TOWARDS LOSSLESS DATA CENTER RECONFIGURATION: CONSISTENT NETWORK UPDATES IN SDNS

KLAUS-TYCHO FOERSTER

AALBORG UNIVERSITY
DENMARK

Joint work with…

1. *Consistent Updates in Software Defined Networks: On Dependencies, Loop Freedom, and Blackholes (IFIP Networking 2016)*
   Klaus-Tycho Foerster, **Ratul Mahajan**, **Roger Wattenhofer**

2. *On Consistent Migration of Flows in SDNs (INFOCOM 2016)*
   Sebastian Brandt, Klaus-Tycho Foerster, **Roger Wattenhofer**

3. *The Power of Two in Consistent Network Updates: Hard Loop Freedom, Easy Flow Migration (ICCCN 2016)*
   Klaus-Tycho Foerster, **Roger Wattenhofer**

4. *Augmenting Flows for the Consistent Migration of Multi-Commodity Single-Destination Flows in SDNs (Pervasive Mob. Comput. 2017)*
   Sebastian Brandt, Klaus-Tycho Foerster, **Roger Wattenhofer**

5. *Local Checkability, No Strings Attached: (A)cyclicity, Reachability, Loop Free Updates in SDNs (Theoret. Comput. Sci 2016)*
   Klaus-Tycho Foerster, **Thomas Luedi**, **Jochen Seidel**, **Roger Wattenhofer**

6. *Understanding and Mitigating Packet Corruption in Data Center Networks (SIGCOMM 2017)*
   Danyang Zhuo, **Monia Ghobadi**, **Ratul Mahajan**, Klaus-Tycho Foerster, **Arvind Krishnamurthy**, **Thomas Anderson**

7. *Survey of Consistent Network Updates (under submission, arXiv 1609.02305)*
   Klaus-Tycho Foerster, **Stefan Schmid**, **Stefano Vissicchio**

8. *Loop-Free Route Updates for Software-Defined Networks (under submission, extended version of their PODC 2015)*
   Klaus-Tycho Foerster, **Arne Ludwig**, **Jan Marcinkowski**, **Stefan Schmid**

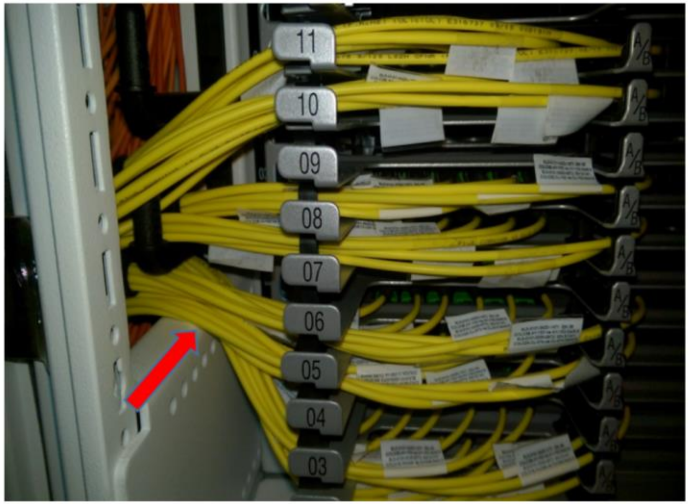
9. *Not so Lossless Flow Migration (under submission, partially contained in Dissertation)*
   Sebastian Brandt, Klaus-Tycho Foerster, **Laurent Vanbever**, **Roger Wattenhofer**

AALBORG UNIVERSITY
DENMARK

First Motivation: Link Repair

Root Cause	Relative Ratio
Connector contamination	17-57%
Bent or damaged cable	14-48%
Decaying transmitter	< 1%
Loose or bad transceiver	6-45%
Shared component failure	10-26%

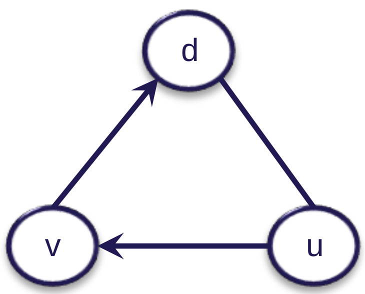
Relative contributions of corruption in 15 DCNs
(350K switch-to-switch optical links, over 7 months)



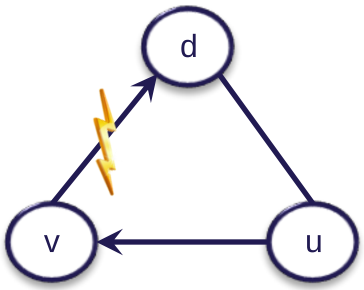
Zhou et al.: Understanding and Mitigating Packet Corruption in Data Center Networks (SIGCOMM 2017).

AALBORG UNIVERSITY
DENMARK

Toy Example

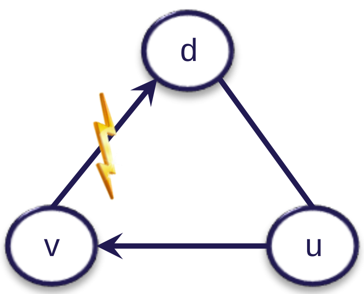


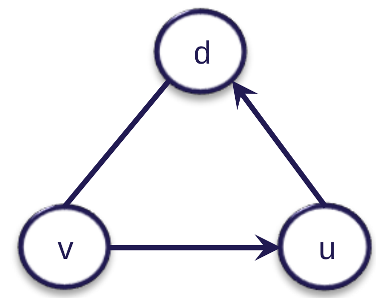
Toy Example



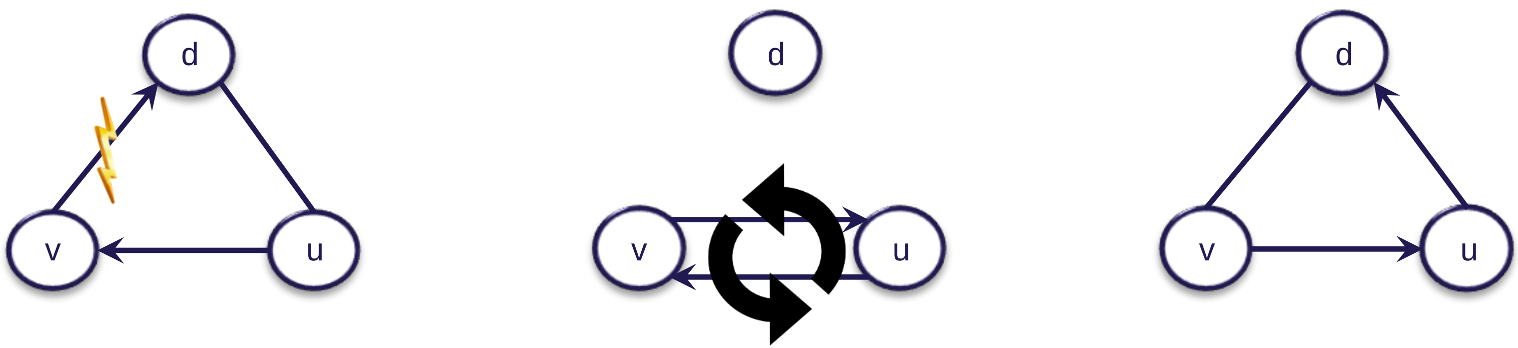
AALBORG UNIVERSITY
DENMARK

Toy Example

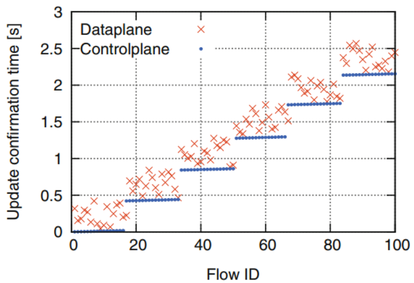




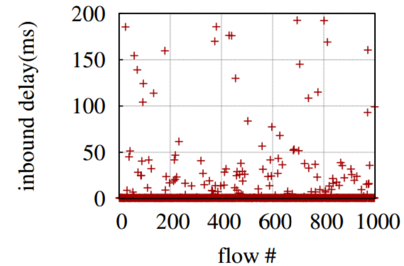
Toy Example



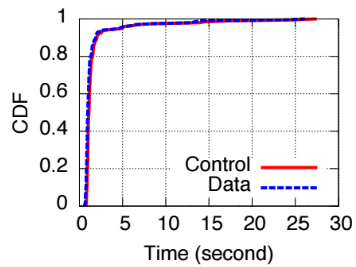
Appears in Practice



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



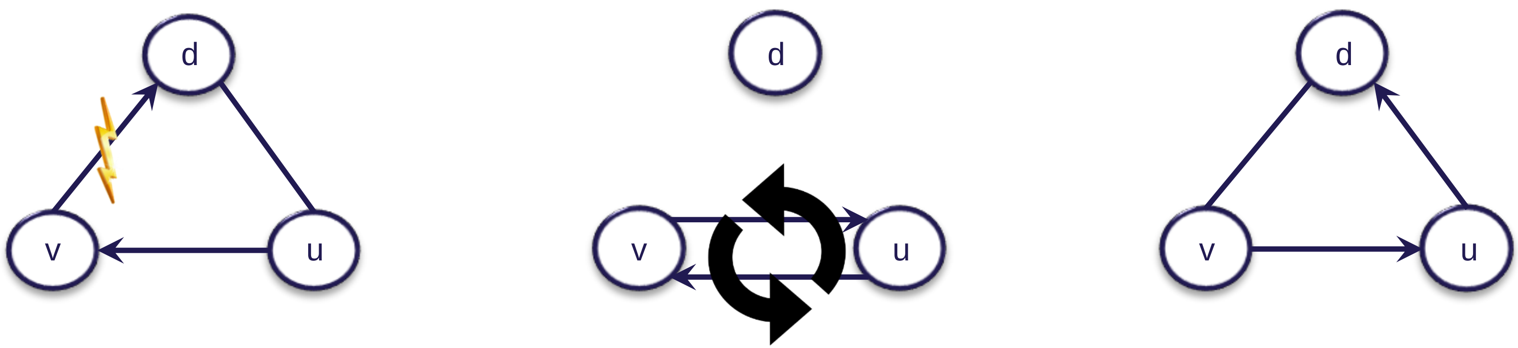
"…*the inbound latency is* ***quite variable*** *with a* […] *standard deviation of 31.34ms…*"
He et al., SOSR 2015



