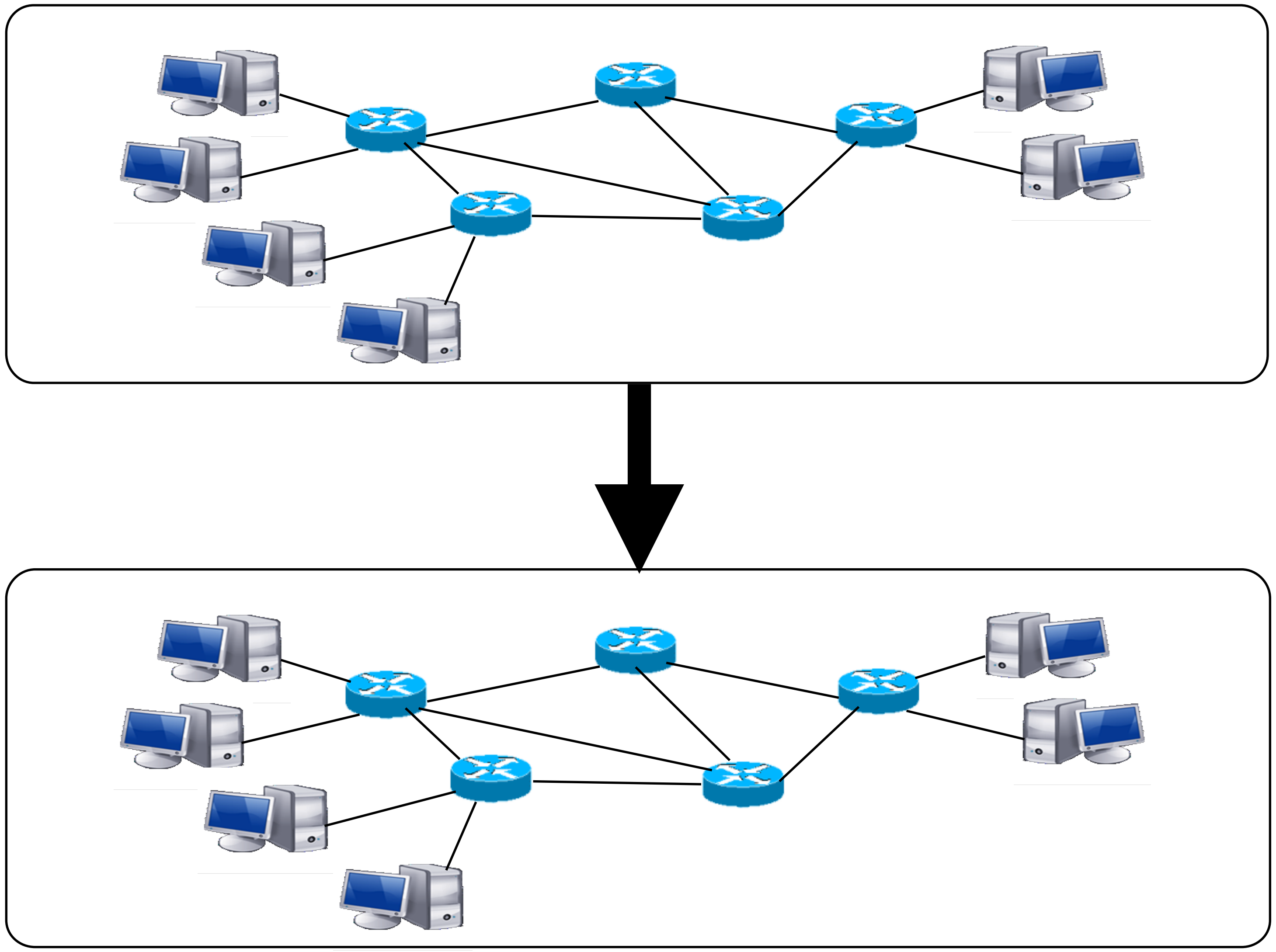
Enforcing Customizable Consistency Properties in Software-Defined Networks

Wenxuan Zhou, Dong Jin, Jason Croft,
Matthew Caesar, Brighten Godfrey

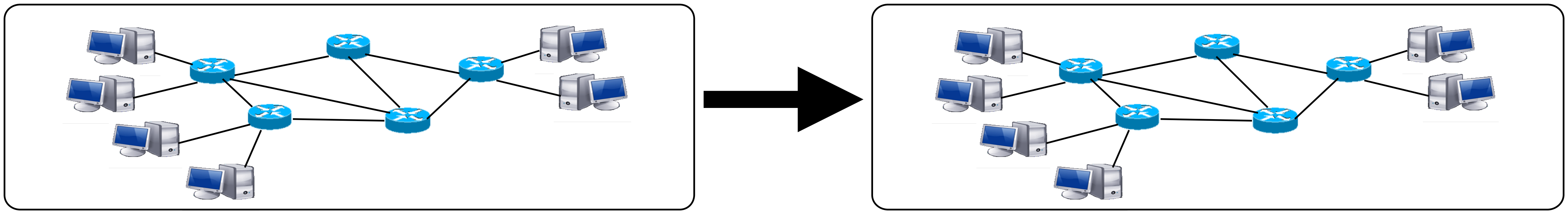


Network changes

- control applications,
- changes in traffic load,
- system upgrades,
- …

Keeping network correct consistently over time.

-- ***Network Consistency***



1. Correctness at every step
2. Customizable properties
3. With efficient update installation

What is Correctness?

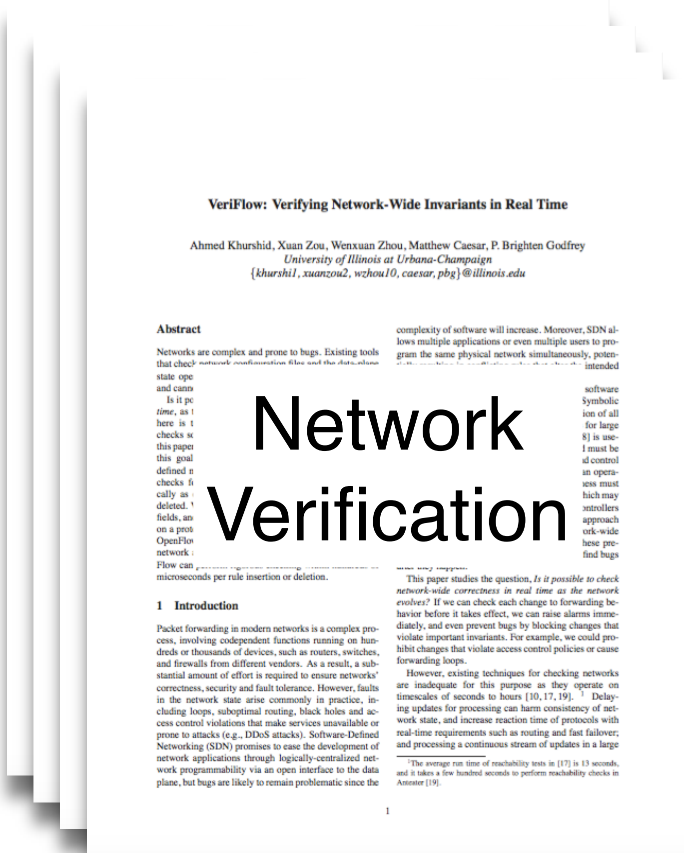
- firewall traversal,
- access control,
- balanced load,
- loop freedom,
- …

