SDN Control Abstractions

Brighten Godfrey

CS 538 March 8, 2017

Outline

Frenetic

Network updates

Beyond the research

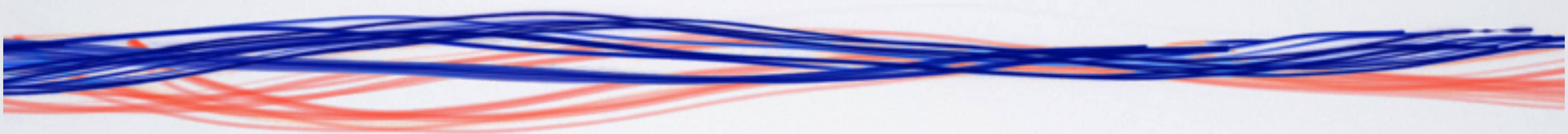
Network Updates

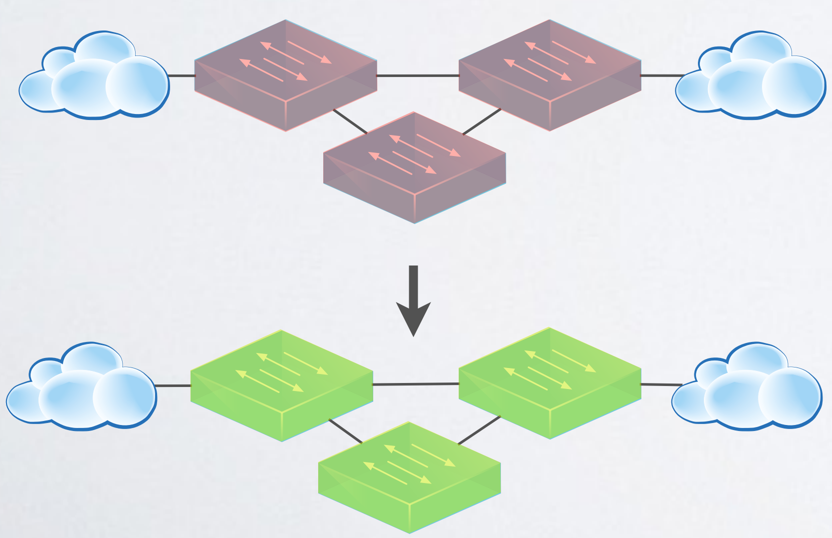
Slides Courtesy
Nate Foster!

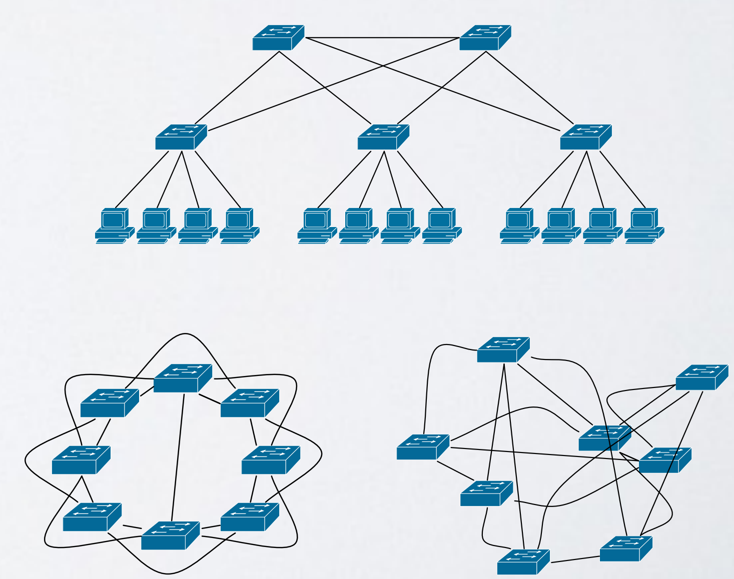
Abstractions for Network Update

Nate Foster
Mark Reitblatt

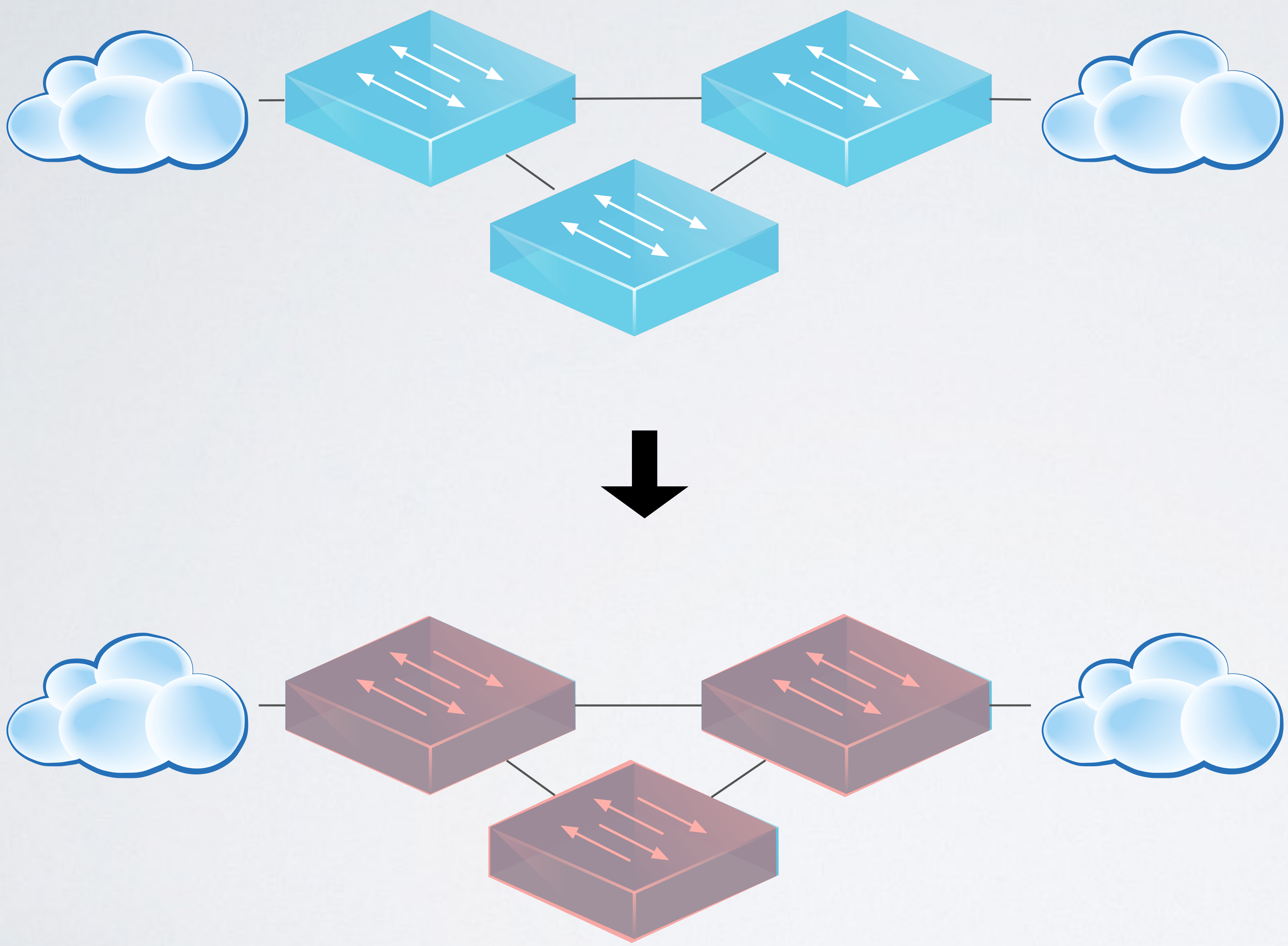
Jen Rexford
Cole Schlesinger
Dave Walker







