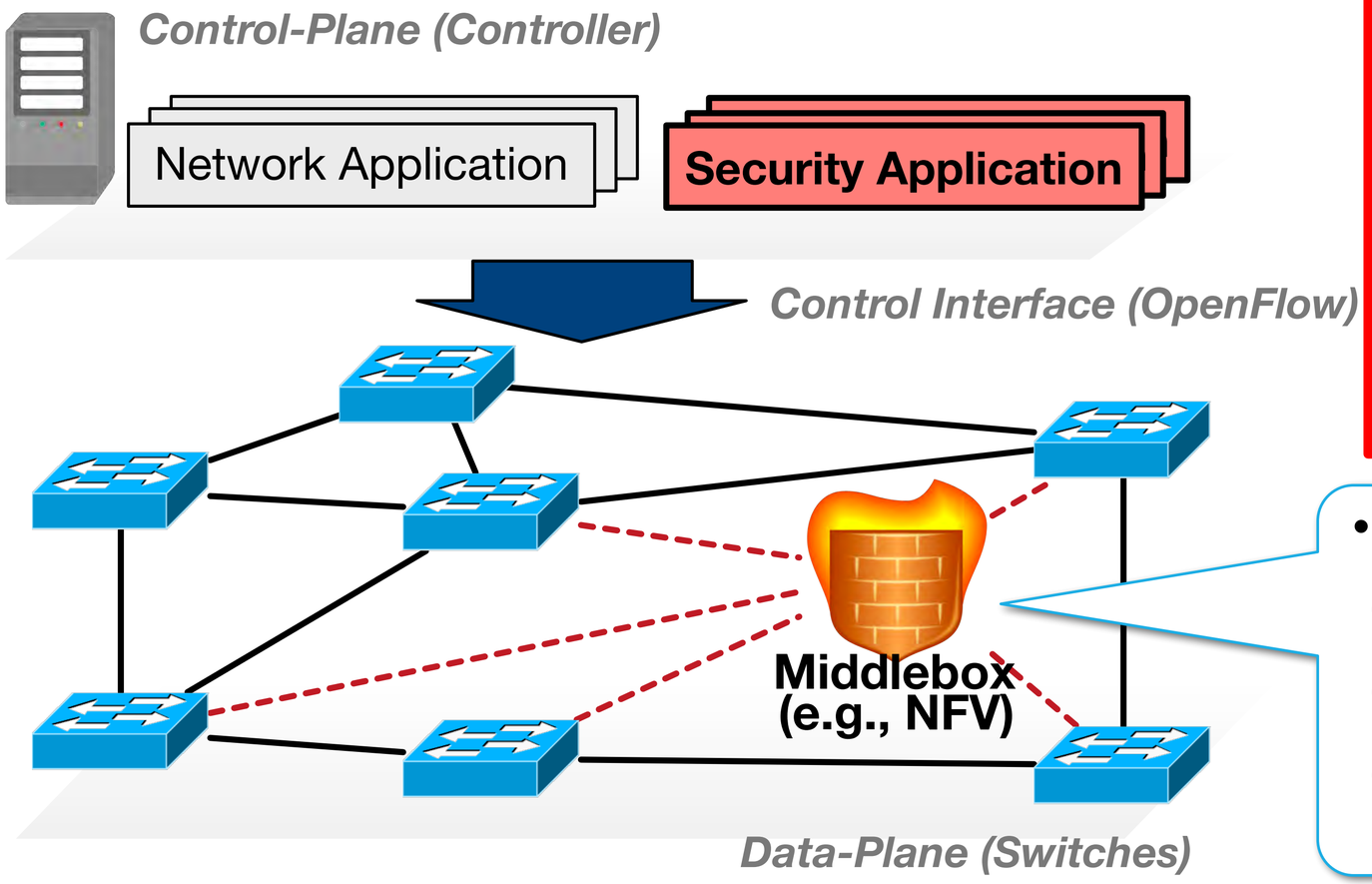
- Limitation
  - Simple security only available
  - Controller overhead
  - Low performance

# Security in Software-Defined Networking




- Middle-boxes (e.g., NFV)
  - Better performance
  - Rich functions (e.g., payload inspection)
  - No controller overhead

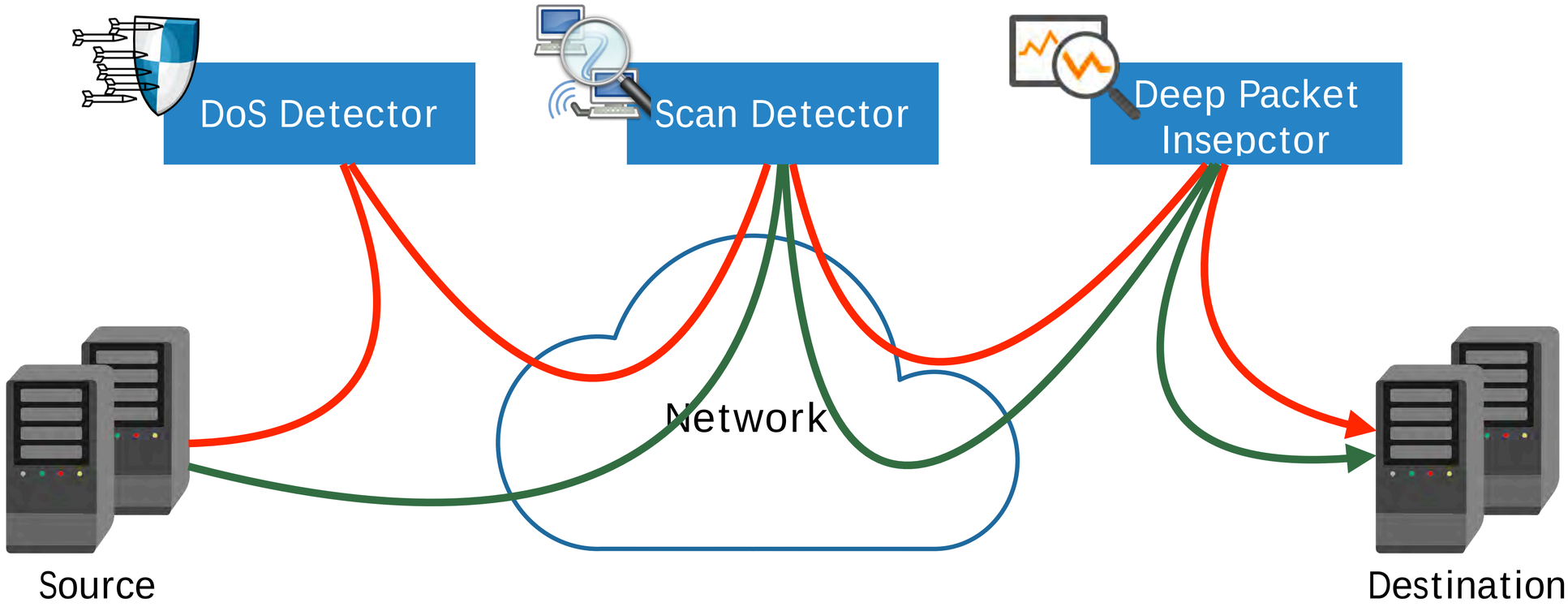
# Security in Software-Defined Networking



