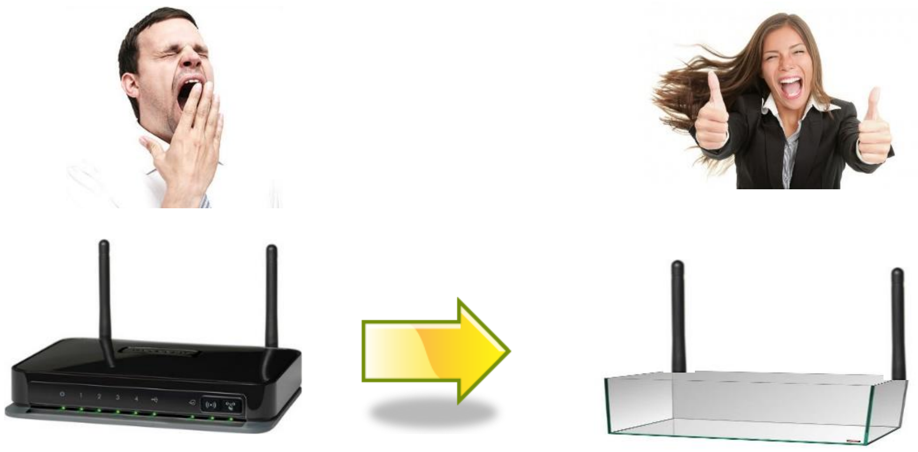
Distributed Consistent Network Updates in SDNs: Local Verification for Global Guarantees

Klaus-T. Foerster, Stefan Schmid

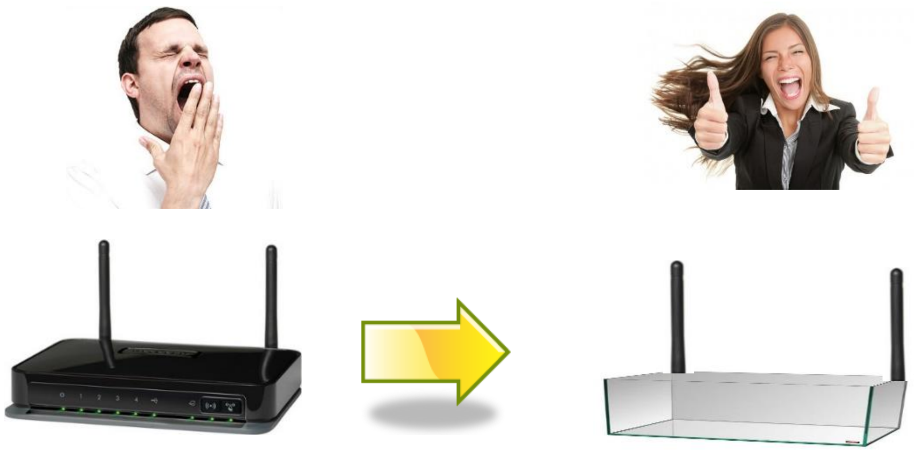
IEEE NCA 2019

Software-Defined Networking



universität wien

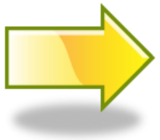
Software-Defined Networking



- General Idea: Separate data & control plane in a network

Software-Defined Networking




- General Idea: Separate data & control plane in a network
- Centralized controller updates networks rules for optimization
  - Controller (*control plane*) updates the switches/routers (*data plane*)

Virtual Services ⟷ Controller ⟷ Physical Network

- Logically centralized controller (eg implemented with replication)