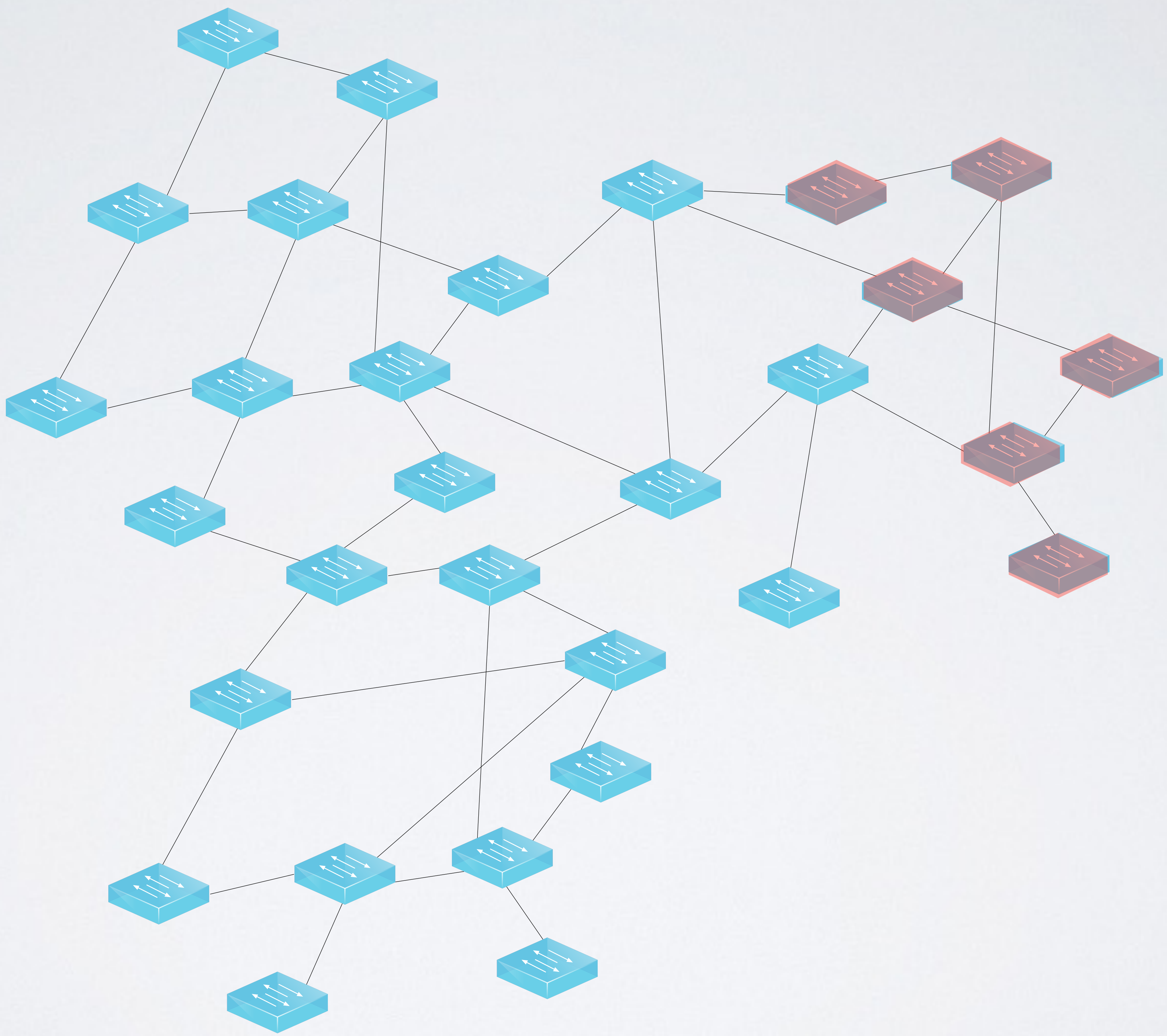
Updates Happen



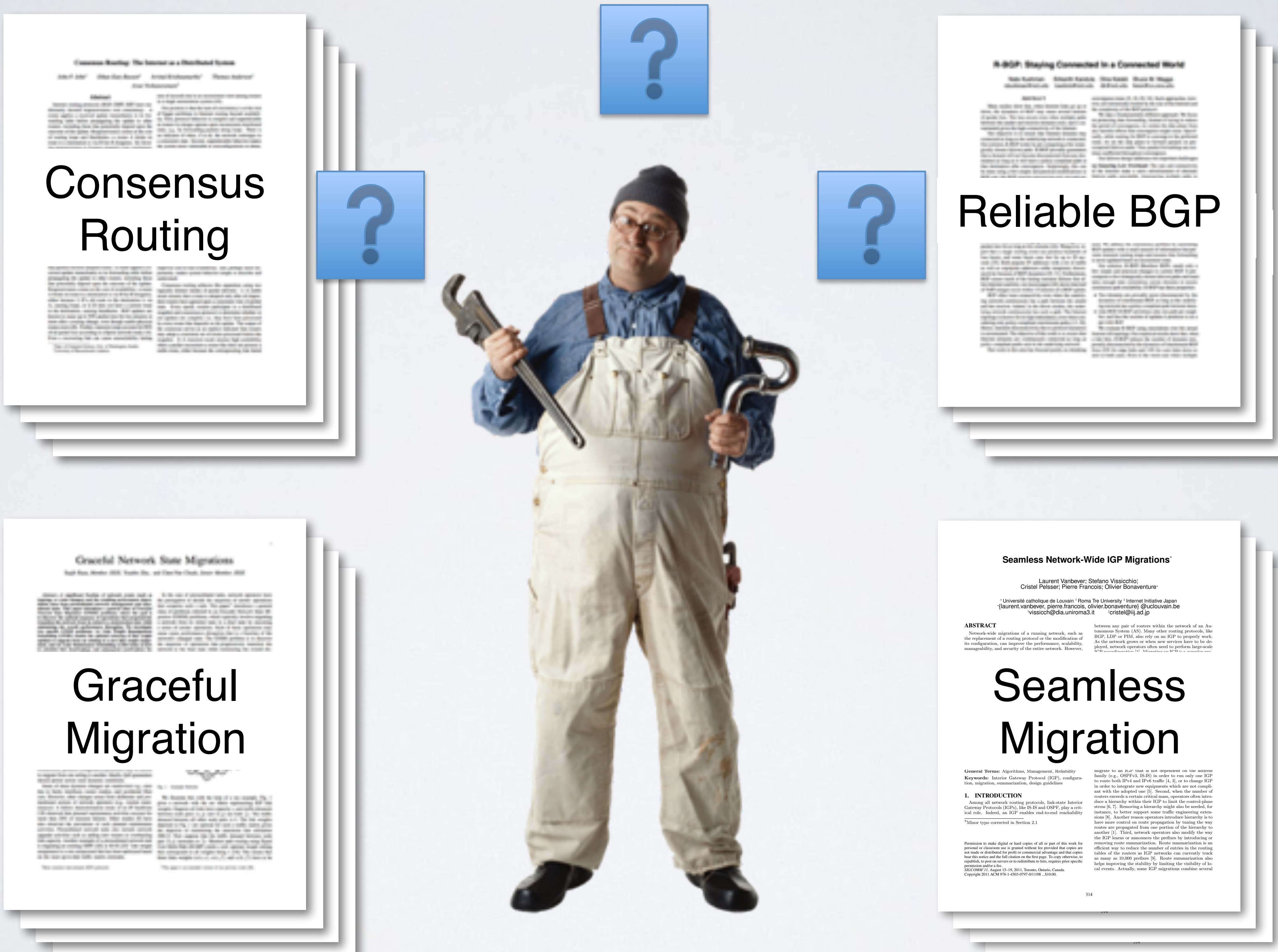
Network Updates
- Maintenance
- Failures
- ACL Updates

Desired Invariants
- No black-holes
- No loops
- No security violations

Network Updates Are Hard



Prior Work


Consensus Routing
Reliable BGP
Graceful Migration
Seamless Migration
Seamless Network-Wide IGP Migrations

