SDNs for Enforcement of Dynamic NFV-Policies

Ashwin Rao

04 / 11 / 2015

Background

- Increasing presence of Network Functions (NF)

  Sherry et al. "Making Middleboxes Someone Else's Problem: Network Processing as a Cloud Service." In SIGCOMM '12.

- Need for dynamic policies
  - "The current service function deployment models are relatively static ... greatly reducing the ability of an operator to introduce new services..."

    J. Halpern et al. RFC 7665. "Service Function Chaining (SFC) Architecture." October 2015.

Challenges in Policy Enforcement

- NF Composition
- Resource Management
- Packet Modification

Challenges in Policy Enforcement

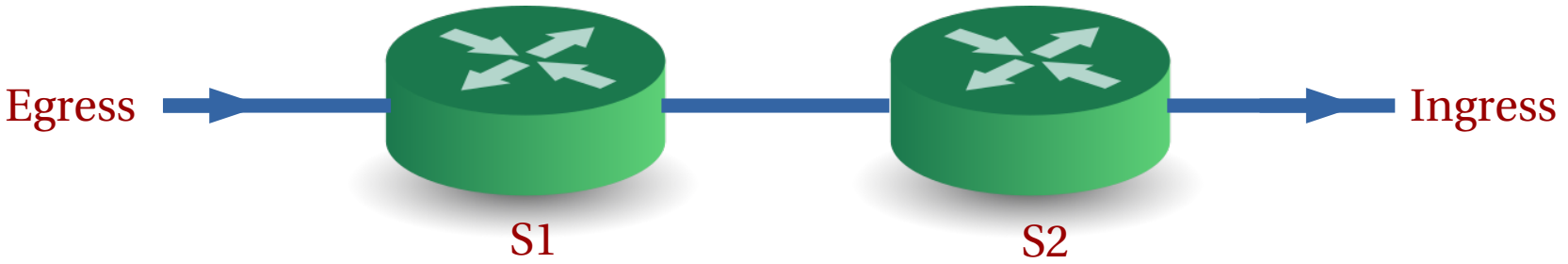
- NF Composition
- Resource Management
- Packet Modification

  Policy: Egress → Firewall → IDS → Proxy → Ingress

Challenges in Policy Enforcement

- NF Composition
- Resource Management
- Packet Modification

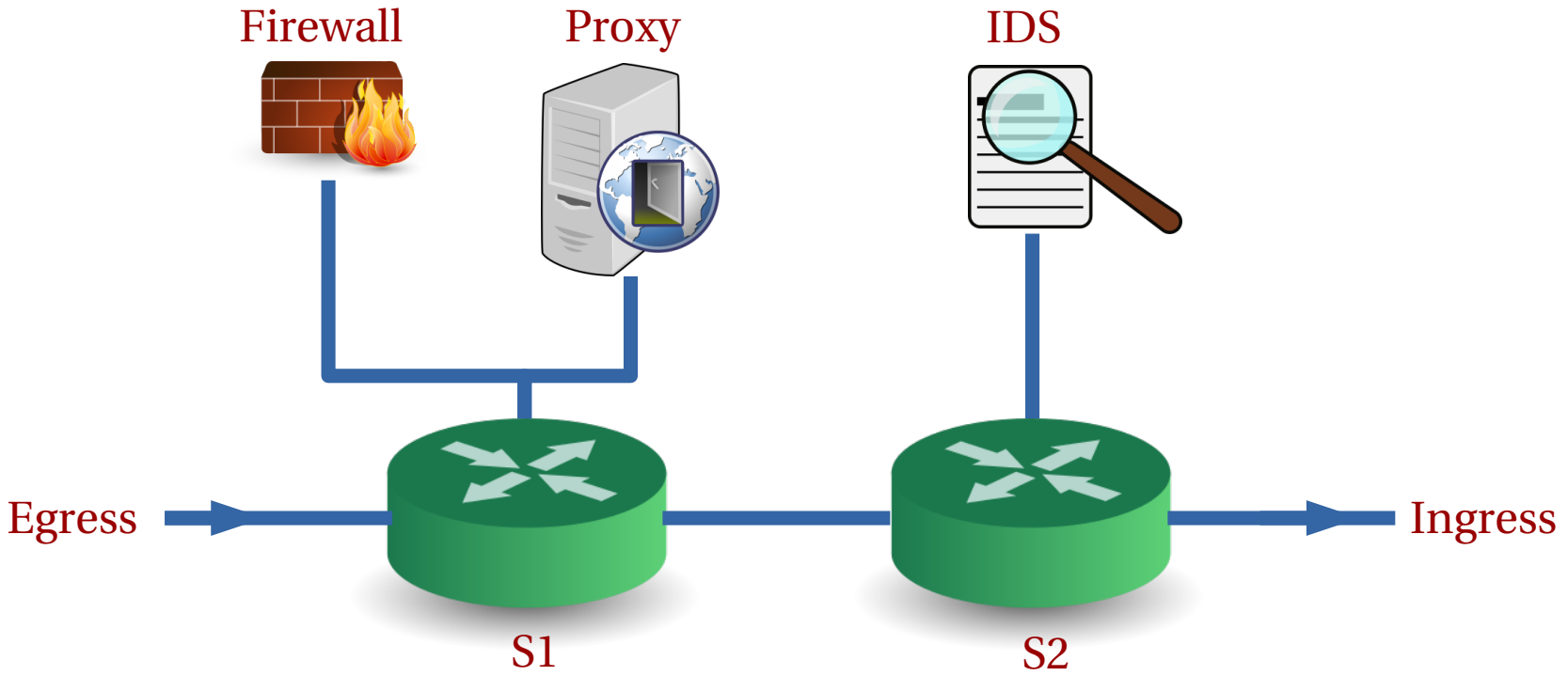
  Policy: Egress → Firewall → IDS → Proxy → Ingress



Challenges in Policy Enforcement

- NF Composition
- Resource Management
- Packet Modification

Policy: Egress → Firewall → IDS → Proxy → Ingress

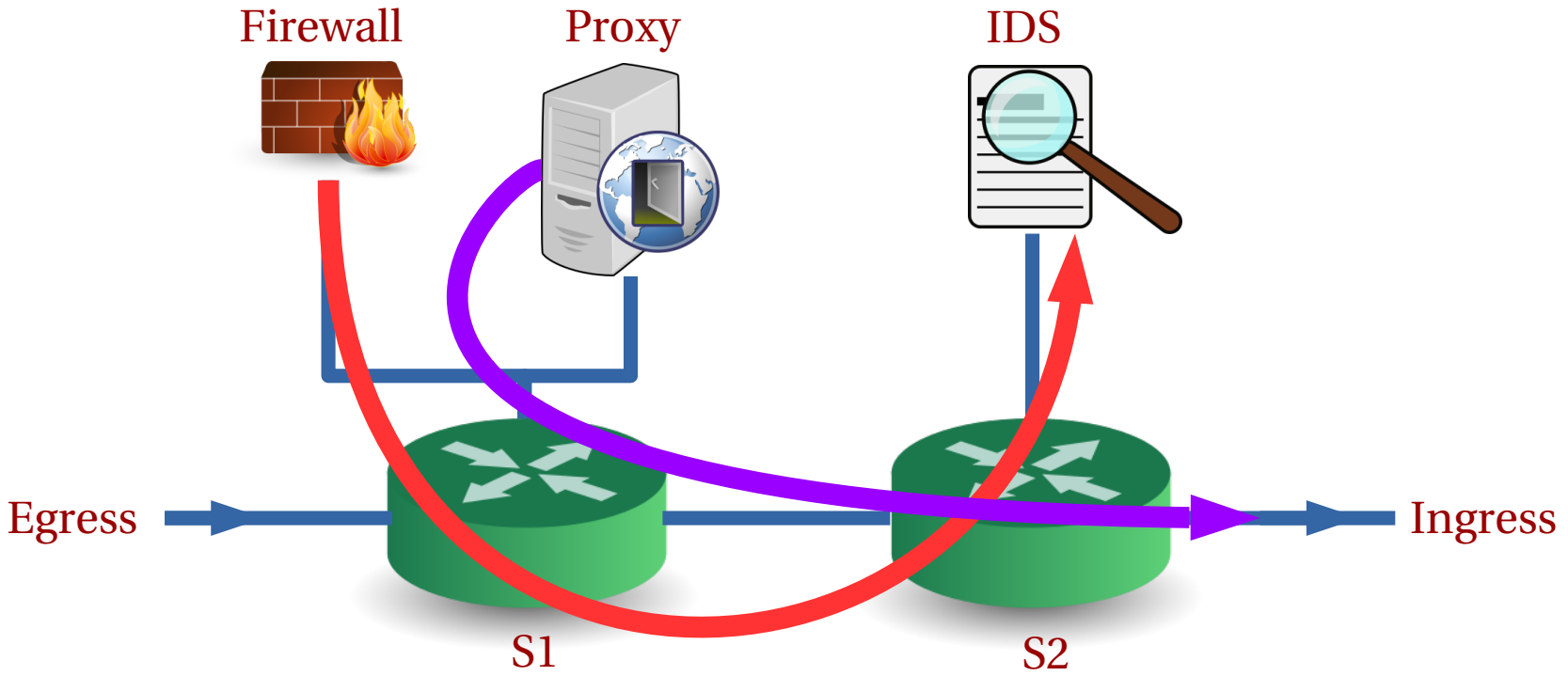


Challenges in Policy Enforcement

- NF Composition
- Resource Management
- Packet Modification

What must S2 do for a packet from S1?