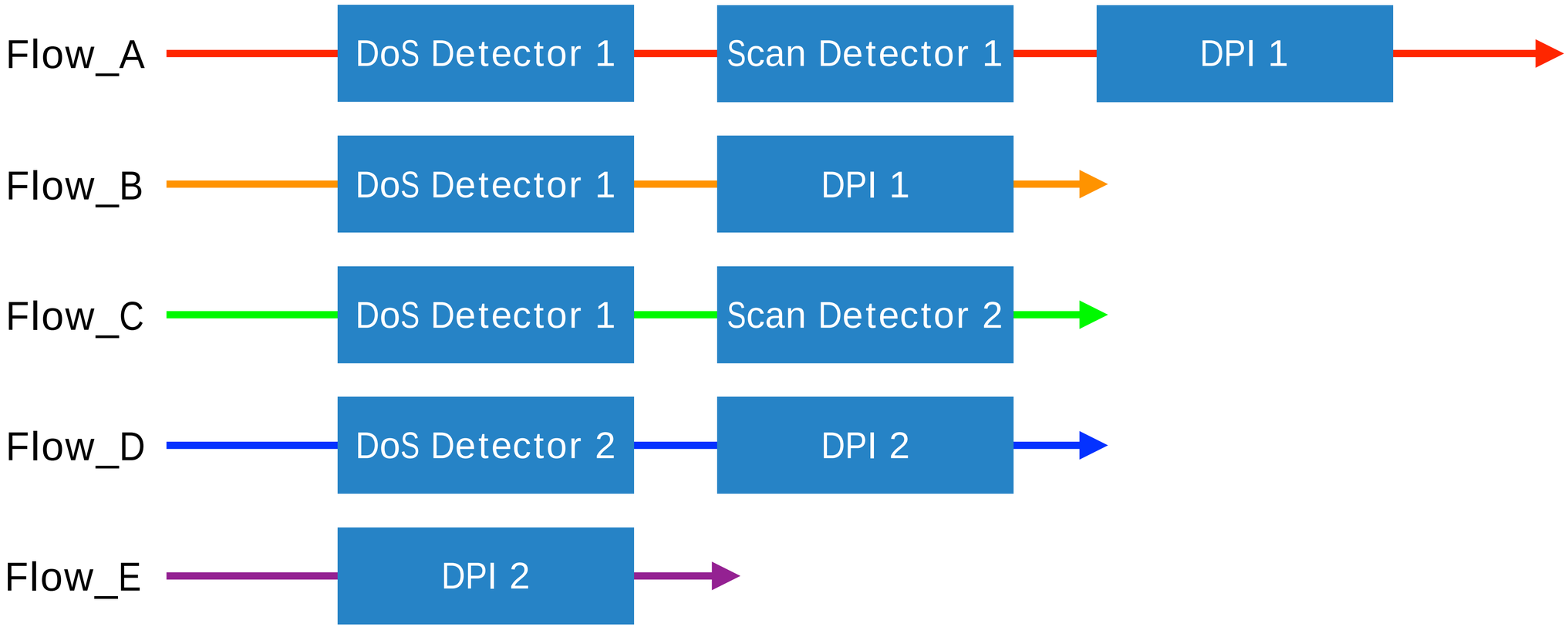

- Limitation
  - Network overhead caused by traffic detouring (Performance loss)
  - Require flow steering for NFs
  - Additional control channels for NFs

- Middle-boxes (e.g., NFV)
  - Better performance
  - Rich functions (e.g., payload inspection)
  - No controller overhead