Network Updates

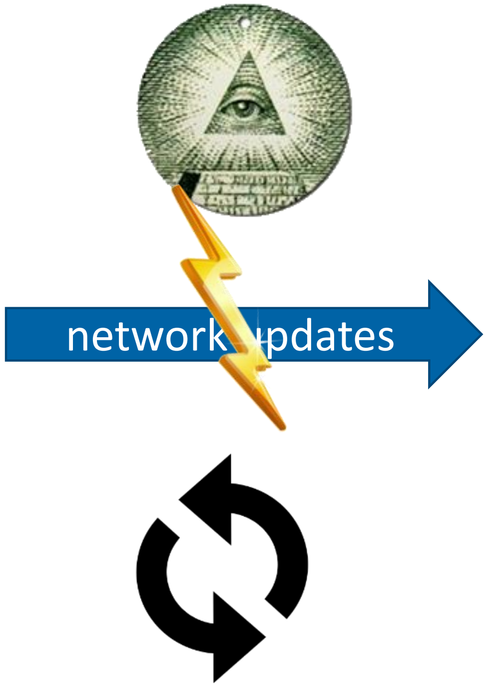
old network rules

network updates

new network rules

Network Updates

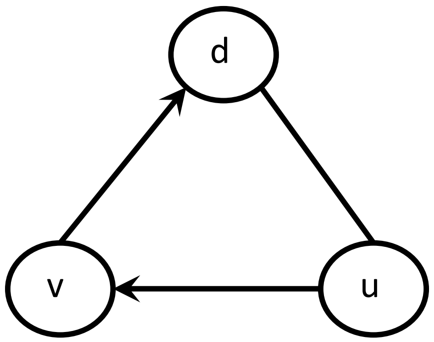
old network rules



new network rules

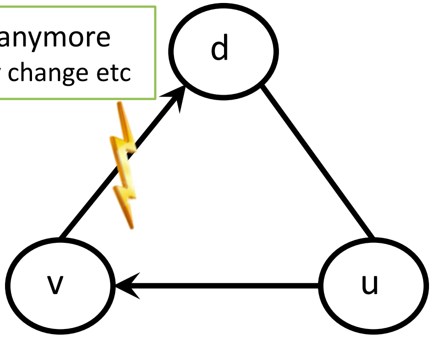
universität
wien

Toy Example

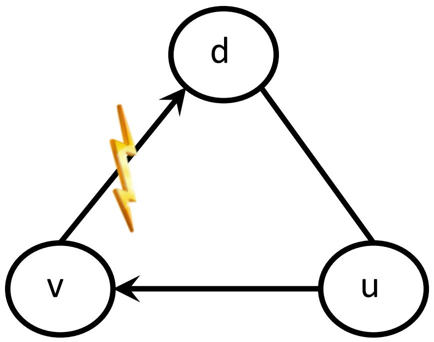


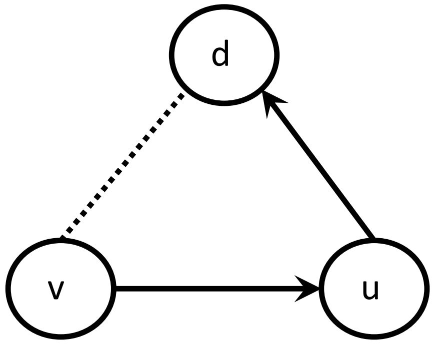
Toy Example

Link should not be used anymore
eg repair, congestion, policy change etc

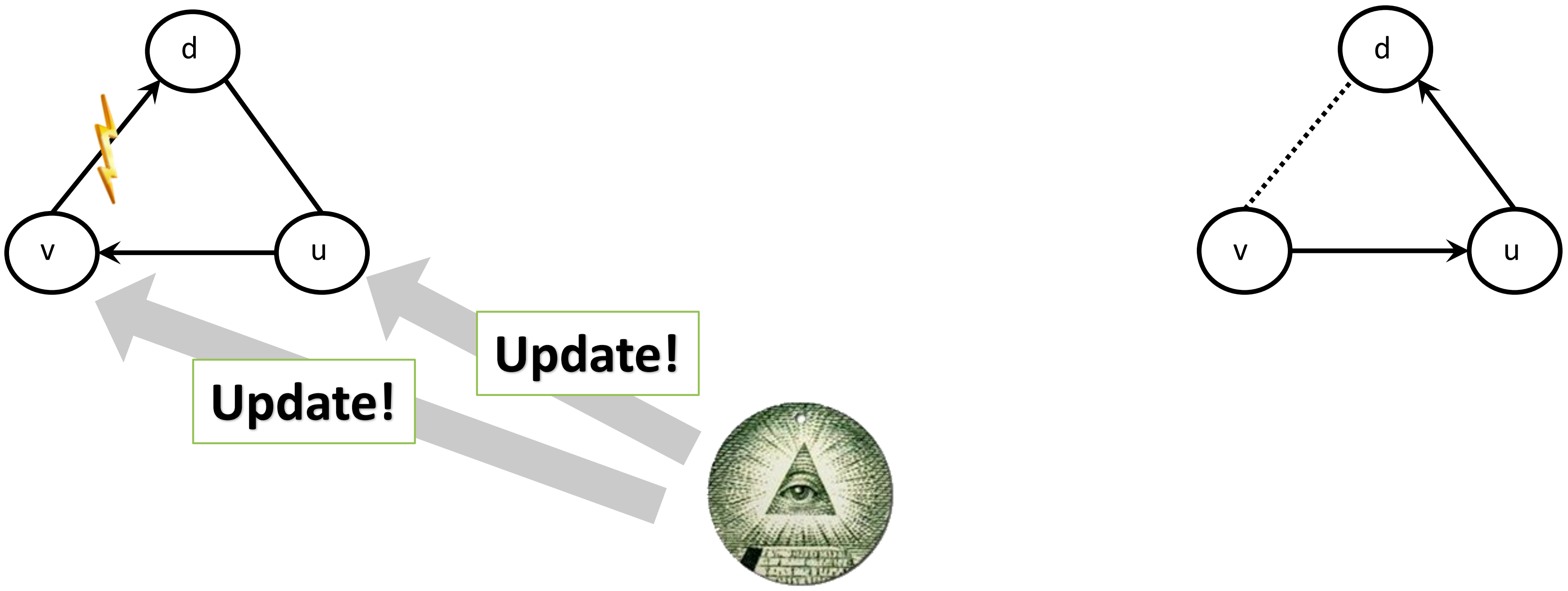


Toy Example



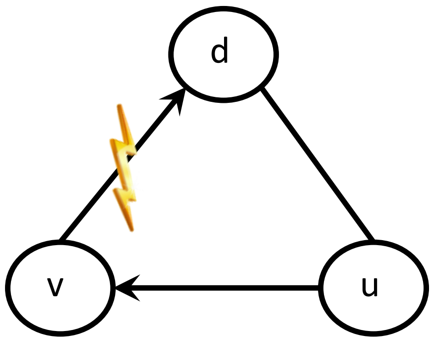


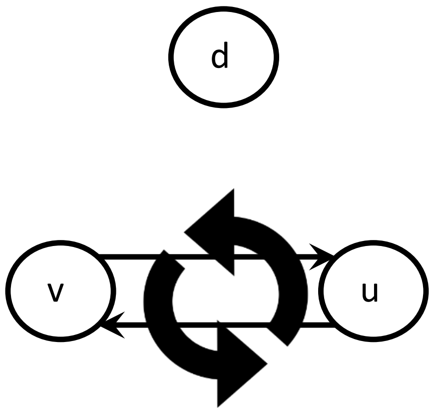
Toy Example

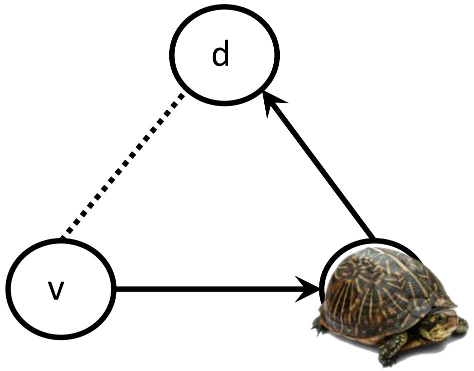


Toy Example

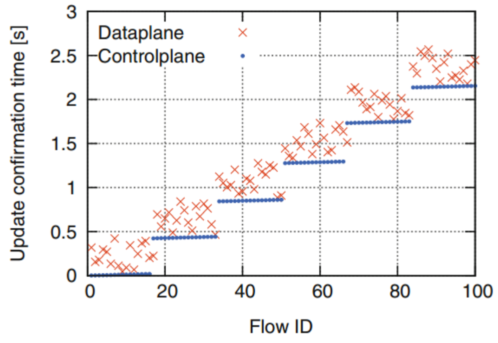
Toy Example



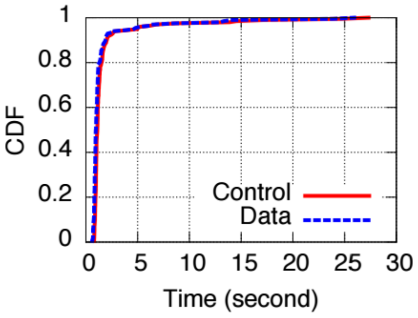




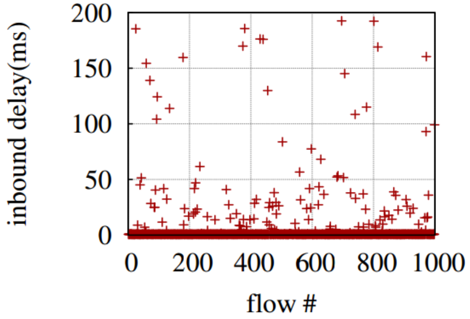
Appears in Practice



“*Data plane **updates may fall behind** the control plane acknowledgments and may be even **reordered**.*”
Kuzniar et al., PAM 2015

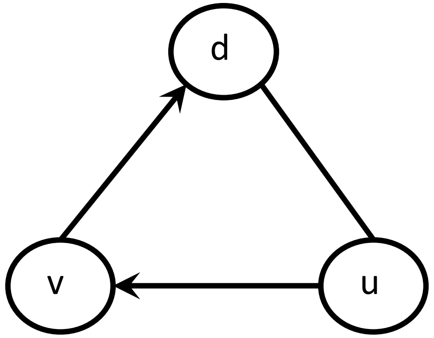


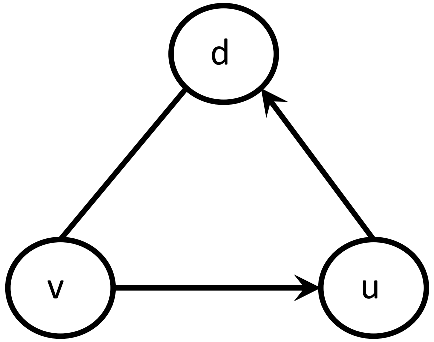
“*some switches can ‘**straggle**,’ taking substantially **more time** than average (e.g., 10-**100x**) to apply an update*”
Jin et al., SIGCOMM 2014



