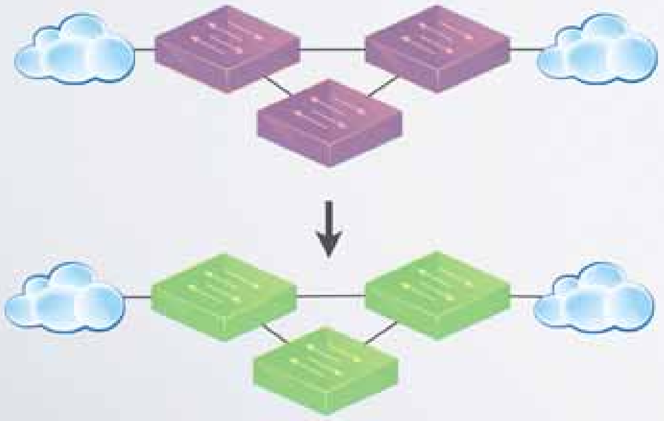
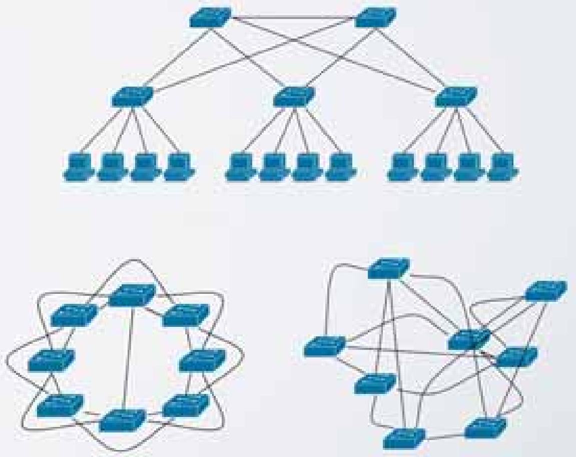
Abstractions for Network Update

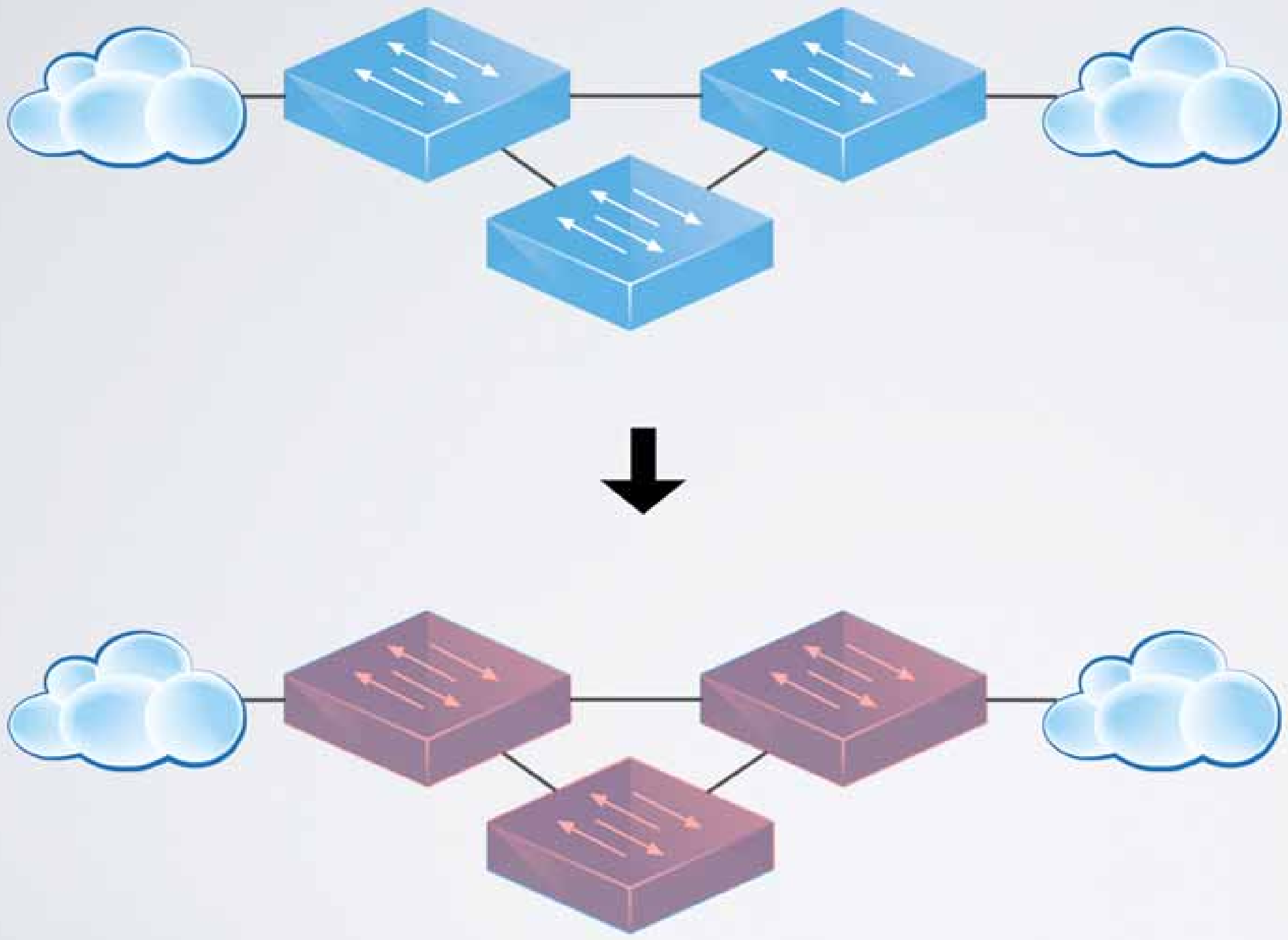
Nate Foster
Mark Reitblatt

Jen Rexford
Cole Schlesinger
Dave Walker





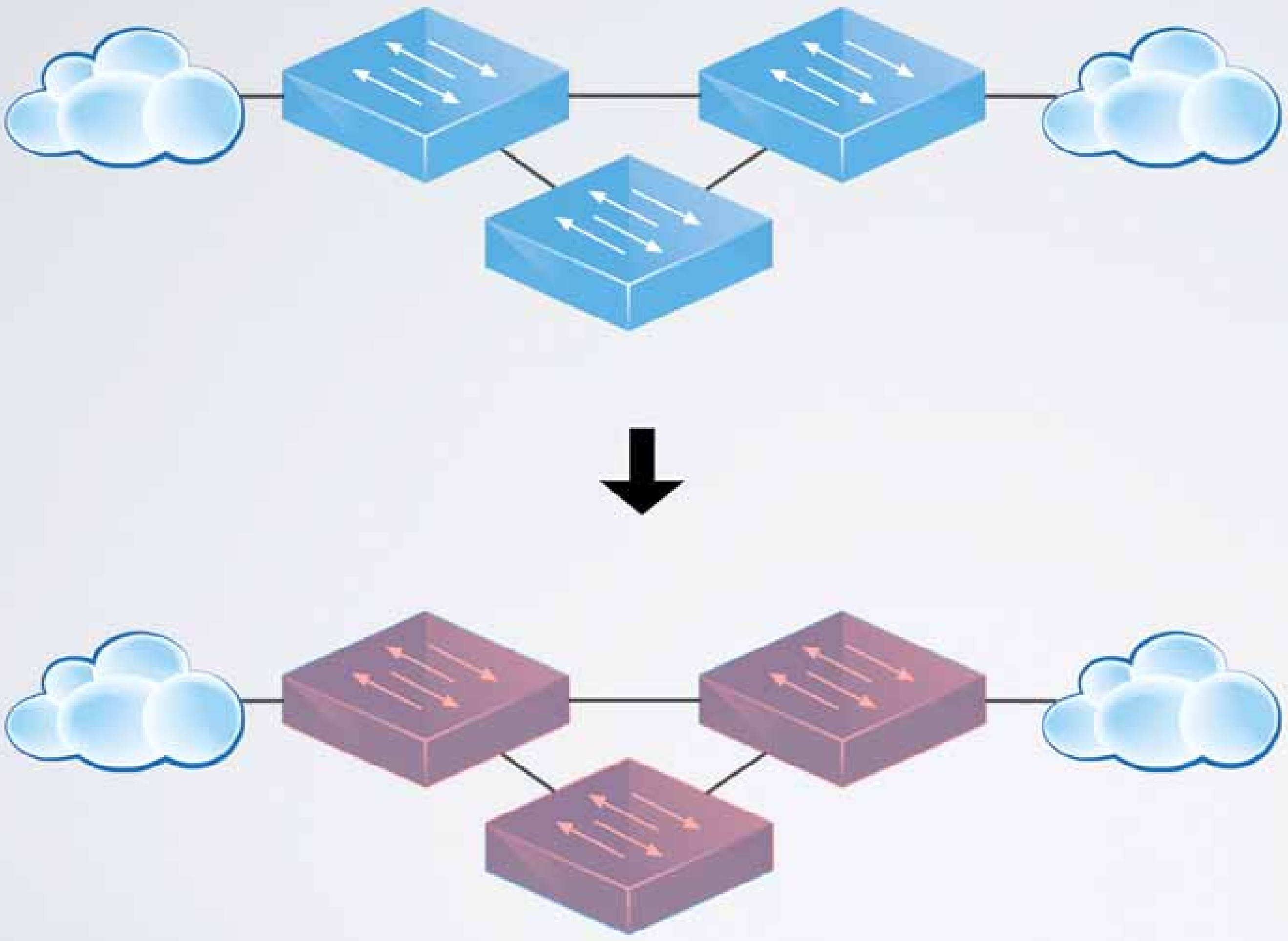
Updates Happen



Network Updates

- Maintenance
- Failures
- ACL Updates

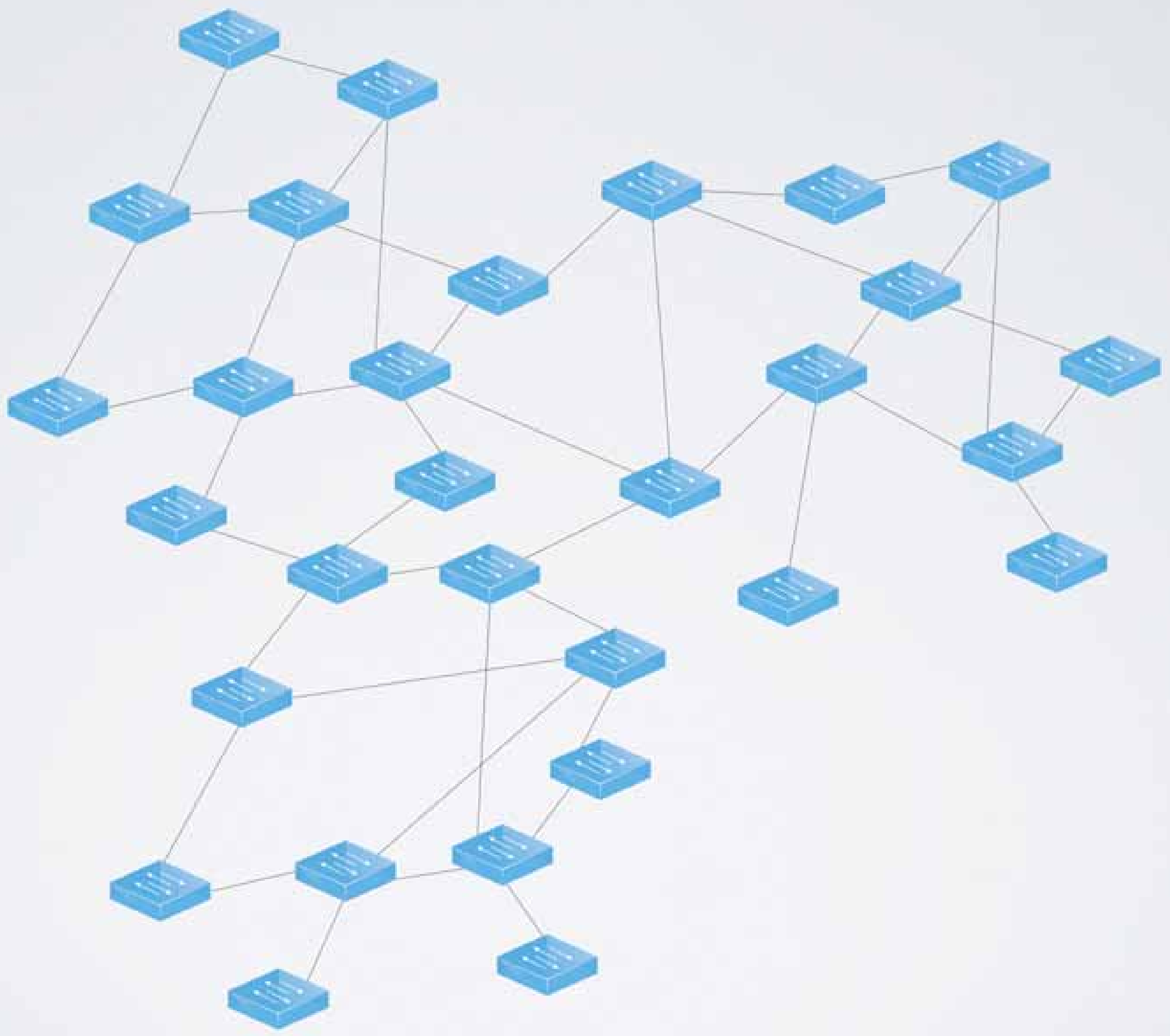
Updates Happen



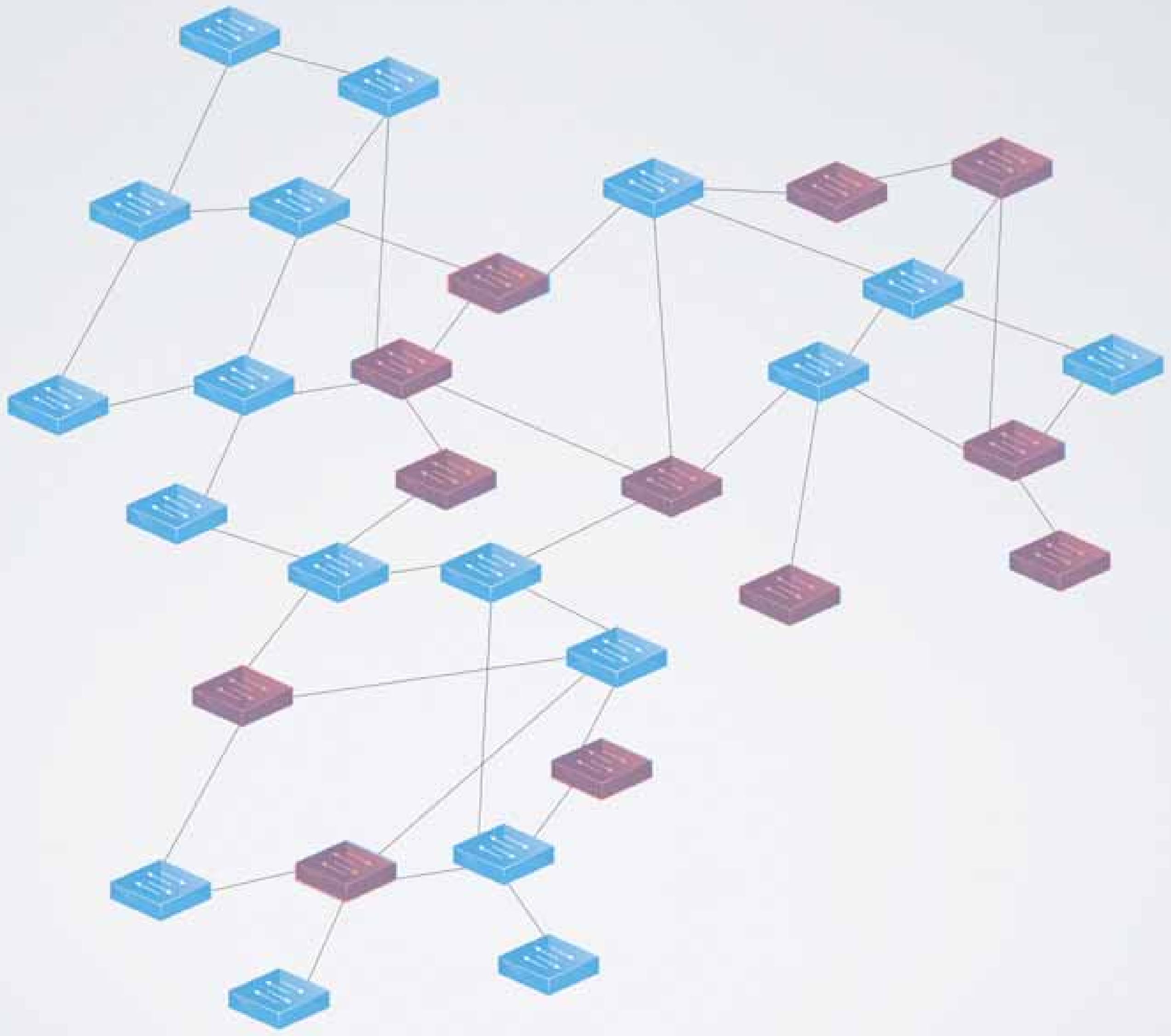
Network Updates
- Maintenance
- Failures
- ACL Updates

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

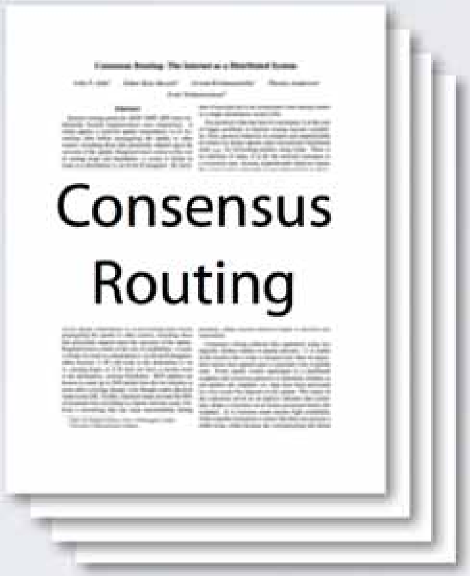
Network Updates Are Hard



Network Updates Are Hard



Prior Work

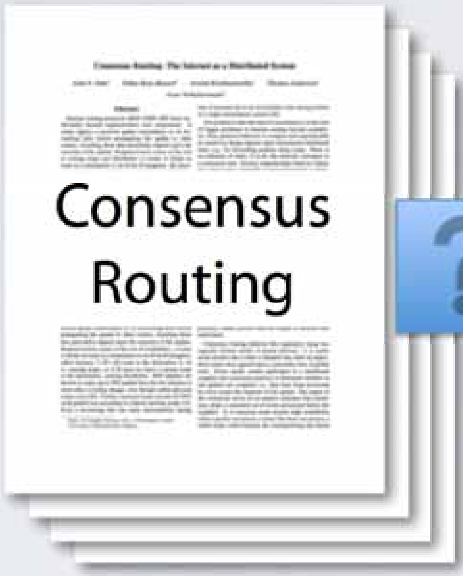

Consensus Routing: The Internet as a Distributed System
Consensus
Routing

R-BGP: Staying Connected In a Connected World
Reliable BGP

Graceful Network State Migrations
Graceful
Migration

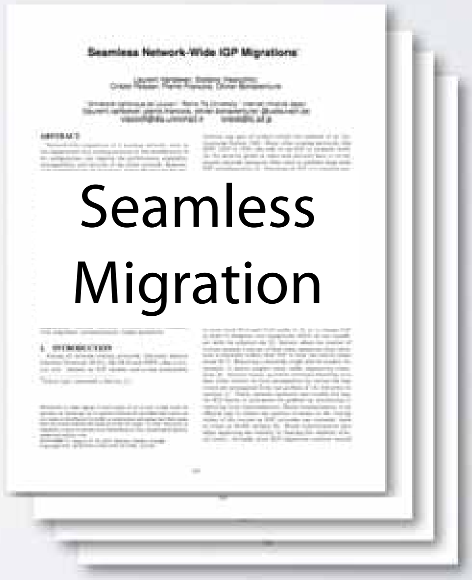
Seamless Network-Wide IGP Migrations
Seamless
Migration

