Language Abstractions for Software-defined Networks

Nate Foster
Alec Story
Mark Reitblatt

Rob Harrison

Michael Freedman
Christopher Monsanto
Jennifer Rexford
David Walker

frenetic >>

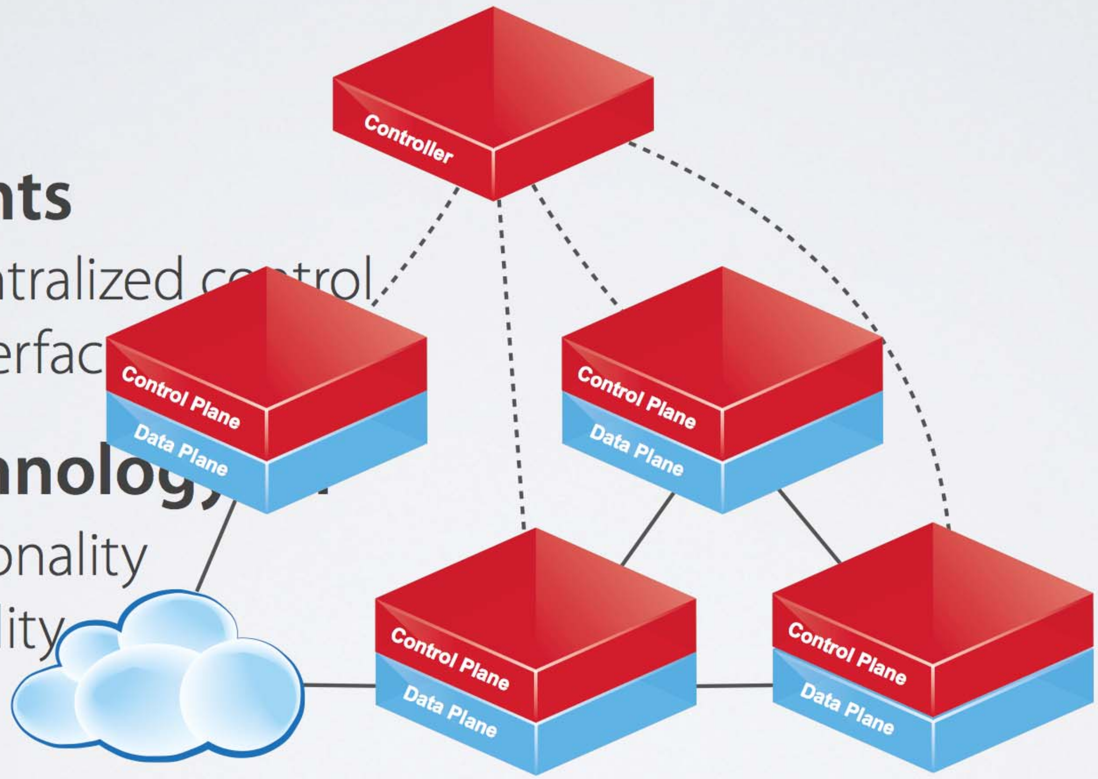
Software-Defined Networks

Key ingredients

- Logically-centralized control
- Standard interfac

Enabling technology

- Novel functionality
- Better reliability



Software-Defined Networks

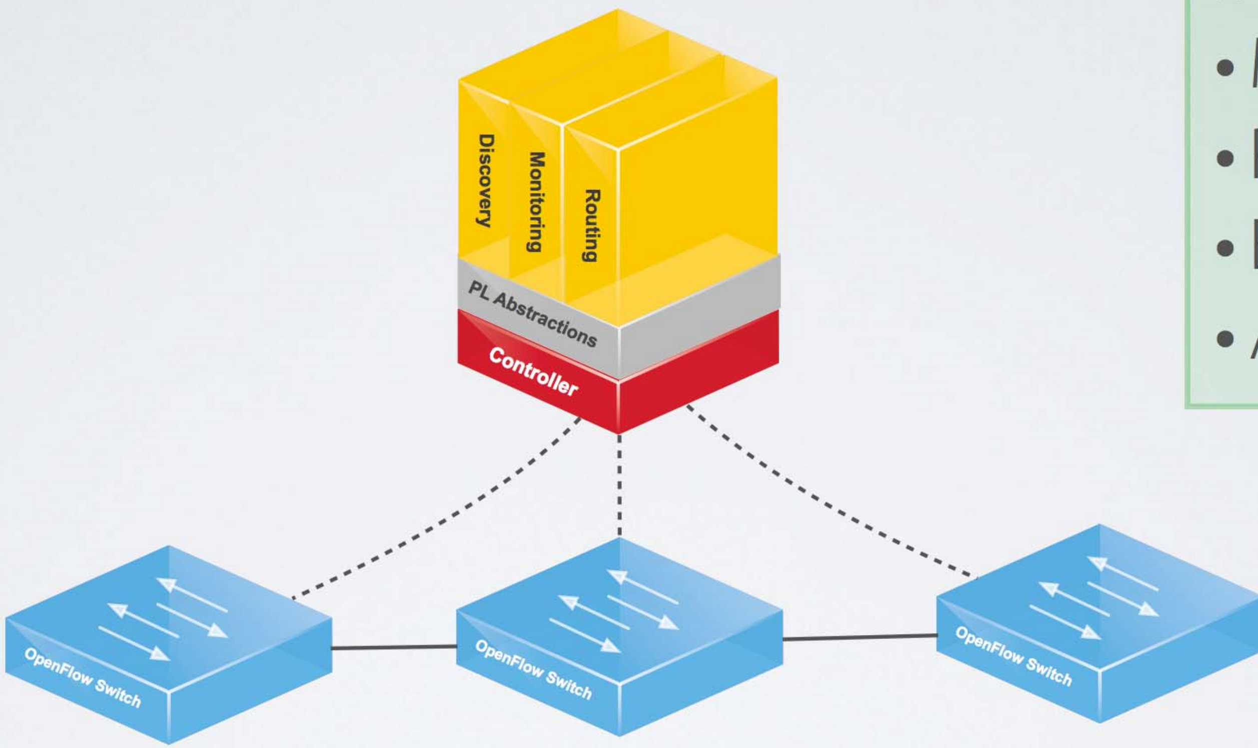
The Good
- Logically-centralized architecture
- Direct control over packet processing