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

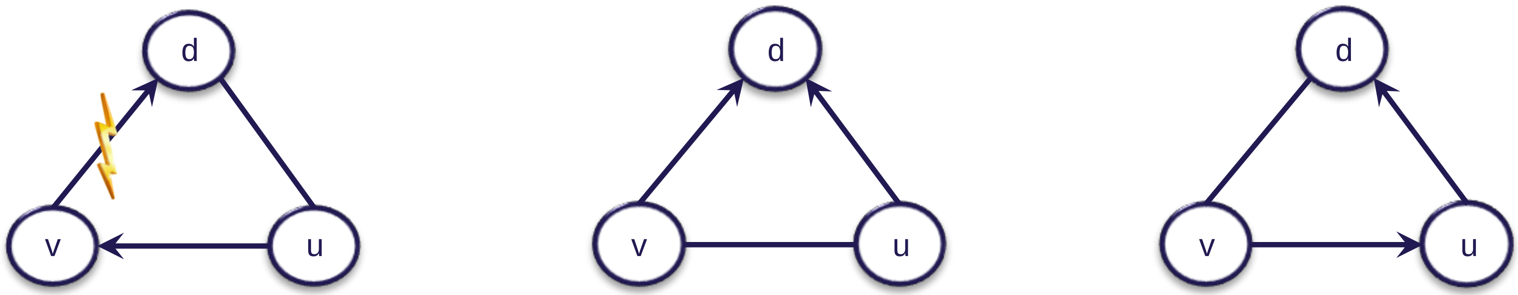
AALBORG UNIVERSITY
DENMARK

Toy Example



AALBORG UNIVERSITY
DENMARK

Toy Example



AALBORG UNIVERSITY
DENMARK

Software-Defined Networking

Centralized controller updates networks rules for optimization

- Controller (*control plane*) updates the switches/routers (*data plane*)

old network
rules

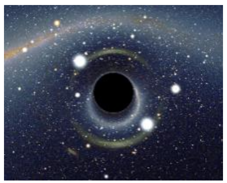
network updates

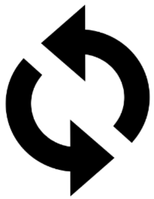
new network
rules

old network
rules

network updates

new network
rules





AALBORG UNIVERSITY
DENMARK

old network
rules

network updates

new network
rules

possible solution: be fast!

e.g., B4 [Jain et al., 2013]

old network
rules

network updates

new network
rules

possible solution: synchronize time well!

e.g., TimedSDN [Mizrahi et al., 2014-17]
Chronus [Zheng et al., 2017]

old network rules

network updates

new network rules

possible solution: be consistent!

e.g.,

- per-router ordering [Vanbever et al., 2012]
- two phase commit [Reitblatt et al., 2012]
- SWAN [Hong et al., 2013]
- Dionysus [Jin et al., 2014]
- ….

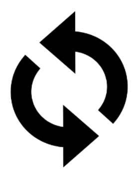
AALBORG UNIVERSITY
DENMARK

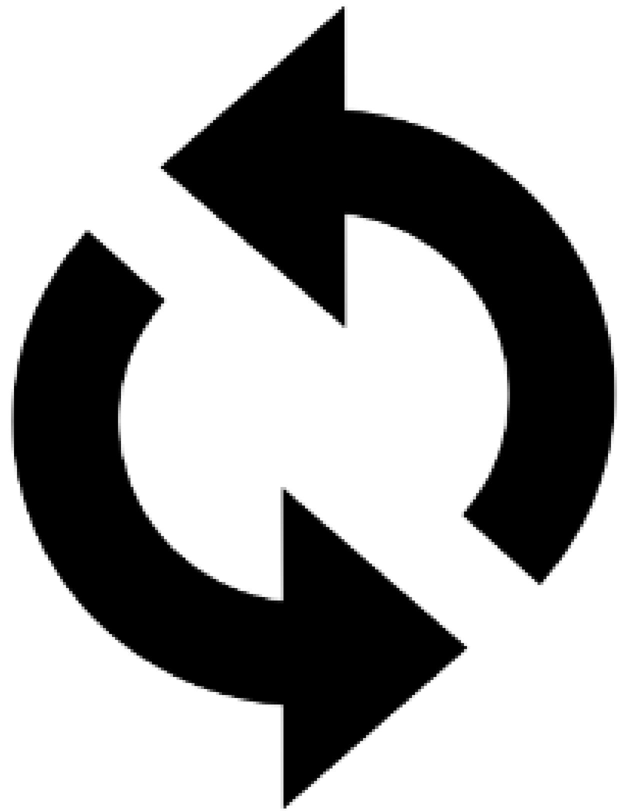
old network
rules

network updates

new network
rules

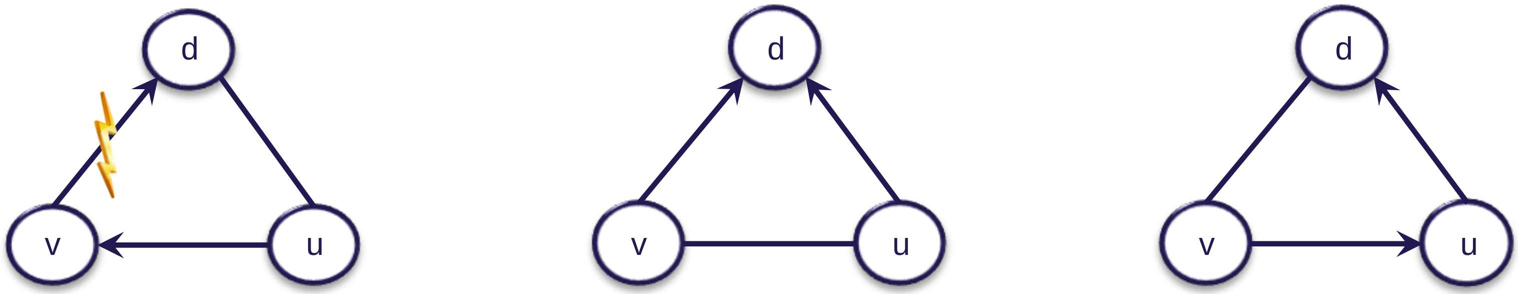
possible solution: be consistent!





AALBORG UNIVERSITY
DENMARK

Ordering Solution: Go backwards through the new Tree



- Always works for single-destination rules
  - Also for multi-destination with sufficient memory („split“)

- Schedule length: tree depth (up to Ω(n))
  - Optimal algorithms?

Optimal Schedule?

- 3-round schedule? NP-complete! [Ludwig et al., 2015]
    - (Sublinear schedule for 2 destinations w/o split: NP-complete)

Optimal Schedule?

- 3-round schedule? NP-complete! [Ludwig et al., 2015]
  - (Sublinear schedule for 2 destinations w/o split: NP-complete)

- However: greedy updates always finish (eventually).

Optimal Schedule?

- 3-round schedule? NP-complete! [Ludwig et al., 2015]
  - (Sublinear schedule for 2 destinations w/o split: NP-complete)

- However: greedy updates always finish (eventually).
  - Maximizing greedy update: NP-complete!
  - But: Can be approximated well.
    - Feedback Arc Set / Max. Acyclic Subgraph

[ICCCN ‘16] & [Amiri et al., ‘16]

Optimal Schedule?

- 3-round schedule? NP-complete! [Ludwig et al., 2015]
  - (Sublinear schedule for 2 destinations w/o split: NP-complete)

- However: greedy updates always finish (eventually).
  - Maximizing greedy update: NP-complete!
  - But: Can be approximated well. [ICCCN ‘16] & [Amiri et al., ‘16]
    - Feedback Arc Set / Max. Acyclic Subgraph

- Bad news: Greedy can turn O(1) instances into Ω(n) schedules ☹ [Ludwig et al., 2015]
  - What to do?