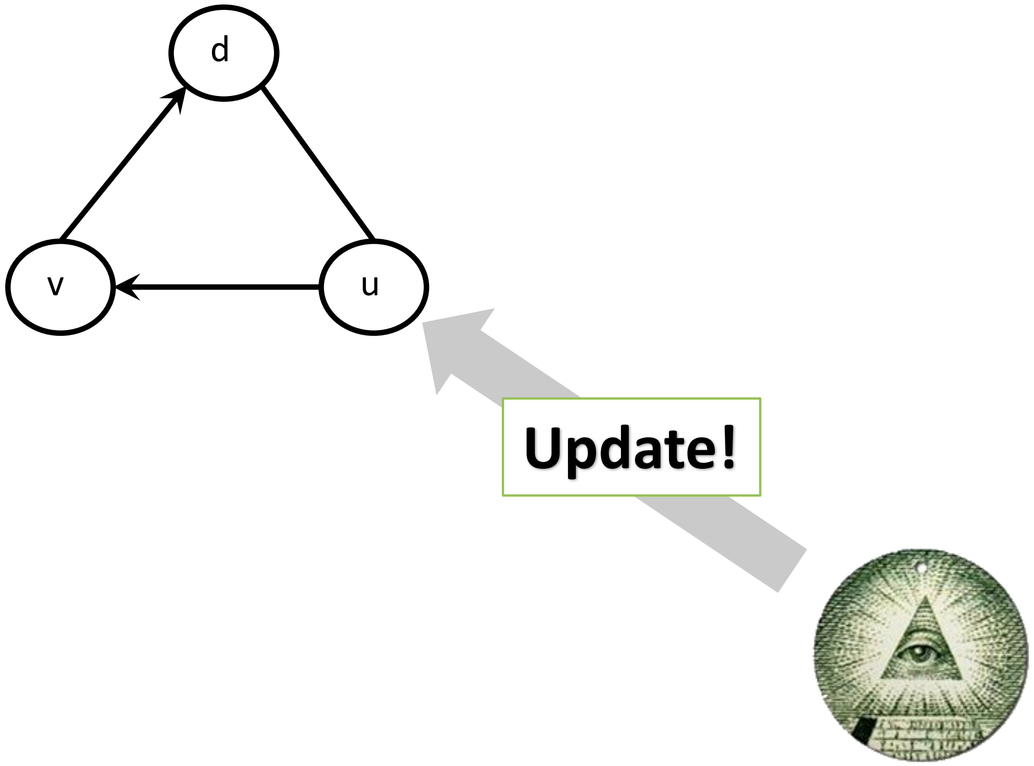
“*…the inbound latency is **quite variable** with a* […] *standard deviation of 31.34ms…*”
He et al., SOSR 2015

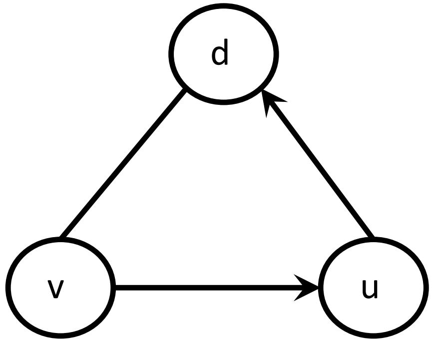
Ordering Solution: Go backwards through the new routing tree



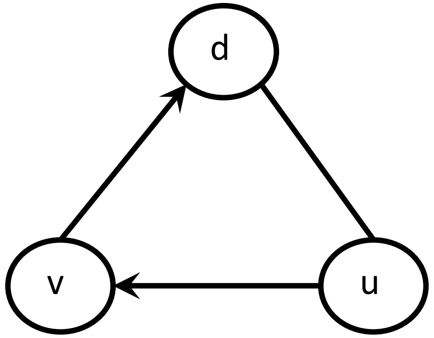


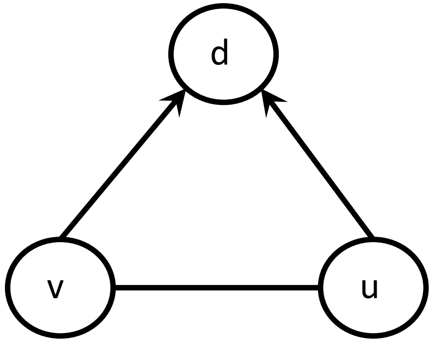
Ordering Solution: Go backwards through the new routing tree

