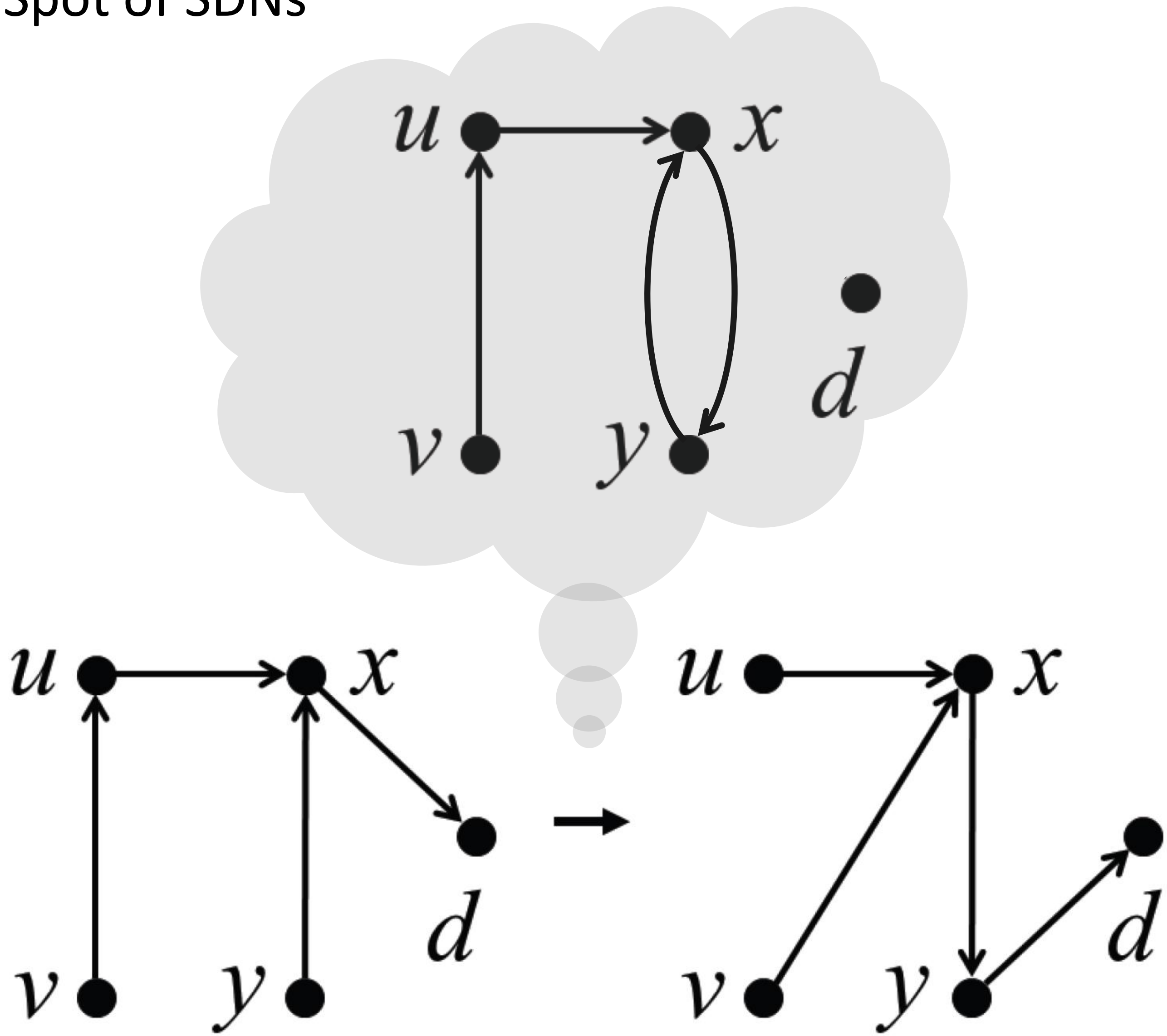
On Consistent Updates in Software Defined Networks

Ratul Mahajan, Microsoft Research
Roger Wattenhofer, Microsoft Research & ETH Zurich