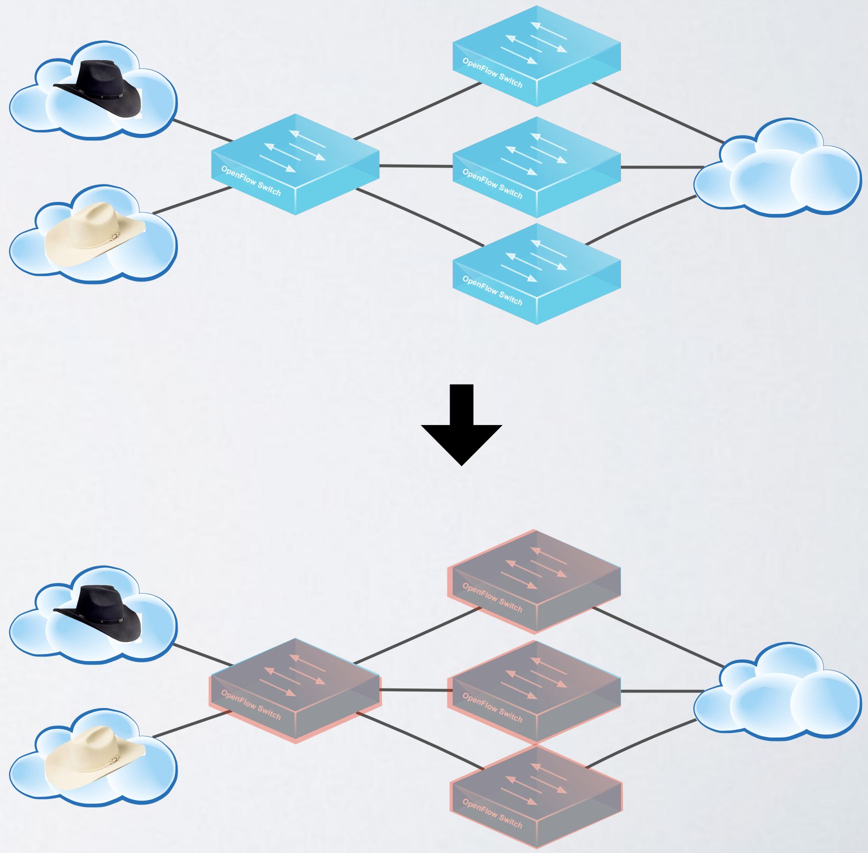
Network Update Abstractions

Goal

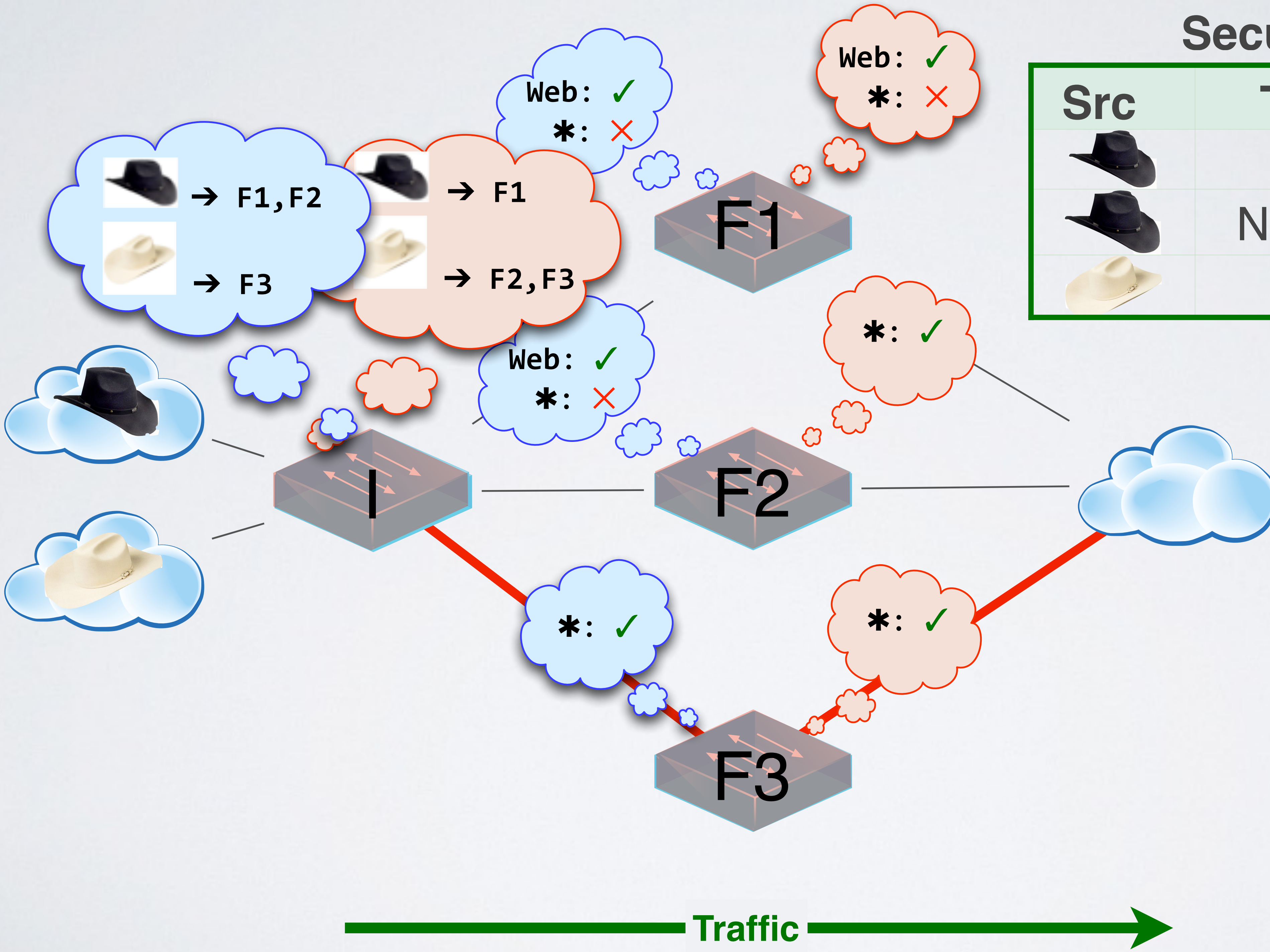
- Tools for whole network update

Our Approach

- Develop update abstractions
- Endow them with strong semantics
- Engineer efficient implementations