Relax! [Ludwig et al., 2015]

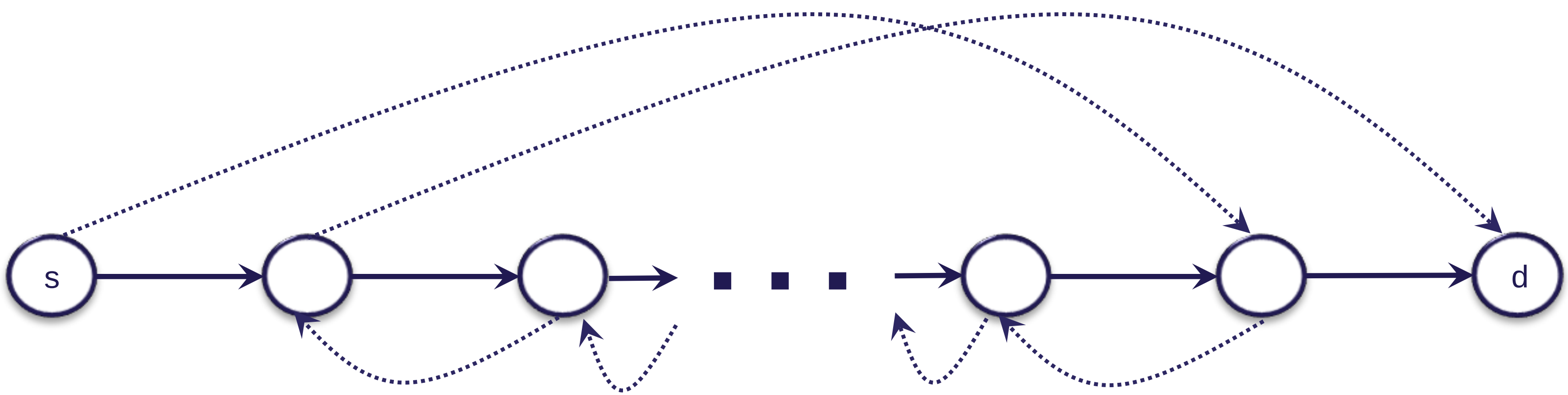
Two key ideas:

1. ~~destination-d-based~~ source-destination pairs (s,d)
2. ~~no loops~~ no loops between (s,d)

Relax! [Ludwig et al., 2015]

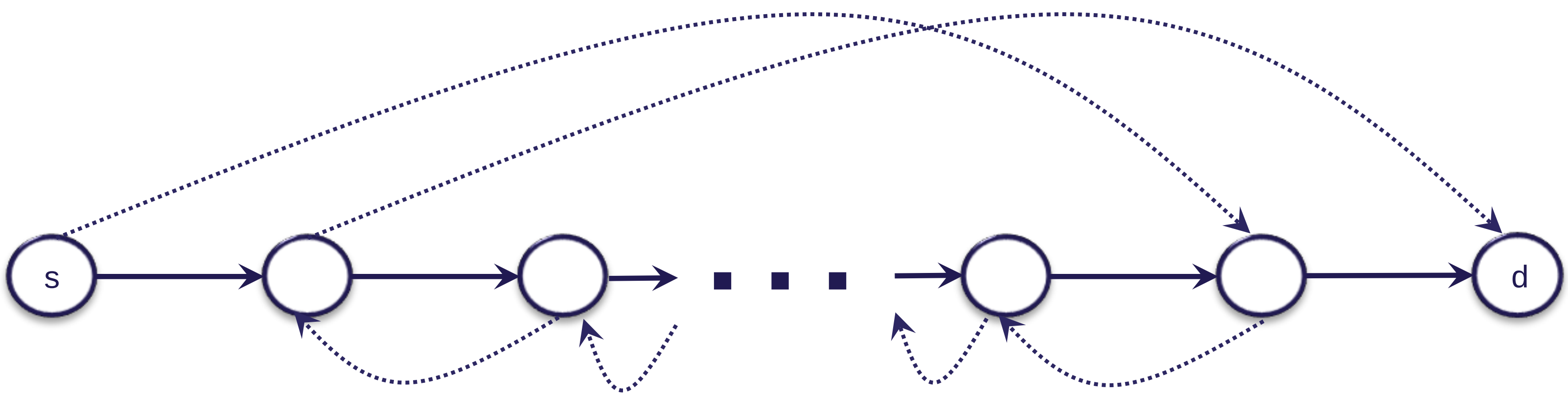
Two key ideas:

1. ~~destination d based~~ source-destination pairs (s,d)
2. ~~no loops~~ no loops between (s,d)



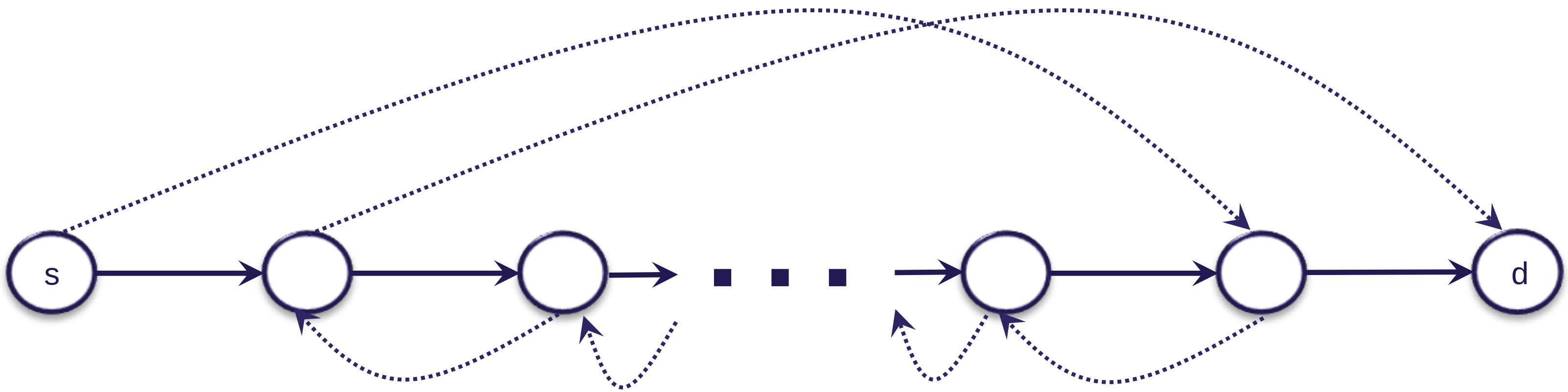
Relax! [Ludwig et al., 2015]

- Non-relaxed: Ω(n) rounds



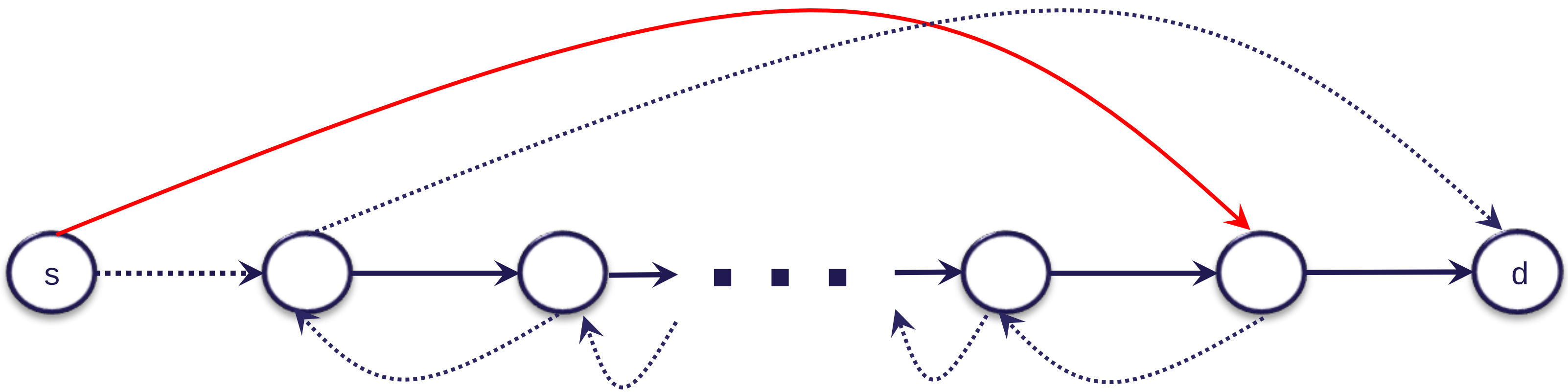
Relax! [Ludwig et al., 2015]

- Non-relaxed: Ω(n) rounds
- Relaxed?



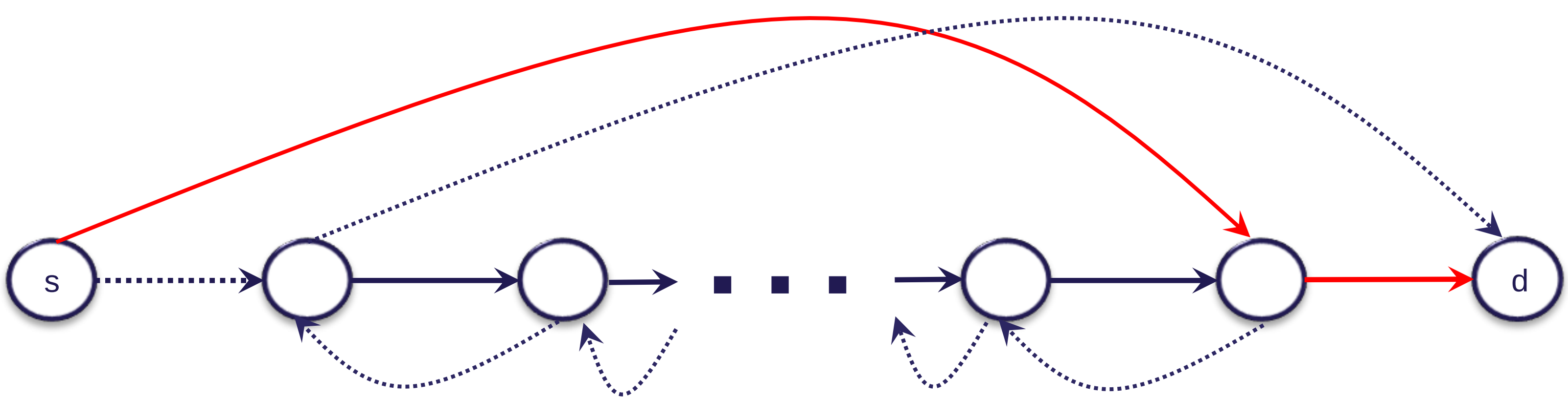
Relax! [Ludwig et al., 2015]

- Non-relaxed: Ω(n) rounds
- Relaxed?