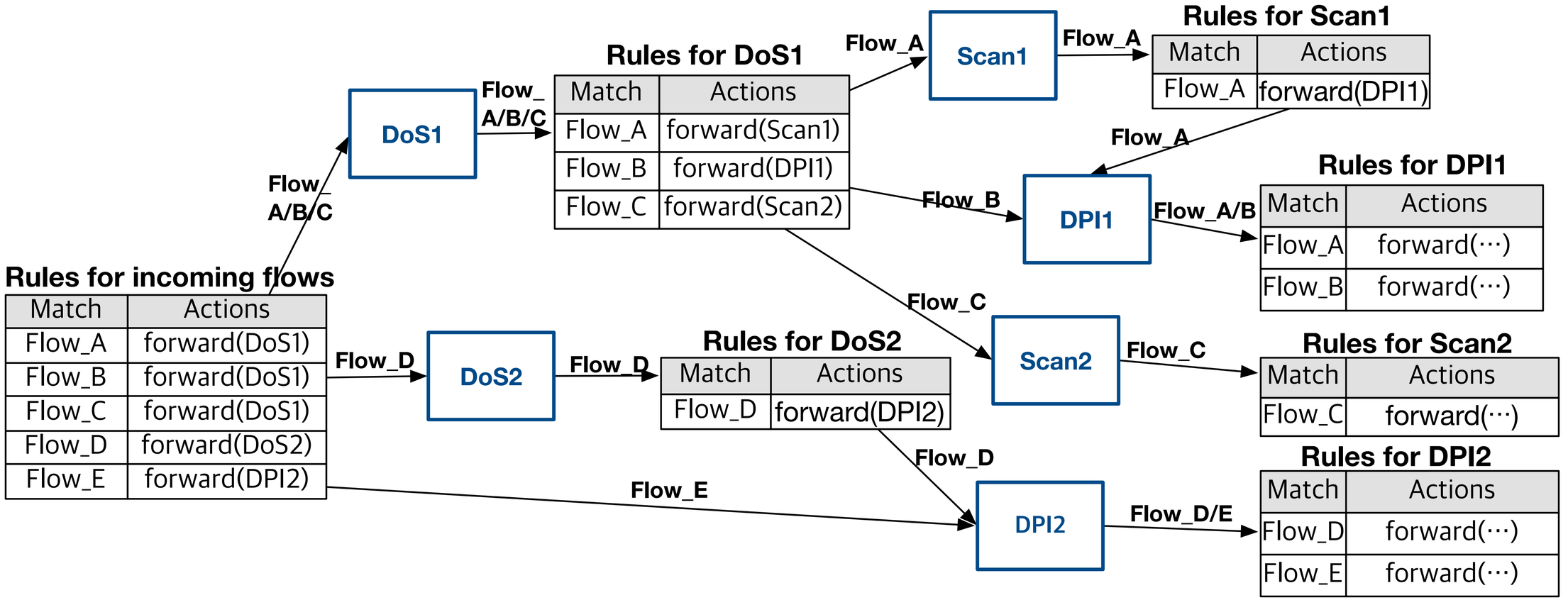
# Service Chaining



# Service Chaining



# Service Chaining



# Challenges of Security in SDN

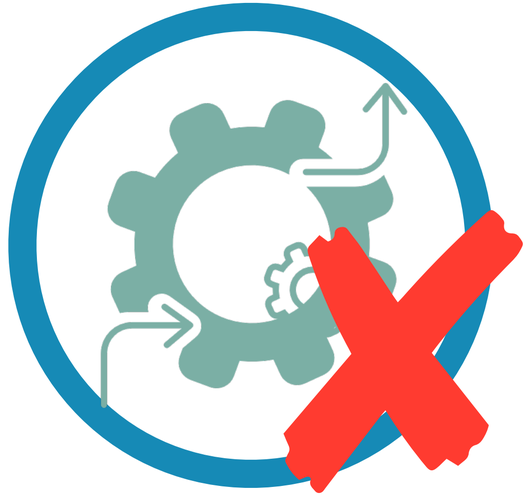
Performance

Management

# Challenges of Security in SDN

Performance

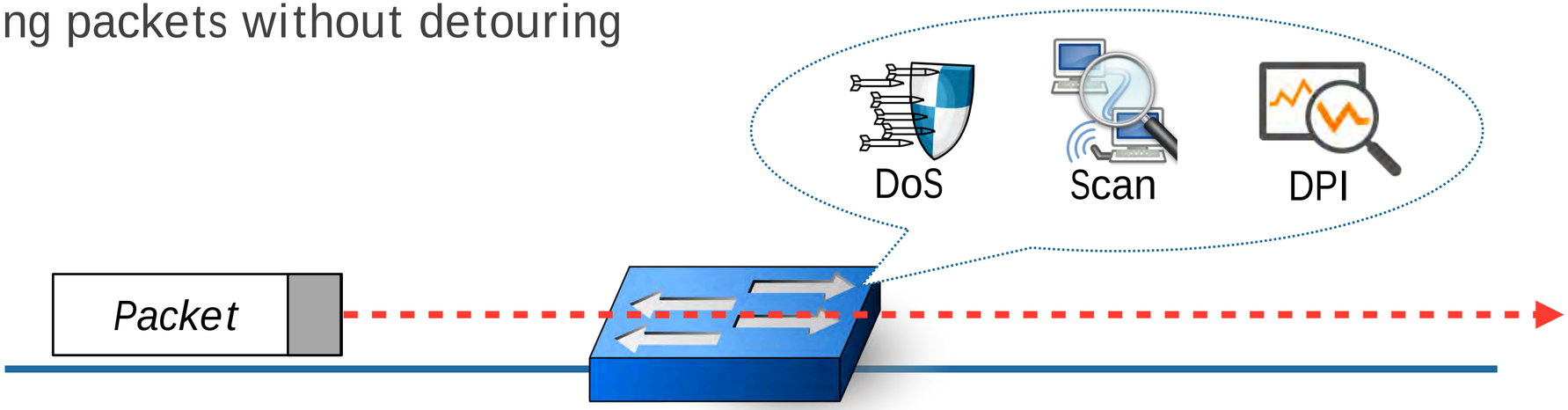
Management



Flow steering/engineering

# **DPX:** Data-Plane eXtensions for SDN Security Service Instantiation

- Provides security services as a part of packet processing logic.
  - Security services as *a set of OpenFlow actions*
  - Processing packets without detouring

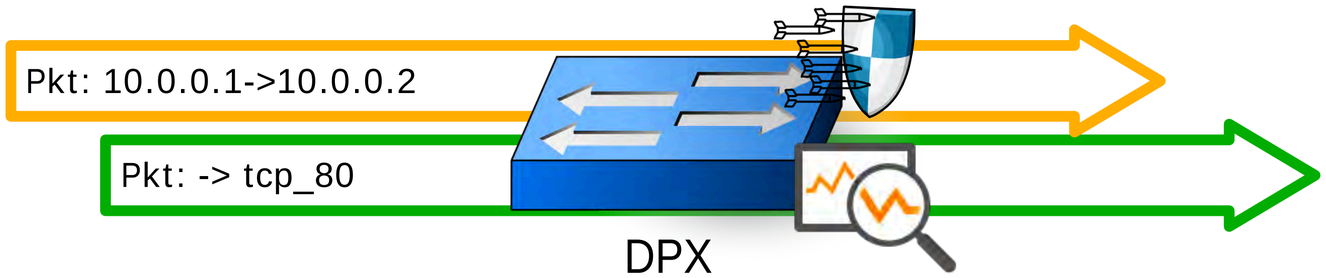



| MATCH | Actions |
|---|---|
| *Flow_A* | **sec_dos(mbps=1000)**, output:2 |
| *Flow_B* | **sec_dos(…), sec_scan(…)**,output:3 |

# Security actions

- Providing security services for an incoming flow

| ip_src | ip_dst | tcp_src | tcp_dst | Actions |
|---|---|---|---|---|
| 10.0.0.1 | 10.0.0.2 | * | * | **sec_dos(mbps=1000, policy=alert)**, output:2 |
| * | * | * | 80 | **sec_dpi(pattern="rule.txt", policy=discard)**,output:3 |




- To deploy, set a *threshold* and *policy* to the parameters of a required security action

*Threshold* *Policy*

**Security Action: sec_dos(mbps=1000, policy=alert)**

# Security actions

- High-compatibility with common OpenFlow actions

| MATCH | Actions |
|---|---|
| *Flow_A* | **sec_dos(mbps=1000)**, **set_ip_dst(10.0.0.2)**, output:2 |

- Fine-grained security deployment per a flow

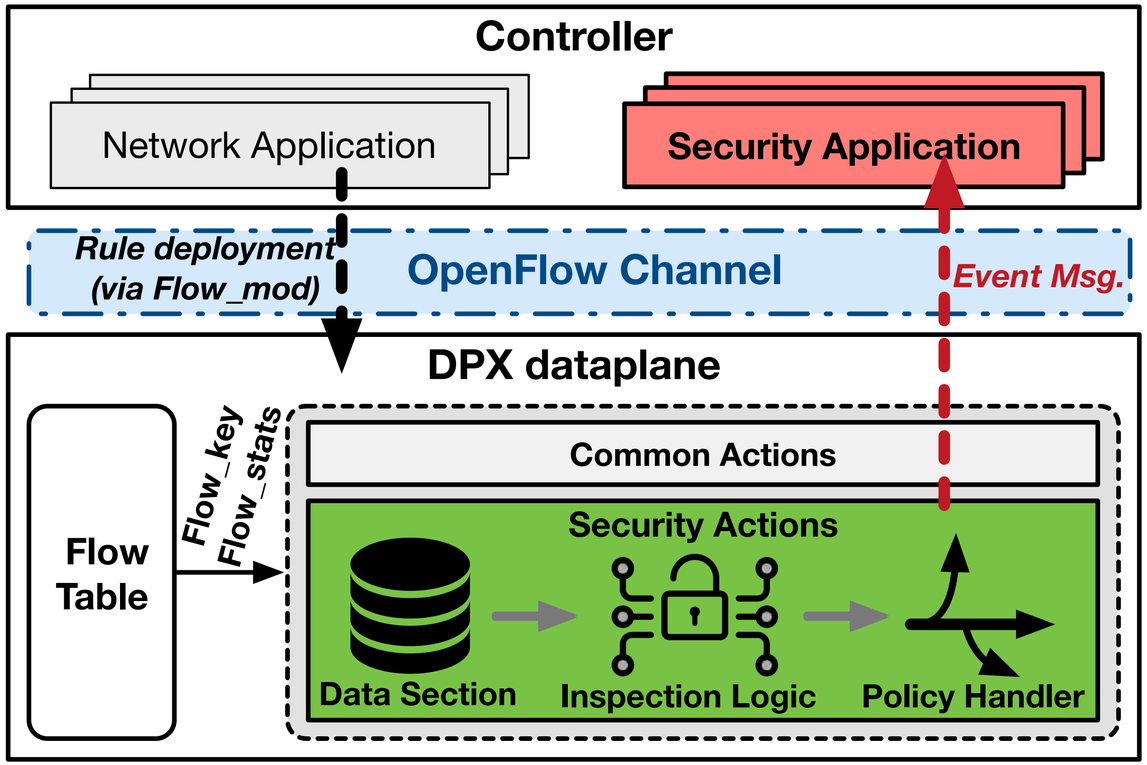
| MATCH | Actions |
|---|---|
| ***Flow_A*** | **sec_dos(mbps=1000)**, output:2 |
| ***Flow_B*** | **sec_dos(mbps=500)**,output:2 |
| ***Flow_C*** | **sec_dos(mbps=750)**,output:2 |

- Easy configuration for a security service chaining

| MATCH | Actions |
|---|---|
| *Flow_A* | **sec_dos(…), sec_scan(…), sec_dpi(…)**, output:2 |