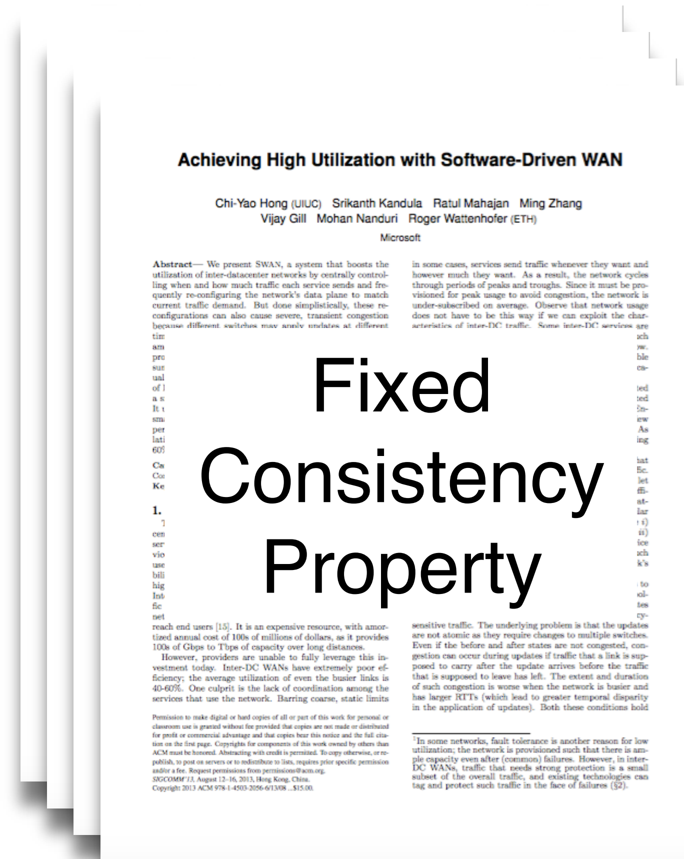
Problem Statement

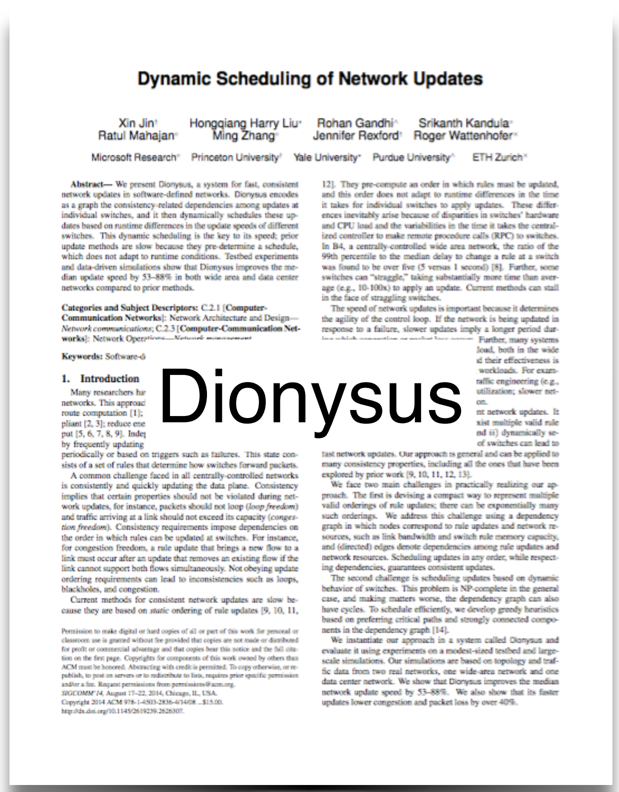
1. Consistency at every step
2. Customizable consistency properties
3. Efficient updates installation

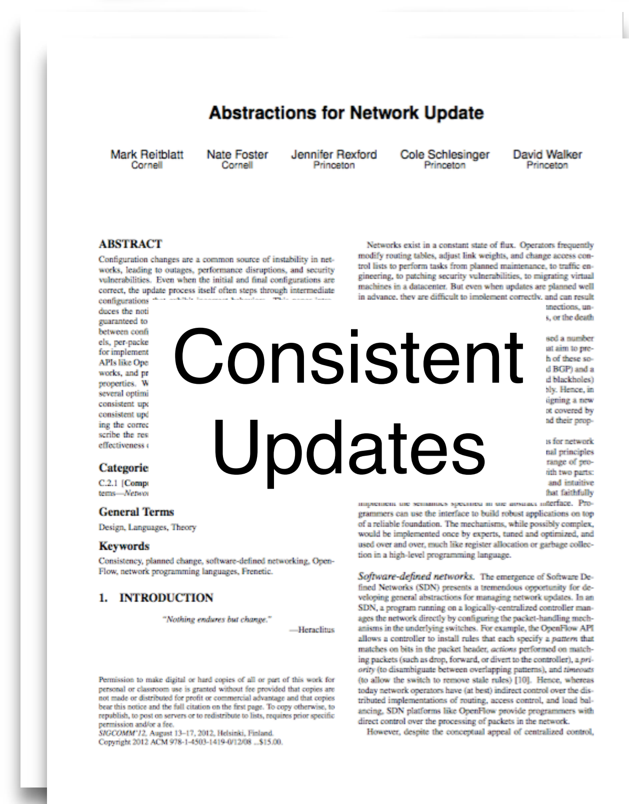
Is it possible to <u>efficiently</u> ensures <u>customizable</u> correctness properties as the network <u>evolves</u>?

Prior Work

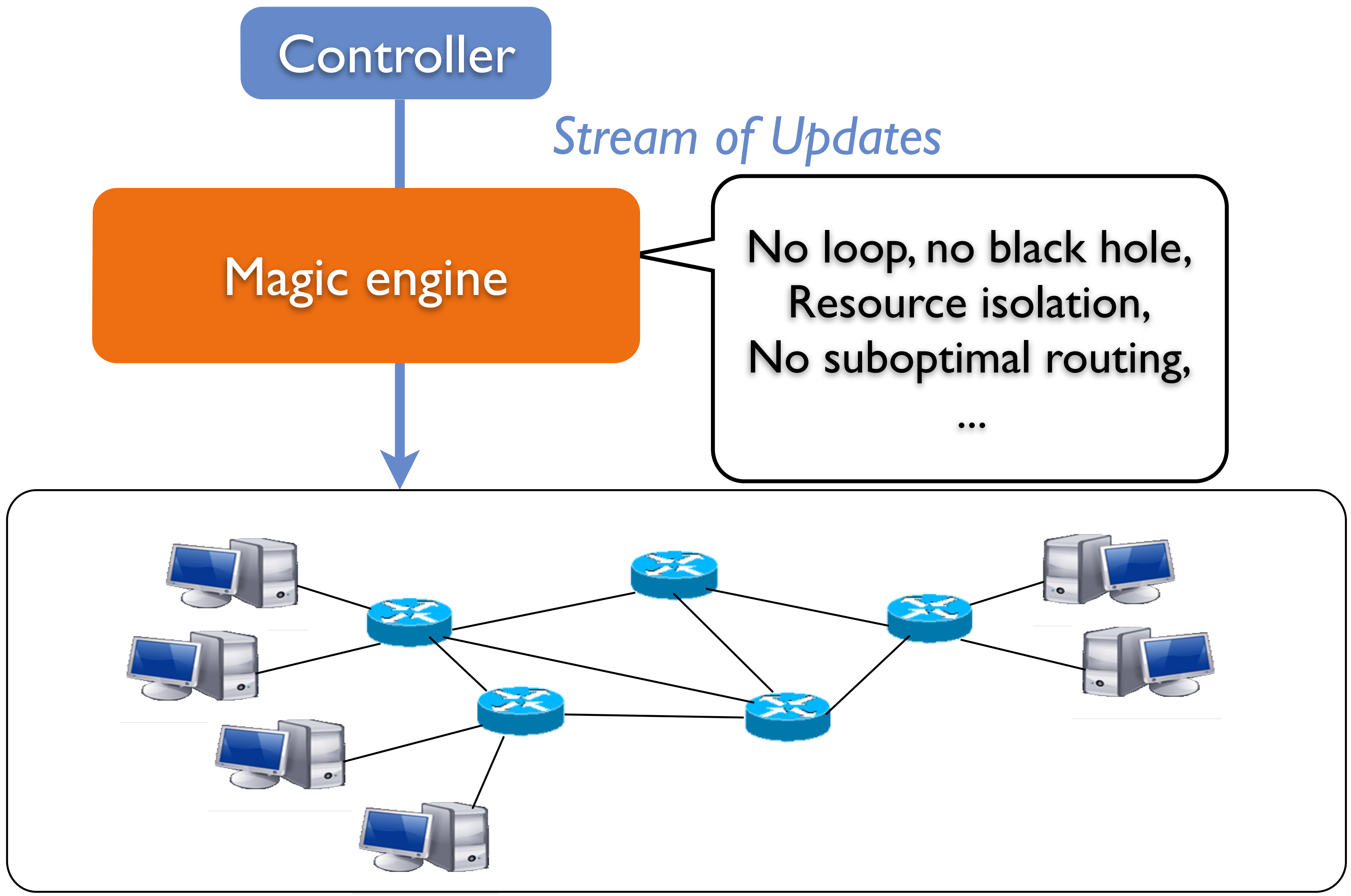

VeriFlow: Verifying Network-Wide Invariants in Real Time
Ahmed Khurshid, Xuan Zou, Wenxuan Zhou, Matthew Caesar, P. Brighten Godfrey
University of Illinois at Urbana-Champaign
Network
Verification