Prior Work


Consensus Routing: The Internet as a Distributed System
Consensus Routing

R-BGP: Staying Connected In a Connected World
Reliable BGP

Graceful Network State Migrations
Graceful Migration


Seamless Network-Wide IGP Migrations
Seamless Migration

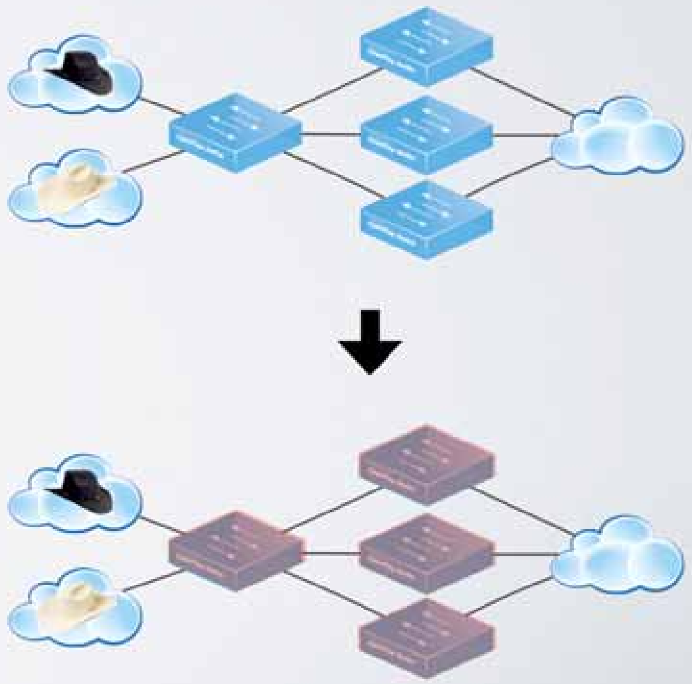
Network Update Abstractions

Goal

- Tools for whole network update

Our Approach

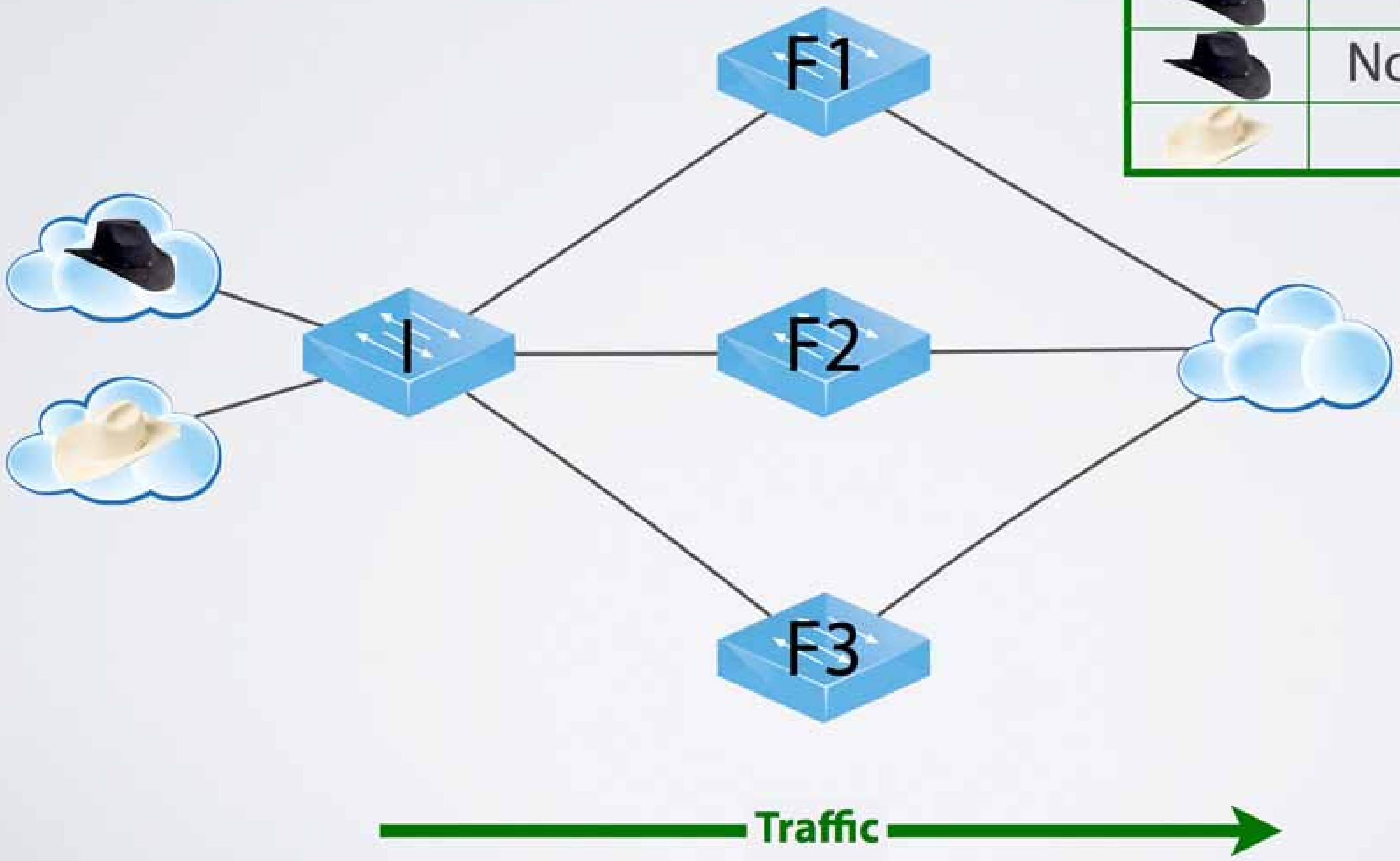
- Develop update abstractions
- Endow them with strong semantics
- Engineer efficient implementations