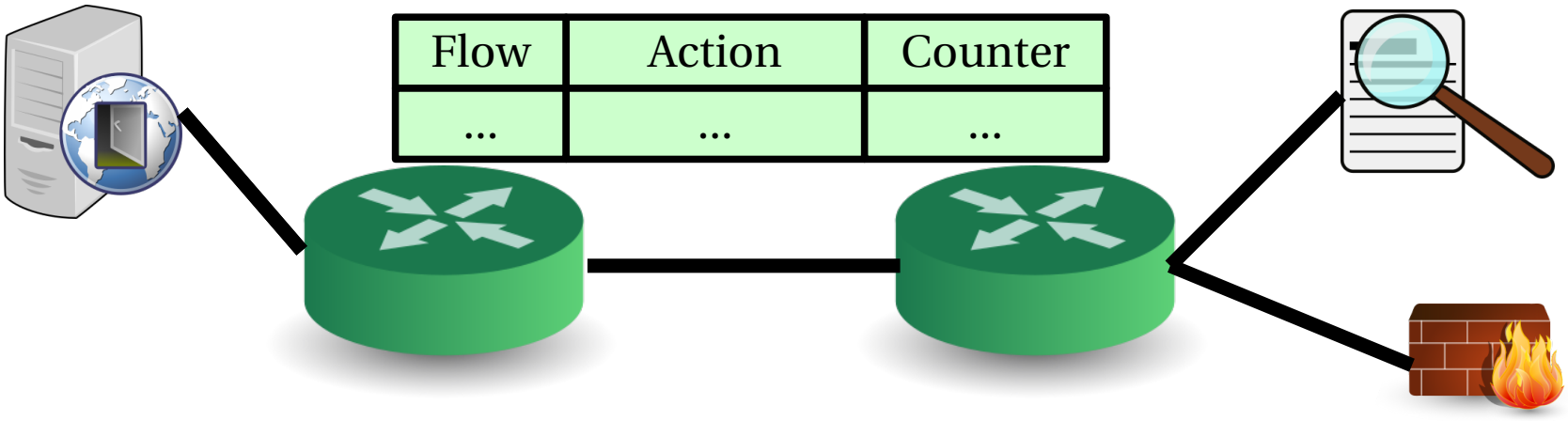
Policy: Egress → Firewall → IDS → Proxy → Ingress



Outline

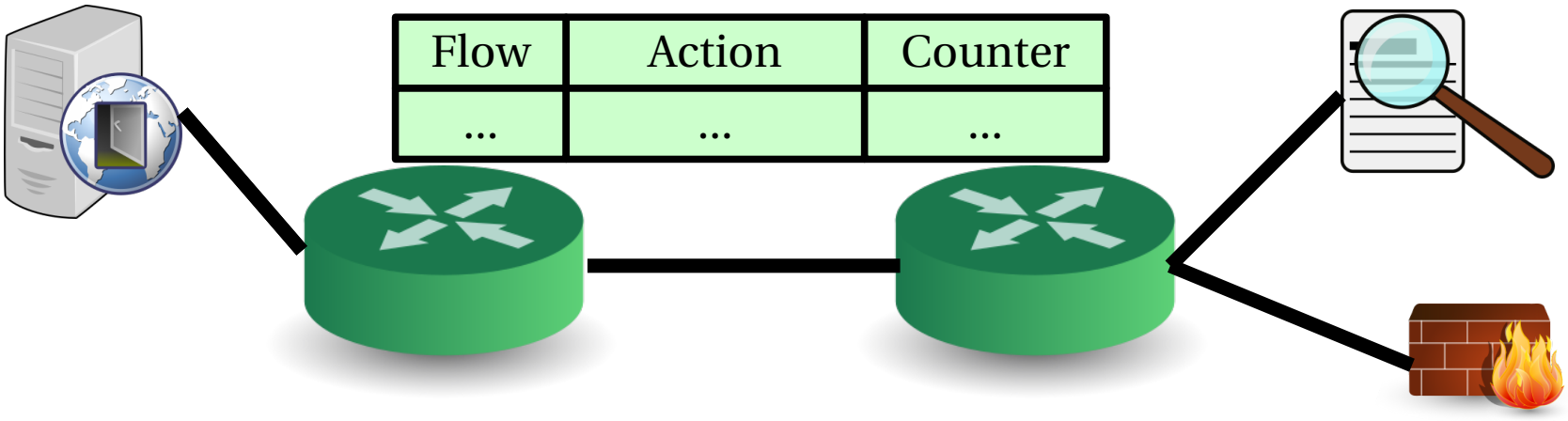
- Background
- **SIMPLE**-fying Middlebox Policy Enforcement Using SDN
- Enforcing Network-Wide Policies in the Presence of Dynamic Middlebox Actions using **FlowTags**
- **IETF Protocols**: NSH and others

