Capturing Intent

Brighten Godfrey

CS 538 March 5, 2018

Project Midterm Presentations

Two key goals

- Benchmark: Demonstrate concrete progress
- Feedback & discussion with your peers

Content

- What problem are you solving?
- Why has past work not addressed the problem?
- What is your approach for solving it?
- What are your preliminary results & progress?

Logistics

- 10 minutes total: 6:40 min presentation + 4 min discuss
- PechaKucha format: 20 slides x 20 seconds, auto-advance

Grand Challenge: Capturing Intent

We need networks that are

- Flexible
  - As adaptable and programmable as a well-designed software system
- Intuitive
  - Given a high level goal, the details are automated

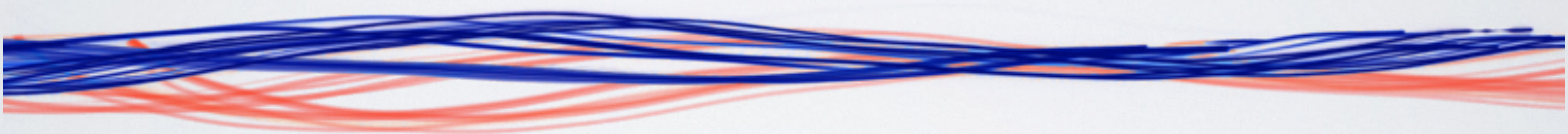
Network Updates

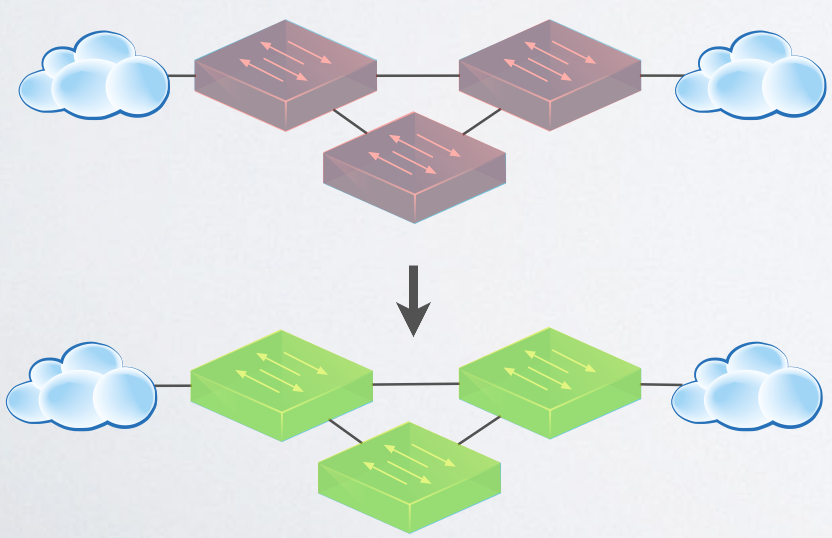
Slides courtesy
Nate Foster

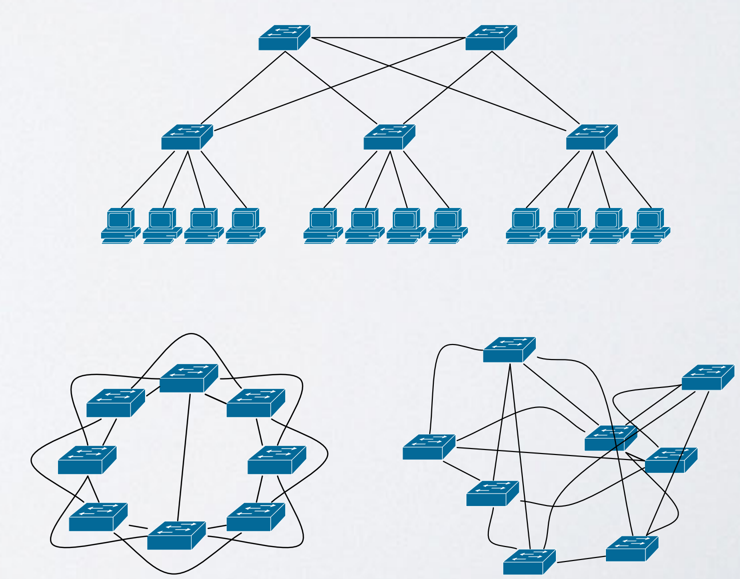
Abstractions for Network Update

Nate Foster
Mark Reitblatt

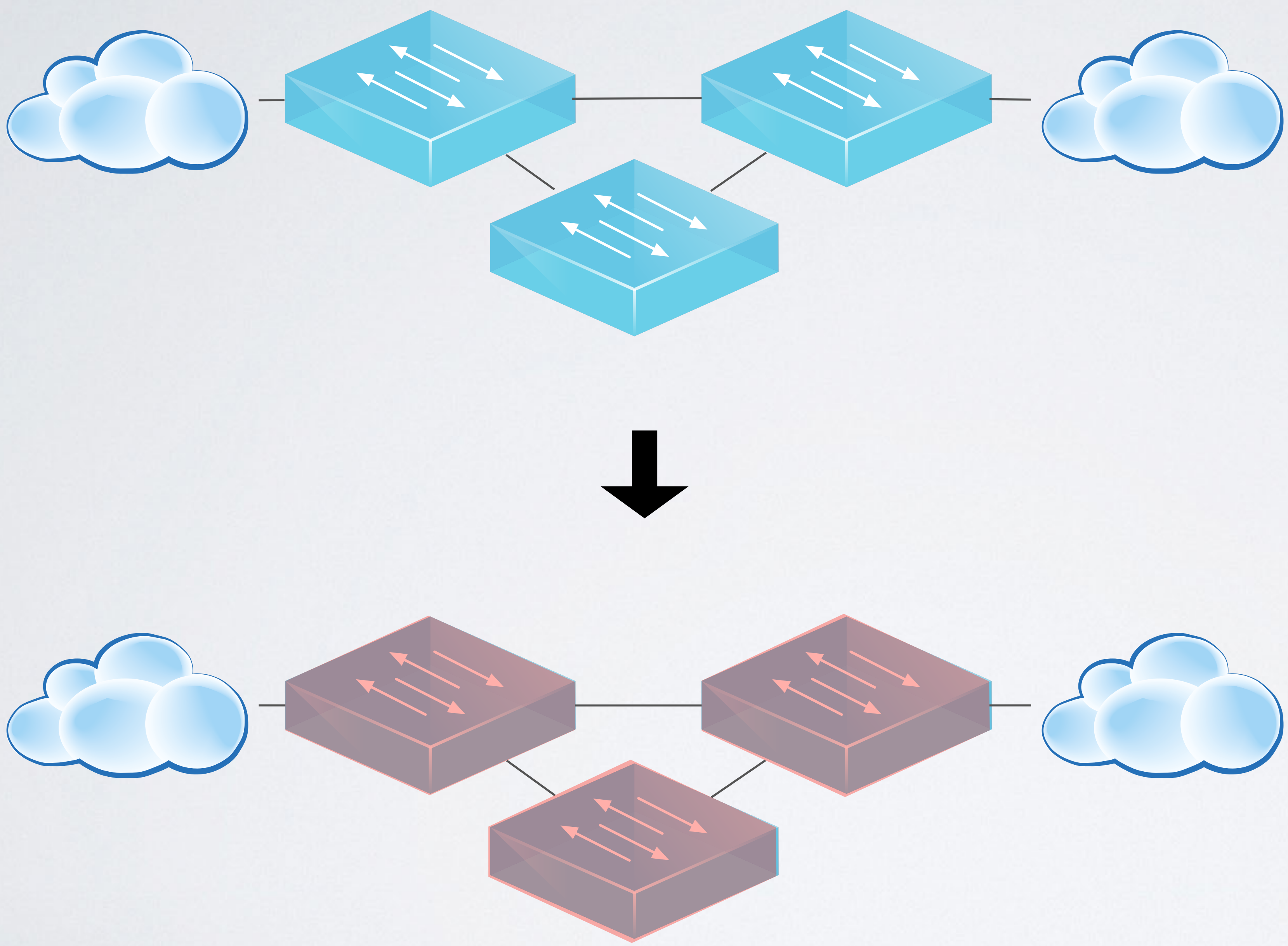
Jen Rexford
Cole Schlesinger
Dave Walker