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

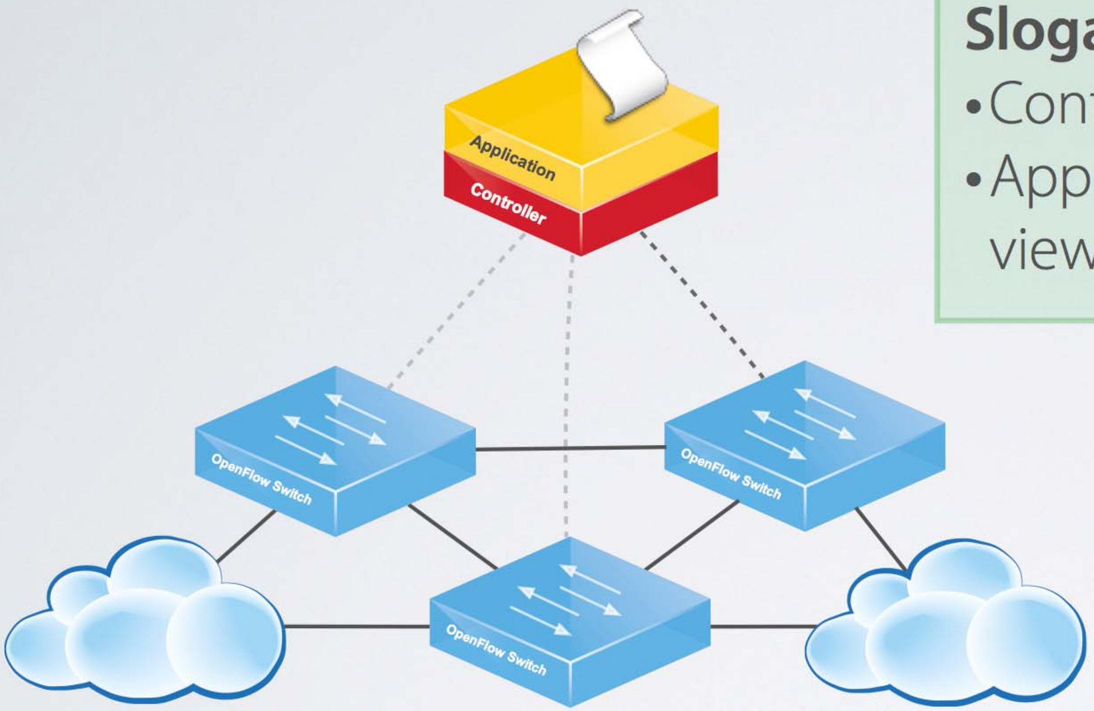


Benefits

- Modularity
- Portability
- Efficiency
- Assurance

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

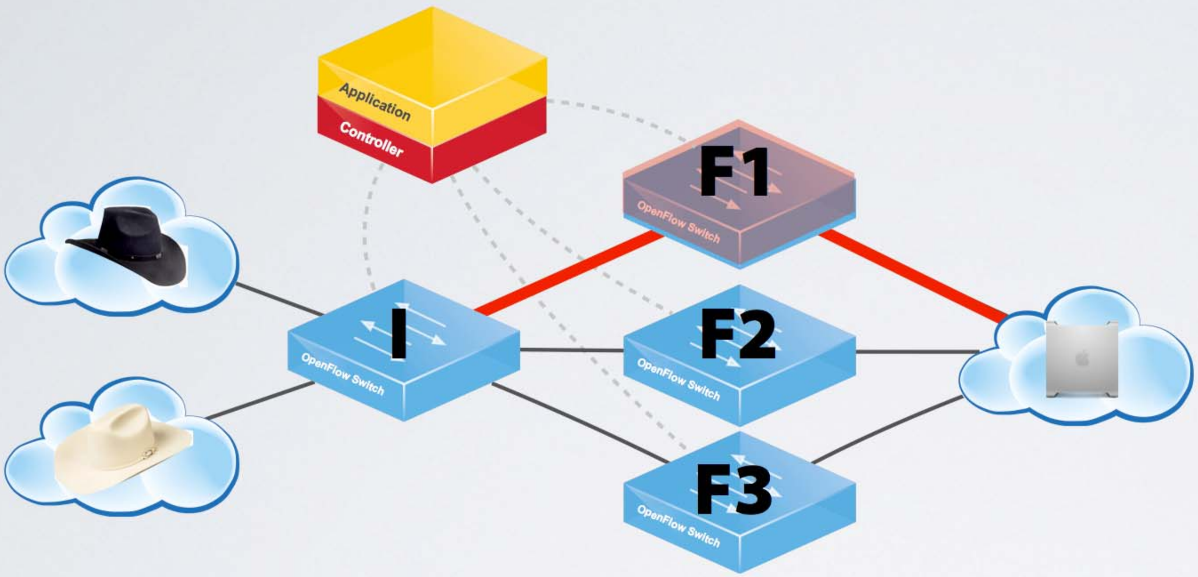
Challenge: Network Updates



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



Security Policy

Src	Traffic	Action
	Web	Allow
	Non-web	Drop
	Any	Allow

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

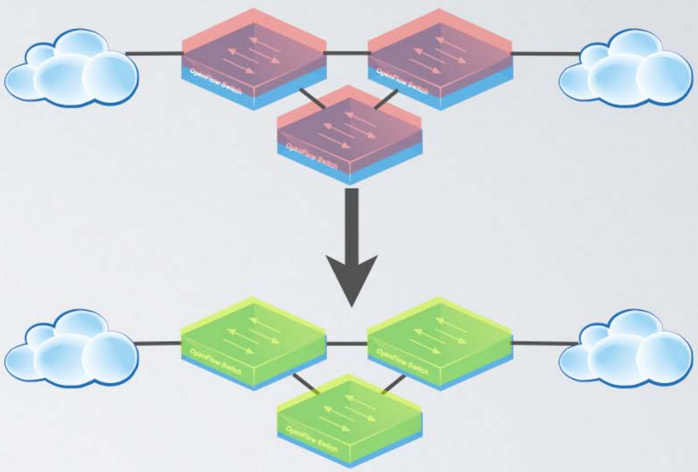
? →

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

