TI-MFA: Keep Calm and Reroute Segments Fast

Klaus-Tycho Foerster
University of Vienna, Austria

Mahmoud Parham
University of Vienna, Austria

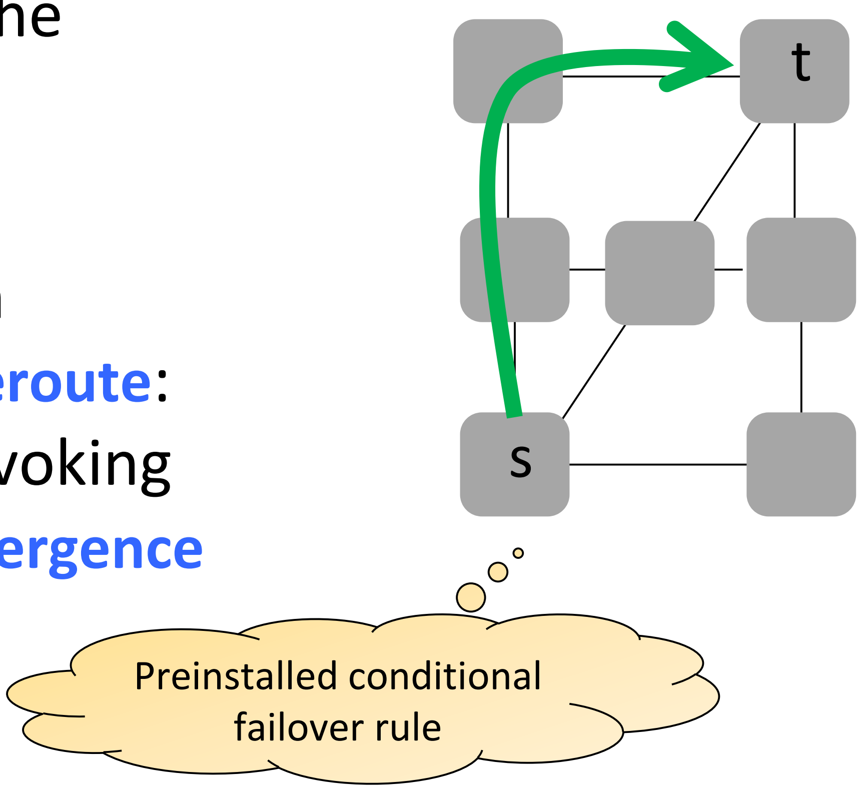
Marco Chiesa
KTH, Sweden

Stefan Schmid
University of Vienna, Austria

Fast Rerouting (FRR)

- Networks (enterprise networks, datacenter networks, Internet): **critical infrastructure** of the information society

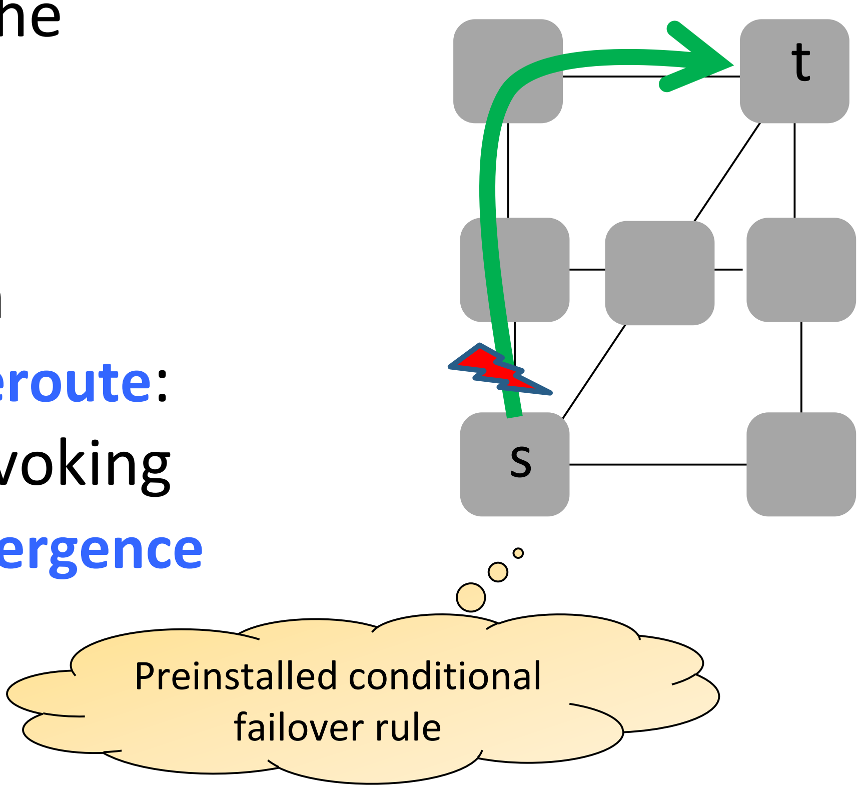
- Modern communication networks support **fast reroute**: local failover without invoking control plane, **no reconvergence**



Fast Rerouting (FRR)

- Networks (enterprise networks, datacenter networks, Internet): **critical infrastructure** of the information society

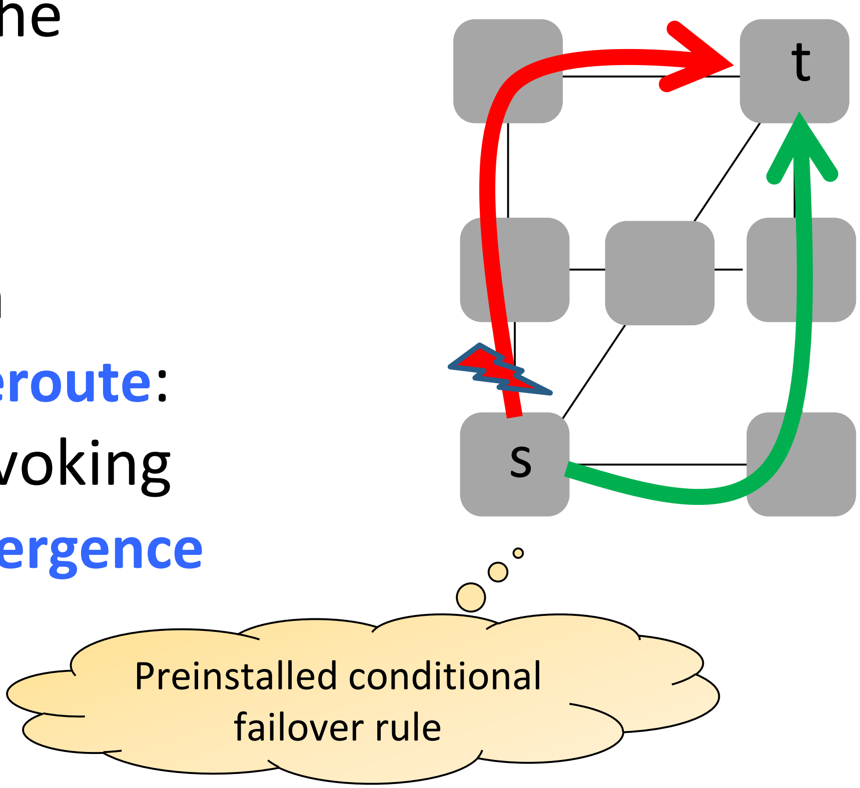
- Modern communication networks support **fast reroute**: local failover without invoking control plane, **no reconvergence**



Fast Rerouting (FRR)

- Networks (enterprise networks, datacenter networks, Internet): **critical infrastructure** of the information society

- Modern communication networks support **fast reroute**: local failover without invoking control plane, **no reconvergence**

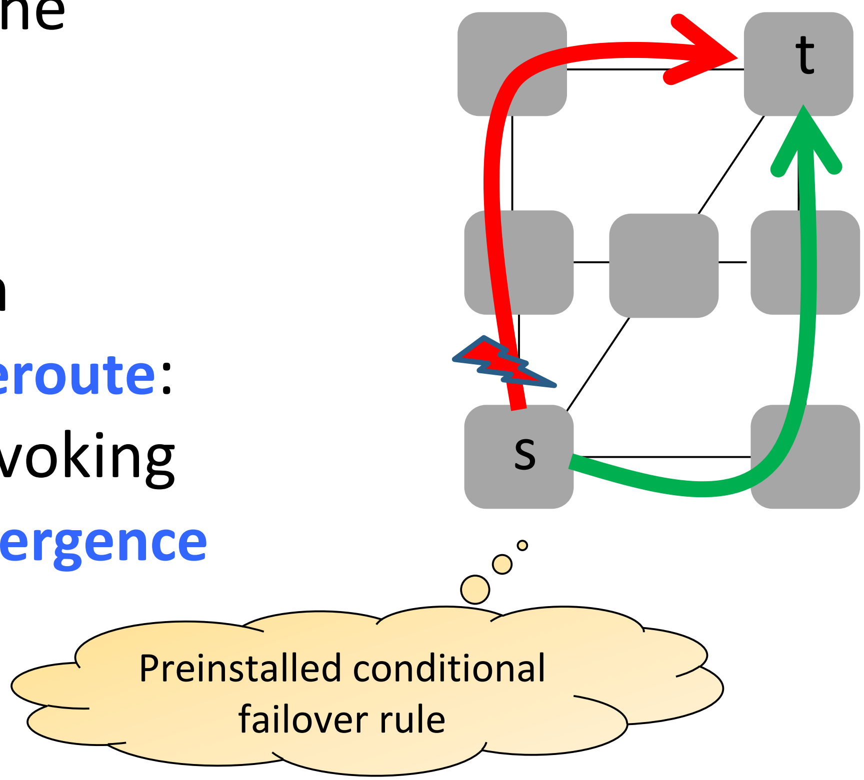


Fast Rerouting (FRR)

- Networks (enterprise networks, datacenter networks, Internet): **critical infrastructure** of the information society

- Modern communication networks support **fast reroute**: local failover without invoking control plane, **no reconvergence**

Good alternative under 1 failure!

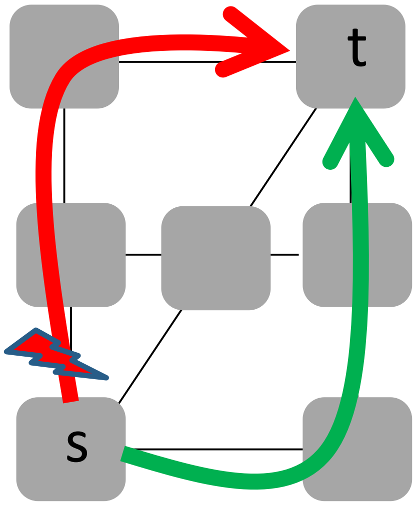


Fast Rerouting (FRR)

- Networks (enterprise networks, datacenter networks, Internet): **critical infrastructure** of the information society

- Modern communication networks support **fast reroute**: local failover without invoking control plane, **no reconvergence**

Good alternative under 1 failure!



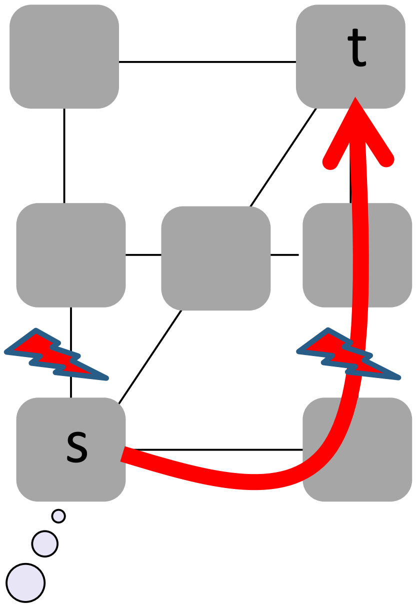
E.g., conventional IGP-based restauration requires notifying all routers about failure: 100s ms. IP FRR much faster.

Fast Rerouting (FRR)

- Networks (enterprise networks, datacenter networks, Internet): **critical infrastructure** of the information society

- Modern communication networks support **fast reroute**: local failover without invoking control plane, **no reconvergence**

What if there is another failure?



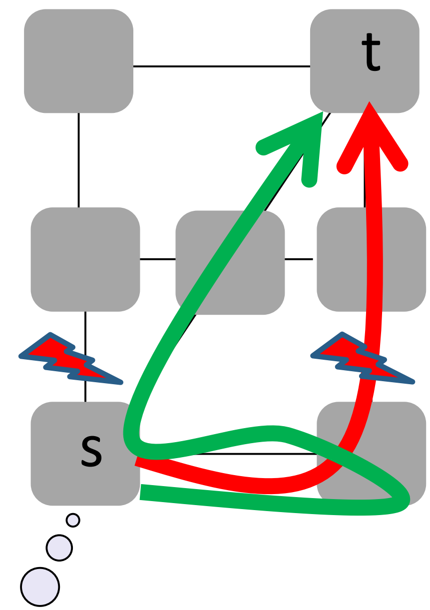
Challenge: conditional rules can only depend on local failures

Fast Rerouting (FRR)

- Networks (enterprise networks, datacenter networks, Internet): **critical infrastructure** of the information society

- Modern communication networks support **fast reroute**: local failover without invoking control plane, **no reconvergence**

What if there is another failure?



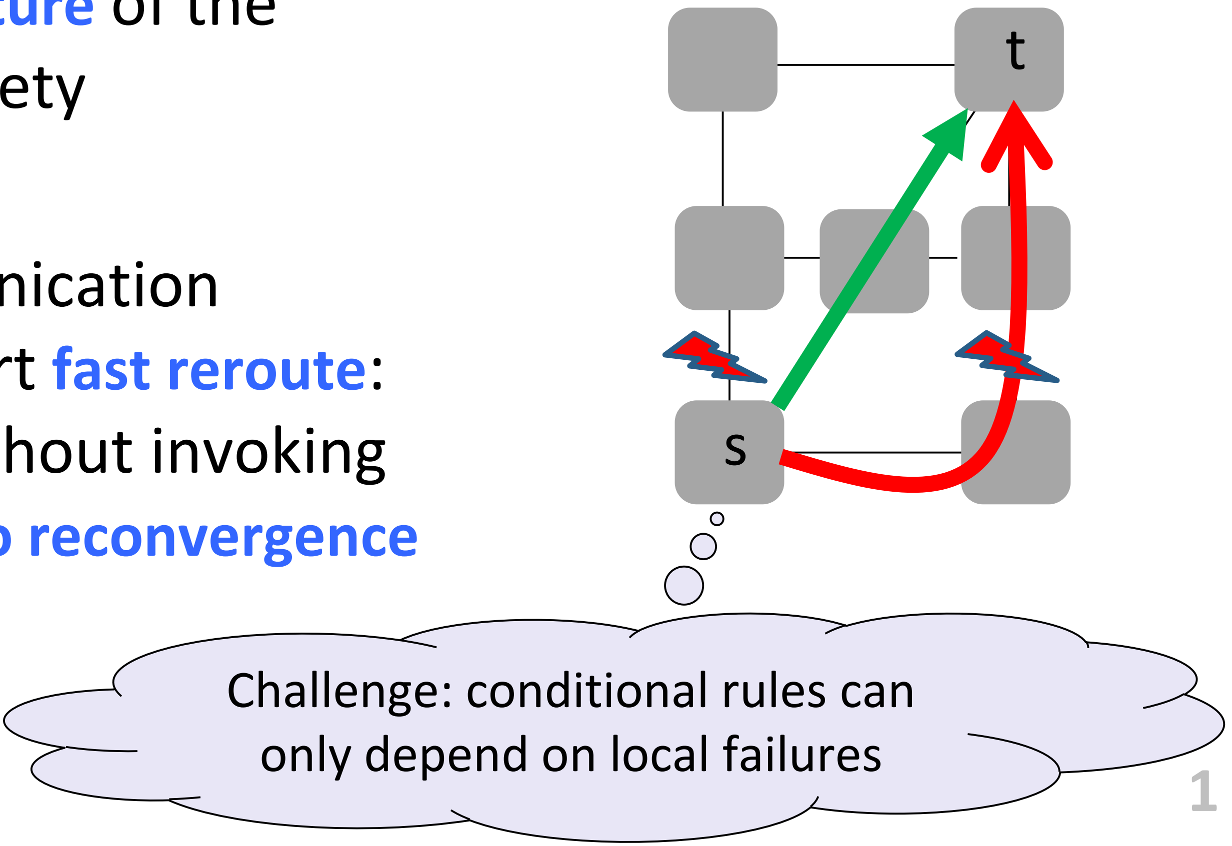
Challenge: conditional rules can only depend on local failures

Fast Rerouting (FRR)

- Networks (enterprise networks, datacenter networks, Internet): **critical infrastructure** of the information society

- Modern communication networks support **fast reroute**: local failover without invoking control plane, **no reconvergence**

Given 2nd failure, this would have been better!