SIMPLE Approach



SIMPLE Approach

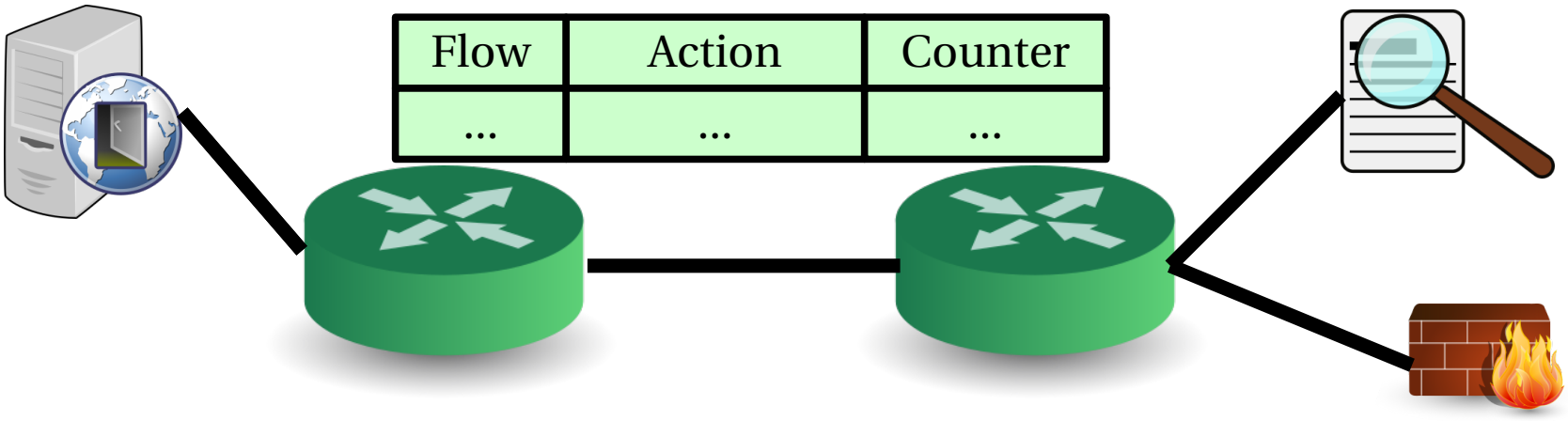
Traffic Matrix



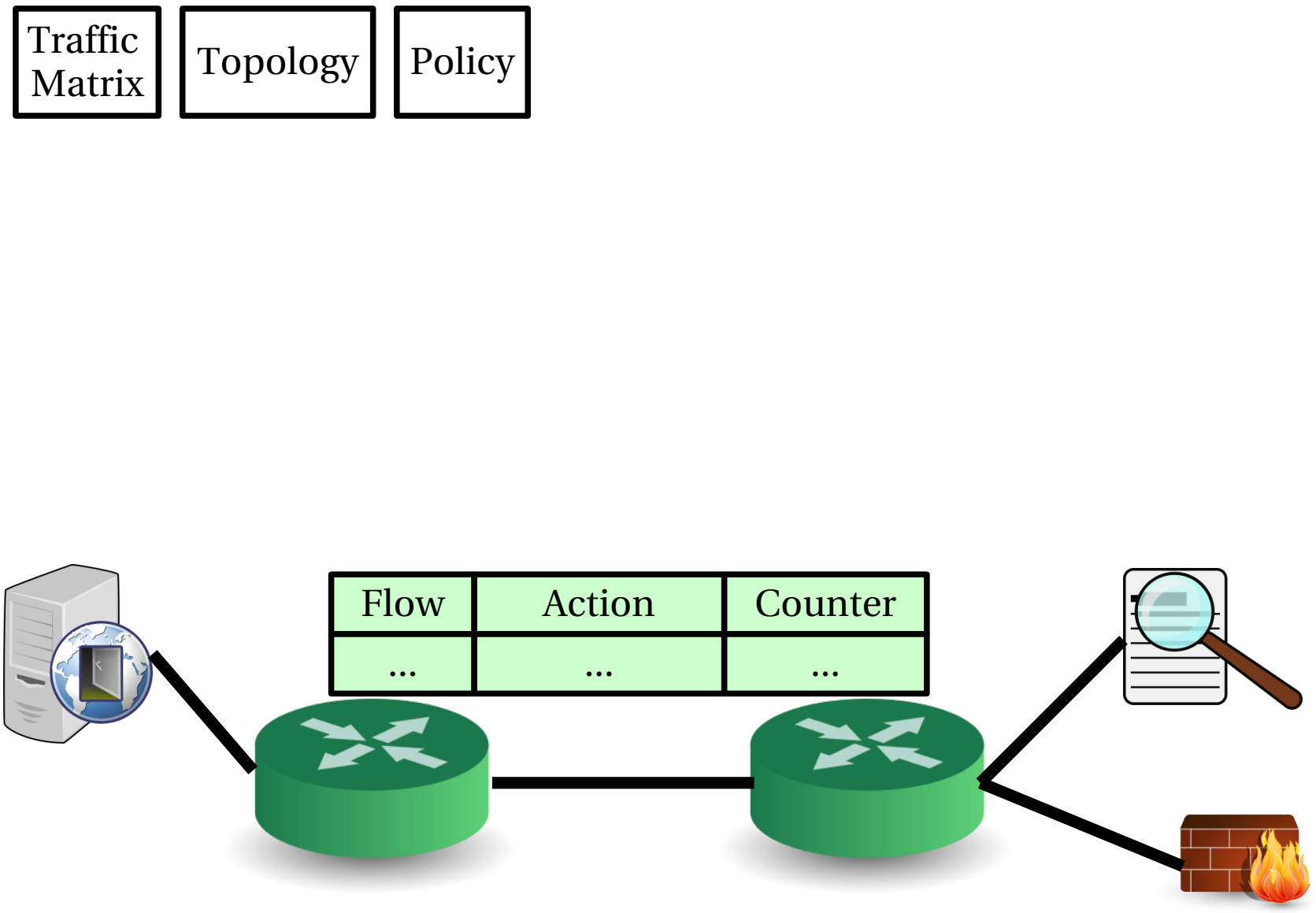
SIMPLE Approach

Traffic Matrix

Topology

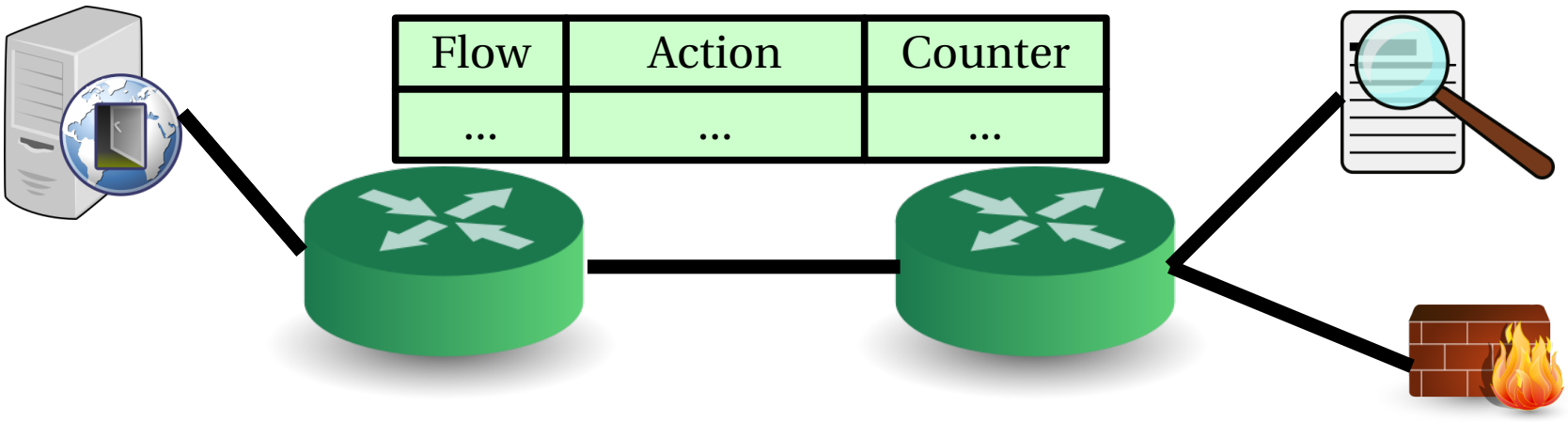


SIMPLE Approach

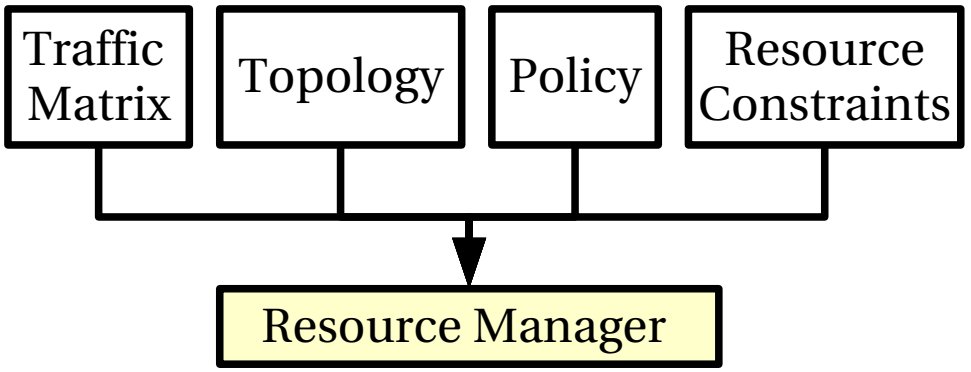


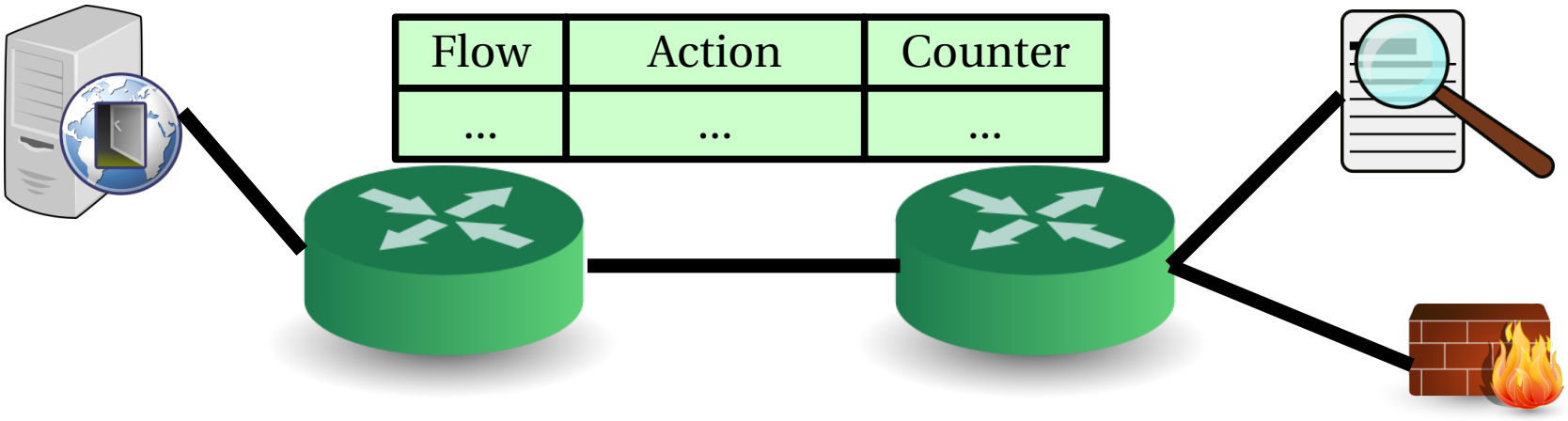
SIMPLE Approach

Traffic Matrix	Topology	Policy	Resource Constraints

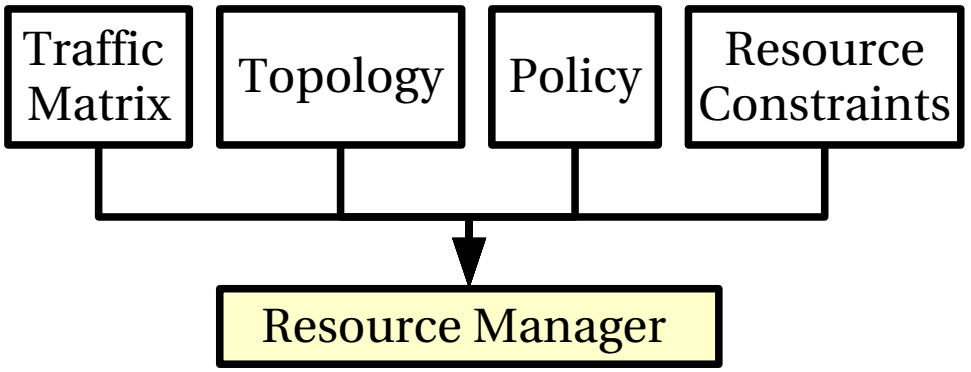


SIMPLE Approach



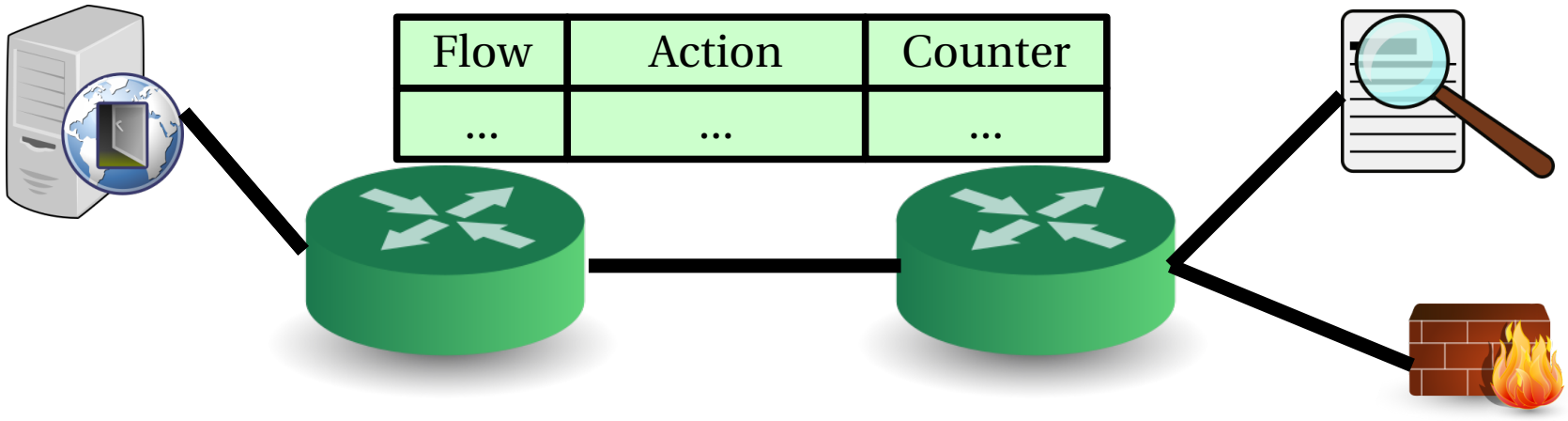


SIMPLE Approach

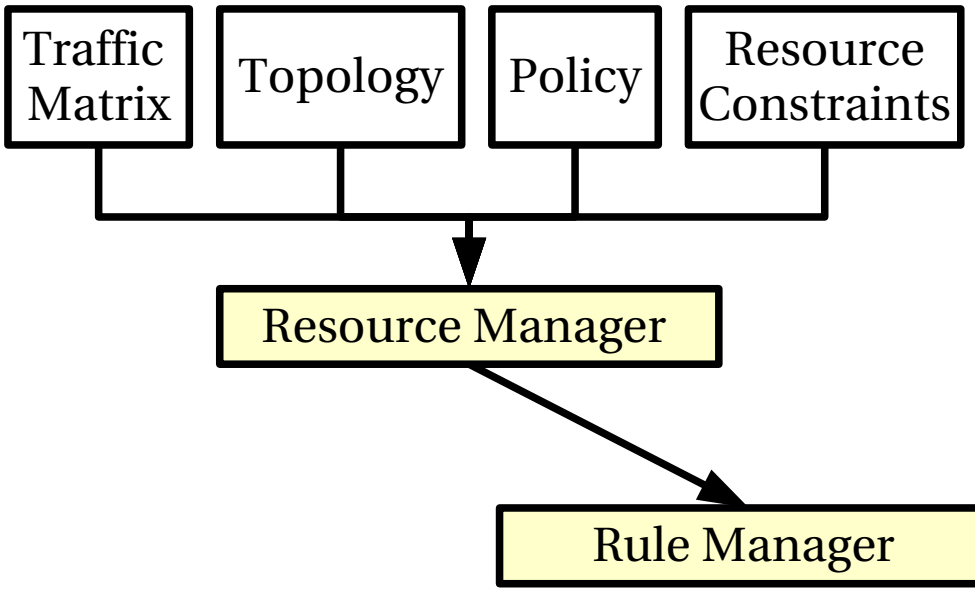


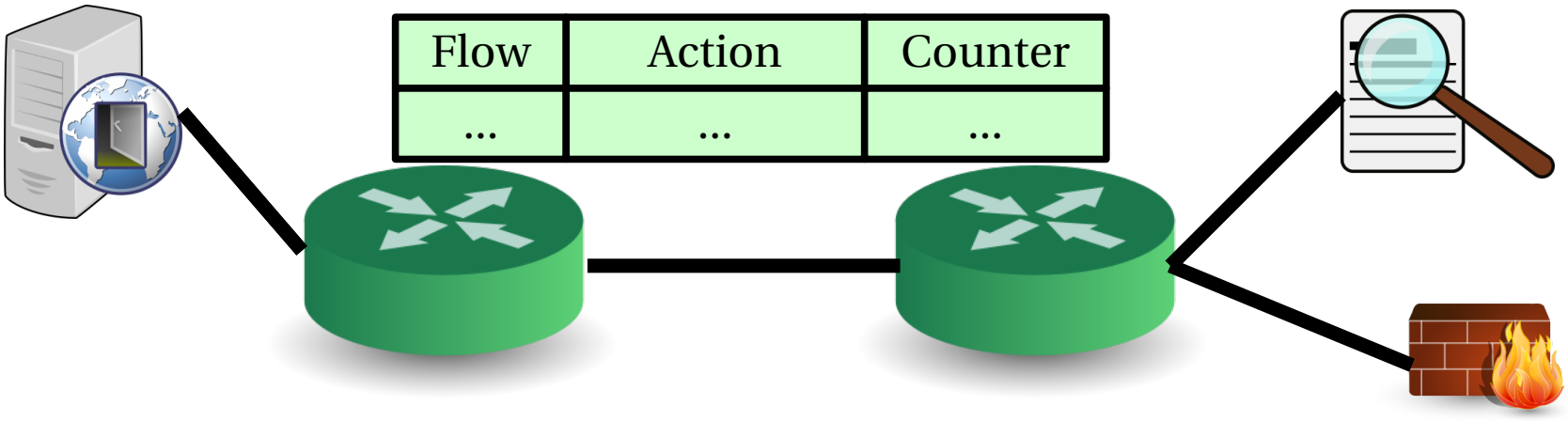
Offline-Online Decomposition

- Offline: Switch constraints
- Online: Load Balancing
- ILP formulation

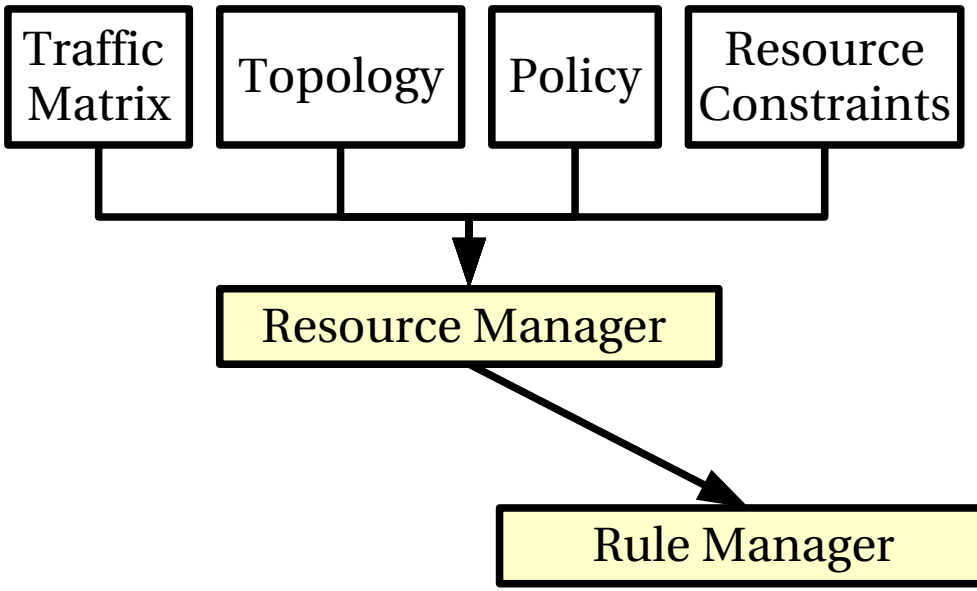


SIMPLE Approach

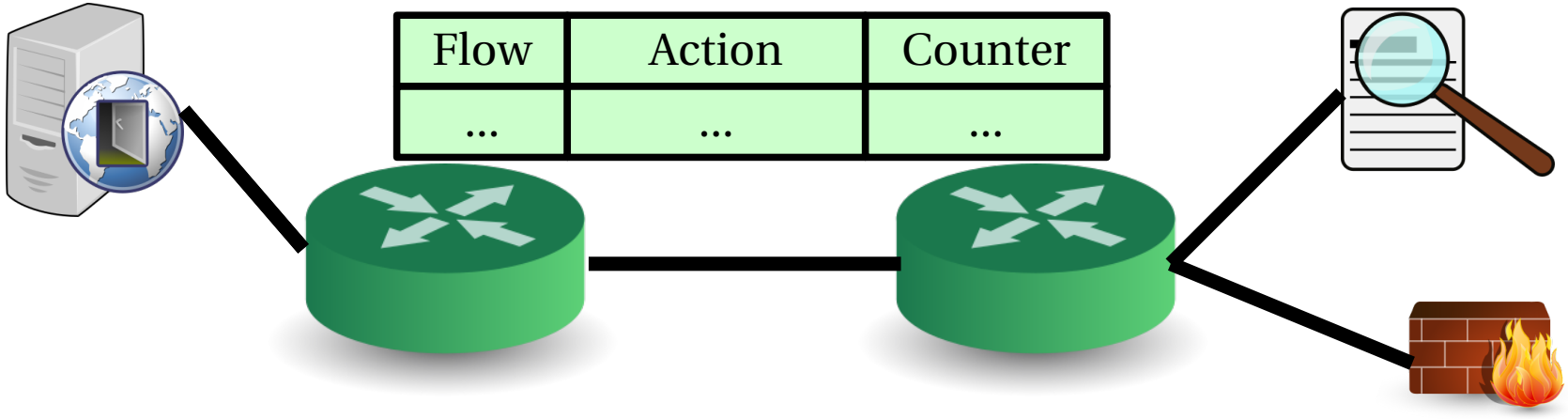




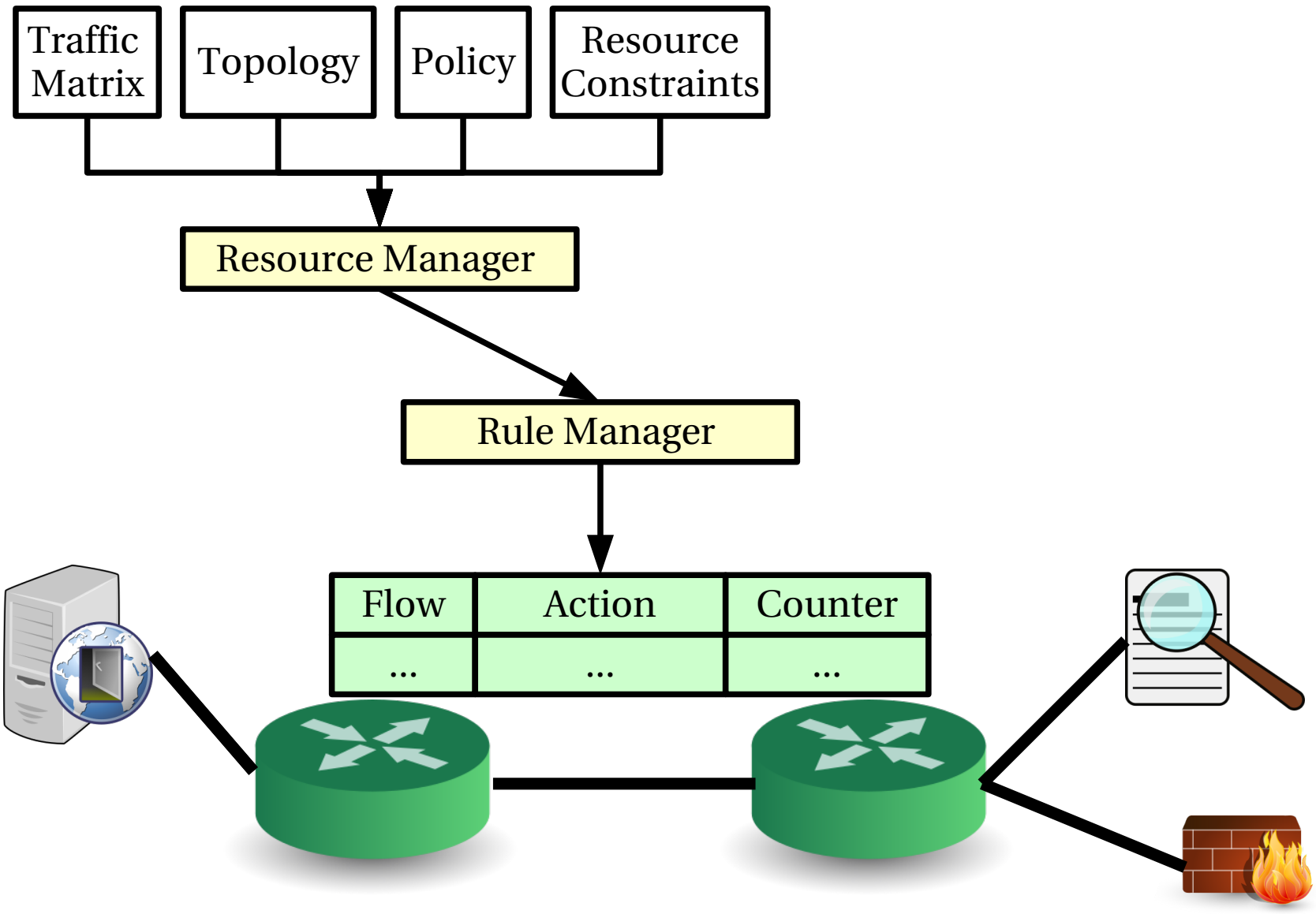
SIMPLE Approach



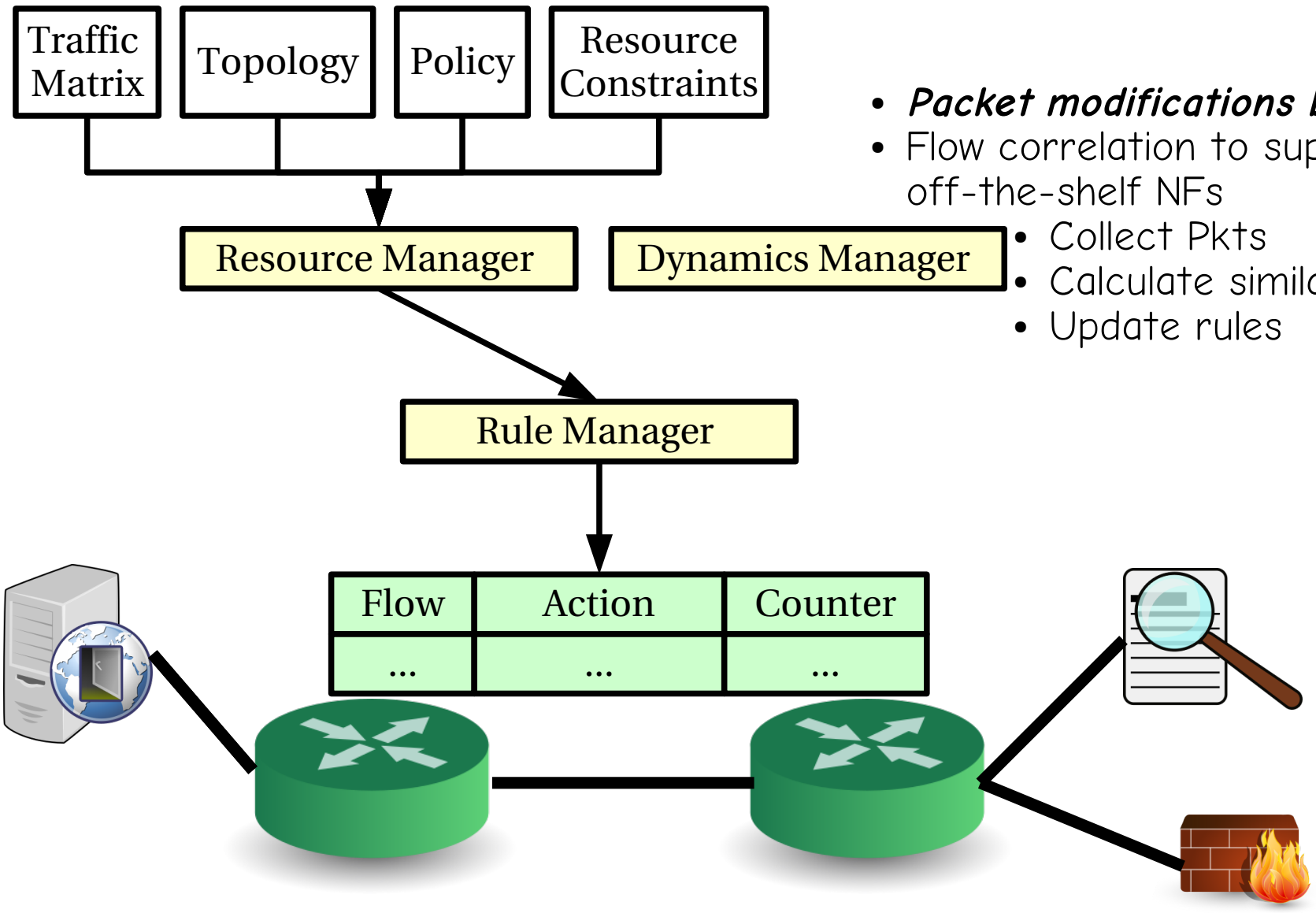
- ***Unambiguous Forwarding***
  - In-port and Out-port
  - Adding Processing State (ProcState) in headers
- ***Compacting Forwarding Rules***



SIMPLE Approach

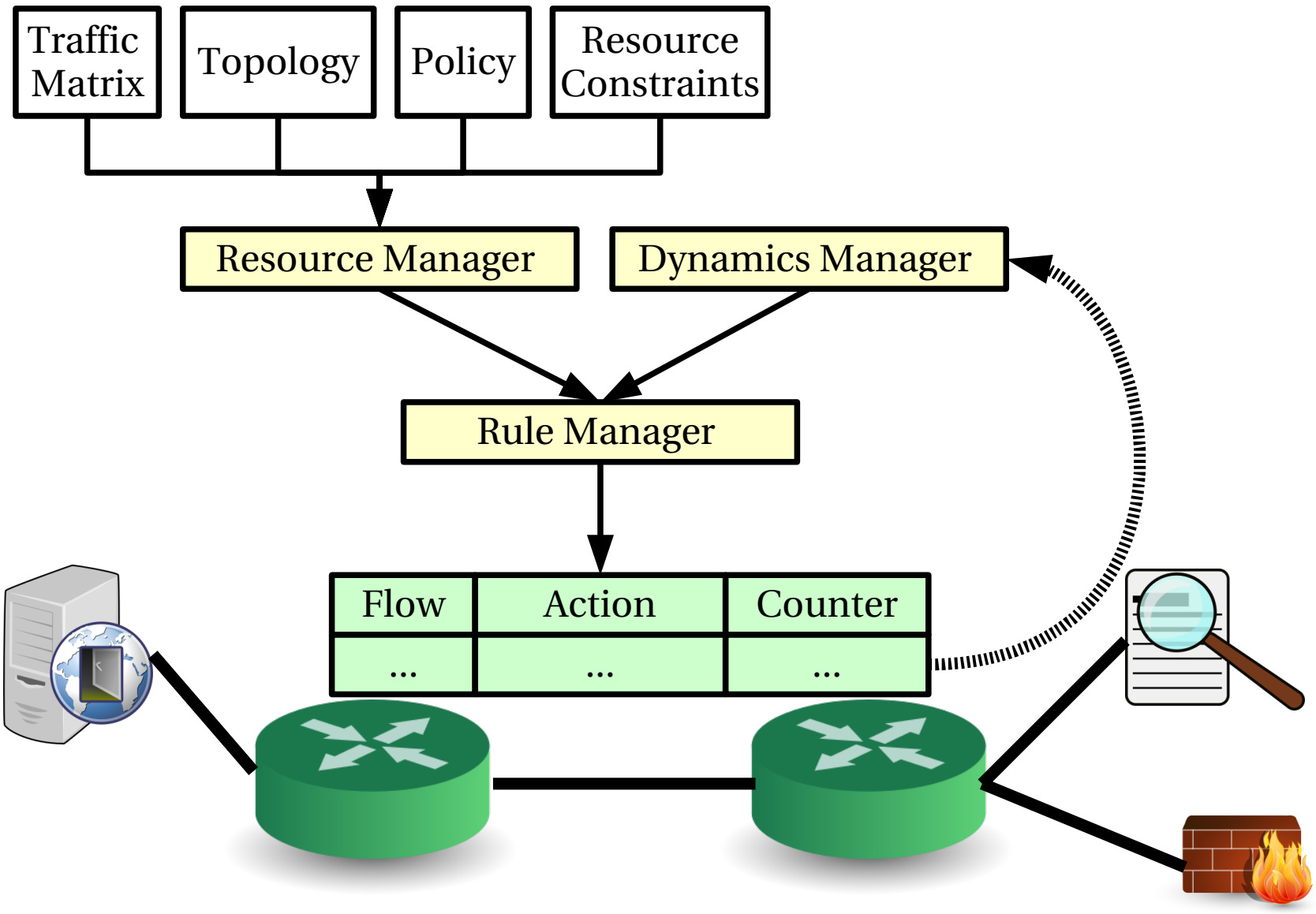


SIMPLE Approach



- ***Packet modifications by NFs***