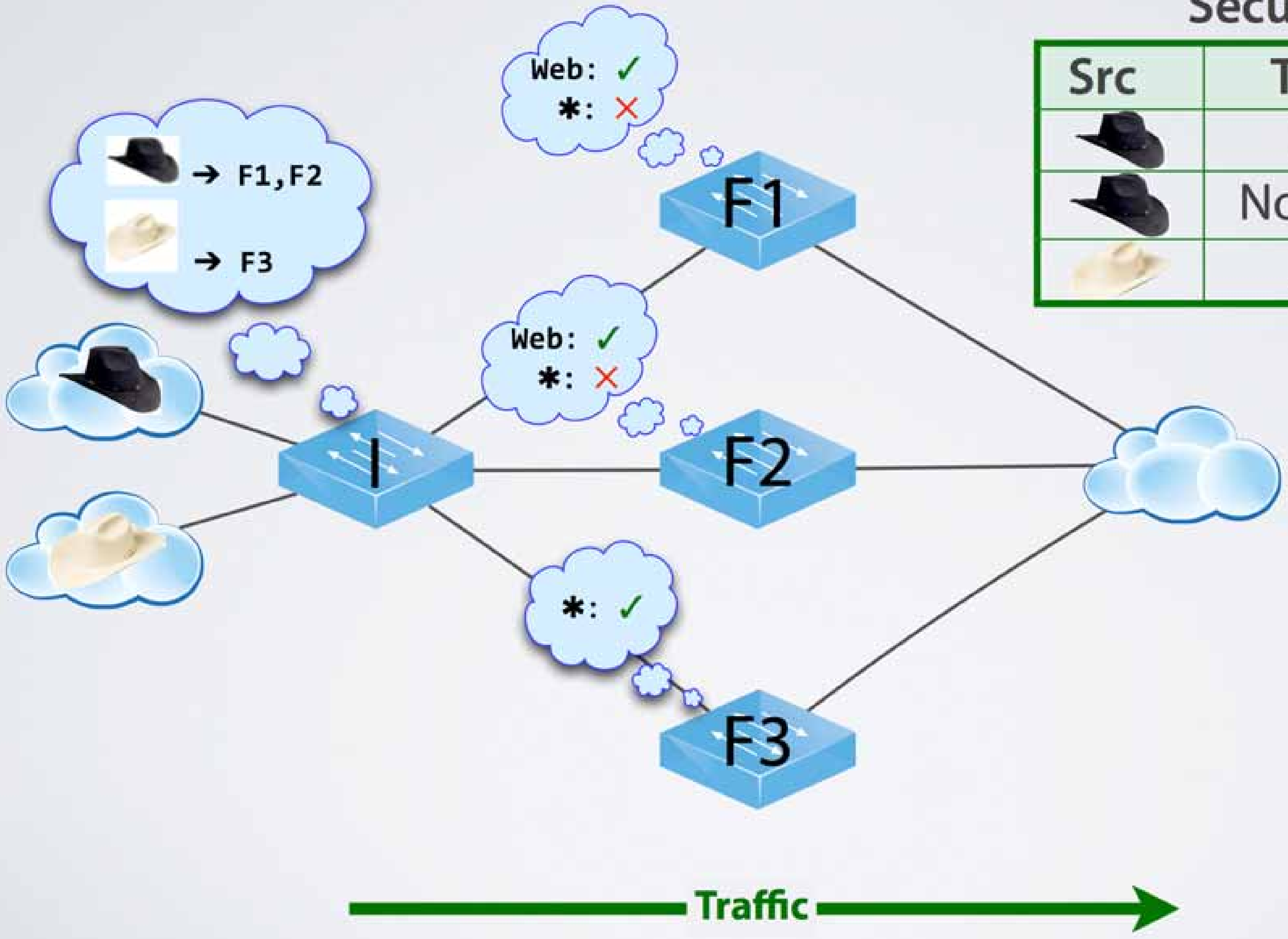
Example: Distributed Access Control

Security Policy

Src	Traffic	Action
	Web	Allow
	Non-web	Drop
	Any	Allow



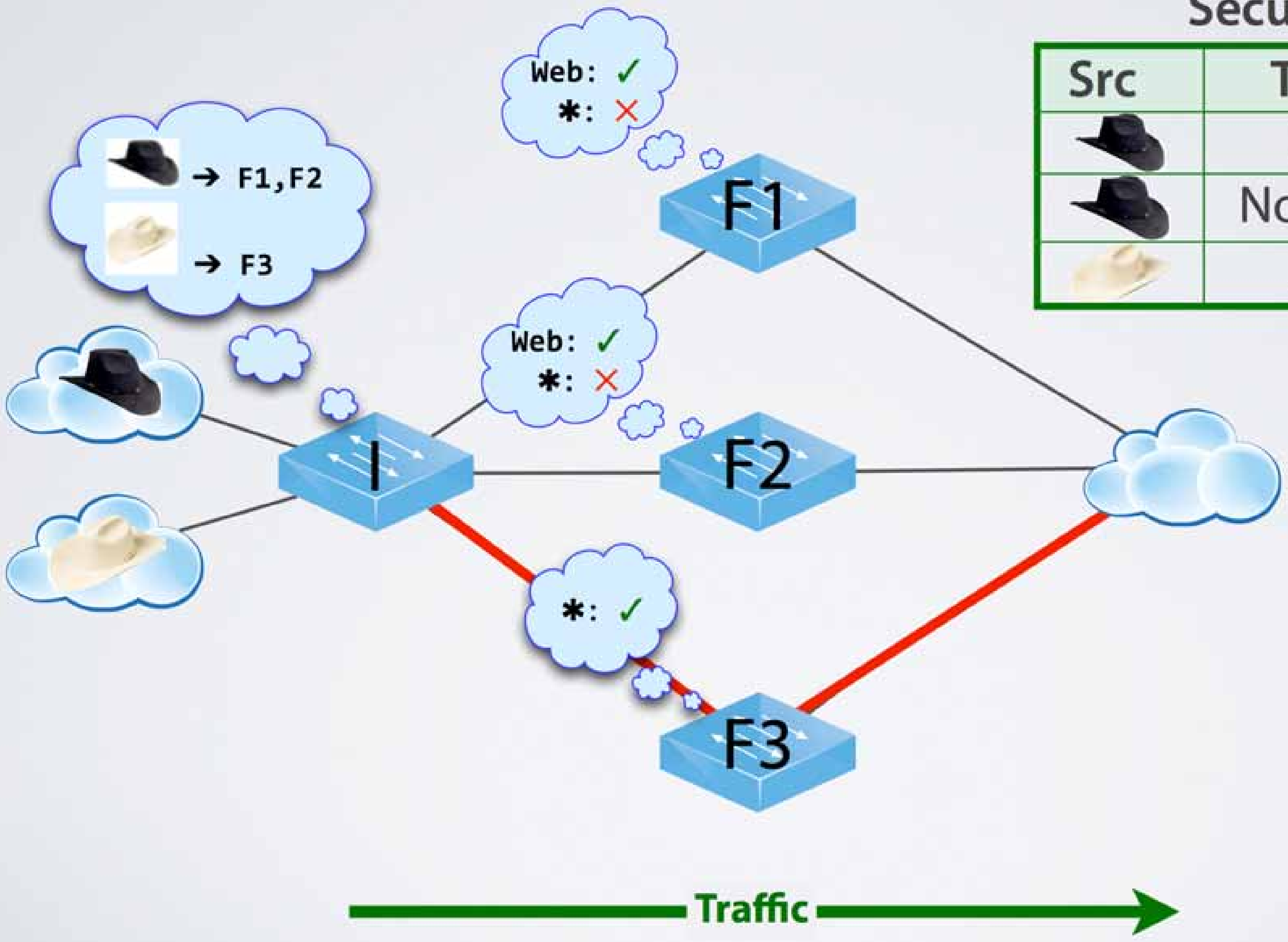
Example: Distributed Access Control



Security Policy

Src	Traffic	Action
🎩	Web	Allow
🎩	Non-web	Drop
🤠	Any	Allow

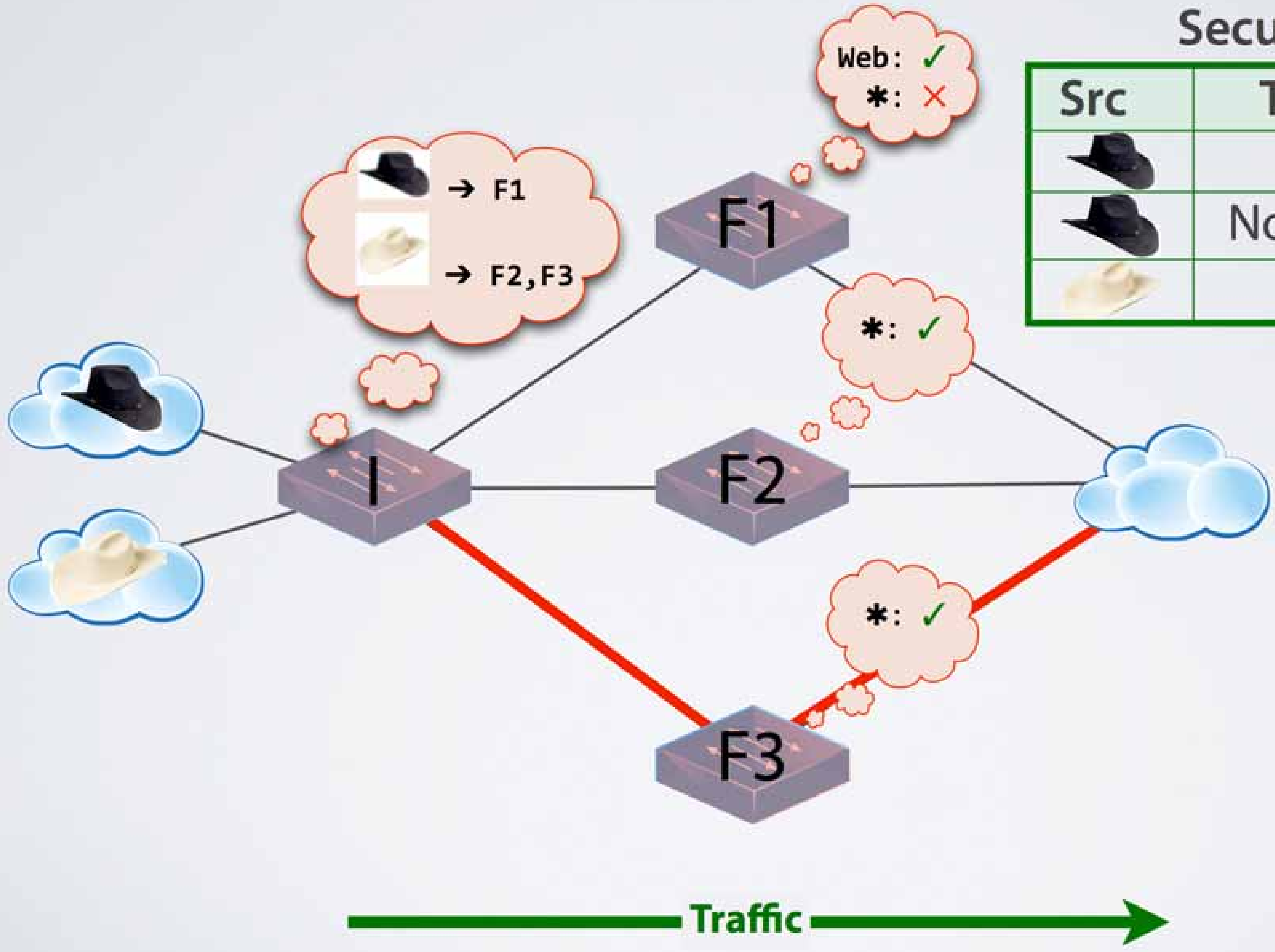
Example: Distributed Access Control



Security Policy

Src	Traffic	Action
	Web	Allow
	Non-web	Drop
	Any	Allow

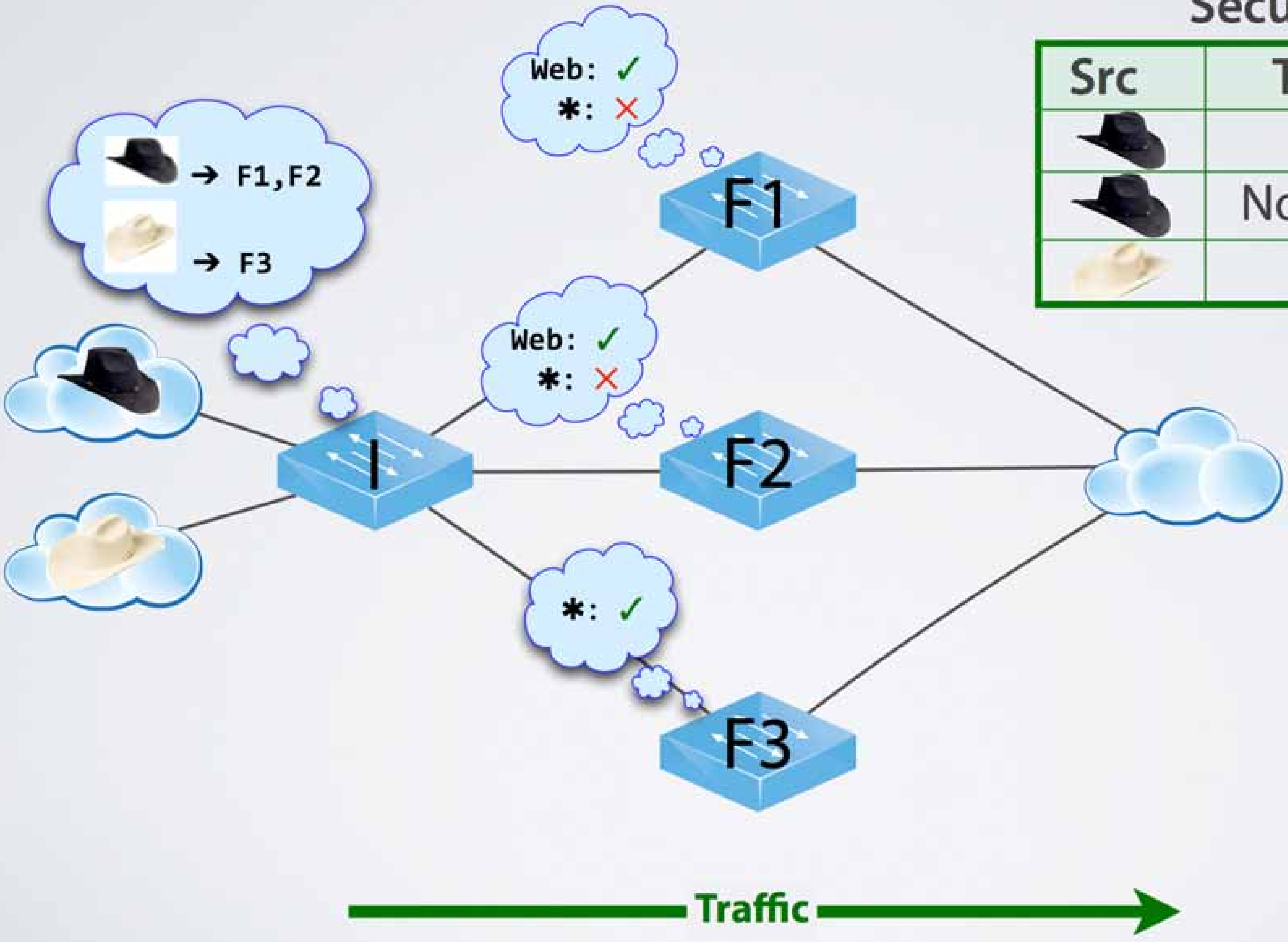
Example: Distributed Access Control



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

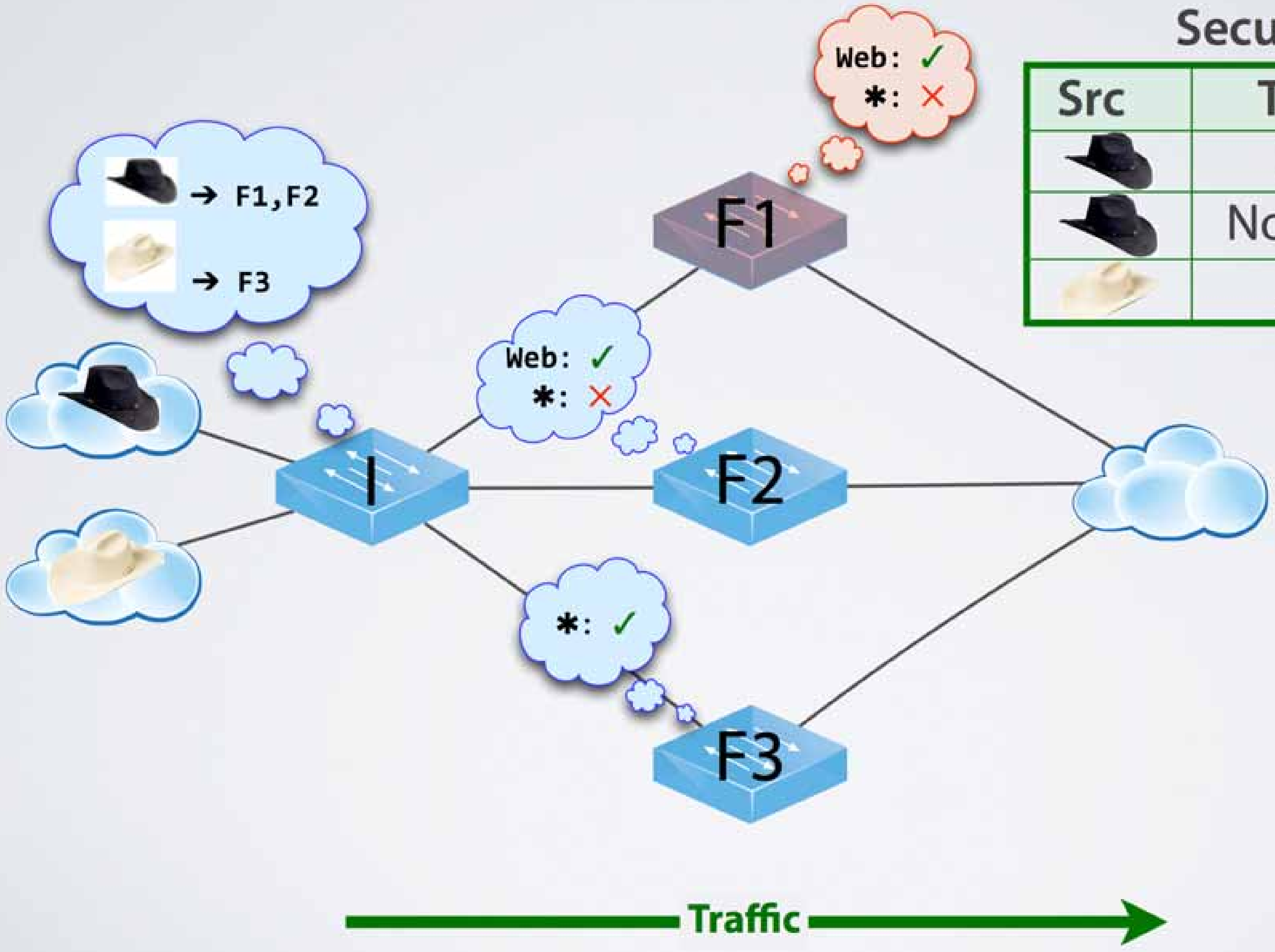
F1

F2

F3

I

Naive Update



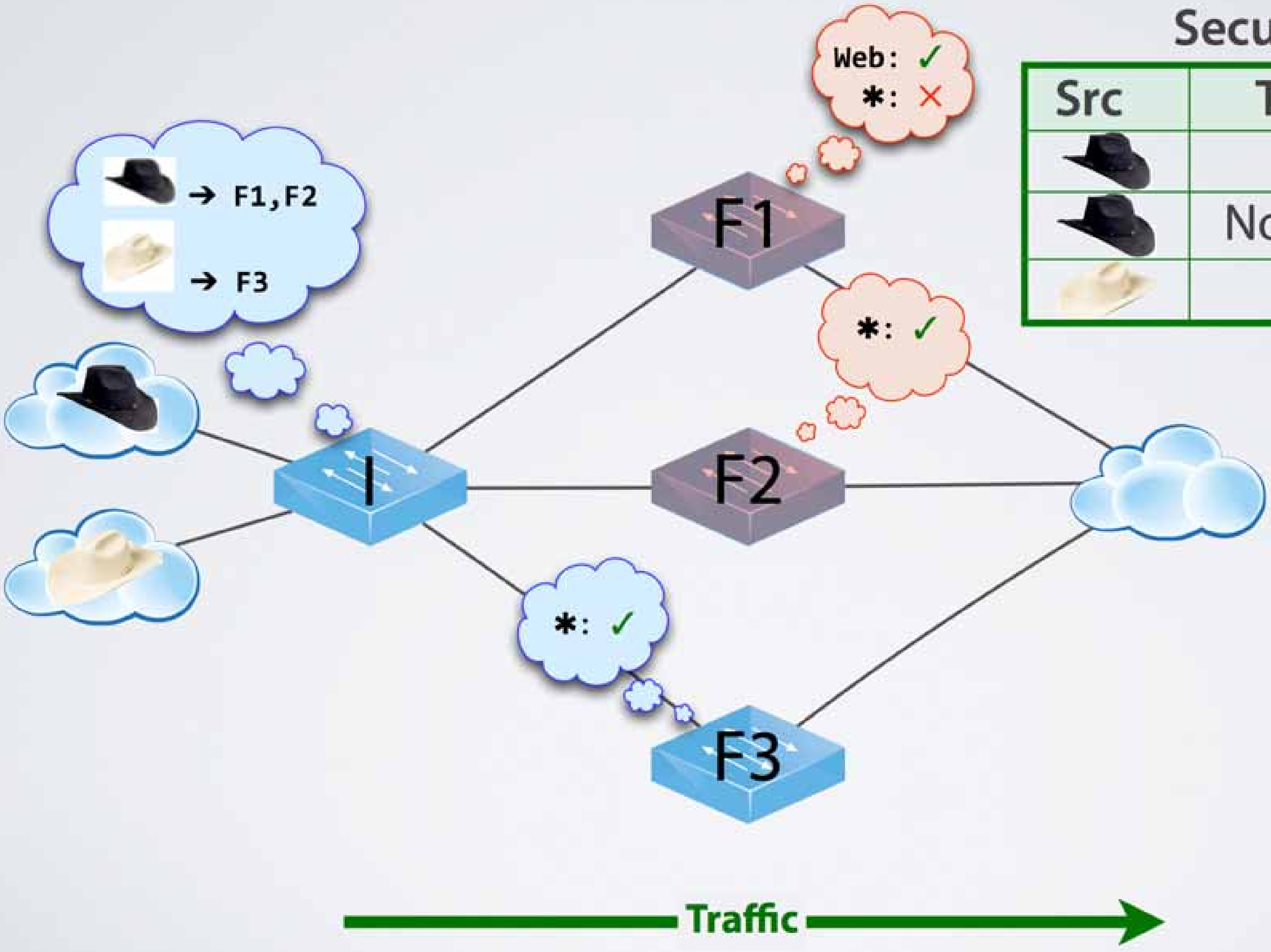
Security Policy

Src	Traffic	Action
	Web	Allow
	Non-web	Drop
	Any	Allow

Order

F2
F3
I

Naive Update



Security Policy

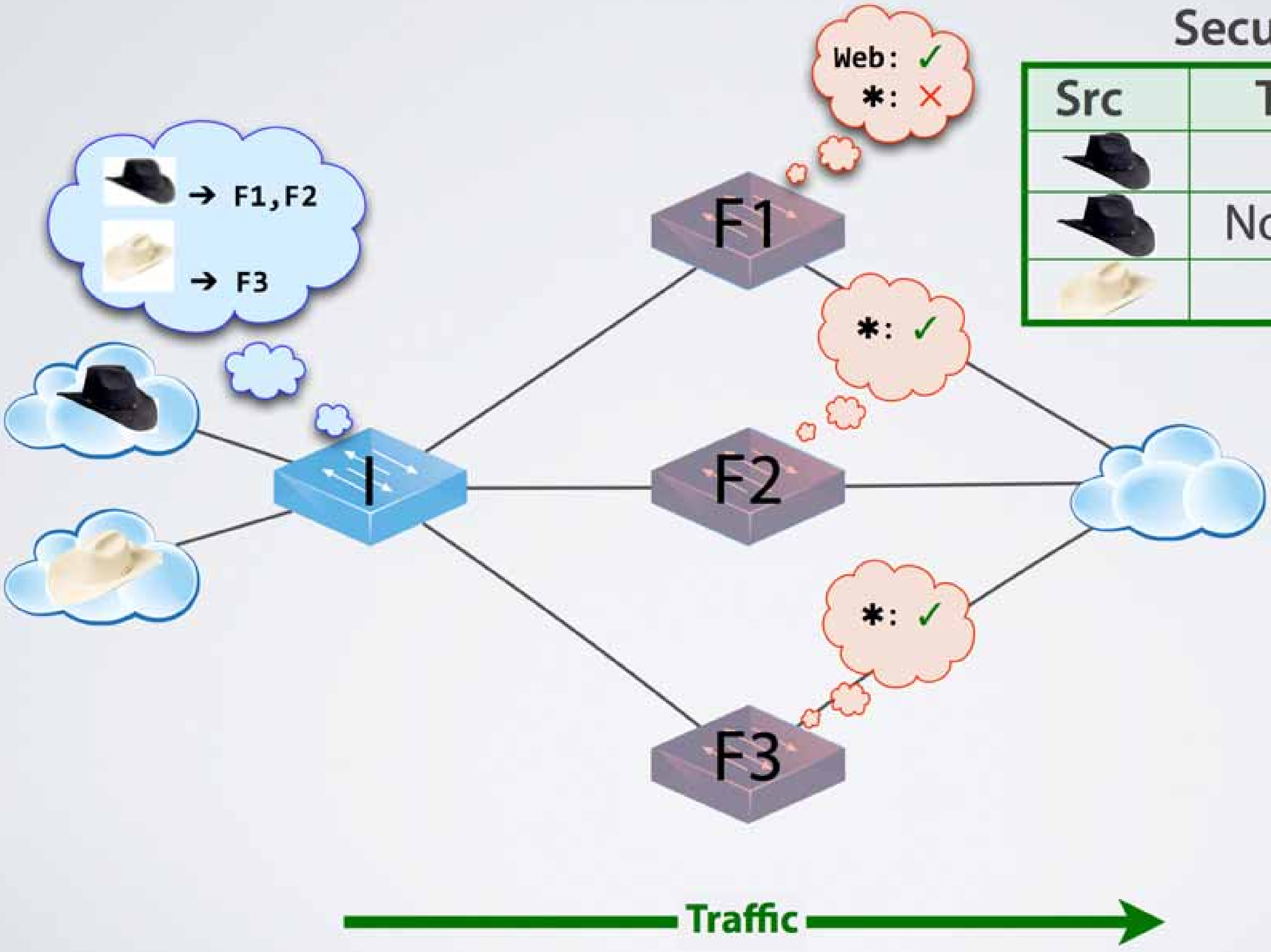
Src	Traffic	Action
🎩	Web	Allow
🎩	Non-web	Drop
🤠	Any	Allow