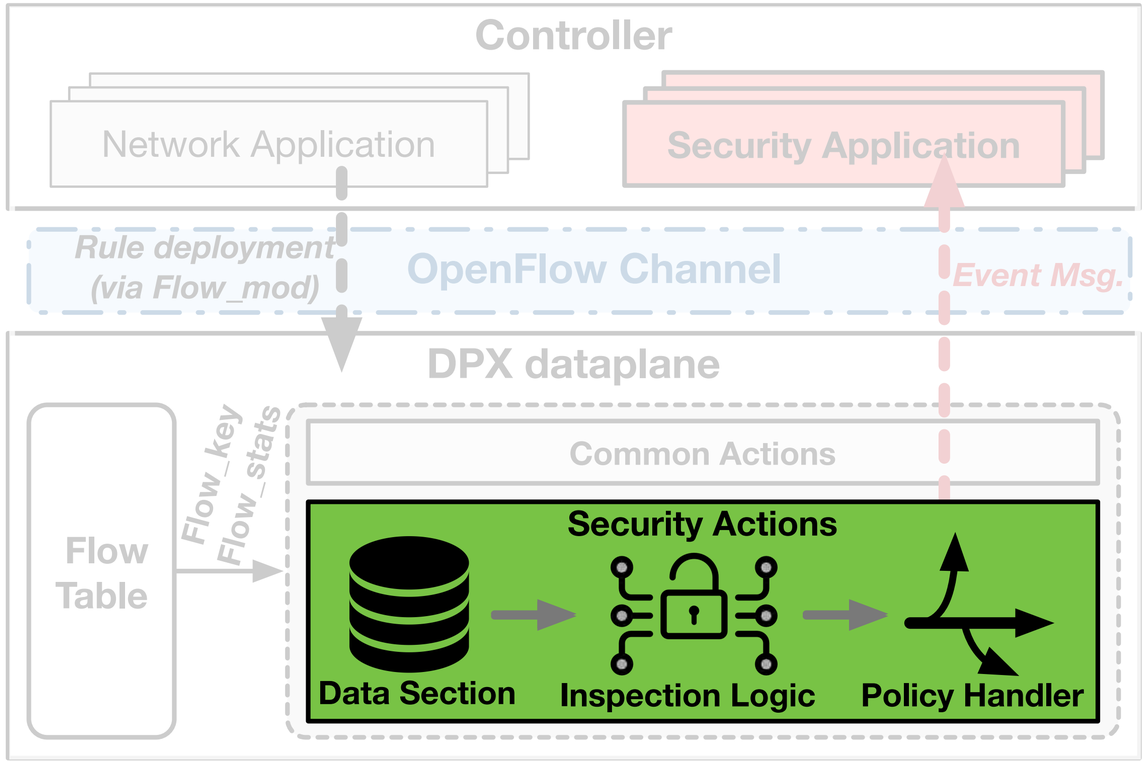
# System Design

- Similar to a conventional SDN
  - *Match a flow rule in a flow table -> Perform actions*
- Security action block
- DPX security application



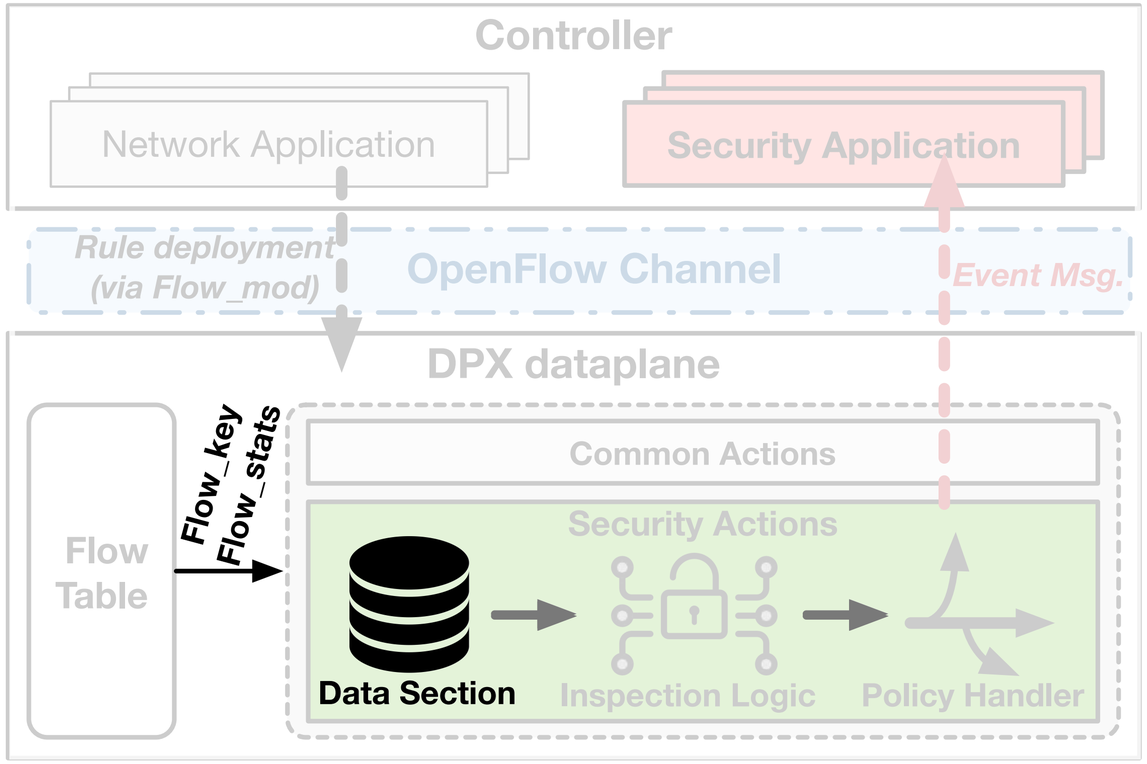
# Security Action Block

- Individual processing block for a security action
  - Data Section
  - Inspection Logic
  - Policy Handler



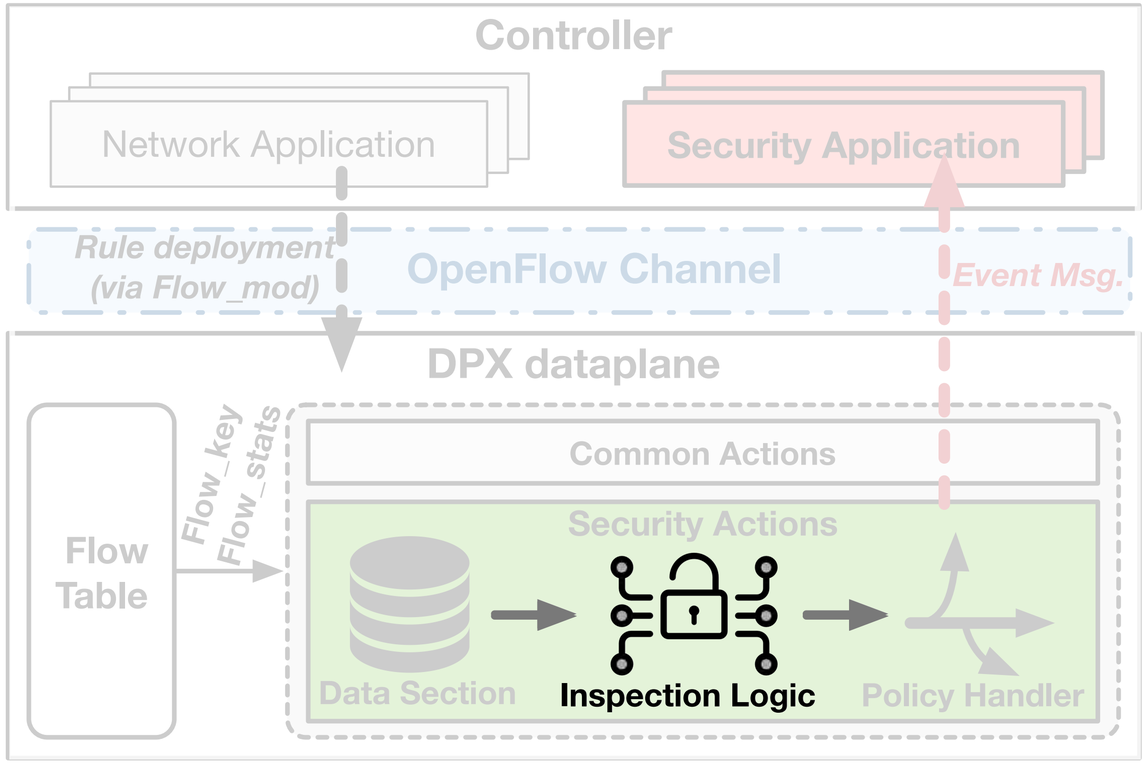
# Security Action Block: Data Section

- Store required statistics data of a packet by
  - *Flow_key*: Packet-level metadata used for indexing a flow table
  - *Flow_stats*: Flow table statistics