Updates Happen



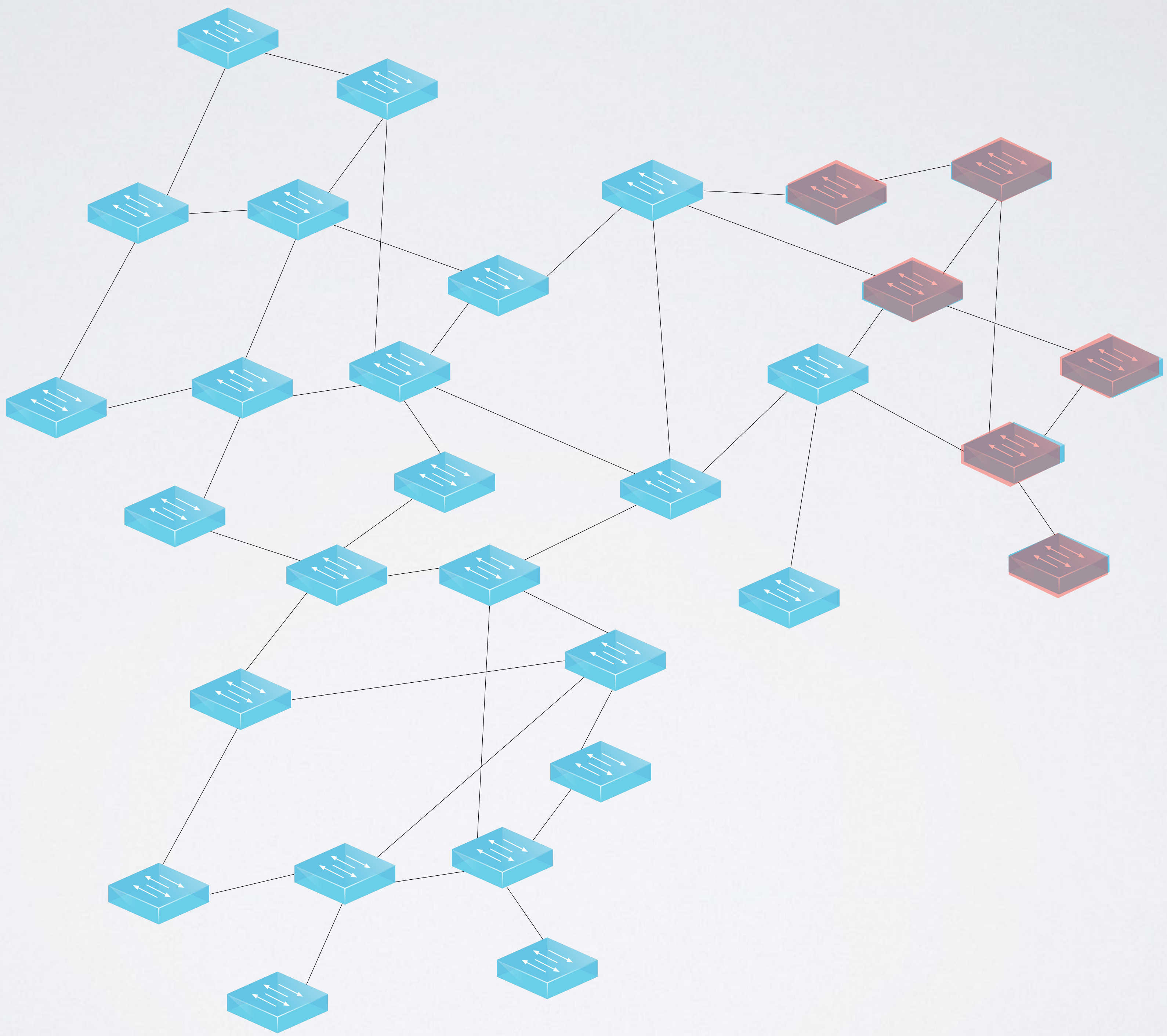
Network Updates

- Maintenance
- Failures
- ACL Updates

Desired Invariants

- No black-holes
- No loops
- No security violations

Network Updates Are Hard



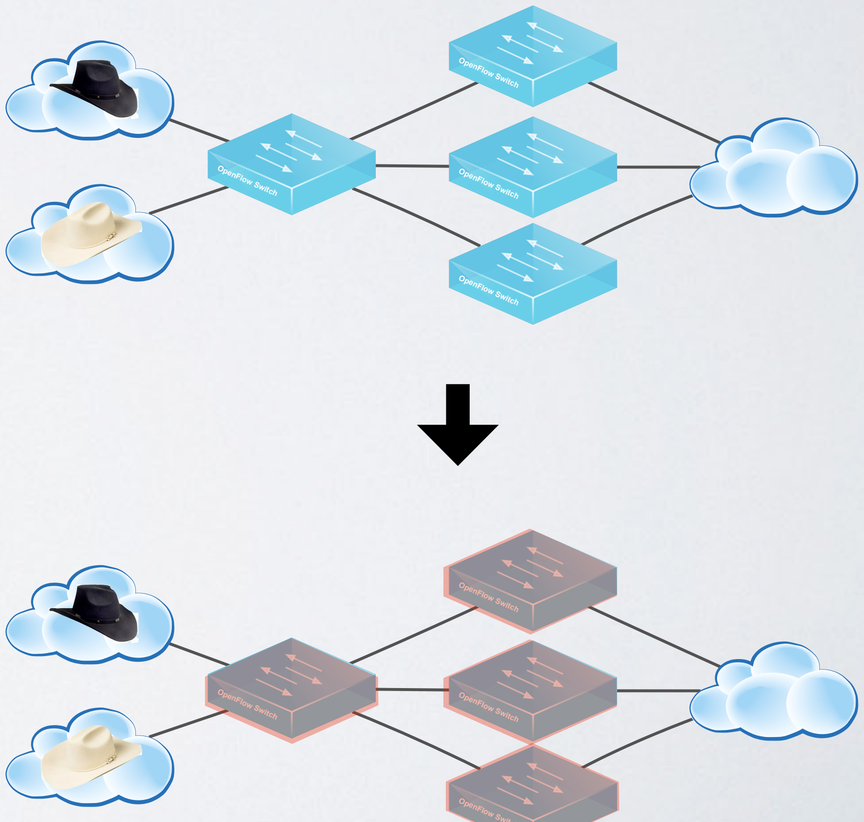
Network Update Abstractions

Goal

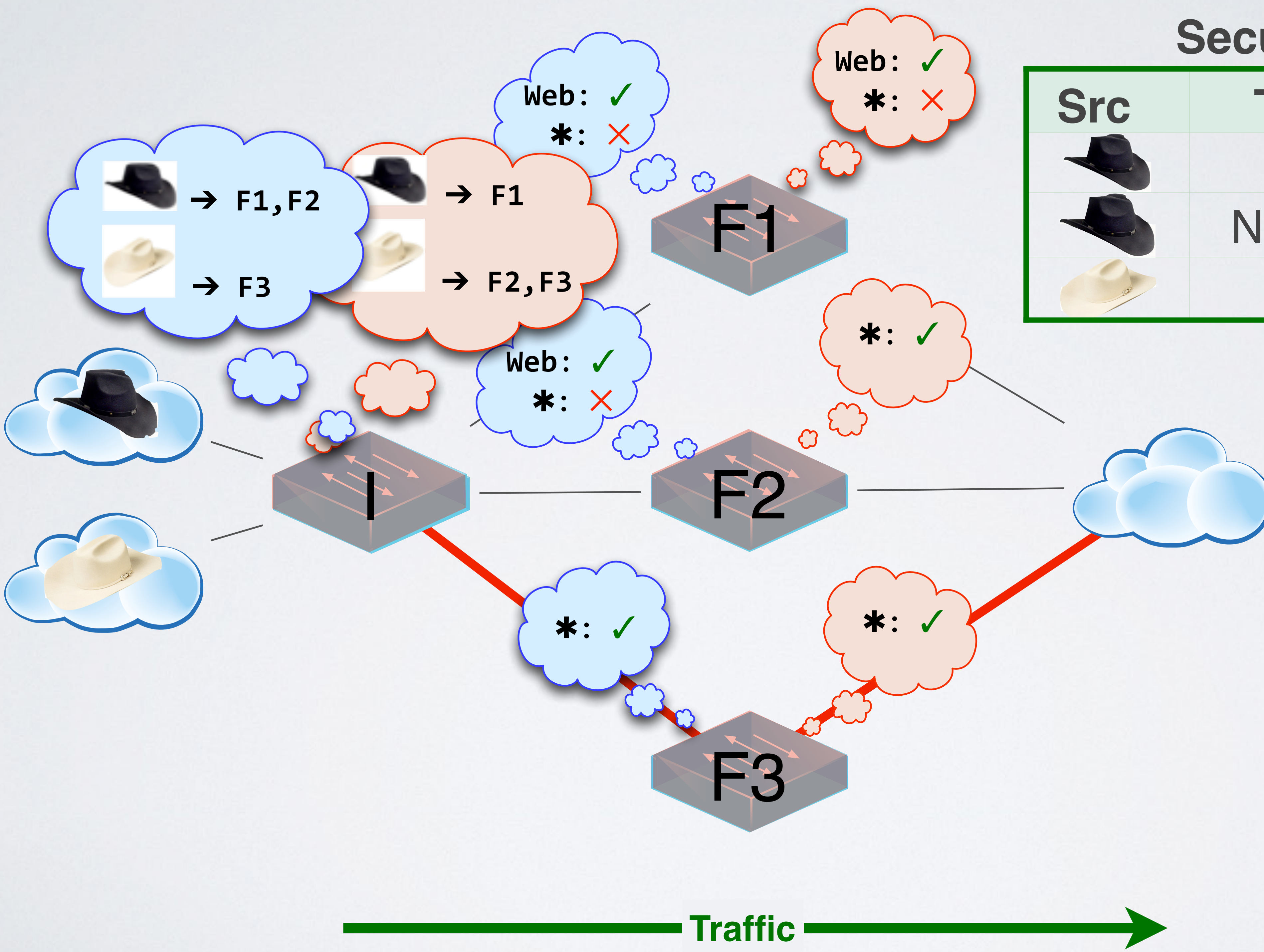
- Tools for whole network update

Our Approach

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