The Blind Spot of SDNs

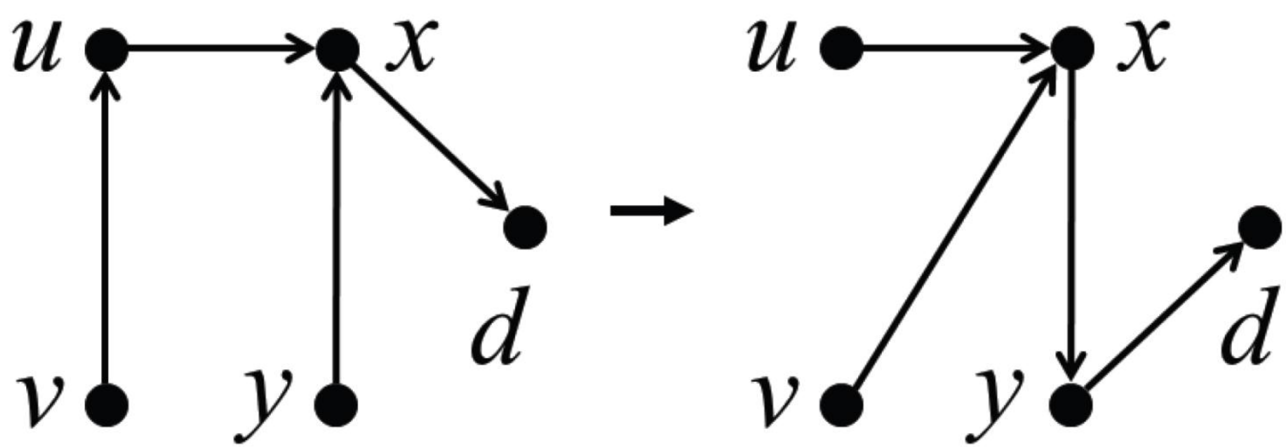


Current State

Future State

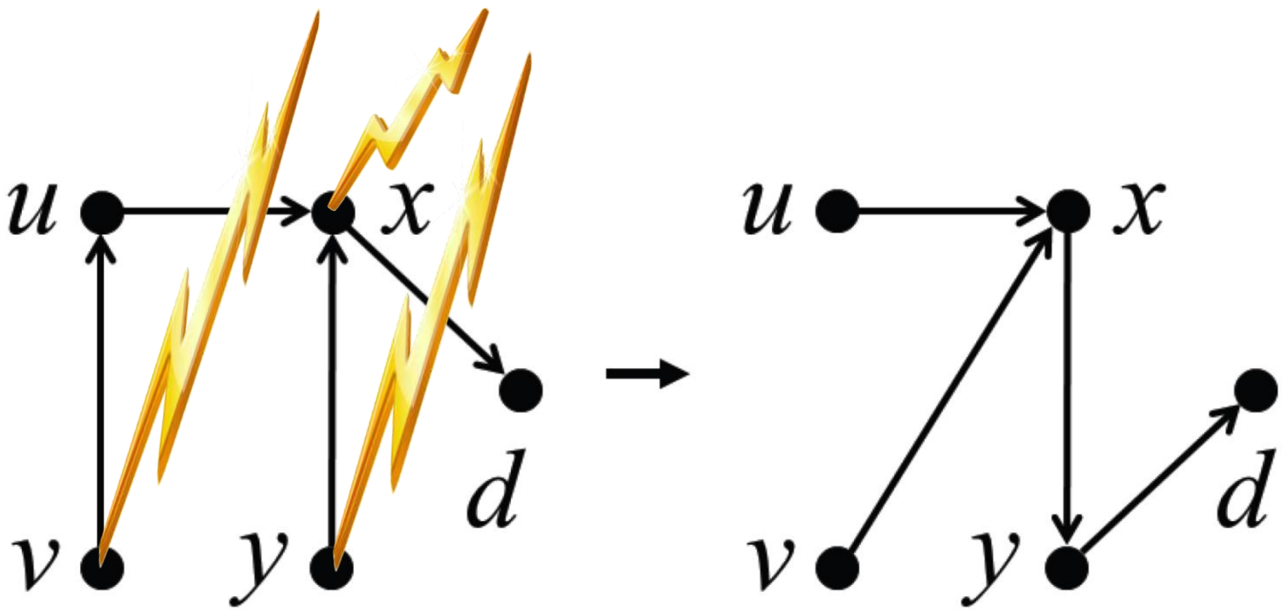
Example

SDN Controller



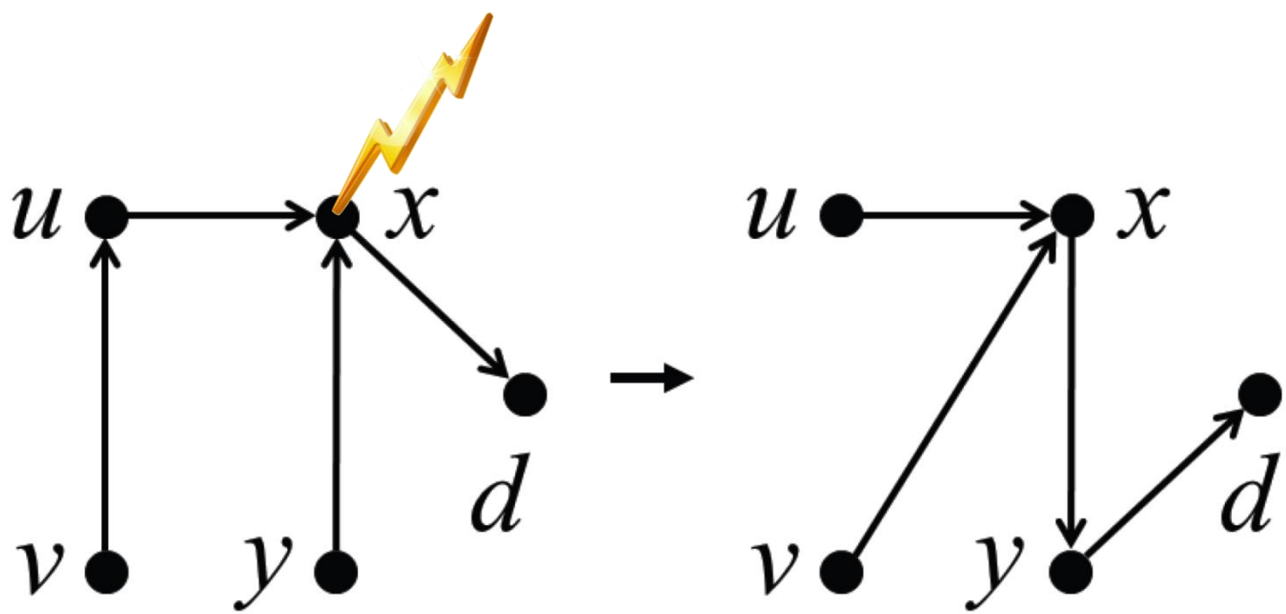
Example

SDN Controller

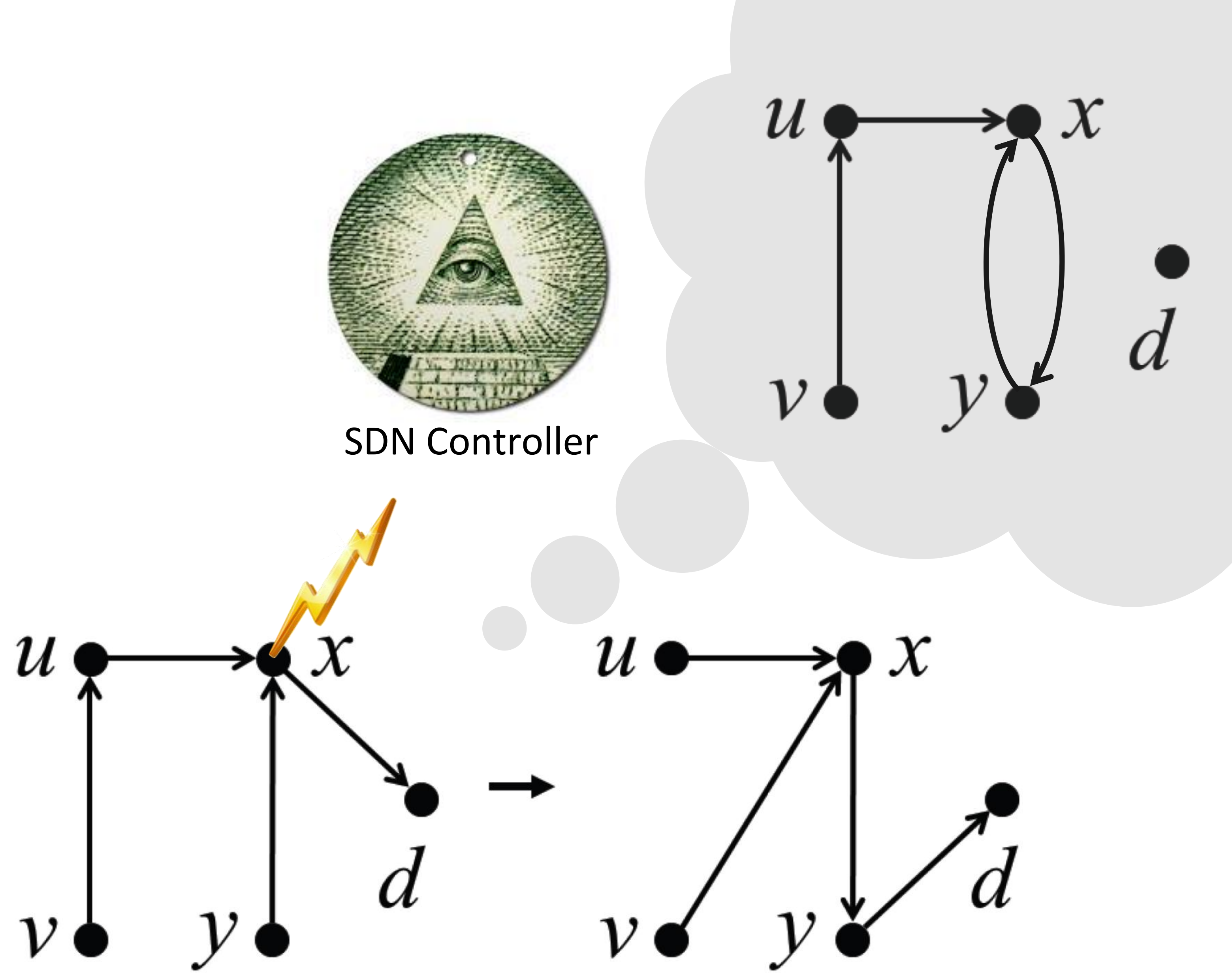


Example

SDN Controller

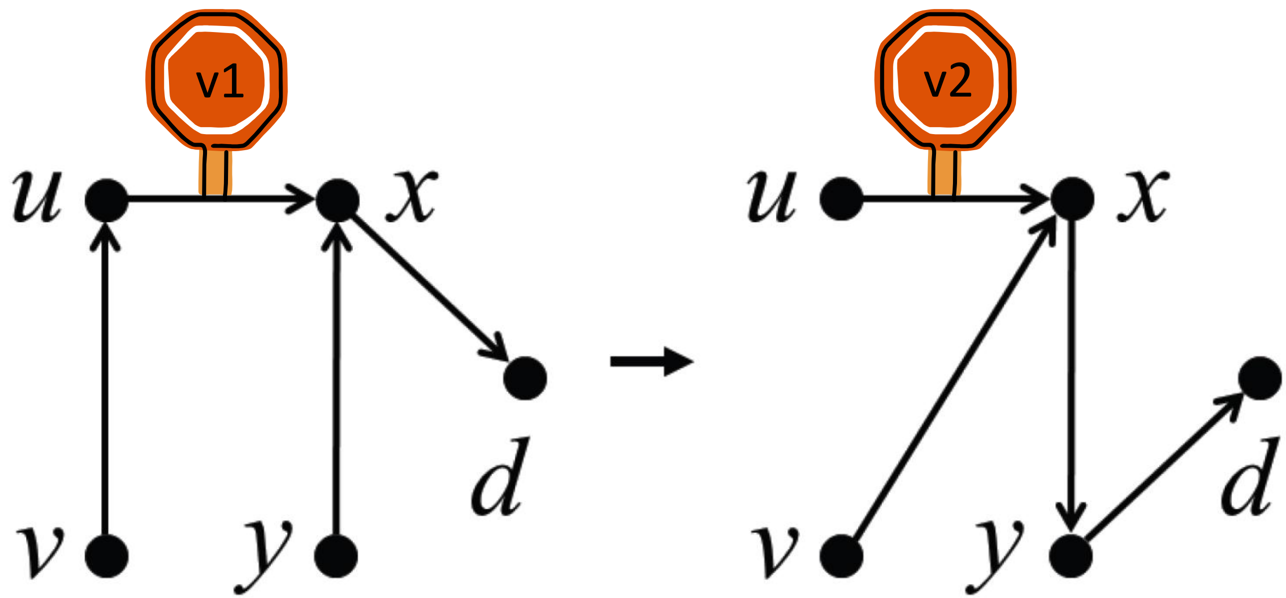


Example



Example

