Language Abstractions for Software-defined Networks

Nate Foster
Alec Story
Mark Reitblatt

Rob Harrison

Michael Freedman
Christopher Monsanto
Jennifer Rexford
David Walker

frenetic
**Software-Defined Networks**

**Key ingredients**
- Logically-centralized control
- Standard interfaces

**Enabling technology**
- Novel functionality
- Better reliability
Software-Defined Networks

The Good
- Logically-centralized architecture
- Direct control over packet processing

The Bad
- Low-level programming interface
- Functionality tied to hardware

The Ugly
- Two-tiered programming model
- Weak consistency model

Images by Billy Perkins
Language-Based Abstractions

We believe that language-based abstractions are crucial for achieving the vision of software-defined networking.

Benefits
- Modularity
- Portability
- Efficiency
- Assurance
Challenge: Network Updates

Slogan: configuration = function(view)
- Controller maintains global view
- Application applies a function to this view to obtain configuration

Problems
- Want to propagate updates atomically
- But can only change one switch at a time
Example: Distributed ACL

Configuration A
- Process black-hat traffic on F1
- Process white-hat traffic on {F2,F3}

Configuration B
- Process black-hat traffic on {F1,F2}
- Process white-hat traffic on F3

Security Policy

<table>
<thead>
<tr>
<th>Src</th>
<th>Traffic</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web</td>
<td>Allow</td>
<td></td>
</tr>
<tr>
<td>Non-web</td>
<td>Drop</td>
<td></td>
</tr>
<tr>
<td>Any</td>
<td>Allow</td>
<td></td>
</tr>
</tbody>
</table>
Network Update Abstractions

Give programmers a collection of functions $\text{update}(\text{config, topo})$

Semantics of $\text{update}$ guarantees reasonable behavior

**Per-Packet Consistency**

Every packet processed by the old policy or the new policy, but not a mixture of the two

**Per-Flow Consistency**

All packets in the same flow processed by the old policy or the new policy, but not a mixture of the two
The Abstractions at Work

### Configuration A

`I_configA = [...]`

### Configuration B

```python
I_configB = [Rule({IN_PORT:1},forward(5)),
             Rule({IN_PORT:2},forward(6)),
             Rule({IN_PORT:3},forward(7)),
             Rule({IN_PORT:4},forward(7))]

F1_configB = [Rule({TP_DST:80}, [forward(2)]),
              Rule({TP_DST:22}, [])]

F2_configB = [Rule({TP_DST:80}, [forward(2)]),
              Rule({TP_DST:22}, [])]

F3_configB = [Rule({}, [forward(2)])]

cfgB = {I: SwitchConfiguration(I_configB),
        F1: SwitchConfiguration(F1_configB),
        F2: SwitchConfiguration(F2_configB),
        F3: SwitchConfiguration(F3_configB)}
```

### Main Function

```python
topo = NXTopo(...)  
per_packet_update(configA, topo)  
...wait for traffic load to shift...  
per_packet_update(configB, topo)
```
Update Mechanisms

Two-phase commit
- Install versioned configuration
- Enable at perimeter

Extension
- Update strictly adds paths

Retraction
- Update strictly removes paths

Path modification
- Update modifies a small number of paths

Topological restriction
- Update only affects a few switches
**Trace:** sequence of link-packet pairs

**Property:** prefix-closed set of traces
- Loop freedom
- Blackhole freedom
- Basic connectivity
- Access control
- Waypointing

---

**Theorem (Universal Preservation)**

If $c_1$ and $c_2$ both satisfy $P$ and $u$ is a per-packet consistent update from $c_1$ to $c_2$, then $u$ also satisfies $P$. 

---

**Diagram:**
- Kripke Structure
- Property
- Model Checker
- ✓ (Correctness)
- ✗ (InCorrectness)
Other Abstractions

**Network Queries** [PRESTO 2010, ICFP 2011]
- Declarative language for reading network state
- Decouples monitoring from forwarding
- Enables modular composition of programs

**Network Policy** [POPL 2012]
- Expressive configuration language
  - Full set-theoretic operators
  - Arbitrary black-box functions
- Compiler generates efficient switch-level rules
## A Closing Analogy

<table>
<thead>
<tr>
<th>Concern</th>
<th>Assembly Languages</th>
<th>Programming Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Languages</strong></td>
<td><strong>Languages</strong></td>
</tr>
<tr>
<td></td>
<td><strong>x86</strong></td>
<td><strong>NOX</strong></td>
</tr>
<tr>
<td>Resource Allocation</td>
<td>Move values to/from register</td>
<td>Manipulate forwarding rules</td>
</tr>
<tr>
<td>Resource Tracking</td>
<td>Have I spilled that register?</td>
<td>Will that packet arrive at the controller?</td>
</tr>
<tr>
<td>Coordination</td>
<td>Unregulated calling conventions</td>
<td>Unregulated rule management</td>
</tr>
<tr>
<td>Portability</td>
<td>Hardware dependent</td>
<td>Hardware dependent</td>
</tr>
<tr>
<td></td>
<td>Hardware independent</td>
<td>Hardware independent</td>
</tr>
</tbody>
</table>

**Hardware dependent**

**Hardware independent**

**Unregulated calling conventions**

**Unregulated rule management**

**Function calls managed automatically**

**Policies managed automatically**

**Queries can read every packet**

**Declare/use variables**

**Declare/install policy**
Thank You!

Collaborators
Mike Freedman
Rob Harrison
Chris Monsanto
Mark Reitblatt
Emin Gün Sirer
Cole Schlesinger
Shrutarshi Basu
Alec Story
Jen Rexford
Dave Walker

http://frenetic-lang.org