Relax! [Ludwig et al., 2015]

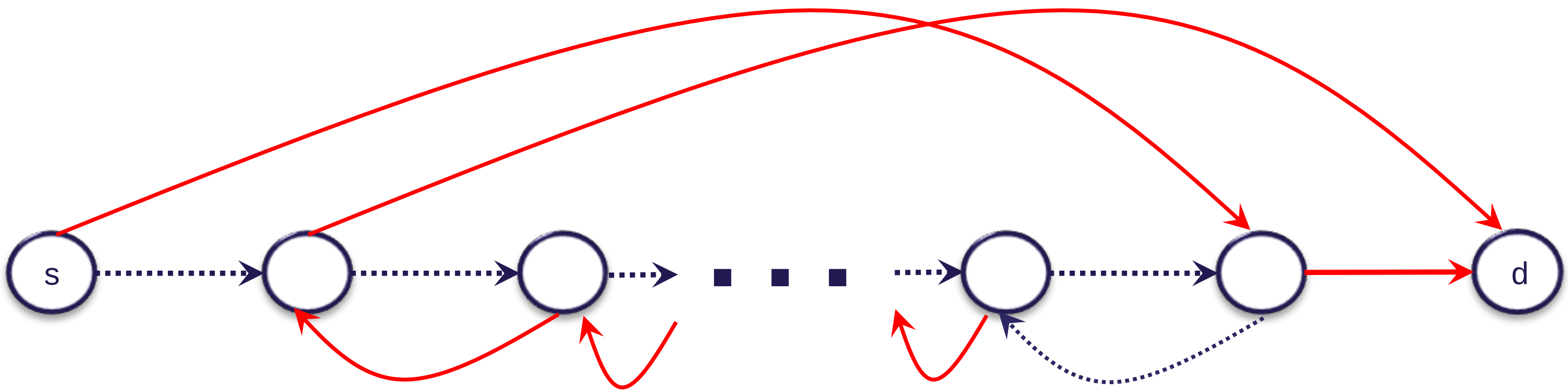
- Non-relaxed: Ω(n) rounds
- Relaxed?



AALBORG UNIVERSITY
DENMARK

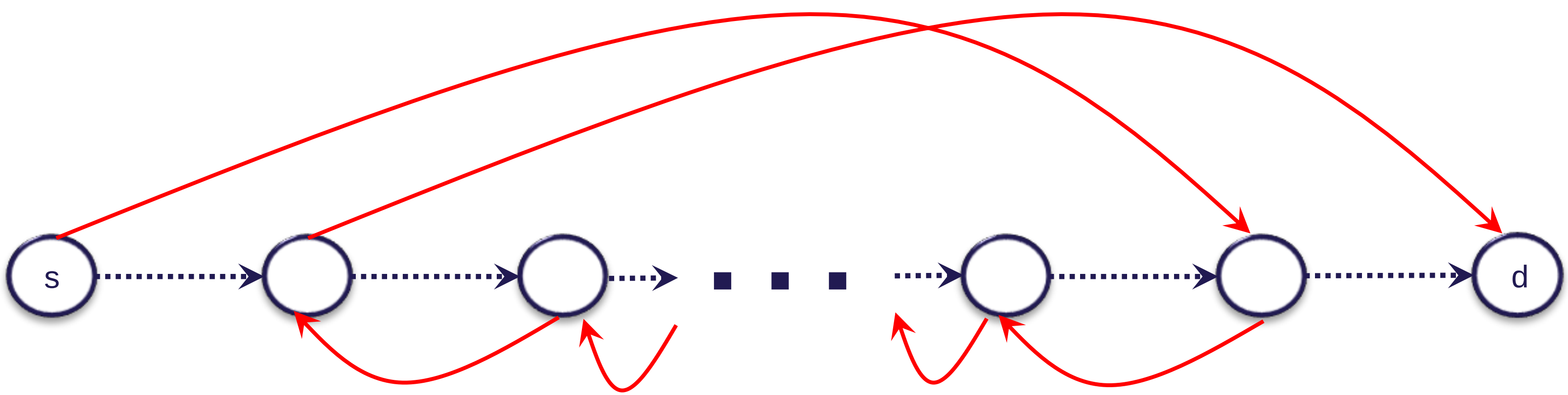
Relax! [Ludwig et al., 2015]

- Non-relaxed: Ω(n) rounds
- Relaxed?



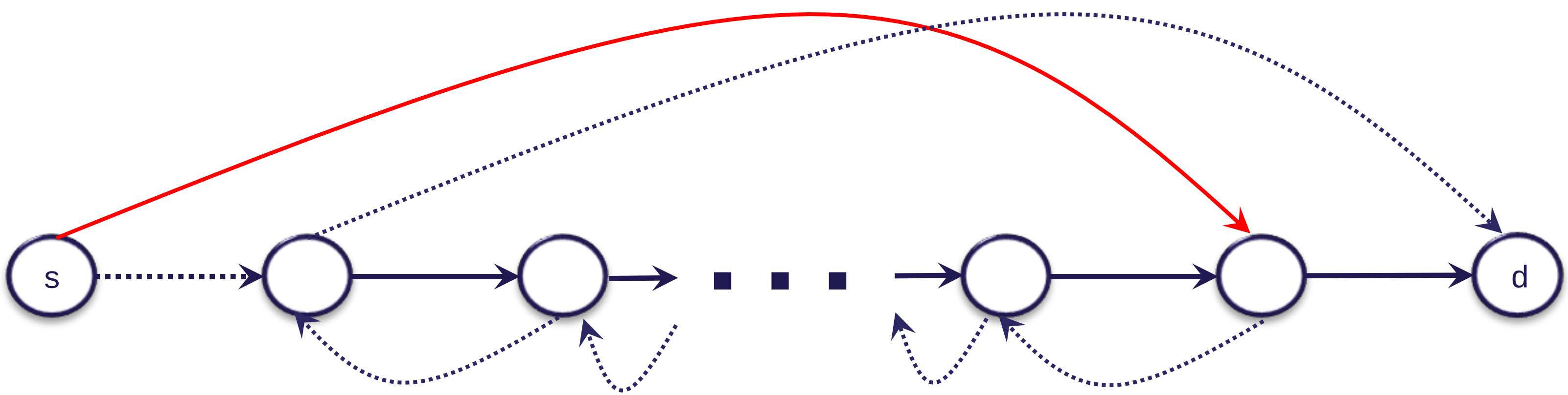
Relax! [Ludwig et al., 2015]

- Non-relaxed: $\Omega(n)$ rounds
- Relaxed? Just 3 rounds



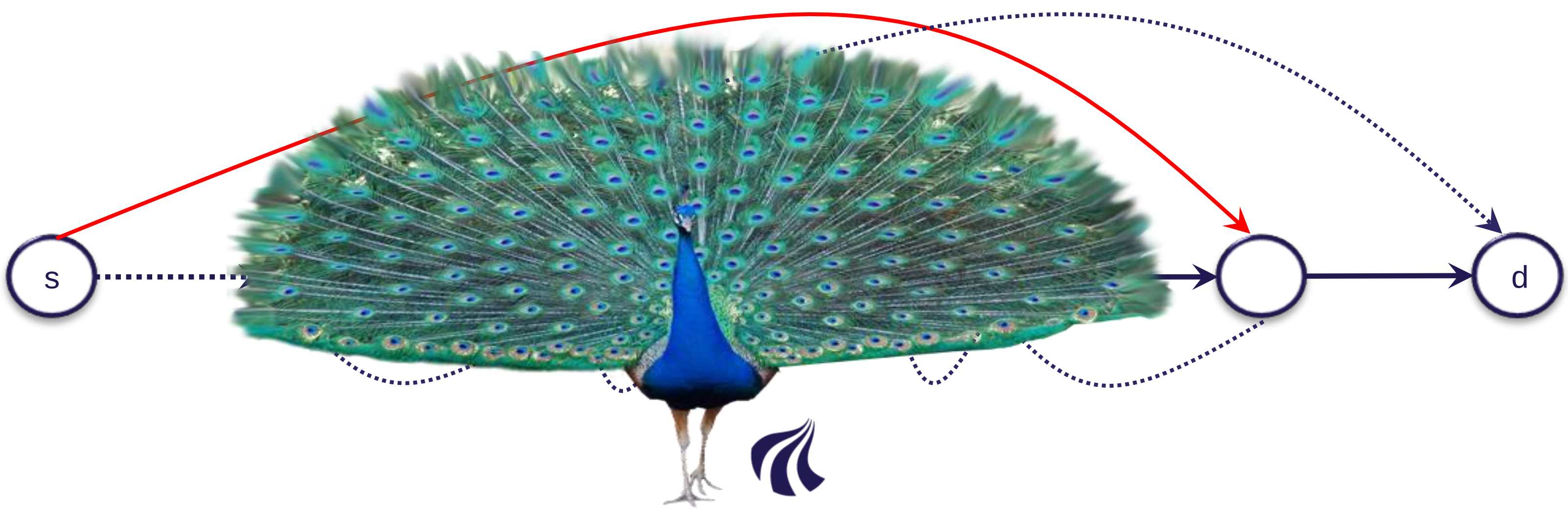
Relax! [Ludwig et al., 2015]

- Non-relaxed: Ω(n) rounds
- Relaxed? Just 3 rounds
  - In general: $O(\log n)$ rounds („*Peacock*")



Relax! [Ludwig et al., 2015]

- Non-relaxed: Ω(n) rounds
- Relaxed? Just 3 rounds
  - In general: $O(\log n)$ rounds („*Peacock*")



Relax!