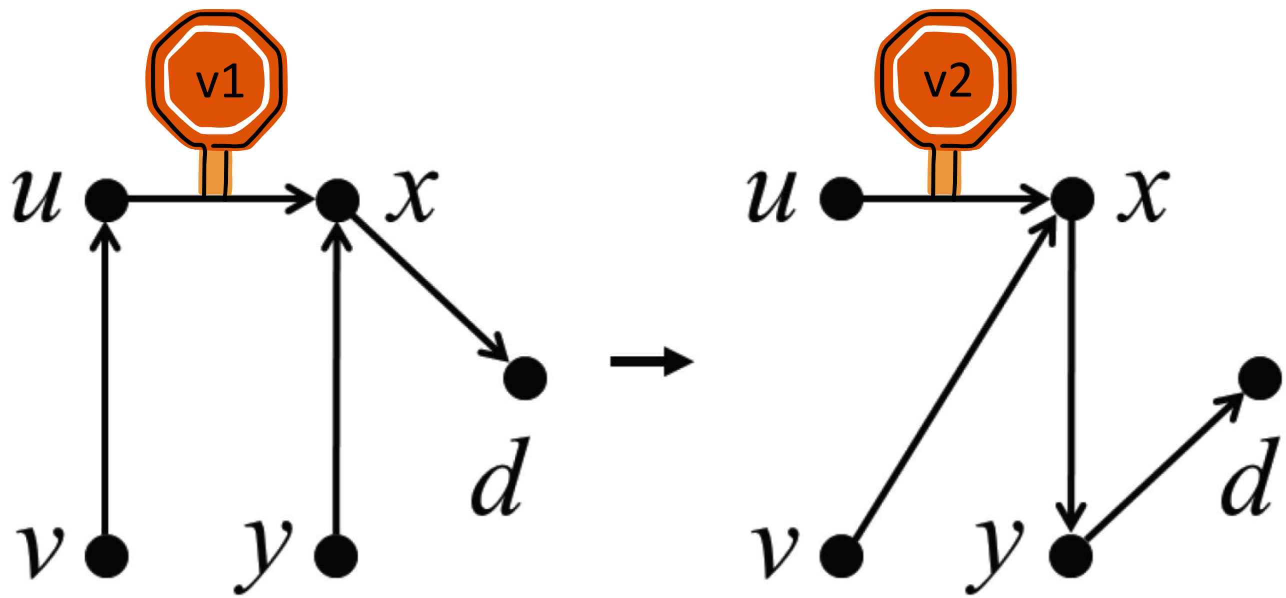
SDN Controller



[Reitblatt et al., SIGCOMM 2012]

Example

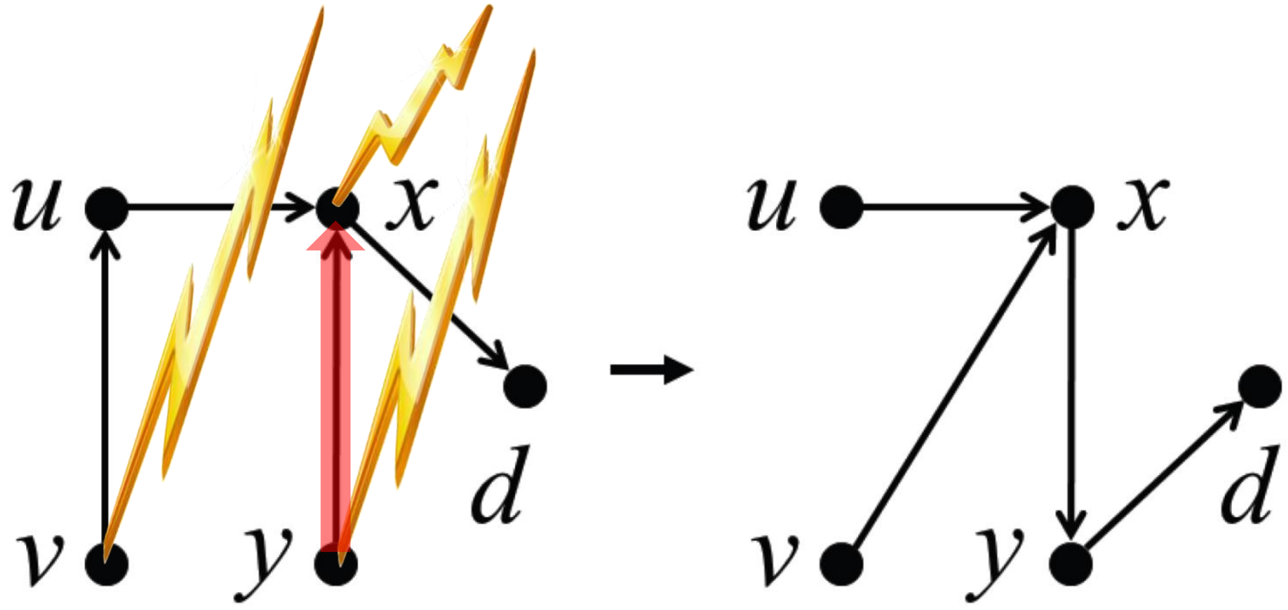
SDN Controller



[Reitblatt et al., SIGCOMM 2012]

Example

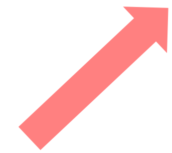
SDN Controller

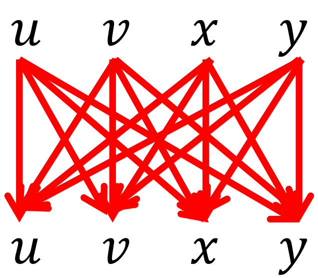


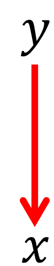
Dependencies

Version Numbers
[Reitblatt et al.]