A Fundamental Algorithmic Problem

How to define these **conditional** (local) failover rules?

Challenges:

- Rules have **local knowledge** only: can depend only on incident failures
- Want to minimize additional information that packets should **carry in header**

Some Recent Results: Arborescence-Based (Chiesa et al.)

Known result: always exist in k-connected graphs (efficient)

E.g., Chiesa et al.:

- Given:
  - **k-connected** network *G*, **destination d**
  - *G* decomposed into **k d-rooted arc-disjoint spanning arborescences**

Basic principle:

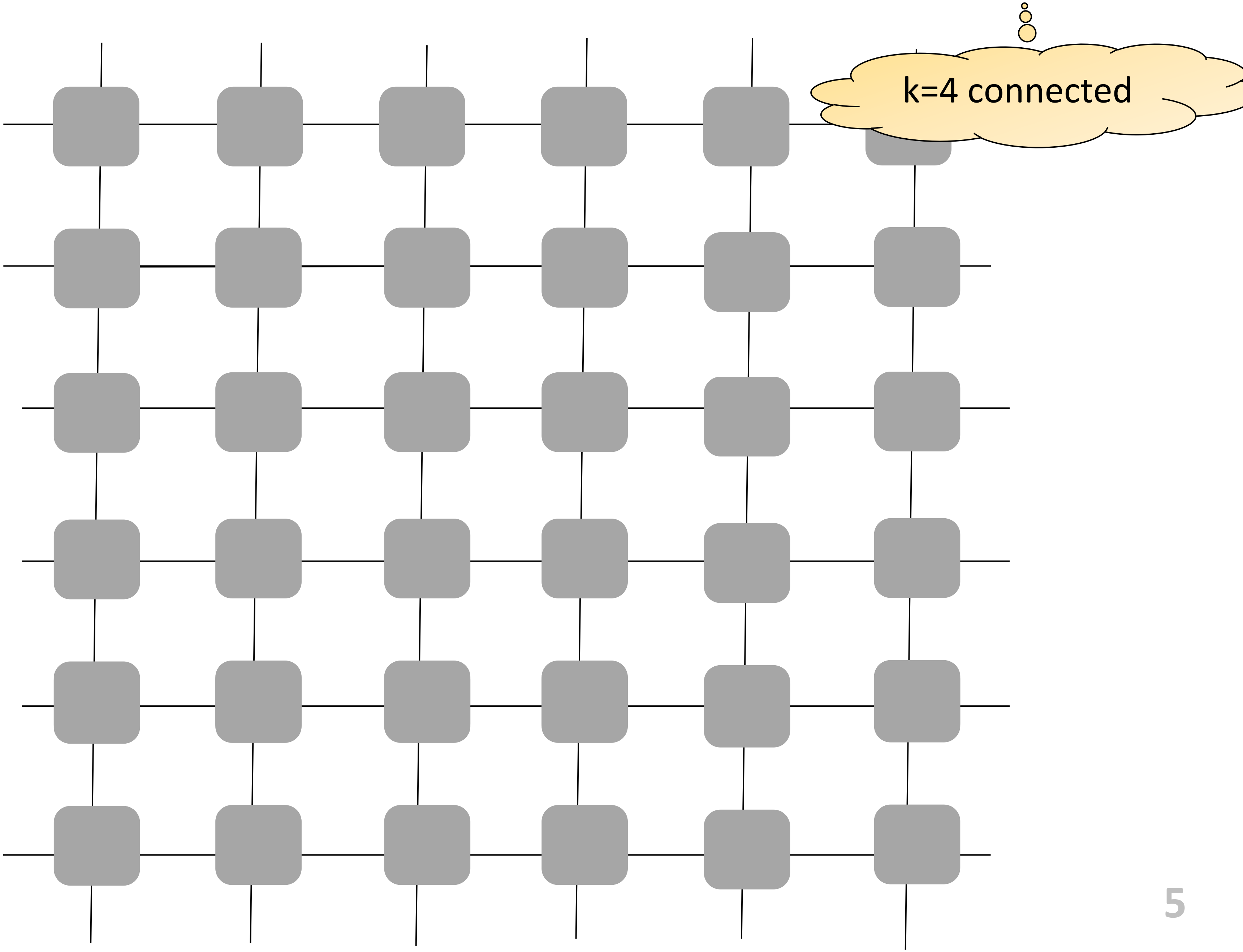
- Route along fixed arborescence ("directed spanning tree") towards the destination *d*
- If packet hits a failed edge at vertex *v*, reroute along a different arborescence

The Crux: which arborescence to choose next? Influences resiliency!

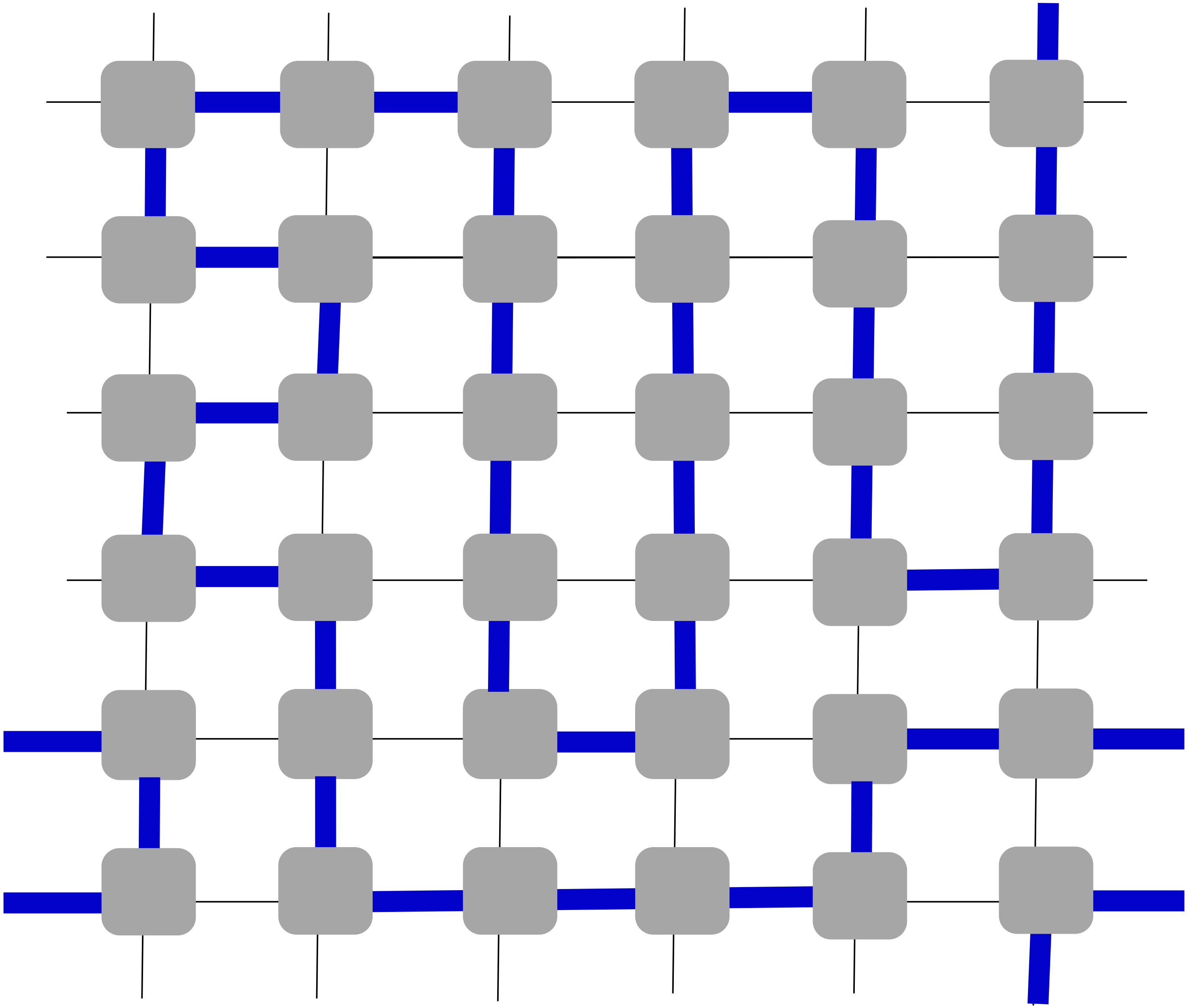
Simple Example: Hamilton Cycle

Chiesa et al.: if k-connected graph has k arc disjoint Hamilton Cycles, k-1 resilient routing can be constructed!