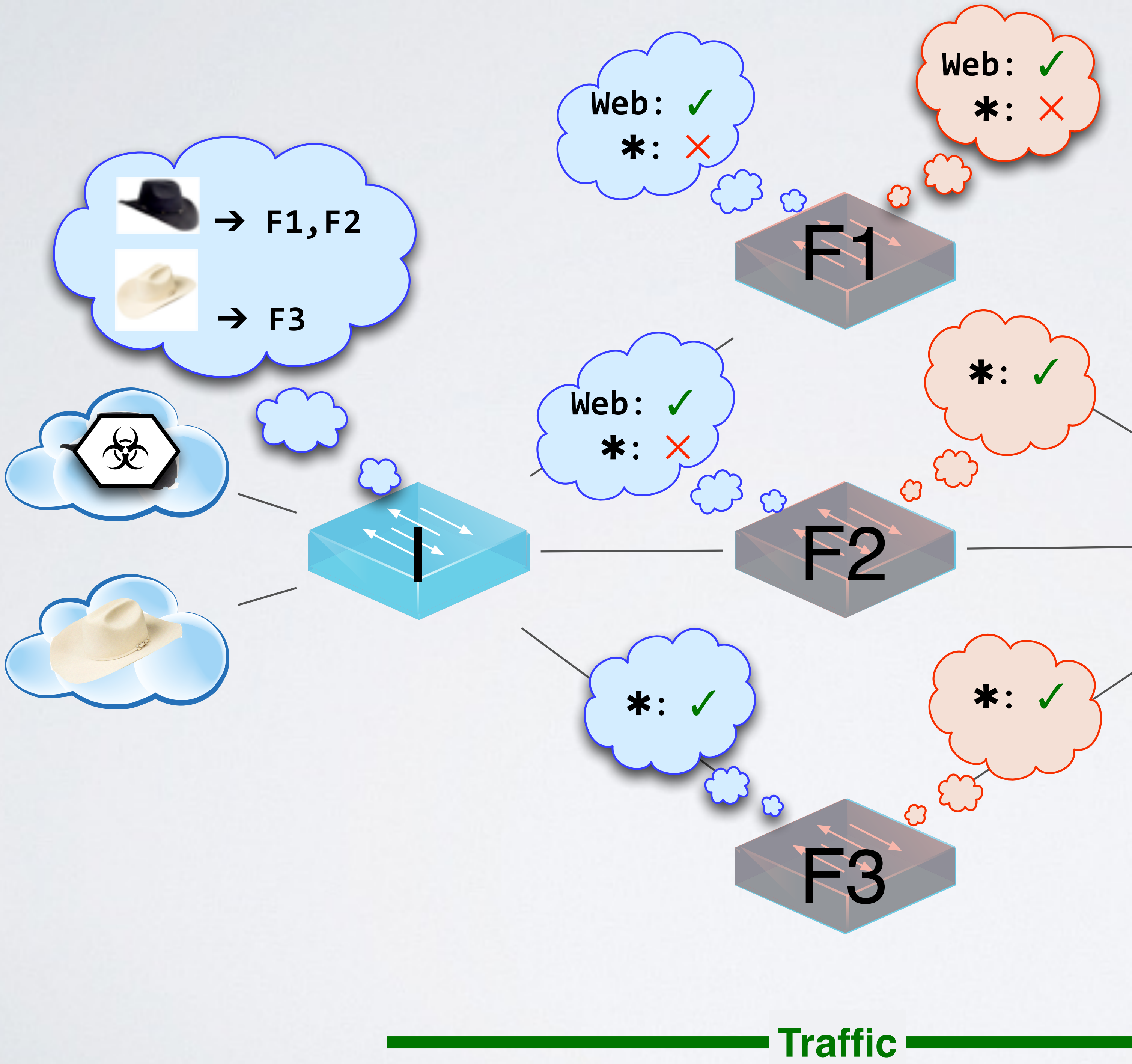
Example: Distributed Access Control



Security Policy

Src	Traffic	Action
	Web	Allow
	Non-web	Drop
	Any	Allow

Naive Update




Security Policy

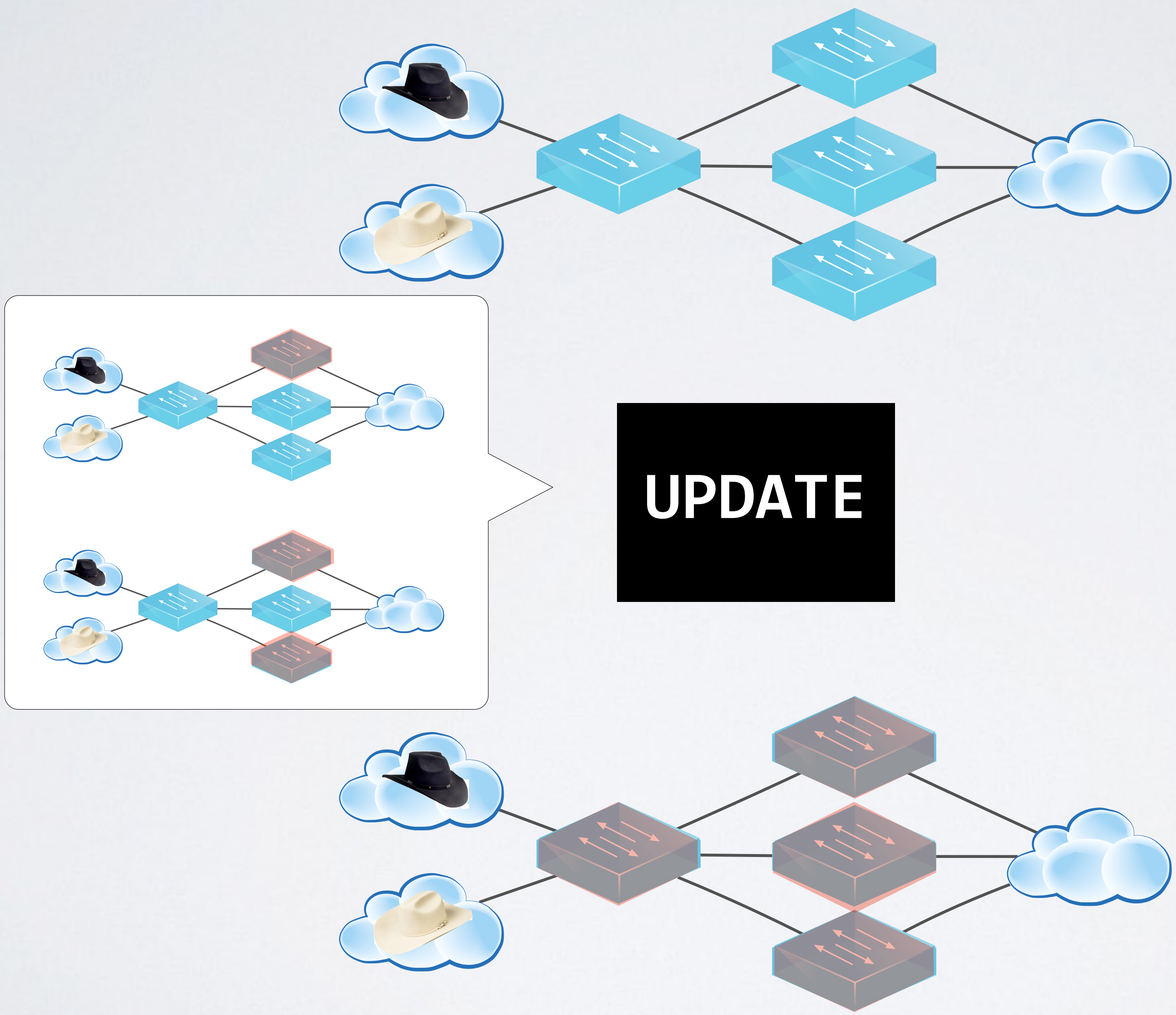
Src	Traffic	Action
	Web	Allow
	Non-web	Drop
	Any	Allow

