Can the Production Network Be the Testbed?

Rob Sherwood
Deutsche Telekom Inc.
R&D Lab

Glen Gibb, KK Yap, Guido Appenzeller, Martin Cassado, Nick McKeown, Guru Parulkar

Stanford University, Big Switch Networks, Nicira Networks
Problem:

Realistically evaluating new network services is hard

• services that require changes to switches and routers
  • e.g.,
    o routing protocols
    o traffic monitoring services
    o IP mobility

Result: Many good ideas don't get deployed;
Many deployed services still have bugs.
Why is Evaluation Hard?

Real Networks

Testbeds
Not a New Problem

• Build open, programmable network hardware
  ◦ NetFPGA, network processors
  ◦ **but**: deployment is expensive, fan-out is small

• Build bigger software testbeds
  ◦ VINI/PlanetLab, Emulab
  ◦ **but**: performance is slower, realistic topologies?

• Convince users to try experimental services
  ◦ personal incentive, SatelliteLab
  ◦ **but**: getting *lots* of users is hard
Solution Overview: Network Slicing

- Divide the production network into logical *slices*
  - each slice/service controls its own packet forwarding
  - users pick which slice controls their traffic: opt-in
  - existing production services run in their own slice
    - e.g., Spanning tree, OSPF/BGP

- Enforce **strong isolation** between slices
  - actions in one slice do not affect another

- Allows the (logical) testbed to **mirror** the production network
  - real hardware, performance, topologies, scale, users
Rest of Talk...

- How network slicing works: FlowSpace, Opt-In
- Our prototype implementation: FlowVisor
- Isolation and performance results
- Current deployments: 8+ campuses, 2+ ISPs
- Future directions and conclusion
Current Network Devices

Switch/Router

Control Plane
- Computes forwarding rules
  - “128.8.128/16 --> port 6”
- Pushes rules down to data plane
- Enforcement of forwarding rules
- Exceptions pushed back to control plane
  - e.g., unmatched packets

Data Plane

Rules

Control/Data Protocol

Excepts

General-purpose CPU

Custom ASIC
Add a Slicing Layer Between Planes

Slice 2 Control Plane

Slice 1 Control Plane

Slice 3 Control Plane

Slicing Layer

Rules

Control/Data Protocol

Data Plane

Slice Policies

Excepts
Network Slicing Architecture

A network slice is a collection of sliced switches/routers

- Data plane is unmodified
  - Packets forwarded with **no performance penalty**
  - Slicing with existing ASIC

- **Transparent** slicing layer
  - each slice believes it owns the data path
  - enforces isolation between slices
    - i.e., rewrites, drops rules to adhere to slice police
  - forwards exceptions to correct slice(s)
Slicing Policies

The policy specifies resource limits for each slice:

– Link bandwidth
– Maximum number of forwarding rules
– Topology
– Fraction of switch/router CPU

– FlowSpace: which packets does the slice control?
FlowSpace: Maps Packets to Slices
Real User Traffic: Opt-In

• Allow users to Opt-In to services in real-time
  o Users can delegate control of individual flows to Slices
  o Add new FlowSpace to each slice's policy

• Example:
  o "Slice 1 will handle my HTTP traffic"
  o "Slice 2 will handle my VoIP traffic"
  o "Slice 3 will handle everything else"

• Creates incentives for building high-quality services
Rest of Talk...

• How network slicing works: FlowSpace, Opt-In
• Our prototype implementation: FlowVisor
• Isolation and performance results
• Current deployments: 8+ campuses, 2+ ISPs
• Future directions and conclusion
Implemented on OpenFlow

- API for controlling packet forwarding
- **Abstraction** of control plane/data plane protocol
- Works on commodity hardware
  - via firmware upgrade
  - [www.openflow.org](http://www.openflow.org)
FlowVisor Implemented on OpenFlow
FlowVisor Message Handling

Policy Check: Is this rule allowed?

Policy Check: Who controls this packet?