Achieving High Utilization with Software-Driven WAN
Chi-Yao Hong (UIUC) Srikanth Kandula Ratul Mahajan Ming Zhang
Vijay Gill Mohan Nanduri Roger Wattenhofer (ETH)
Microsoft
Fixed
Consistency
Property


Dynamic Scheduling of Network Updates
Xin Jin Hongqiang Harry Liu Rohan Gandhi Srikanth Kandula
Ratul Mahajan Ming Zhang Jennifer Rexford Roger Wattenhofer
Microsoft Research Princeton University Yale University Purdue University ETH Zurich
Dionysus

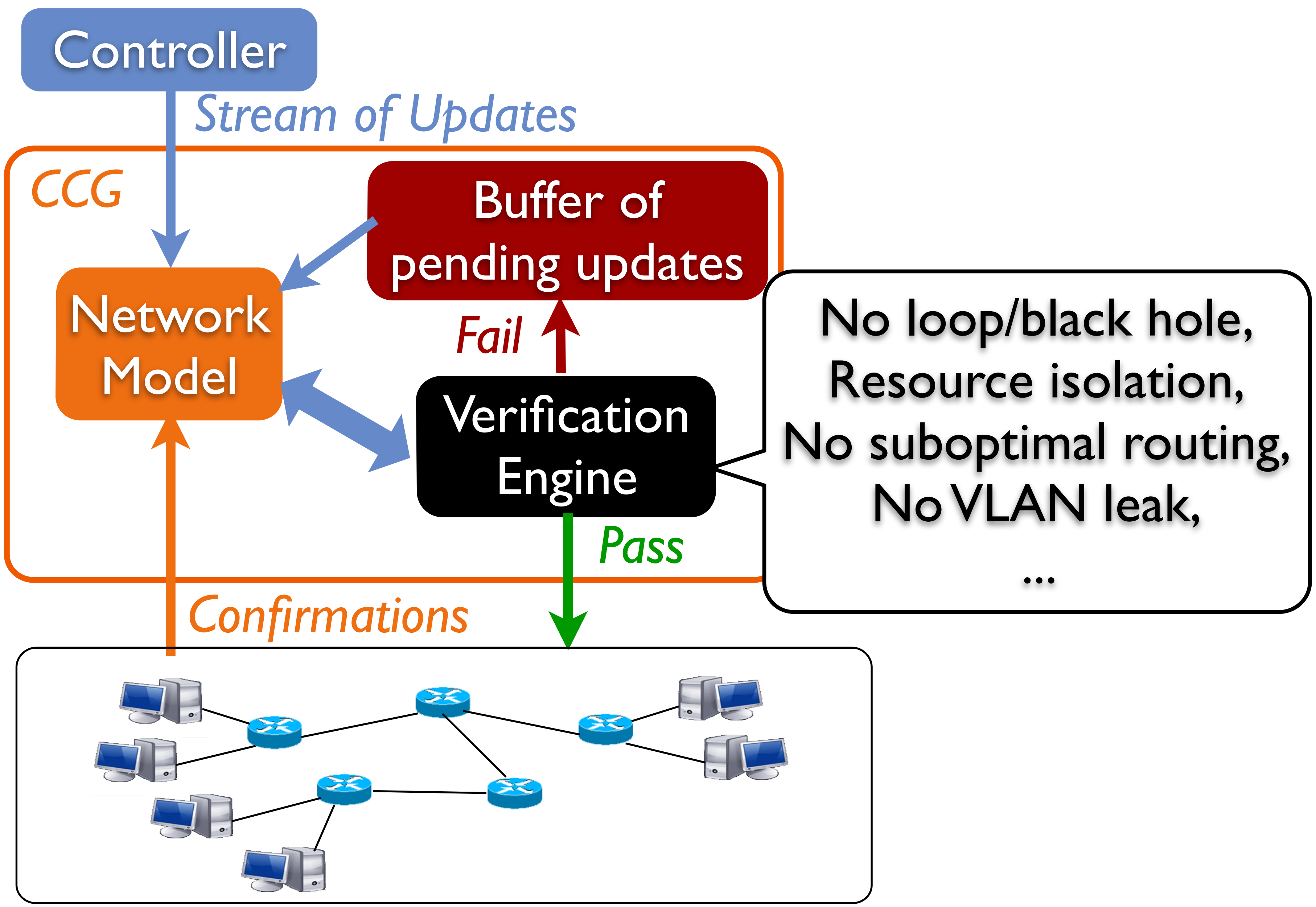

Abstractions for Network Update
Mark Reitblatt Cornell
Nate Foster Cornell
Jennifer Rexford Princeton
Cole Schlesinger Princeton
David Walker Princeton
Consistent
Updates

Ideally given **arbitrary** invariants, a sequence with **minimized** overhead is produced