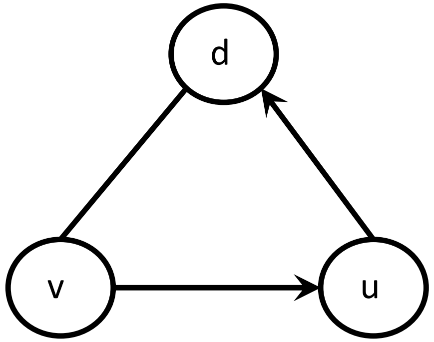




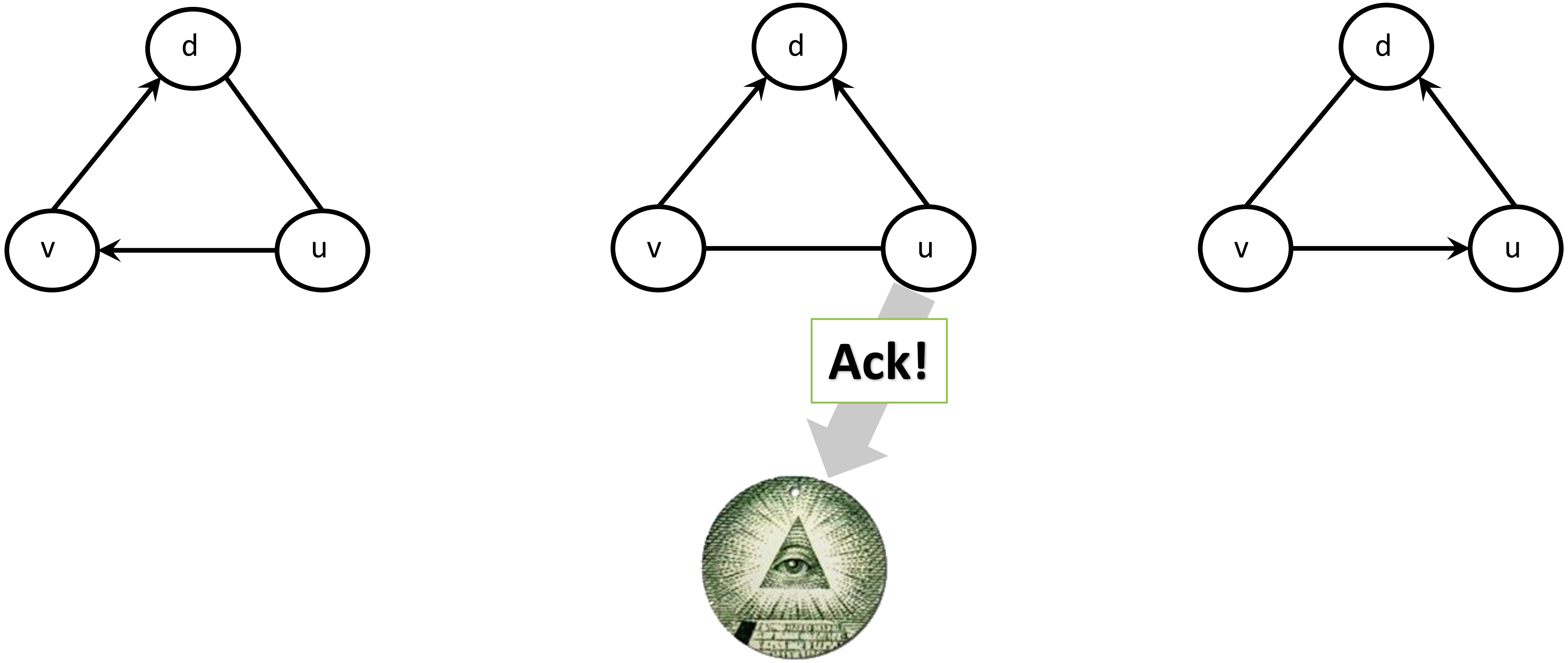
Ordering Solution: Go backwards through the new routing tree



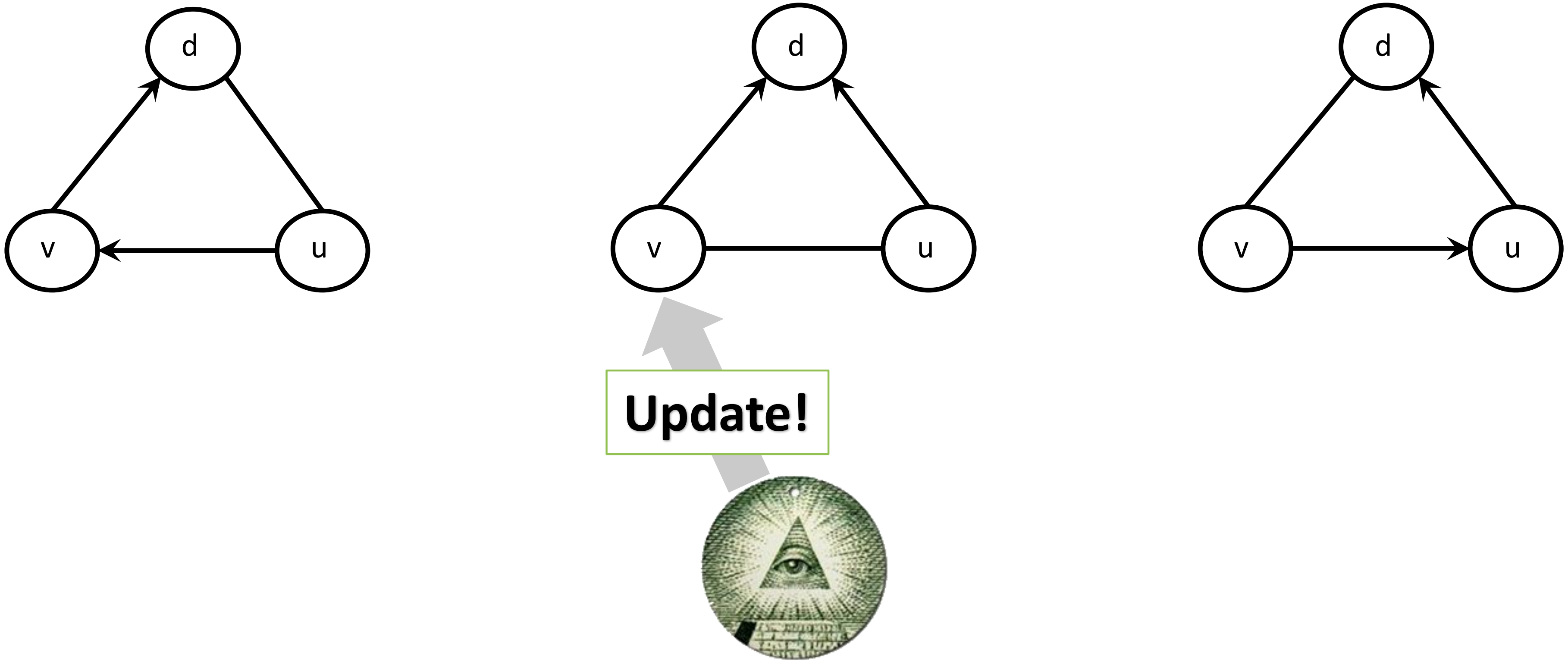




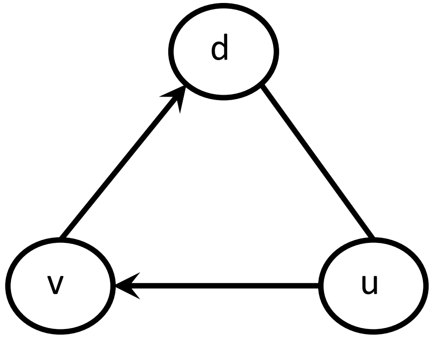
Ordering Solution: Go backwards through the new routing tree



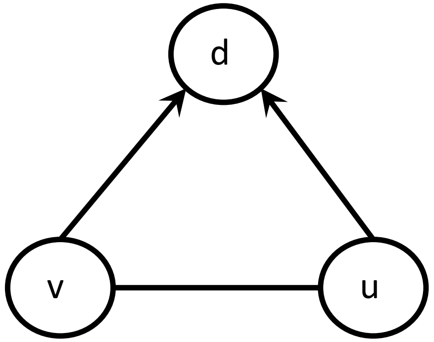
Ordering Solution: Go backwards through the new routing tree



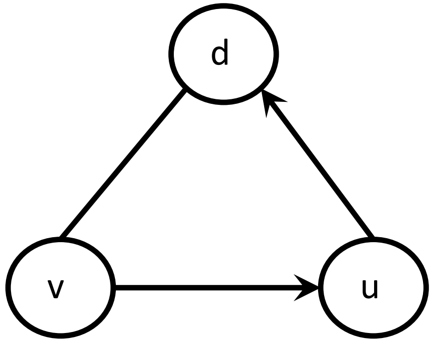
Ordering Solution: Go backwards through the new routing tree



Round **0** *(old)*



Round **1**



Round **2** *(new)*

General Consistent Update Scheme

- So far: every round:
  - Controller computes and sends out updates
  - Switches implement them and send acks
  - Controller receives acks

General Consistent Update Scheme

- So far: every round:
  - Controller computes and sends out updates
  - Switches implement them and send acks
  - Controller receives acks

Downsides:

- Controller keeps being involved
  - Load on centralized instance
- Need to wait until round is finished
- Latency to controller, many messages

General Consistent Update Scheme

- So far: every round:
  - Controller computes and sends out updates
  - Switches implement them and send acks
  - Controller receives acks

- How to decentralize such updates?

