Policy and Resource Orchestration in Software defined Networks

Anduo Wang
adw@temple.edu

Jie Wu
jiewu@temple.edu

Temple University
traditional network management
traditional network management
traditional network management

classic network management

policy & resource management
traditional network management

policy & resource management

Control Software

Control Software

Control Software

Control Software

Control Software
traditional network management
traditional network management
traditional network management

vertically integrated heterogenous hardware and virtual functions

Control Software

data plane

Control Software

data plane

Control Software

data plane

Control Software

data plane

policy & resource management
software-defined networking (SDN)
software-defined networking (SDN)
software-defined networking (SDN)
software-defined networking (SDN)
software-defined networking (SDN)
software-defined networking (SDN)
software-defined networking (SDN)
software-defined networking (SDN)
software-defined networking (SDN)

vertically integrated heterogeneous hardware and virtual functions
orchestrating policies & resources in SDN

- policies & resource management
- controller
- switches & virtual functions
orchestrating policies & resources in SDN

**control policies of disparate nature**

- monitor
- firewall
- routing
- load balancer

**heterogenous devices and virtual functions**

- switch
- proxy
- firewall
- NAT

controller
control policies of disparate nature

monitor | firewall | routing | load balancer

controller

heterogenous devices and virtual functions

switch | proxy | firewall | NAT
Policy Orchestration

today, the onus of coordinating SDN policies falls on the admin to write modular control application

- policy prefixed in specific controller program — syntax varies from one domain specific language to another
- manual composition of controller programs relies on the internalized knowledge of experienced admin

Our Approach

- orchestration as a controller primitive
- policy as semantic units that maintain properties
- automating policy coordination by logical reasoning about network properties
model SDN policies as data \textit{query/update}

\[ \Delta = \text{state}_{\text{new}} - \text{state}_{\text{current}} \]

\textit{query} invariant

\textit{update} \( \Delta \)

\textit{check violation}

\textit{repair}
semantic dependency

policy $x$ depends on $y$ (denoted by $x \rightarrow y$) if
Policy \( x \) depends on \( y \) (denoted by \( x \rightarrow y \)) if \( x \) can violate \( y \) invariant and trigger \( y \) action.
semantic dependency

policy $x$ depends on $y$ (denoted by $x \rightarrow y$) if

$x$ can violate $y$ invariant and trigger $y$ action

but $y$ will never affect $x$
data (ir)relevance reasoning

- update $\Delta_x$ is relevant to query $i_y$ if $\Delta_x \land i_y$ is SAT
- $\Delta_y$ is irrelevant to $i_x$ if $\Delta_y \land i_x$ is UNSAT
running example: SDN policies

\[
\begin{align*}
& \text{clients} & \{ H_1, H_2 \} \\
& \text{FW} & \text{A} & \text{LB} \\
& S_1 & S_2 \text{ servers} \\
\end{align*}
\]
running example: SDN policies

fw, firewall blocks traffic from/to H₂
running example: SDN policies

fw, firewall blocks traffic from/to H2
lb, load balancer directs H1 traffic from/to S
**Running example: SDN policies**

fw, *firewall* blocks traffic from/to H₂  
lb, *load balancer* directs H₁ traffic from/to S

\[  
\begin{align*}  
\text{fw} & \triangleq \text{firewall} \\
\text{lb} & \triangleq \text{load balancer} \\
\end{align*}  
\]

\[  
\text{lb} \triangleq \text{if} (\text{client traffic?}, \text{lb}_1, \text{lb}_2) \]  
where

\[  
\text{lb}_1 \triangleq \text{pick a server from } S_1, S_2  \\
\text{lb}_2 \triangleq \text{restore public server address}  
\]
running example: SDN policies

fw, firewall blocks traffic from/to H₂
lb, load balancer directs H₁ traffic from/to S

lb ≜ if (client traffic?, lb₁, lb₂) where
    lb₁ ≜ pick a server from S₁, S₂
    lb₂ ≜ restore public server address

rt, routing between H₁,₂ and S
semantic layering

construct layering with stratification number

- correctness guarantee: the semantics of every policy will be preserved

stratified dependency graph

synthesized layering

lb₂
fw
lb₁,rt
resource orchestration

control policies of disparate nature

- monitor
- firewall
- routing
- load balancer

controller

placement strategies

heterogenous devices and virtual functions

- switch
- proxy
- firewall
- NAT

Switch-connected Servers
Middlebox

- **Network Function Virtualization (NFV)**
  - Technology of virtualizing network functions into software building blocks

- **Middlebox**: software implementation of network services
  - Improve the network performance:
    - Web proxy and video transcoder, load balancer, ...
  - Enhance the security:
    - Firewall, IDS/IPS, passive network monitor, ...

- **Examples**
  - Web Proxy
  - Firewall
  - NAT
Flows-to-Middlebox Requirement

- Multiple middleboxes may/may not have a serving order
  - Examples
    - Firewall usually before Proxy
    - Virus scanner either before or after NAT gateway

- Categories
  - Non-ordered middlebox set (i.e., independent)
  - Totally-ordered middlebox set (*service chain*)
  - Partially-ordered middlebox set

Middlebox Placement Problems

- **Graph embedding**
  - Middlebox graph, $G_m$, of multiple service chains that needs to be embedded in a given network graph, $G_n$. 

![Diagram](image-url)
Middlebox Placement Problems

- **Graph flow routing**
  - Shortest path or maximum flow between a given source and destination that have to go through a given middlebox in $G_n$.

Middlebox Placement Problem

- **Facility allocation**
  - Optimal placement of facilities (i.e., middlebox) to minimize transportation costs (i.e., traffic, including detour traffic from flows to middleboxes).

- **Cost**
  - Setup cost
  - Communication cost

- **Objective**
  - Minimizing sum of middlebox setup cost and communication cost

Middlebox Placement Problems

- Set covering
  - Minimize the number of middleboxes used to cover all flows.

- Middleboxes may change flow rates in different ways
  - Citrix CloudBridge WAN accelerator: 20% (diminishing)
  - BCH(63,48) encoder: 130% (expanding)

- Objective: minimizing total traffic

Middlebox Placement Examples

- **Independent** middleboxes

- **Dependent** middleboxes (m₂ before m₁)
Flow Placement Examples (cont’d)

- A flow covered by multiple middleboxes

[5] NFV Middlebox Placement with Balanced Set-up Cost and Bandwidth Consumption (ICPP ’18)
### Challenges: NP-completeness

<table>
<thead>
<tr>
<th></th>
<th>✔</th>
<th>*</th>
<th>✔</th>
<th>*</th>
<th>*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node capacity</td>
<td>✔</td>
<td>*</td>
<td>✔</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Edge capacity</td>
<td>✔</td>
<td>✗</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Node placement</td>
<td>*</td>
<td>✔</td>
<td>*</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>constraint</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edge routing</td>
<td>*</td>
<td>*</td>
<td>✔</td>
<td>✔</td>
<td>*</td>
</tr>
<tr>
<td>constraint</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Latency constraint</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>✔</td>
</tr>
</tbody>
</table>

NP-completeness and inapproximability under any objective[6]

**Middlebox graph** $(G_m)$

**Network graph** $(G_n)$

[6] Charting the Complexity Landscape of Virtual Network Embeddings (IFIP '18)
Other Challenges

- Special network graphs
  - Such as trees to make embedding tractable

- Other flow-to-middlebox policy
  - Forbidden to pass through certain middleboxes

- Other scheduling problems
  - Such as classic flow shop