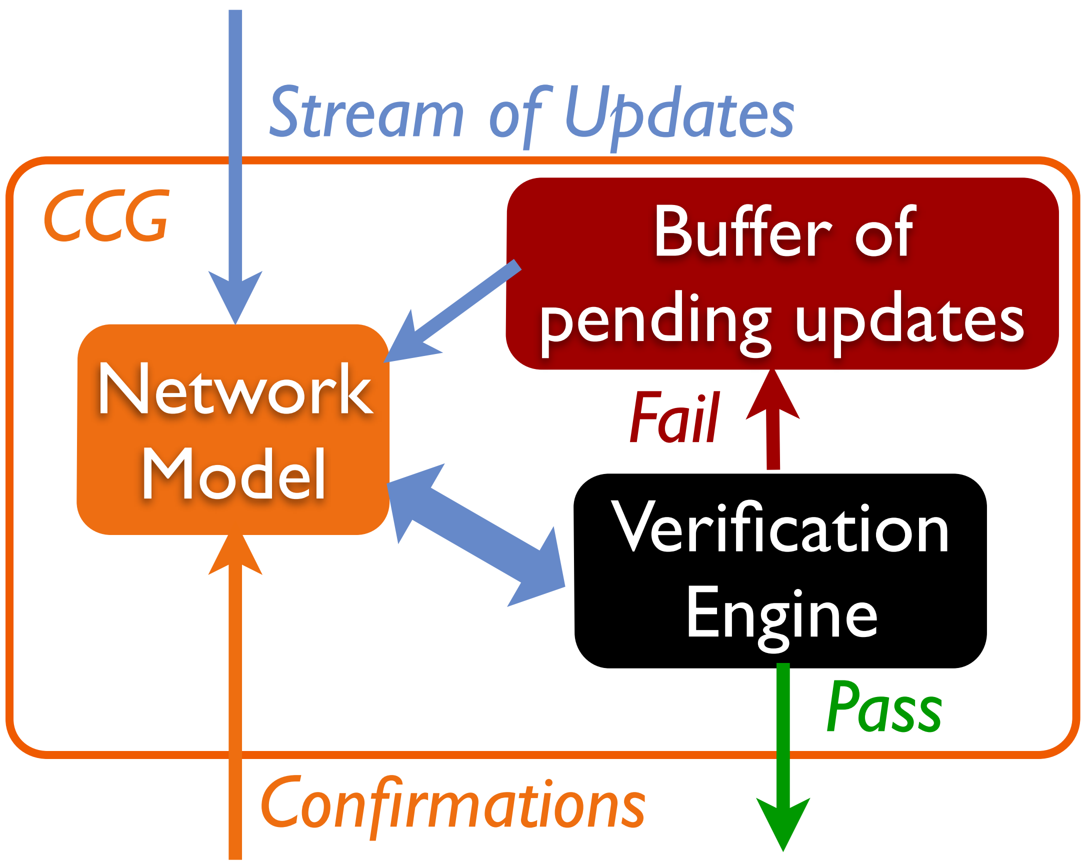


Our design: Customizable Consistency Generator

Key insight: Synthesis → Verification



Our design: Customizable Consistency Generator



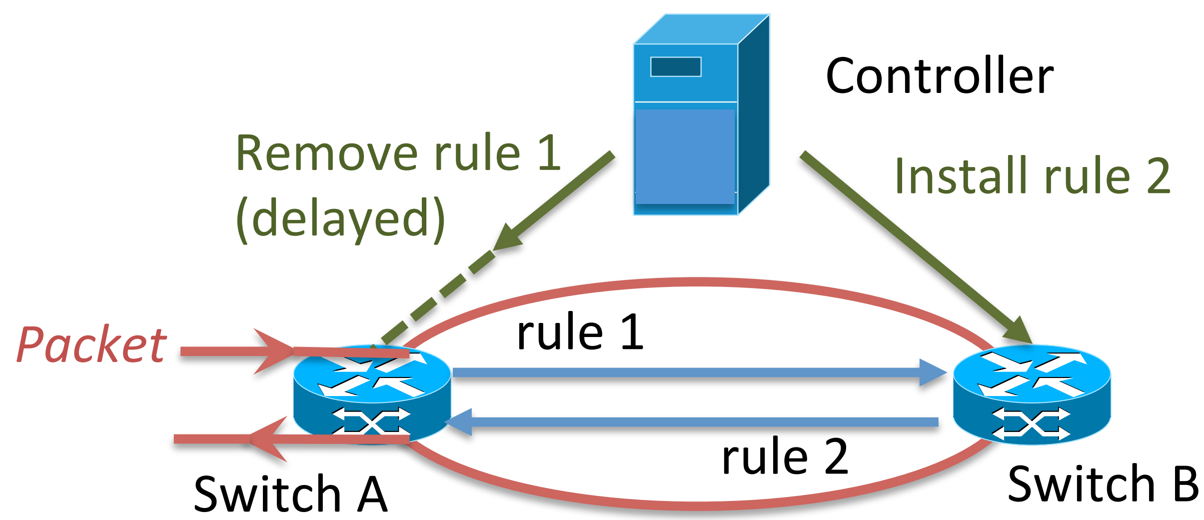
Challenges:

- Greedy algorithm may get stuck
  - ✳ identify the scope of cases that guarantees no deadlock
  - ✳ For other cases, a more heavyweight update technique as a fallback, triggered rarely in practice
- Distributed nature of networks (uncertainty)
  - ✳ compact *uncertain forwarding graph*
  - ✳ verification optimization

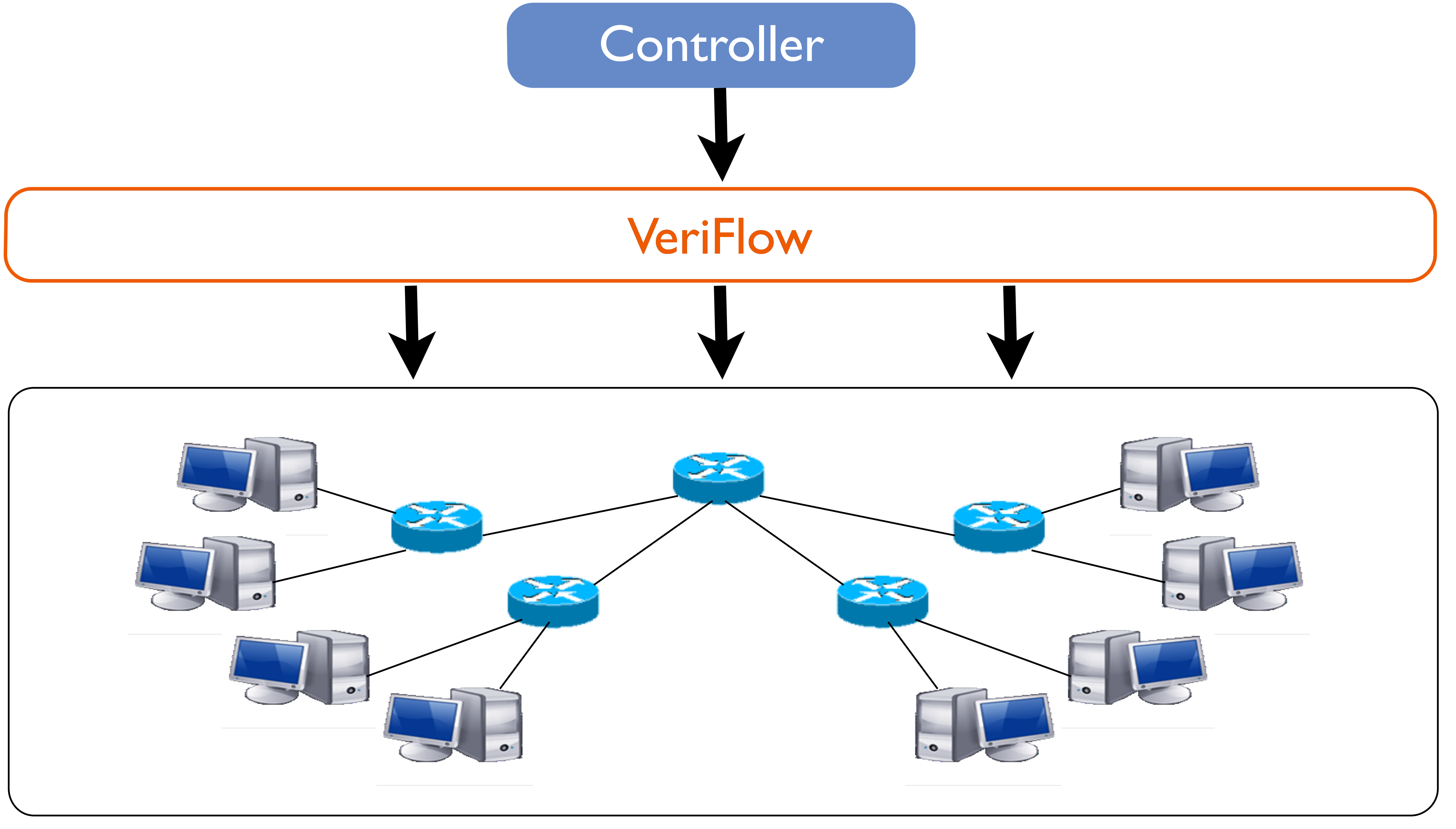
Network Uncertainty

The "uncertainty" of an observation point tasked with instilling updates in knowing the current network state.

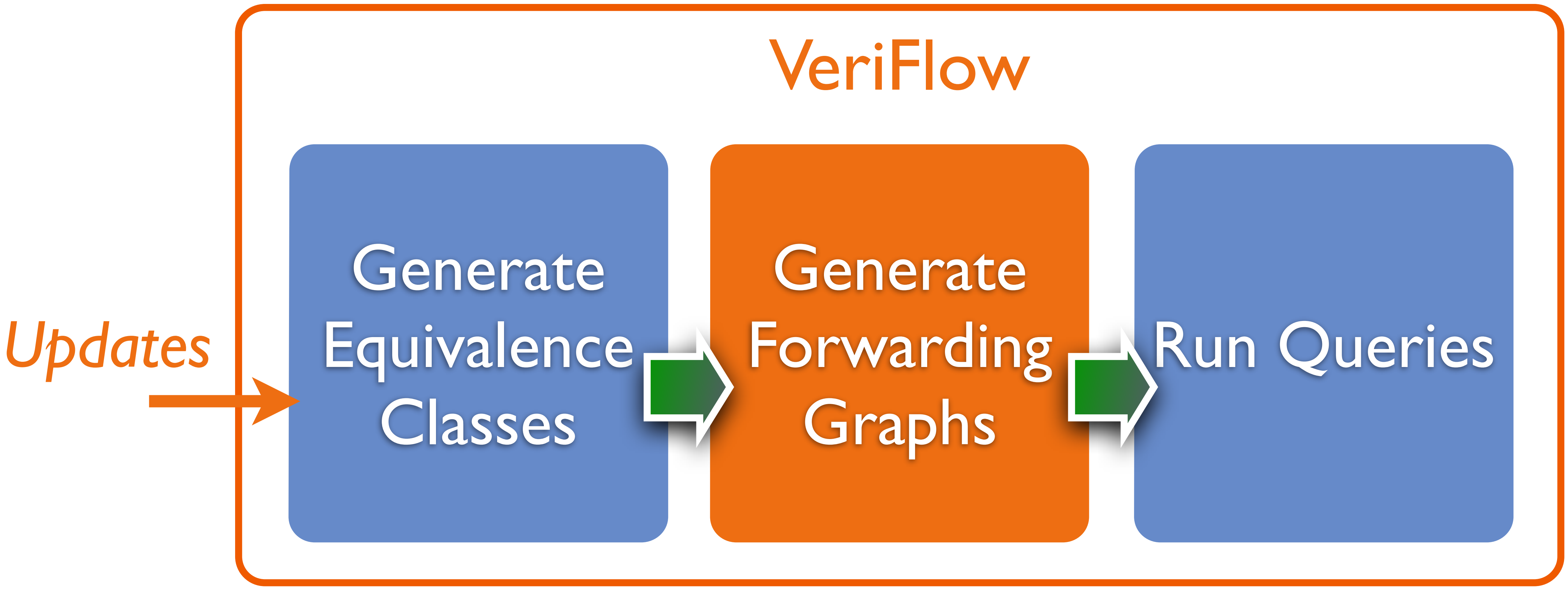
May deviate network behavior away from desired properties.