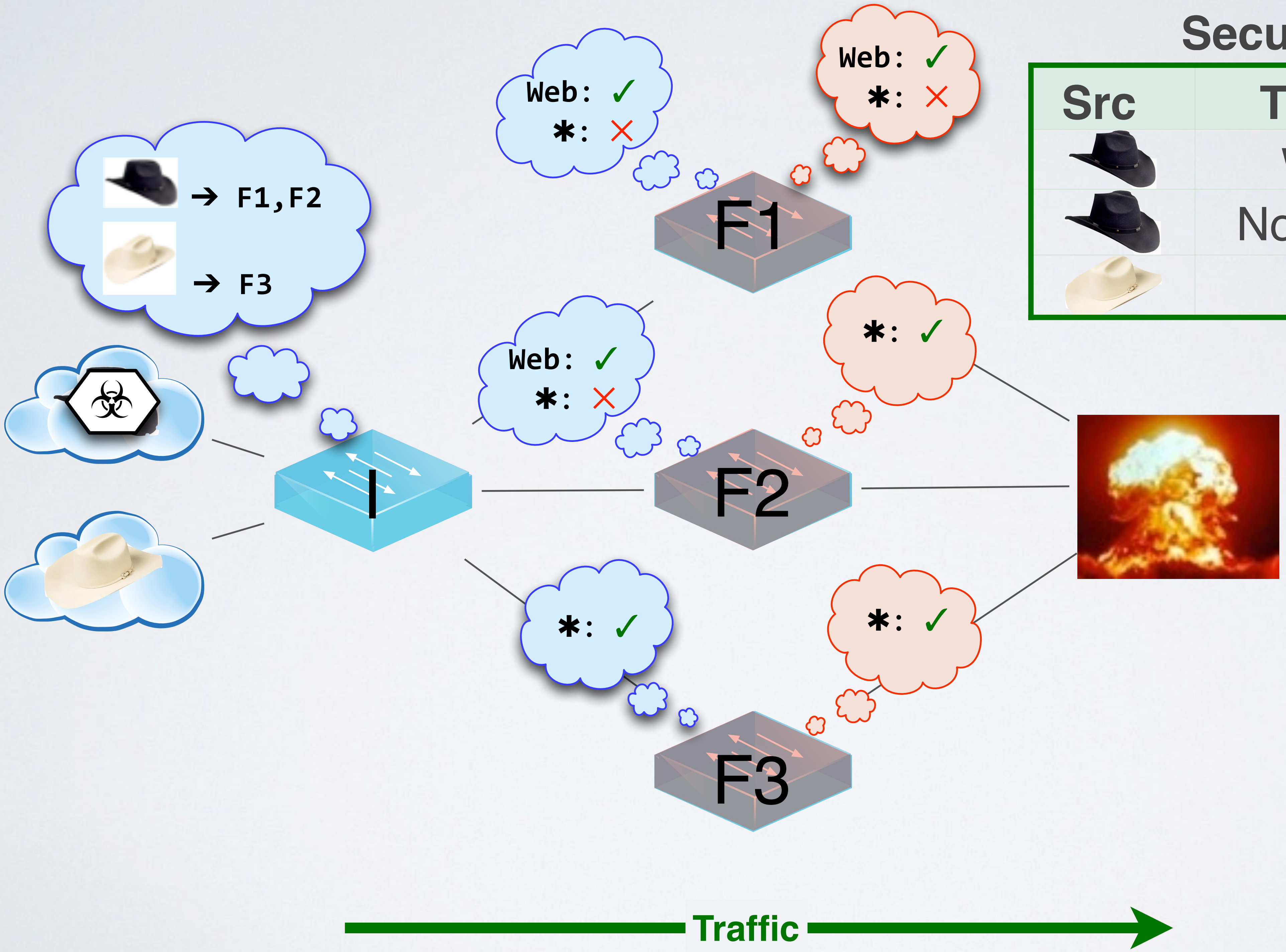
Example: Distributed Access Control



Security Policy

Src	Traffic	Action
	Web	Allow
	Non-web	Drop
	Any	Allow

Naive Update



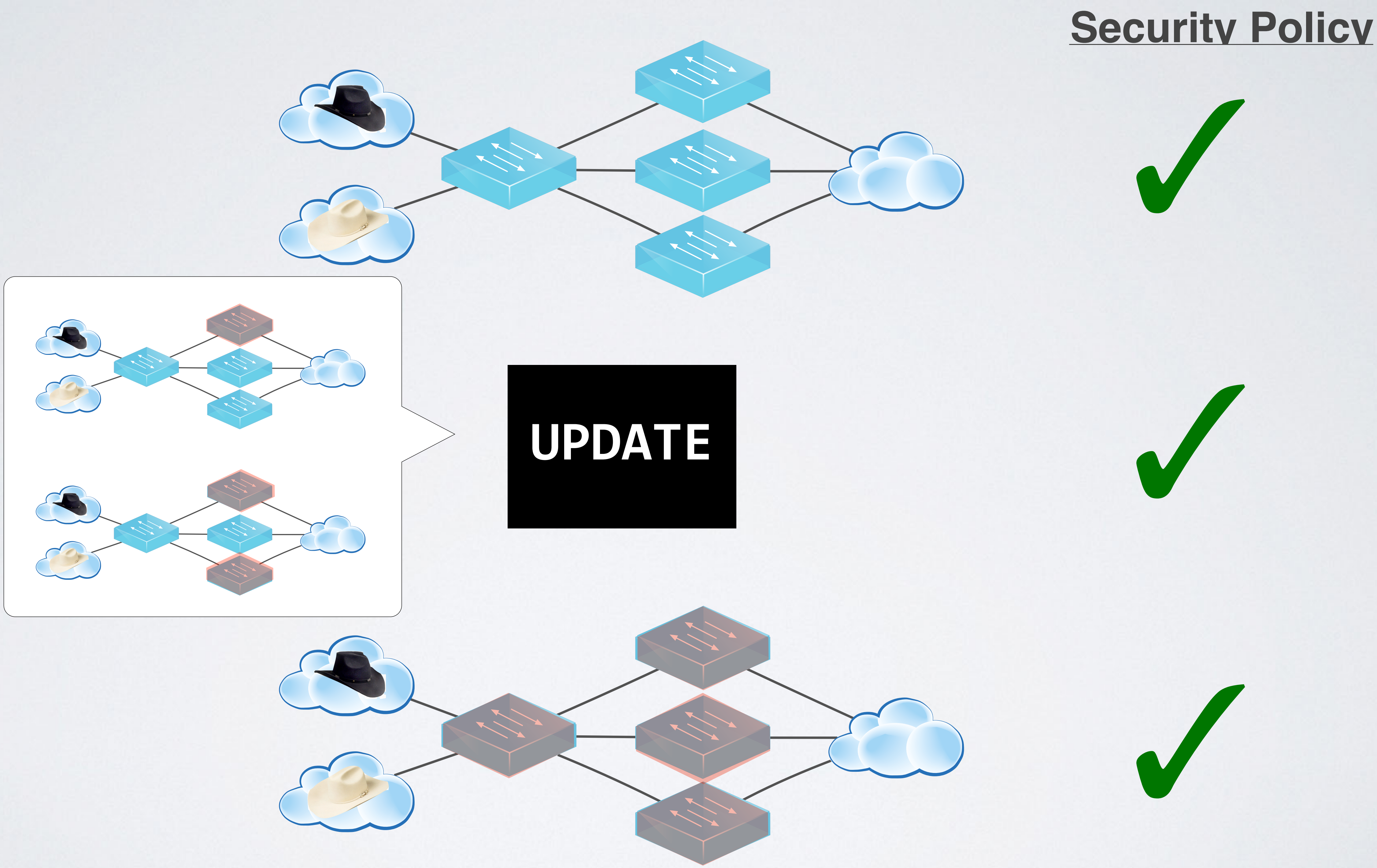
Security Policy

Src	Traffic	Action
	Web	Allow
	Non-web	Drop
	Any	Allow

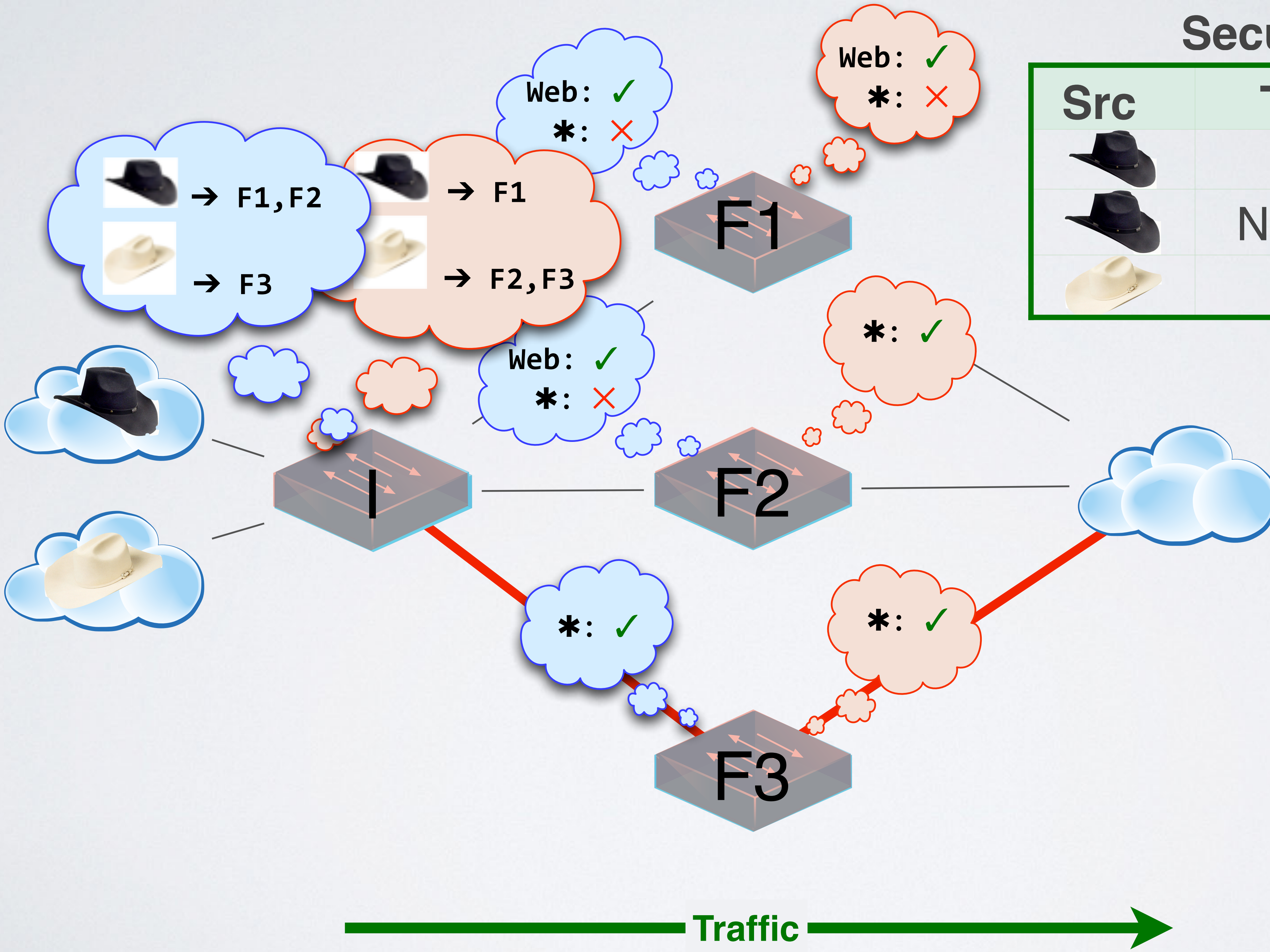
Order

F1
F2
F3
I

Use an Abstraction!



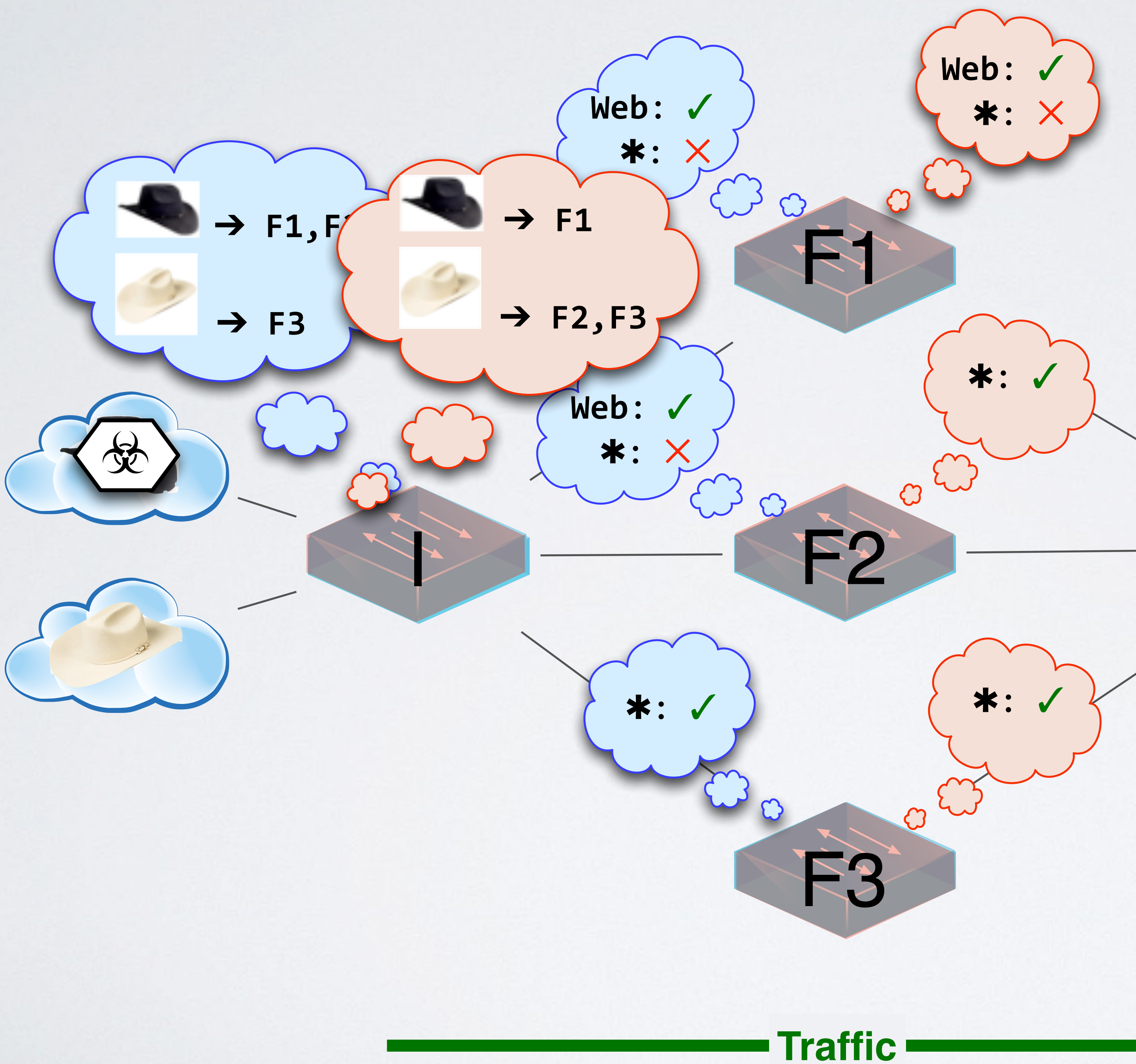
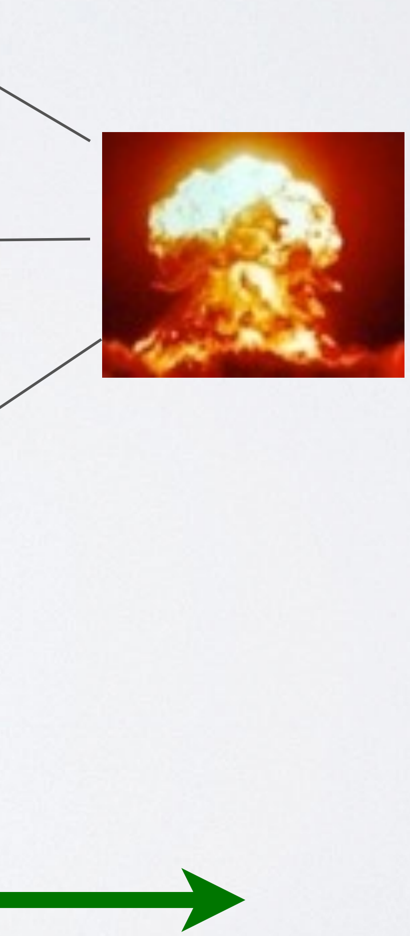