Downsides:
- **Controller keeps being involved**
  - **Load on centralized instance**
- **Need to wait until round is finished**
- **Latency to controller, many messages**

universität wien

General Consistent Update Scheme

- So far: every round:
  - Controller computes and sends out updates
  - Switches implement them and send acks
  - Controller receives acks

- How to decentralize such updates?
  - Idea: Controller sends out updates initially

Downsides:
- **Controller keeps being involved**
  - **Load on centralized instance**
- **Need to wait until round is finished**
- **Latency to controller, many messages**

General Consistent Update Scheme

- So far: every round:
  - Controller computes and sends out updates
  - Switches implement them and send acks
  - Controller receives acks

- How to decentralize such updates?
  - Idea: Controller sends out updates initially
  - Then: Switches tell neighbors when to update

Downsides:
- **Controller keeps being involved**
  - **Load on centralized instance**
- **Need to wait until round is finished**
- **Latency to controller, many messages**

General Consistent Update Scheme

- So far: every round:
  - Controller computes and sends out updates
  - Switches implement them and send acks
  - Controller receives acks

Downsides:
- **Controller keeps being involved**
  - **Load on centralized instance**
- **Need to wait until round is finished**
- **Latency to controller, many messages**

- How to decentralize such updates?
  - Idea: Controller sends out updates initially
  - Then: Switches tell neighbors when to update
  - Correctness can be verified locally

General Consistent Update Scheme

- So far: every round:
  - Controller computes and sends out updates
  - Switches implement them and send acks
  - Controller receives acks

- How to decentralize such updates?
  - Idea: Controller sends out updates initially
  - Then: Switches tell neighbors when to update
  - Correctness can be verified locally

Downsides:
- **Controller keeps being involved**
  - **Load on centralized instance**
- **Need to wait until round is finished**
- **Latency to controller, many messages**

Nguyen et al. (SOSR'17): Implemented in P4/OpenFlow

General Consistent Update Scheme

- So far: every round:
  - Controller computes and sends out updates
  - Switches implement them and send acks
  - Controller receives acks

Downsides:
- **Controller keeps being involved**
  - **Load on centralized instance**
- **Need to wait until round is finished**
- **Latency to controller, many messages**

- How to decentralize such updates?
  - Idea: Controller sends out updates initially
  - Then: Switches tell neighbors when to update
  - Correctness can be verified locally

Nguyen et al. (SOSR’17): Implemented in P4/OpenFlow

Foerster et al. (TCS’16): Via proof labeling schemes

General Consistent Update Scheme

- So far: every round:
  - Controller computes and sends out updates
  - Switches implement them and send acks
  - Controller receives acks

Downsides:
- **Controller keeps being involved**
  - **Load on centralized instance**
- **Need to wait until round is finished**
- **Latency to controller, many messages**

- How to decentralize such updates?
  - Idea: Controller sends out updates initially
  - Then: Switches tell neighbors when to update
  - Correctness can be verified locally

Nguyen et al. (SOSR'17): Implemented in P4/OpenFlow

Foerster et al. (TCS'16): Via proof labeling schemes

This paper: #1) General application to loop freedom and 2) routing path deployment via 2-phase commit

universität wien

How to Verify Correctness?

- Problem: Loops are a “global“ property
  - Might need to investigate complete downstream route to see if loop will appear
    - Slow and might require a locking mechanism ☹

universität wien

How to Verify Correctness?

- Problem: Loops are a “global“ property
  - Might need to investigate complete downstream route to see if loop will appear
    - Slow and might require a locking mechanism ☹

- However: Verifying is easier than Proving (*Concept of Proof Labeling Schemes*)

universität wien

How to Verify Correctness?

- Problem: Loops are a “global“ property
  - Might need to investigate complete downstream route to see if loop will appear
    - Slow and might require a locking mechanism ☹

Initially introduced by Korman et al. (2005)

- However: Verifying is easier than Proving (*Concept of Proof Labeling Schemes*)

How to Verify Correctness?

- Problem: Loops are a “global“ property
  - Might need to investigate complete downstream route to see if loop will appear
    - Slow and might require a locking mechanism ☹

Initially introduced by Korman et al. (2005)

- However: Verifying is easier than Proving (*Concept of Proof Labeling Schemes*)
  - “Proof” of correctness is distributed to nodes by the controller

How to Verify Correctness?

- Problem: Loops are a “global“ property
  - Might need to investigate complete downstream route to see if loop will appear
    - Slow and might require a locking mechanism ☹

Initially introduced by Korman et al. (2005)

- However: Verifying is easier than Proving (*Concept of Proof Labeling Schemes*)
  - “Proof” of correctness is distributed to nodes by the controller
    - Nodes can verify by checking proofs of their neighbors

universität wien

How to Verify Correctness?

- Problem: Loops are a “global“ property
  - Might need to investigate complete downstream route to see if loop will appear
    - Slow and might require a locking mechanism ☹

Initially introduced by Korman et al. (2005)

- However: Verifying is easier than Proving (*Concept of Proof Labeling Schemes*)
  - “Proof” of correctness is distributed to nodes by the controller
    - Nodes can verify by checking proofs of their neighbors
      - Idea: Something is incorrect, don’t update/raise alarm

universität wien

How to Verify Correctness?

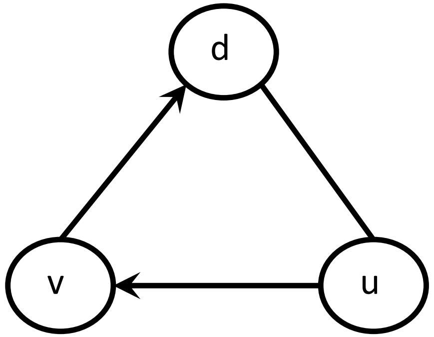
- Problem: Loops are a “global“ property
  - Might need to investigate complete downstream route to see if loop will appear
    - Slow and might require a locking mechanism ☹

Initially introduced by Korman et al. (2005)

- However: Verifying is easier than Proving (*Concept of Proof Labeling Schemes*)
  - “Proof” of correctness is distributed to nodes by the controller
    - Nodes can verify by checking proofs of their neighbors
      - Idea: Something is incorrect, don’t update/raise alarm