Example: 3-Resilient Routing Function for 2-dim Torus

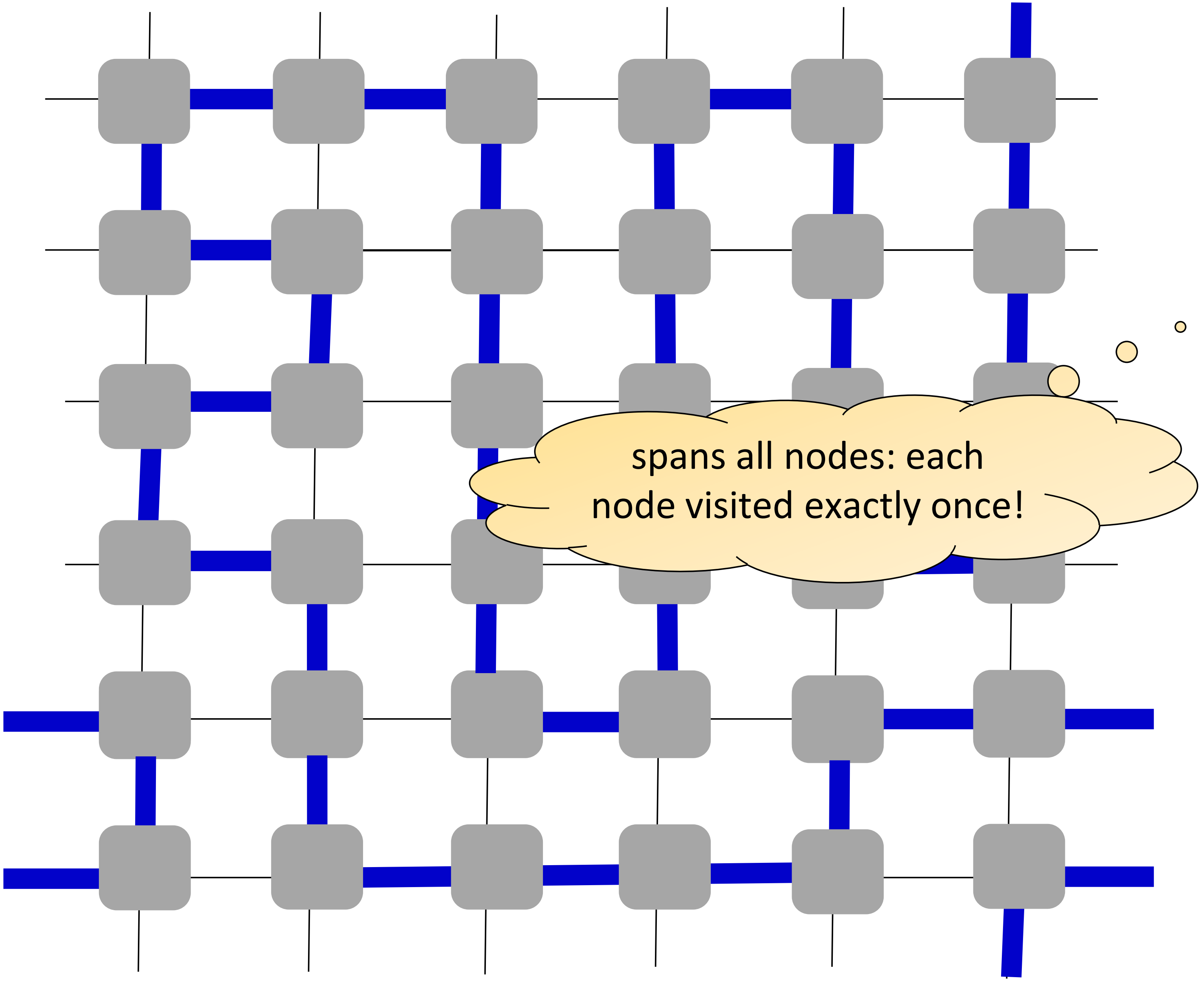


Example: 3-Resilient Routing Function for 2-dim Torus



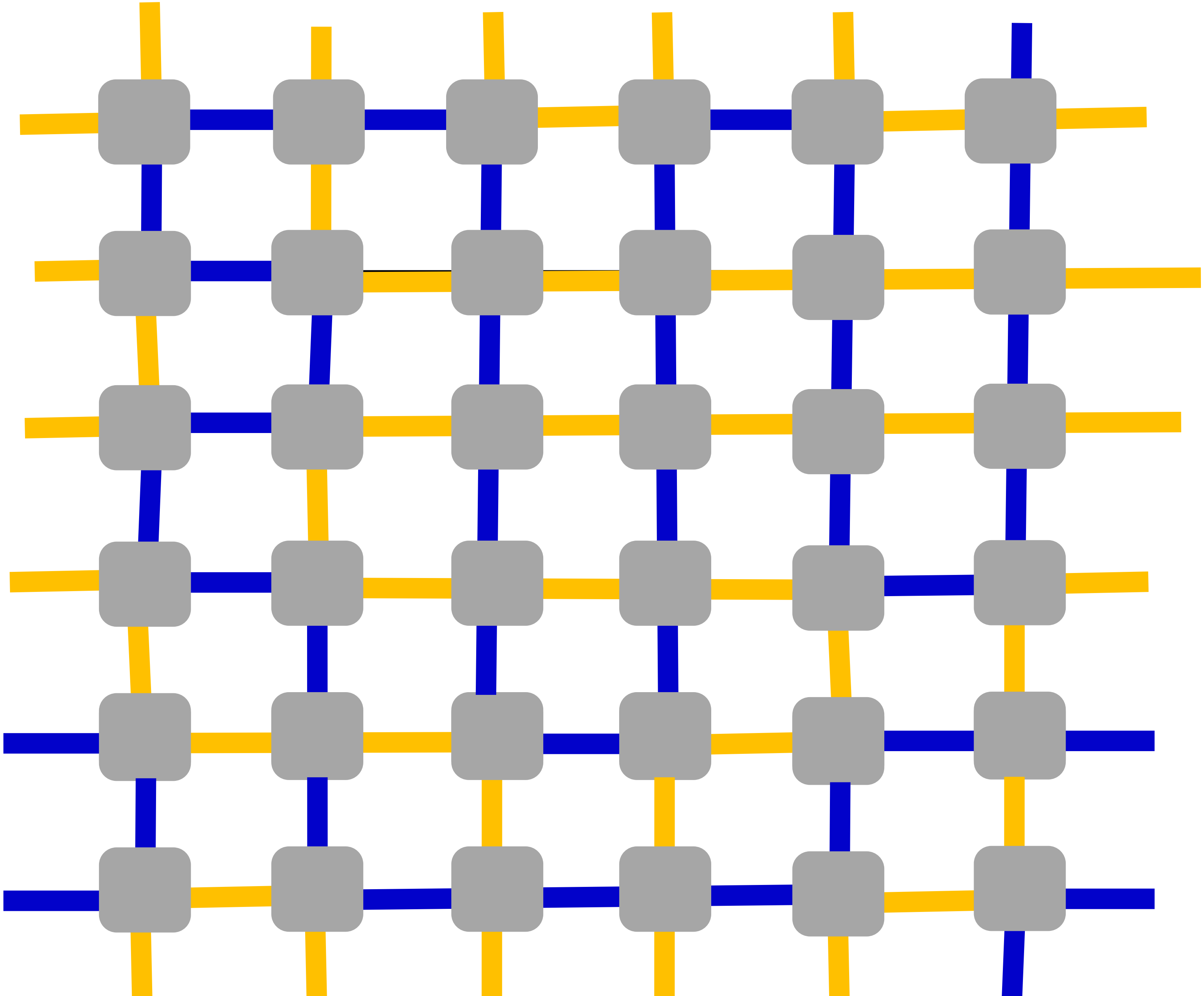
Edge-Disjoint Hamilton Cycle 1

Example: 3-Resilient Routing Function for 2-dim Torus



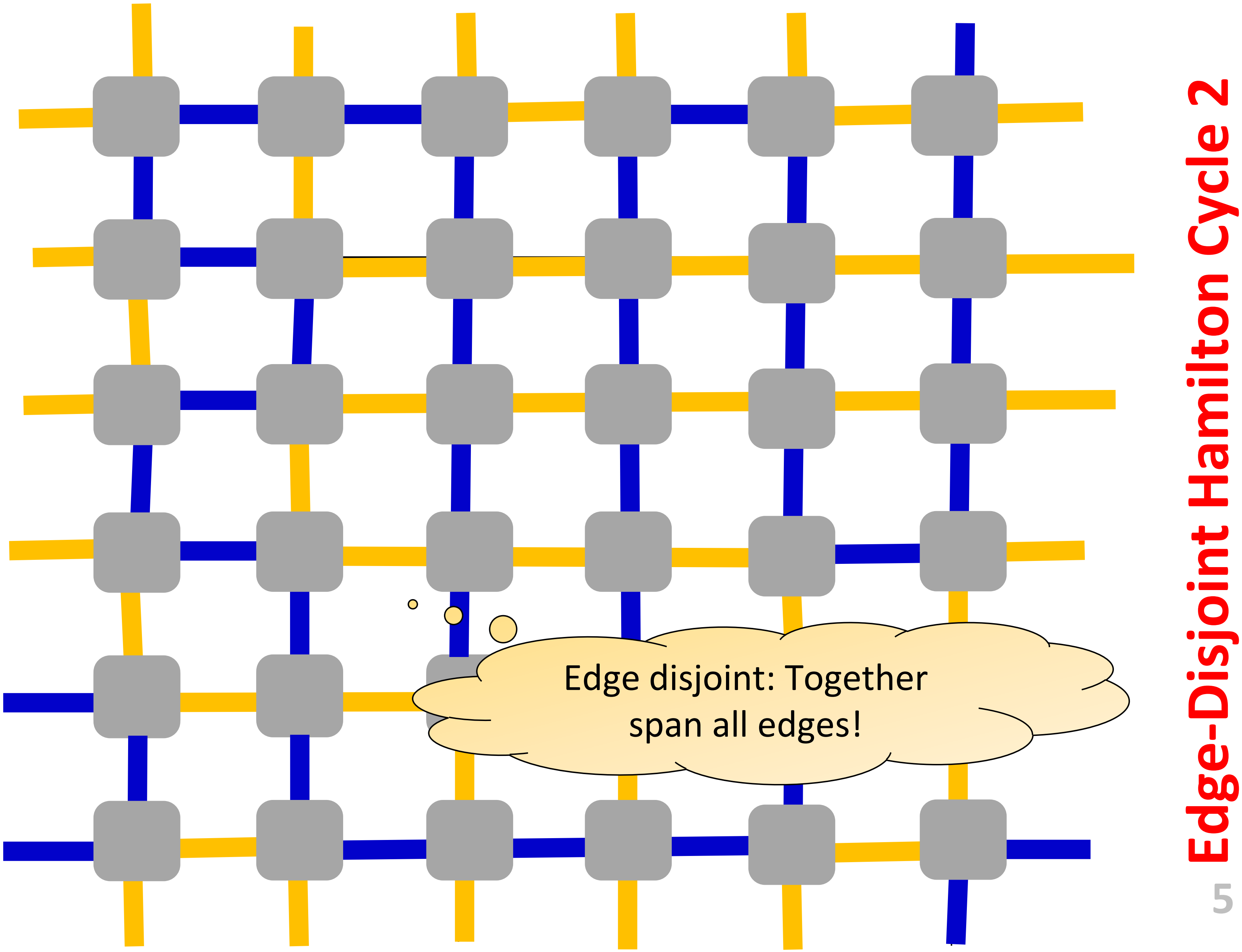
Edge-Disjoint Hamilton Cycle 1

Example: 3-Resilient Routing Function for 2-dim Torus

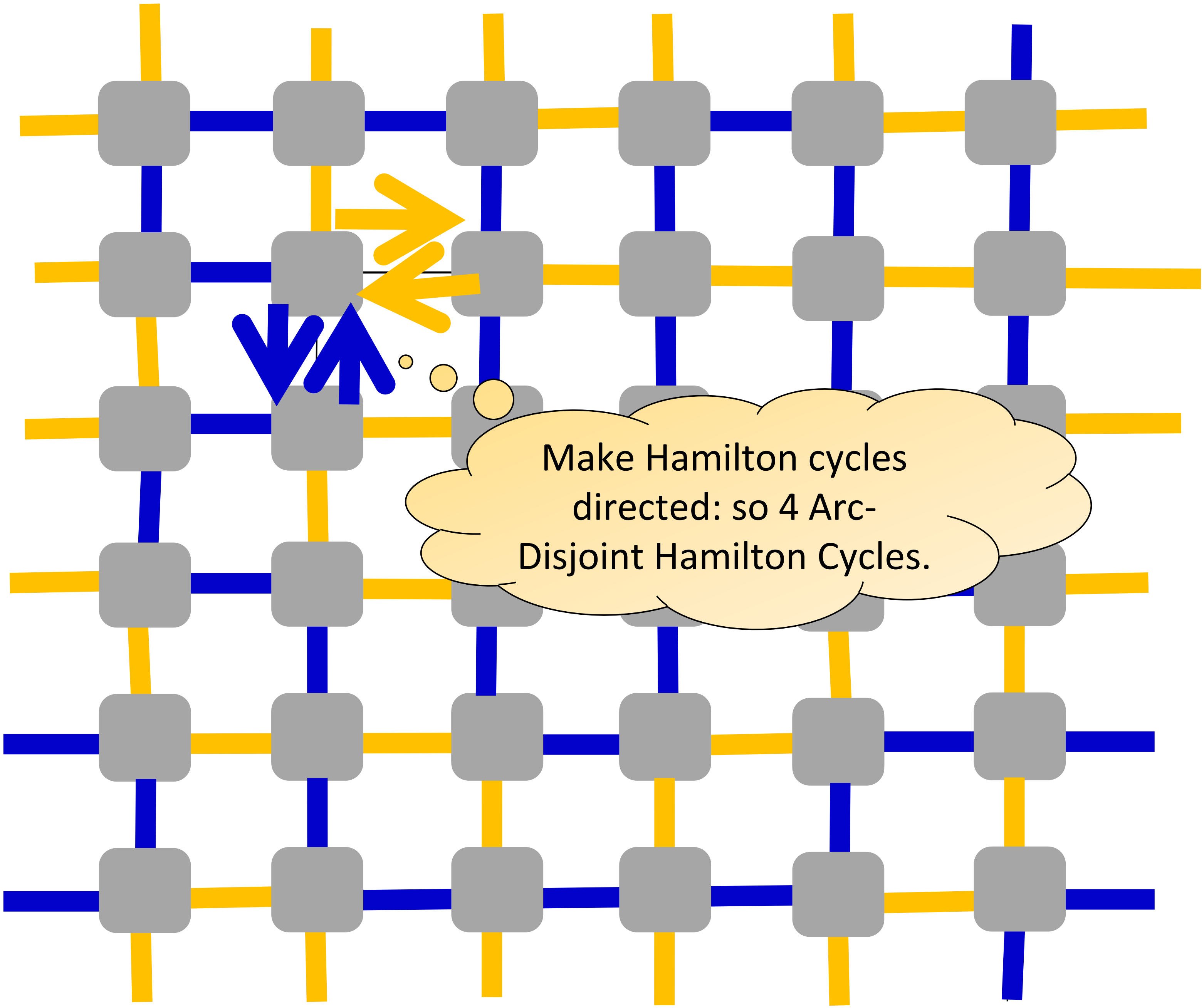


Edge-Disjoint Hamilton Cycle 2

Example: 3-Resilient Routing Function for 2-dim Torus

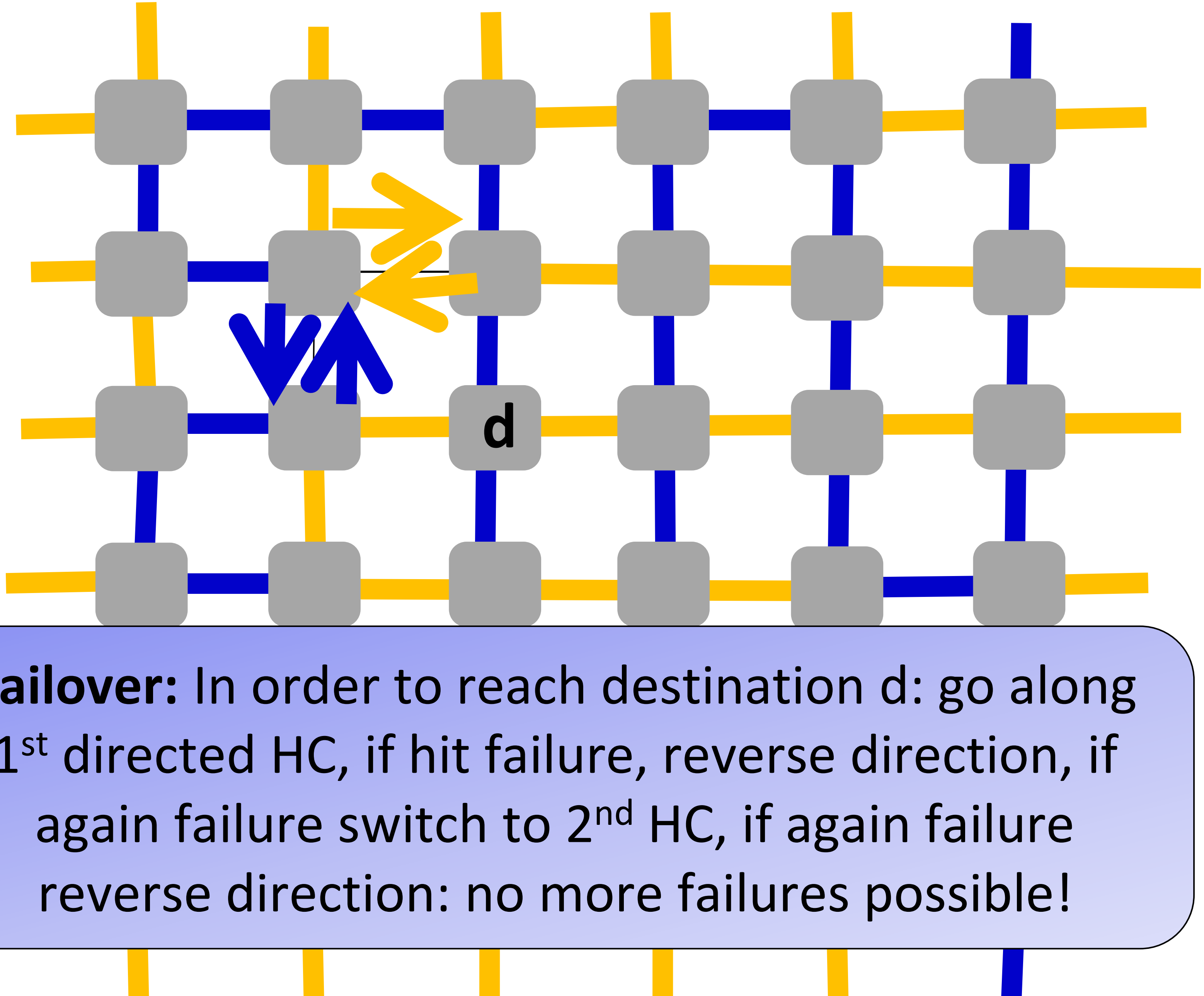


Example: 3-Resilient Routing Function for 2-dim Torus



4 Arc-Disjoint Arborescences

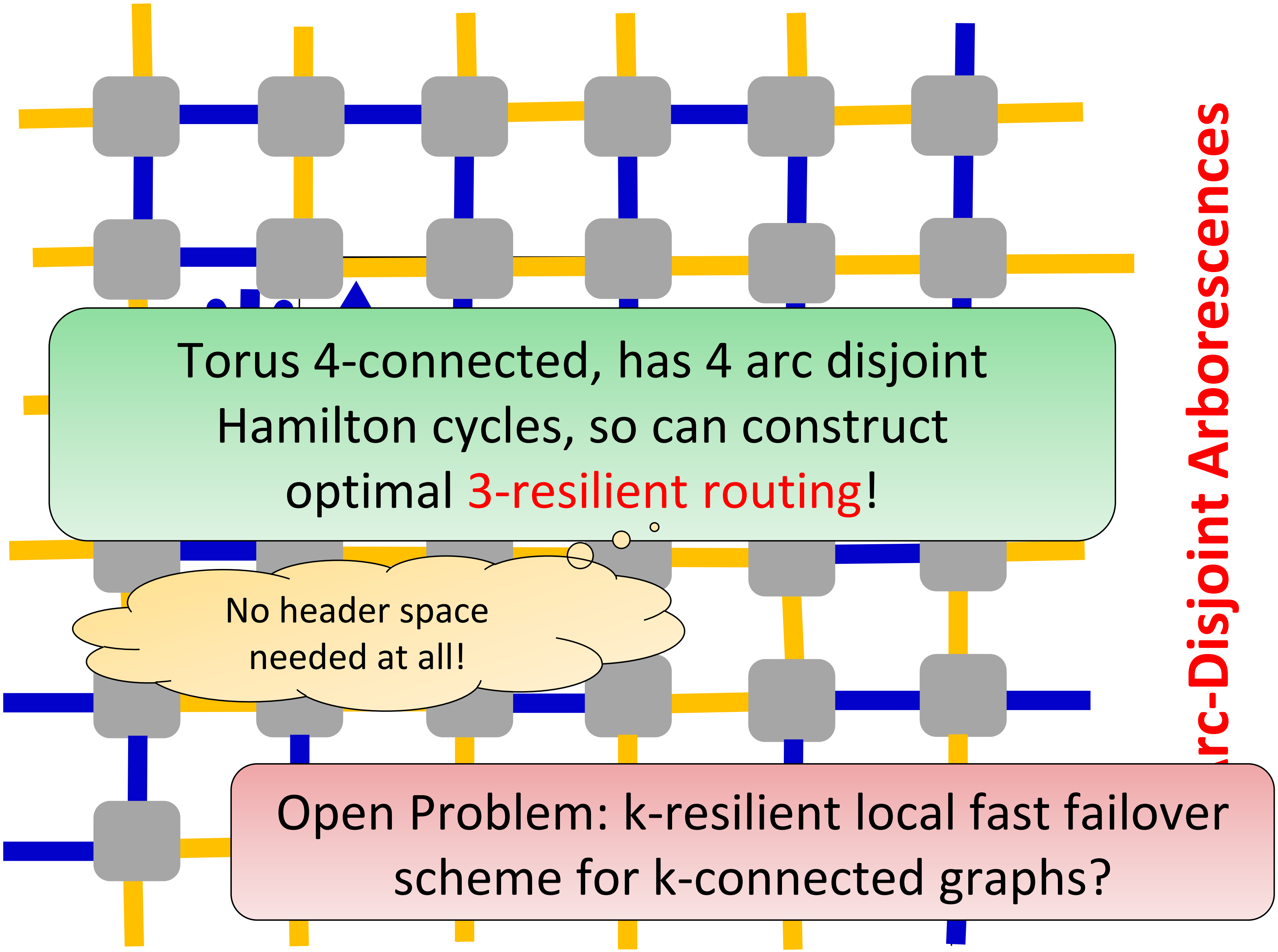
Example: 3-Resilient Routing Function for 2-dim Torus



Failover: In order to reach destination d: go along 1st directed HC, if hit failure, reverse direction, if again failure switch to 2nd HC, if again failure reverse direction: no more failures possible!

4 Arc-Disjoint Arborescences

Example: 3-Resilient Routing Function for 2-dim Torus



Variants with Stretch and Load Guarantees: Pignolet et al. & Foerster et al.

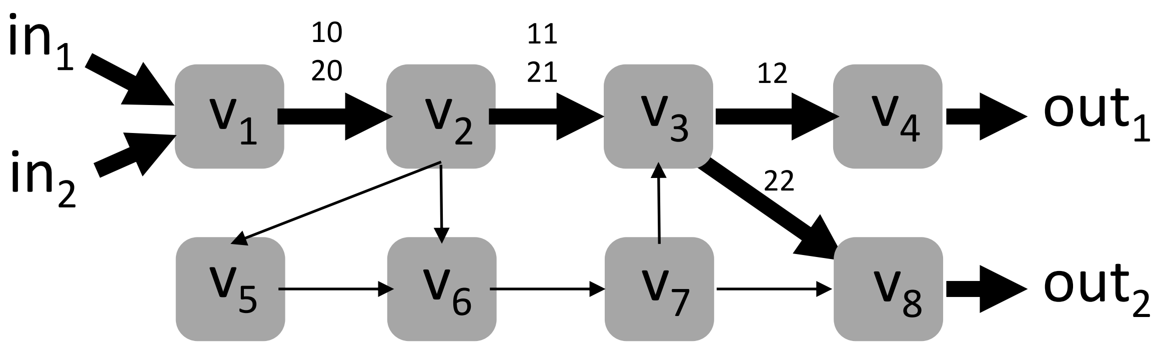
- Local Fast Failover Routing With Low Stretch