Q: What's the right order to update?



Security Policy

Src	Traffic	Action
(black hat)	Web	Allow
(black hat)	Non-web	Drop
(white hat)	Any	Allow

A: Even atomic update doesn’t work!

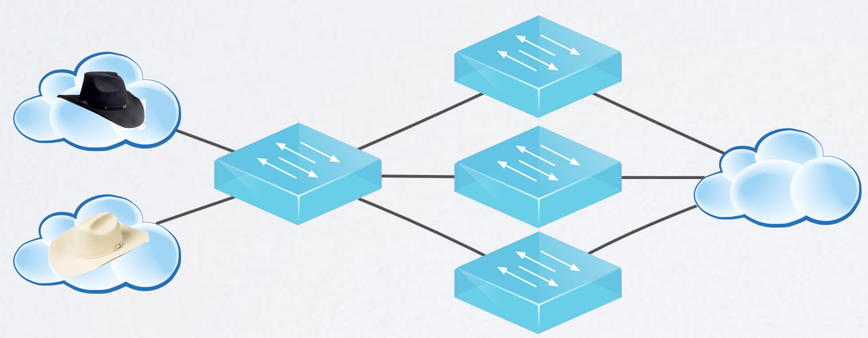
Security Policy

Src	Traffic	Action
	Web	Allow
	Non-web	Drop
	Any	Allow

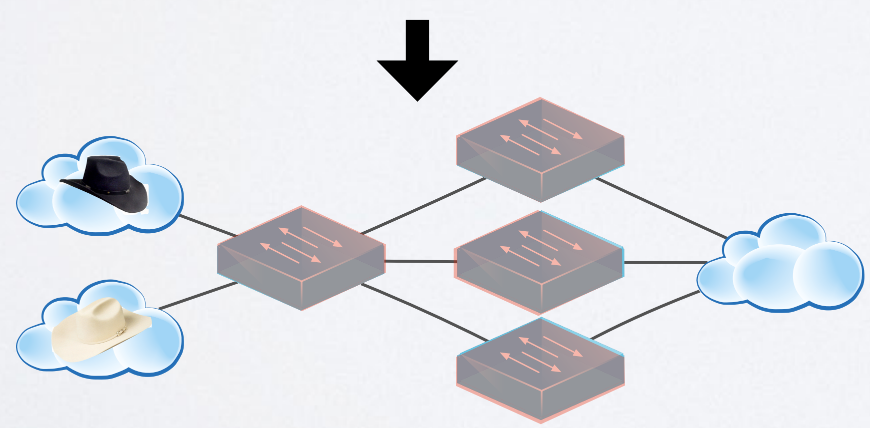
Per-Packet Consistent Updates

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.