Order

F3

I

Naive Update



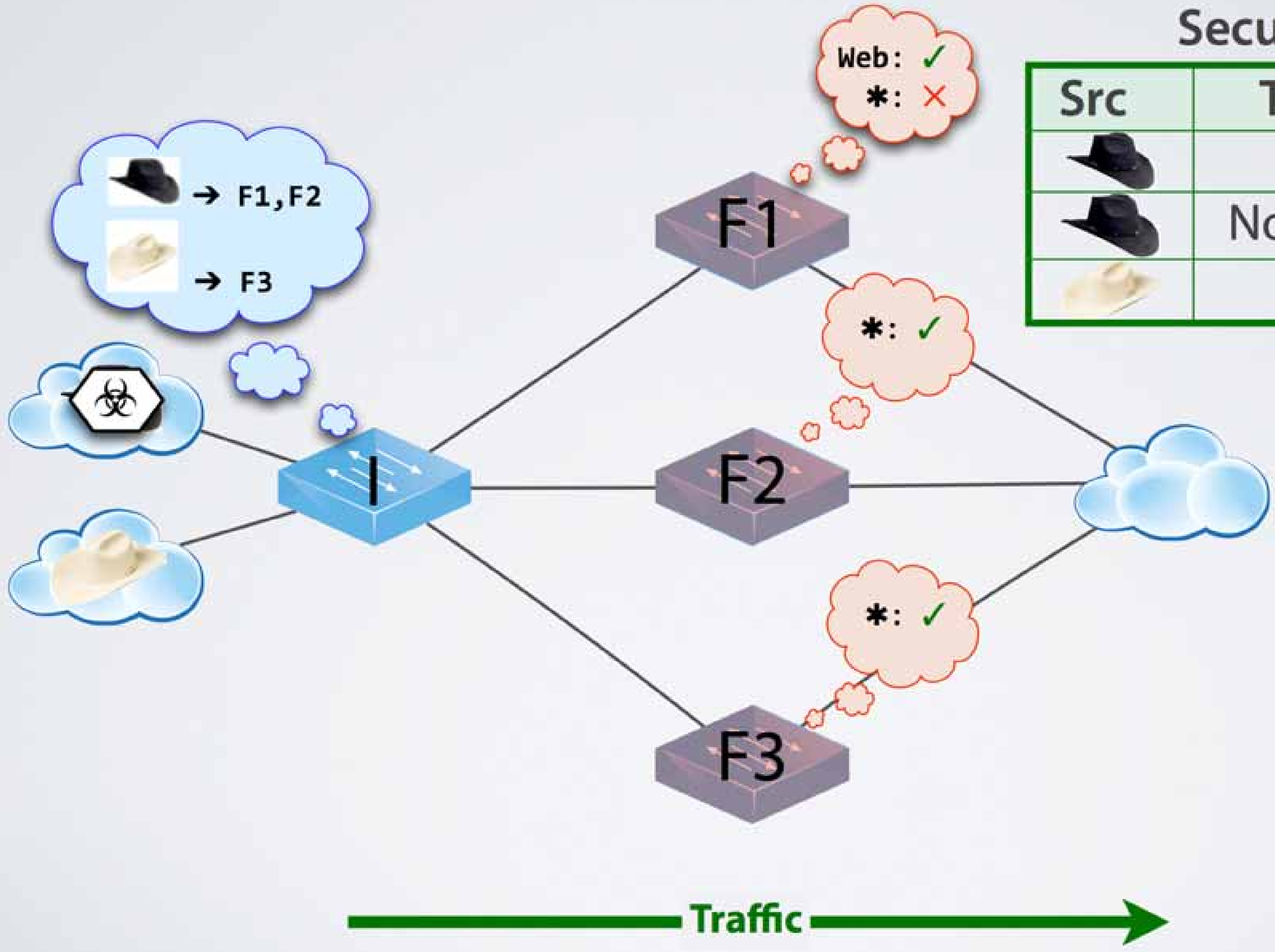
Security Policy

Src	Traffic	Action
	Web	Allow
	Non-web	Drop
	Any	Allow

Order

I

Naive Update



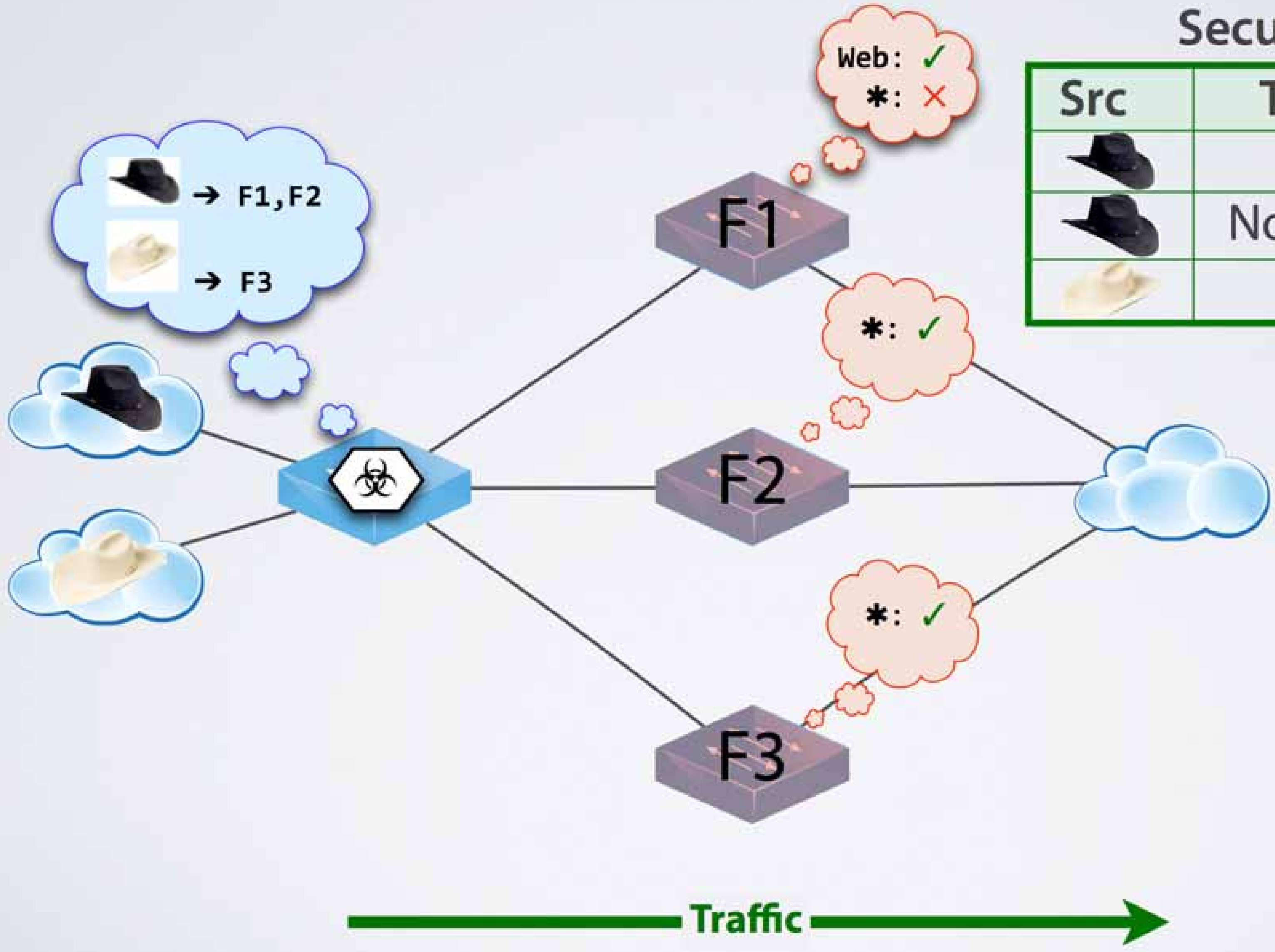
Security Policy

Src	Traffic	Action
🎩	Web	Allow
🎩	Non-web	Drop
🎩	Any	Allow

Order

I

Naive Update



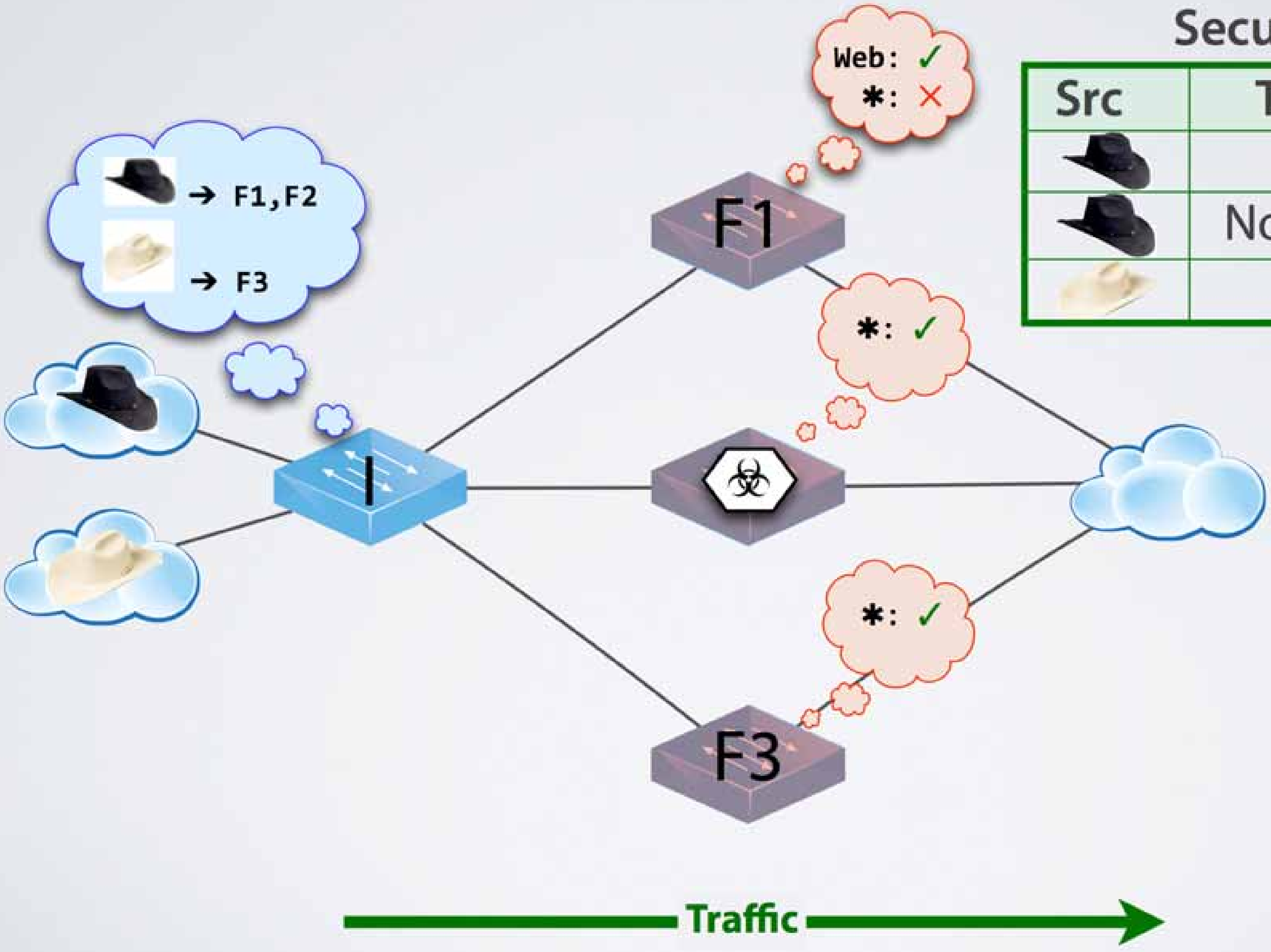
Security Policy

Src	Traffic	Action
🎩	Web	Allow
🎩	Non-web	Drop
🎩	Any	Allow

Order

I

Naive Update



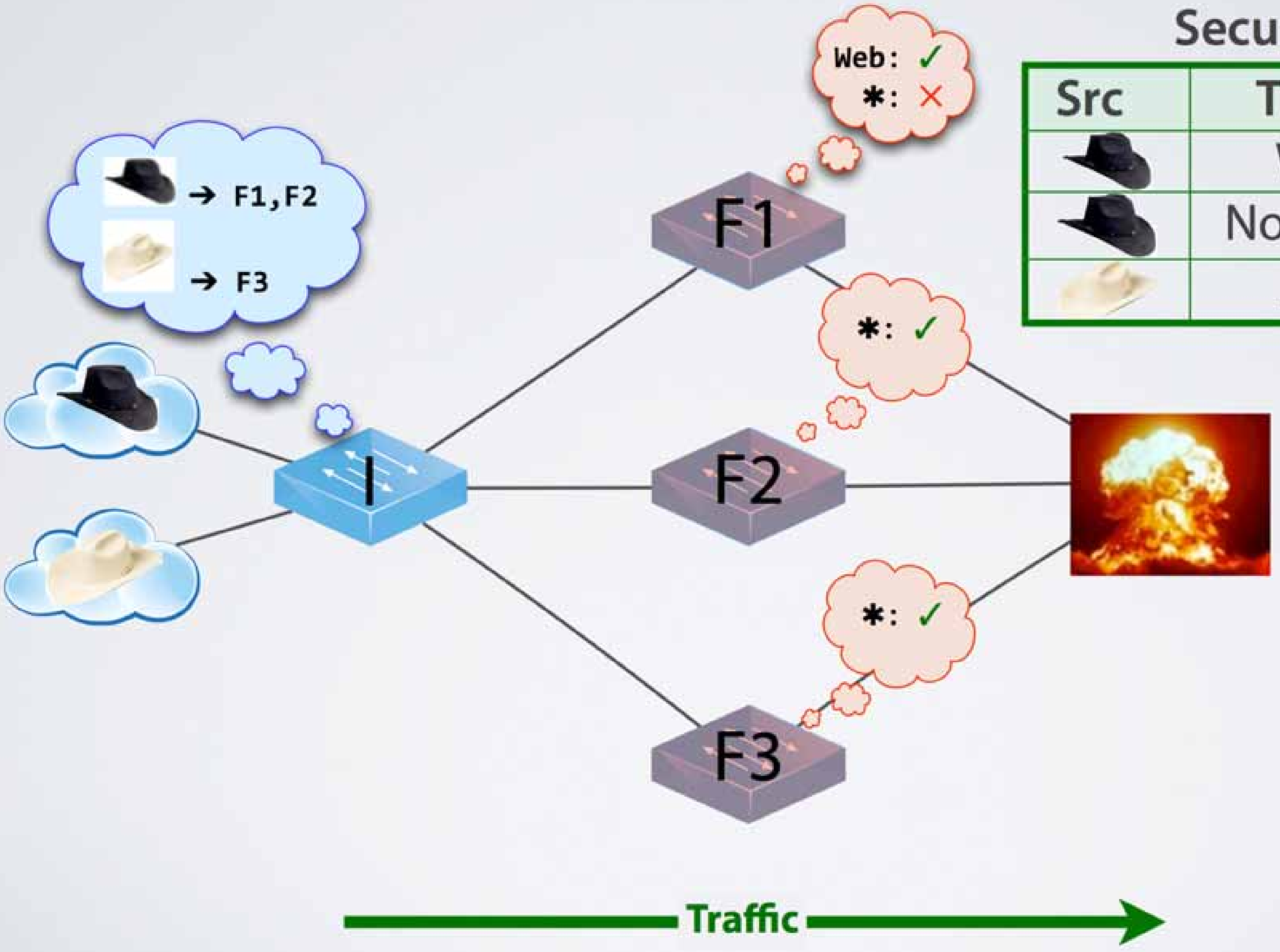
Security Policy

Src	Traffic	Action
●	Web	Allow
●	Non-web	Drop
○	Any	Allow

Order

I

Naive Update

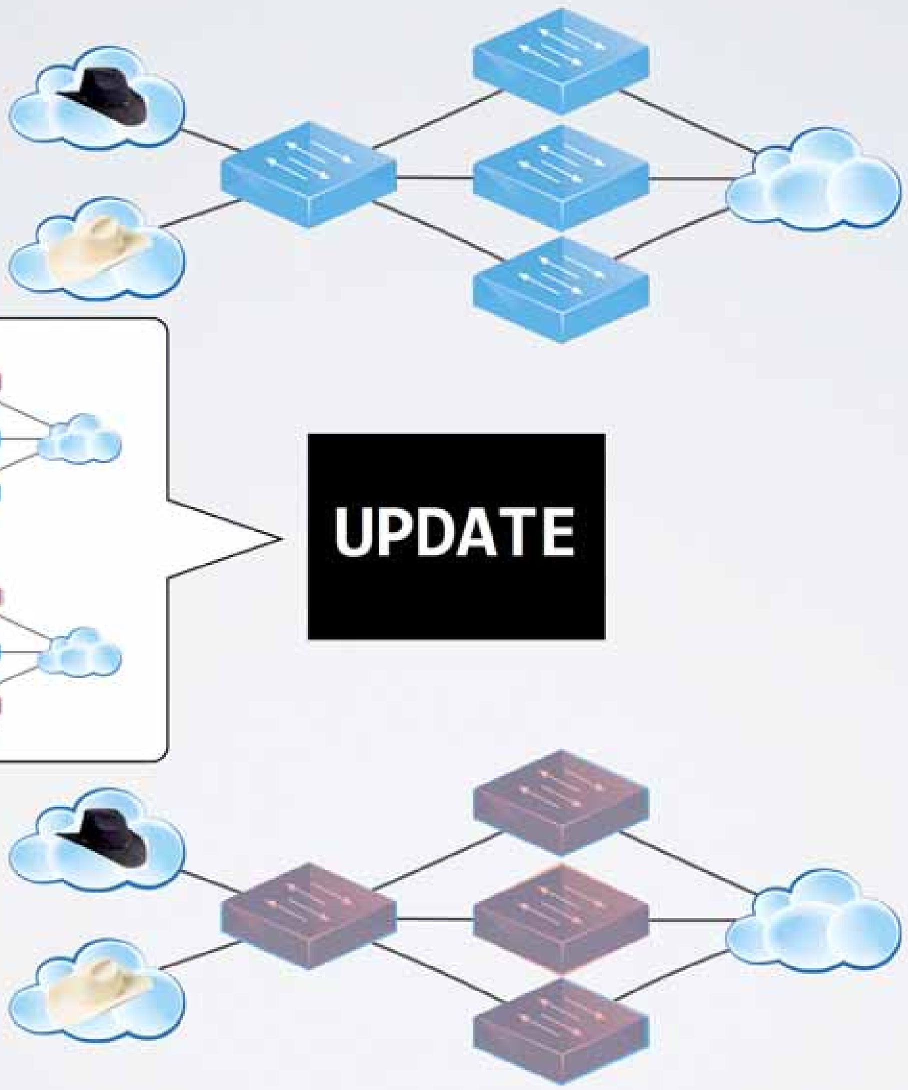


Security Policy

Src	Traffic	Action
	Web	Allow
	Non-web	Drop
	Any	Allow

Order

Use an Abstraction!



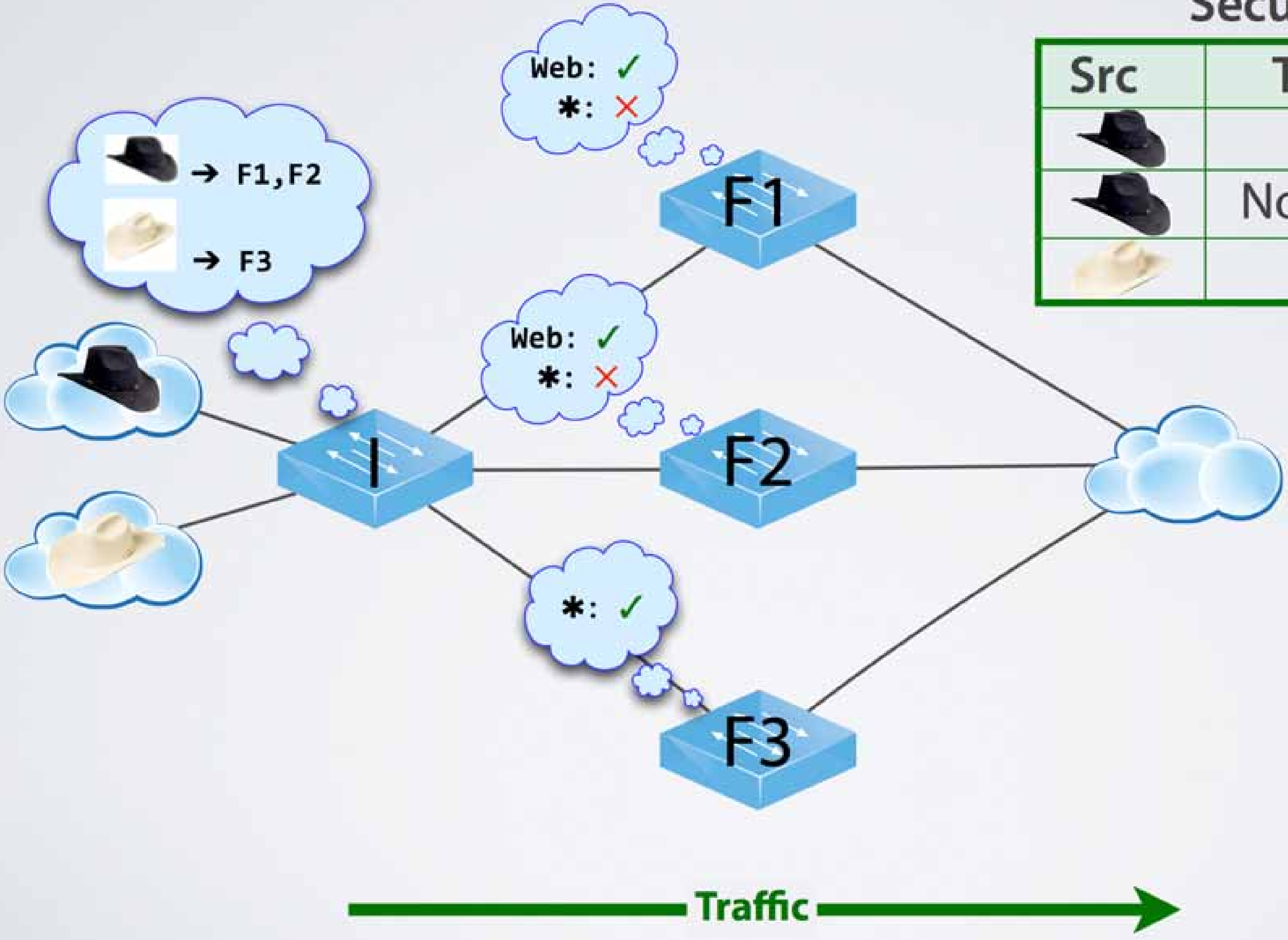
Security Policy

✓

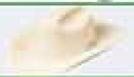
✓

✓

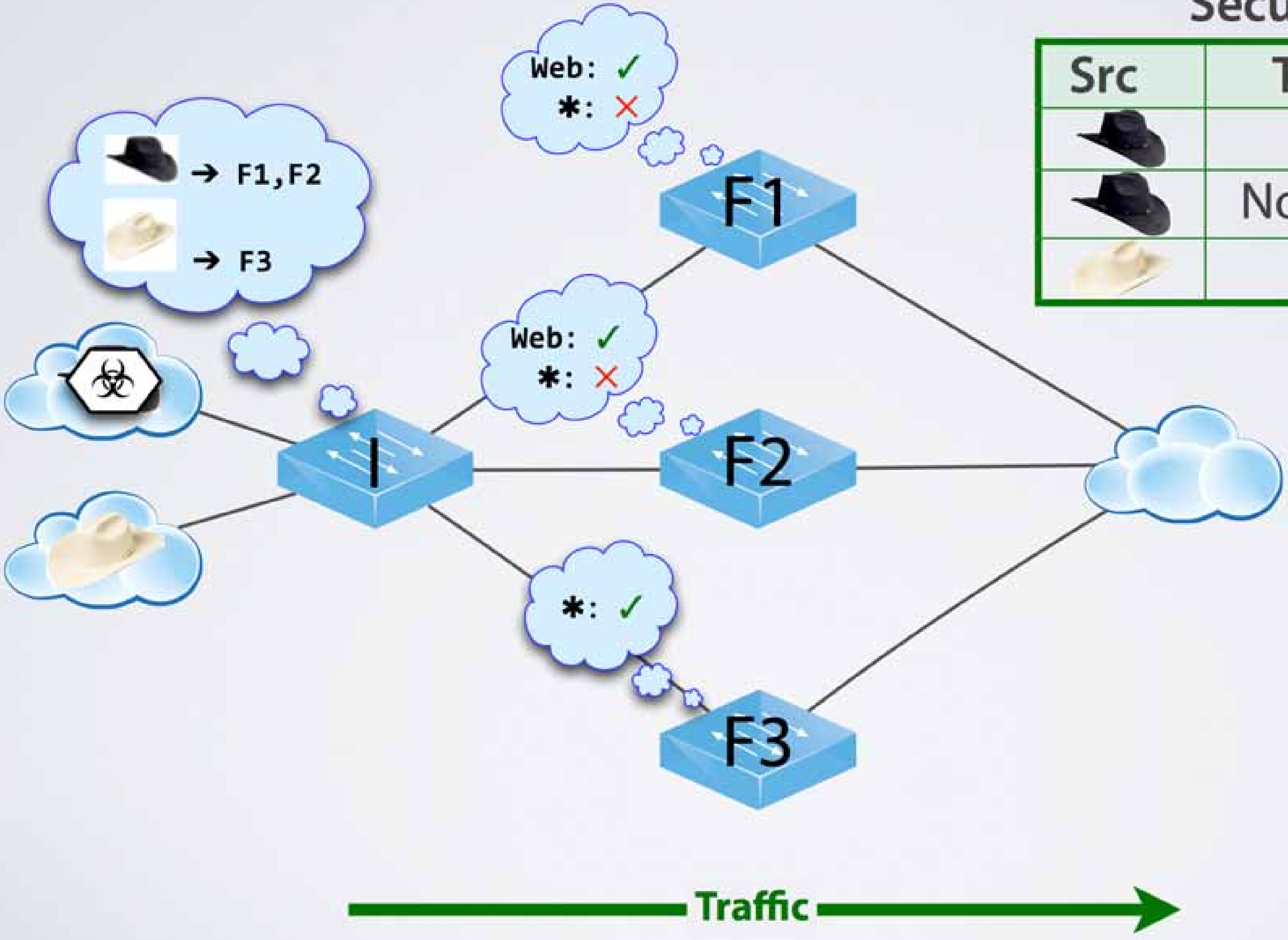
Atomic Update?