Uncertainty-aware Modeling Basis: VeriFlow

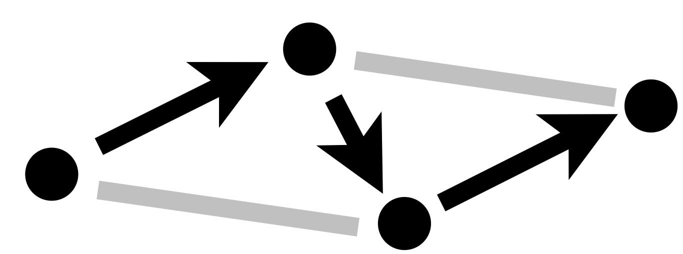


Uncertainty-aware Modeling Basis: VeriFlow



Equivalence class: Packets experiencing the same forwarding actions throughout the network.

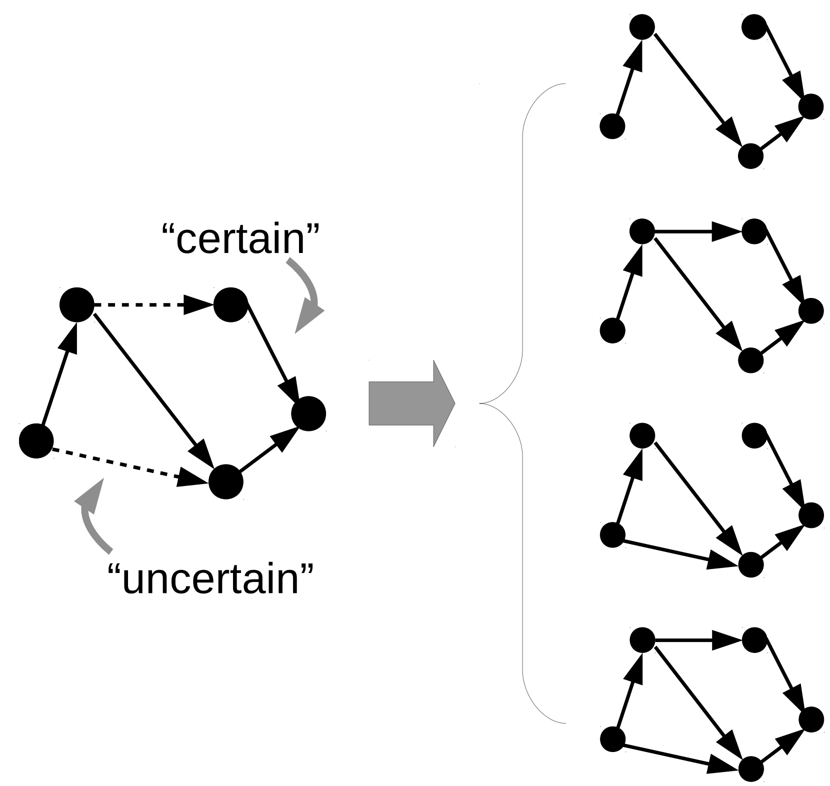
Forwarding graphs:



Uncertainty-aware Modeling

Naively, represent every possible network state O(2^n)

Uncertain graph: represent all possible combinations



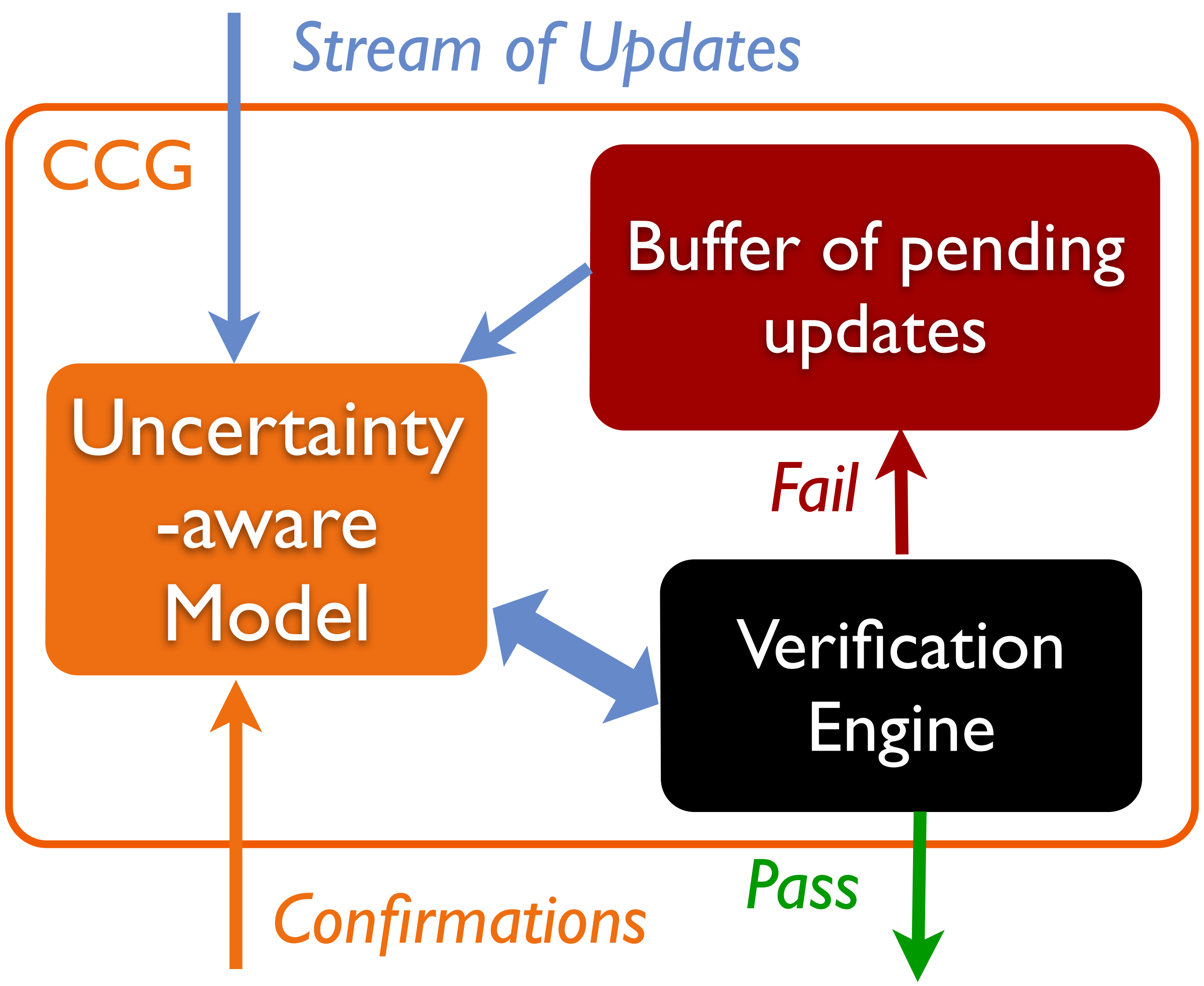
The model captures packets' view of the network, assuming controller initiates changes.

When to change "uncertain" to "certain"?

How to verify the network under "uncertainty"?