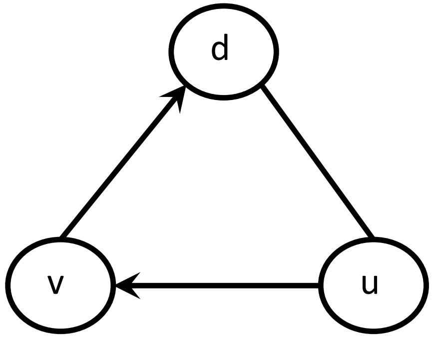
- Intuition on next slide

Proof Labeling – Without Network Updates



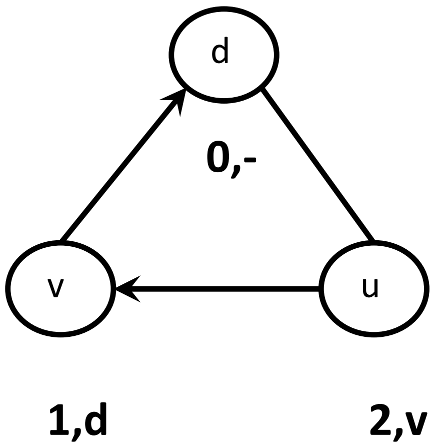
Proof Labeling – Without Network Updates

- Prover (Controller) gives:
  - Distance to root d
  - Parent in tree



Proof Labeling – Without Network Updates

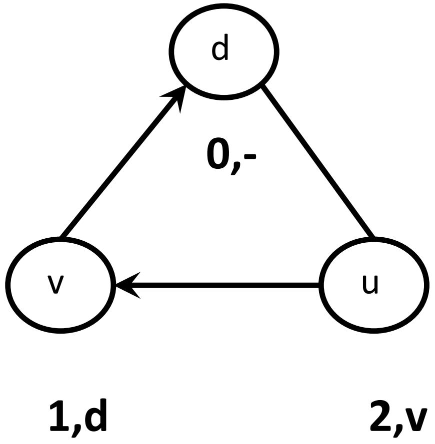
- Prover (Controller) gives:
  - Distance to root d
  - Parent in tree



universität wien

Proof Labeling – Without Network Updates

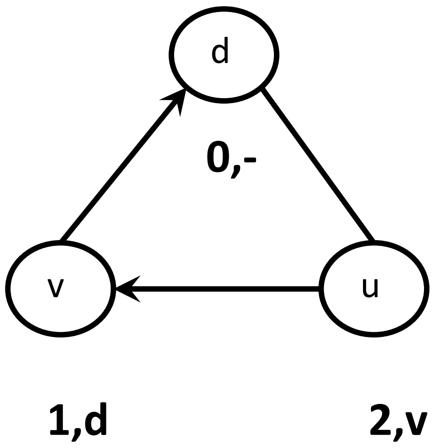
- Prover (Controller) gives:
  - Distance to root d
  - Parent in tree
- Verifier (at node) checks:
  - Has my parent* a smaller distance



**We assume node Ids cannot be faked and d doesn't need a parent*

Proof Labeling – Without Network Updates

- Prover (Controller) gives:
  - Distance to root d
  - Parent in tree
- Verifier (at node) checks:
  - Has my parent* a smaller distance

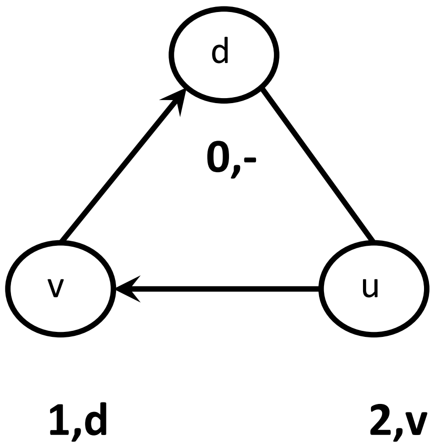


If prover sends correct labels:
- **All nodes will output YES**

**We assume node Ids cannot be faked and d doesn‘t need a parent*

Proof Labeling – Without Network Updates

- Prover (Controller) gives:
  - Distance to root d
  - Parent in tree
- Verifier (at node) checks:
  - Has my parent* a smaller distance



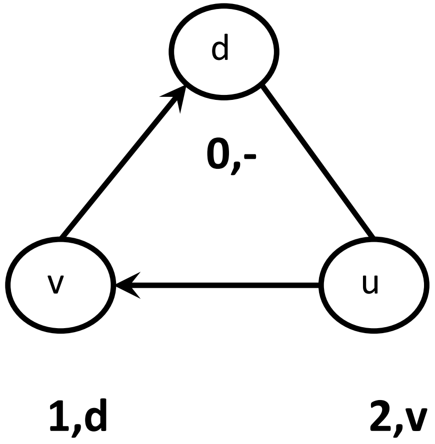
If prover sends correct labels:
- **All nodes will output YES**

If no tree rooted at d:
- **At least one node outputs NO**

**We assume node Ids cannot be faked and d doesn‘t need a parent*

Proof Labeling – Without Network Updates

- Prover (Controller) gives:
  - Distance to root d
  - Parent in tree
- Verifier (at node) checks:
  - Has my parent* a smaller distance



If prover sends correct labels:
- **All nodes will output YES**

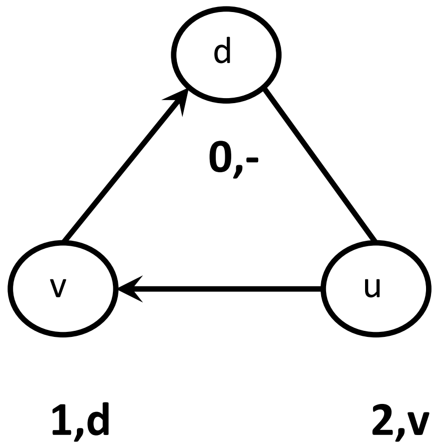
If no tree rooted at d:
- **At least one node outputs NO**

- Note:
  - Requires O(log |V|) bits (optimal,Korman et al. 2005)
  - Already explored in SDN context by Schmid/Suomela, 2013