- Non-relaxed: Ω(n) rounds
- Relaxed? Just 3 rounds
  - In general: $O(\log n)$ rounds („*Peacock*“) – Optimal?

Relax!

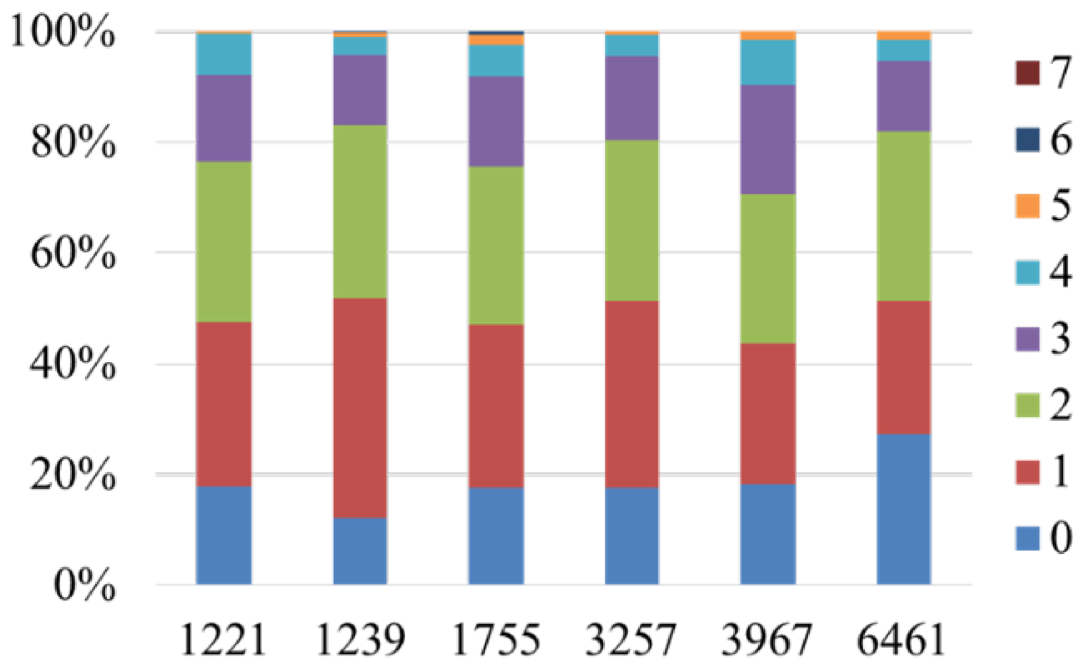
- Non-relaxed: $\Omega(n)$ rounds
- Relaxed? Just 3 rounds
  - In general: $O(\log n)$ rounds („*Peacock*“) – Optimal?
    - $\Omega(\log n)$ instances exist for Peacock

Relax!

- Non-relaxed: $\Omega(n)$ rounds
- Relaxed? Just 3 rounds
  - In general: $O(\log n)$ rounds („*Peacock*“) – Optimal?
    - $\Omega(\log n)$ instances exist for Peacock
  - Worst case for relaxed? – Unknown!
    - Worst known: 7 rounds ($n > 1000$) [Ludwig et al., 2015]

AALBORG UNIVERSITY
DENMARK

Greedy updates



Decentralized Updates for „Tree-Ordering“

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

Decentralized Updates for „Tree-Ordering“

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

- Alternative: Use dualism to so-called *proof labeling schemes*

Centralized Controller
(Prover)

SDN switch
(Verifier)

Decentralized Updates for „Tree-Ordering“

When should I update?

Decentralized Updates for „Tree-Ordering“

Once my parent updates!

AALBORG UNIVERSITY
DENMARK

Decentralized Updates for „Tree-Ordering“

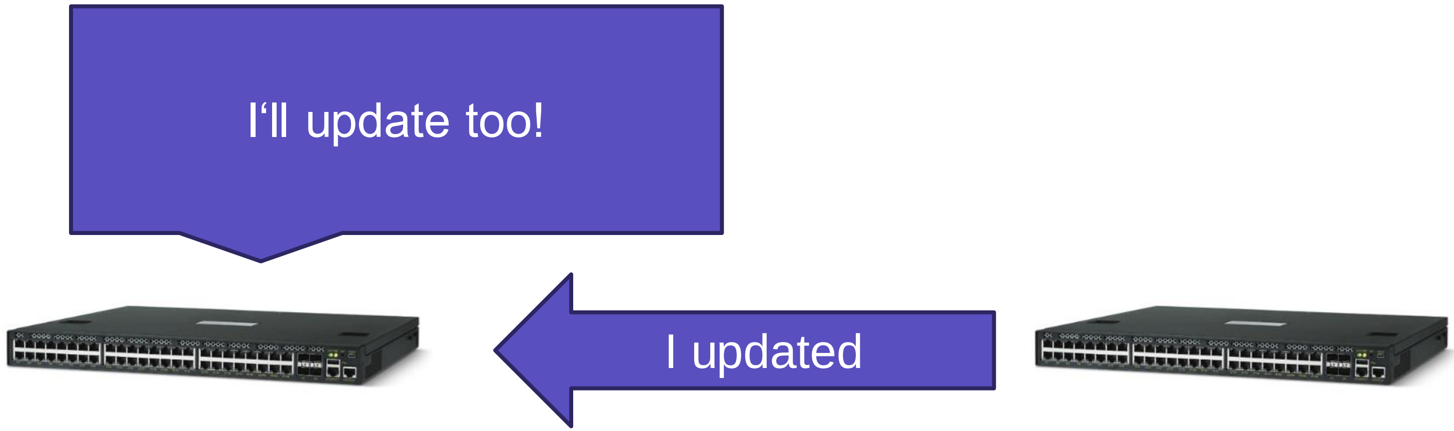
Once my parent updates!

Send parent ID

AALBORG UNIVERSITY
DENMARK

Decentralized Updates for „Tree-Ordering“

Decentralized Updates for „Tree-Ordering“



Decentralized Updates for „Tree-Ordering“

+ Only one controller-switch interaction per route change
+ New route changes can be pushed before old ones done
+ Incorrect updates can be locally detected

- Requires switch-to-switch communication e.g., [Nguyen et al., SOSR 2017]

Foerster et al.: Local Checkability, No Strings Attached: (A)cyclicity, Reachability, Loop Free Updates in SDNs (Theoret. Comput. Sci 2017)


CONGESTION
AHEAD
NEXT 20
YEARS

Saeed Akhoondian Amiri, Szymon Dudycz, Stefan Schmid, Sebastian Wiederrecht: Congestion-Free Rerouting of Flows on DAGs. CoRR abs/1611.09296 (2016)

AALBORG UNIVERSITY
DENMARK

Consistent Migration of Flows

Introduced in *SWAN* (Hong et al., SIGCOMM 2013)

Idea: Flows can be on the **old** or **new** route

For all edges: $\sum_{\forall F} \max(\mathbf{old}, \mathbf{new}) \leq capacity$

Unsplittable flows: Hard… (Algorithms out there: integer programs..)

What about Splittable flows?

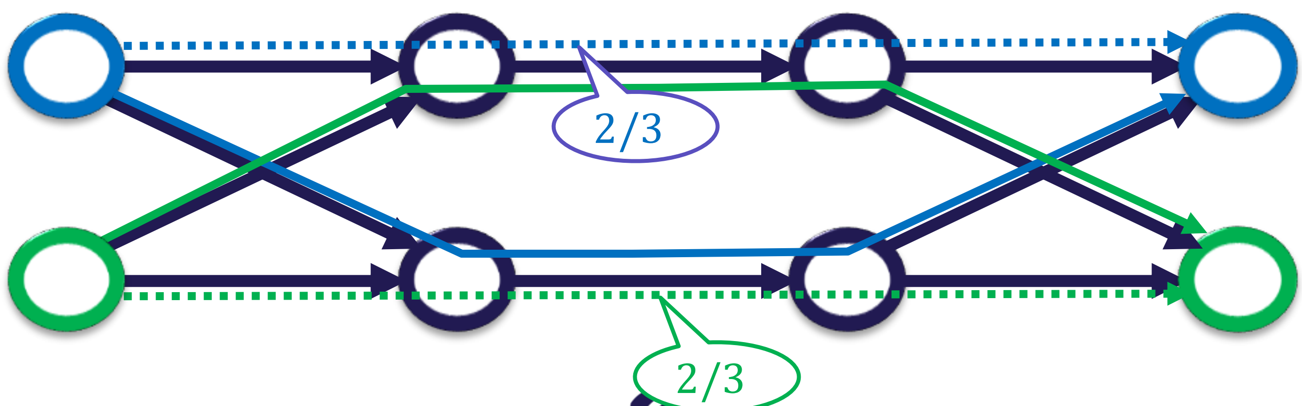
Consistent Migration of Flows

Introduced in *SWAN* (Hong et al., SIGCOMM 2013)

