SDNs for Enforcement of Dynamic NFV-Policies

Ashwin Rao

04 / 11 / 2015
Background

• Increasing presence of Network Functions (NF)


• Need for dynamic policies

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

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

Policy: Egress → Firewall → IDS → Proxy → Ingress

What must S2 do for a packet from S1?
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

<table>
<thead>
<tr>
<th>Flow</th>
<th>Action</th>
<th>Counter</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
SIMPLE Approach

Traffic Matrix

<table>
<thead>
<tr>
<th>Flow</th>
<th>Action</th>
<th>Counter</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Diagram showing network traffic matrix with flow, action, and counter columns.
SIMPLE Approach

Traffic Matrix  Topology

<table>
<thead>
<tr>
<th>Flow</th>
<th>Action</th>
<th>Counter</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Diagram showing network traffic and topology with traffic matrix and action counter.
SIMPLE Approach

Traffic Matrix  Topology  Policy

Flow  Action  Counter
...  ...  ...

[Diagram showing traffic flow and network topology with traffic matrix and policy rules]
SIMPLE Approach

Traffic Matrix  Topology  Policy  Resource Constraints

Flow  Action  Counter
...  ...  ...

Diagram showing network components and flow.
SIMPLE Approach

Traffic Matrix  Topology  Policy  Resource Constraints

Resource Manager

Flow  Action  Counter
...  ...  ...

Flowchart showing the relationships between Traffic Matrix, Topology, Policy, Resource Constraints, and Resource Manager.
SIMPLE Approach

Traffic Matrix  Topology  Policy  Resource Constraints

Resource Manager

Offline–Online Decomposition
- Offline: Switch constraints
- Online: Load Balancing
- ILP formulation
SIMPLE Approach

Traffic Matrix  Topology  Policy  Resource Constraints

Resource Manager

Rule Manager

Flow  Action  Counter

...  ...  ...

16
SIMPLE Approach

- Traffic Matrix
- Topology
- Policy
- Resource Constraints

Resource Manager

Rule Manager

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

Flow | Action | Counter
---|---|---
... | ... | ...
SIMPLE Approach

Traffic Matrix  Topology  Policy  Resource Constraints

Resource Manager

Rule Manager

Flow | Action | Counter
----|--------|--------
...  | ...    | ...    

18
SIMPLE Approach

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

**Flow** | **Action** | **Counter**
--- | --- | ---
... | ... | ...
SIMPLE Approach

Traffic Matrix | Topology | Policy | Resource Constraints

Resource Manager

Dynamics Manager

Rule Manager

Flow | Action | Counter

... | ... | ...

Flow action counter diagram.
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 $\approx 1$ 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 $\leftrightarrow$ 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

- Host 2
- Proxy
- ACL
- Internet
- DROP
Tag Generation

1) Static Policy Graph (DPG)
   - Traffic ↔ Chain to NFs

2) Dynamic Policy Graph (DPG)
   - IN and OUT Nodes
   - NF Nodes

Host 2 → Proxy → ACL

Internet

DROP
Tag Generation

1) Static Policy Graph (DPG)
   - Traffic $\leftrightarrow$ Chain to NFs

2) Dynamic Policy Graph (DPG)
   - IN and OUT Nodes
   - NF Nodes

Diagram:
- Host 2
- Proxy
- ACL
- Internet
- DROP
Tag Generation

1) Static Policy Graph (DPG)
   - Traffic ↔ Chain to NFs

2) Dynamic Policy Graph (DPG)
   - IN and OUT Nodes
   - NF Nodes

Diagram:
- Host 2
- Proxy
- ACL
- Internet
- DROP

Traffic Flow:
- {H2}, HIT, allow
- {H2}, MISS
- {H2}
- {H2}, HIT
Tag Generation

1) Static Policy Graph (DPG)
   - Traffic ↔ Chain to NFs

2) Dynamic Policy Graph (DPG)
   - IN and OUT Nodes
   - NF Nodes

Host 2 → Proxy → ACL → Internet
{H2} → {H2}, HIT, allow → {H2}, MISS, allow → {H2}, MISS, allow → DROP
Tag Generation

1) Static Policy Graph (DPG)
   - Traffic ↔ Chain to NFs

2) Dynamic Policy Graph (DPG)
   - IN and OUT Nodes
   - NF Nodes

Diagram:
- Host 2
- Proxy
- ACL
- Internet
- DROP

Node labels:
- {H2}
- {H2}, HIT
- {H2}, MISS
- {H2}, *, block
- {H2}, MISS, allow
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

Diagram:
- Host 2
- Proxy
- ACL
- Internet
- DROP

Rules:
- {H2}, HIT, allow
- {H2}, MISS, allow
- {H2}, *, block
FlowTags Components

- FlowTags Controller
- SDN Capable Switch
- FlowTags-capable NF
FlowTags Components

- FlowTags Controller
- SDN Capable Switch
- FlowTags-capable NF

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

SDN Capable Switch  FlowTags Controller  FlowTags-capable NF

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

<table>
<thead>
<tr>
<th>NSH Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base NSH Header (version, length, ...)</td>
</tr>
<tr>
<td>Network Platform Context</td>
</tr>
<tr>
<td>Network Shared Context</td>
</tr>
<tr>
<td>Service Platform Context</td>
</tr>
<tr>
<td>Service Shared Context</td>
</tr>
</tbody>
</table>
Network Service Header (NSH)

- A dataplane header for carrying information along a service path

<table>
<thead>
<tr>
<th>NSH Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base NSH Header (version, length, ...)</td>
</tr>
<tr>
<td>Service Path ID</td>
</tr>
<tr>
<td>Network Platform Context</td>
</tr>
<tr>
<td>Network Shared Context</td>
</tr>
<tr>
<td>Service Platform Context</td>
</tr>
<tr>
<td>Service Shared Context</td>
</tr>
</tbody>
</table>
Network Service Header (NSH)

- A dataplane header for carrying information along a service path

```
<table>
<thead>
<tr>
<th>NSH Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base NSH Header</td>
</tr>
<tr>
<td>(version, length, ...)</td>
</tr>
<tr>
<td>Service Path ID</td>
</tr>
<tr>
<td>Service Index</td>
</tr>
<tr>
<td>Network Platform Context</td>
</tr>
<tr>
<td>Network Shared Context</td>
</tr>
<tr>
<td>Service Platform Context</td>
</tr>
<tr>
<td>Service Shared Context</td>
</tr>
</tbody>
</table>
```
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!