**We assume node Ids cannot be faked and d doesn‘t need a parent*

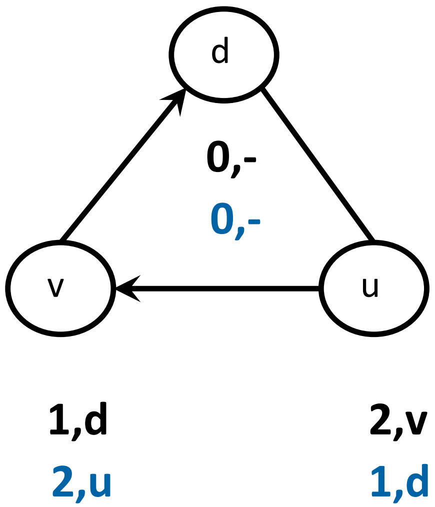
Proof Labeling – *With* Network Updates

- Prover sends out **new labels**



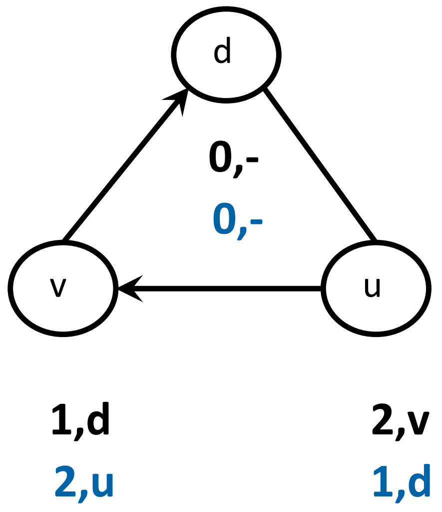
Proof Labeling – *With* Network Updates

- Prover sends out **new labels**



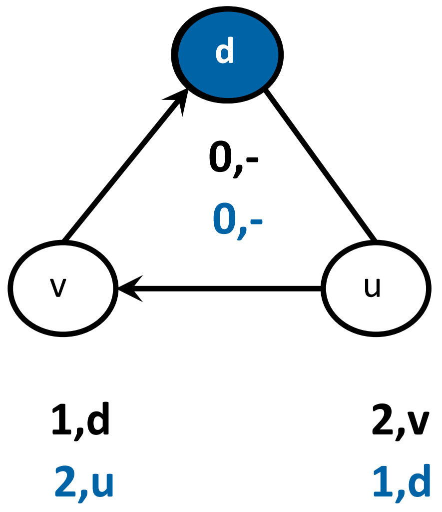
Proof Labeling – *With* Network Updates

- Prover sends out **new labels**
- Nodes check if they can switch:
  - Did my parent update?



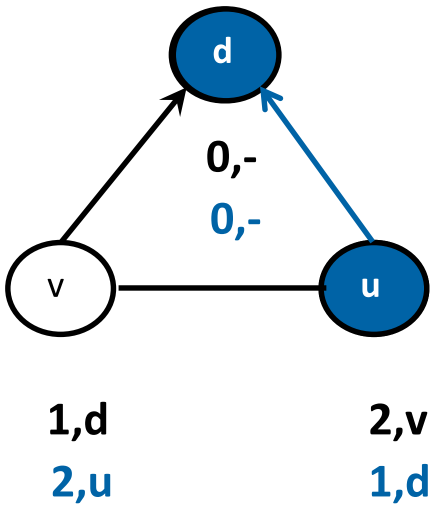
Proof Labeling – *With* Network Updates

- Prover sends out **new labels**
- Nodes check if they can switch:
  - Did my parent update?



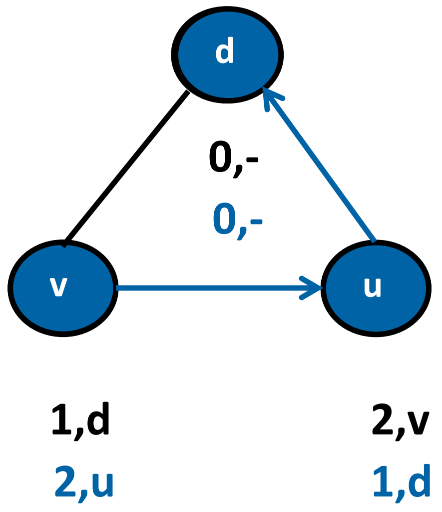
Proof Labeling – *With* Network Updates

- Prover sends out **new labels**
- Nodes check if they can switch:
  - Did my parent update?



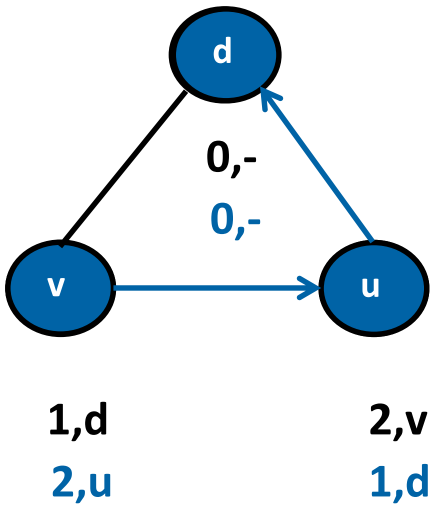
Proof Labeling – *With* Network Updates

- Prover sends out **new labels**
- Nodes check if they can switch:
  - Did my parent update?



Proof Labeling – *With* Network Updates

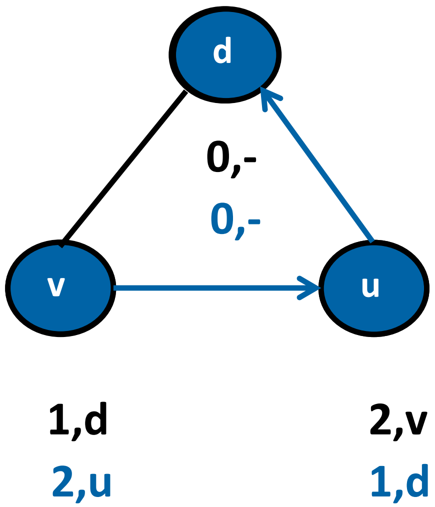
- Prover sends out **new labels**
- Nodes check if they can switch:
  - Did my parent update?



- Advantages:

Proof Labeling – *With* Network Updates

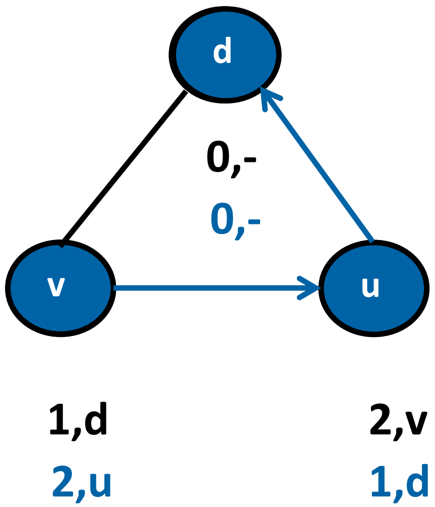
- Prover sends out **new labels**
- Nodes check if they can switch:
  - Did my parent update?



- Advantages:
  - Controller only sends labels once

Proof Labeling – *With* Network Updates

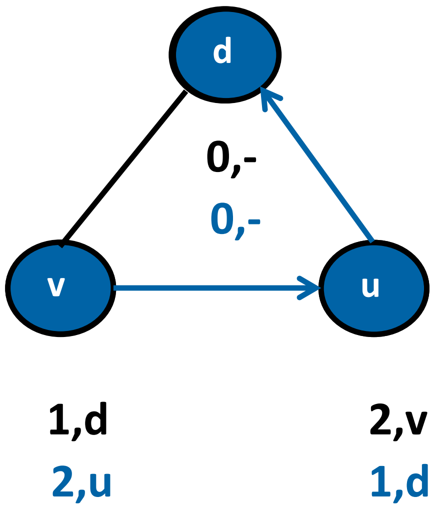
- Prover sends out **new labels**
- Nodes check if they can switch:
  - Did my parent update?