- Flow correlation to support off-the-shelf NFs
  - Collect Pkts
  - Calculate similarity
  - Update rules

SIMPLE Approach



SIMPLE Contributions

- SDN based Policy Enforcement Layer for NF-specific traffic steering

SIMPLE Contributions

- SDN based Policy Enforcement Layer for NF-specific traffic steering
- Supports off-the-shelf NFs
  - Leverages tunnels between switches and SDN capabilities to modify packet headers

SIMPLE Contributions

- SDN based Policy Enforcement Layer for NF-specific traffic steering
- Supports off-the-shelf NFs
  - Leverages tunnels between switches and SDN capabilities to modify packet headers
- Decompose optimization problem
  - Offline component for switch capabilities
  - Online component for load balancing

SIMPLE Contributions

- SDN based Policy Enforcement Layer for NF-specific traffic steering
- Supports off-the-shelf NFs
  - Leverages tunnels between switches and SDN capabilities to modify packet headers
- Decompose optimization problem
  - Offline component for switch capabilities
  - Online component for load balancing
- Learns packet modifications by NFs

Discussion on SIMPLE

- Benefits
  - Flexibility in NF placement
  - Reconfigure rules on NF failure
  - Takes ≈ 1 second for large AS topology
- Issues
  - Flow correlation
    - Latency & True-Positive/False-Positive Rates of popular NFs unknown
  - Short-term solution
    - Will networks have SDN switches with legacy NFs?
    - **What if NFs can be modified?**

SDN Tenets violated by NFs

- **ORIGINBINDING**
  - Strong binding between packet and its origin
  - Example culprit: NAT, load balancers
- **PATHSFOLLOWPOLICY**
  - Explicit policies should determine the packet path
  - Example culprit: Caches
- **HIGHLEVELNAMES**
  - Network policies should be expressed in terms of high-level names.
  - Example culprit: NAT

Concept of FlowTags

NFs add tags that contain

1) Missing Bindings to ensure ORIGINBINDINGS
2) Missing Context to ensure PATHFOLLOWPOLICY

Assumptions

1) Adequate header bits available
2) Possibility to modify/extend NF software

Tag Generation

1) Static Policy Graph (DPG)
   - Traffic ↔ Chain to NFs
2) Dynamic Policy Graph (DPG)
   - IN and OUT Nodes 
   - NF Nodes 

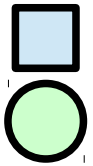
Tag Generation

1) Static Policy Graph (DPG)

- Traffic ↔ Chain to NFs

2) Dynamic Policy Graph (DPG)