“Better” Solution
[This paper]



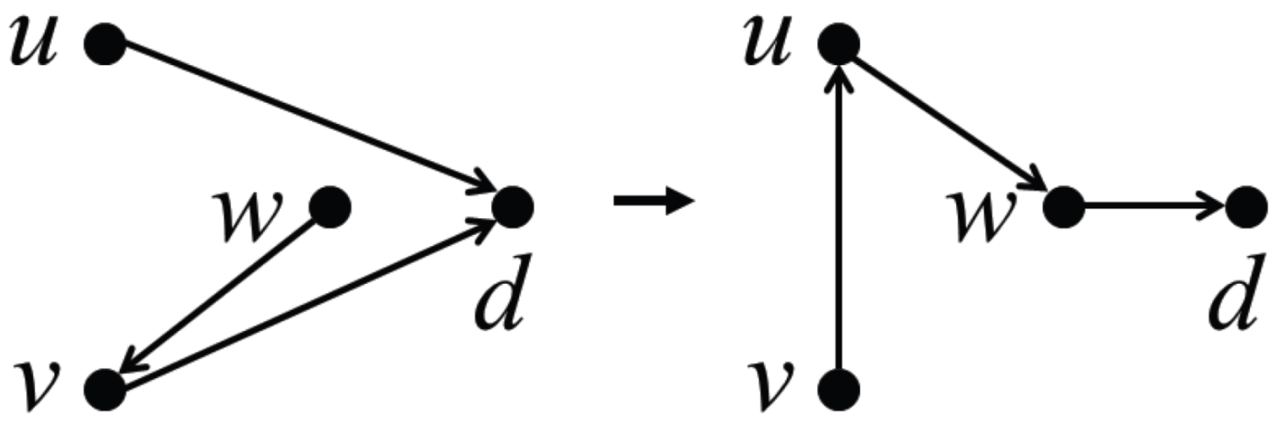


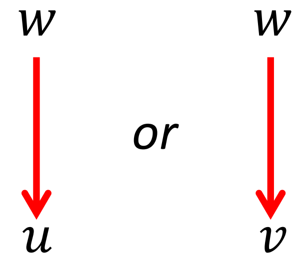


+ stronger packet coherence
– version number in packets
– switches need to store both versions

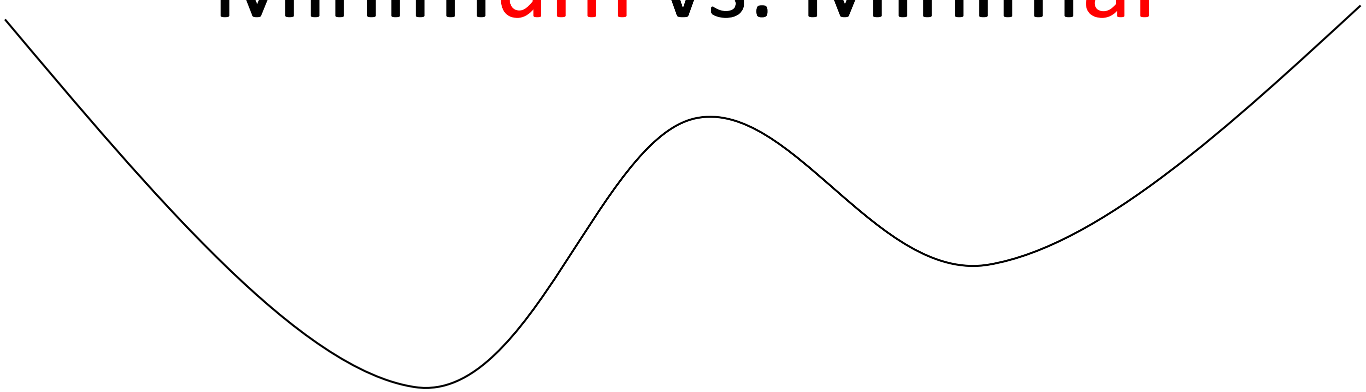
Minimum SDN Updates?

Minimum Updates: Another Example



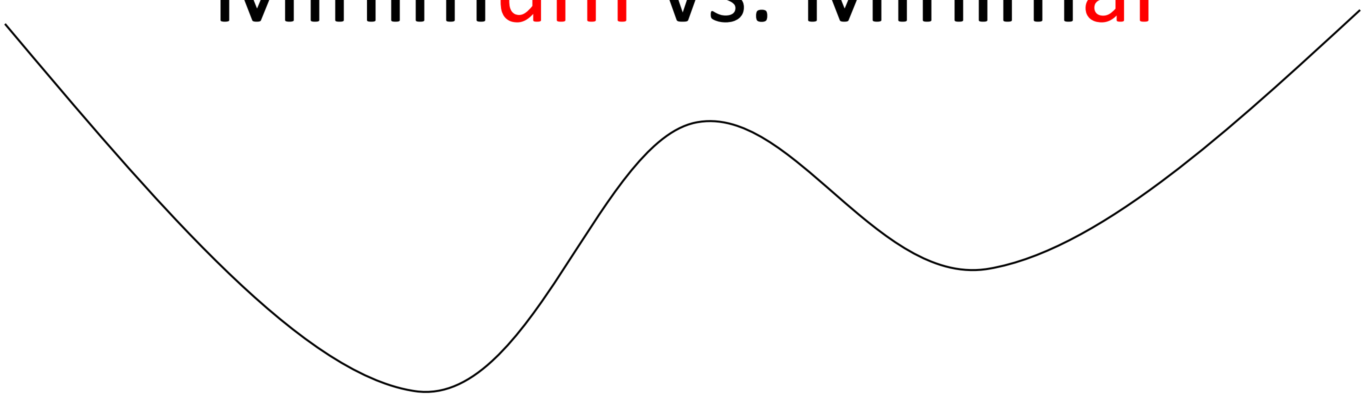


Minimum vs. Minimal

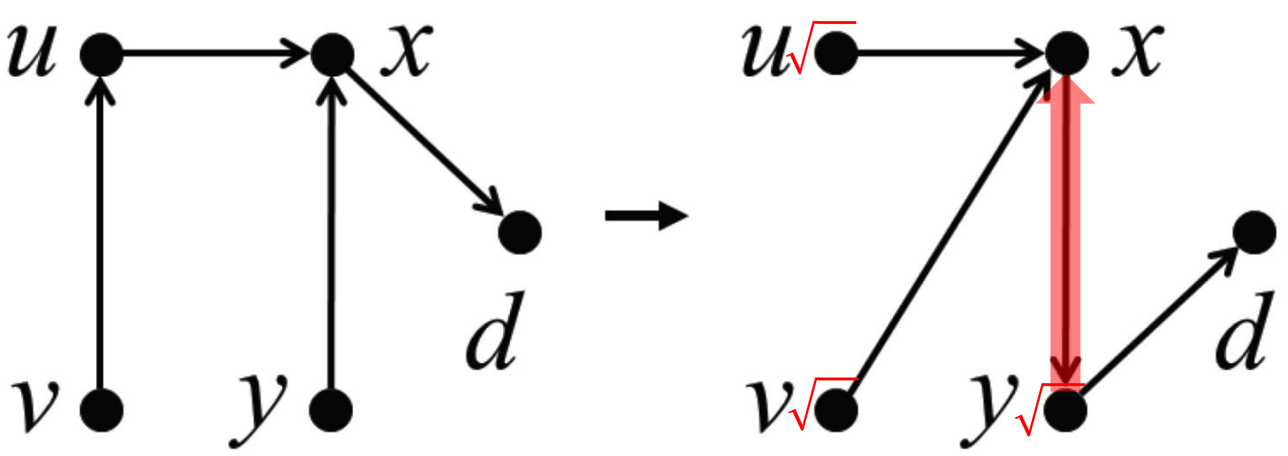


No node can improve without hurting another node

Minimum vs. Minimal



Minimal Dependency Forest



In the paper, we present an algorithm to compute such a minimal dependency forest.

Main Contribution

For a given consistency property, what is the minimal dependency possible?