  Klaus-Tycho Foerster, Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Tredan. ACM SIGCOMM Computer Communication Review (**CCR**), 2018.

Based on Balanced Incomplete Block Designs (BIBDs): Distributed computing without communication.

- Load-Optimal Local Fast Rerouting for Dependable Networks
  Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Tredan.
  47th IEEE/IFIP International Conference on Dependable Systems and Networks (**DSN**), Denver, Colorado, USA, June 2017.

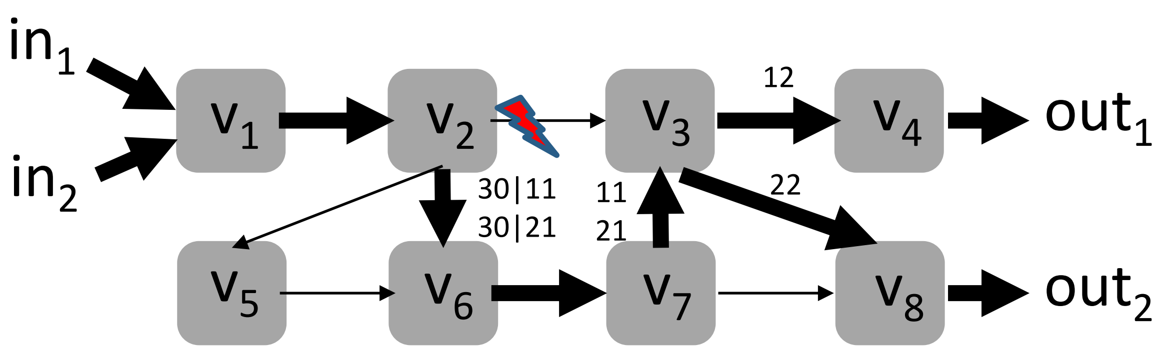
Some Recent Results: Polynomial-Time What-If Analysis for MPLS

- MPLS: forwarding based on **top label** of label **stack**



Default routing of two flows

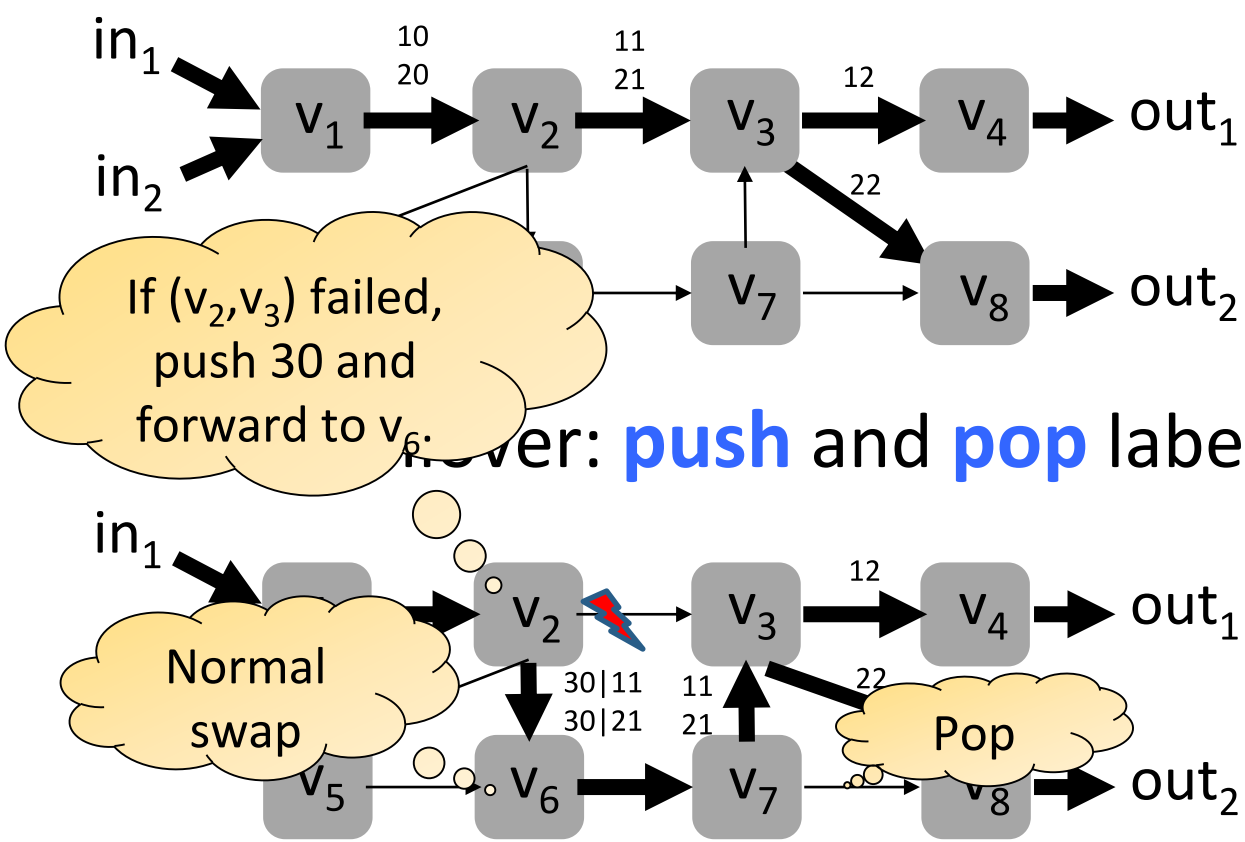
- For failover: **push** and **pop** label



One failure: push 30: route around (v_2,v_3)

Some Recent Results: Polynomial-Time What-If Analysis for MPLS

- MPLS: forwarding based on **top label** of label **stack**



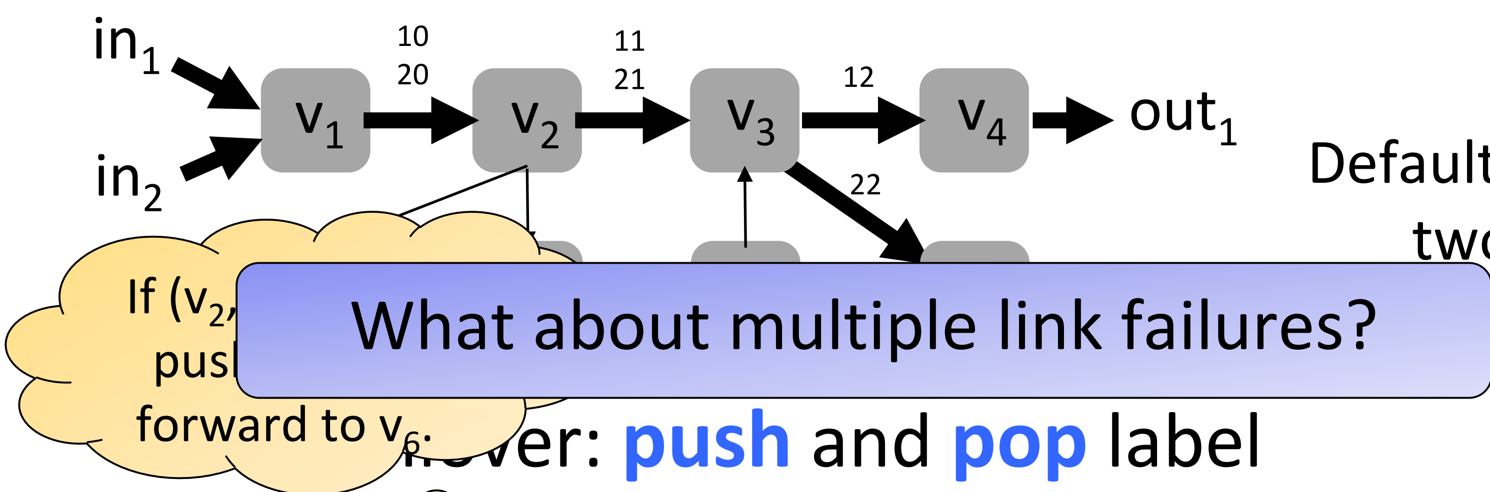
Default routing of two flows

- Failover: **push** and **pop** label

One failure: push 30: route around (v_2,v_3)

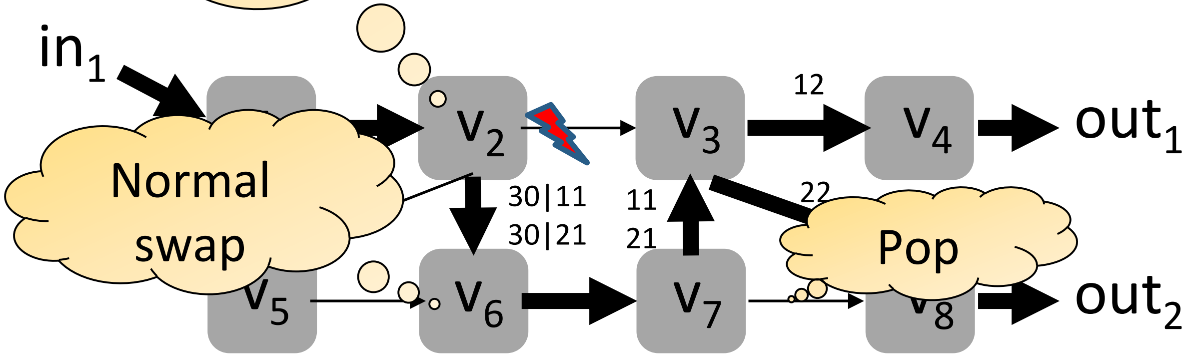
Some Recent Results: Polynomial-Time What-If Analysis for MPLS