Security Policy

Src	Traffic	Action
	Web	Allow
	Non-web	Drop
	Any	Allow

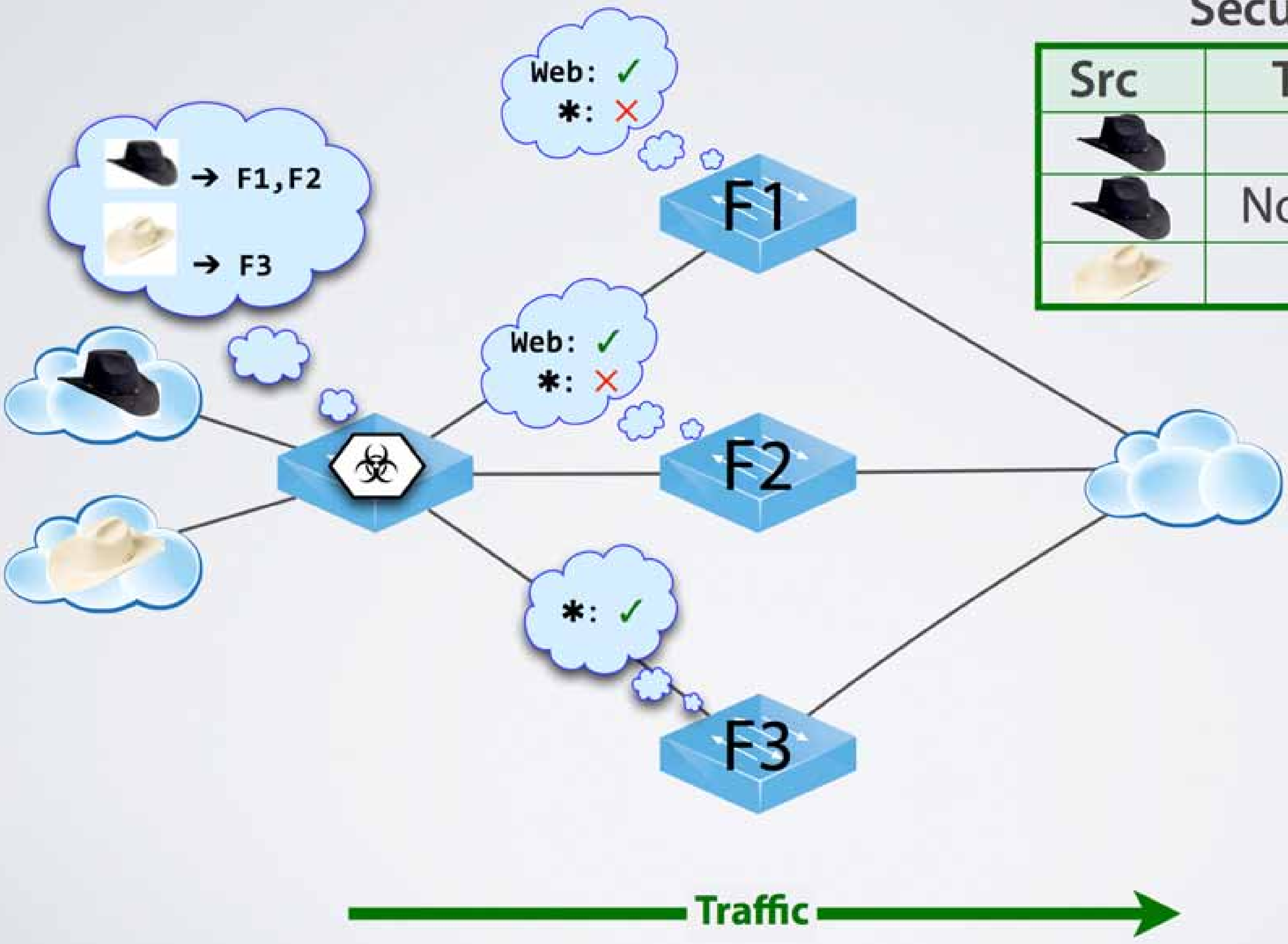
Atomic Update?



Security Policy

Src	Traffic	Action
	Web	Allow
	Non-web	Drop
	Any	Allow

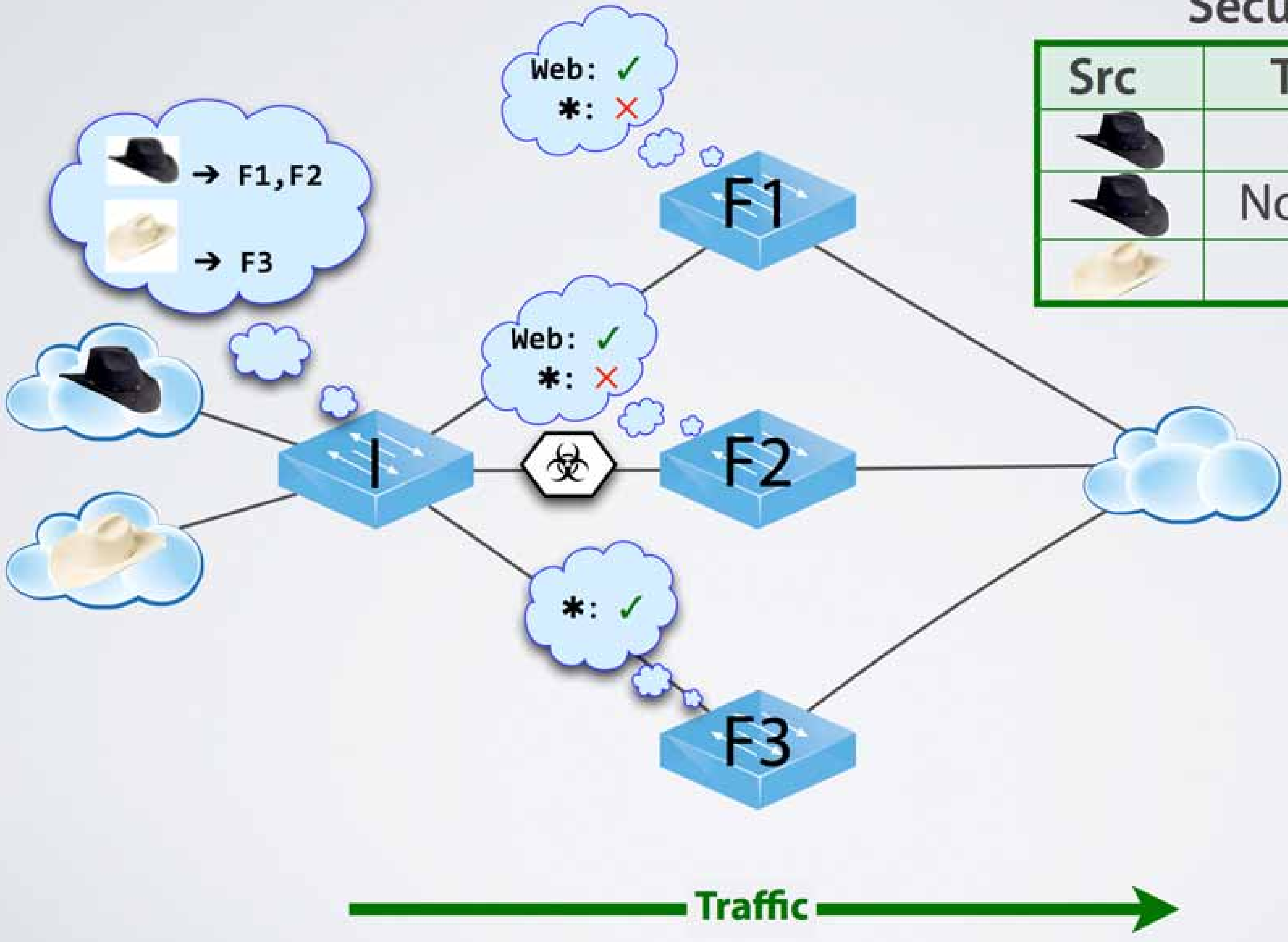
Atomic Update?



Security Policy

Src	Traffic	Action
■	Web	Allow
■	Non-web	Drop
□	Any	Allow

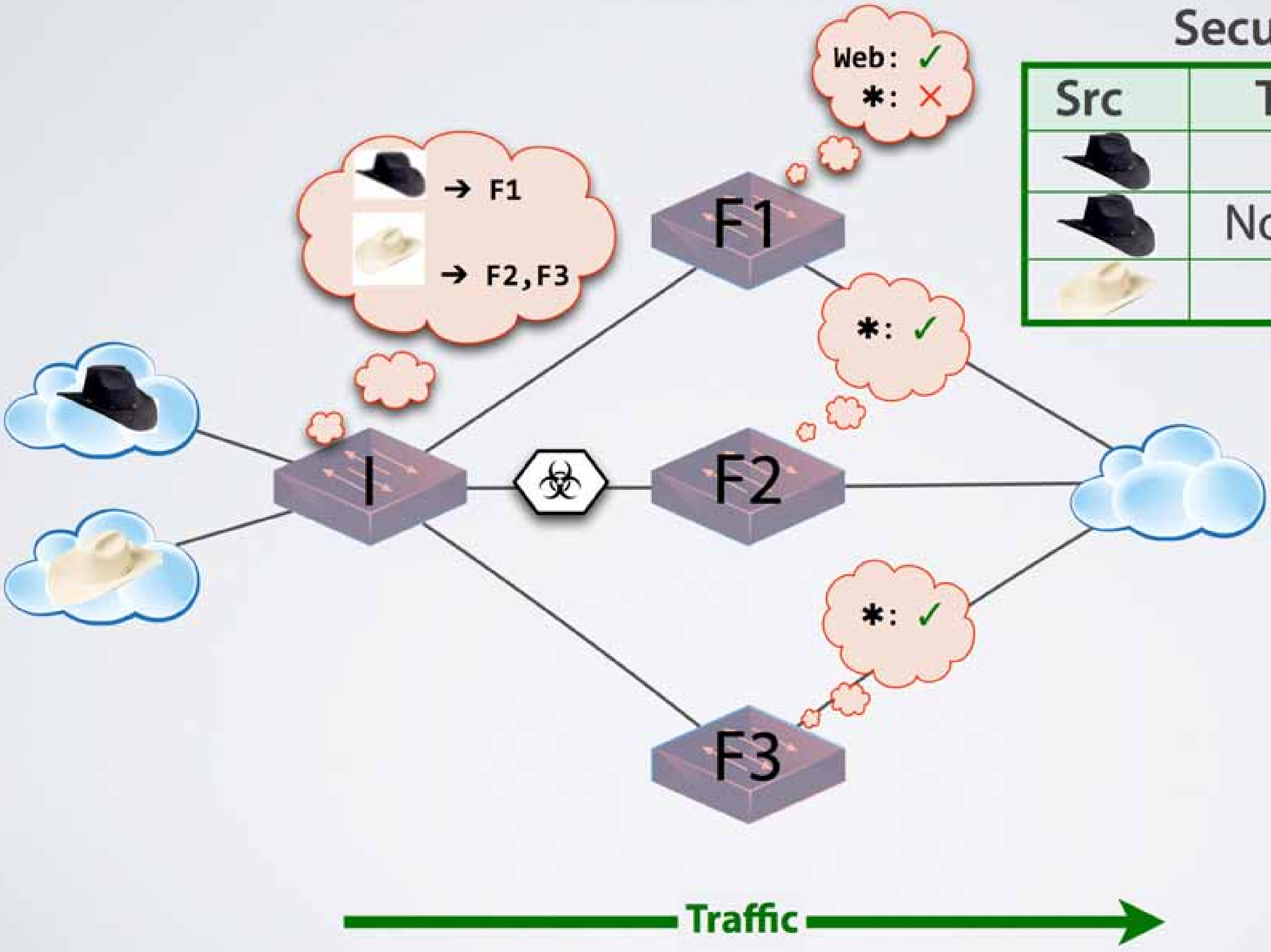
Atomic Update?



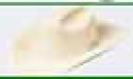
Security Policy

Src	Traffic	Action
●	Web	Allow
●	Non-web	Drop
○	Any	Allow

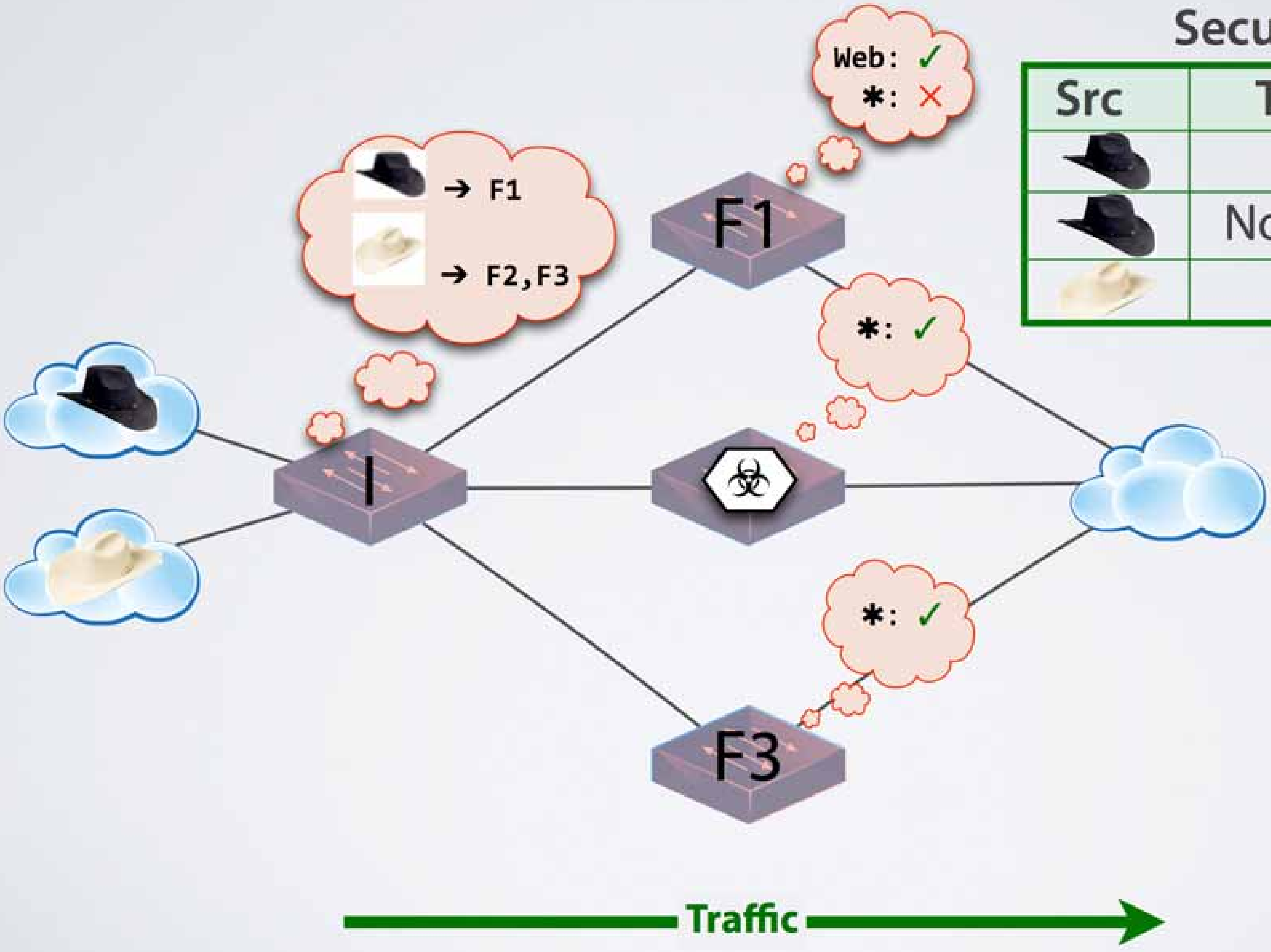
Atomic Update?



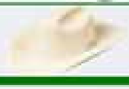
Security Policy

Src	Traffic	Action
	Web	Allow
	Non-web	Drop
	Any	Allow

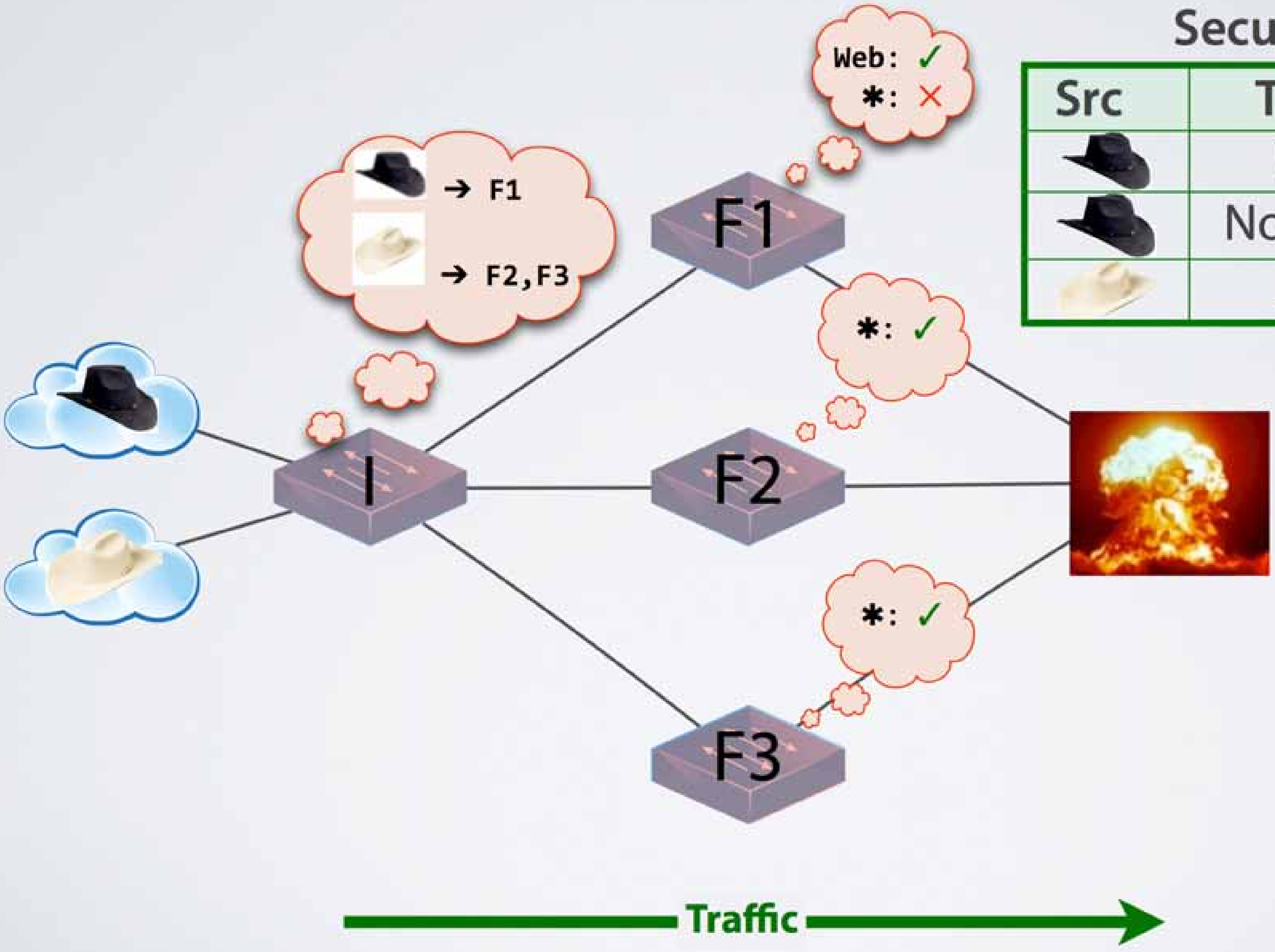
Atomic Update?