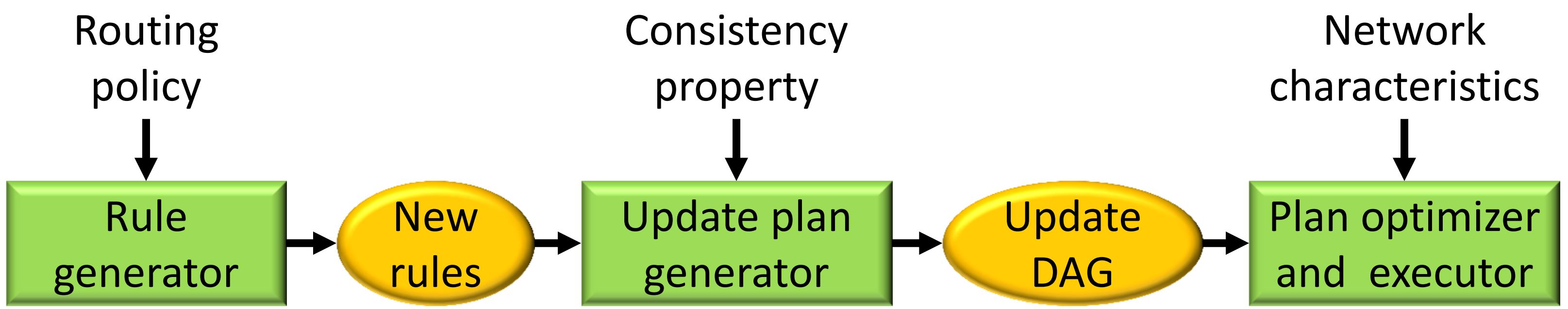
Consistency Space

	None	**Self**	**Downstream subset**	**Downstream all**	**Global**
Eventual consistency	Always guaranteed				
Drop freedom	Impossible	Add before remove			
Memory limit	Impossible	Remove before add			
Loop freedom	Impossible (Lemma 6)		Rule dep. forest (§2.2)	Rule dep. tree (§2.1)	
Packet coherence	Impossible (Lemma 7)			Per-flow ver. numbers	Global ver. numbers [8]
Bandwidth limit	Impossible (Lemma 8)				Staged partial moves [5]

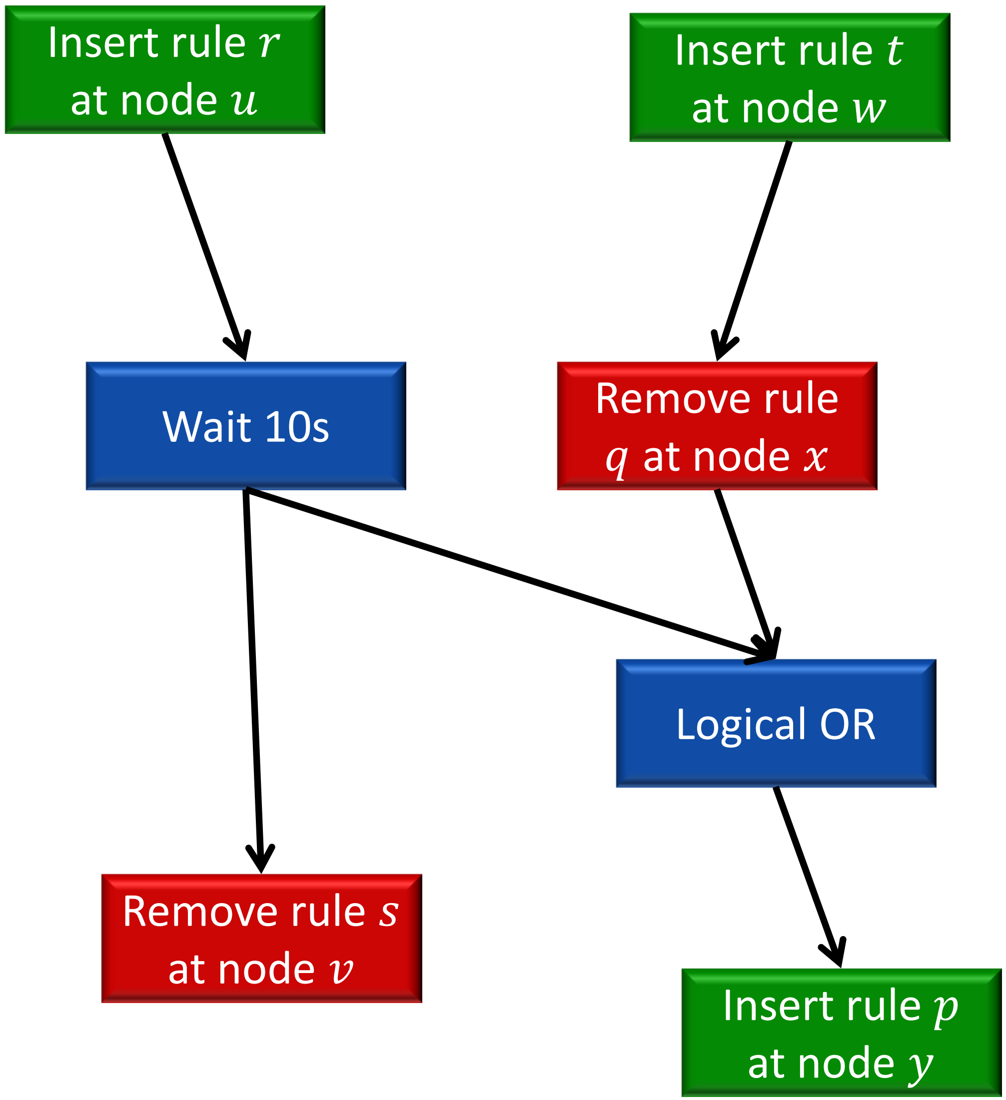
It’s *not* just how to compute new rules.

It is also how to gracefully get from current to new configuration, respecting consistency.

