A Programmable Policy Engine to Facilitate Time-efficient Science DMZ Management

Chen Xu, Peilong Li, Yan Luo
University of Massachusetts Lowell
11/12/2017

Acknowledgement: This work was supported by NSF CC* Program (No. 1440737)
Content

- Background of our FLowell Science DMZ Network
- Existing Problems
- Motivations
- Design of our Policy Engine
- Performance Evaluation
- Conclusion
FLowell Science DMZ network refers to ESnet model to accelerate large data transfers from UML to MGHPCC

- OpenFlow switch: direct elephant flow & mice flow separately
- OpenDayLight (ODL): OpenFlow controller
- Big Monitoring Fabric (BMF): provide service chains
- Bro: determine if a flow is an elephant flow
The entire existing workflow requires *human intervention*.  
- Our ITs review every Thursday  
- Need hours to decide and apply

Key requirements to accelerate inefficient user-admin interaction  
- Submit user requests and express user intention in a declarative ("what-to") manner  
- Automatic configuration optimization to reduce processing time & labor work  
- Supervised by admins to control risk
Question 1: How Can We Speedup the Service Delivery Process for an End User’s Request?
• An OpenFlow switch functions as a Layer 2 firewall
• User must request access from admin for a network resource
• The user-admin interaction is inefficient and rely on network admins manual work

Solution
• We propose a policy engine that enables network users and network admins to work in a time-efficient manner
• The user can receive an immediate response to their request, depending on whether the request complies with a predefined set of criteria, i.e. a white list
Question 2: Why Not Use OpenFlow Rules?
- Course-grained
  - Routing conflicts may occur
- Fine-grained
  - Memory resource limitation
  - Lookup time increases

Solution
- A white list is necessary and only necessary flow rules will be installed in the switch later
- We design a module in our policy engine to check flow conflict dynamically
Motivation

- Question 3: How Can We Map an End User’s Request to a Set of Network Rules?
  - An end user typically has no prior knowledge regarding network operation and network hardware configurations
  - Simply submit a network resource request and have the system determine the necessary optimal path

- Solution
  - We provide a set of policy rules to help express the user’s intention
  - We design a policy manager inside a policy engine in order to parse the intentions from users as well as to generate the final set of OpenFlow rules
Design

- **Policy Engine**
  - Web GUI: create & implement policy rules & check results
  - Policy Manager: receive policies, parse the policies, check for conflicts, determine the shortest forwarding paths, generate the necessary OpenFlow rules, and store the rules to the policy repository
  - Policy Repo
    - WL: predefined permitted paths by network admins
    - Flow array: paths can be approved by WL as well as flow info
    - Pending array: paths not allowed by WL
Design

Policy Rule

- Intuitive while expressive

<table>
<thead>
<tr>
<th>Keywords</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Src_IP</td>
<td>IP address of the end host</td>
</tr>
<tr>
<td>Dst_IP</td>
<td>IP address of network resource</td>
</tr>
<tr>
<td>Flow_OP</td>
<td>install, remove</td>
</tr>
</tbody>
</table>

- Use cases:
  - User1 requests to have access to DTN from host1
  - Admin1 requests to have access to perfSONAR2 from perfSONAR1
  - Admin2 requests to remove user1’s request

User1:
- Src_IP: 10.0.0.1,
- Dst_IP: 10.0.0.240,
- Flow_OP: install

Admin1:
- Src_IP: 10.0.0.200,
- Dst_IP: 10.0.0.201,
- Flow_OP: install

Admin2:
- Src_IP: 10.0.0.1,
- Dst_IP: 10.0.0.240,
- Flow_OP: remove
Policy Manager Workflow

Goals:
- Reduce human intervention as much as we can.
  - Full automation for end user
- The only manual step for admin is dealing with pending paths
  - Automatic path configuration calculation
- Automatic conversion from policies to OpenFlow rules.
Scenario 1
- Evaluate the network performance to illustrate the advantages of the Science DMZ infrastructure
  - Compare *latency* & *throughput* on private & public network
Evaluation

- Latency (RTT)
  - Public network w/o Science DMZ: 5.424ms
  - Private w/ Science DMZ: 3.483ms

- Throughput

<table>
<thead>
<tr>
<th>Tool</th>
<th>w/DMZ</th>
<th>File Size</th>
<th>Parallelism</th>
<th>Ave Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>iperf</td>
<td>No</td>
<td>N/A</td>
<td>4</td>
<td>95.6 Mbps</td>
</tr>
<tr>
<td>iperf</td>
<td>Yes</td>
<td>N/A</td>
<td>4</td>
<td>9.40 Gbps</td>
</tr>
<tr>
<td>FDT</td>
<td>No</td>
<td>10 GB</td>
<td>1</td>
<td>91.605 Mbps</td>
</tr>
<tr>
<td>FDT</td>
<td>Yes</td>
<td>10 GB</td>
<td>1</td>
<td>6.681 Gbps</td>
</tr>
<tr>
<td>FDT</td>
<td>No</td>
<td>10 GB</td>
<td>4</td>
<td>91.792 Mbps</td>
</tr>
<tr>
<td>FDT</td>
<td>Yes</td>
<td>10 GB</td>
<td>4</td>
<td>9.191 Gbps</td>
</tr>
</tbody>
</table>
Evaluation

- Scenario 2
  - Evaluate the performance and overhead of the policy manager
    - **Policy manager response time** with different size of white list, Flow array and number of switches
Evaluation

- **Policy Manager Response Time**

  - **Worst Case**
    - 100k entries in WL and Flow array, 7-layer fat-tree topology
    - Time: 619.561 ms

  - **Good Scalability**
    - Immediately respond to a user’s request
Conclusion

- FLowell Science DMZ infrastructure
  - Accelerate large-volume data transfers
    - Reduce latency from 5.4 ms to 3.5 ms
    - Increase throughput from 91.8 Mbps to 9.2 Gbps

- Policy engine on top of the network control plane
  - Enables network users to submit demands for network resources
  - User to administrator interactions are simplified and can be finished in 1 second
  - Enable network admins to manage the data paths within the network
Questions?