Order

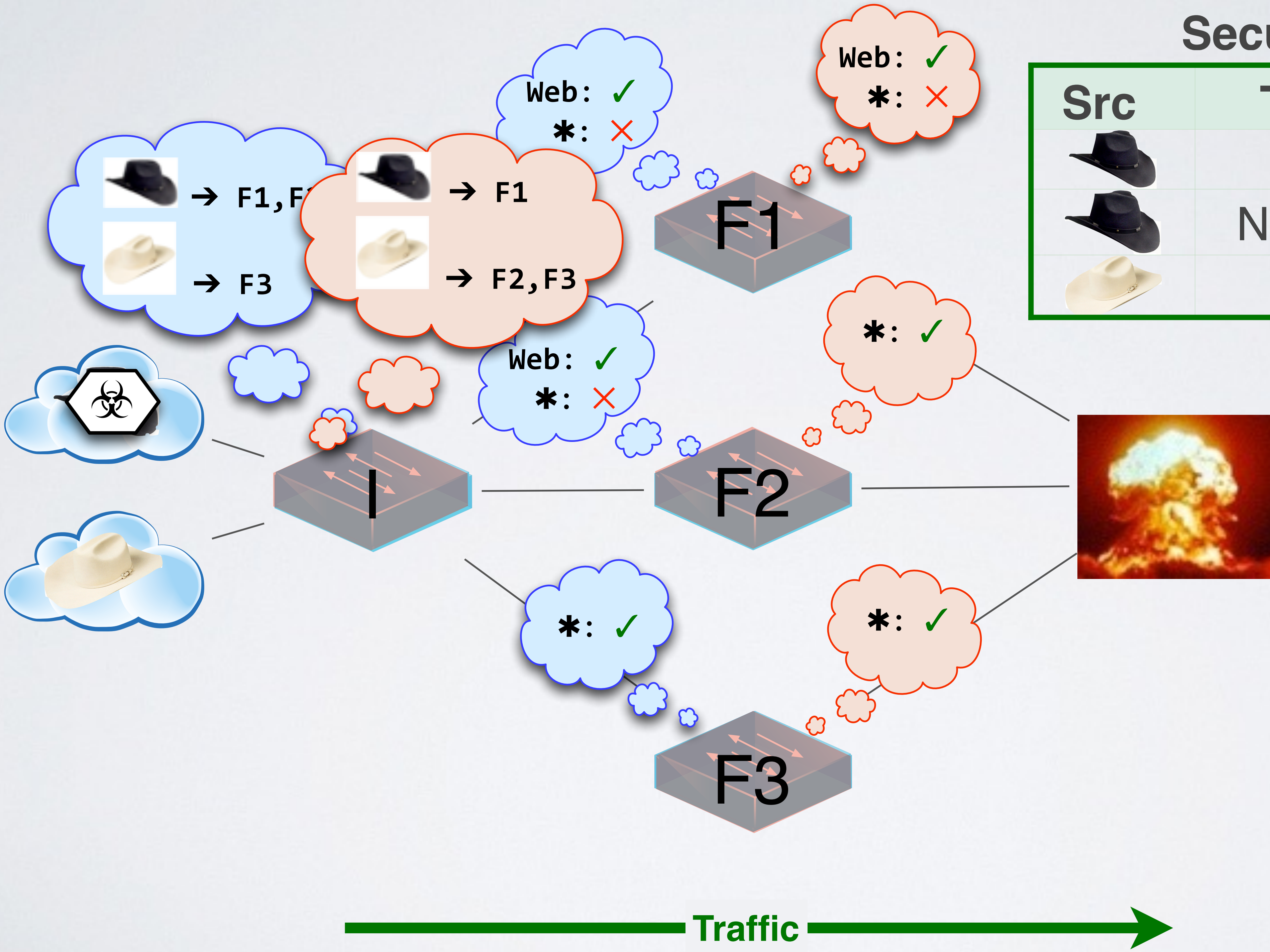
F1
F2
F3
I

Use an Abstraction!

Security Policy



Atomic Update?



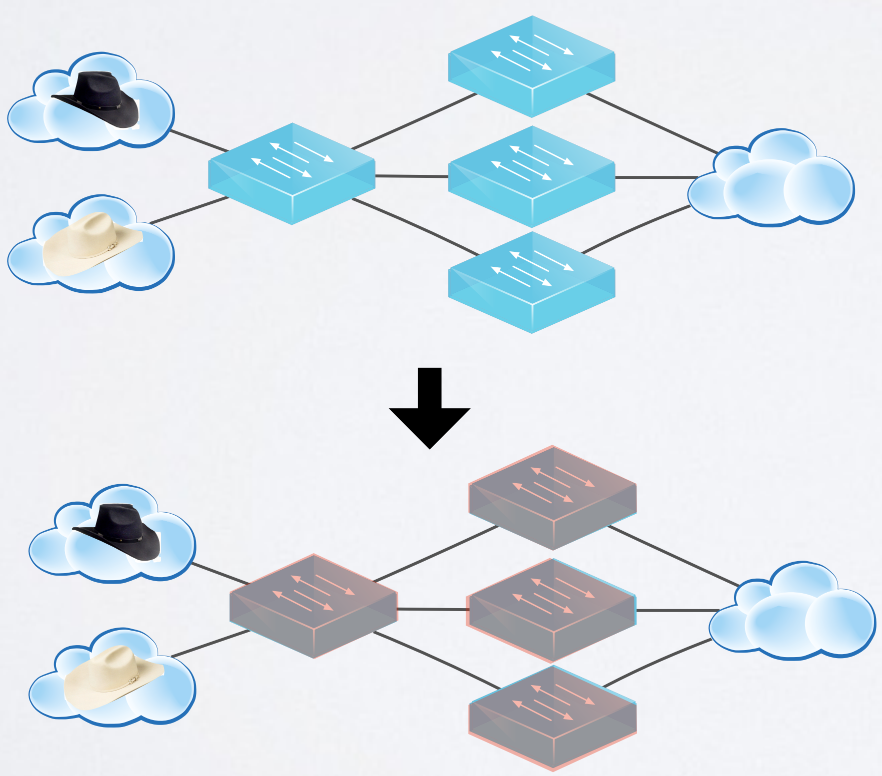
Security Policy

Src	Traffic	Action
	Web	Allow
	Non-web	Drop
	Any	Allow

Per-Packet Consistent Updates

Per-Packet Consistent Update
Each packet processed with old or new configuration, but not a mixture of the two.

Obeys policy:



Obeys policy:

Security Policy

Src	Traffic	Action
	Web	Allow
	Non-web	Drop
	Any	Allow

Universal Property Preservation

Theorem: Per-packet consistent updates preserve all trace properties.

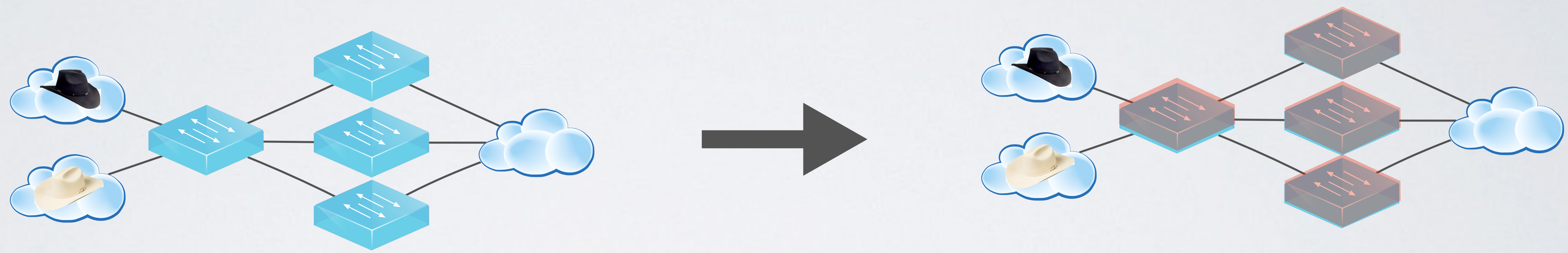
Trace Property
Any property of a *single* packet's path through the network.

Examples of Trace Properties:
Loop freedom, access control, waypointing ...

Trace Property Verification Tools:
Anteater , Header Space Analysis, ConfigChecker ...

Formal Verification

Corollary: To check an invariant, verify the old and new configurations.



Security Policy → Analyzer ✓

Security Policy → Analyzer ✓

Verification Tools

- Anteater [SIGCOMM ’11]
- Header Space Analysis [NSDI ’12]
- ConfigChecker [ICNP ’09]