Architecture

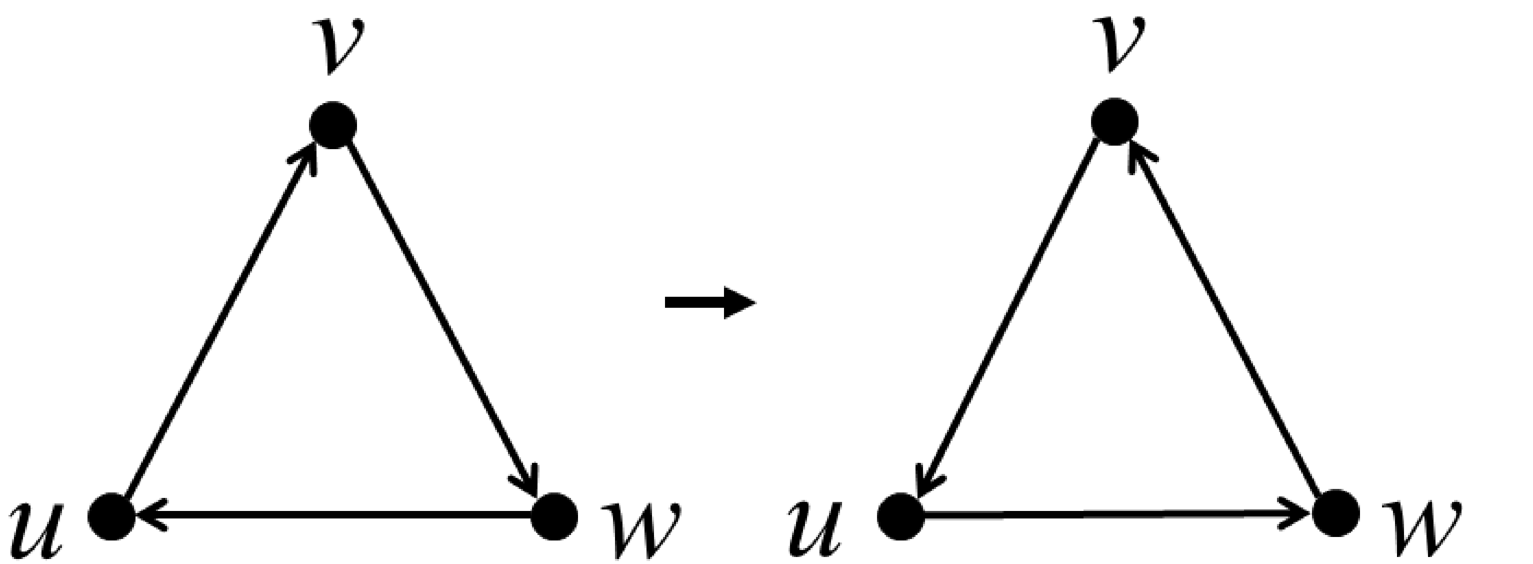


Update DAG



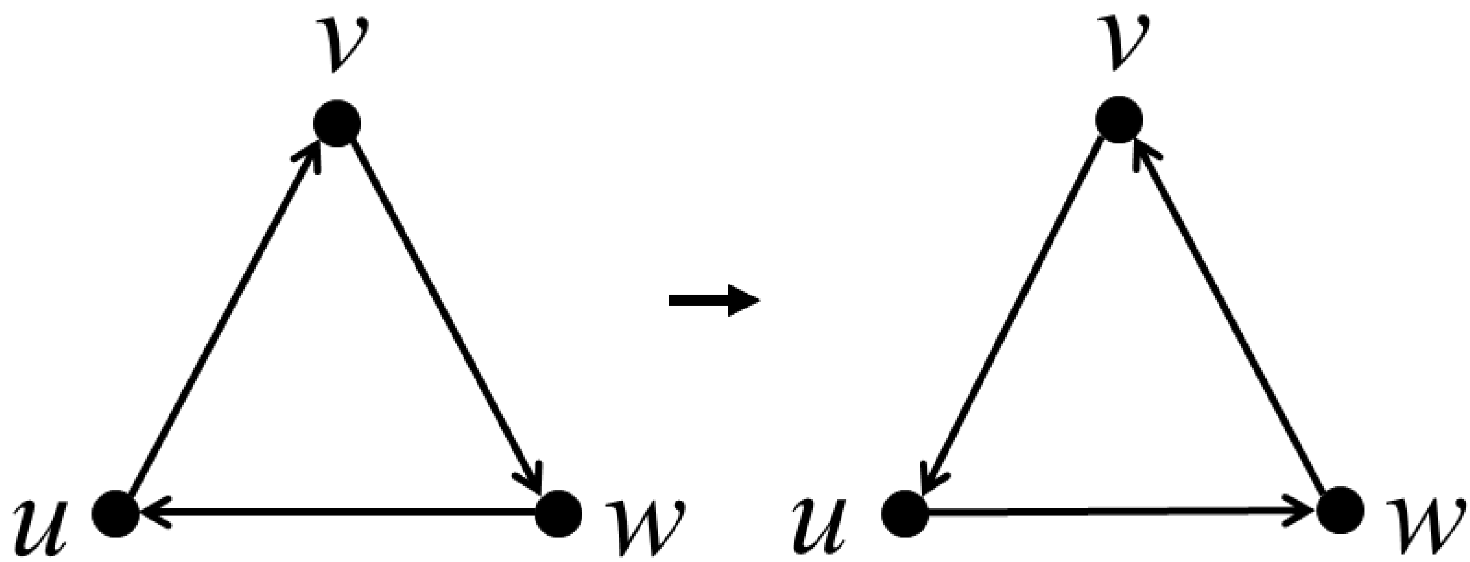
Insert rule r at node u
Insert rule t at node w
Wait 10s
Remove rule q at node x
Logical OR
Remove rule s at node v
Insert rule p at node y

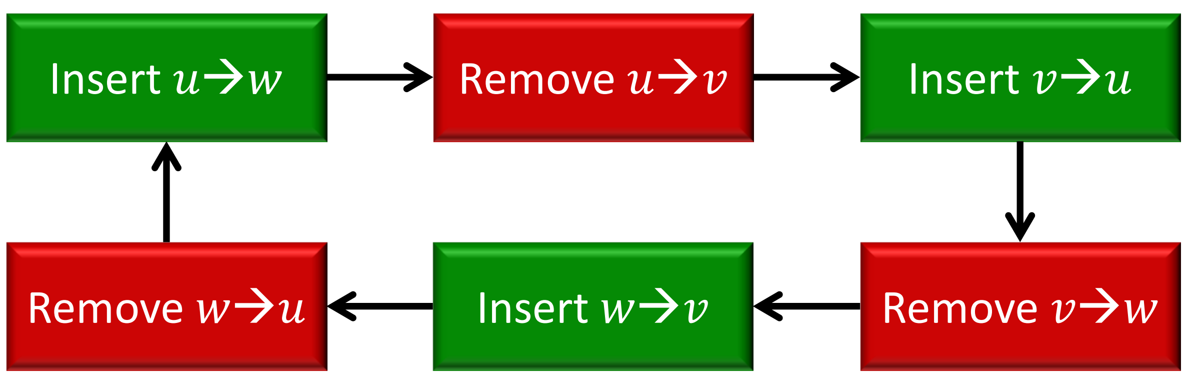
Multiple Destinations using Prefix-Based Routing



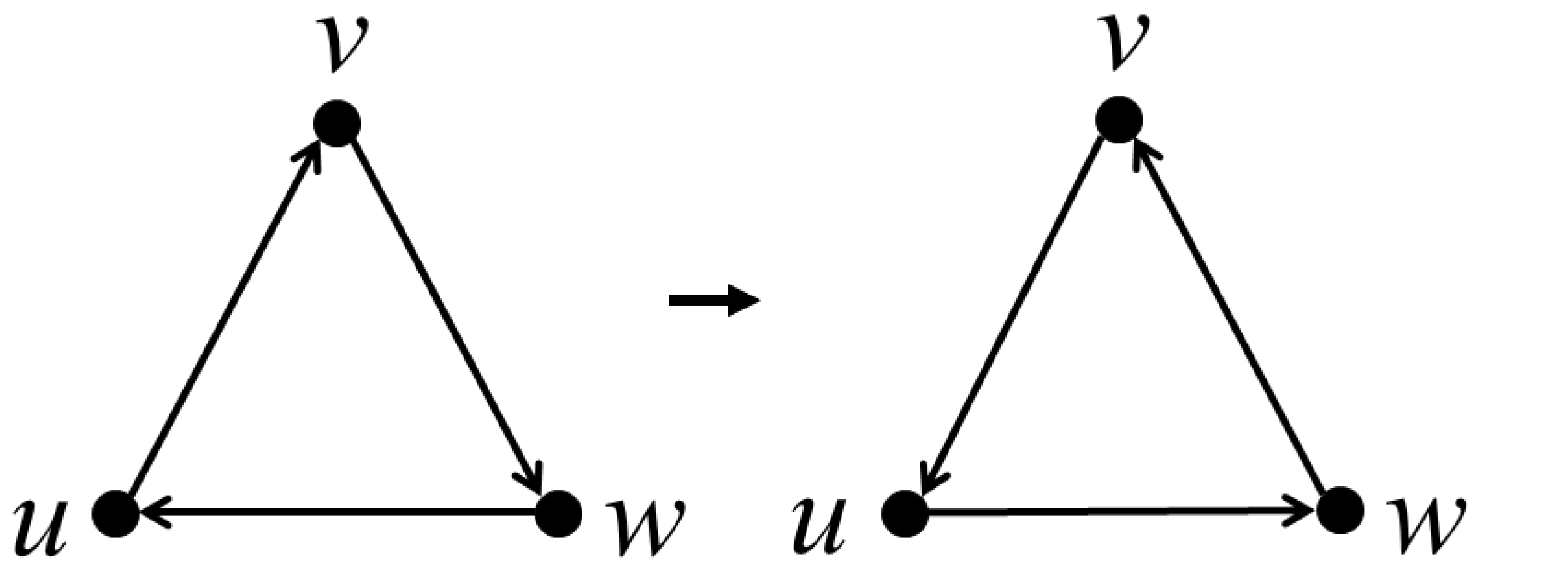
- No new “default” rule can be introduced without causing loops
- Solution: Rule-Dependency Graphs!
- Deciding if simple update schedule exists: [Vanbever et al., TON 2012]