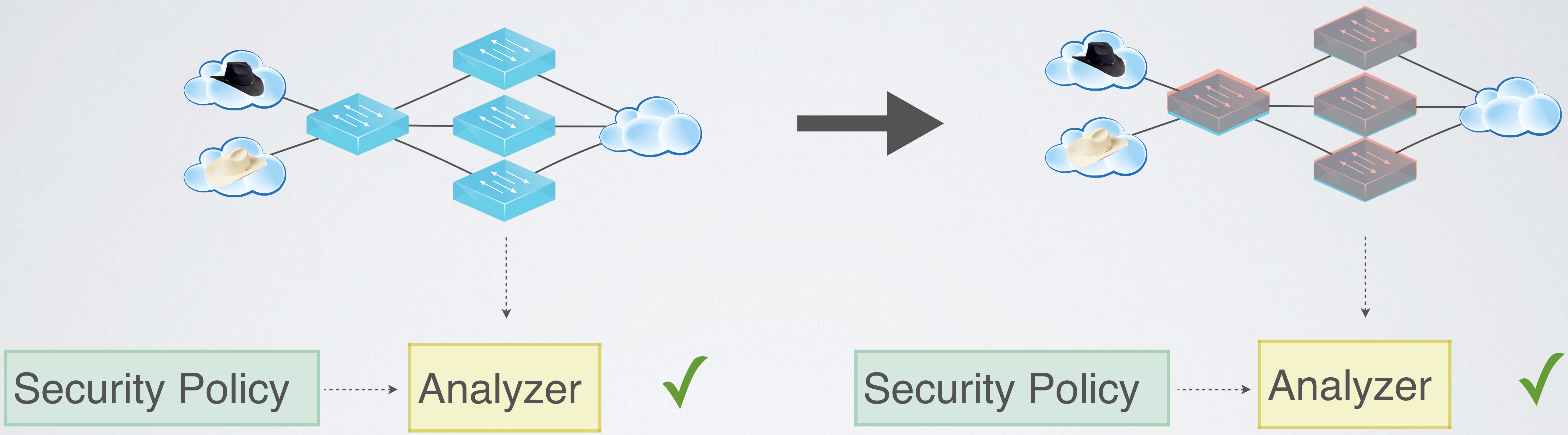
Trace Property
Any property of a *single* packet's path through the network.

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.



Verification Tools

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

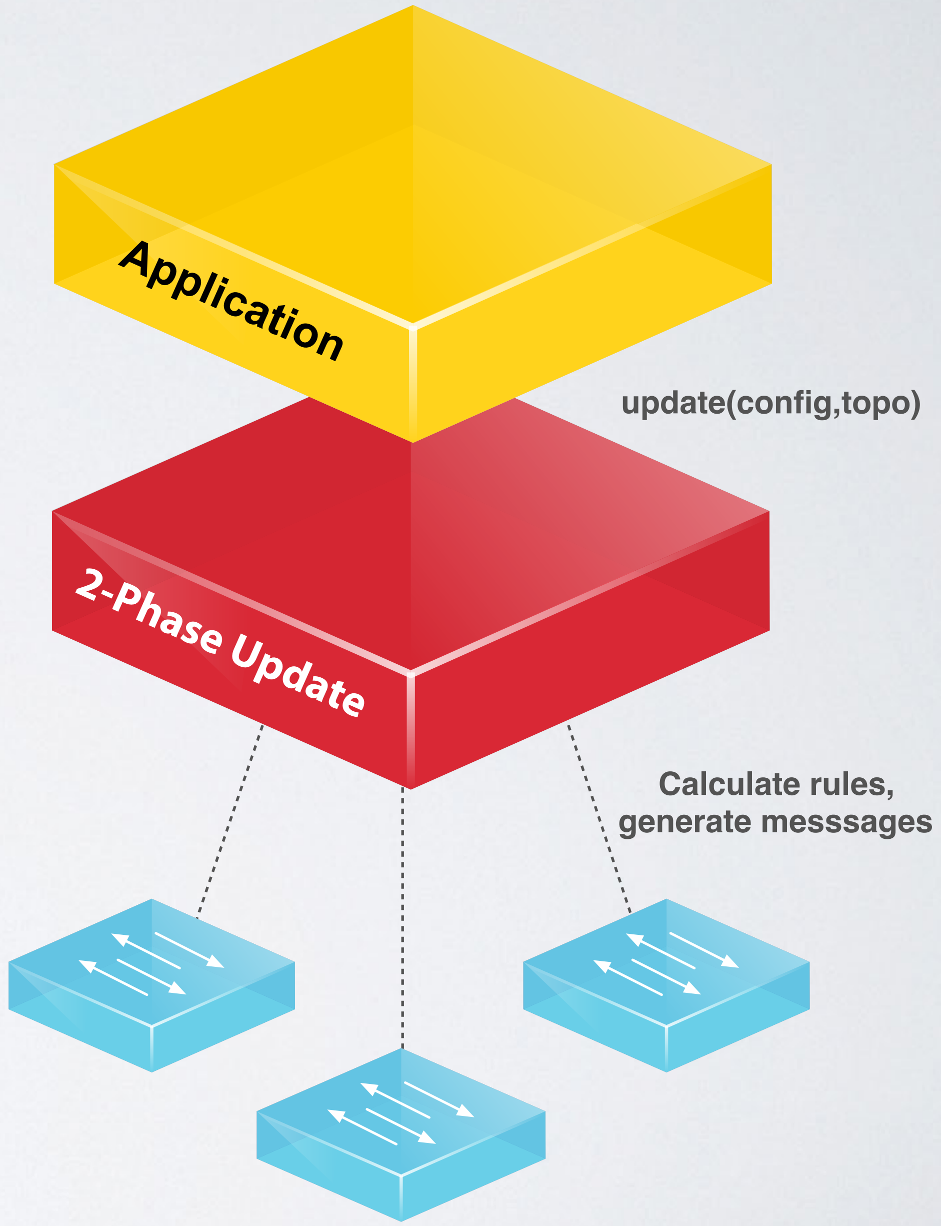
MECHANISMS

2-Phase Update

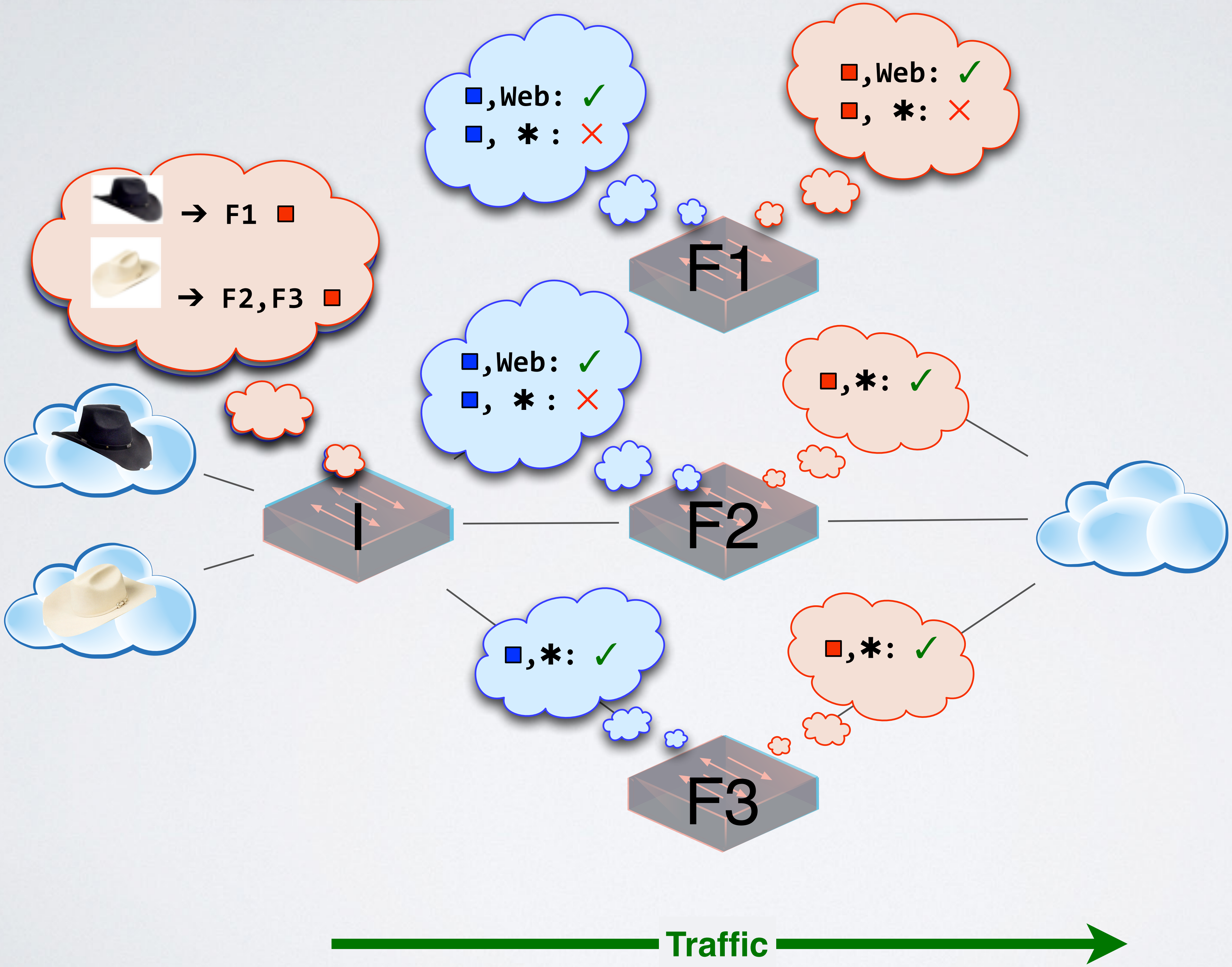
Overview
- Runtime instruments configurations
- Edge rules stamp packets with version
- Forwarding rules match on version