Full Line Rate Forwarding

OpenFlow Firmware

Packet

Data Path

Exception
FlowVisor Implementation

- Custom handlers for each of OpenFlow's 20 message types
  - Transparent OpenFlow proxy
  - 8261 LOC in C
  - New version with extra API for GENI

- Could extend to non-OpenFlow (ForCES?)

- Code: `git clone git://openflow.org/flowvisor.git`
Rest of Talk...

- How network slicing works: FlowSpace, Opt-In
- Our prototype implementation: FlowVisor
- Isolation and performance results
- Current deployments: 8+ campuses, 2+ ISPs
- Future directions and conclusion
Isolation Techniques

Isolation is critical for slicing

In talk:
• Device CPU

In paper:
• FlowSpace
• Link bandwidth
• Topology
• Forwarding rules

As well as performance and scaling numbers
Device CPU Isolation

- Ensure that no slice monopolizes Device CPU

- CPU exhaustion
  - prevent rule updates
  - drop LLDPs ---> Causes link flapping

- Techniques
  - Limiting rule insertion rate
  - Use periodic drop-rules to throttle exceptions
  - Proper rate-limiting coming in OpenFlow 1.1
CPU Isolation: Malicious Slice
Rest of Talk...

• How network slicing works: FlowSpace, Opt-In
• Our prototype implementation: FlowVisor
• Isolation and performance results

Current deployments: 8+ campuses, 2+ ISPs

• Future directions and conclusion
FlowVisor Deployment: Stanford

- Our real, production network
  - 15 switches, 35 APs
  - 25+ users
  - 1+ year of use
  - my personal email and web-traffic!

- Same physical network hosts Stanford demos
  - 7 different demos
FlowVisor Deployments: GENI

- Washington
- Stanford
- Wisconsin
- Indiana
- Princeton
- Rutgers
- Clemson
- NLR
- Internet2
- GATech
Future Directions

- Currently limited to subsets of actual topology
  - Add virtual links, nodes support

- Adaptive CPU isolation
  - Change rate-limits dynamically with load
  - ... message type

- More deployments, experience
Conclusion: Tentative Yes!

- Network slicing can help perform more realistic evaluations

- FlowVisor allows experiments to run concurrently but safely on the production network
  - CPU isolation needs OpenFlow 1.1 feature

- Over one year of deployment experience

- FlowVisor+GENI coming to a campus near you!

Questions?

git://openflow.org/flowvisor.git
Backup Slides
What about VLANs?

- Can't program packet forwarding
  - Stuck with learning switch and spanning tree
- OpenFlow per VLAN?
  - No obvious opt-in mechanism:
    - Who maps a packet to a vlan? By port?
  - Resource isolation more problematic
    - CPU Isolation problems in existing VLANs
FlowSpace Isolation

<table>
<thead>
<tr>
<th>Policy</th>
<th>Desired Rule</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP</td>
<td>ALL</td>
<td>HTTP-only</td>
</tr>
<tr>
<td>HTTP</td>
<td>VoIP</td>
<td>Drop</td>
</tr>
</tbody>
</table>

- Discontinuous FlowSpace:
  - (HTTP or VoIP) & ALL == two rules

- Isolation by rule priority is hard
  - longest-prefix-match-like ordering issues
  - need to be careful about preserving rule ordering
Scaling

<table>
<thead>
<tr>
<th>Number of slices</th>
<th>% CPU utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Synthetic:</td>
</tr>
<tr>
<td></td>
<td>1000 rules/slice</td>
</tr>
<tr>
<td></td>
<td>10 switches</td>
</tr>
<tr>
<td></td>
<td>100 new flows/switch/s</td>
</tr>
<tr>
<td></td>
<td>Observed:</td>
</tr>
<tr>
<td></td>
<td>37 rules/slice</td>
</tr>
<tr>
<td></td>
<td>28 switches</td>
</tr>
<tr>
<td></td>
<td>1.55 new flows/switch/s</td>
</tr>
</tbody>
</table>

---

*RSS is the region of interest for the system.*
Performance

Avg overhead: 0.483ms

Cumulative Probability vs. OpenFlow Port Status Latency (ms)

without FlowVisor
with FlowVisor