# Security Action Block: Inspection Logic

- Perform actual inspection
  - Calculate statistics using the data section
- Determine a security violation with threshold values in the parameter
  - *sec_dos(mbps=1000,…)*

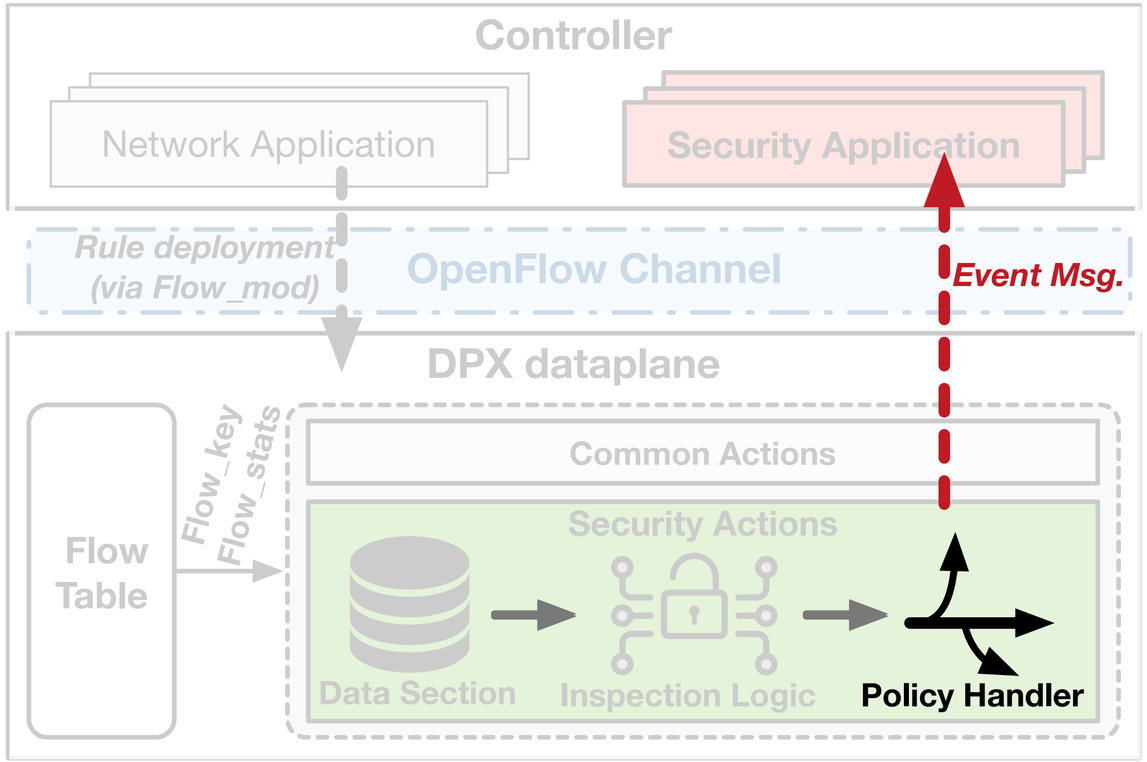


# Security Action Block: Policy Handler

- Handle a violation according to a policy
  - *sec_dos(…,policy=redirect:2)*

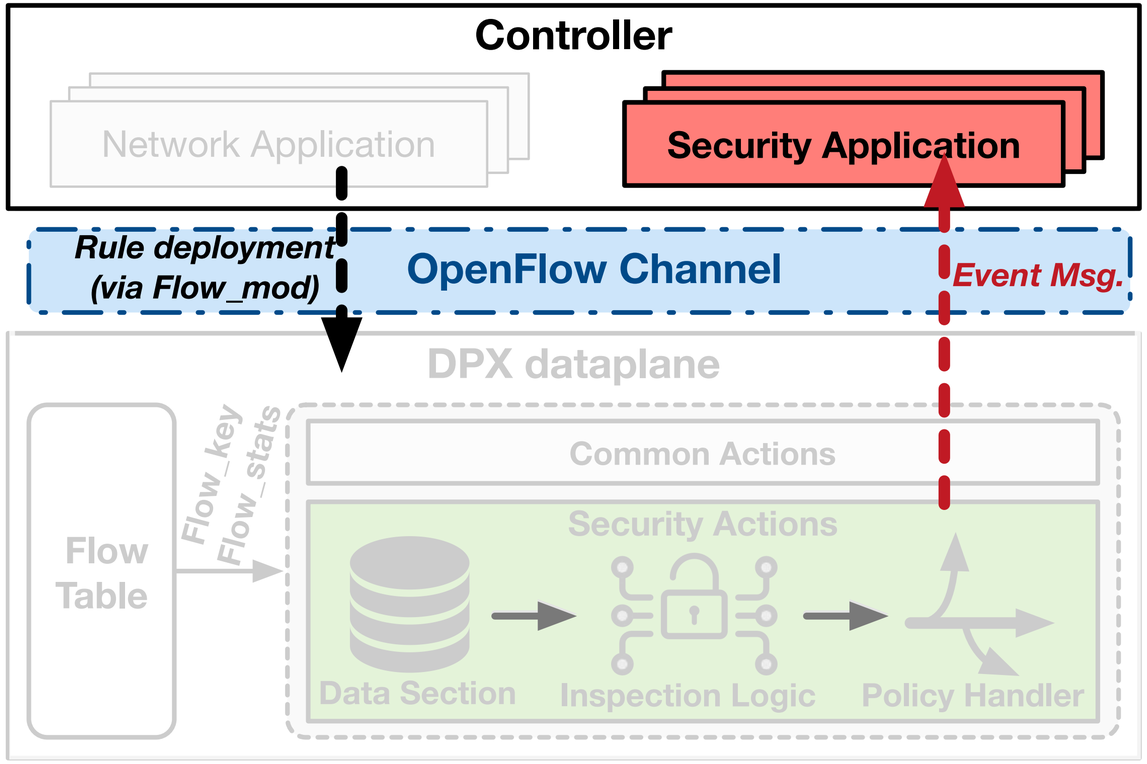
    => If the current bps exceeds a threshold, redirect the flow to Port 2.

- Three polices
  - *Alert: Send an alert msg to a controller*
  - *Discard: Terminates the packet processing and drop the packet*
  - *Redirect: Forward packets to an alternative port*



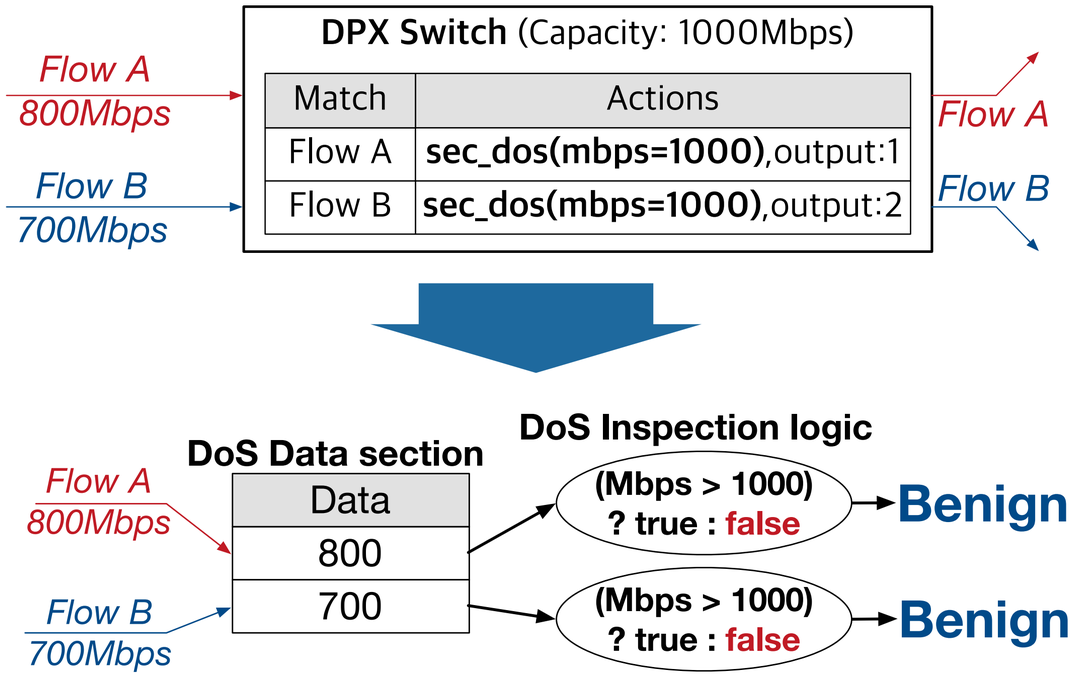
# Action Enforcement

- DPX provides a controller API set for the security actions
  - Listen and process an alert message
  - Install the security actions to the data-plane
  - Security application on a controller