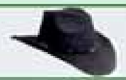
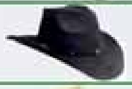
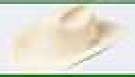
Security Policy

Src	Traffic	Action
	Web	Allow
	Non-web	Drop
	Any	Allow

Atomic Update?



Security Policy

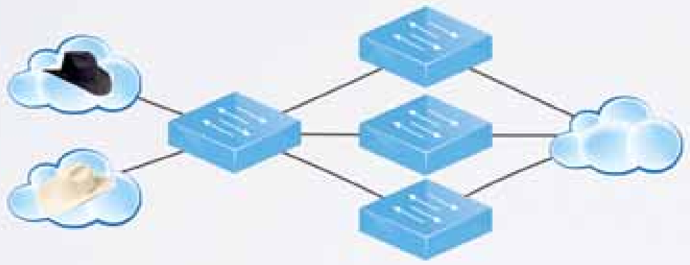
Src	Traffic	Action
	Web	Allow
	Non-web	Drop
	Any	Allow

Per-Packet Consistent Updates

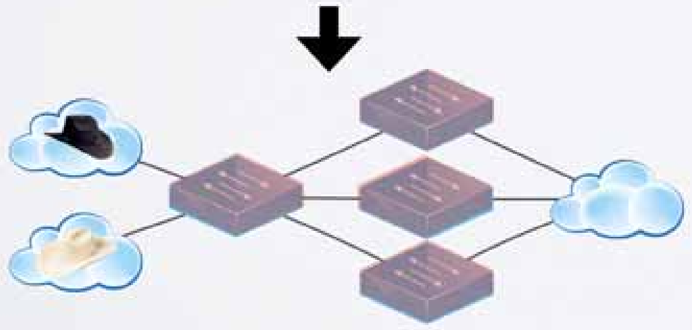
Per-Packet Consistent Update

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.

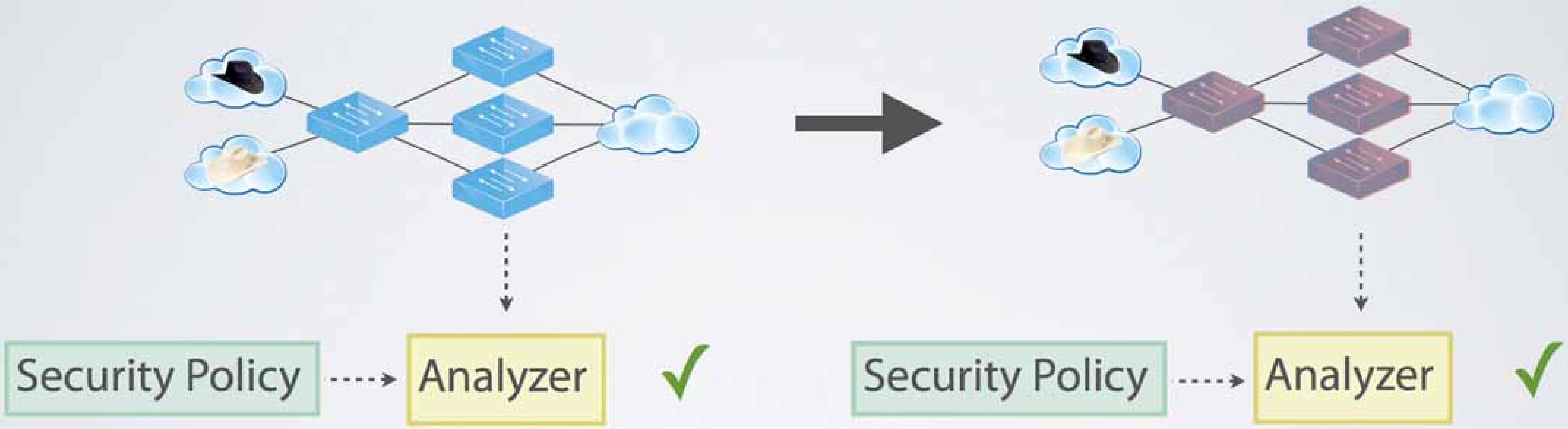
Trace Property
Any property of a *single* packet's path through the network.

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.



Verification Tools

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

MECHANISMS

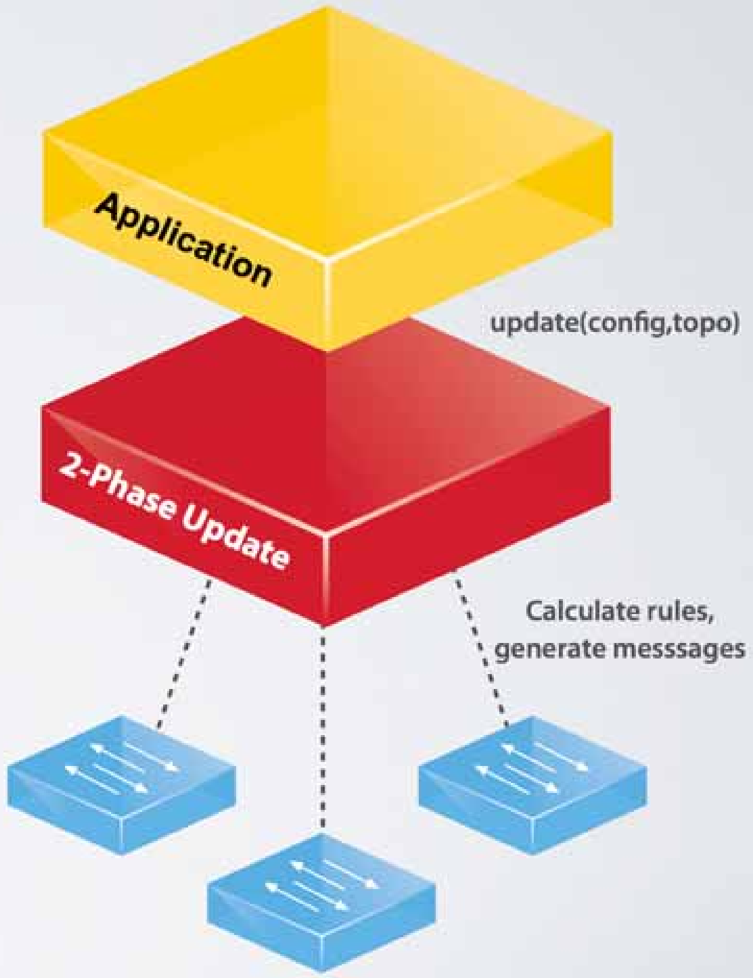
2-Phase Update

Overview

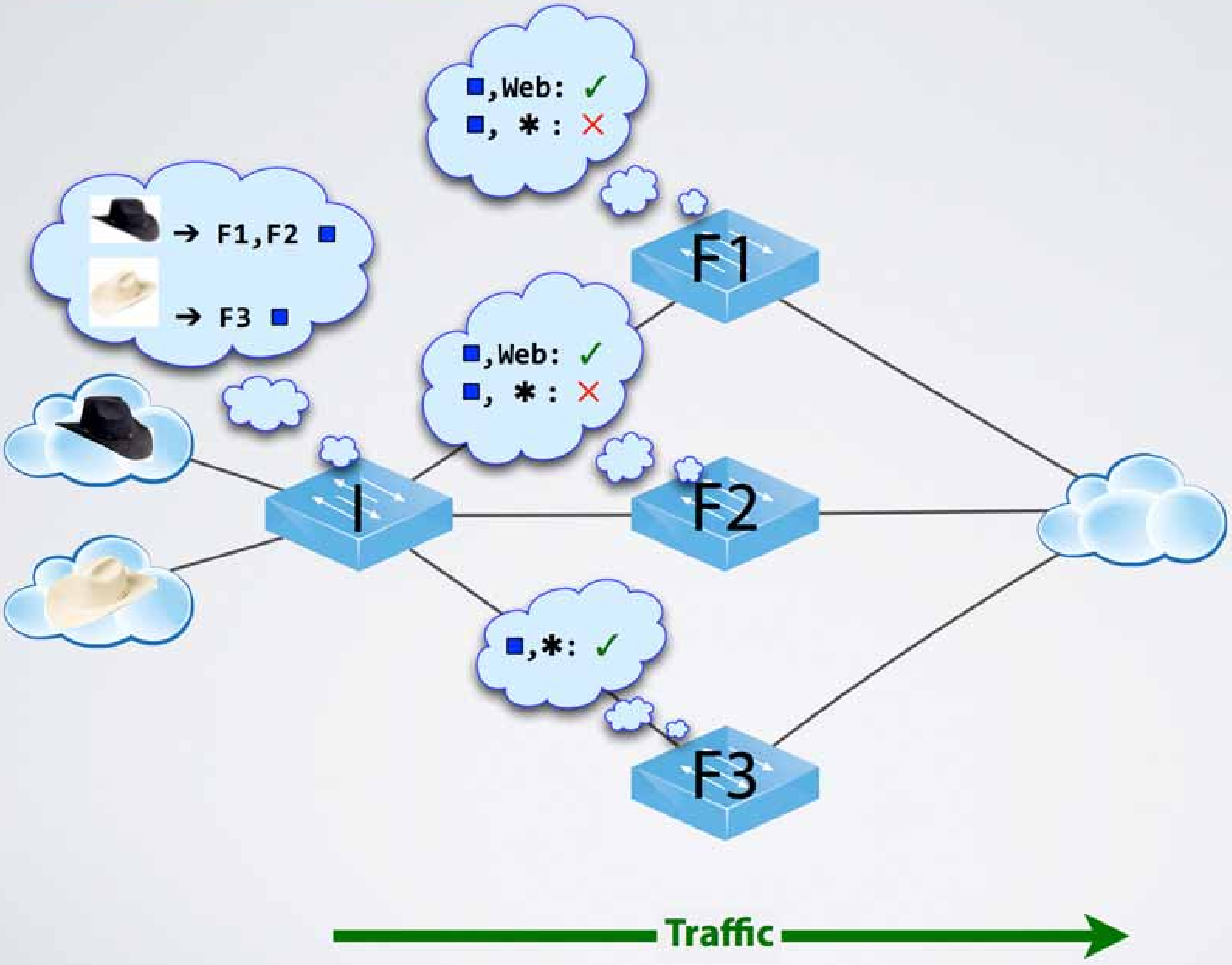
- Runtime instruments configurations
- Edge rules stamp packets with version
- Forwarding rules match on version

Algorithm (2-Phase Update)

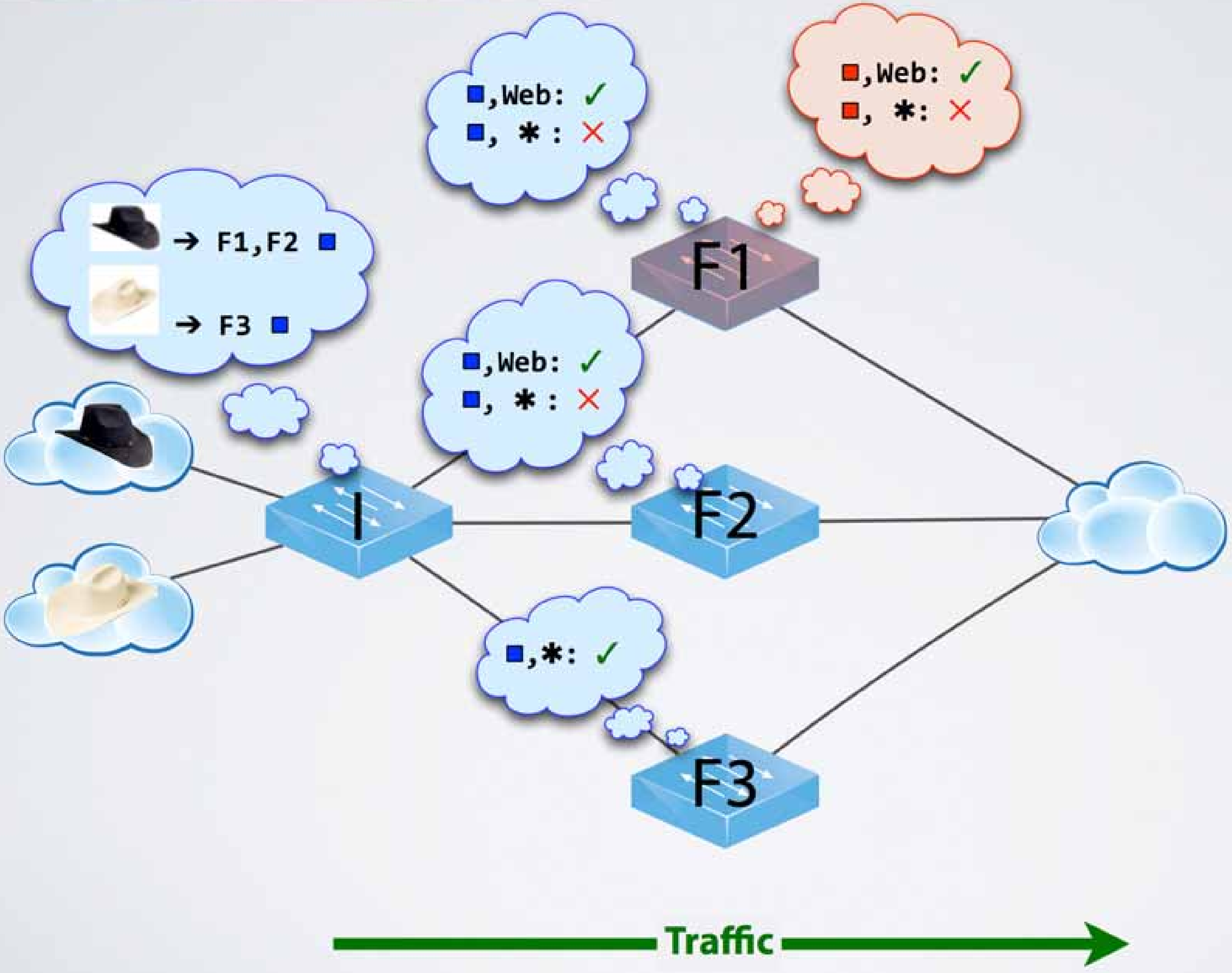
1. Install new rules on internal switches, leave old configuration in place
2. Install edge rules that stamp with the new version number