- MPLS: forwarding based on **top label** of label **stack**



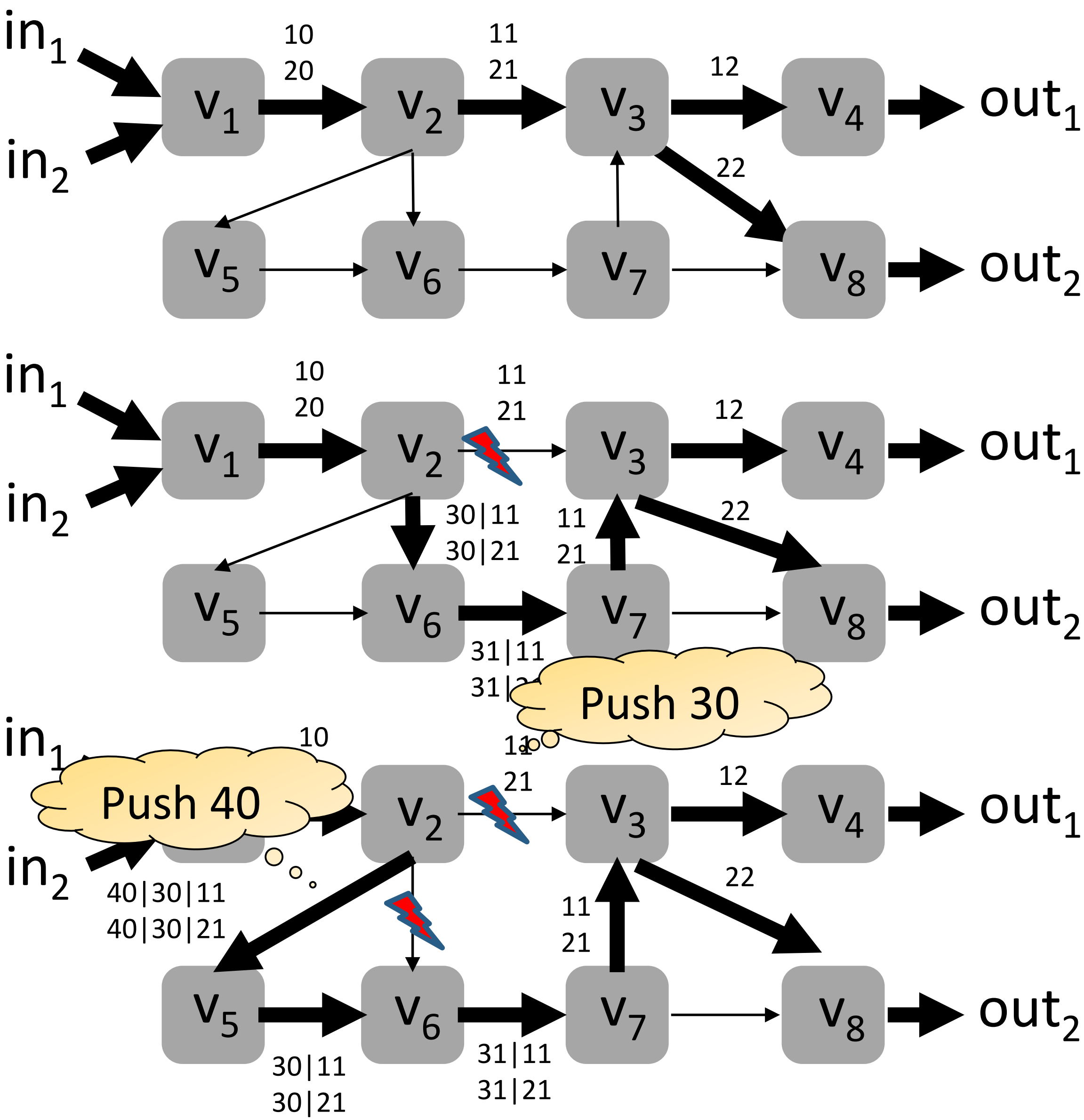
Default routing of two flows

- Failover: **push** and **pop** label



One failure: push 30: route around (v_2,v_3)

MPLS Networks: 2 Failures



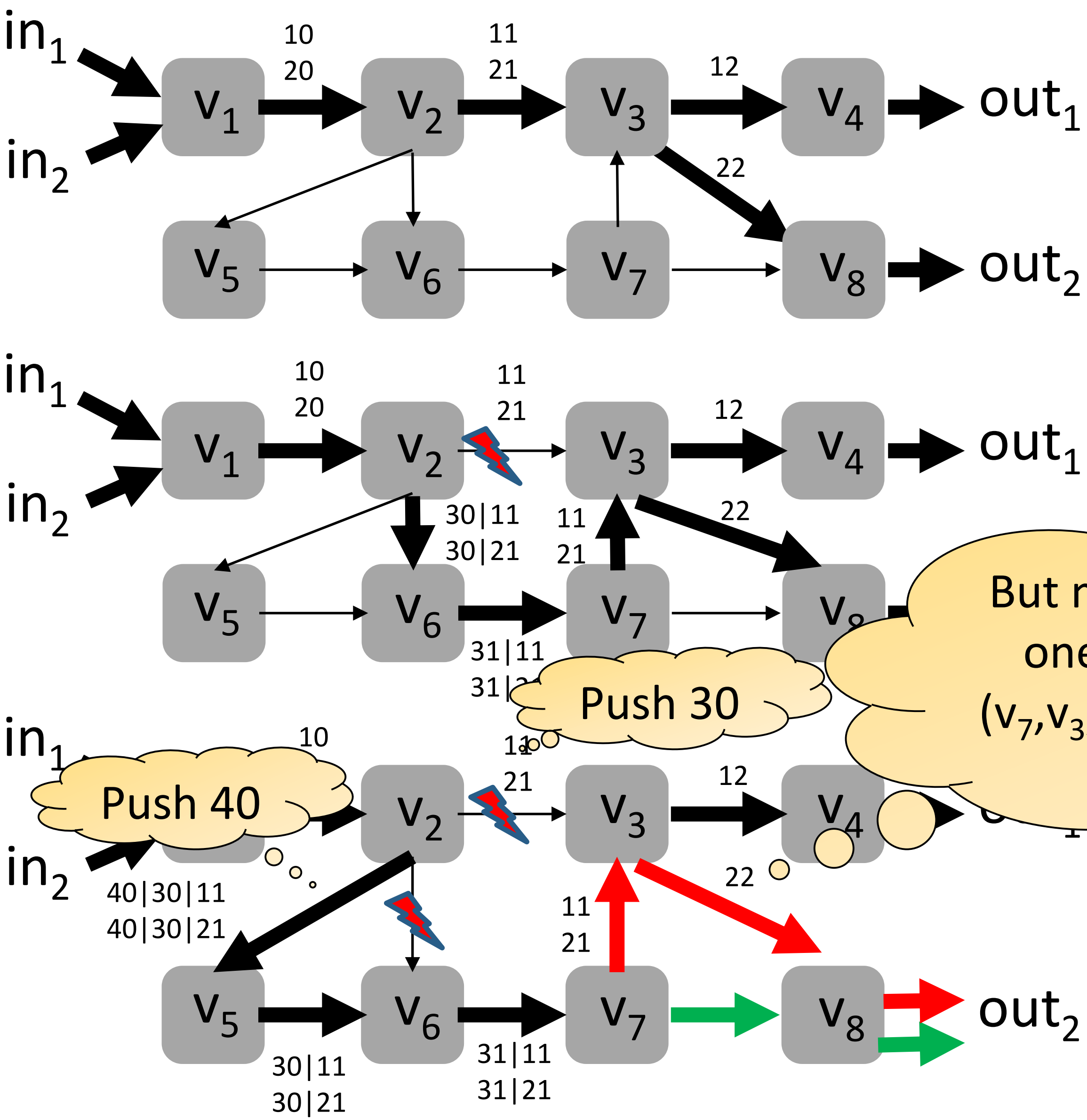
Original Routing

One failure: push 30: route around (v_2,v_3)

Two failures: first push 30: route around (v_2,v_3)

Push recursively 40: route around (v_2,v_6)

MPLS Networks: 2 Failures



Original Routing

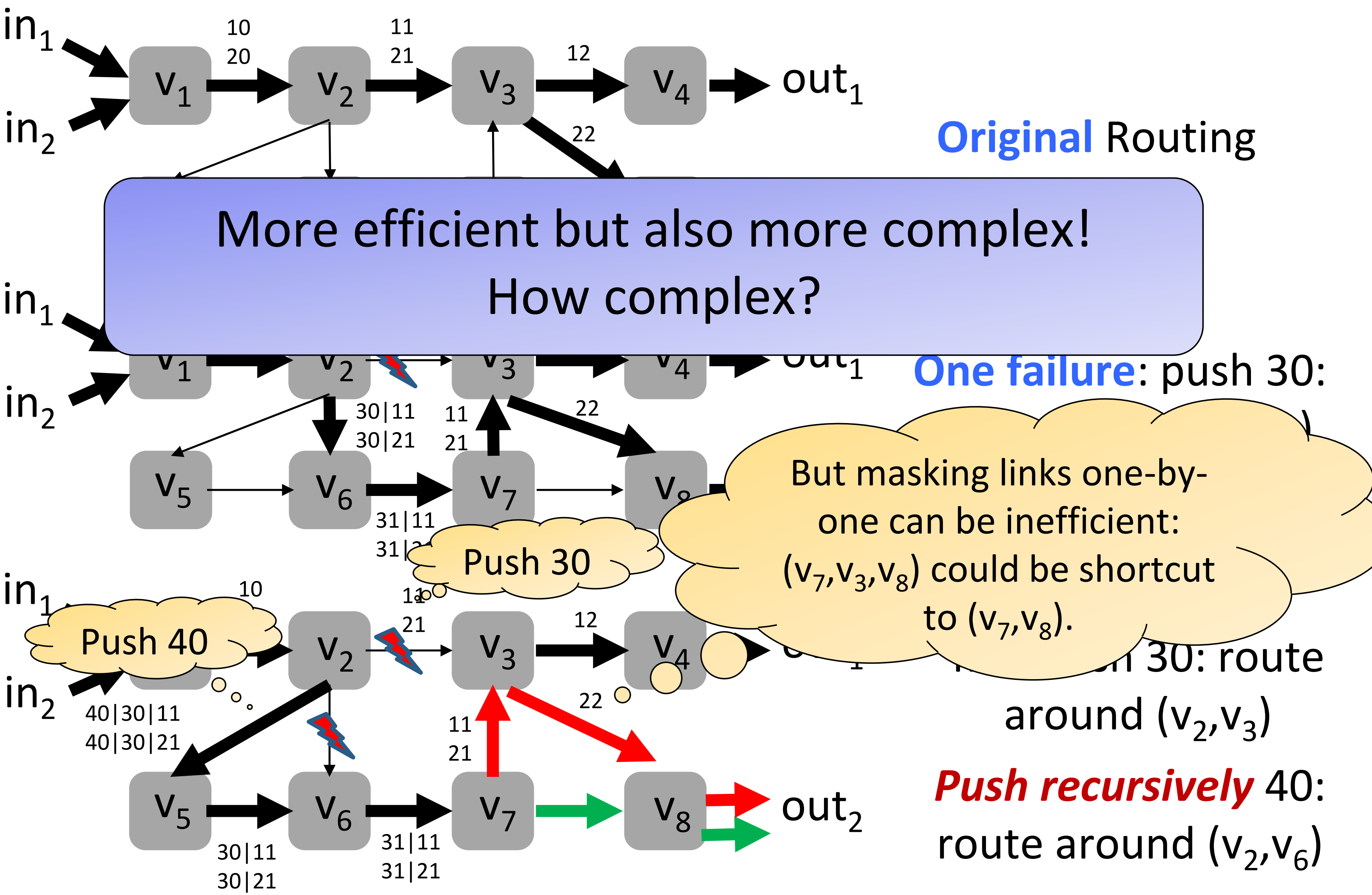
One failure: push 30:

But masking links one-by-one can be inefficient: (v_7,v_3,v_8) could be shortcut to (v_7,v_8).

30: route around (v_2,v_3)

Push recursively 40: route around (v_2,v_6)

MPLS Networks: 2 Failures



Forwarding Tables for Our Example

FT	In-I	In-Label	Out-I	op
τ_{v_1}	in_1	$\perp$	(v_1, v_2)	$push(10)$
	in_2	$\perp$	(v_1, v_2)	$push(20)$
τ_{v_2}	(v_1, v_2)	10	(v_2, v_3)	$swap(11)$
	(v_1, v_2)	20	(v_2, v_3)	$swap(21)$
τ_{v_3}	(v_2, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_2, v_3)	21	(v_3, v_8)	$swap(22)$
	(v_7, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_7, v_3)	21	(v_3, v_8)	$swap(22)$
τ_{v_4}	(v_3, v_4)	12	out_1	pop
τ_{v_5}	(v_2, v_5)	40	(v_5, v_6)	pop
τ_{v_6}	(v_2, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	61	(v_6, v_7)	$swap(62)$
	(v_5, v_6)	71	(v_6, v_7)	$swap(72)$
τ_{v_7}	(v_6, v_7)	31	(v_7, v_3)	pop
	(v_6, v_7)	62	(v_7, v_3)	$swap(11)$
	(v_6, v_7)	72	(v_7, v_8)	$swap(22)$
τ_{v_8}	(v_3, v_8)	22	out_2	pop
	(v_7, v_8)	22	out_2	pop

Flow Table

Protected link

Alternative link

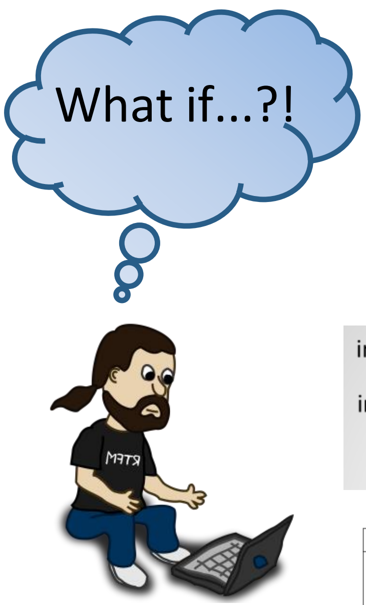
Label

local FFT	Out-I	In-Label	Out-I	op
τ_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$push(30)$
	(v_2, v_3)	21	(v_2, v_6)	$push(30)$
	(v_2, v_6)	30	(v_2, v_5)	$push(40)$

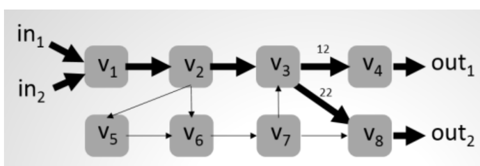
global FFT	Out-I	In-Label	Out-I	op
τ'_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$swap(61)$
	(v_2, v_3)	21	(v_2, v_6)	$swap(71)$
	(v_2, v_6)	61	(v_2, v_5)	$push(40)$
	(v_2, v_6)	71	(v_2, v_5)	$push(40)$

Failover Tables

Can be verified in polynomial time via automata-theoretic approach

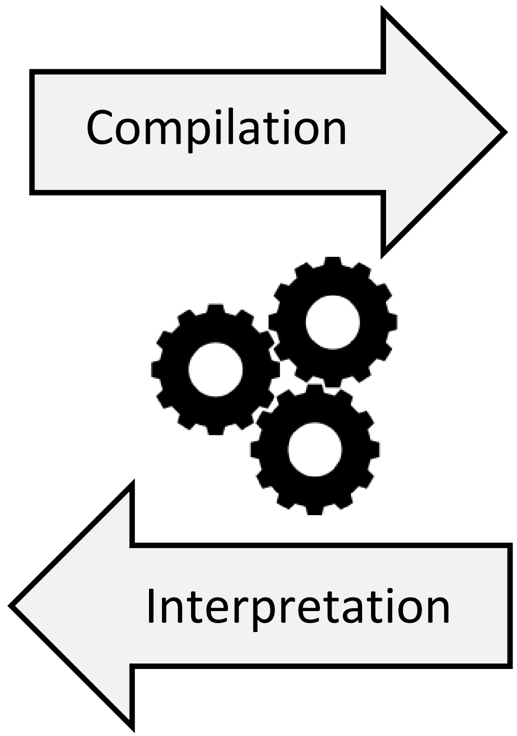


FT	In-I	In-Label	Out-I	op
τ_{v_1}	in_1	$\perp$	(v_1, v_2)	$push(10)$
	in_2	$\perp$	(v_1, v_2)	$push(20)$
τ_{v_2}	(v_1, v_2)	10	(v_2, v_3)	$swap(11)$
	(v_1, v_2)	20	(v_2, v_3)	$swap(21)$
τ_{v_3}	(v_2, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_2, v_3)	21	(v_3, v_8)	$swap(22)$
	(v_7, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_7, v_3)	21	(v_3, v_8)	$swap(22)$
τ_{v_4}	(v_3, v_4)	12	out_1	pop
τ_{v_5}	(v_2, v_5)	40	(v_5, v_6)	pop
τ_{v_6}	(v_2, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	61	(v_6, v_7)	$swap(62)$
	(v_5, v_6)	71	(v_6, v_7)	$swap(72)$
τ_{v_7}	(v_6, v_7)	31	(v_7, v_3)	pop
	(v_6, v_7)	62	(v_7, v_3)	$swap(11)$
	(v_6, v_7)	72	(v_7, v_8)	$swap(22)$
τ_{v_8}	(v_3, v_8)	22	out_2	pop
	(v_7, v_8)	22	out_2	pop



local FFT	Out-I	In-Label	Out-I	op
τ_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$push(30)$
	(v_2, v_3)	21	(v_2, v_6)	$push(30)$
	(v_2, v_6)	30	(v_2, v_5)	$push(40)$
global FFT	**Out-I**	**In-Label**	**Out-I**	**op**
τ'_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$swap(61)$
	(v_2, v_3)	21	(v_2, v_6)	$swap(71)$
	(v_2, v_6)	61	(v_2, v_5)	$push(40)$
	(v_2, v_6)	71	(v_2, v_5)	$push(40)$