# Challenge in the flow-level security deployment

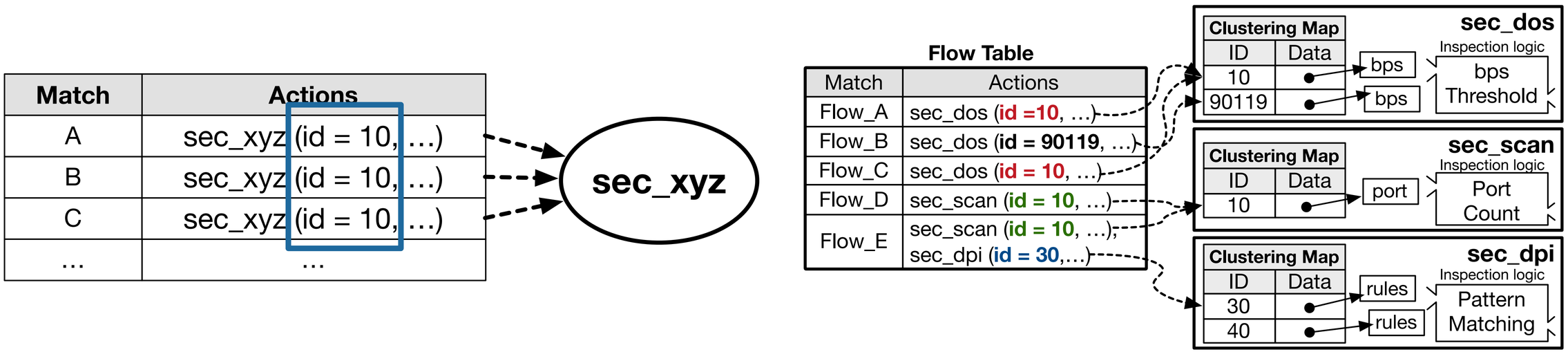
- The flow-level security deployment can't represent a security policy across multiple flows
  - *Simple example:*




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

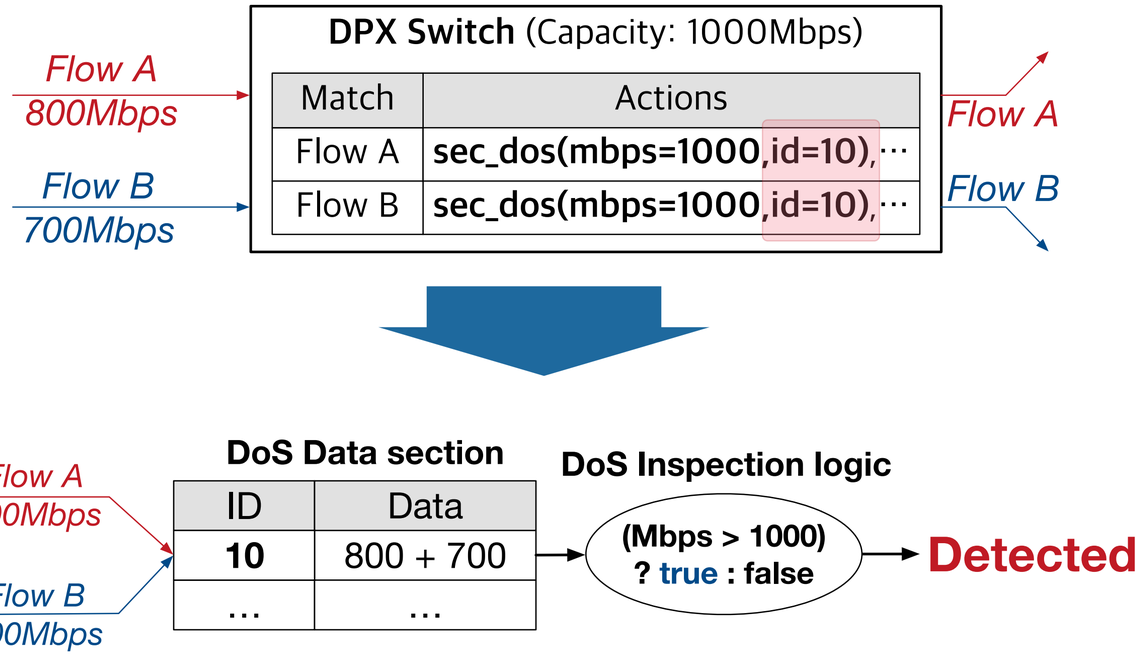
# Action Clustering

- All security actions have a cluster ID in their parameter
  - The security 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 section by the cluster map