Consistency under Uncertainty

Enforcing consistency with max parallelism heuristically

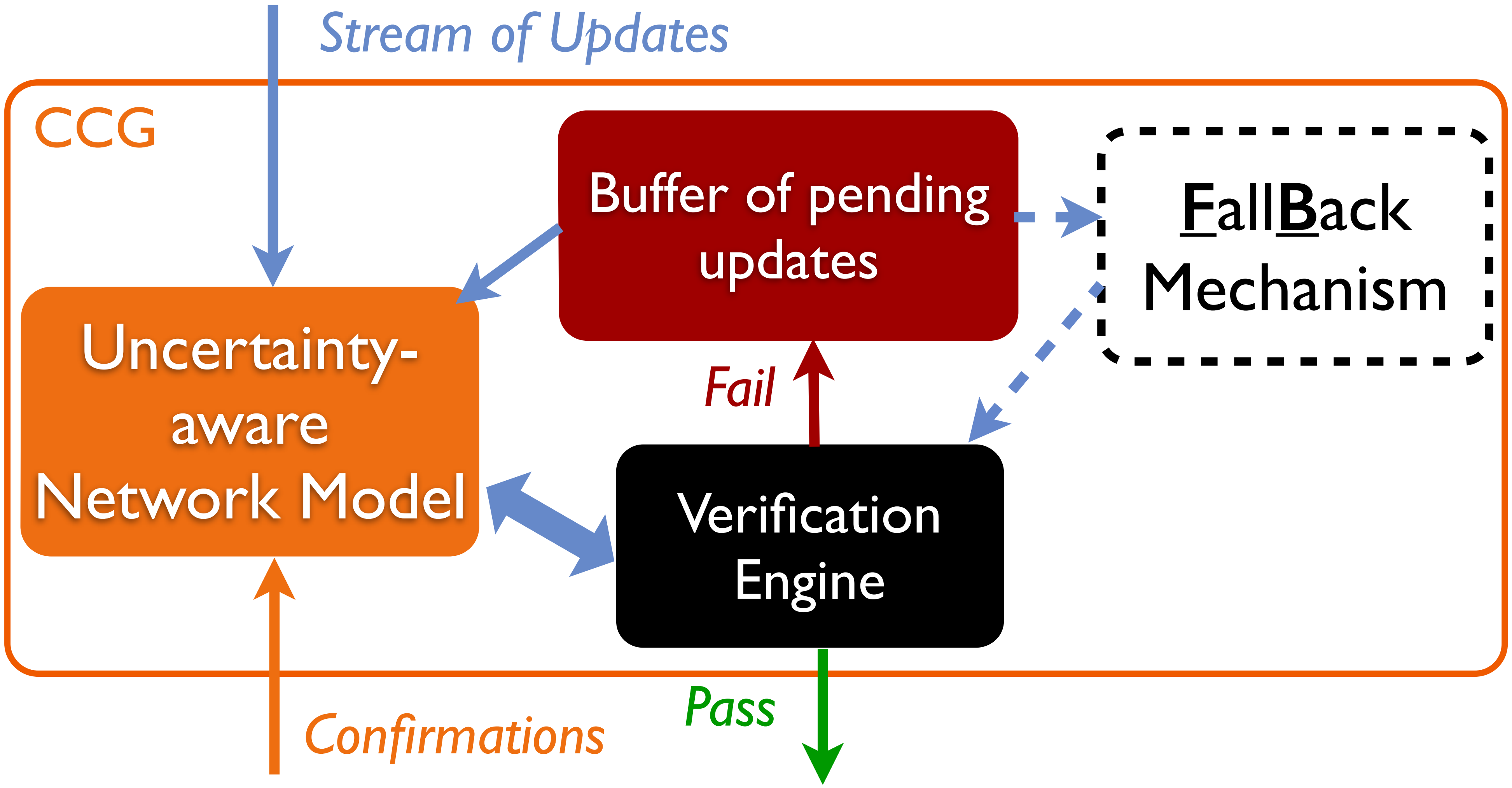


Waypoint Properties: flows are required to traverse a set of waypoints

- connectivity,
- waypointing,
- access control,
- service chaining, …

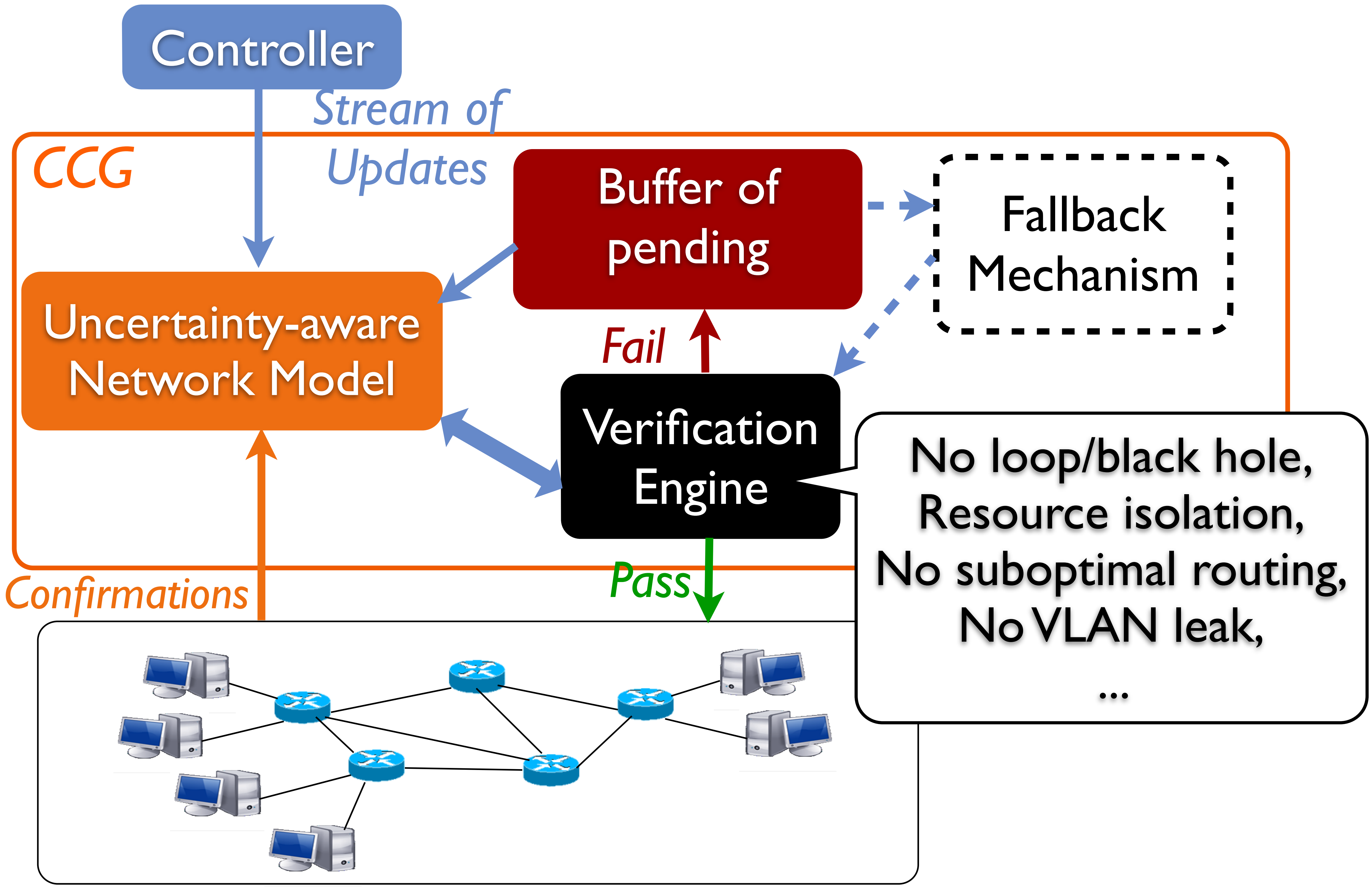
Theorem: *Segment independent* properties is guaranteed by the heuristic.

Consistency under Uncertainty



Even with FB triggered, CCG achieves better efficiency than using FB alone.