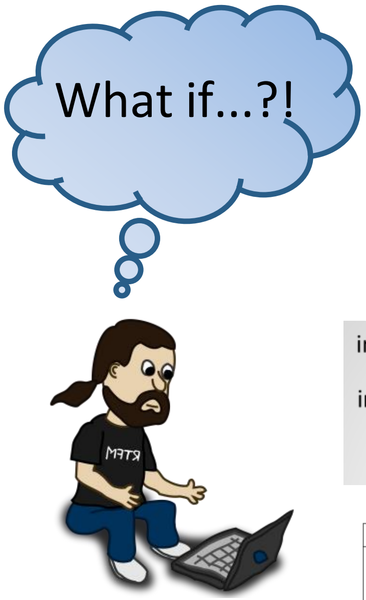
MPLS **configurations**



$$pX \Rightarrow qXX$$
$$pX \Rightarrow qYX$$
$$qY \Rightarrow rYY$$
$$rY \Rightarrow r$$
$$rX \Rightarrow pX$$

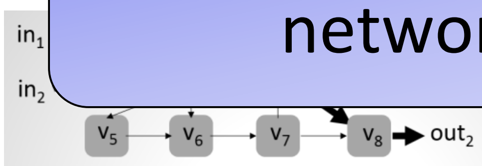
Pushdown Automaton and **Prefix Rewriting Systems** Theory

Can be verified in polynomial time via automata-theoretic approach

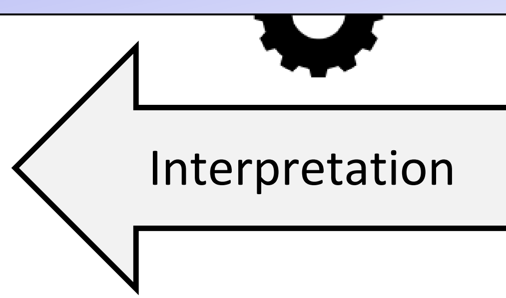


FT	In-I	In-Label	Out-I	op
τ_{v_1}	in_1	$\bot$	(v_1, v_2)	$push(10)$
	in_2	$\bot$	(v_1, v_2)	$push(20)$
τ_{v_2}	(v_1, v_2)	10	(v_2, v_3)	$swap(11)$
	(v_1, v_2)	20	(v_2, v_3)	$swap(21)$
τ_{v_3}	(v_2, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_2, v_3)	21	(v_3, v_8)	$swap(22)$
	(v_7, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_7, v_3)	21	(v_3, v_8)	$swap(22)$
τ_{v_4}	(v_3, v_4)	12	out_1	pop
τ_{v_5}	(v_2, v_5)	40	(v_5, v_6)	pop
τ_{v_6}	(v_2, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	61	(v_6, v_7)	$swap(62)$

Extends to Segment Routing networks (SR-MPLS)!



$$pX \Rightarrow qXX$$
$$pX \Rightarrow qYX$$
$$qY \Rightarrow rYY$$
$$rY \Rightarrow r$$
$$rX \Rightarrow pX$$

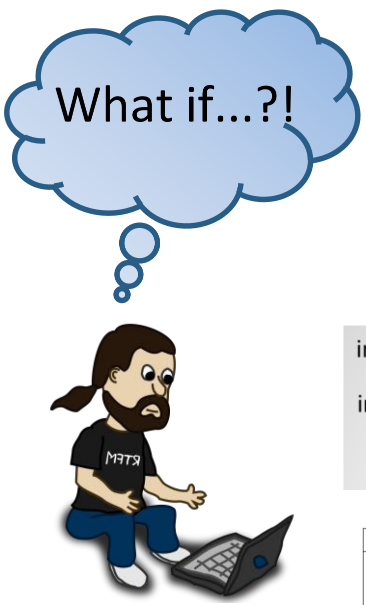


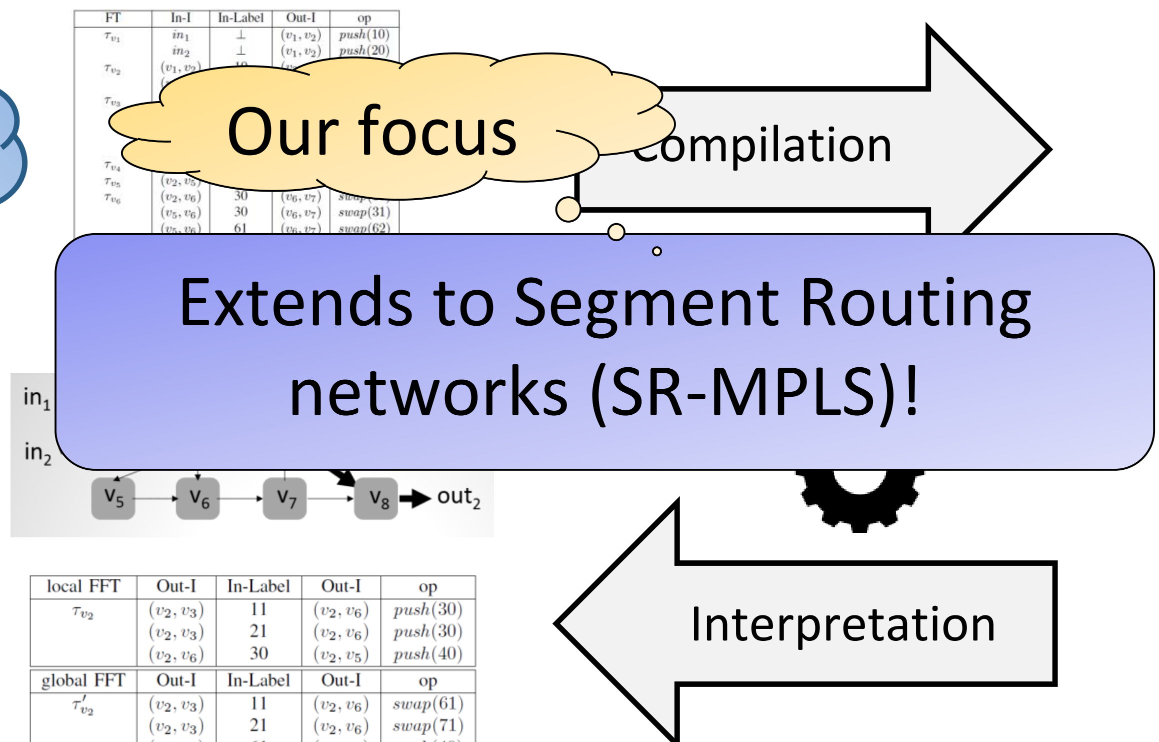
local FFT	Out-I	In-Label	Out-I	op
τ_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$push(30)$
	(v_2, v_3)	21	(v_2, v_6)	$push(30)$
	(v_2, v_6)	30	(v_2, v_5)	$push(40)$
global FFT	**Out-I**	**In-Label**	**Out-I**	**op**
τ'_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$swap(61)$
	(v_2, v_3)	21	(v_2, v_6)	$swap(71)$
	(v_2, v_6)	61	(v_2, v_5)	$push(40)$
	(v_2, v_6)	71	(v_2, v_5)	$push(40)$

MPLS **configurations**

Pushdown Automaton and **Prefix Rewriting Systems** Theory

Can be verified in polynomial time via automata-theoretic approach





local FFT	Out-I	In-Label	Out-I	op
τ_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$push(30)$
	(v_2, v_3)	21	(v_2, v_6)	$push(30)$
	(v_2, v_6)	30	(v_2, v_5)	$push(40)$
global FFT	Out-I	In-Label	Out-I	op
τ'_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$swap(61)$
	(v_2, v_3)	21	(v_2, v_6)	$swap(71)$
	(v_2, v_6)	61	(v_2, v_5)	$push(40)$
	(v_2, v_6)	71	(v_2, v_5)	$push(40)$

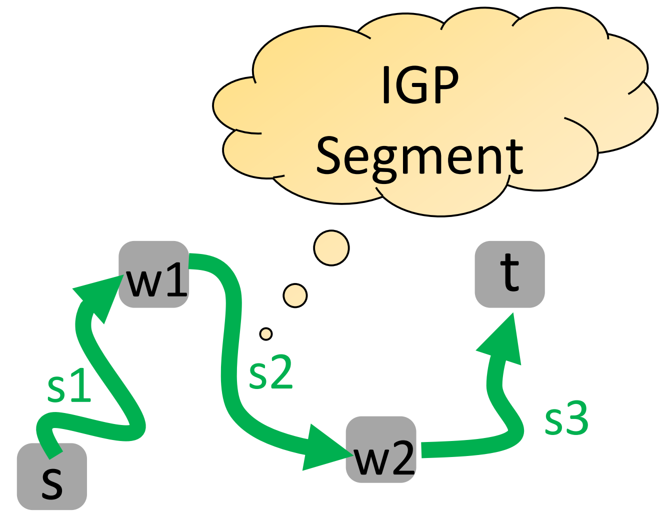
MPLS **configurations**

$$\mathrm{pX} \Rightarrow \mathrm{qXX}$$
$$pX \Rightarrow qYX$$
$$qY \Rightarrow rYY$$
$$rY \Rightarrow r$$
$$rX \Rightarrow pX$$

Pushdown Automaton and **Prefix Rewriting Systems** Theory

Segment Routing Networks

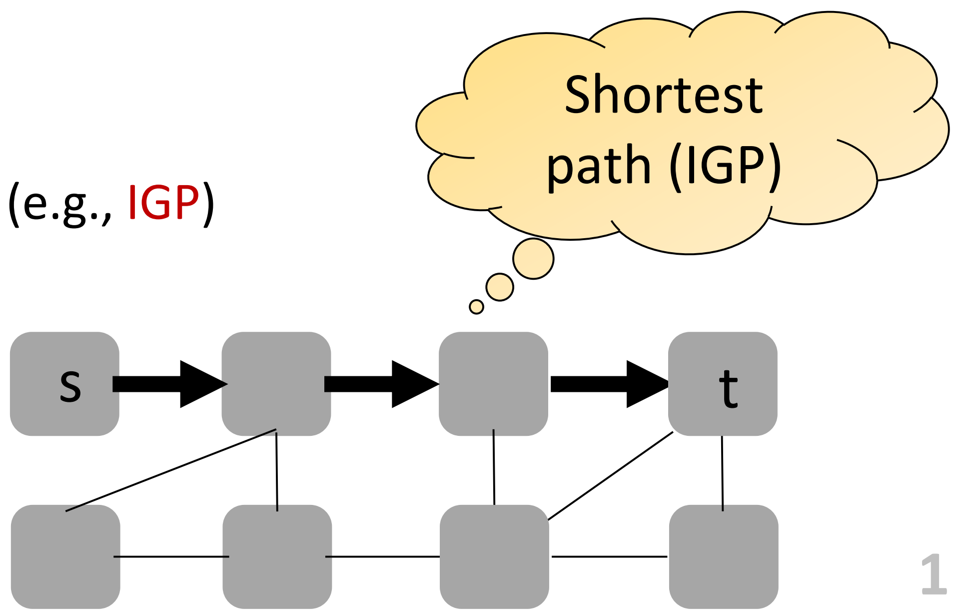
- Attractive: high **path diversity** (compared to, e.g., OSPF), more **scalable** than MPLS (not require state /reservations on all routers), **backward-compatible**, etc.



- Packet can carry in its header, information about a **sequence of segments** it should traverse

- Within segment (i.e., to the next «waypoint»): **shortest path** routing (e.g., IGP)

E.g., by default, single segment shortest path:

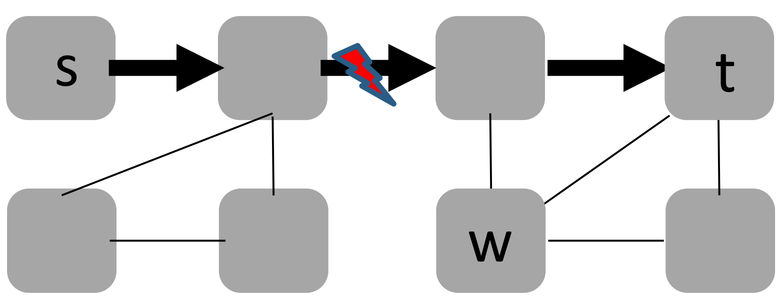


Segment Routing Networks

Upon failure: can **push** an intermediate (remote) destination (waypoint), or an adjacent **link** (force)

- Resp. a **sequence** of segments
- Along segments: **shortest paths** (IGP)

Failover: packet header

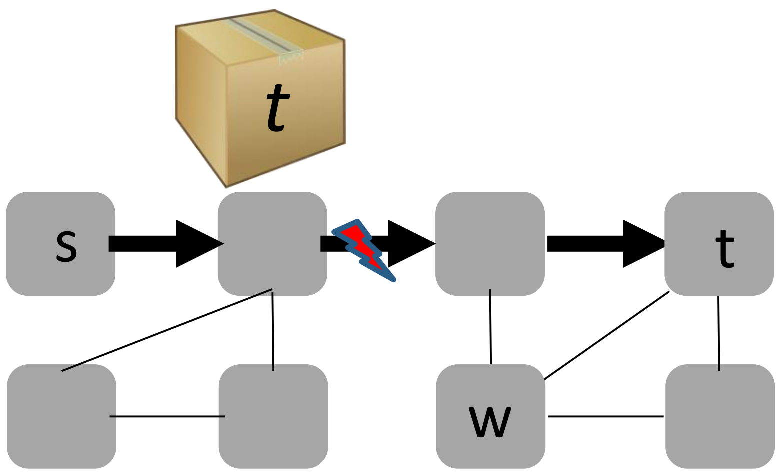


Segment Routing Networks

Upon failure: can **push** an intermediate (remote) destination (waypoint), or an adjacent **link** (force)

- Resp. a **sequence** of segments
- Along segments: **shortest paths** (IGP)