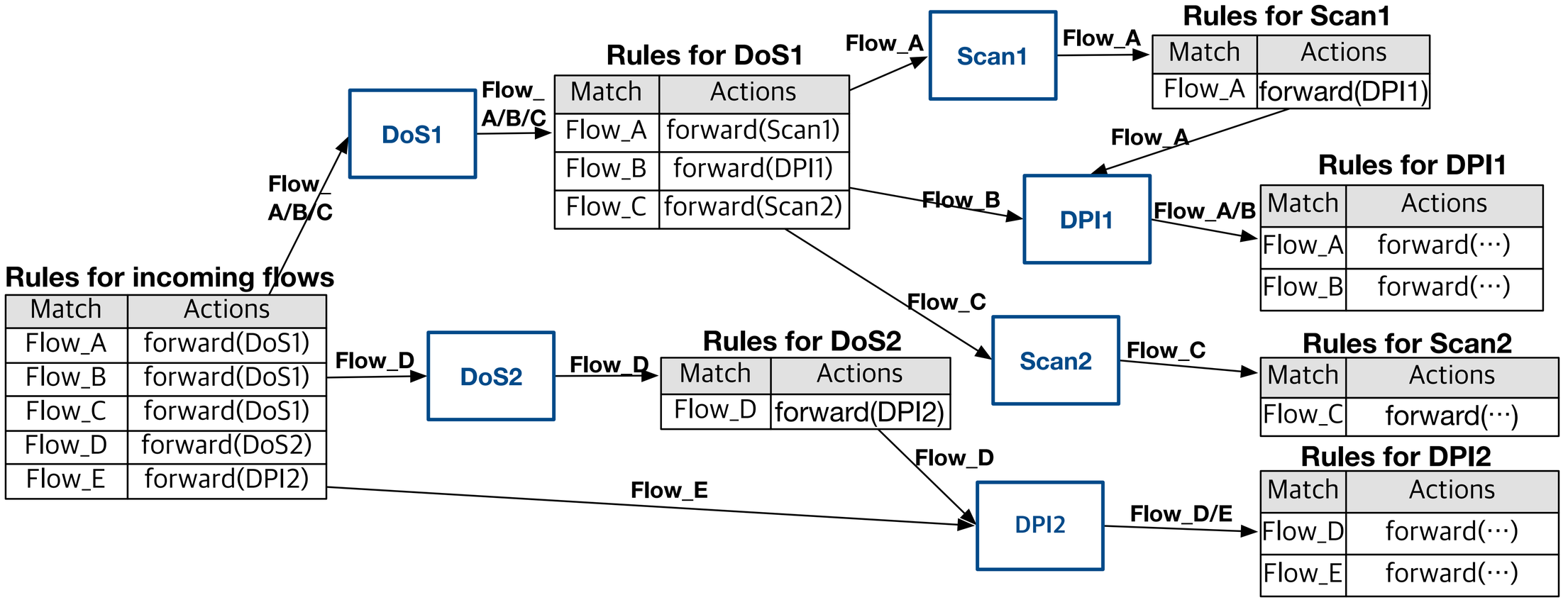
# Applying Action Clustering

- Applying the action clustering to the previous example




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

# Applying DPX



# Applying DPX

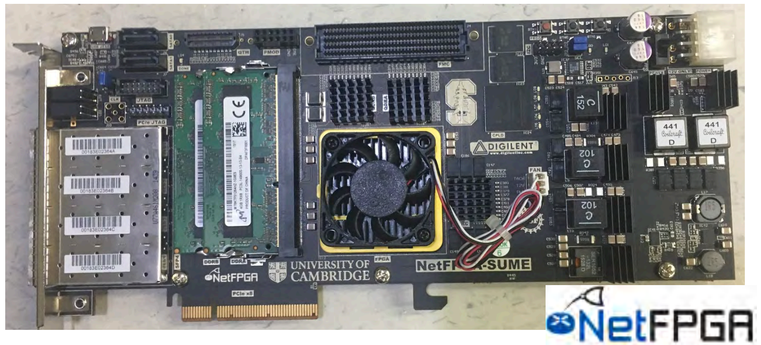
| Match | Actions |
|---|---|
| Flow_A | sec_dos(**id=10**, ...), sec_scan(**id=10**, ...),sec_dpi(**id=10**, ...), ... |
| Flow_B | sec_dos(**id=10**, ...), sec_dpi(**id=10**, ...), ... |
| Flow_C | sec_dos(**id=10**, ...), sec_scan(**id=20**, ...), ... |
| Flow_D | sec_dos(**id=20**, ...), sec_dpi(**id=20**, ...), ... |
| Flow_E | sec_dpi(**id=20**, ...), ... |

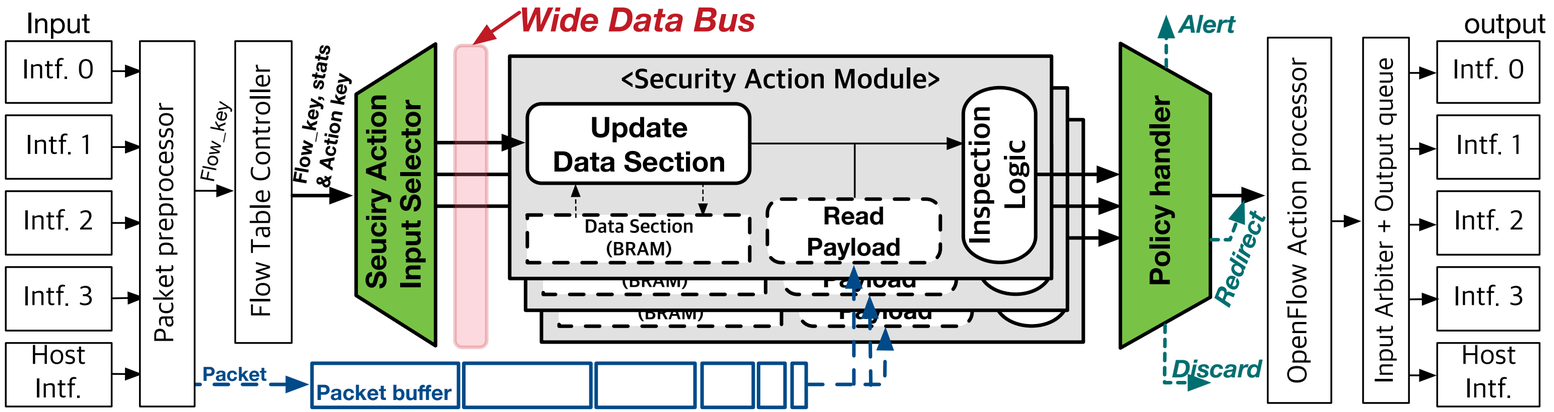
# Implementation

- Prove our design in both hardware and software:

# Implementation

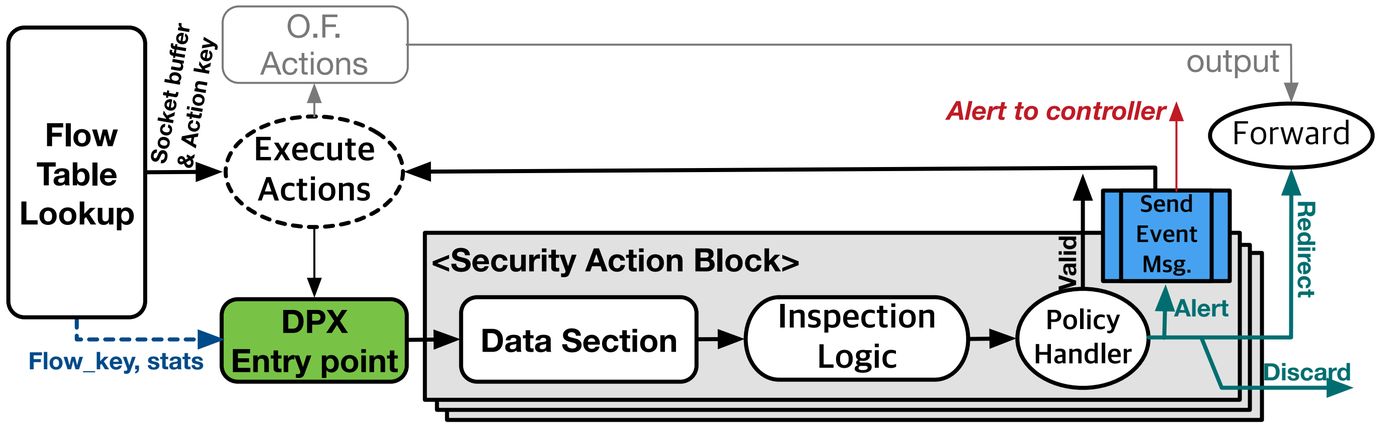
- Prove our design in both hardware and software:
  - NetFPGA-SUME, FPGA-based PCI Express board for 10 and 100 Gbps operation
  - Support *DoS detector* and *Deep Packet Inspector*



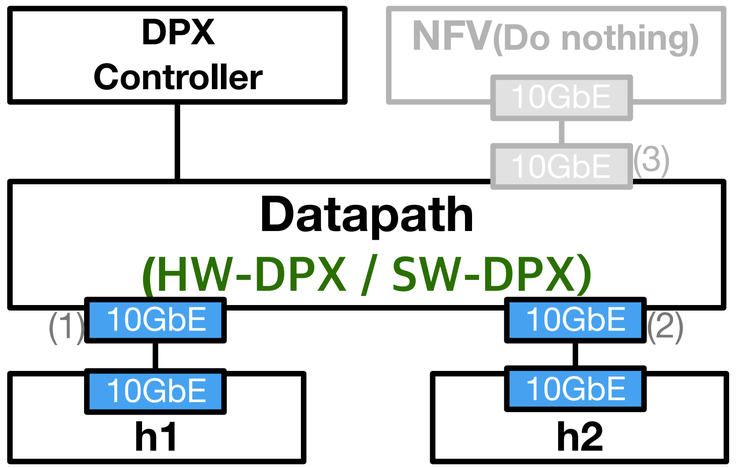
# Implementation

- Prove our design in both hardware and software:
  - Open vSwitch, Open-source implementation of a distributed virtual switch
  - Support *DoS detector, Vertical/Horizontal Scanning detector, Anomaly Detector, Session Monitor* and *Deep Packet Inspector*

