SCALING SOFTWARE DEFINED NETWORKS

Chengyu Fan (edited by Lorenzo De Carli)
Network management is driven by policy requirements

Network Policy
“Guests must access Internet via web-proxy”
Option 1: Command-line Interface (CLI)

- Vendor-specific, low-level -> misconfiguration
- Script needs frequent maintenance

(Huawei)

Switch# display interface ethernet 1/0/0.1
Ethernet1/0/0.1 current state : UP
Line protocol current state : UP
Description : HUAWEI, AR Series, Ethernet1/0/0.1 Interface

(Cisco)

Switch# show interfaces ethernet 0
Ethernet0 is up, line protocol is up
Hardware is Lance, address is 0060.3ef1.702b (bia 0060.3ef1.702b)
Internet address is 172.21.102.33/24

........
Option 2: Simple Network Management Protocol (SNMP)

- A framework that provides facilities for managing or monitoring network devices in IP networks
  - Query/Set provided variables in devices

Takeaway: limited capabilities (e.g. cannot forward packet to manager)
Network Management is Challenging

- Current networks don’t have good management approaches
  - CLI: vendor-specific, low-level
  - SNMP: limited capabilities
  - Third party management tools: bind to software vendor

- Today’s management requires
  - Easy configuration, fine-grained control, and react to continually changing network conditions timely

- Traditional approaches cannot satisfy today’s management requirements
What Is the Root Cause for the Complex Network Management?

- Control plane has no good abstractions
  - Okay for a single device, not good for the whole network
  - No modularity and no standard way of defining forwarding state

Diagram:

- **Data Plane**
  - Simple packet processing
  - Decide how to handle packets (routing, isolation, traffic engineering et al.)

- **Control Plane**
  - Control Plane
  - Data Plane

Network diagram showing interconnections between Data Plane and Control Plane components.
A network architecture where network control is decoupled from forwarding and is directly programmable.
Programmable Control Plane

Events from switches
- Topology changes
- Traffic statistics
- Arriving packets

Commands to switches
- (un)install rules
- Query statistics
- Send packets

[1] OpenFlow: enabling innovation in campus networks. (SigComm CCR'08)
Standard SDN Workflow

Network Policy
“Guests must access Interest via web-proxy”

[2] Rethinking Enterprise Network Control, TON'09
Summary of SDN

- Decouple control plane and data plane
  - The software control of the network can evolve independently of the hardware

- Logically centralized network control
  - Network-wide view, more deterministic, more efficient
A SDN Use Case

- Google inter-datacenters WAN (B4) is the first and largest SDN/OpenFlow deployment

- Overprovisioning introduces low link utilization (30-40%)

[3] B4: Experience with a globally-deployed software defined WAN. SigComm’13
SDN drives link utilization to almost 100%

- TE server assign flows to idle network links according to the application priority
- Controllers enforce the assignment and control bursts
Scalability Issues of SDN

- Flow-level granularity places significant stress on switch state size and controller involvement

- A switch can support a limited number of OpenFlow entries
  - OpenFlow rules are stored in TCAM (only support ~1500 entries in HP 5406zl switch)
  - Ethernet forwarding uses hash lookup in standard memory (support ~64k entries)

- Limited computation resources on controller
  - NOX controller can handle 30k flows/s
  - Datacenters with 100 edge-switches may have more than 10M flows/s

Scalability Issues of SDN (cont.)

- **Flow setup delay**
  - Forward the initial packet: 5us on ASIC vs. 5ms if controller involved
  - Unacceptable flow setup delay for high performance networks

- **Frequent statistics-pulling reduces the flow setup rate**
  - Tasks (e.g. traffic engineering) need statistics gathering
  - 275 connections/s (no pull) vs. 50 connections/s (5 requests/s) using HP 5406zl switch

Objectives

- Categorize current approaches used to scale SDN networks
- Identify research topics yet to be addressed
Outline

- Introduction
- Taxonomy of Scaling SDN Approaches
- Summary of Research Area Gaps
- Conclusion
Taxonomy of Scaling SDN Approaches

Scaling SDN

Architecture-based
- Control-plane
  - ElastiCon
- Data-plane
  - CPU as Co-processing Unit
- Hybrid
  - DIFANE

Rule-based
- Rule Deployment
  - One big switch
- Rule Optimization
  - Maple
Scaling Control-plane

- Approaches in this category aim to improve the limited computation resources on control layer
  - Use better hardware, make it distributed, etc.

- Distributed system is better than centralized system
  - Better fault tolerance
  - Better scalability
  - Better load balancing

- How to turn networking problems into distributed system problems?
ElastiCon

- Make central controller a controller pool
- Dynamically adds/removes controllers according to traffic conditions

Load Adaptation Decisions
- Load Balance
- Scale Up
- Scale Down

Actions:
- Migrate switch
- Remove controller
- Add controller

Distributed Controllers

Distributed Data Store

Controller 1  Controller N

[5] ElastiCon: An Elastic Distributed SDN Controller, ANCS’14
Evaluation

Adding controller nodes increases the throughput.