Network Update Abstractions [HotNets '11]

Give programmers a collection of functions **`update(config,topo)`**

Semantics of **`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

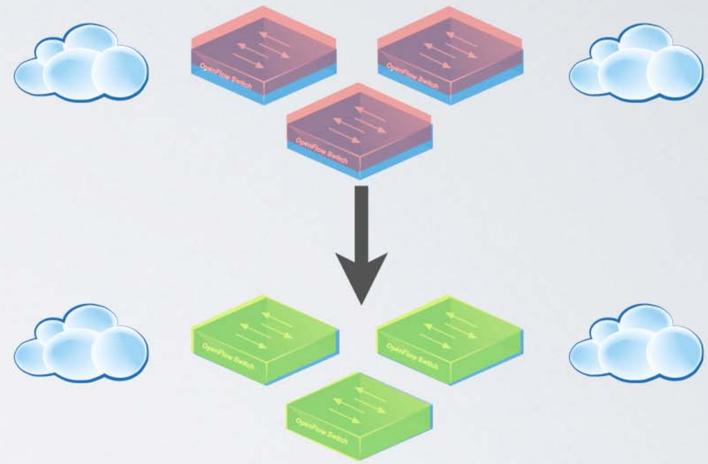
Per-Flow Consistency

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_c
F1_
F2_
F3_
cor
```

```
# Configuration B
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)])]
configB = {I:SwitchConfiguration(I_configB),
           F1:SwitchConfiguration(F1_configB),
           F2:SwitchConfiguration(F2_configB),
           F3:SwitchConfiguration(F3_configB)}
```