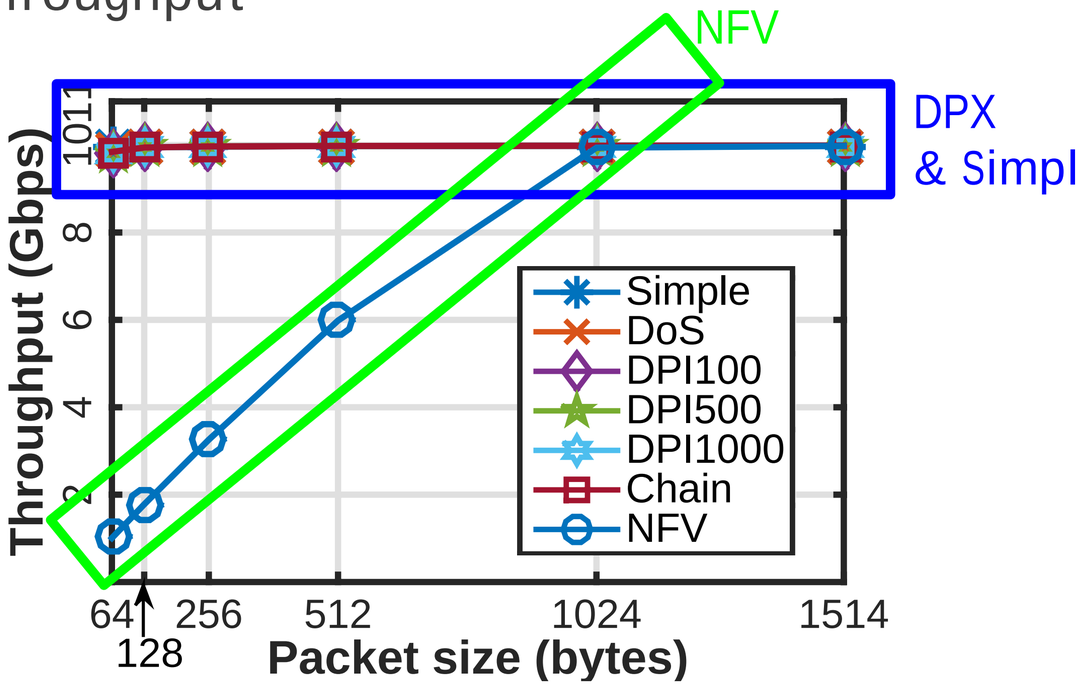
# Performance Evaluation

- Measured the performance of
  - Each security action
  - Service chain of all available security actions
  - Simple forwarding
  - Naive NFV which does nothing

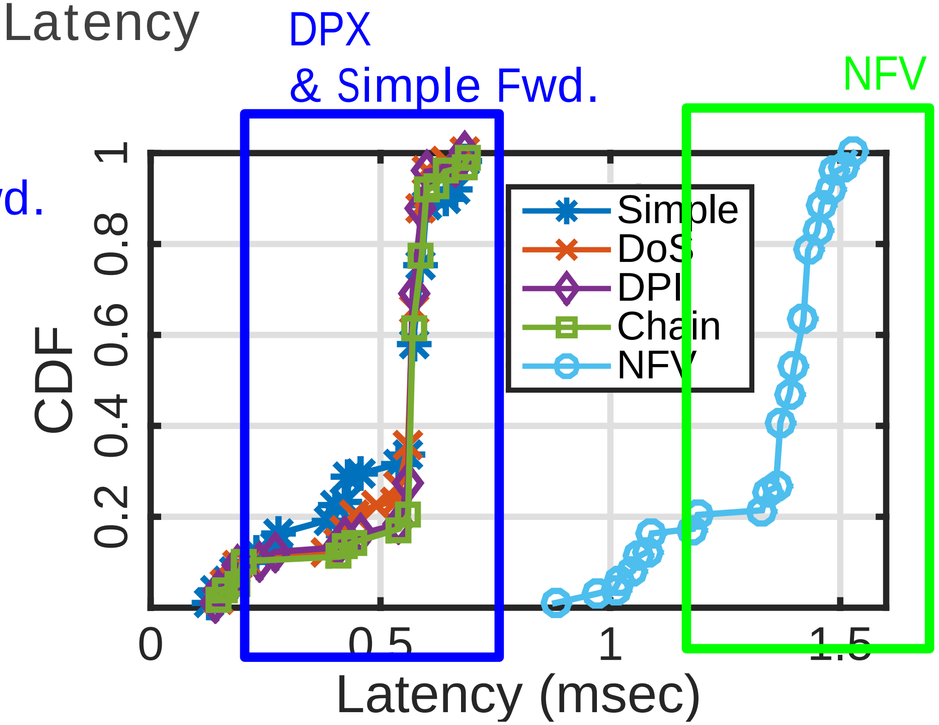


# Performance Evaluation_hardware

- Throughput

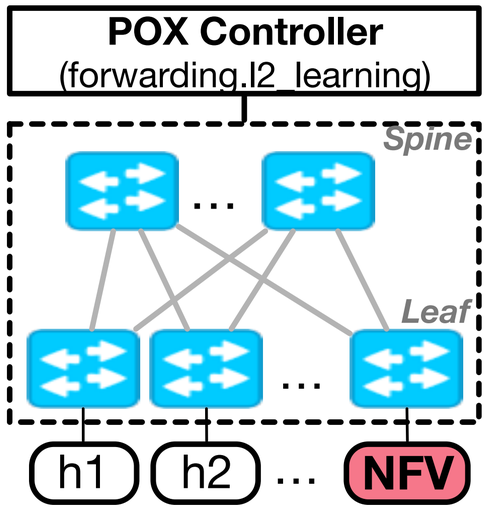


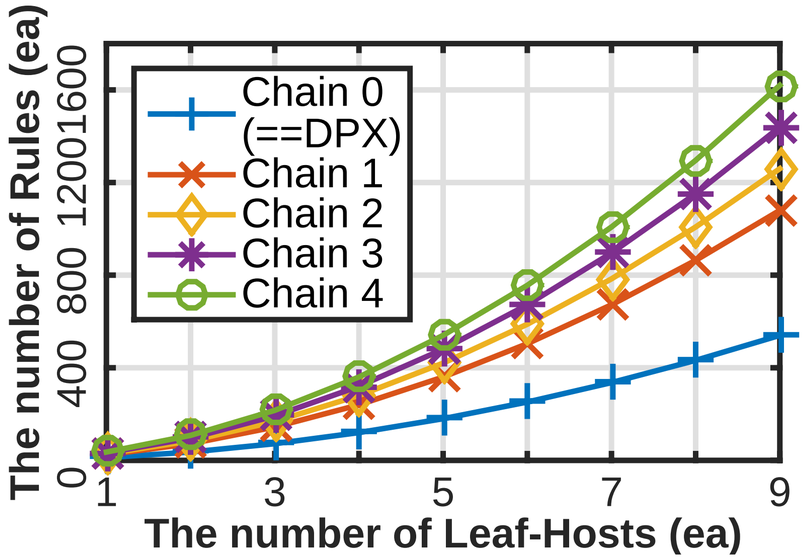

- Latency



# Flow-table Simplification

- Assuming a leaf-spine topology network with increasing the number of hosts
  - The hosts have to visit/use arbitrary service chains varying length.
- Measured the required number of rules for passing a *pingall* test



# Conclusion

- Provide security services as a part of packet processing logic
  - As as a set of actions
  - Support the security policy and the controller API set
  - Action clustering
- Achieve the simplified management and high-performance
  - Catches both advantages of a middlebox and SDN application for security in SDN.
- Expect that the approach of DPX has high-potential in complex network nowadays

Thank you!

Questions?

http://nss.kaist.ac.kr
taejune.park@kaist.ac.kr