System Structure



Evaluation

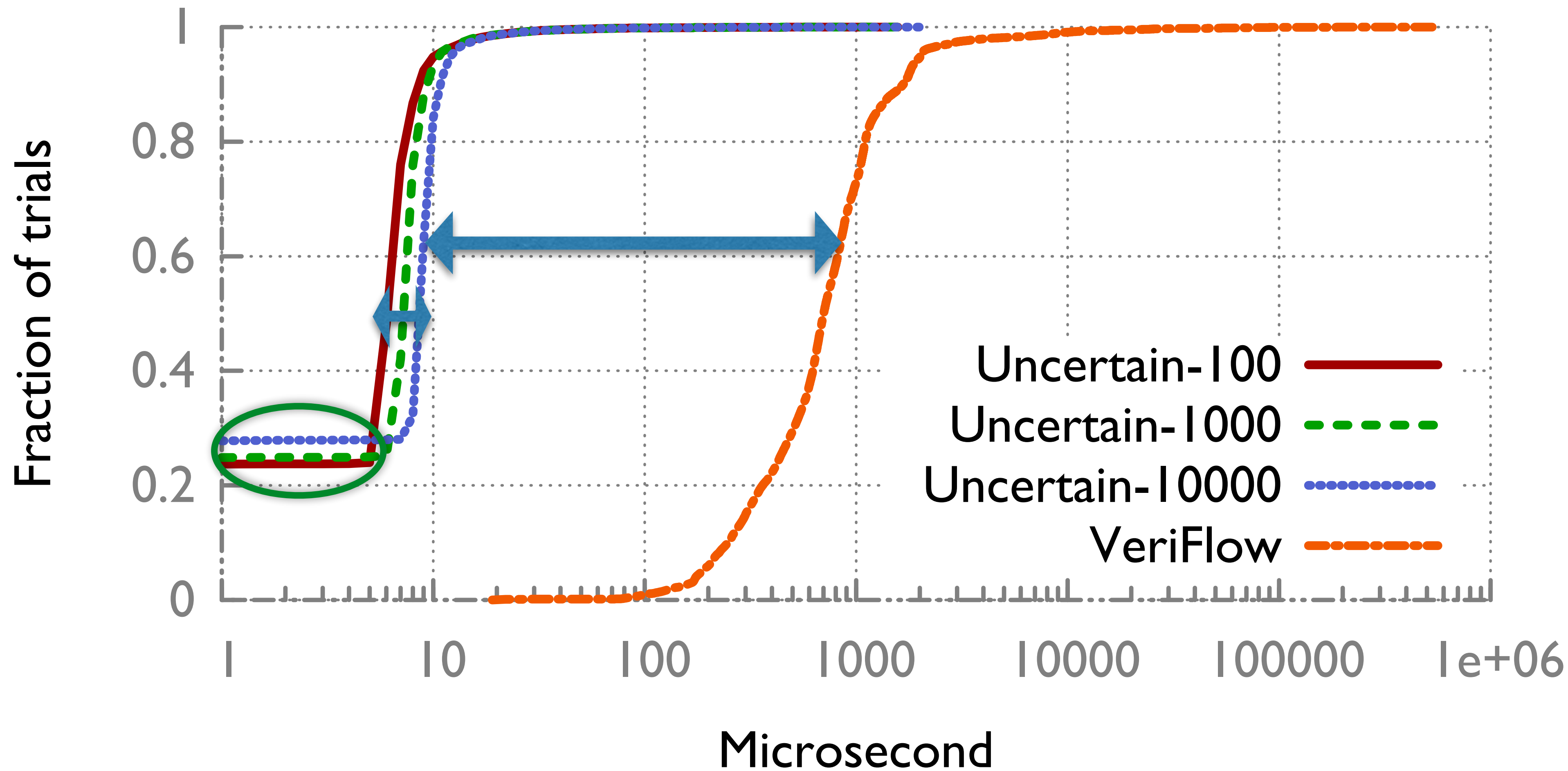
Can CCG verify network invariants in ***real time***?

Can CCG achieve ***performance gain*** during network transitions with its algorithm for maximizing the parallelism of applying updates?

- Segment-independent Policies
- Non-segment-independent Policies

- Emulations
- Testbed experiments

Speed Analysis



15X less memory overhead (540MB vs. 9GB)

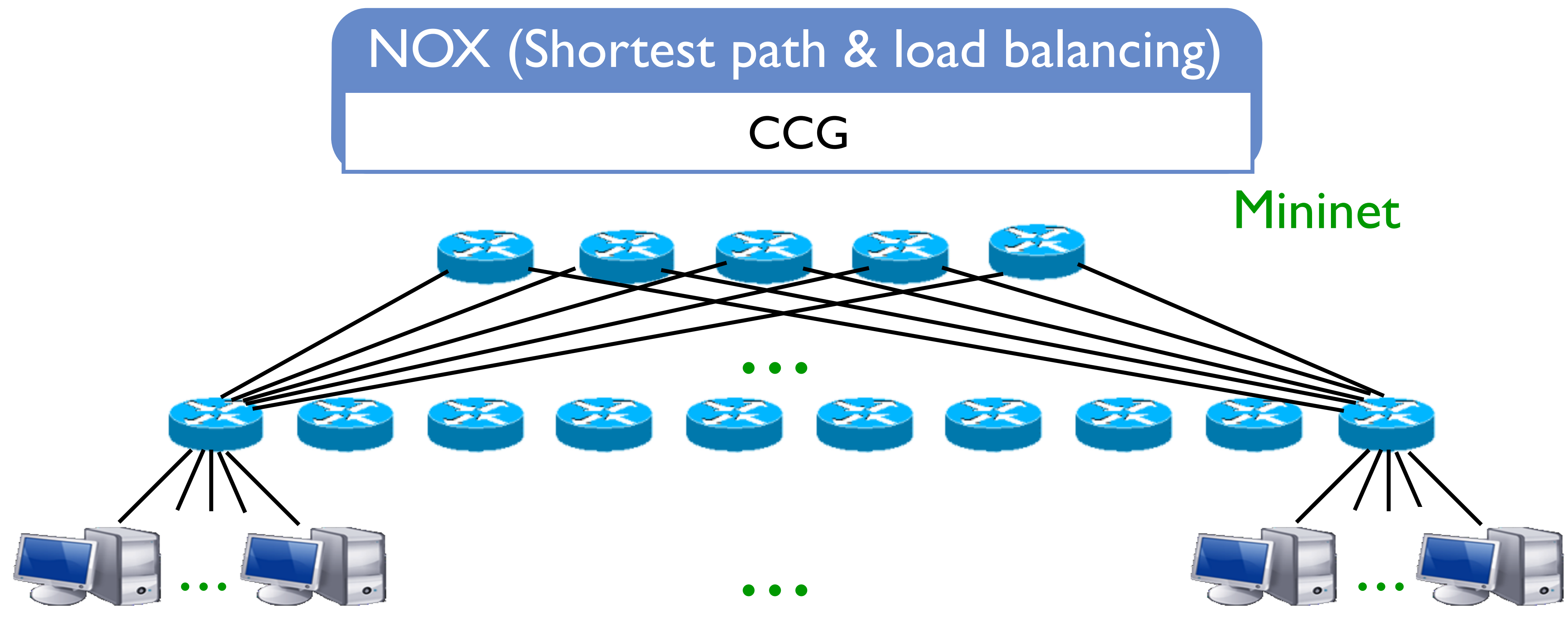
Simulated network: BGP RIBs and update trace from RouteViews injected into 172-router AS 1755 topology, checking reachability invariant

Emulation: Segment-independent Policies

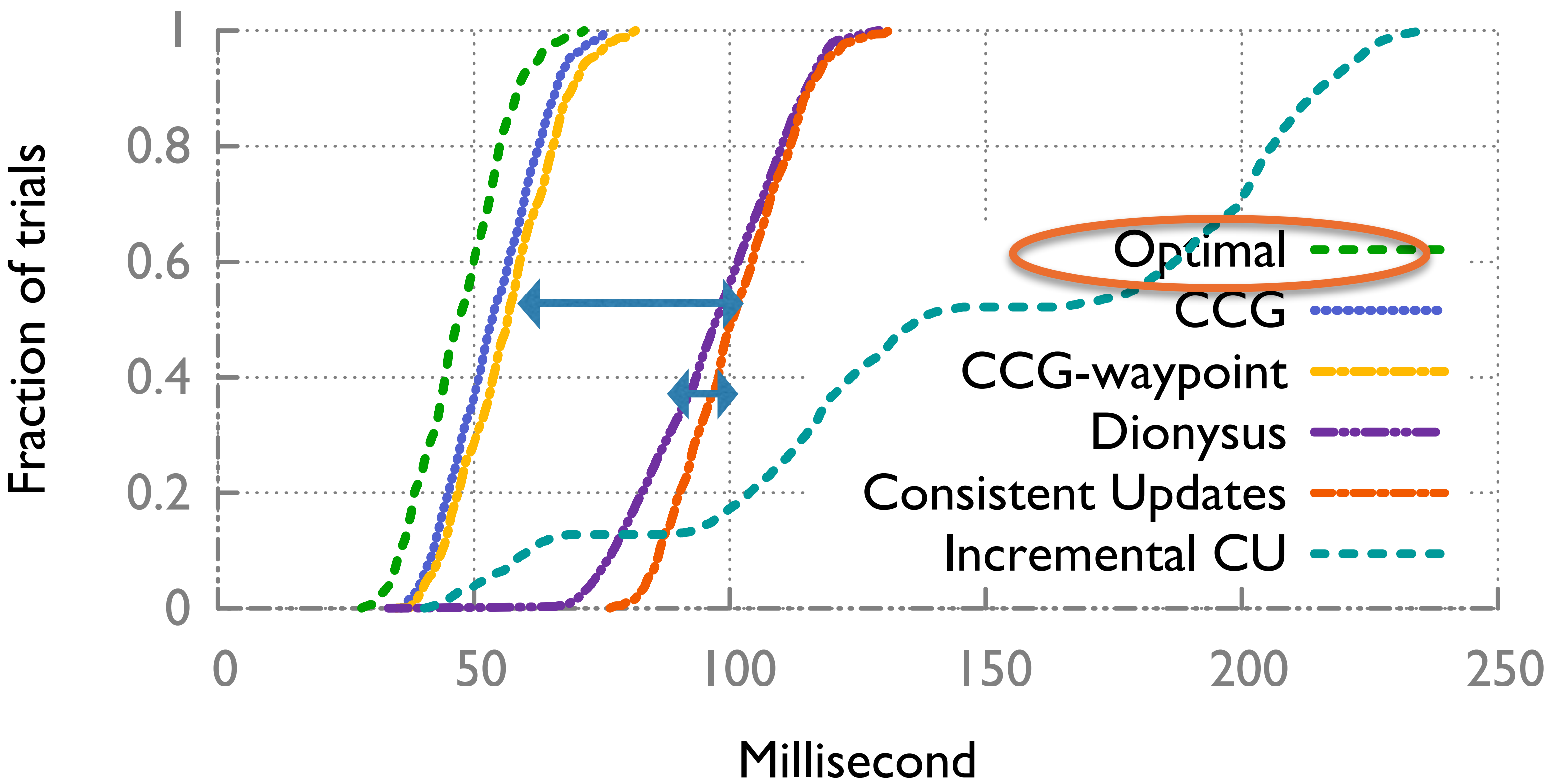
Controller-switch delay = network delay + processing delay

