Application-aware
Data Plane Processing in SDN

Hesham Mekky*    Fang Hao    Sarit Mukherjee    Zhi-Li Zhang*    T. V. Lakshman

*University of Minnesota    Bell Labs, Alcatel-Lucent
Motivating Examples

NAT

- IP & port rewrite
- Keep track of available public IP & port

Content based server selection

- Assign server IP based on URL
- Maintain flow state to support TCP splicing

Can we extend the SDN data path to directly support such network services?
Application-aware data plane processing in SDN

Alternatives:

• Using middleboxes
  - More equipments to maintain
  - Extra hops (detour) in data path
  - Complexity of service chaining

• Using SDN controller to implement application processing logic
  - Switch-Controller delay cause slowdown in data path
  - Control plane not designed to handle every packet -> throughput bottleneck

Application-aware data plane processing: One system for both routing and network services with uniform central control and scale-out data plane
Open vSwitch

Design Choice: Where to intercept the packet and implement application processing logic?

Option 1: connection manager
- Pros: modular design
- Cons: redundant coding, slow
  - Unnecessary encap/decap
  - Redundant flow table

Option 2: kernel flow table
- Pros: best performance
- Cons: hard to implement

Option 3: user space flow table
- Good tradeoff between 1 & 2: easy implementation & reasonable performance
Application-aware Data Plane for Open vSwitch
Example: Firewall & Load Balancer

Required Policy:

Web traffic to server x

#TCP conn to each server < 1000

Send to server s1 or s2 by using hash(src_ip)

App table rule: dst_ip=x, tcp, dport=80: fw, lb, fwd, install
Firewall & Load Balancer: Implementation

Standard user space flow table

Kernel flow table

App table

<table>
<thead>
<tr>
<th>Flow</th>
<th>Action List</th>
</tr>
</thead>
<tbody>
<tr>
<td>dst_ip=x, tcp, dport=80</td>
<td>fw, lb, fwd, install</td>
</tr>
</tbody>
</table>

HotSDN 2014

PACKET
src_ip=a, sport=6000, tcp, dst_ip=x, dport=80

nFlow = 998
nFlow = 999

Hash(a) = s1

break = false (continue)

src_ip=a, sport=6000, tcp, dst_ip=x, dport=80:
set dst_ip=s1, out port1

Scratch pad
Example 2: Content Aware Server Selection

- Return traffic does not have to go through front-end vSwitch
- Both front-end & back-end vSwitches can be scaled out independently
Experimentation: TCP splicing

- TCP sequence number rewriting implemented in kernel space
- Very good performance: same as native OVS kernel actions
Experimentation: Server Selection

- Server selection action implemented in user space
- Performance can be significantly improved by using the new multi-threaded OVS daemon implementation (on-going work)
Conclusions and Future Work

• A first step towards enabling application-aware SDN data plane
• Overall, extending software switches like Open vSwitch to support network service applications is a promising direction
• Kernel to user space copying is still a performance barrier
  - Careful design choices to trade-off between ease of implementation and performance
  - Current design doesn’t suite applications that require processing *every packet*
• Future work
  - Take advantage of multi-threaded OVS daemon in recent versions
  - Make the apps “pluggable”
  - DPDK
Design Details: App Table, App Actions, App Chaining

- App actions implemented as *vendor actions*, only run in user space
  - Determine actions for current packet
  - Modify local app state
  - Generate/modify rule set to be installed in standard flow table
  - Remove flows from standard flow table
  - Generate packet out
  - Send Packet_In, Flow_Removed, or app update vendor msg to controller
    ... by calling Openflow API module