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



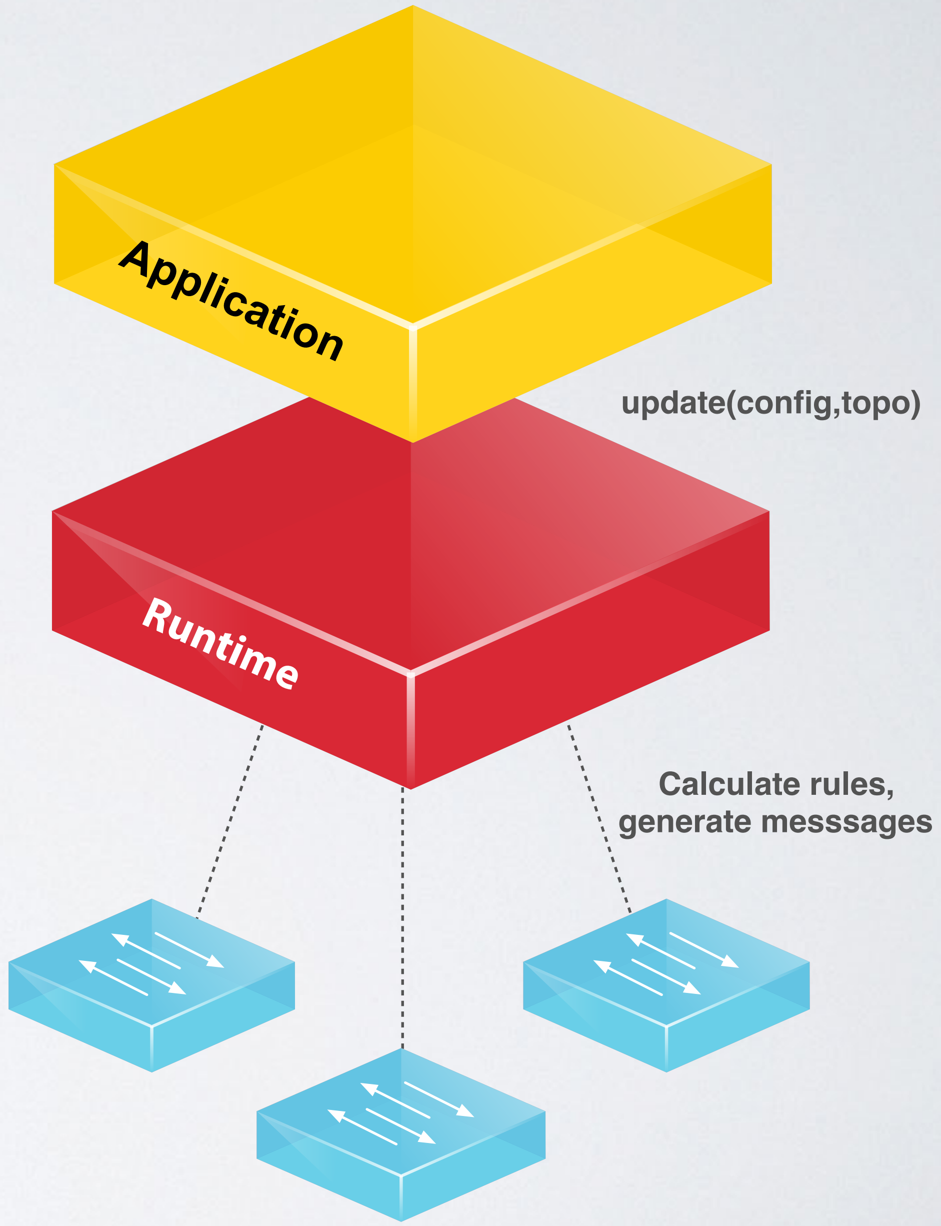
Optimized Mechanisms

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



IMPLEMENTATION
&
EVALUATION

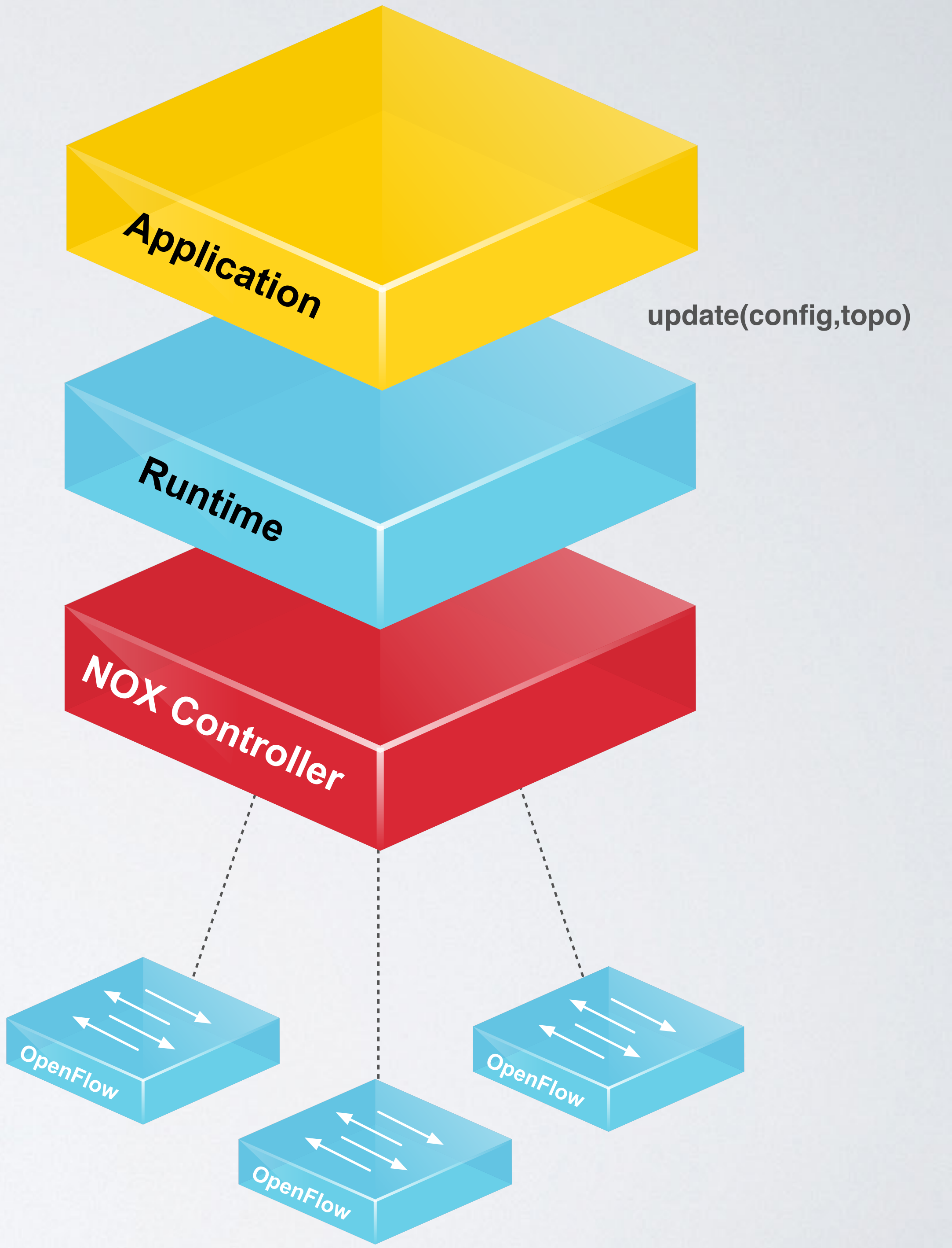
Implementation

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

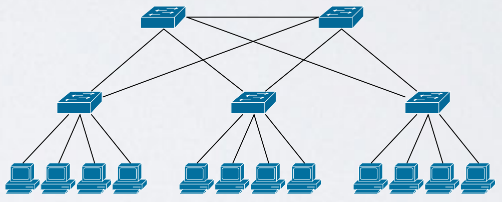
Applications

- Routing and Multicast

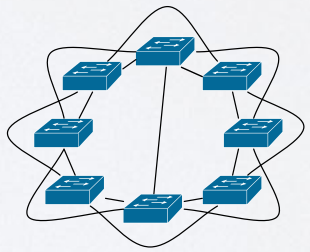
Scenarios

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

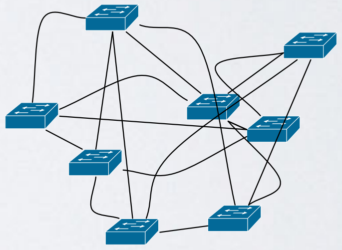