- Local (4ms)
- Wide area (100ms)

Measure: path completion time



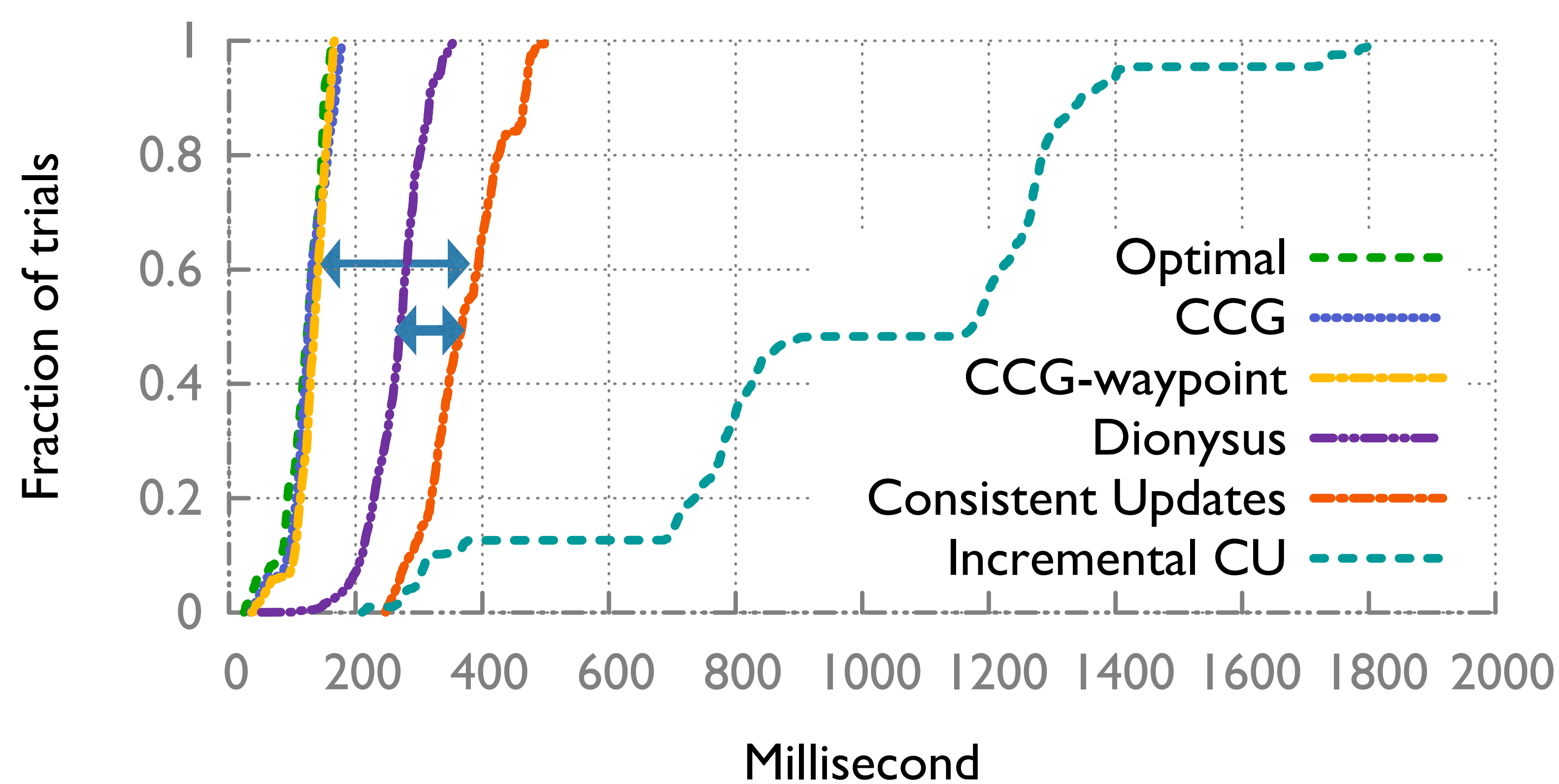
Emulation: Segment-independent Policies



No fallback triggered
No additional memory

Local

Wide area



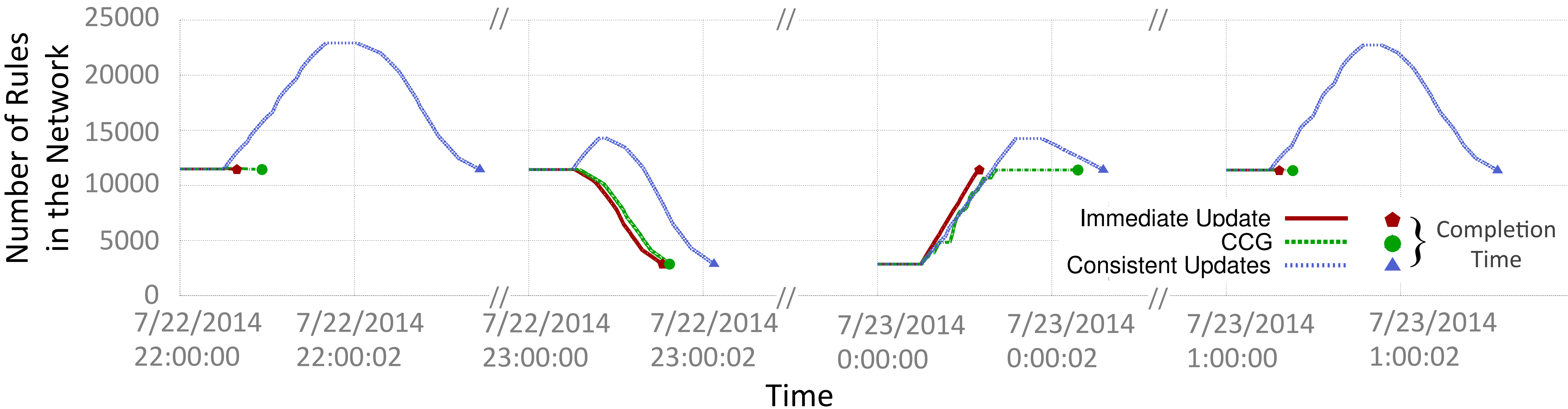
Emulation: Non-segment-independent Policies

Traces from a enterprise network with 200+ layer-3 devices.

One day, one snapshot per hour, 24 transitions, 4ms delay.

- New rules were added first, then old rules deleted.

Rules overlapped with longest prefix match, not segment-independent.



Fallbacks happened rarely.

Overhead close to Immediate Update, with no transient connectivity violations.

Conclusion

Uncertainty problem with network control

Uncertainty-aware network model

GCC, a system that

- enforces customizable network consistency properties with
- heuristically optimized efficiency.

Ongoing work:

- Study the generality of *segment independency*
- Test with more data traces, and compare against the original implementation of Dionysus
- Handle changes initiated from the network.