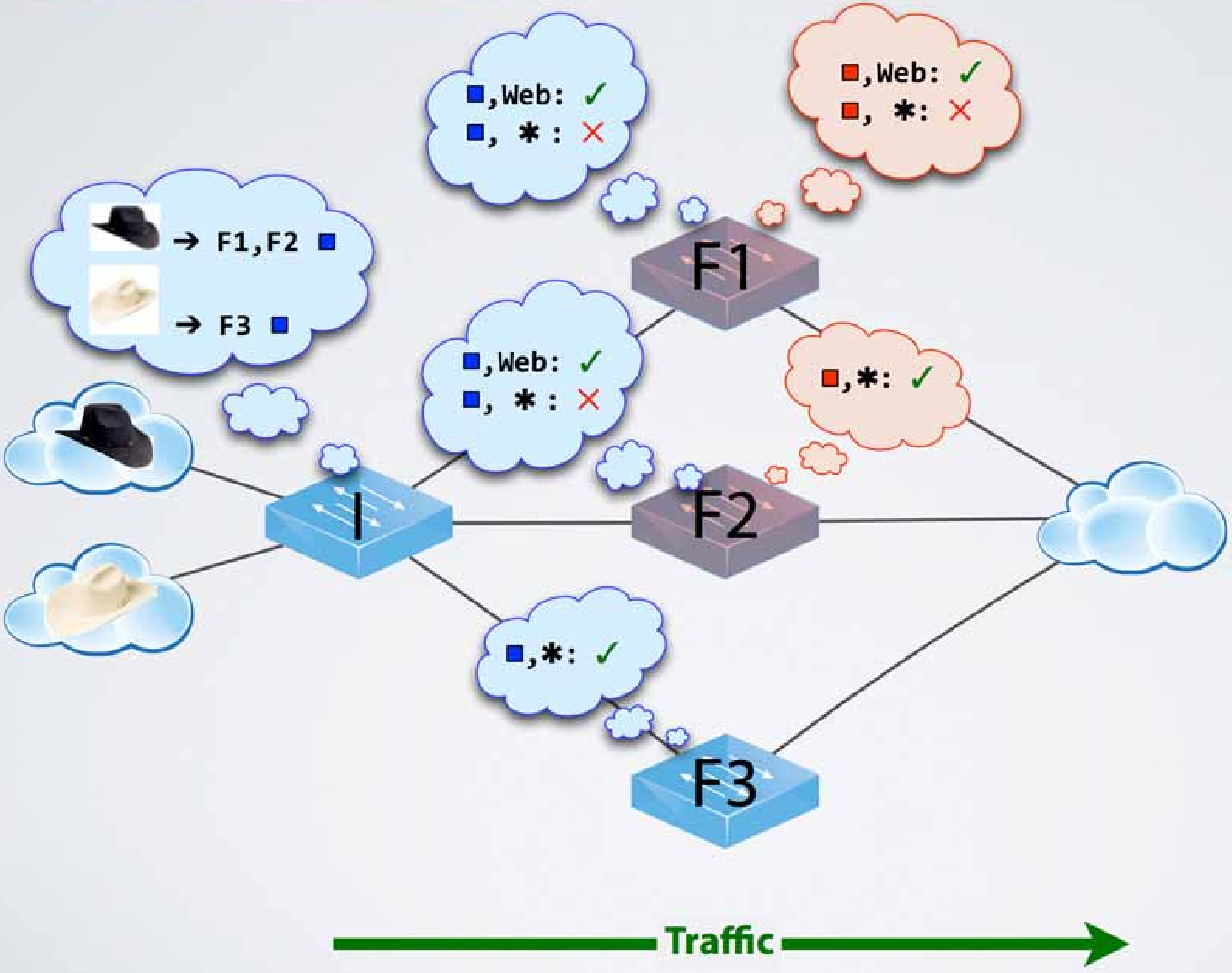
2-Phase Update in Action



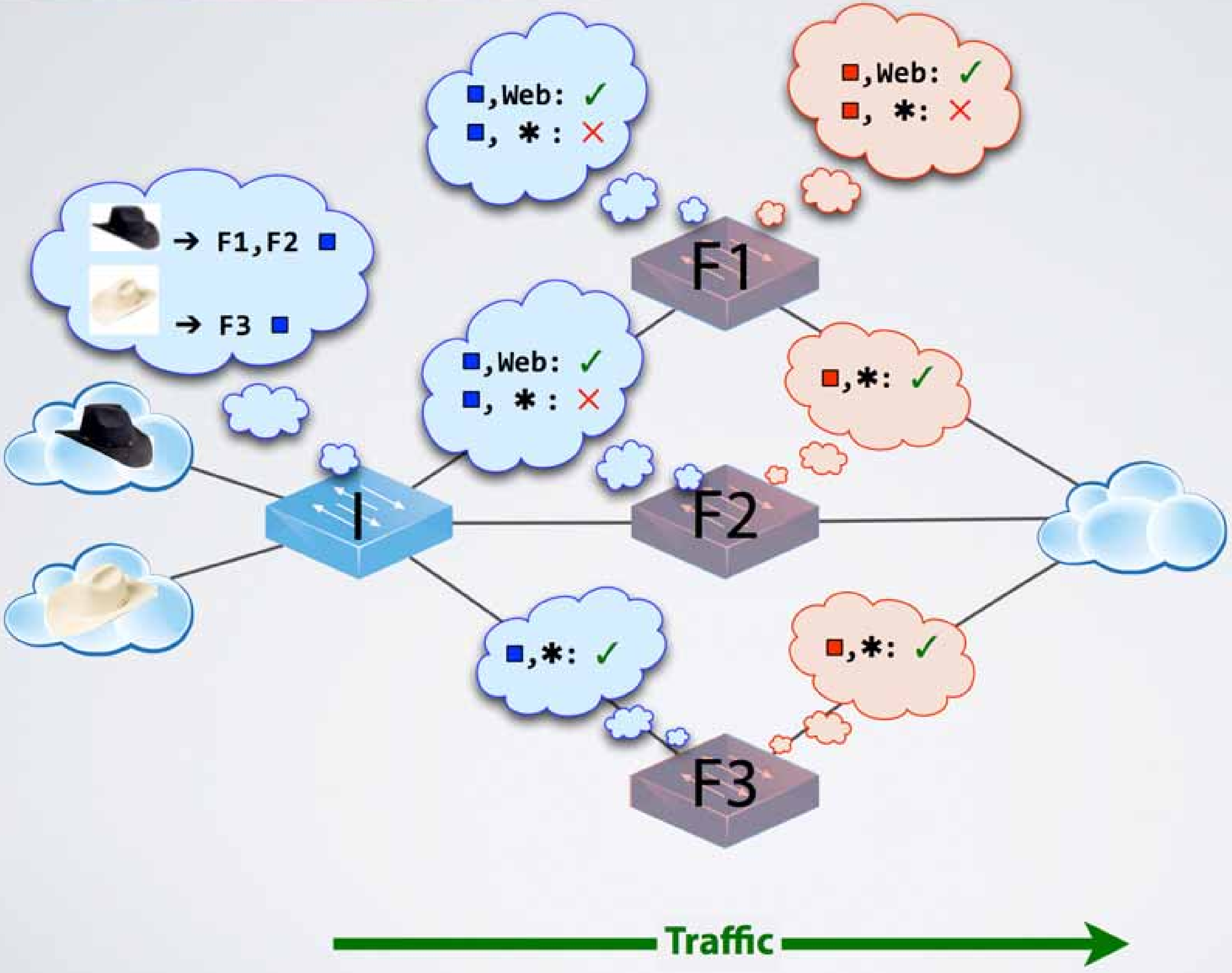
2-Phase Update in Action



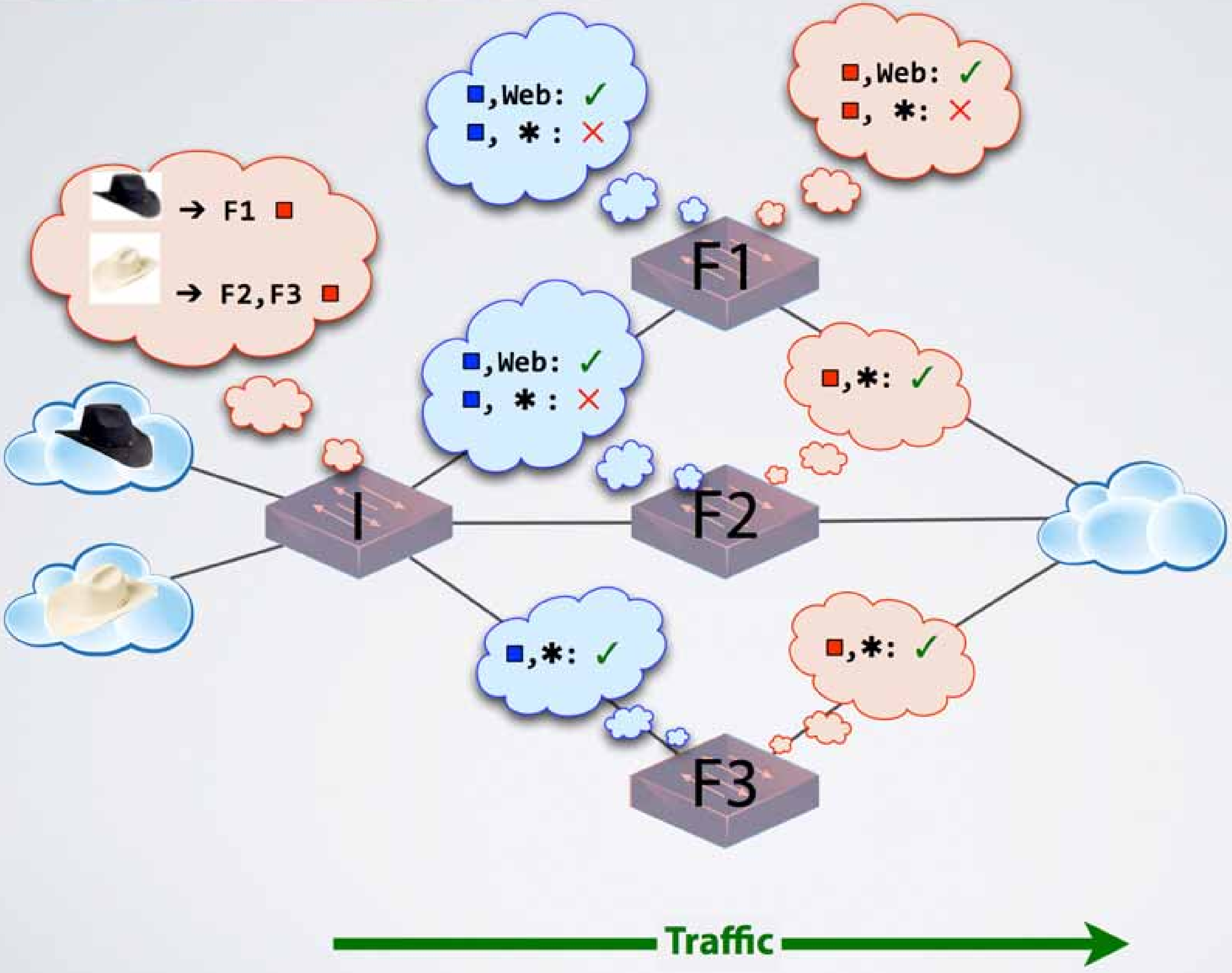
2-Phase Update in Action



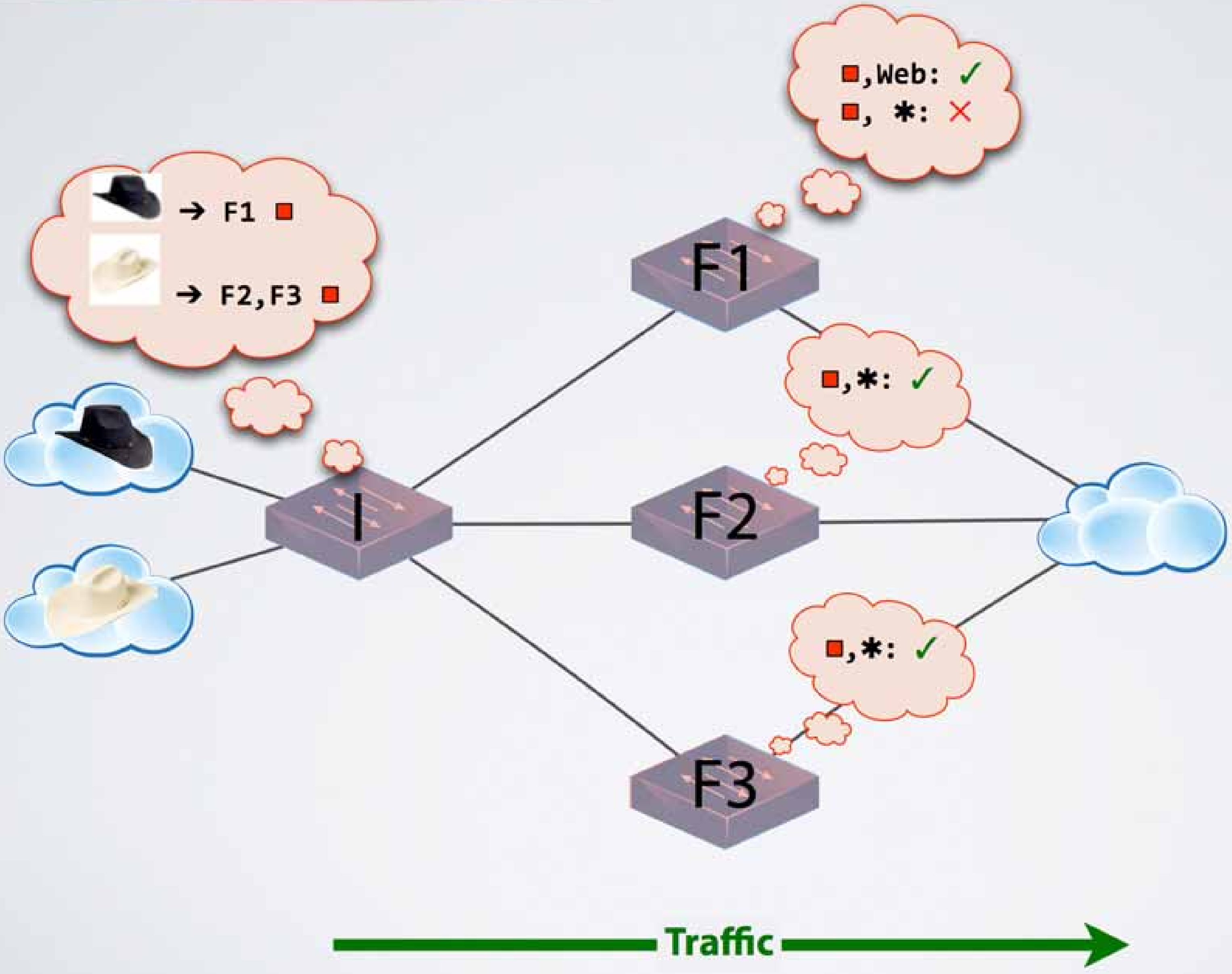
2-Phase Update in Action



2-Phase Update in Action



2-Phase Update in Action



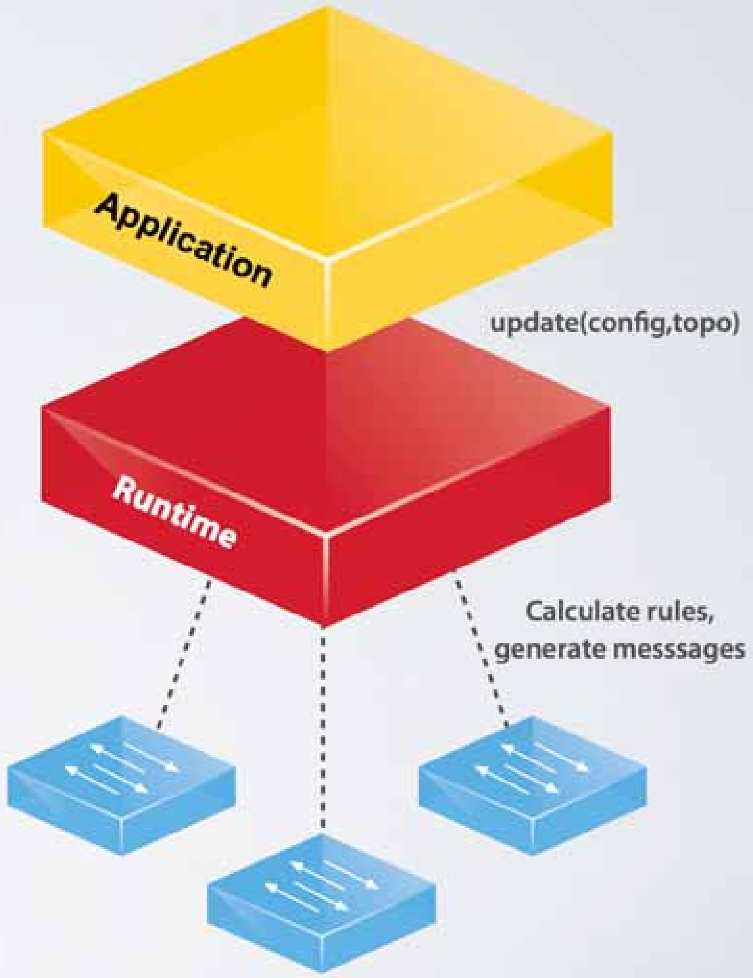
Optimized Mechanisms

Optimizations

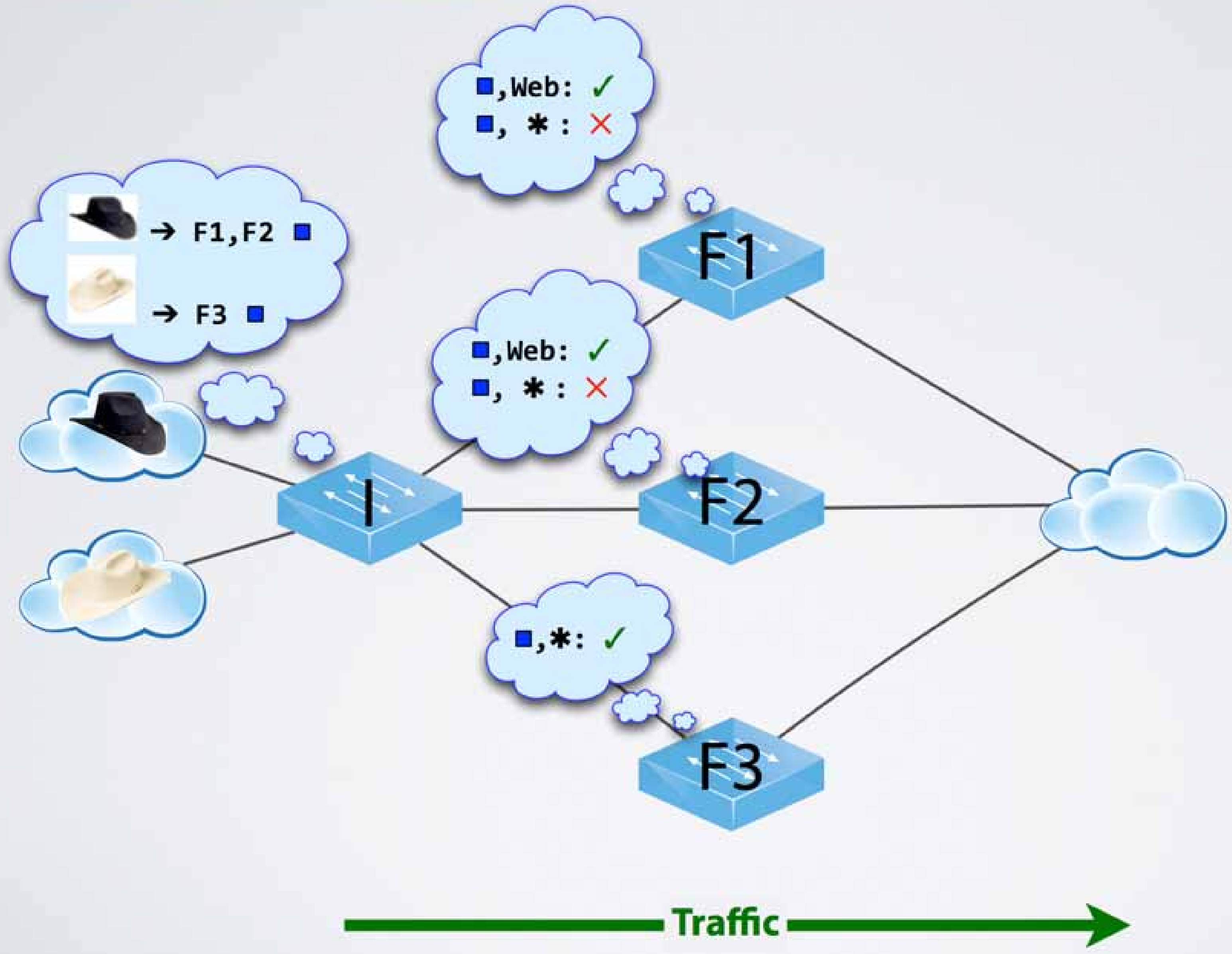
- Extension: strictly adds paths
- Retraction: strictly removes paths
- Subset: affects small # of paths
- Topological: affects small # of switches

Runtime

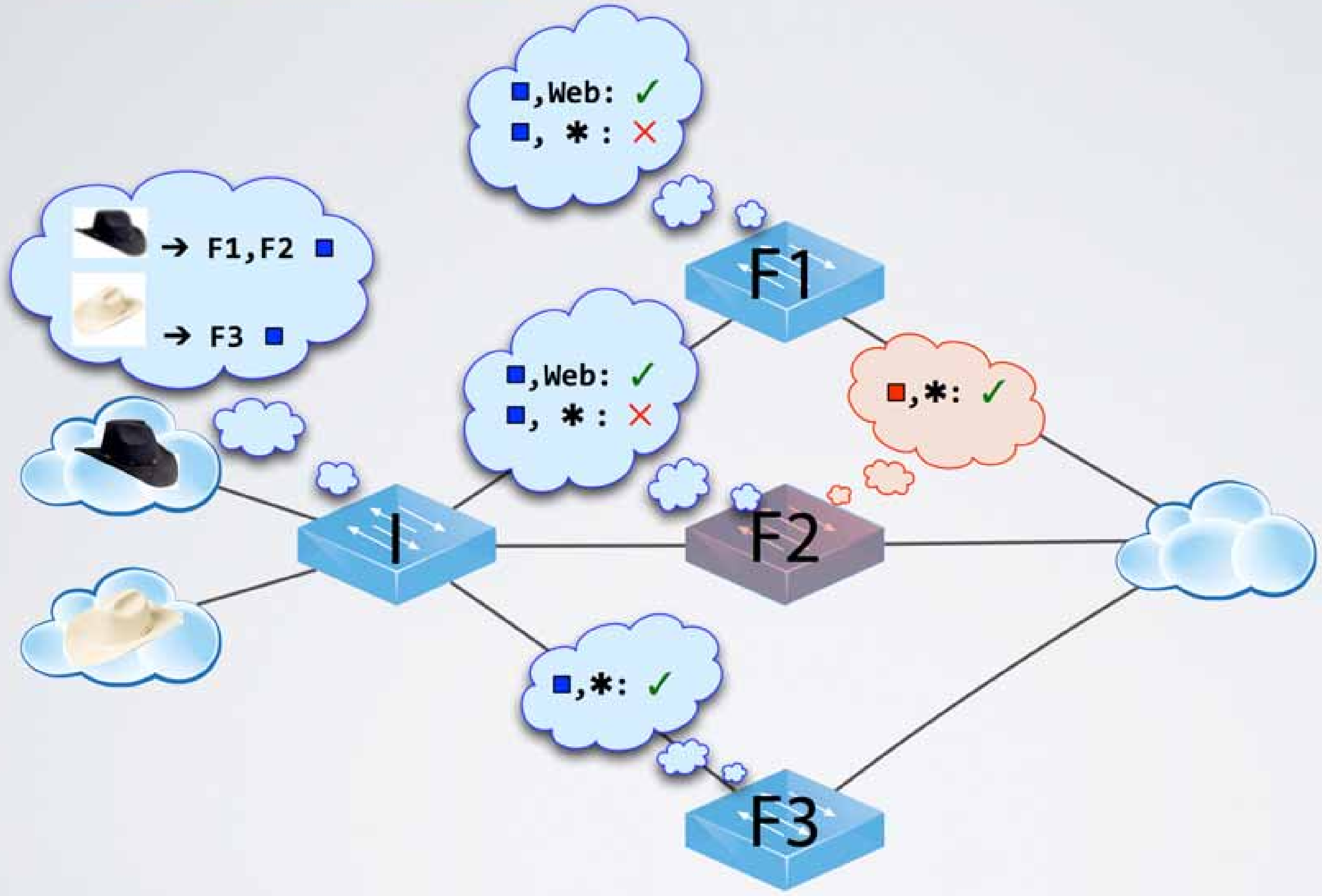
- Automatically optimizes
- Power of using abstraction