- IN and OUT Nodes
- NF Nodes



Internet

Host 2

DROP

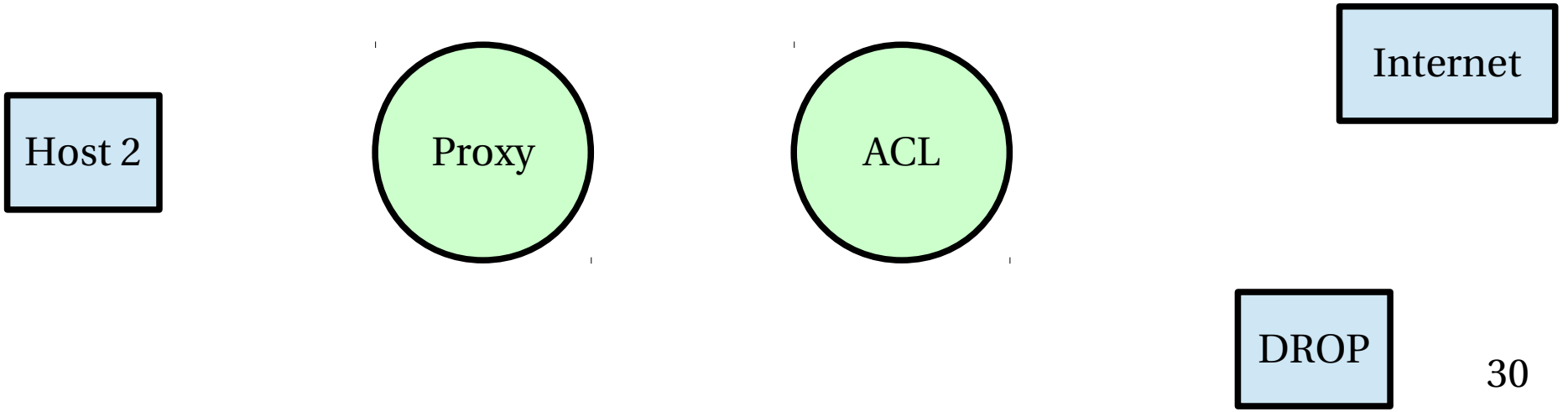
Tag Generation

1) Static Policy Graph (DPG)

- Traffic ↔ Chain to NFs

2) Dynamic Policy Graph (DPG)

- IN and OUT Nodes
- NF Nodes



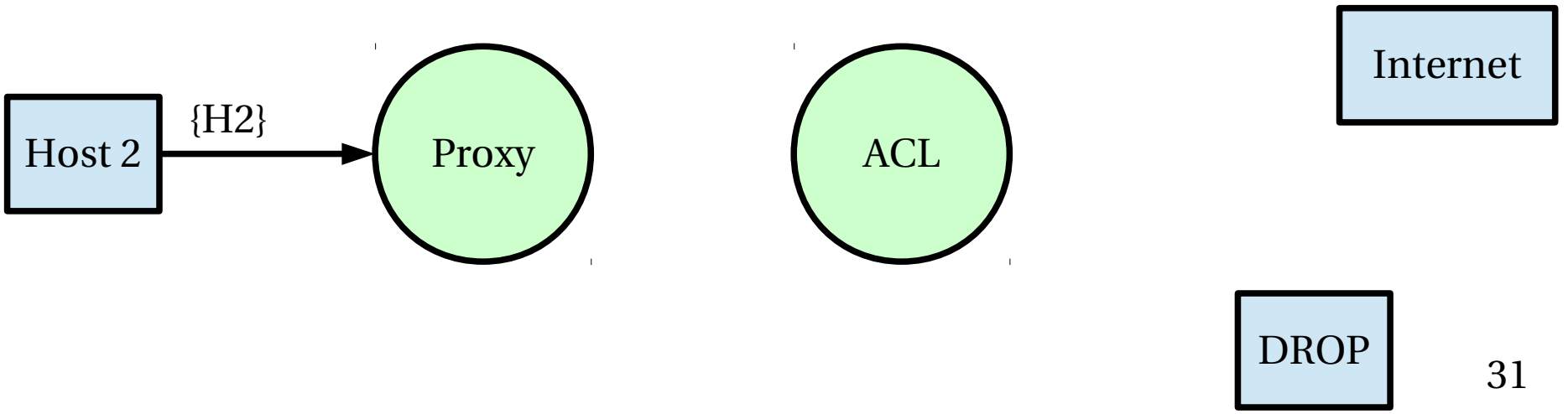
Tag Generation

1) Static Policy Graph (DPG)

- Traffic ↔ Chain to NFs

2) Dynamic Policy Graph (DPG)

- IN and OUT Nodes
- NF Nodes



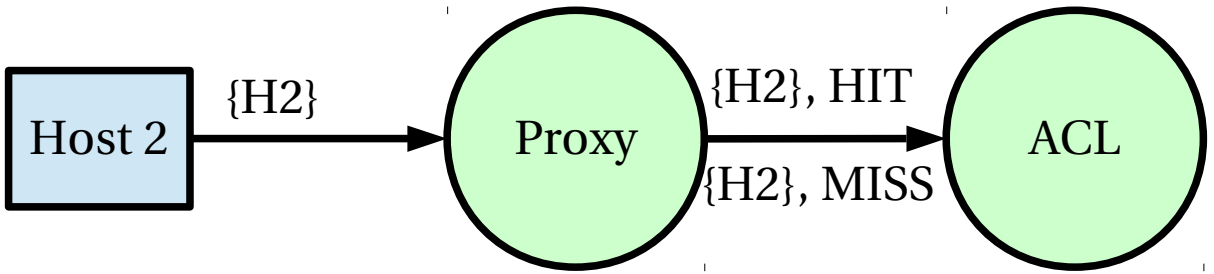
Tag Generation

1) Static Policy Graph (DPG)

- Traffic ↔ Chain to NFs

2) Dynamic Policy Graph (DPG)

- IN and OUT Nodes
- NF Nodes



Internet

DROP

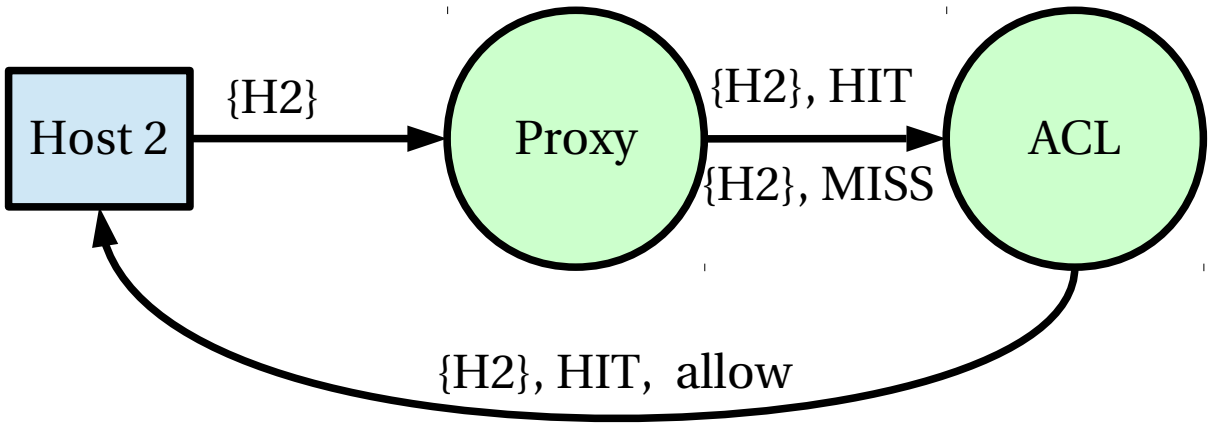
Tag Generation

1) Static Policy Graph (DPG)

- Traffic ↔ Chain to NFs

2) Dynamic Policy Graph (DPG)

- IN and OUT Nodes
- NF Nodes

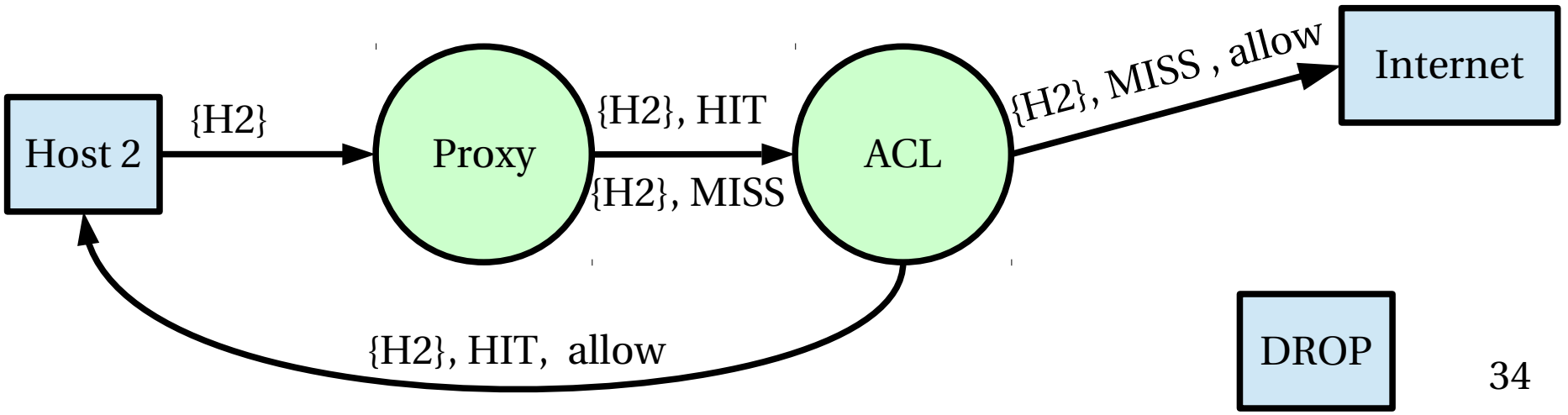


Internet

DROP

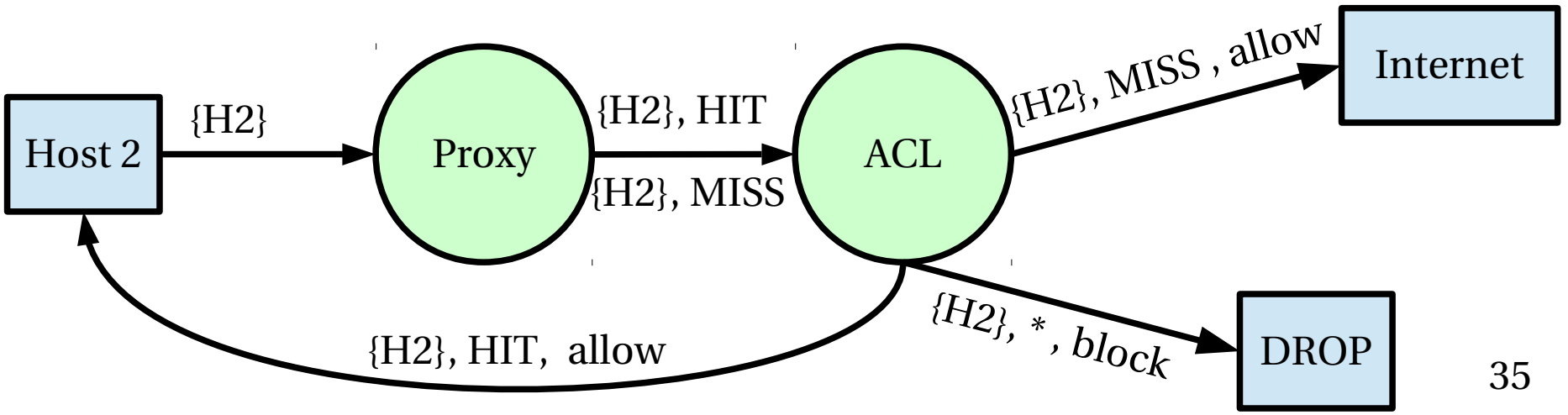
Tag Generation

1) Static Policy Graph (DPG)
   - Traffic ↔ Chain to NFs
2) Dynamic Policy Graph (DPG)
   - IN and OUT Nodes
   - NF Nodes