Topologies



Fattree

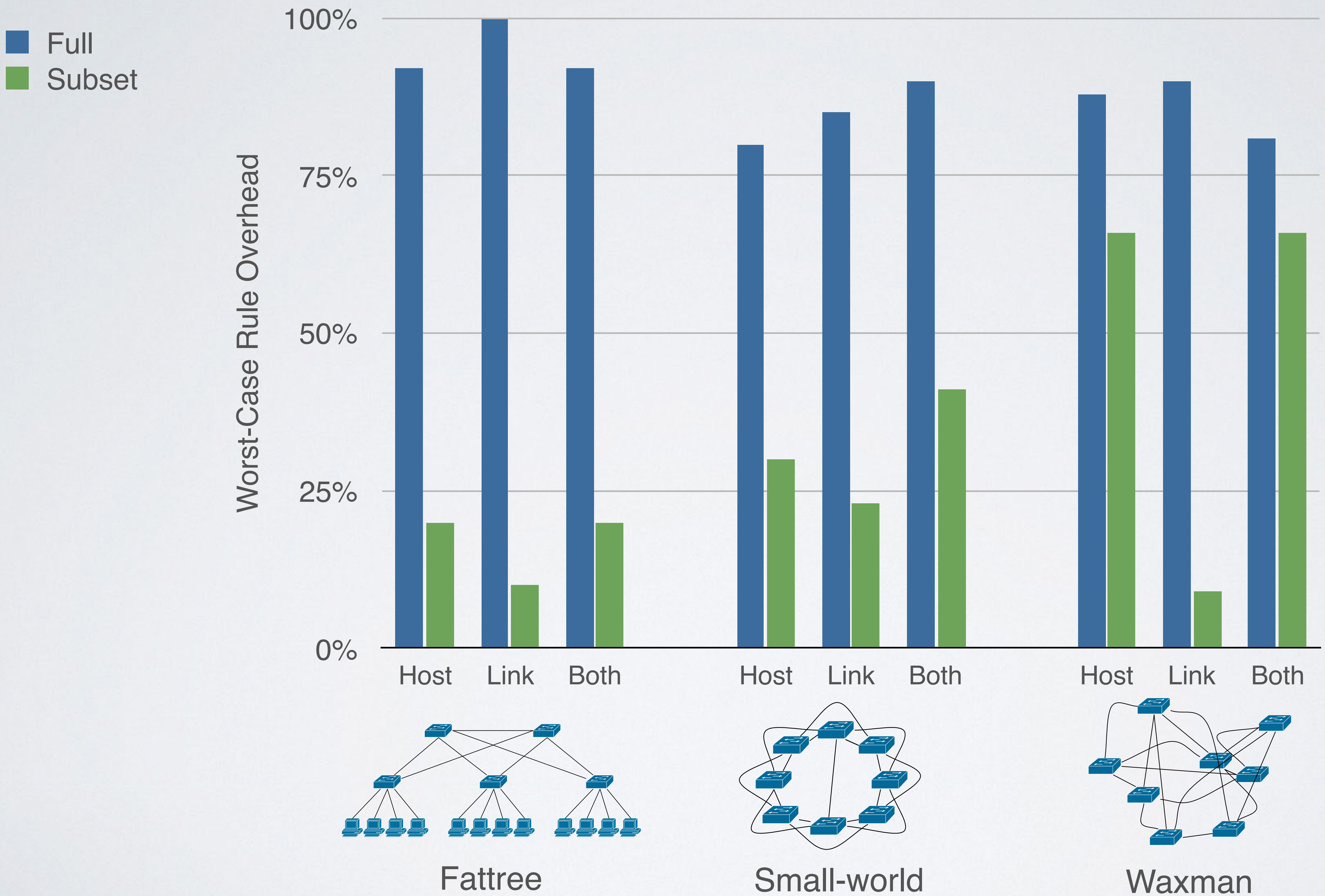


Small-world



Waxman

Results: Routing Application



Propane

Propane: Key Concepts

Don't Mind the Gap: Bridging Network-wide Objectives and Device-level Configurations

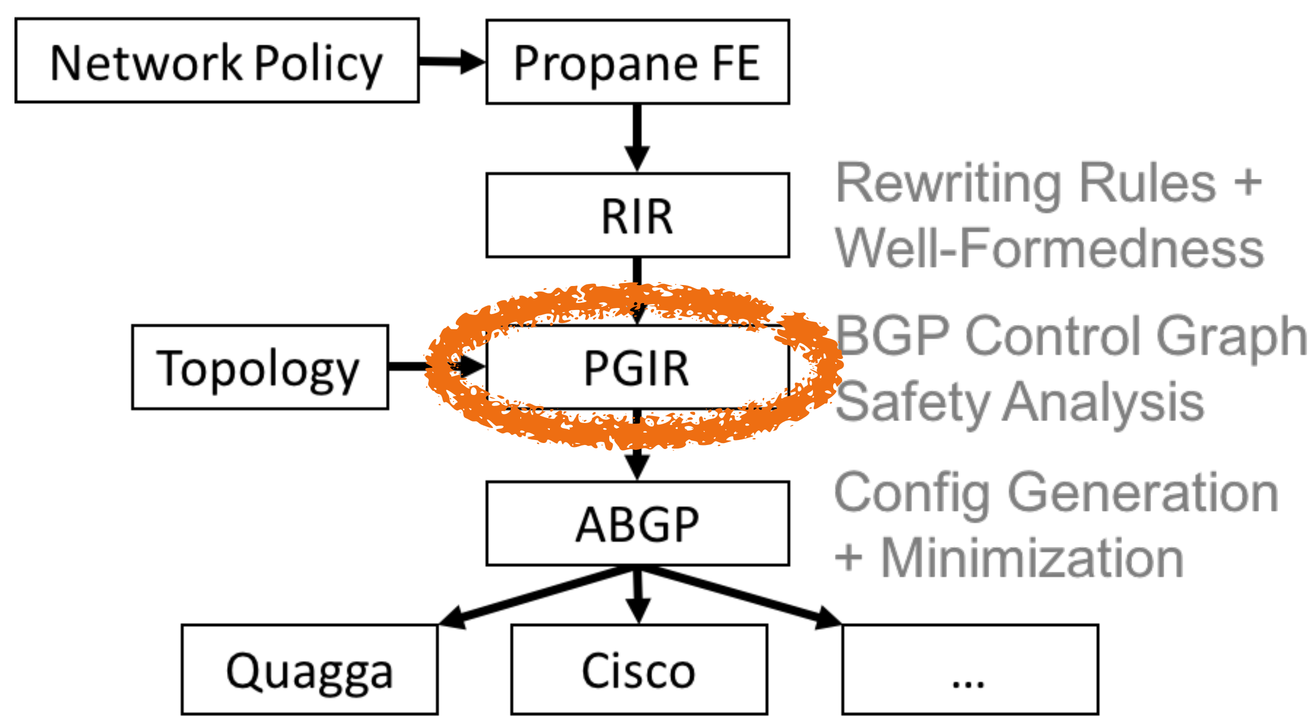
Beckett, Mahajan, Millstein, Padhye, Walker

SIGCOMM 2016

Capabilities

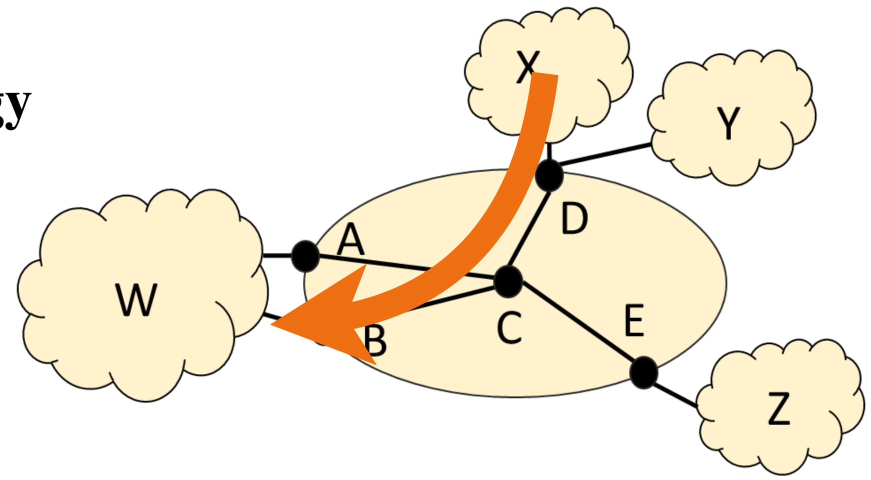
- Ability to express network-wide goals
- Domain-specific language to describe policy conveniently
- Compiled to distributed control plane configurations (BGP)