More controllers can make the response time stay in a certain level.

Limitations: it does not address the issue of flow setup delay directly, and introduces another layer of complexity.
Taxonomy of Scaling SDN Approaches

Scaling SDN

Architecture based
- Control-plane
  - ElastiCon
- Data-plane
  - CPU as Co-processing Unit
- Hybrid
  - DIFANE

Rule-based
- Rule Deployment
  - One big switch
- Rule Optimization
  - Maple
Data-plane Extension

- High latency for the initial packet forwarding via controller
  - 5us on ASIC vs. 5ms if controller involved
  - Reduce controller’s involvement can improve the flow setup delay
  - Introduce new functionalities, improve statistic gathering approaches, and increase flow table size, etc.

- Functionality-based
  - Introduce new functionalities in switches (e.g. DevoFlow [4])

- Workflow-based
  - Change the default workflow (e.g. Using CPU as a Traffic Co-processing Unit [6])

[6] Using CPU as a Traffic Co-processing Unit in Commodity Switches, HotSDN’12
Using CPU as a Traffic Co-processing Unit

- Make traffic stay in data plane by installing complete forwarding table in software
- TCAM holds partial forwarding table for the fast flows
  - CPU ranks the flows periodically, and swap flows if necessary

Results
- Synthetic traffic shows 1,743 flows swaps per second on average, below the TCAM capacity on most switches

Limitation: need to change all switches; assumption of high BW between CPU and TCAM may not hold
Taxonomy of Scaling SDN Approaches

Scaling SDN

Architecture based
- Control-plane
  - ElastiCon
- Data-plane
  - CPU as Co-processing Unit
- Hybrid
  - DIFANE

Rule-based
- Rule Deployment
  - One big switch
- Rule Optimization
  - Maple
Controller delegates some switches to handle real-time jobs
- Authority switches with larger TCAM space and processing capability
- Controller generates the rules, and installs them on switches
DIFANE (cont.)

Feedback:
Cache rules

Authority Switch

Redirect

Forward

Hit cached rules and forward

Ingress Switch

Egress Switch

First packet

Following packets
DIFANE (cont.)

- DIFANE outperforms NOX ... 

DIFANE achieves small delay for first packet (0.4ms vs 10ms)

Throughput in DIFANE increases linearly with the number of switches.
### Architecture-based approach: Summary

#### Observations:
- No all-mighty solution
- Possible to combine some approaches

<table>
<thead>
<tr>
<th></th>
<th>ElastiCon</th>
<th>CPU Co-processing</th>
<th>DIFANE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch change</td>
<td>None</td>
<td>All</td>
<td>Authority switches</td>
</tr>
<tr>
<td>Delay</td>
<td>Switch-controller</td>
<td>Local switch</td>
<td>Switch-authority switch</td>
</tr>
<tr>
<td>Statistic gathering</td>
<td>Active flows</td>
<td>Active and inactive flows</td>
<td>N/A</td>
</tr>
<tr>
<td>Reactive/proactive</td>
<td>Reactive</td>
<td>Proactive</td>
<td>Hybrid</td>
</tr>
</tbody>
</table>
Taxonomy of Scaling SDN Approaches

Scaling SDN

Architecture based
- Control-plane
  - ElastiCon
- Data-plane
  - DevoFlow
  - CPU as Co-processing Unit
- Hybrid
  - DIFANE

Rule-based
- Rule Deployment
  - One big switch
- Rule Optimization
  - Maple
Rule Deployment Approaches

- SDN controller naturally has the network wide view

- Controller properly utilizes TCAM space to install more rules in the data plane, reducing controller involvement (e.g. One big switch\textsuperscript{[8]})

\textsuperscript{[8]} Optimizing the “One Big Switch” Abstraction in Software-Defined Networks. CoNEXT’13
One Big Switch

- Controller installs rules along the path while obeys the table size constraints
  - Assign subset of the endpoint rules (e.g. ACL) to a routing path
  - Place part of the rules on each switch along the path according to the table size

![Diagram showing network topology with subnet 1 to subnet 2 through switches S1, S2, S3 with rules C1 = 4, C2 = 3, C3 = 2]
One Big Switch (cont.)