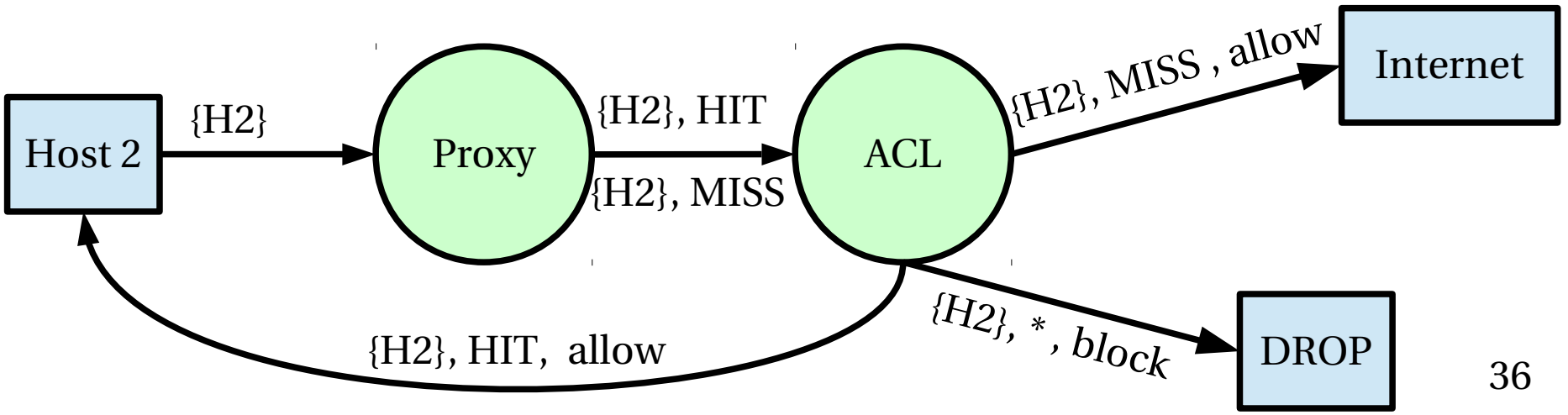
Tag Generation

1) Static Policy Graph (DPG)
   - Traffic ↔ Chain to NFs
2) Dynamic Policy Graph (DPG)
   - IN and OUT Nodes
   - NF Nodes

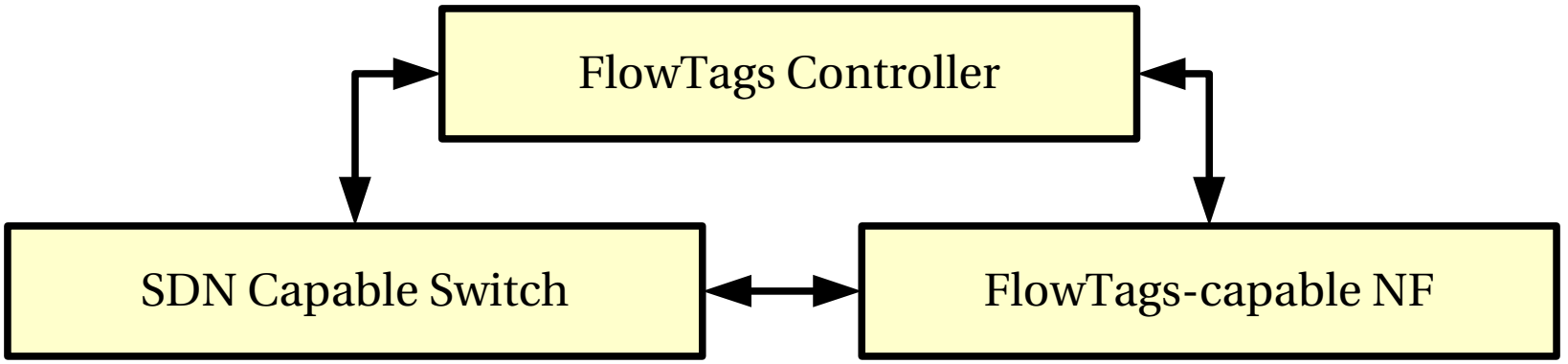


Tag Generation

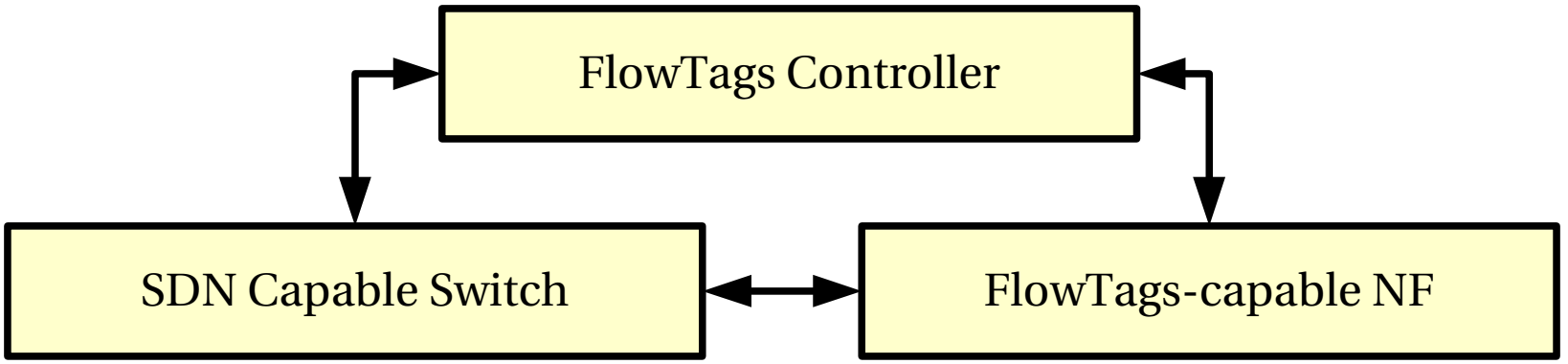
1) Static Policy Graph (DPG)
   - Traffic ↔ Chain to NFs
2) Dynamic Policy Graph (DPG)
   - IN and OUT Nodes
   - NF Nodes
   - **Unique Tag for a packet on the edge of the graph**



FlowTags Components



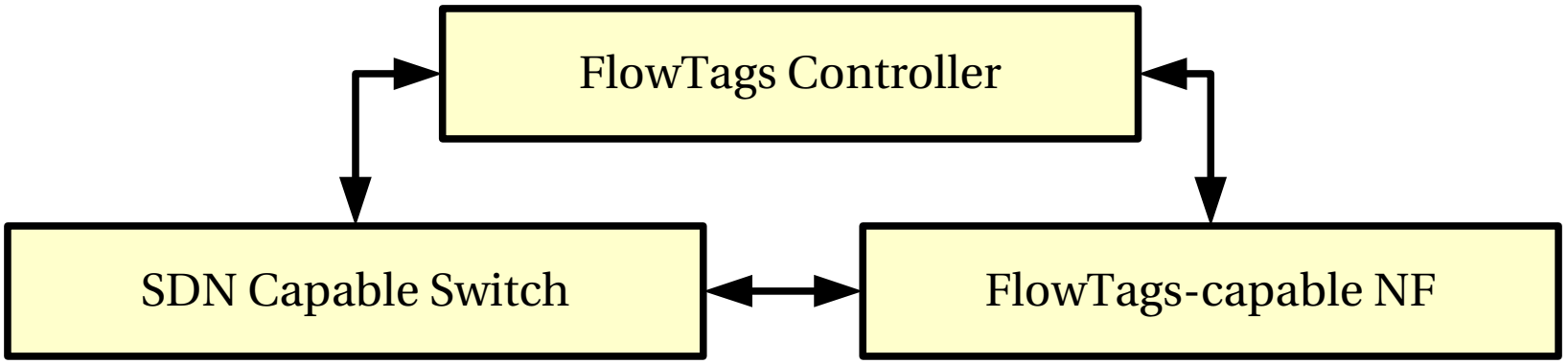
FlowTags Components



API

- FT_GENERATE_QRY & FT_GENERATE_RSP – for Tag generation

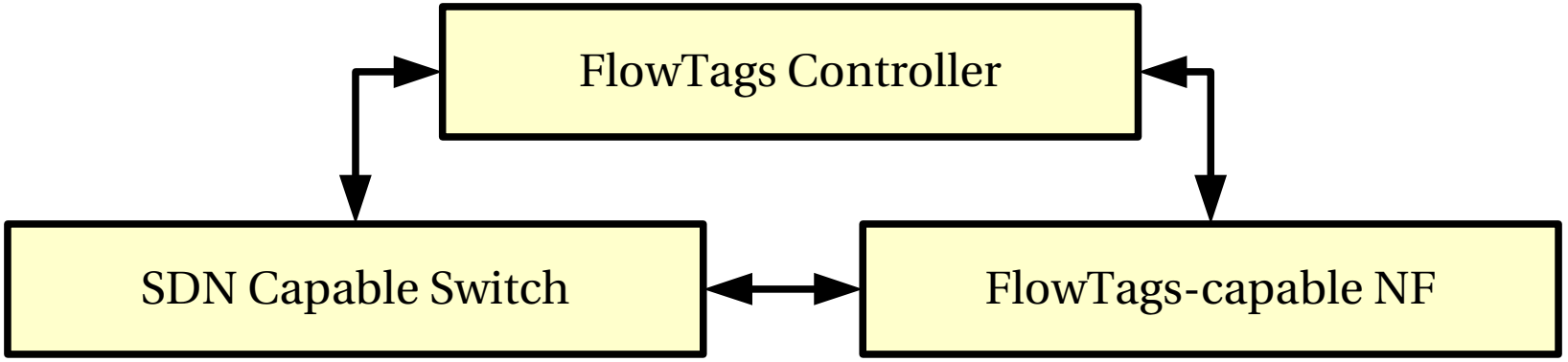
FlowTags Components



API

- FT_GENERATE_QRY & FT_GENERATE_RSP – for Tag generation
- FT_CONSUME_QRY & FT_CONSUME_RSP - for Tag consumption