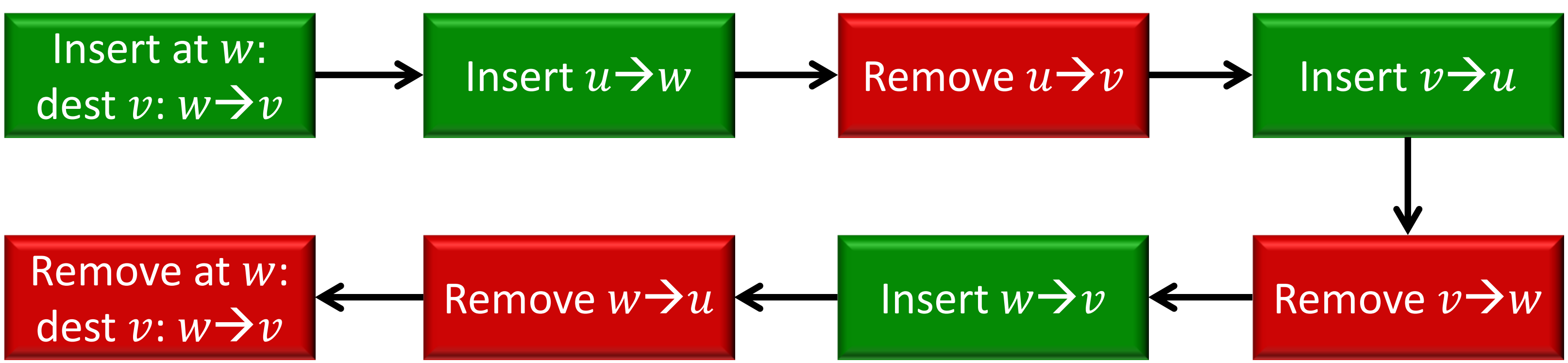
Breaking Cycles



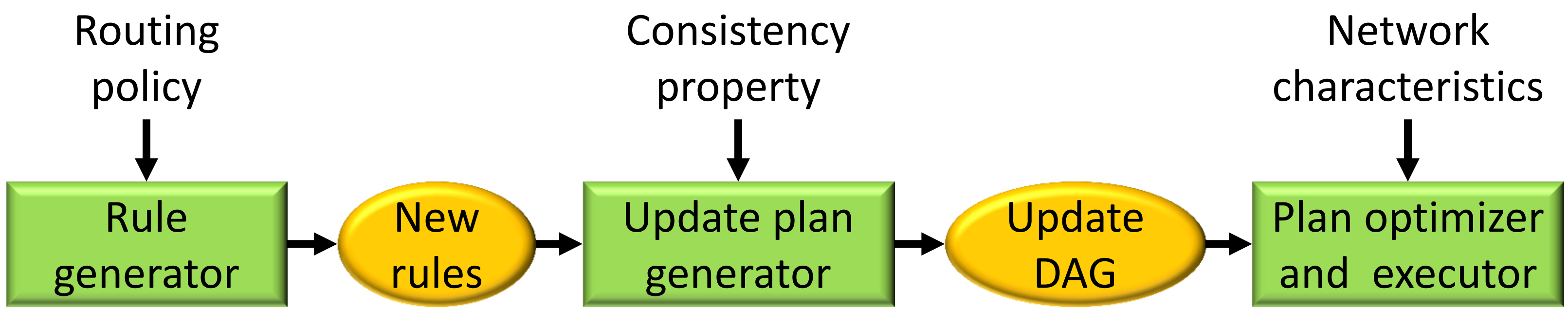


Breaking Cycles

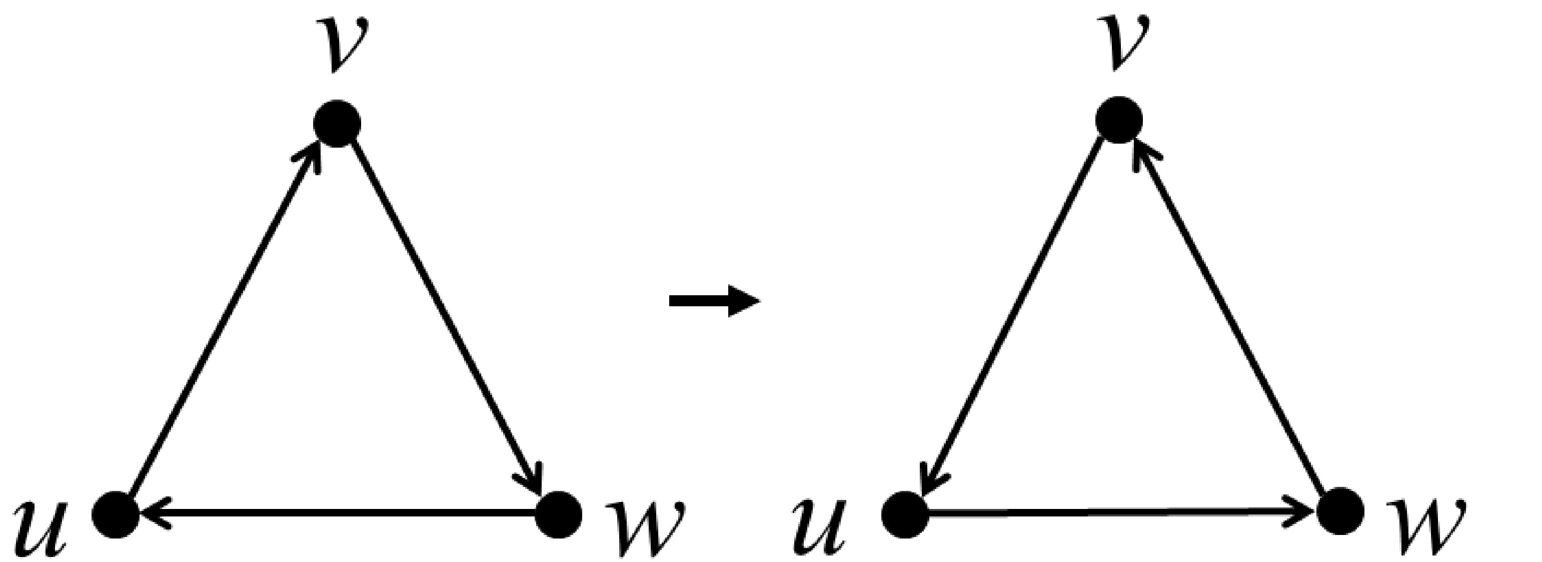


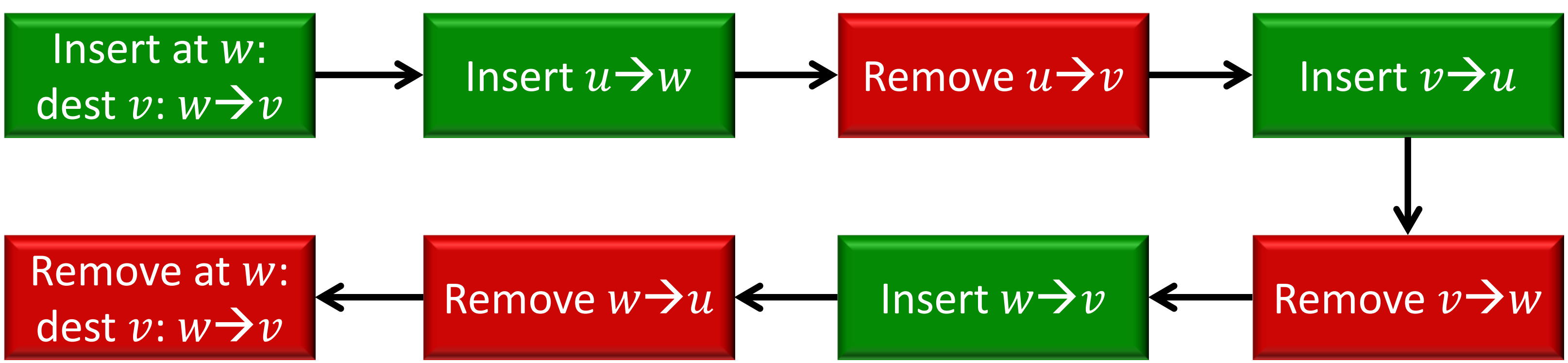


Architecture



Breaking Cycles





Are Minimal Dependencies Good?