MECHANISMS

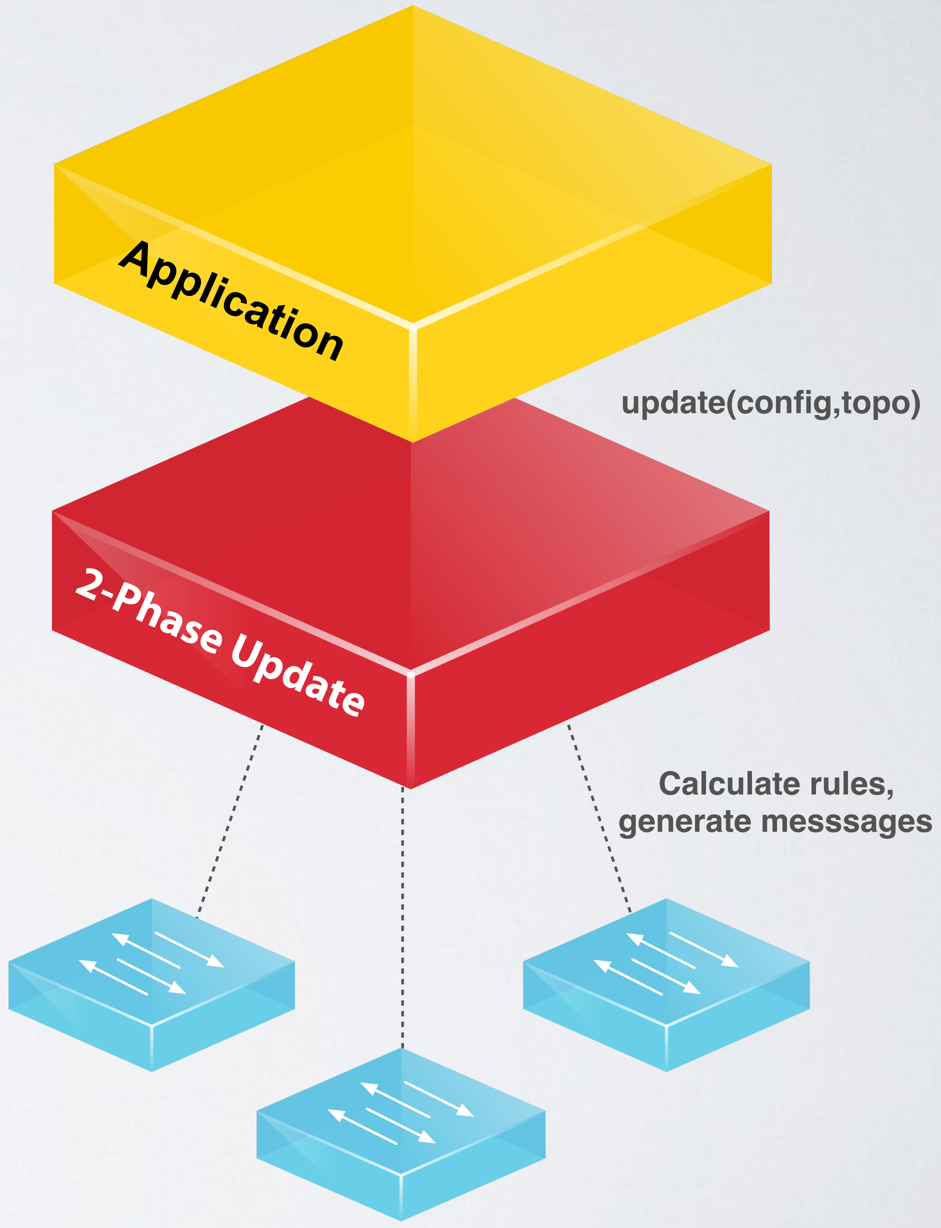
2-Phase Update

Overview

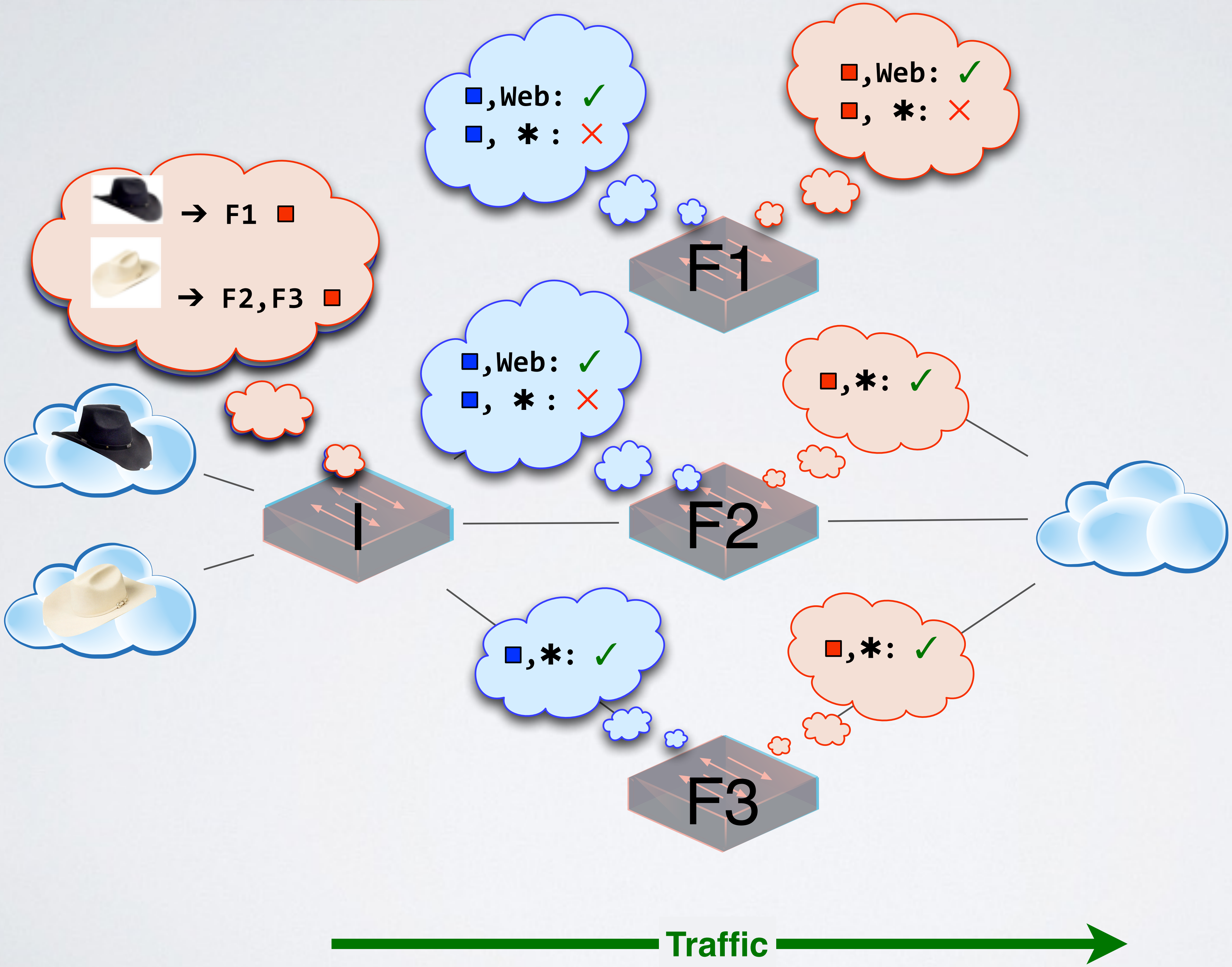
- Runtime instruments configurations
- Edge rules stamp packets with version
- Forwarding rules match on version

Algorithm (2-Phase Update)

1. Install new rules on internal switches, leave old configuration in place
2. Install edge rules that stamp with the new version number