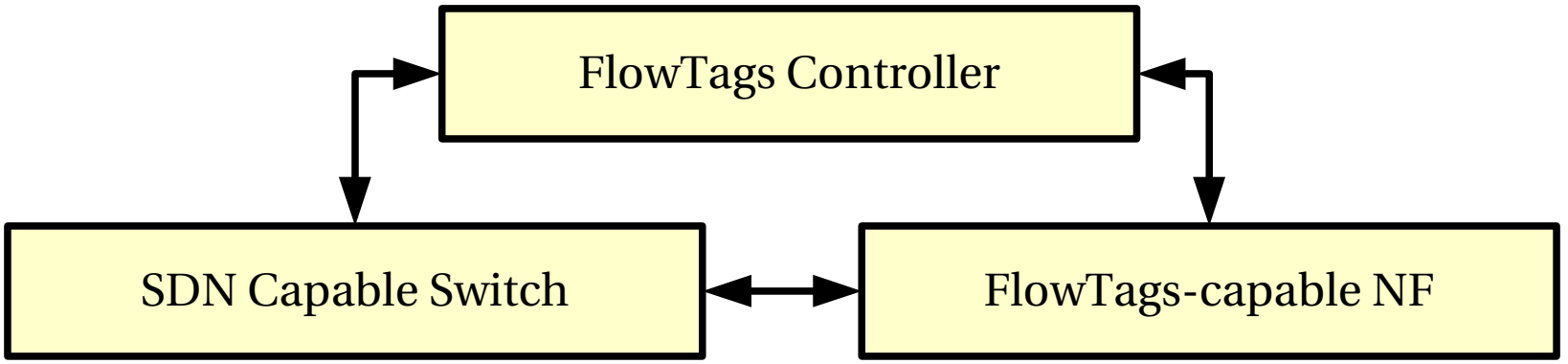
FlowTags Components



API

- FT_GENERATE_QRY & FT_GENERATE_RSP – for Tag generation
- FT_CONSUME_QRY & FT_CONSUME_RSP - for Tag consumption
- OFPT_PACKET_IN - switch notifying reception of Tag'ed packet

FlowTags Components



API

- FT_GENERATE_QRY & FT_GENERATE_RSP – for Tag generation
- FT_CONSUME_QRY & FT_CONSUME_RSP - for Tag consumption
- OFPT_PACKET_IN - switch notifying reception of Tag'ed packet

Modifications to NFs

1) Modify internal functions to generate and consume Tags
2) Shim layer for Tag generation and consumption

FlowTags Contribution

- Propose Tagging for ORIGINBINDINGS and PATHSFOLLOWPOLICY
- Paper details the tagging mechanism
  - Rule generation, policy abstraction, & controller interface
- Open avenues for verification and network diagnosis

FlowTags Contribution

- Propose Tagging for ORIGINBINDINGS and PATHSFOLLOWPOLICY
- Paper details the tagging mechanism
  - Rule generation, policy abstraction, & controller interface
- Open avenues for verification and network diagnosis

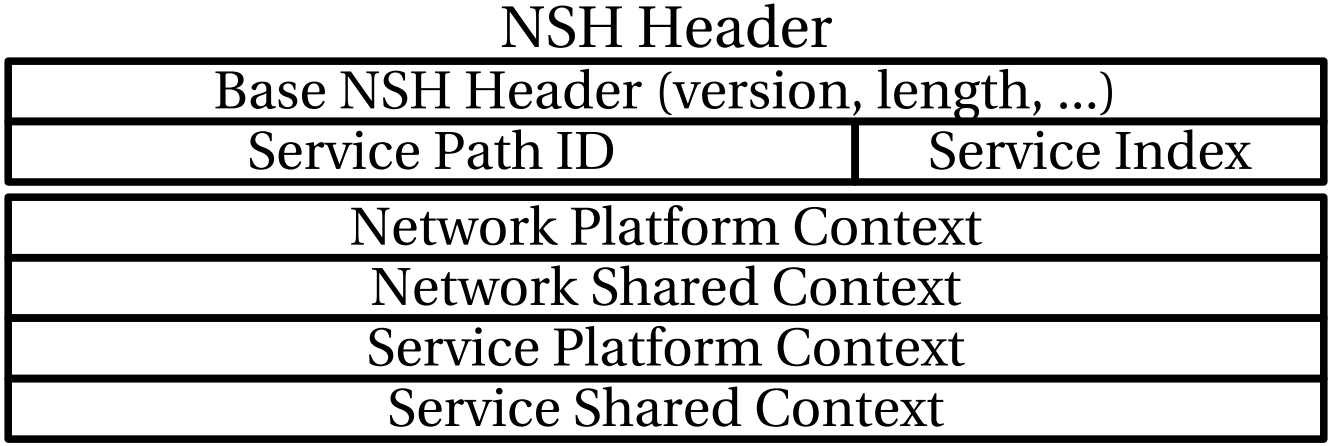
What if additional header(s) are available for Tags?

Outline

- Background
- **SIMPLE**-fying Middlebox Policy Enforcement Using SDN
- Enforcing Network-Wide Policies in the Presence of Dynamic Middlebox Actions using **FlowTags**
- **IETF Protocols**: NSH and others

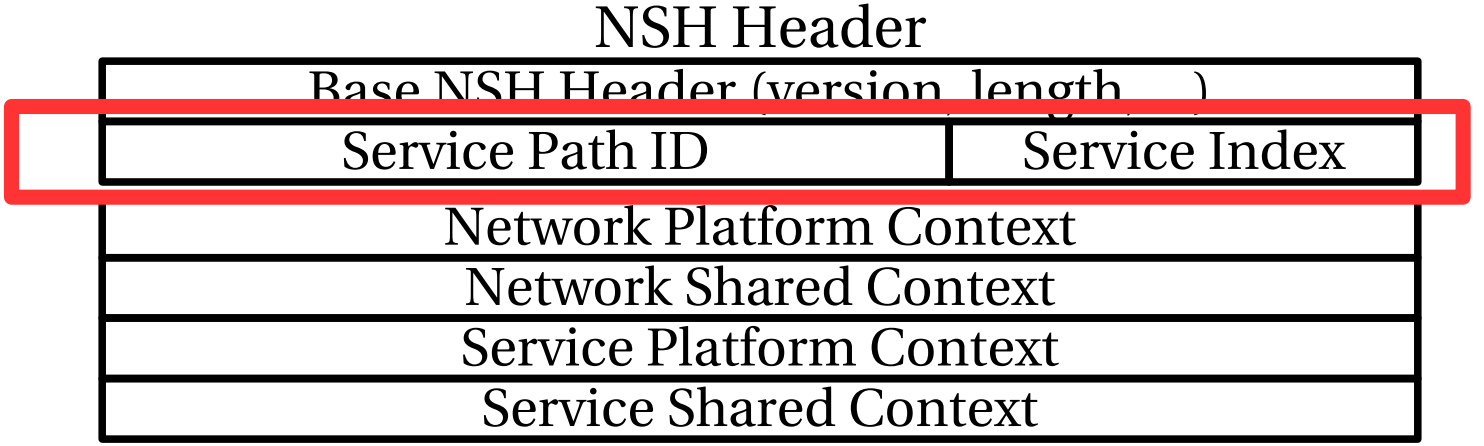
Network Service Header (NSH)

- A dataplane header for carrying information along a service path



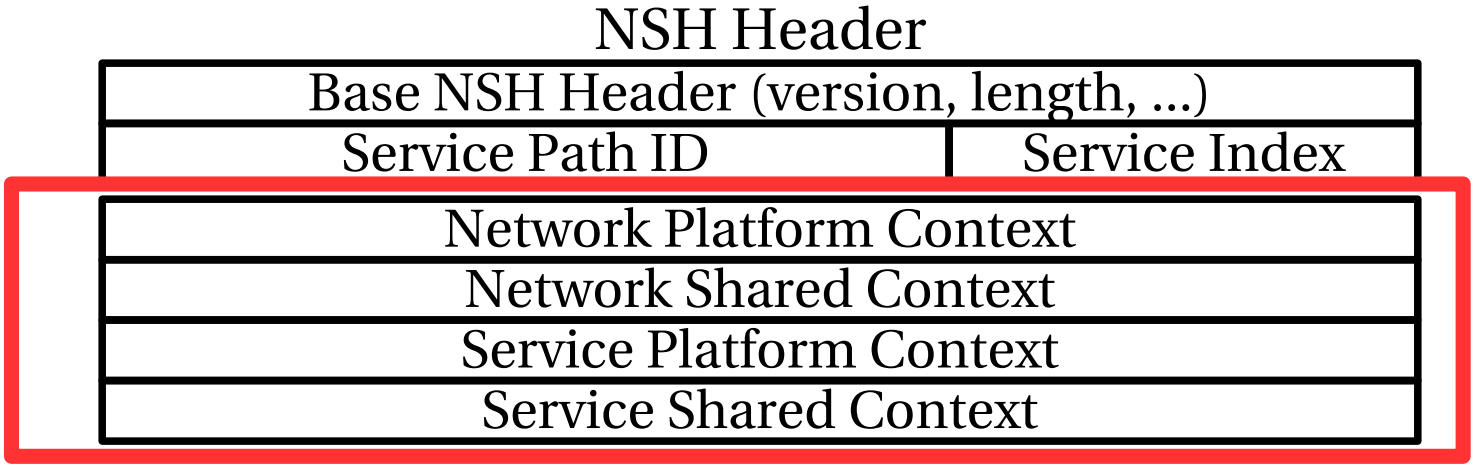
Network Service Header (NSH)

- A dataplane header for carrying information along a service path



Network Service Header (NSH)

- A dataplane header for carrying information along a service path



Summary

- SIMPLE
  - Works with Off-the-shelf NFs
  - Suffers from uncertainty on policy realization
- FlowTags
  - Tagging flows as they traverse NFs
  - NFs need to be modified
  - Existing headers can be used
- IETF proposals like NSH
  - Context information can be shared

Thank You!