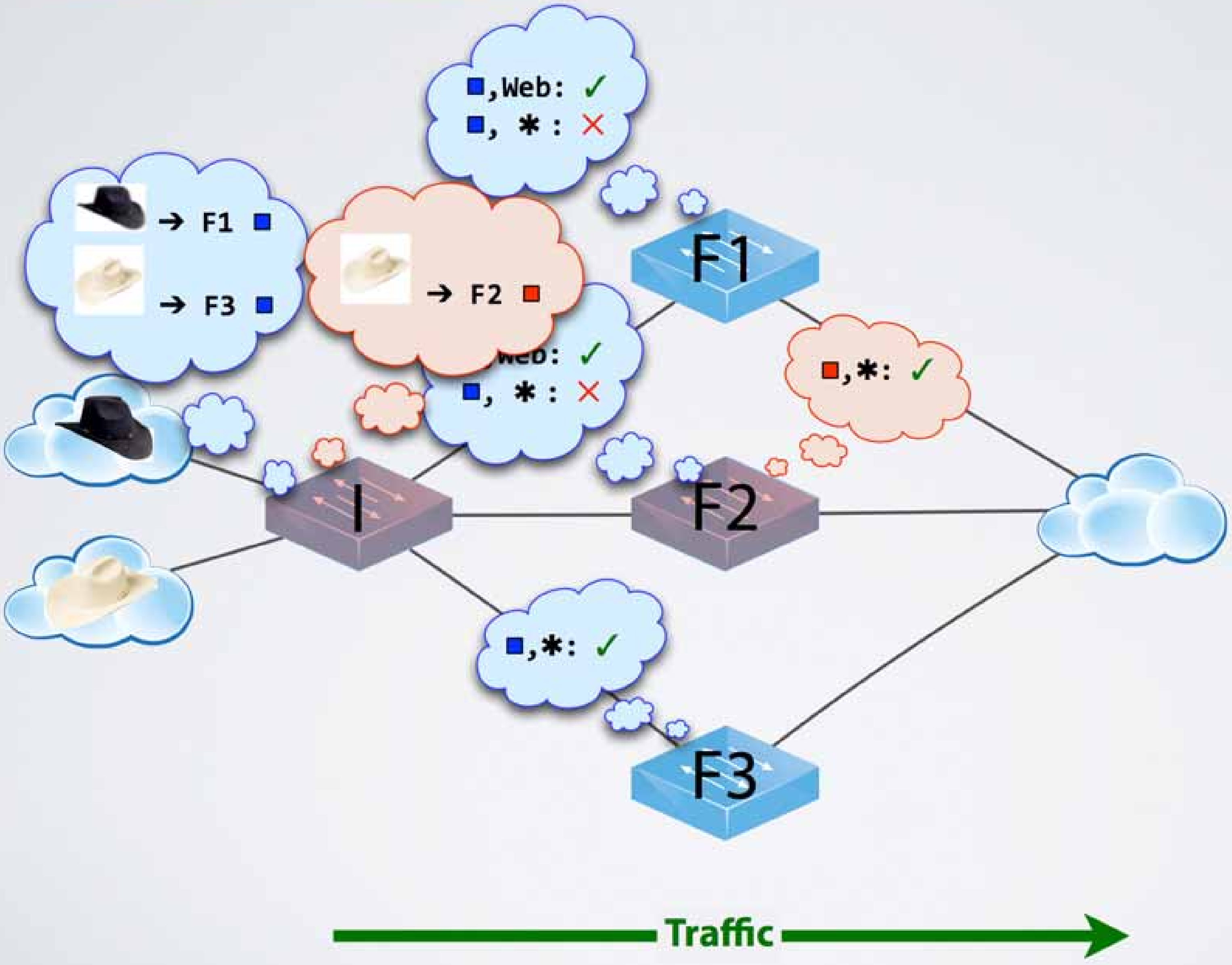
Subset Optimization



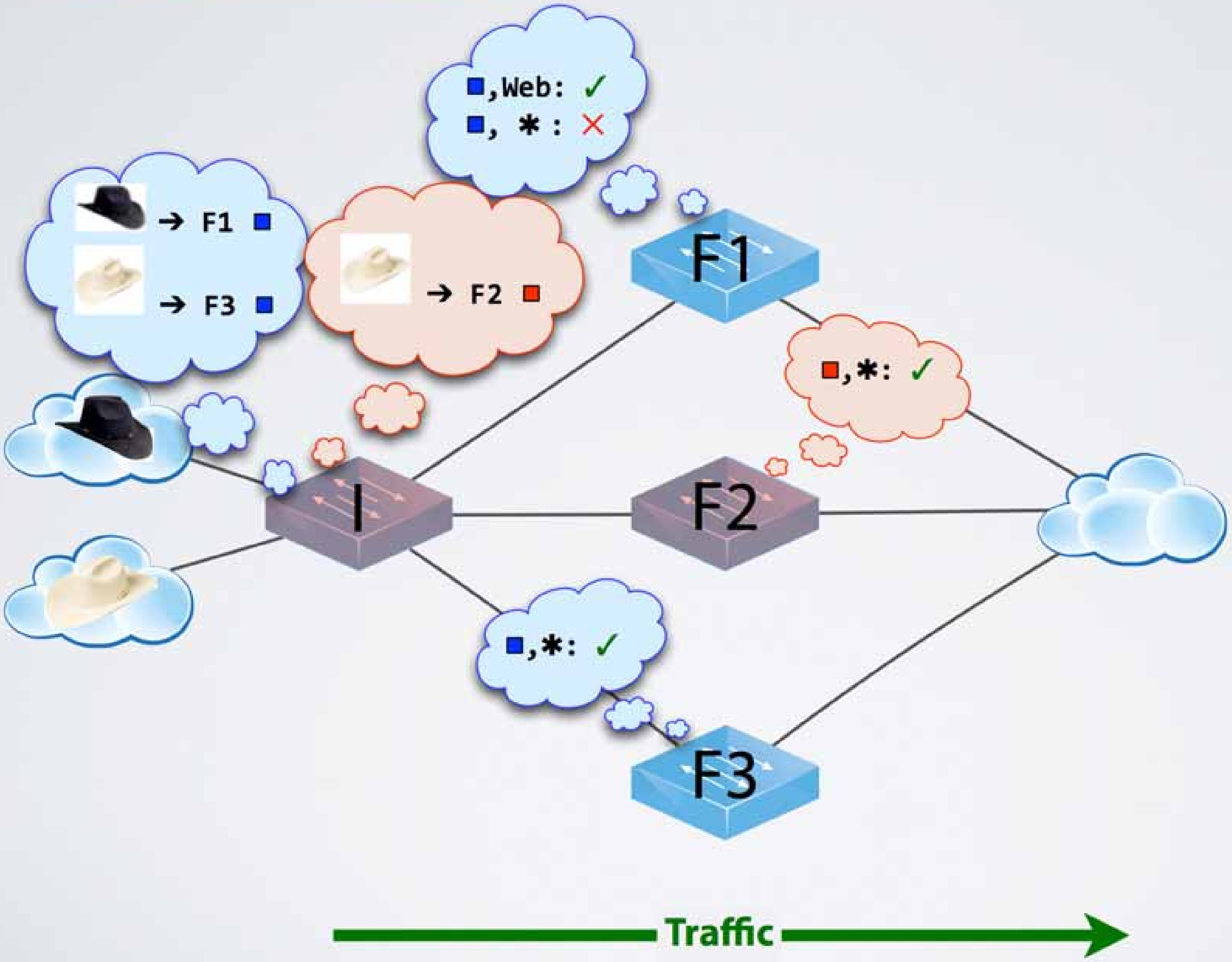
Subset Optimization



Subset Optimization



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 } Unobservable
2. Install edge rules that stamp with the new version number } One-touch

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 } Unobservable
2. Install edge rules that stamp with the new version number } One-touch

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

IMPLEMENTATION & EVALUATION

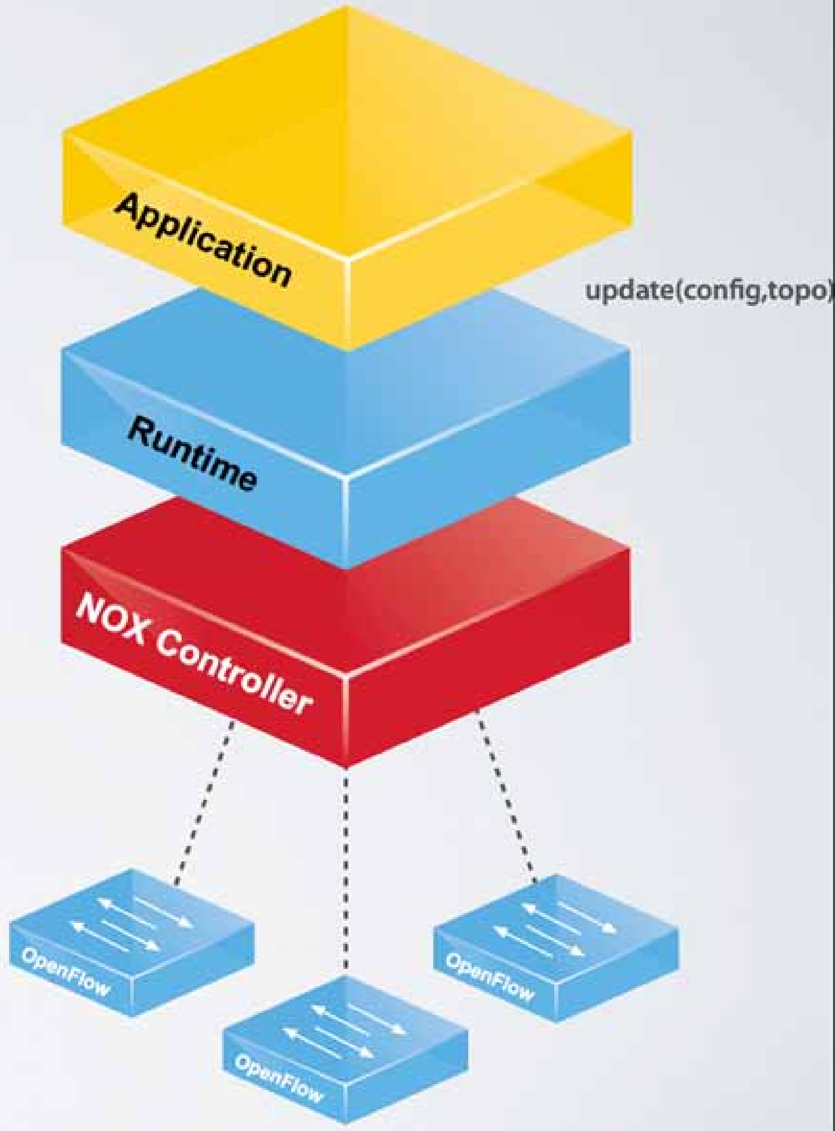
Implementation

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

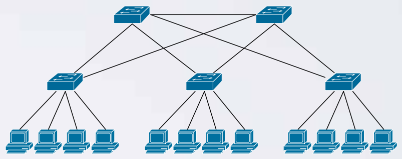
Applications

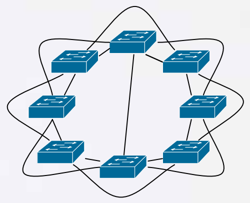
- Routing and Multicast

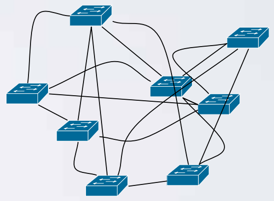
Scenarios

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

Topologies

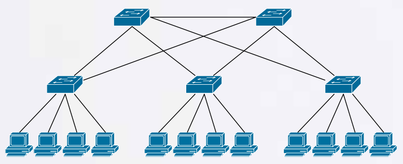

Fattree

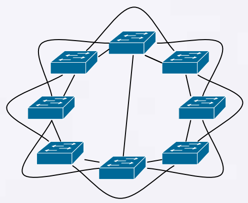

Small-world

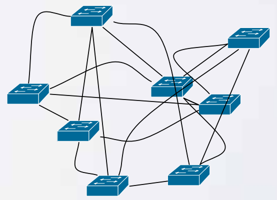

Waxman

Results: Routing Application

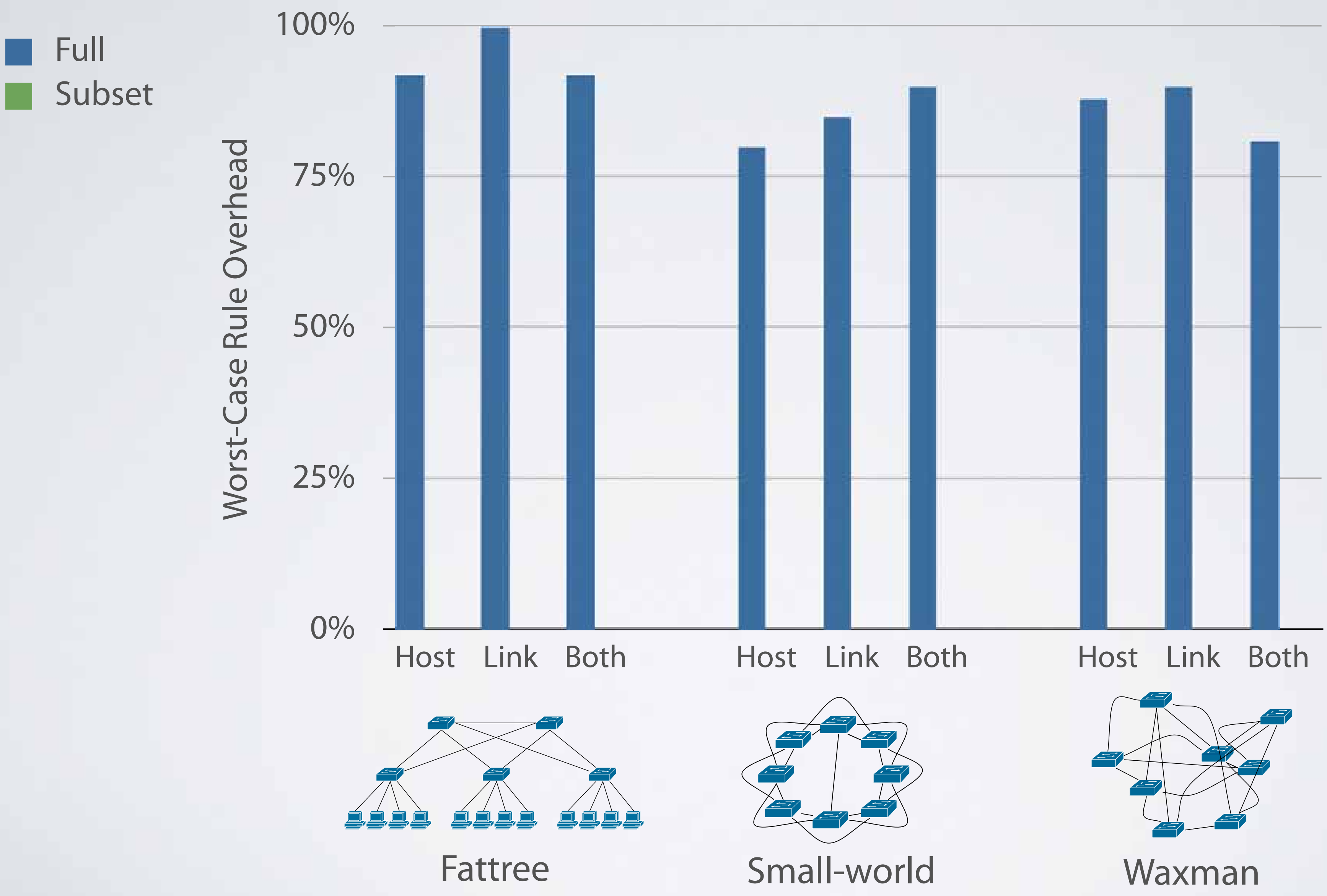
🟦 Full
🟩 Subset


Fattree

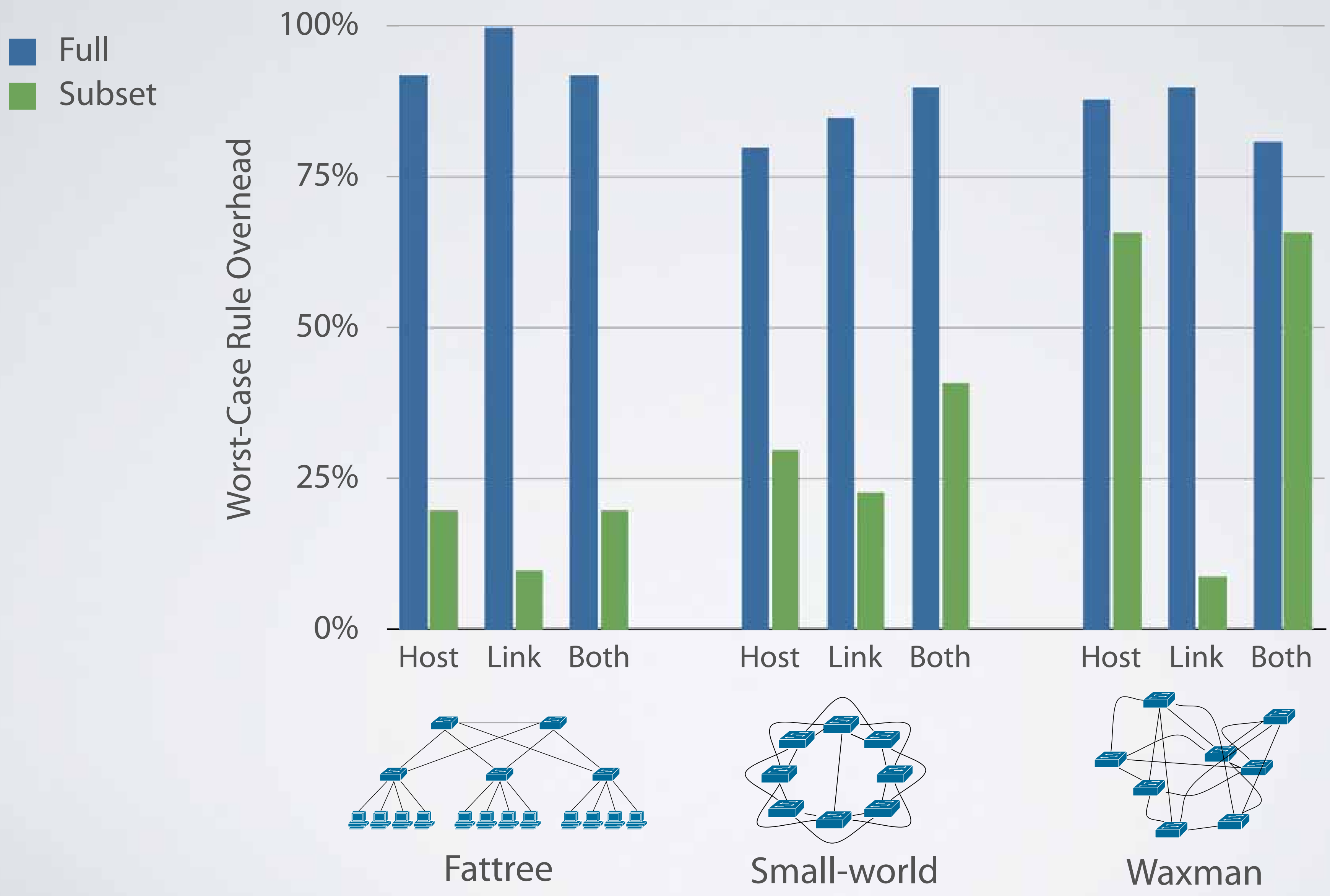

Small-world


Waxman

Results: Routing Application



Results: Routing Application



WRAP UP

Conclusion

Update abstractions

- Per-packet
- Per-flow

Mechanisms

- 2-Phase Update
- Optimizations

Implementation

- Runtime
- Verifier

Formal model

- Network operational semantics
- Universal property preservation

Thank You!

Collaborators

Shrutarshi Basu (Cornell)
Arjun Guha (Cornell)
Stephen Gutz (Cornell)
Rob Harrison (West Point)
Nanxi Kang (Princeton)
Naga Praveen Katta (Princeton)
Chris Monsanto (Princeton)
Josh Reich (Princeton)
Cole Schlesinger (Princeton)
Robert Soulé (Cornell)
Alec Story (Cornell)
Nate Foster (Cornell)
Mike Freedman (Princeton)
Jen Rexford (Princeton)
Emin Gün Sirer (Cornell)
Dave Walker (Princeton)

frenetic >>

http://frenetic-lang.org

BACKUP SLIDES

Beyond Per-Packet

Per-flow consistent update
Each *set* of related packets processed with old or new configuration, but not a mixture of the two.

Use Cases

- Load balancer
- Flow affinity
- In-order delivery

Mechanism

- 2-Phase Update + "flow tracking"