Internal design

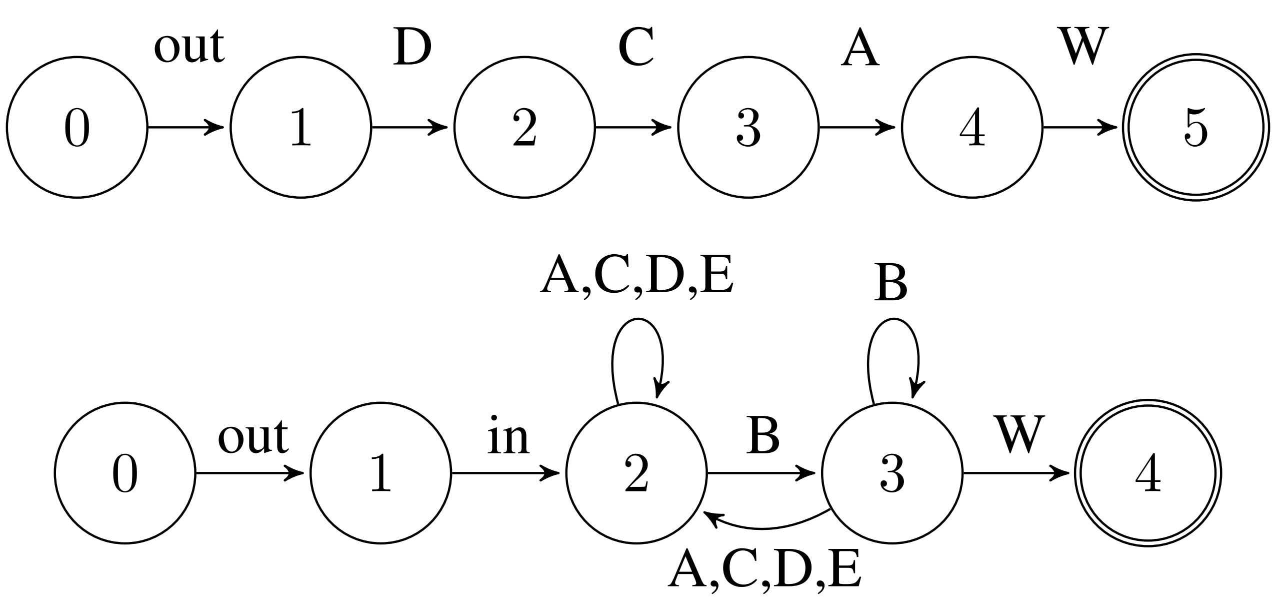


Propane product graphs

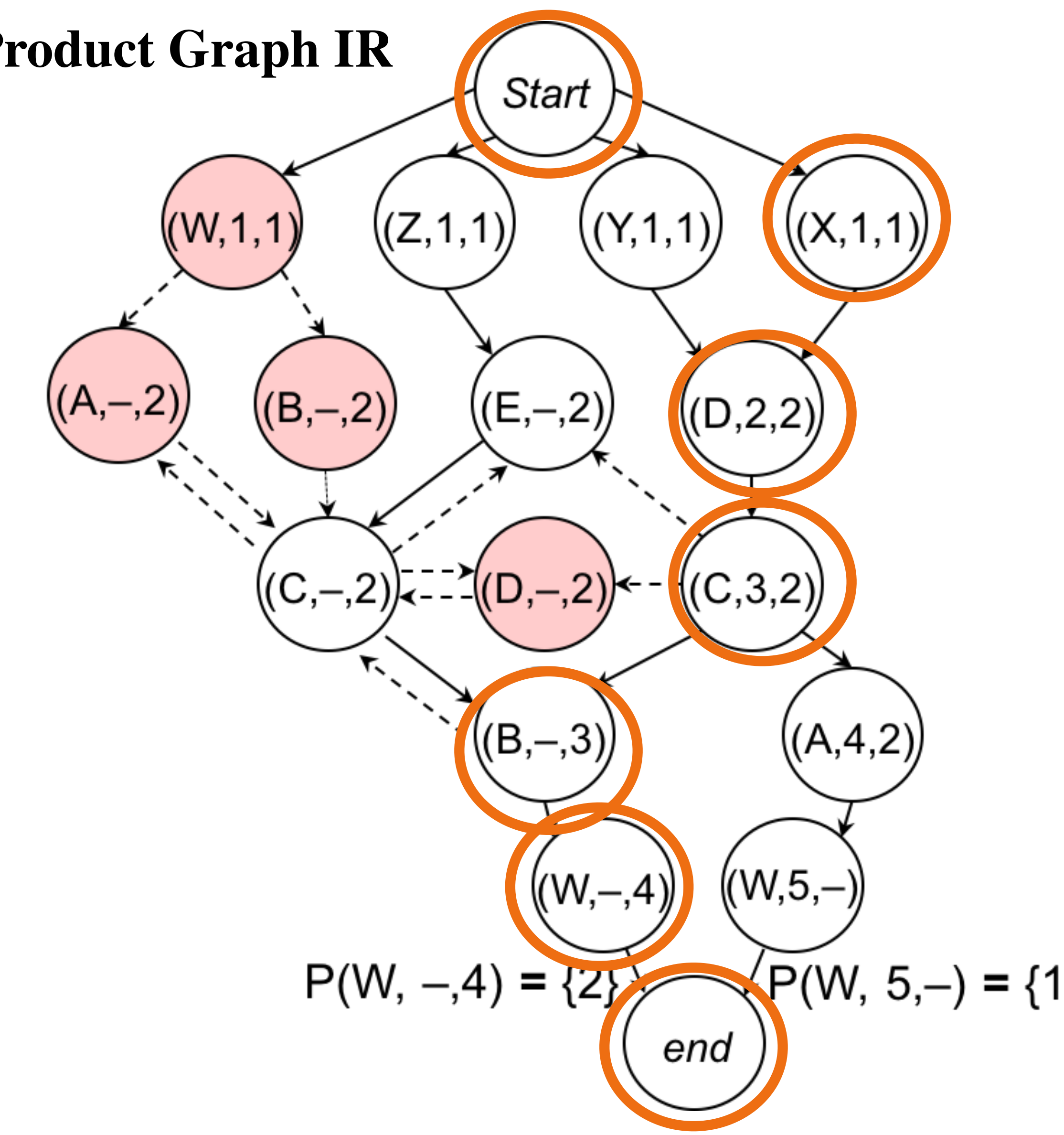
Topology



Policy Automata



Product Graph IR



[Figure from Propane paper]

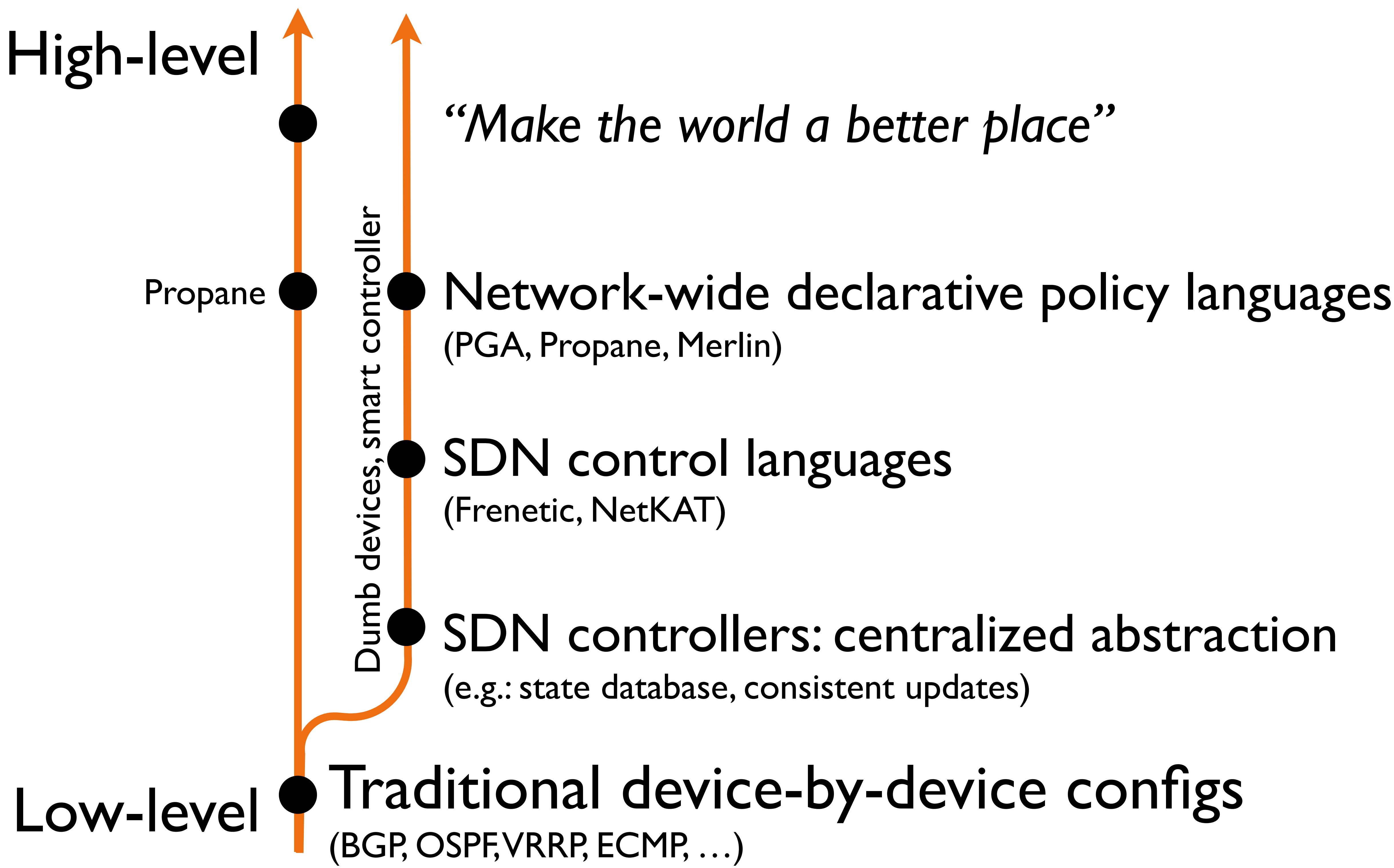
Propane: Discussion

How broad is the policy coverage?

Did they solve the configuration complexity problem?

Does the Propane system help detect errors?

Towards high-level abstractions



Beyond today's research

OpenConfig

- Industry effort to abstract vendor-specific details
- Analogous to Propane's ABGP

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."*

Beyond today's research

"Intent-based networking"

- Category of industry products aiming to help control and verify networks based on network-wide business goals

High-level abstractions for flexible programmability a grand challenge for networking

- What can people use? Who is doing the programming?
- Does different hardware change the abstraction?
- Can we carve out killer apps?