2-Phase Update in Action



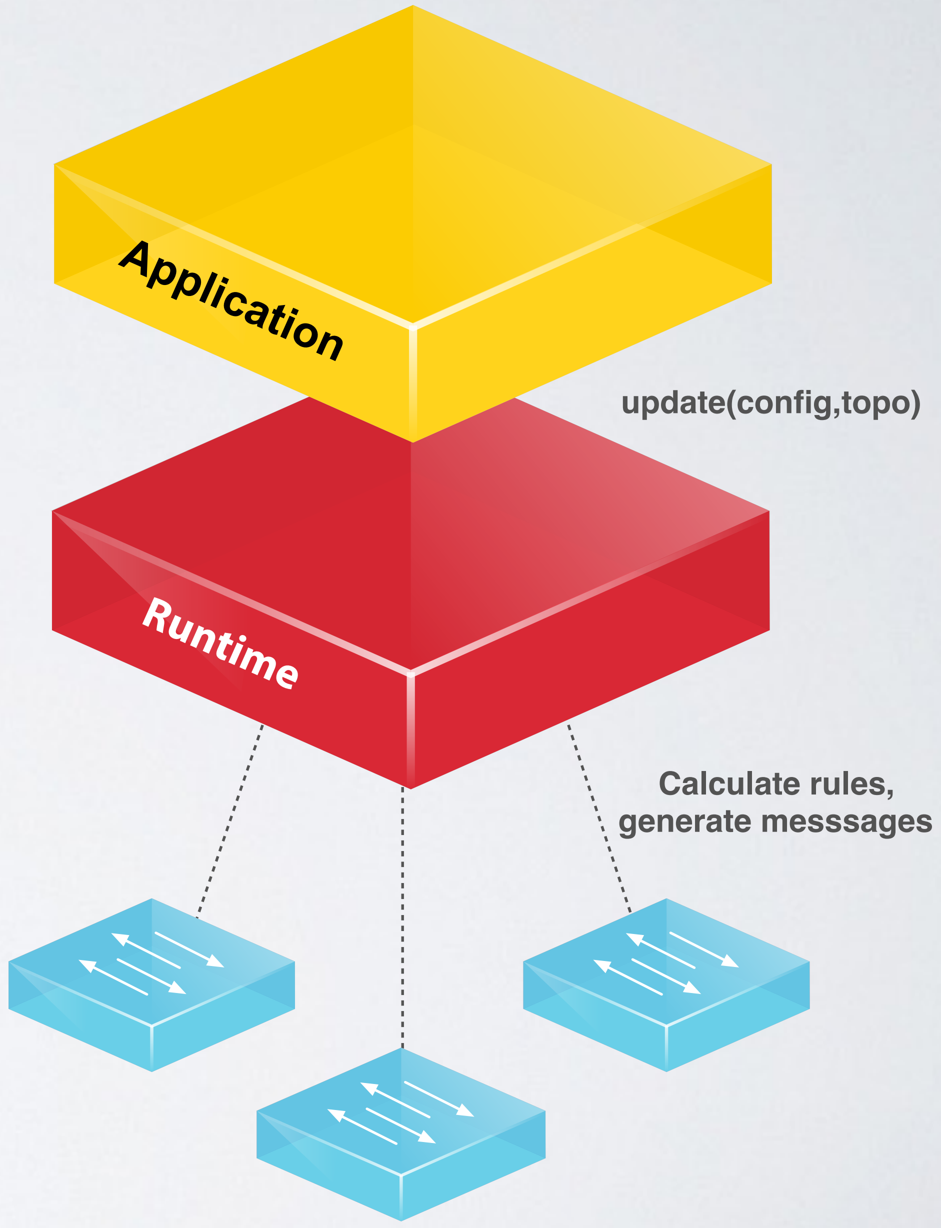
Optimized Mechanisms

Optimizations

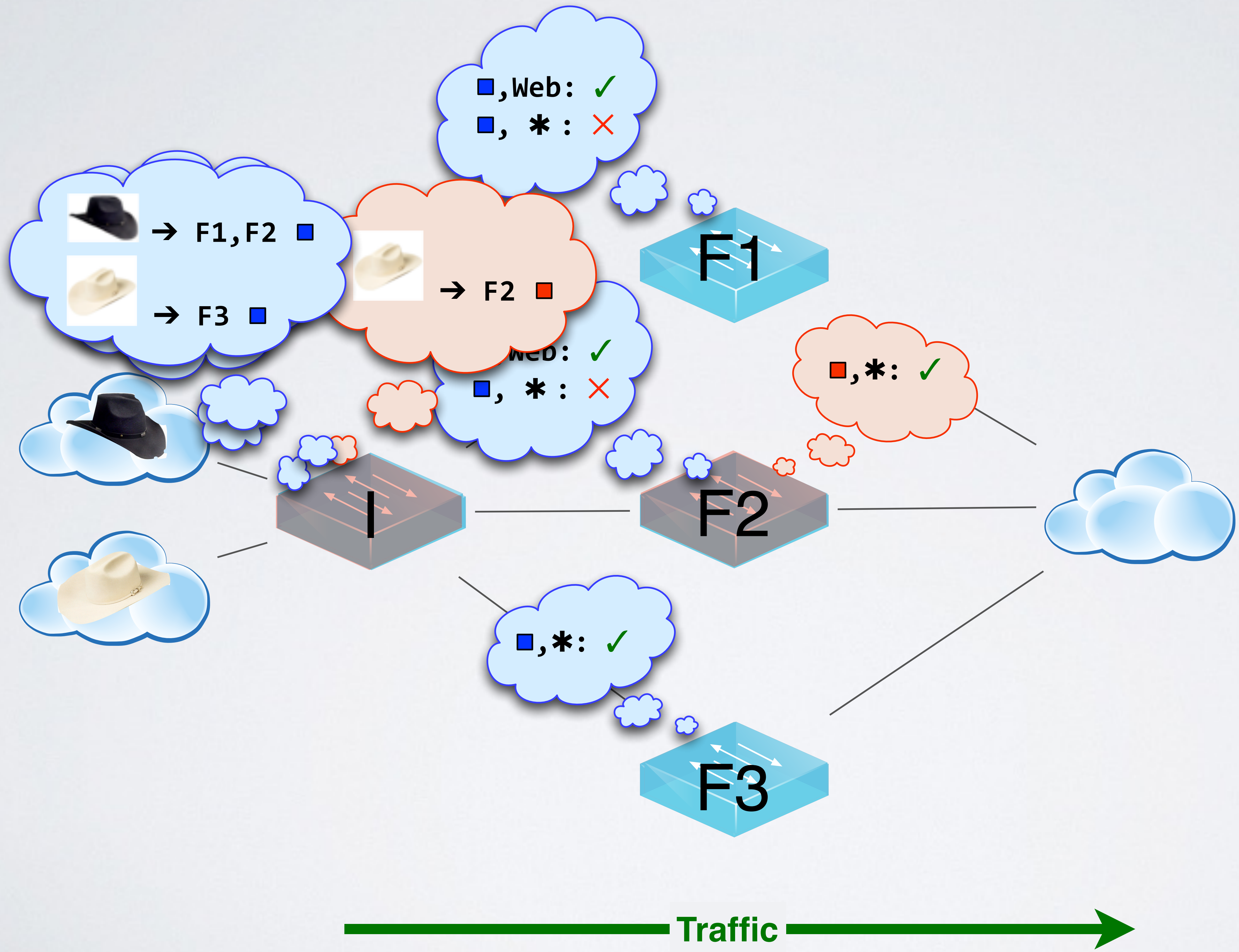
- Extension: strictly adds paths
- Retraction: strictly removes paths
- Subset: affects small # of paths
- Topological: affects small # of switches

Runtime

- Automatically optimizes
- Power of using abstraction



Subset Optimization



Correctness

Question: How do we convince ourselves these mechanisms are correct?

Solution: We built an operational semantics, formalized our mechanisms and proved them correct

Example: 2-Phase Update

1. Install new rules on internal switches, leave old configuration in place
2. Install edge rules that stamp with the new version number

$\}$ Unobservable

$\}$ One-touch

Theorem: Unobservable + one-touch = per-packet.