Failover: packet header

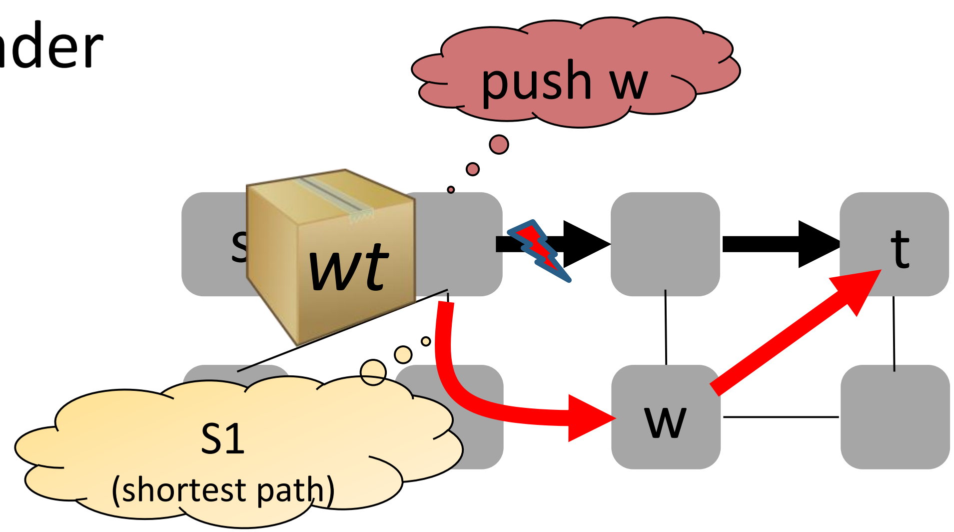


Segment Routing Networks

Upon failure: can **push** an intermediate (remote) destination (waypoint), or an adjacent **link** (force)

- Resp. a **sequence** of segments
- Along segments: **shortest paths** (IGP)

Failover: packet header

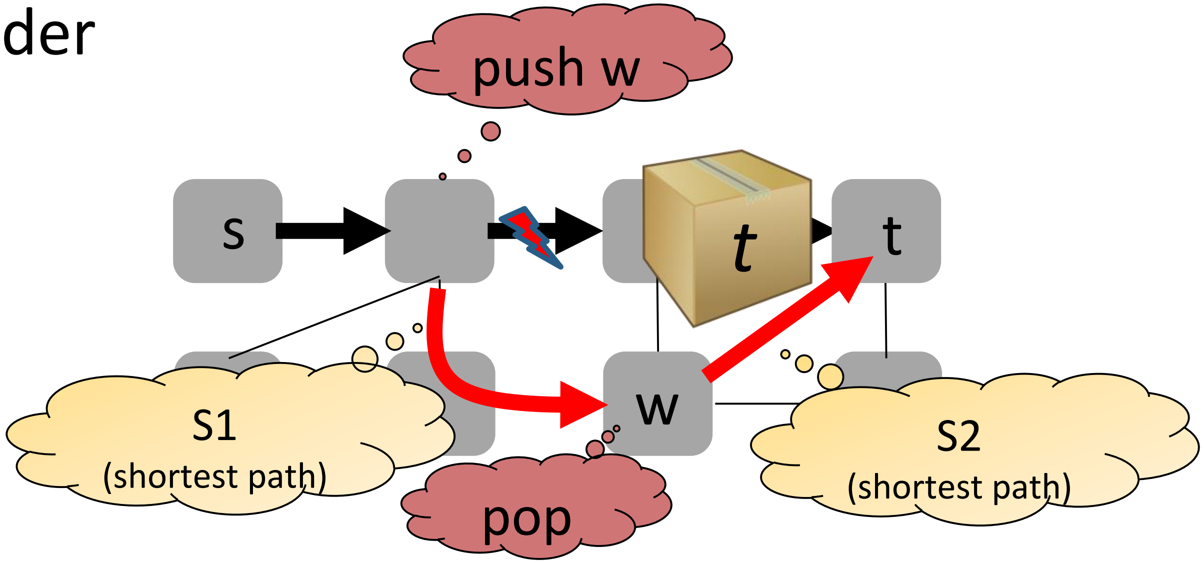


Segment Routing Networks

Upon failure: can **push** an intermediate (remote) destination (waypoint), or an adjacent **link** (force)

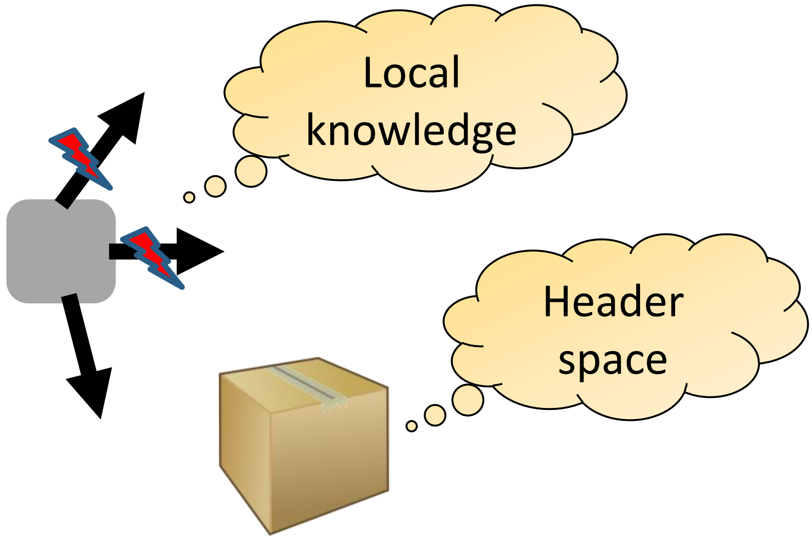
- Resp. a **sequence** of segments
- Along segments: **shortest paths** (IGP)

Failover: packet header



Challenges (1)

- Combination of «**stack-based** forwarding» and **shortest path** (IGP) routing
- Failover path should never use failed links *again*
- **Local** knowledge only
- Limited **header space**
- **Multiple failures**

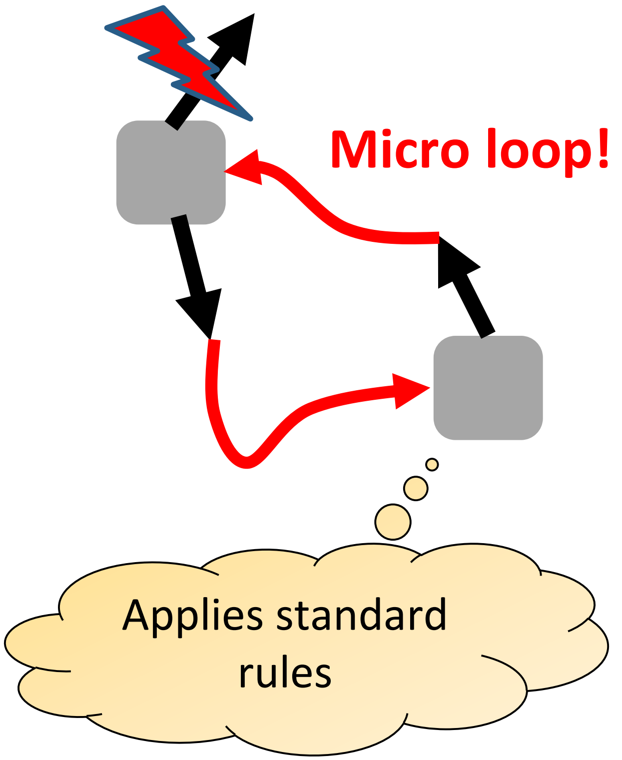


Failover Rules:

f(status incident links, header) → push waypoint(s)

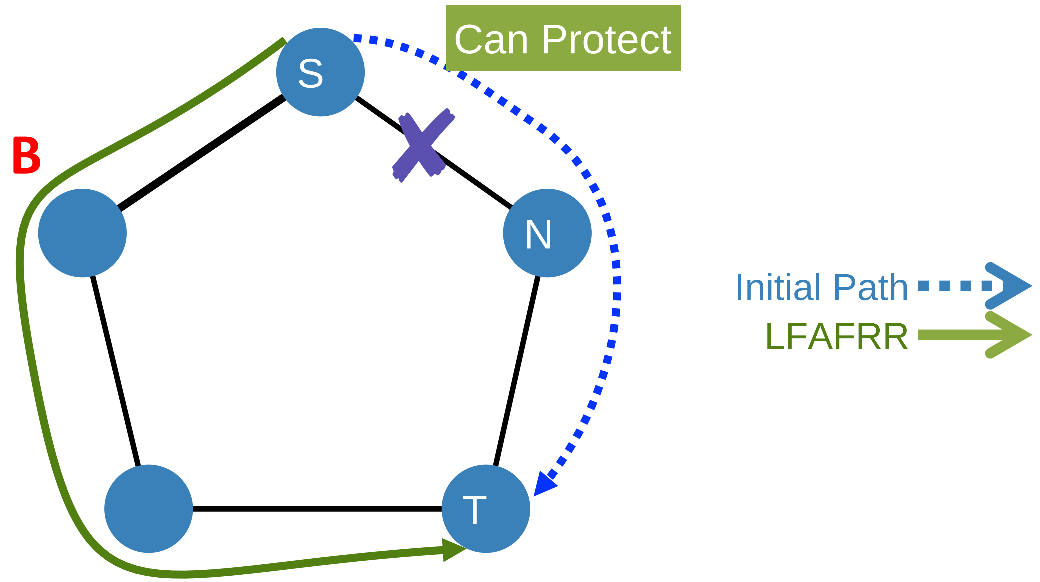
Challenges (2)

Without header info: does not know that packet failed over, applies standard rules, i.e., default shortest path to destination: may **loop**



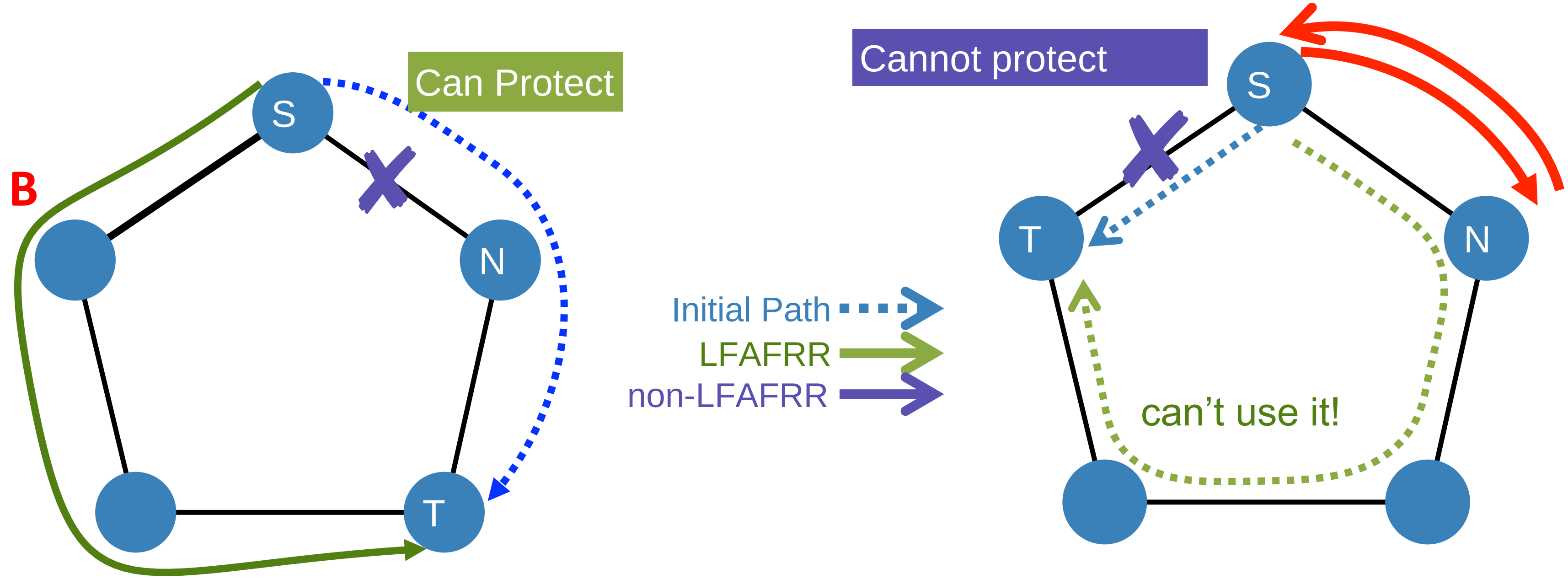
FRR has to ensure loop-freedom!

Solution: Loop-Free Alternative (LFA)?



- If (S,N) **fails**, S can failover to B
- X has shortest path to T that does not go through (S,N) **again**
- **WORKS: can protect (S,N)**

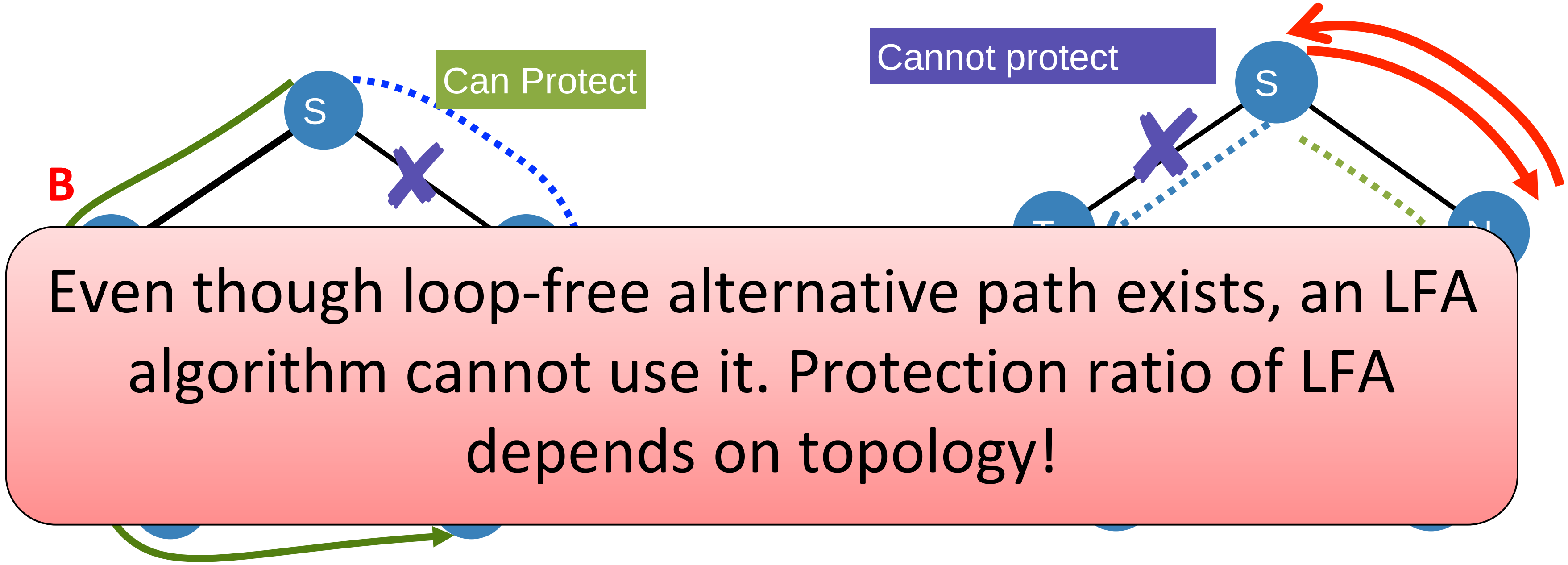
Solution: Loop-Free Alternative (LFA)?