Idea: Flows can be on the **old** or **new** route

For all edges: $\sum_{\forall F} \max(\mathbf{old}, \mathbf{new}) \leq capacity$

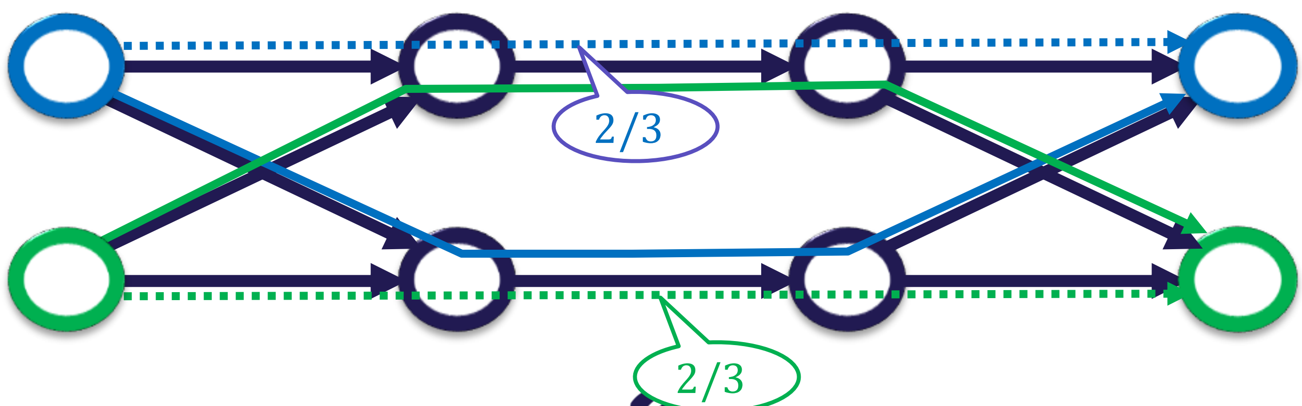
No ordering exists $(2/3 + 2/3 > 1)$



Consistent Migration of Flows

Approach of *SWAN*: use slack x (i.e., **%**)

- Here $x = 1/3$
- Move slack $x \Rrightarrow \lceil 1/x \rceil - 1$ staged partial moves

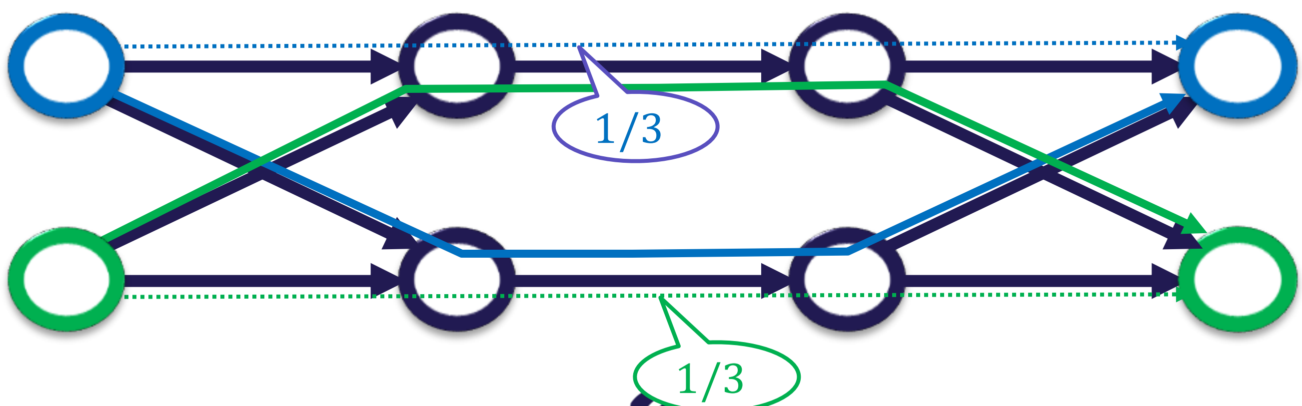


Consistent Migration of Flows

Approach of *SWAN*: use slack x (i.e., **%**)

- Here $x = 1/3$
- Move slack $x \Rrightarrow \lceil 1/x \rceil - 1$ staged partial moves

Update 1 of 2

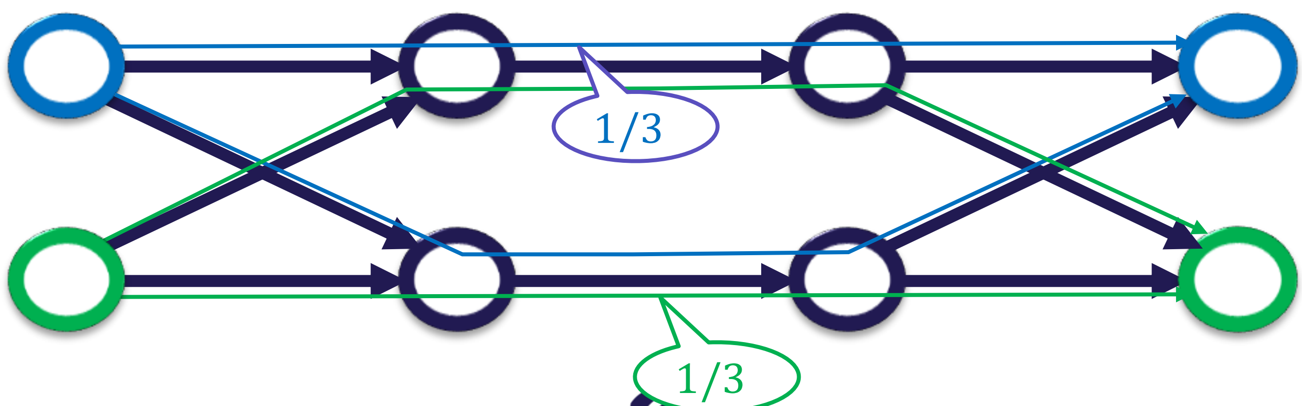


Consistent Migration of Flows

Approach of *SWAN*: use slack x (i.e., **%**)

- Here $x = 1/3$
- Move slack $x \Rrightarrow \lceil 1/x \rceil - 1$ staged partial moves

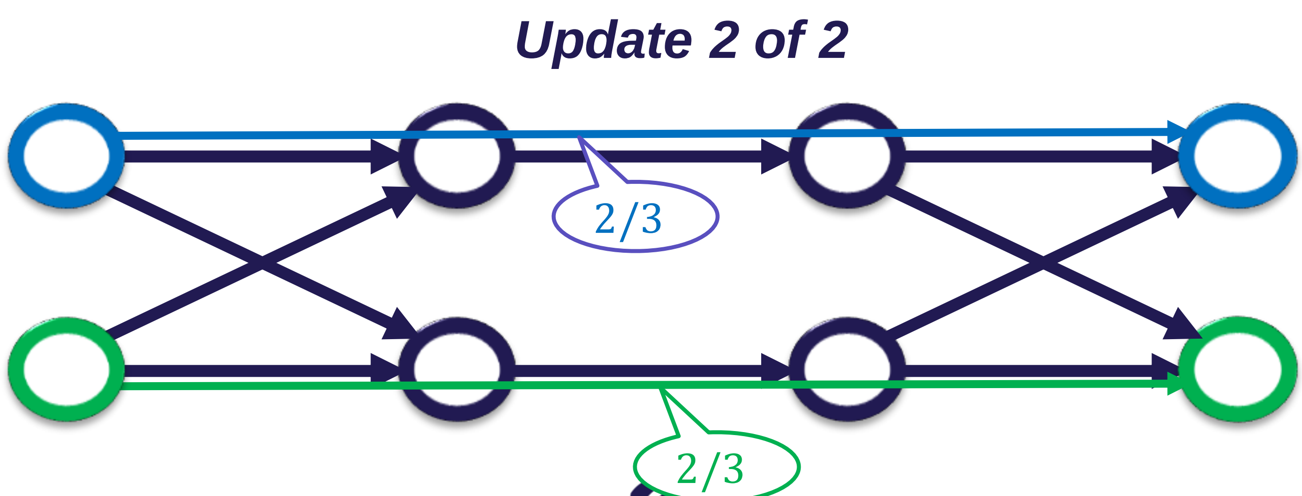
Update 1 of 2



Consistent Migration of Flows

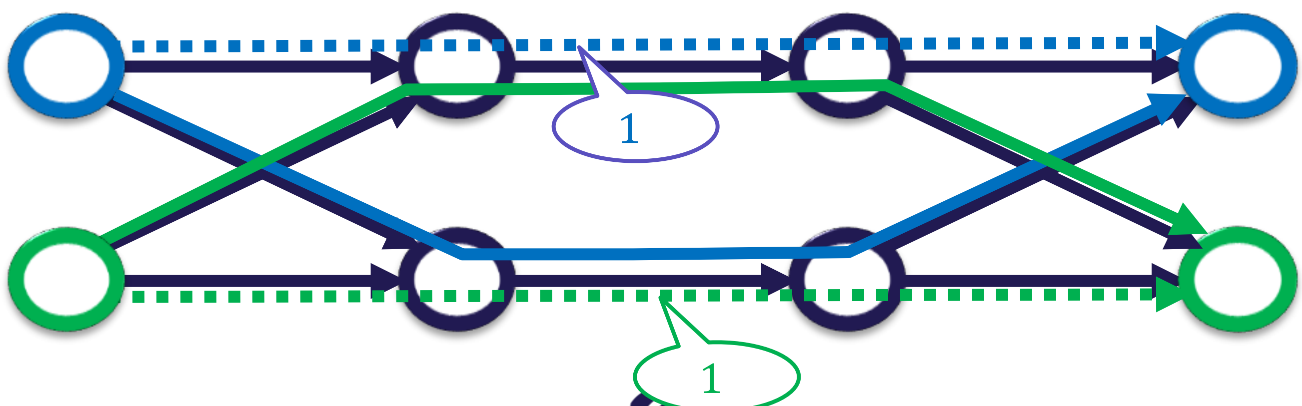
Approach of *SWAN*: use slack x (i.e., **%**)