IMPLEMENTATION
&
EVALUATION

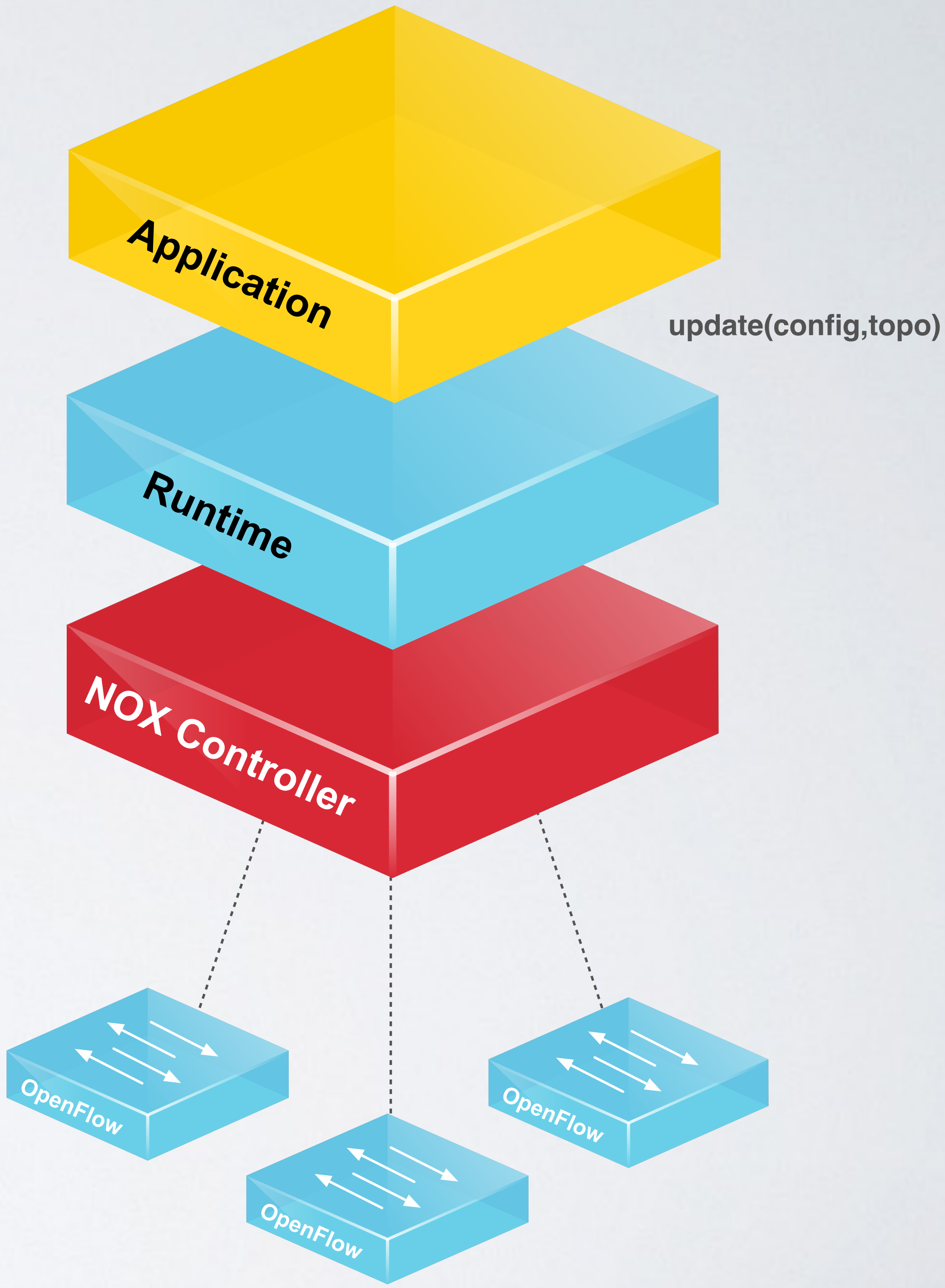
Implementation

Runtime

- NOX Library
  - OpenFlow 1.0
- 2.5k lines of Python
- `update(config, topology)`
  - Uses VLAN tags for versions
- Automatically applies optimizations

Verification Tool

- Checks OpenFlow configurations
- CTL specification language
- Uses NuSMV model checker



Evaluation

Question: How much extra rule space is required?

Setup

- Mininet VM

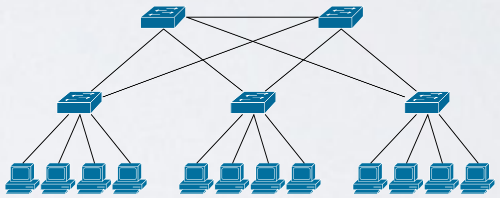
Applications

- Routing and Multicast

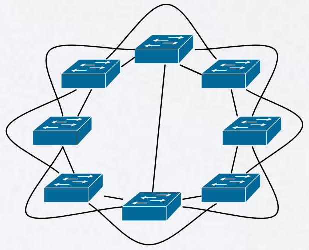
Scenarios

- Adding/removing hosts
- Adding/removing links
- Both at the same time

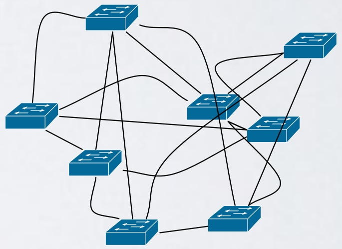
Topologies



Fattree

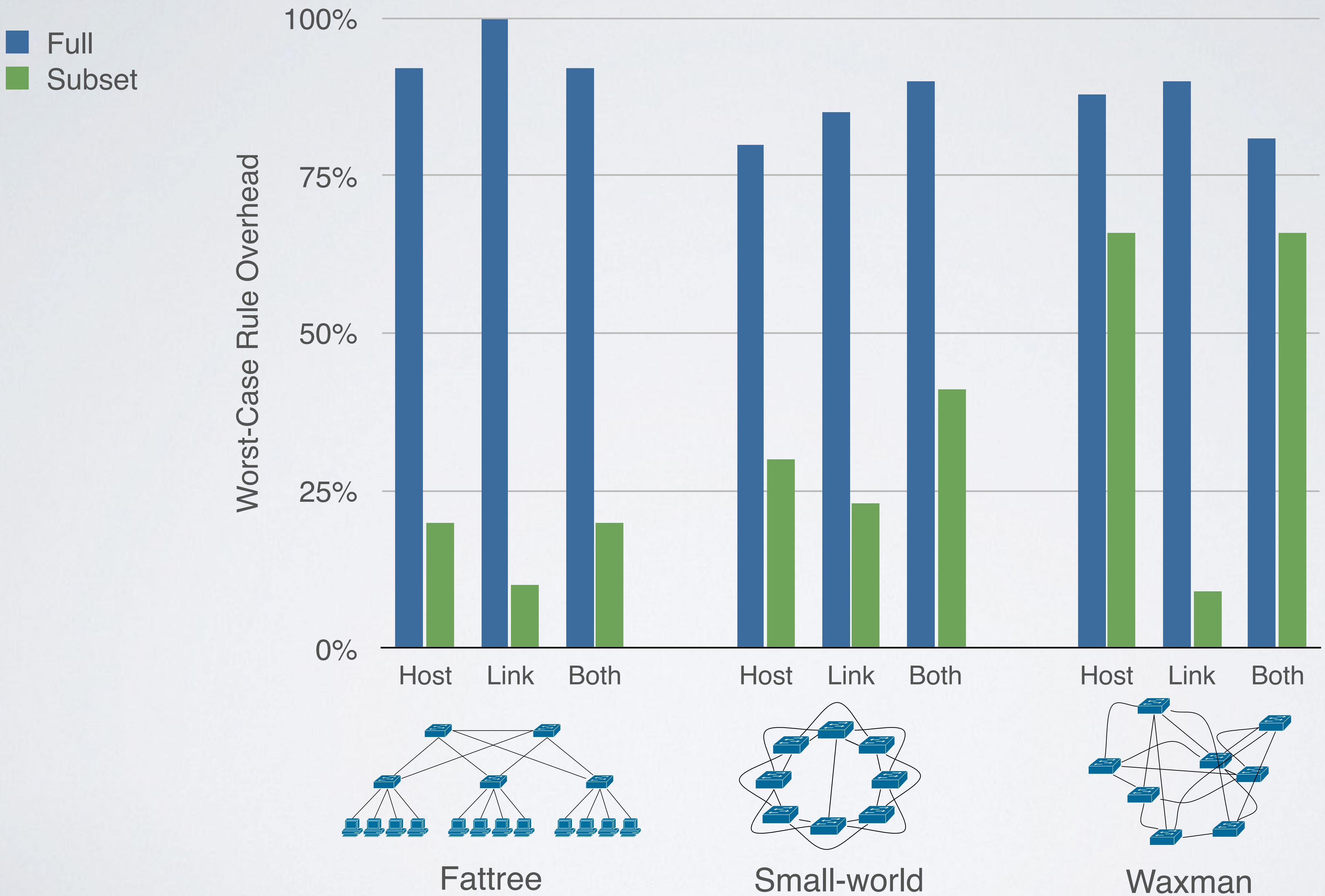


Small-world



Waxman

Results: Routing Application



Beyond today's research

Industry policy languages, e.g. OpenStack Congress [https://wiki.openstack.org/wiki/Congress]

- *"App A is only allowed to communicate with app B."*
- *"Virtual machine owned by tenant A should always have a public network connection if tenant A is part of the group B."*
- *"Virtual machine A should never be provisioned in a different geographic region than storage B."*

High-level abstractions one of the big open questions for SDN

- What can people use? Who is doing the programming?
- What's the killer app?
- Does different hardware change the abstraction?