```
# Main Function
topo = NXTopo(...)
per_packet_update(configA, topo)
...wait for traffic load to shift...
per_packet_update(configB, topo)
```

Other Examples

- Point-to-point routing
- Multicast routing
- Application-level load balancer

Update Mechanisms

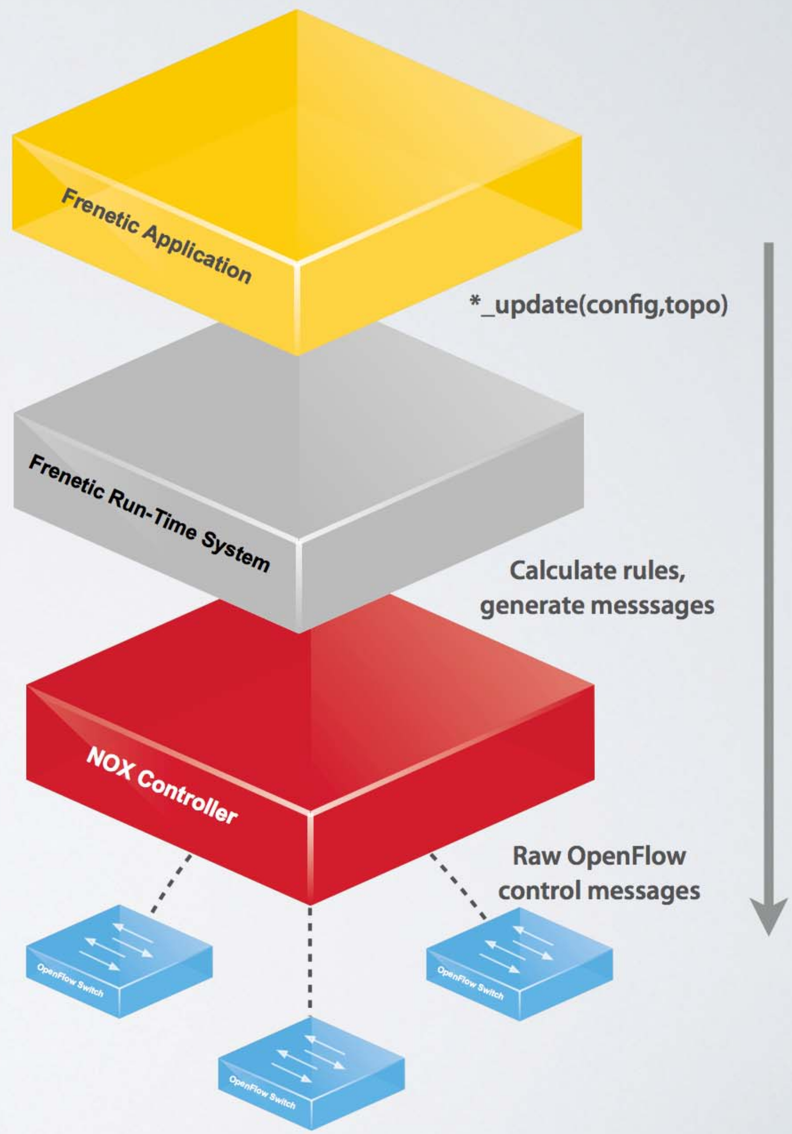
Two-phase commit
- Install versioned configuration
- Enable at perimeter

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

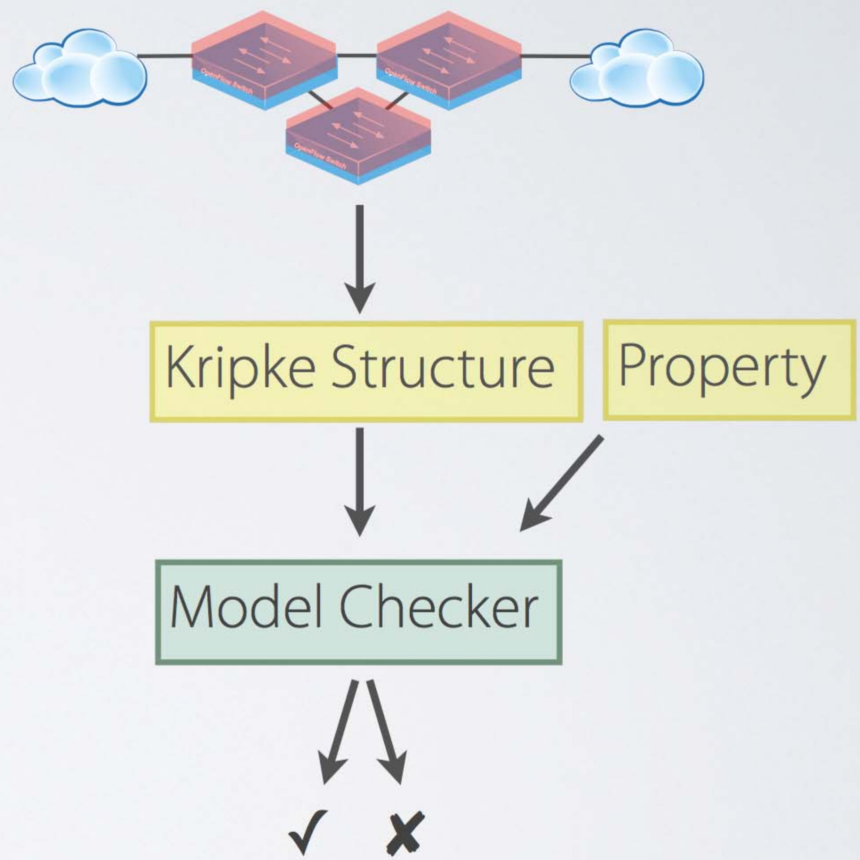


Formal Properties

Trace: sequence of link-packet pairs

Property: prefix-closed set of traces

- Loop freedom
- Blackhole freedom
- Basic connectivity
- Access control
- Waypointing



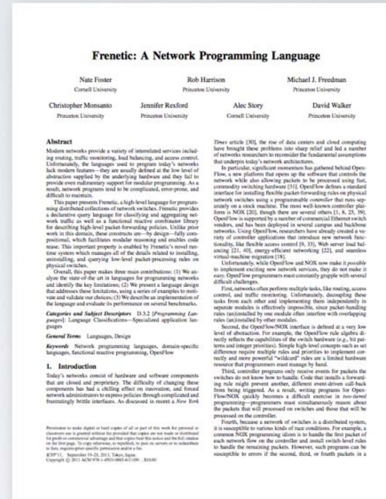
Theorem (Universal Preservation)

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.

Other Abstractions

Network Queries [PRESTO 2010, ICFP 2011]

- Declarative language for reading network state
- Decouples monitoring from forwarding
- Enables modular composition of programs

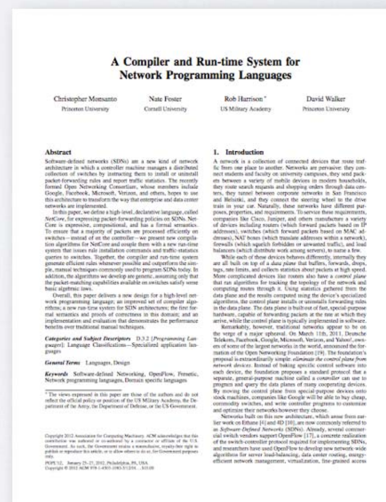


Frenetic: A Network Programming Language

Nate Foster, Cornell University
Rob Harrison, Princeton University
Michael J. Freedman, Princeton University
Christopher Monsanto, Princeton University
Jennifer Rexford, Princeton University
Alec Story, Cornell University
David Walker, Princeton University

Network Policy [POPL 2012]

- Expressive configuration language
  - Full set-theoretic operators
  - Arbitrary black-box functions
- Compiler generates efficient switch-level rules



A Compiler and Run-time System for Network Programming Languages

Christopher Monsanto, Princeton University
Nate Foster, Cornell University
Rob Harrison*, US Military Academy
David Walker, Princeton University

A Closing Analogy

Concern	Assembly Languages		Programming Languages	
	x86	**NOX**	**ML/Haskell**	**Frenetic**
Resource Allocation	Move values to/from register	Manipulate forwarding rules	Declare/use variables	Declare/install policy
Resource Tracking	Have I spilled that register?	Will that packet arrive at the controller?	Variables always available	Queries can read every packet
Coordination	Unregulated calling conventions	Unregulated rule management	Function calls managed automatically	Policies managed automatically
Portability	Hardware dependent	Hardware dependent	Hardware independent	Hardware Independent

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