- Here $x = 1/3$
- Move slack $x \Rrightarrow \lceil 1/x \rceil - 1$ staged partial moves



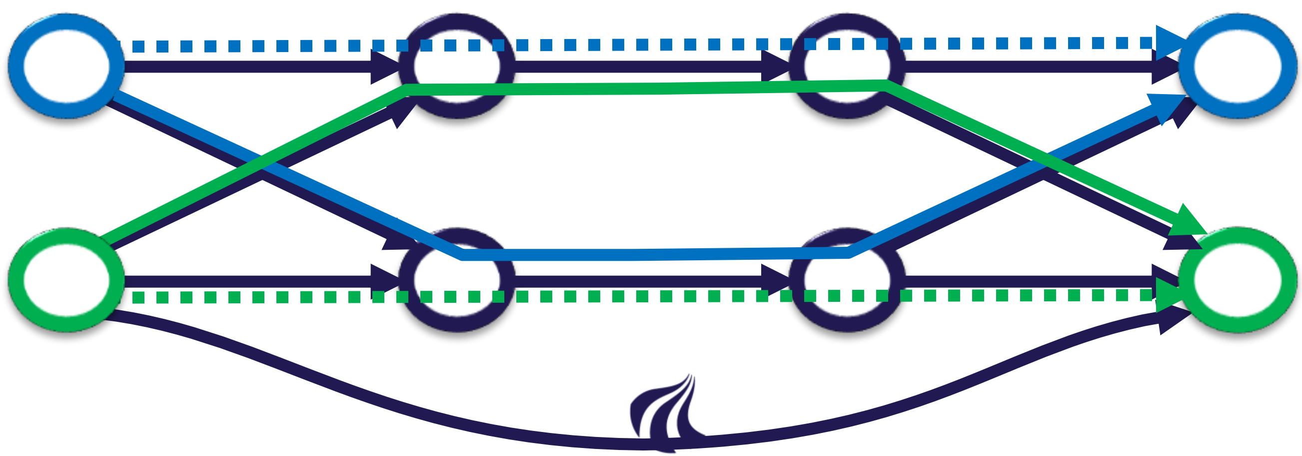
Consistent Migration of Flows

No slack on flow edges?

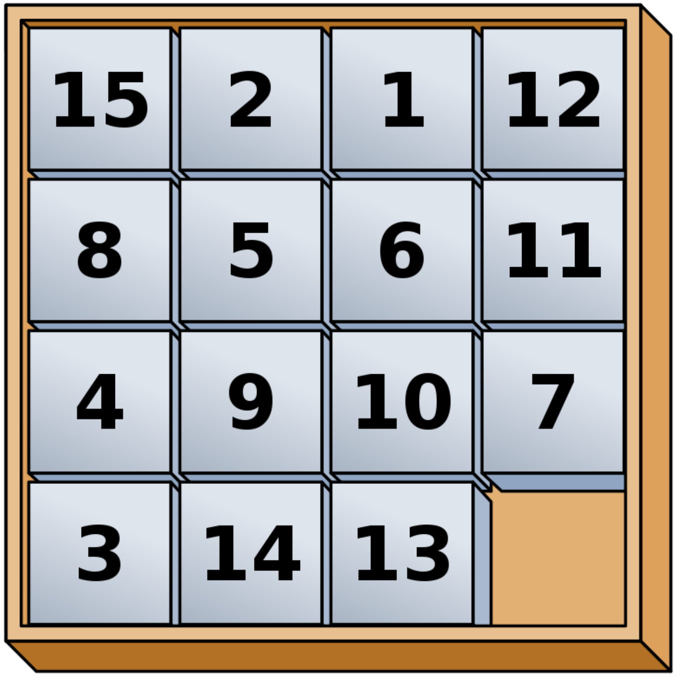


Consistent Migration of Flows

Alternate routes?



Conceptually similar: 15-puzzle



How to move to reach goal?

Generalized:

- Exponentially many possibilities

This variant in **P** (also on graphs)

- 15 puzzle: Johnson 1879, Am. J. of Math.
- ….
- n-1 agents: Kornhauser et al., FOCS 1984
- ….
- n agents (rotations): Foerster et al., CIAC 2017
- Etc…

To Slack or not to Slack?

Slack of x on all flow edges?

$\lceil 1/x \rceil - 1$ updates

To Slack or not to Slack?

What if not?

Try to create slack

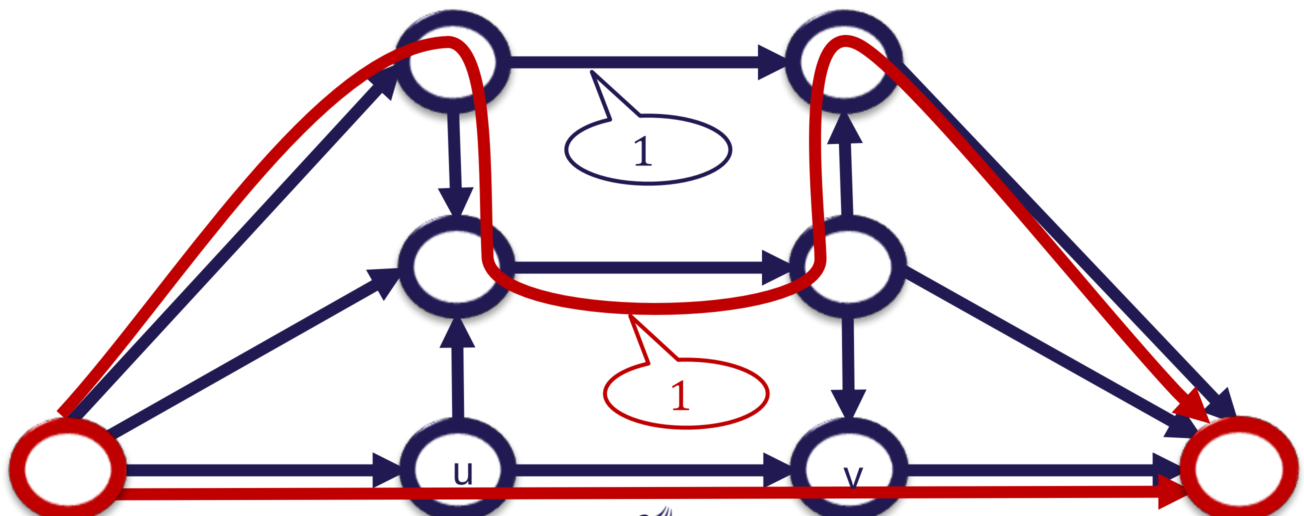
To Slack or not to Slack?