- Advantages:
  - Controller only sends labels once
  - Captures asynchrony, nodes refuse incorrect updates

Proof Labeling – *With* Network Updates

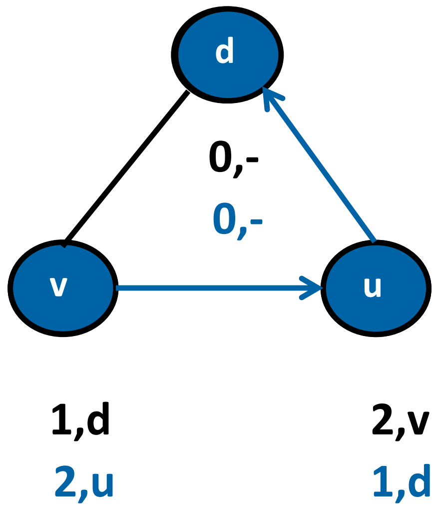
- Prover sends out **new labels**
- Nodes check if they can switch:
  - Did my parent update?



- Advantages:
  - Controller only sends labels once
  - Captures asynchrony, nodes refuse incorrect updates
  - **New labels** can be sent before **old labels** are finished

Proof Labeling – *With* Network Updates

- Prover sends out **new labels**
- Nodes check if they can switch:
  - Did my parent update?



- Advantages:
  - Controller only sends labels once
  - Captures asynchrony, nodes refuse incorrect updates
  - **New labels** can be sent before **old labels** are finished
    - Look at tree #, only update to higher tree #

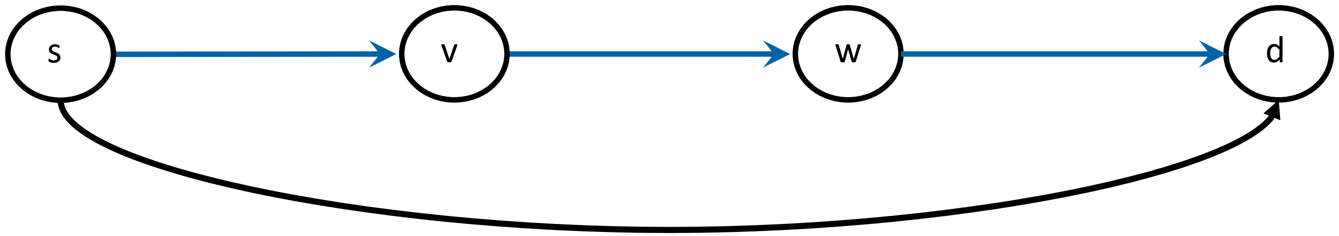
Can Standard Proof Labeling Methods Always be Directly Applied?

Can Standard Proof Labeling Methods Always be Directly Applied?

- Case study: Deployment of **new** s-d **flow** routing path

Can Standard Proof Labeling Methods Always be Directly Applied?

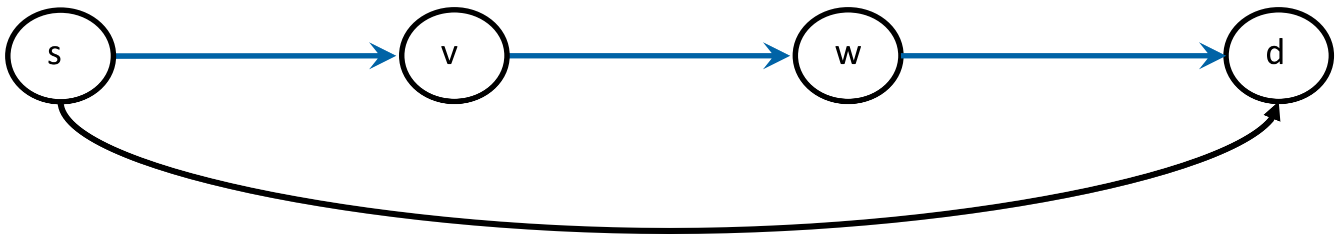
- Case study: Deployment of **new** s-d **flow** routing path



Can Standard Proof Labeling Methods Always be Directly Applied?

- Case study: Deployment of **new** s-d **flow** routing path

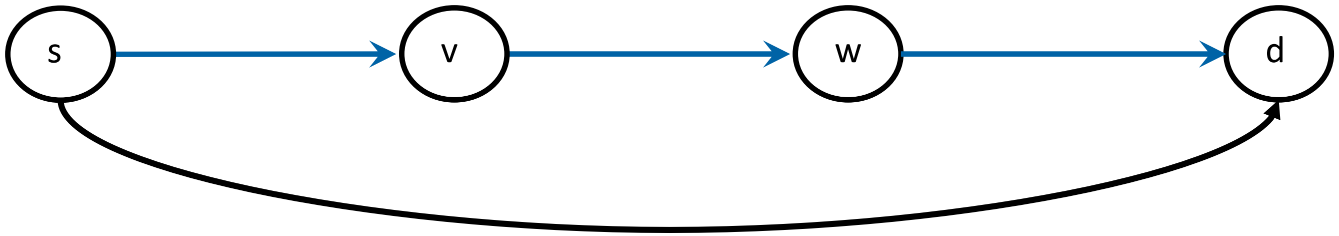
- Standard proof labeling method:
  - Point to successor/predecessor (“Hand holding”)
    - O(log max degree) bits with 2-hop coloring



Can Standard Proof Labeling Methods Always be Directly Applied?

- Case study: Deployment of **new** s-d **flow** routing path

- Standard proof labeling method:
  - Point to successor/predecessor (“Hand holding”)
    - O(log max degree) bits with 2-hop coloring

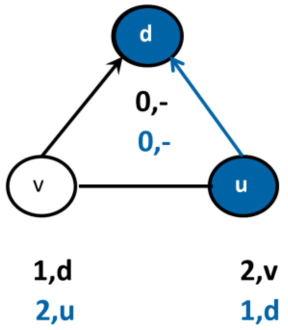


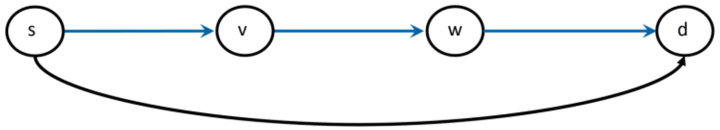
- Problem: v and w can never update!
  - v needs w to update before and vice versa ☹
  - Can be fixed with distance-labeling again ☺

Summary

- We investigated verifiable distributed consistent network updates

- With applications to:
  - Loop-free routing trees (destination based)
  - Path deployment (flow based)





- Next challenge: Deploy proof labeling concepts in P4/OpenFlow hardware and/or Mininet

universität wien

References

- Thanh Dang Nguyen, Marco Chiesa, Marco Canini. Decentralized Fast Consistent Updates . In Symposium on SDN Research (SOSR 2017), ACM, pages 21-33, 2017.
- Local Checkability, No Strings Attached: (A)cyclicity, Reachability, Loop Free Updates in SDNs. Klaus-Tycho Foerster, Thomas Luedi, Jochen Seidel, and Roger Wattenhofer. Theoretical Computer Science (TCS), Volume 709, pp. 48-63, January 2018. Accepted November 2016
- Exploiting Locality in Distributed SDN Control. Stefan Schmid and Jukka Suomela. ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN), Hong Kong, China, August 2013.
- Amos Korman, Shay Kutten, David Peleg: Proof labeling schemes. PODC 2005: 9-18

Distributed Consistent Network Updates in SDNs: Local Verification for Global Guarantees

Klaus-T. Foerster, Stefan Schmid

IEEE NCA 2019