- Synthetic 100-node topologies with real firewall configurations
- More than 60% unwanted traffic are dropped in the first 20% of the path

Limitations
- Introduce unwanted traffic
- Statistic gathering is slow
Taxonomy of Scaling SDN Approaches

Scaling SDN

Architecture based

Control-plane
- ElastiCon

Data-plane
- DevoFlow
- CPU as Co-processing Unit

Hybrid
- DIFANE

Rule-based

Rule Deployment
- One big switch

Rule Optimization
- Maple
Rule Optimization

- Rule optimization approaches aim to produce high quality rules
  - Rules that finish the same task with less number of rules, and/or less number of priority
  - Rules that trigger controller less frequently
    - e.g. rules with proper timeout

- Proactive approach
  - Generate rules without the runtime information (e.g. SwitchReduce [9])

- Reactive approach
  - Generate rules based on the runtime information (e.g. Maple [10])

Maple

- Reactively record reusable rules for each switch
- Optimize in-order traversal algorithm to generate high quality rules
Aggregated Rules Improve the Flow Table Hit Rates

- **Exact match controller introduces high flow table miss rates**

<table>
<thead>
<tr>
<th>Priority</th>
<th>TcpDst</th>
<th>EthSrc</th>
<th>EthDst</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>23</td>
<td>4</td>
<td>6</td>
<td>Port 30</td>
</tr>
<tr>
<td>0</td>
<td>22</td>
<td>4</td>
<td>6</td>
<td>Drop</td>
</tr>
<tr>
<td>0</td>
<td>80</td>
<td>2</td>
<td>7</td>
<td>Drop</td>
</tr>
</tbody>
</table>

- **Maple could aggregate rules based on TT**

**New flow:** TcpDst=80, EthSrc=4, EthDst=6

if (TcpDst == 22) drop;
else {
  if (EthSrc == 2) drop;
  if (EthSrc == 4 && EthDst == 6) forward along the path;
}

**Maple could aggregate rules based on TT**

<table>
<thead>
<tr>
<th>Priority</th>
<th>TcpDst</th>
<th>EthSrc</th>
<th>EthDst</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>22</td>
<td>*</td>
<td>*</td>
<td>Drop</td>
</tr>
<tr>
<td>0</td>
<td>*</td>
<td>4</td>
<td>6</td>
<td>Port 30</td>
</tr>
<tr>
<td>0</td>
<td>*</td>
<td>2</td>
<td>*</td>
<td>Drop</td>
</tr>
</tbody>
</table>
Maple (cont.)

- Maple generates high quality rules
  - Compact switch flow tables (1006 rules for 973 ACL filters)
  - Small number of priorities (9 priorities for 973 ACL filters)

- Maple reduces flow setup time by aggregating rules
## Rule-based approaches: summary

<table>
<thead>
<tr>
<th></th>
<th>One Big Switch</th>
<th>Maple</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduce #rules</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Additional control overhead</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Controller involvement</td>
<td>Rarely</td>
<td>Sometimes</td>
</tr>
<tr>
<td>Statistics gathering</td>
<td>Slow</td>
<td>Alleviated</td>
</tr>
</tbody>
</table>

- **Conclusions:**
  - No design changes on switches, can be done locally by ISPs
  - Hard to solve the statistics gathering issue
## Architecture-based vs. Rule-based

<table>
<thead>
<tr>
<th>Pros</th>
<th>Architecture-based</th>
<th>Rule-based</th>
</tr>
</thead>
<tbody>
<tr>
<td> </td>
<td>Can improve the limited computation resources on control plane</td>
<td>No design change on switches or controller</td>
</tr>
<tr>
<td> </td>
<td>Can make smarter/powerful switches, reduce controller’s involvement</td>
<td>Easier to deploy, can be done locally</td>
</tr>
<tr>
<td> </td>
<td>Possible to reduce statistics gathering overhead greatly</td>
<td></td>
</tr>
<tr>
<td>Cons</td>
<td>Design change is usually expensive, need vendors involved</td>
<td>Limited computation resources issue still exists, may throttle performance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hard to solve the statistics gathering issue</td>
</tr>
</tbody>
</table>

- **Conclusions:**
  - Both have pros and cons, but it is possible to combine them
Outline

- Introduction
- Taxonomy of Scaling SDN Approaches
  - Summary of Research Area Gaps
- Conclusion
Summary of Research Area Gaps

- Distributed controllers need further research
- No reliable measurement methods and benchmark
- No efficient statistics gathering mechanisms
- Need further optimization on TCAM
  - E.g. How to remove inactive rules ASAP?
- Utilize new features to improve scalability, or even propose new protocols
- What is the intelligence for switches?
Outline

- Introduction
- Taxonomy of Scaling SDN Approaches
- Summary of Research Area Gaps
- Conclusion
Conclusion

- SDN is an emerging network architecture with scalability issues
- Rule-based approaches can be implemented and deployed as SDN applications, could be short term solutions
- Scaling control plane approaches do not address flow setup delay directly, need more research
- Data plane extension and hybrid approaches need to change switch design, could be long term solutions
Paper list