Combinatorial approach

Augmenting paths

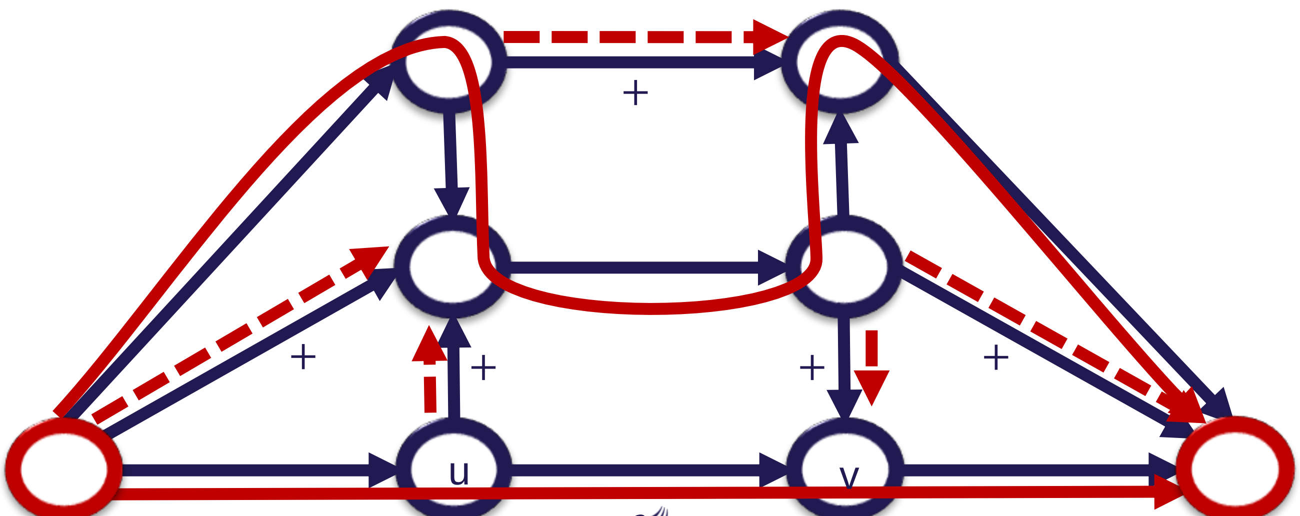
Combinatorial Approach

Move single commodities at a time



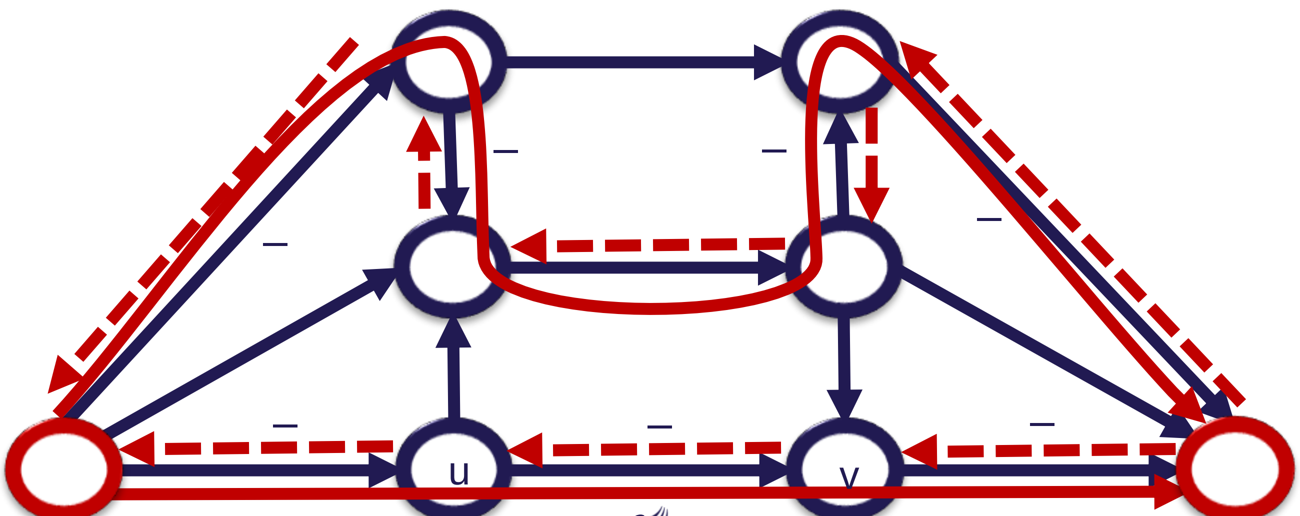
Combinatorial Approach

Where to increase flow?



Combinatorial Approach

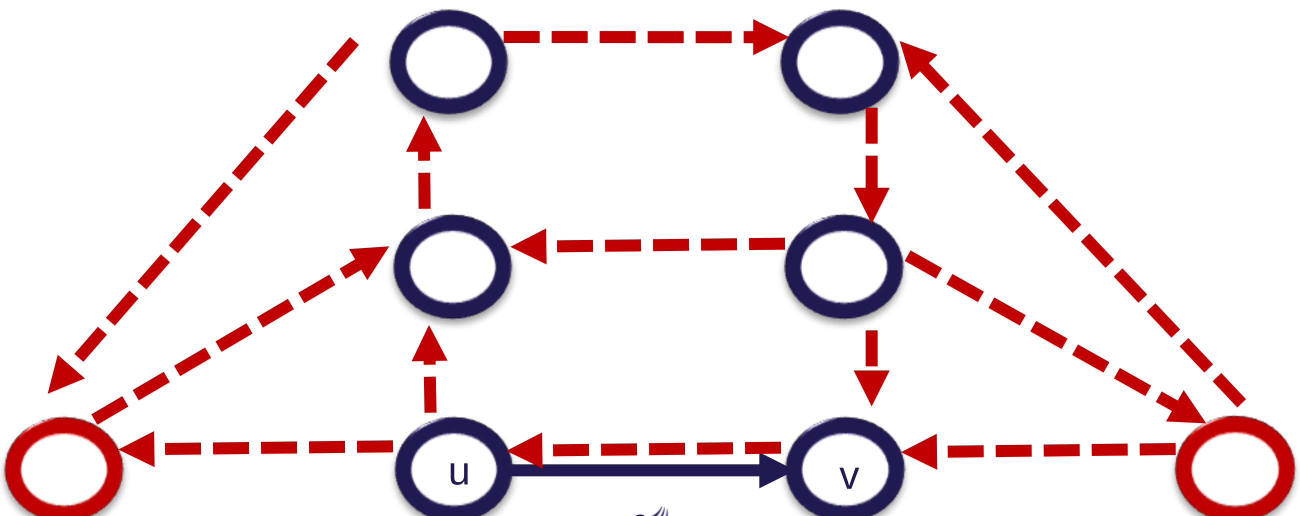
Where to push back flow?



AALBORG UNIVERSITY
DENMARK

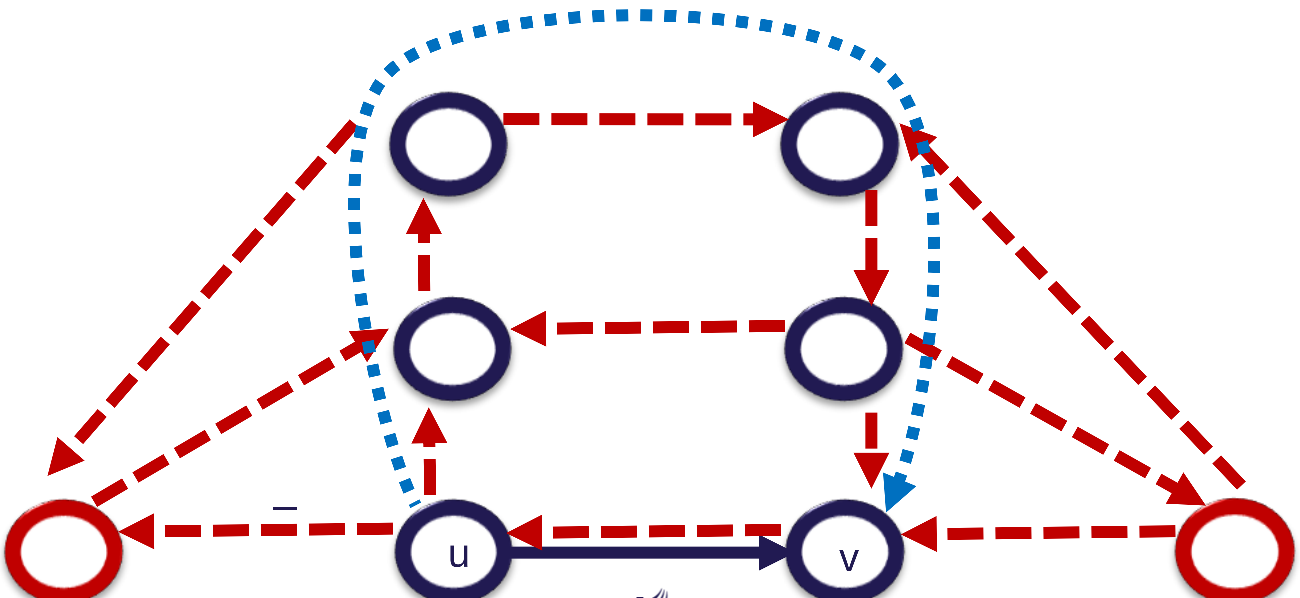
Combinatorial Approach

Resulting residual network



Combinatorial Approach

We found an augmenting path $\Rightarrow$ create slack on e



High-level Algorithm Idea

No slack on flow edges? Find augmenting paths

On both initial and desired state

Success? Use slack to migrate

Can't create slack on some flow edge?

Consistent migration impossible
By contradiction (else augmenting paths would create slack)

Runtime: $O(Fm^3)$

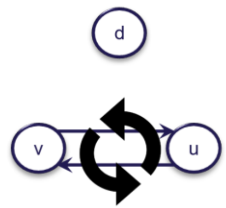
(F being #commodities, m being #edges)

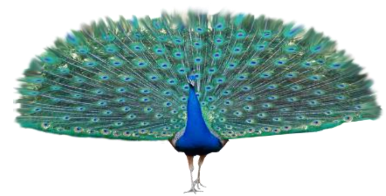
Brandt et al.: On Consistent Migration of Flows in SDNs (INFOCOM 2016).

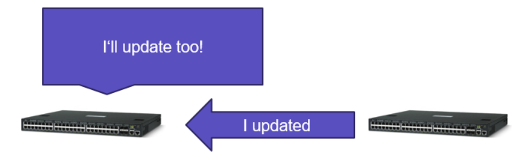
Algorithmic Ideas Overview

Loop Freedom

- Greedy
- Relax: Peacock
- Proof Labeling

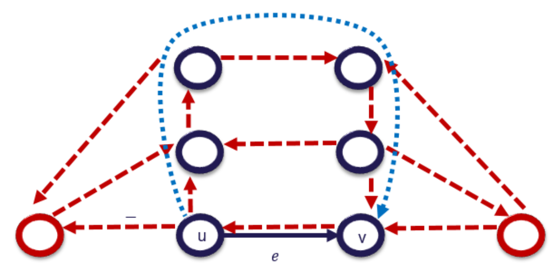






Consistent Flow Migration

- Standard: integer/linear* programs
- Alternative: augmenting flows



*polynomial runtime***?**

TOWARDS LOSSLESS DATA CENTER RECONFIGURATION: CONSISTENT NETWORK UPDATES IN SDNS

KLAUS-TYCHO FOERSTER

AALBORG UNIVERSITY
DENMARK