Are Minimal Dependencies Good?

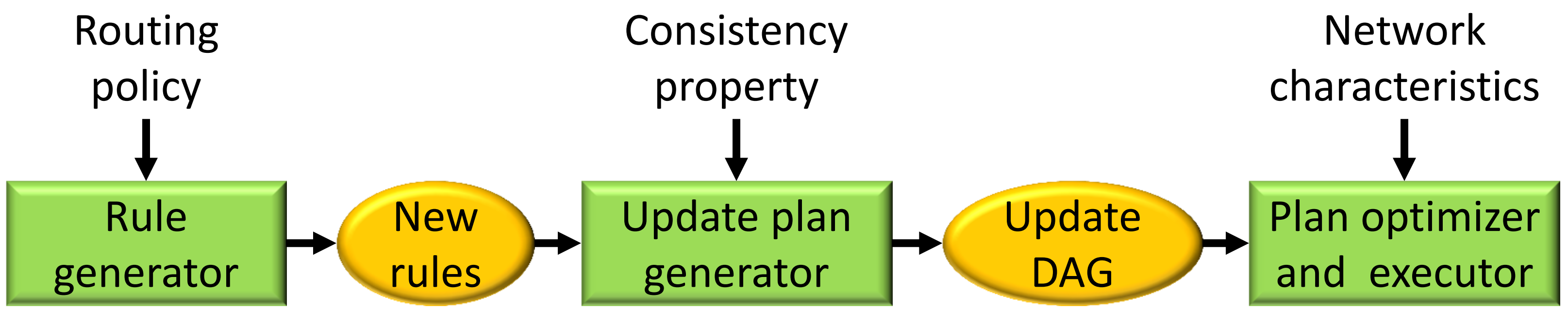
…it depends

Are Minimal Dependencies Good?

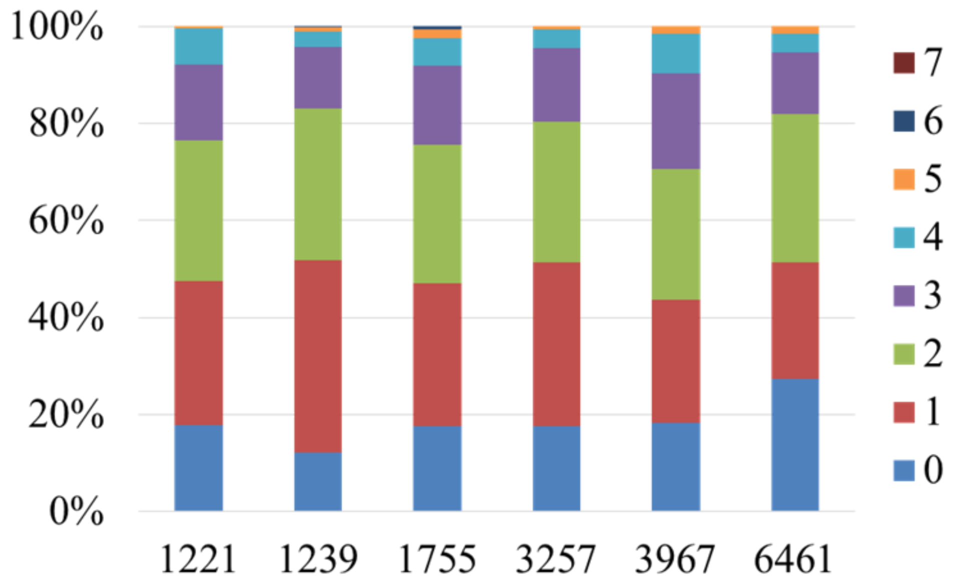
...it depends

(But Plan optimizer and executor will fix it.)

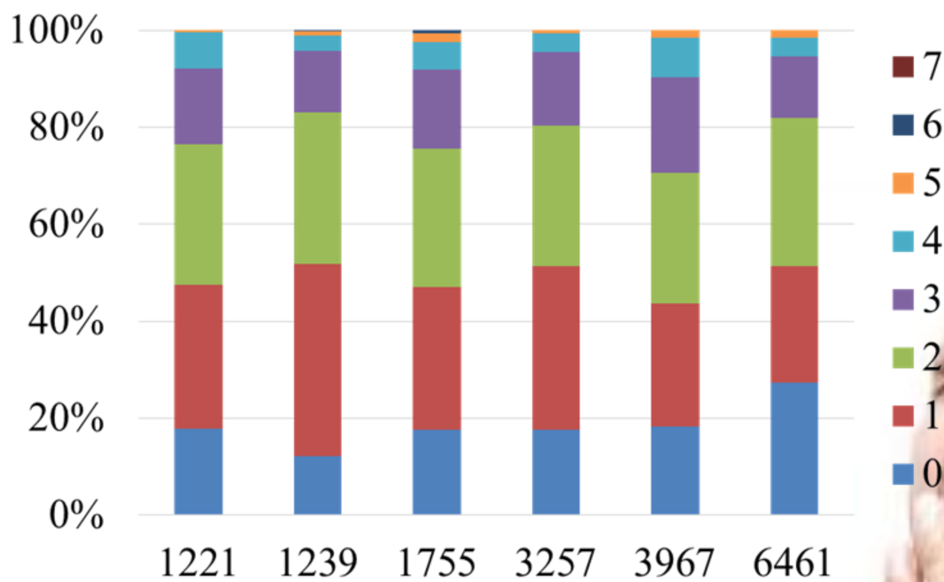
Architecture



Evaluation

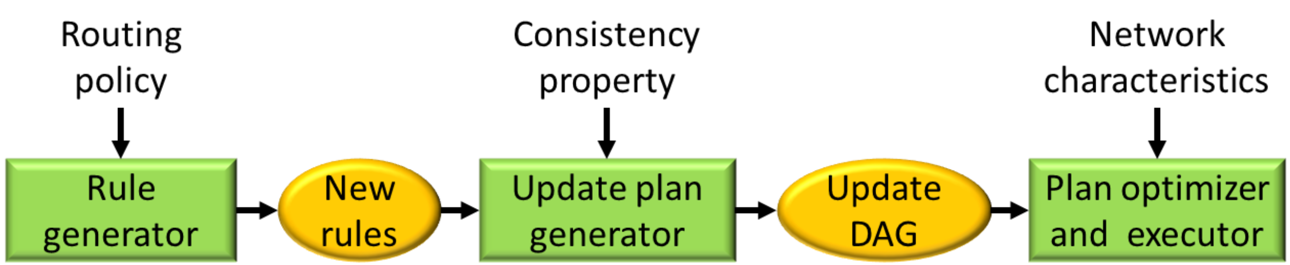


Evaluation



Summary

	None	Self	Downstream subset	Downstream all	Global
Eventual consistency	Always guaranteed				
Drop freedom	Impossible	Add before remove			
Memory limit	Impossible	Remove before add			
Loop freedom	Impossible (Lemma 6)		Rule dep. forest (§2.2)	Rule dep. tree (§2.1)	
Packet coherence	Impossible (Lemma 7)			Per-flow ver. numbers	Global ver. numbers [8]
Bandwidth limit	Impossible (Lemma 8)				Staged partial moves [5]



At ETH Zurich, we’re looking for a colleague in networking!
Please ask me for details.

Thank You!

Questions & Comments?