- If (S,N) **fails**, S can failover to B
- X has shortest path to T that does not go through (S,N) **again**
- **WORKS: can protect (S,N)**

- If (S,T) **fails**, S can only try to failover to N
- However, when N's shortest route to T goes along S again: **loop**
- **DOES NOT:** Cannot protect (S,T)

Solution: Loop-Free Alternative (LFA)?



Even though loop-free alternative path exists, an LFA algorithm cannot use it. Protection ratio of LFA depends on topology!

- If (S,N) **fails**, S can failover to B
- X has shortest path to T that does not go through (S,N) **again**
- **WORKS: can protect (S,N)**

- If (S,T) **fails**, S can only try to failover to N
- However, when N's shortest route to T goes along S again: **loop**
- **DOES NOT:** Cannot protect (S,T)

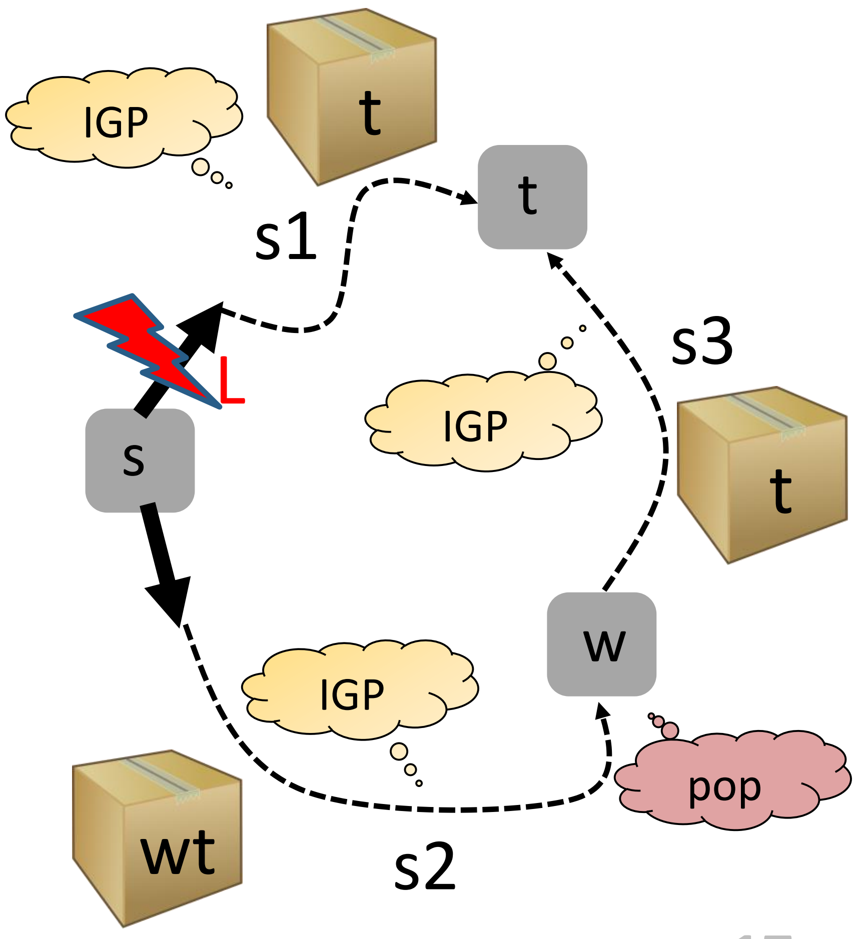
Even though alternative paths exist, I cannot use it. Protection ratio of LFA depends on topology…

Can we fix it with Segment Routing?

Topology-Independent LFA (TI-LFA)

- Yes we can! Idea: **push a segment**, i.e., certain waypoint w

- It must be ensured: second (IGP) segment w → t does **not go** via L again!

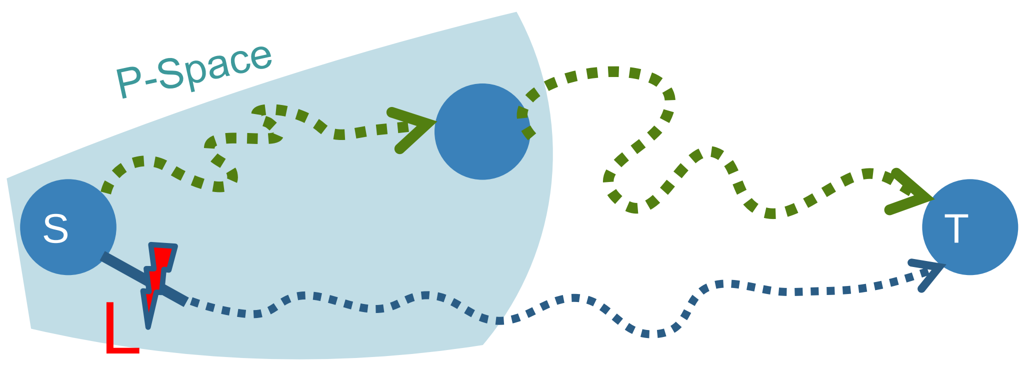


How to find such a w? Is it always possible? I.e., Topology-Independent?

TI-LFA

- Yes it is always possible but we need a **twist**

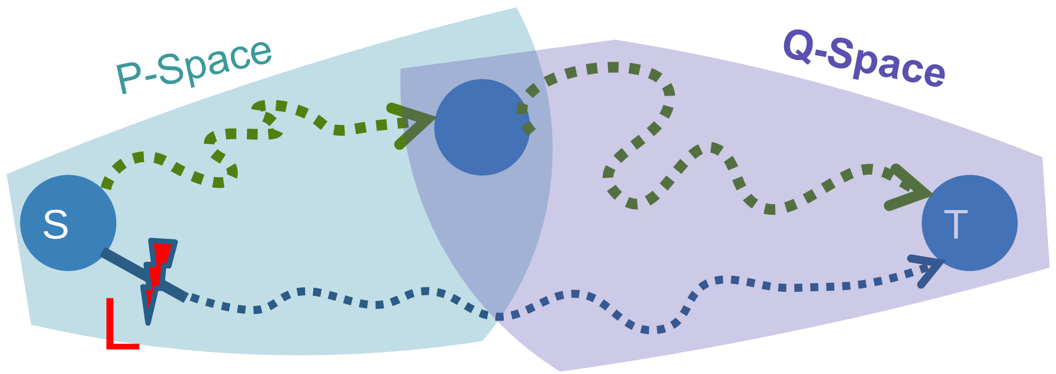
- We need two definitions:
  - **P-Space**: the nodes whose shortest path from S does not use L



TI-LFA

- Yes it is always possible but we need a **twist**

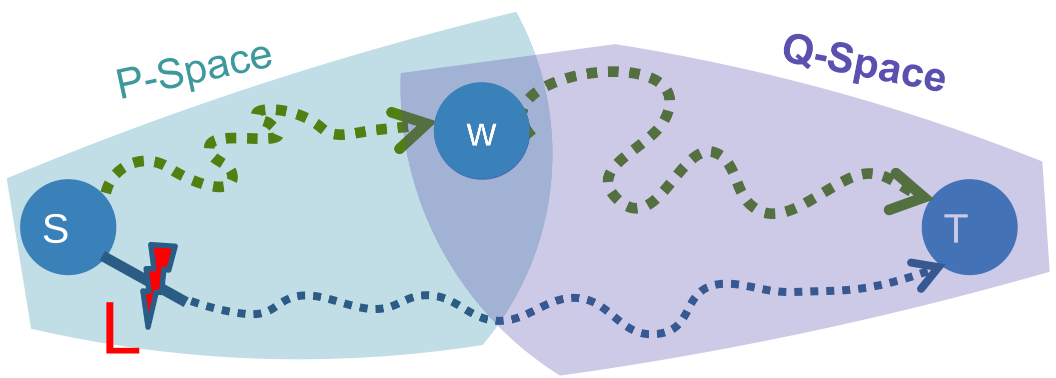
- We need two definitions:
  - **P-Space**: the nodes whose shortest path from S does not use L
  - **Q-Space**: the nodes whose shortest path to T does not use L



TI-LFA

- Yes it is always possible but we need a **twist**

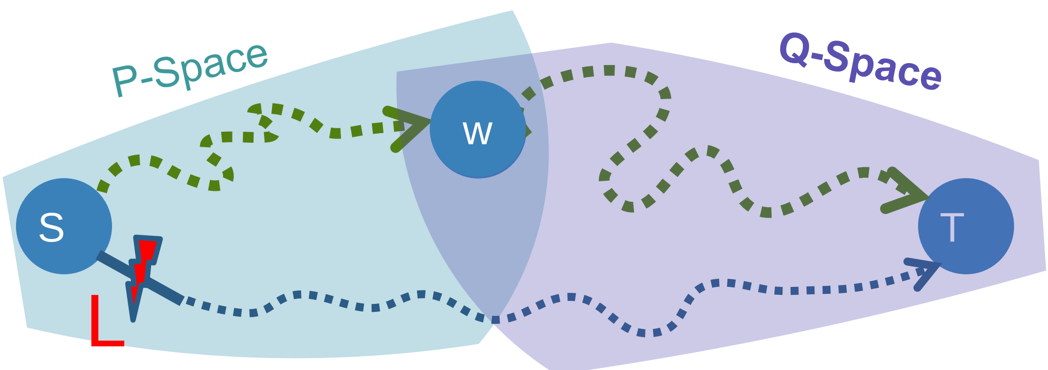
- We need two definitions:
  - **P-Space**: the nodes whose shortest path from S does not use L
  - **Q-Space**: the nodes whose shortest path to T does not use L



- Idea: choose segment endpoint w at intersection!
  - There are IGP routes from s to w and w to t without failures

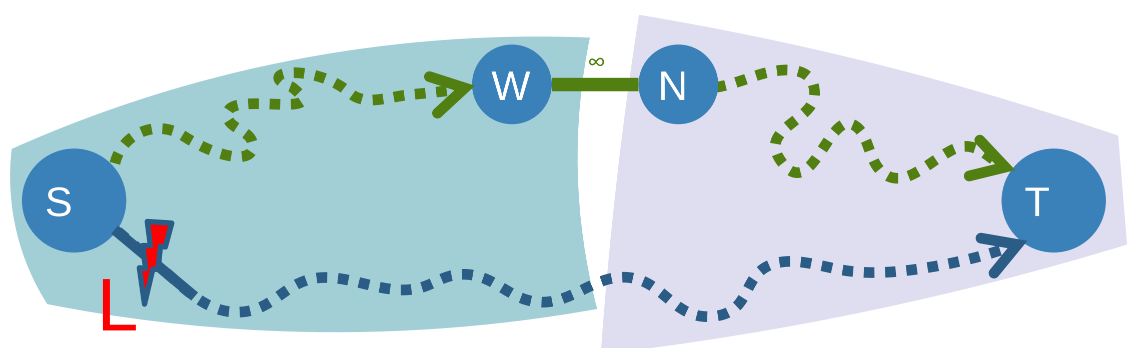
TI-LFA: Properties

P-Space and Q-Space: Are **connected** subgraphs, **cover** all nodes, overlap or are adjacent

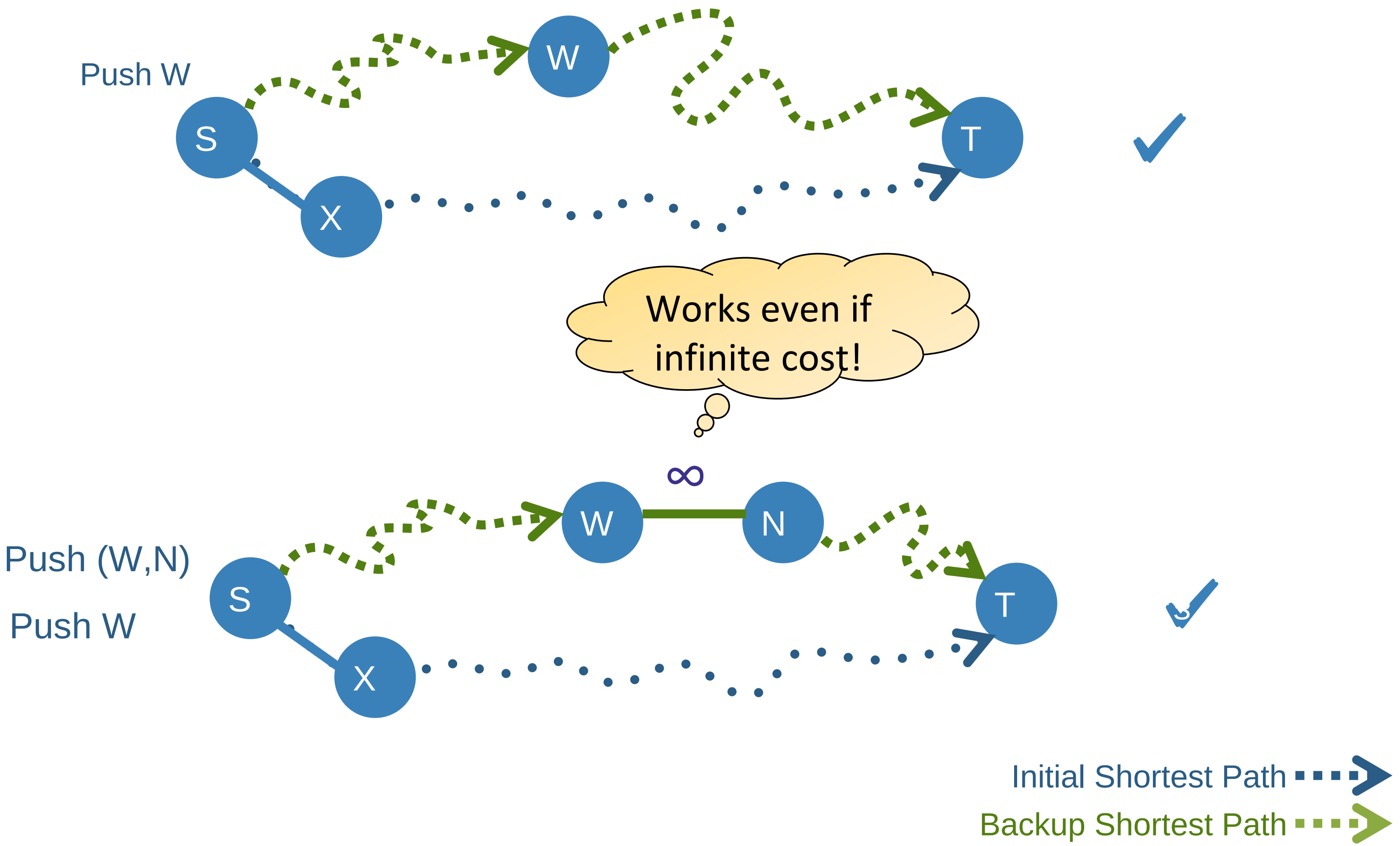


Case 1: S can simply push W

Case 2: S pushes W and (W,N), forces packet to enter Q-space



TI-LFA Summary

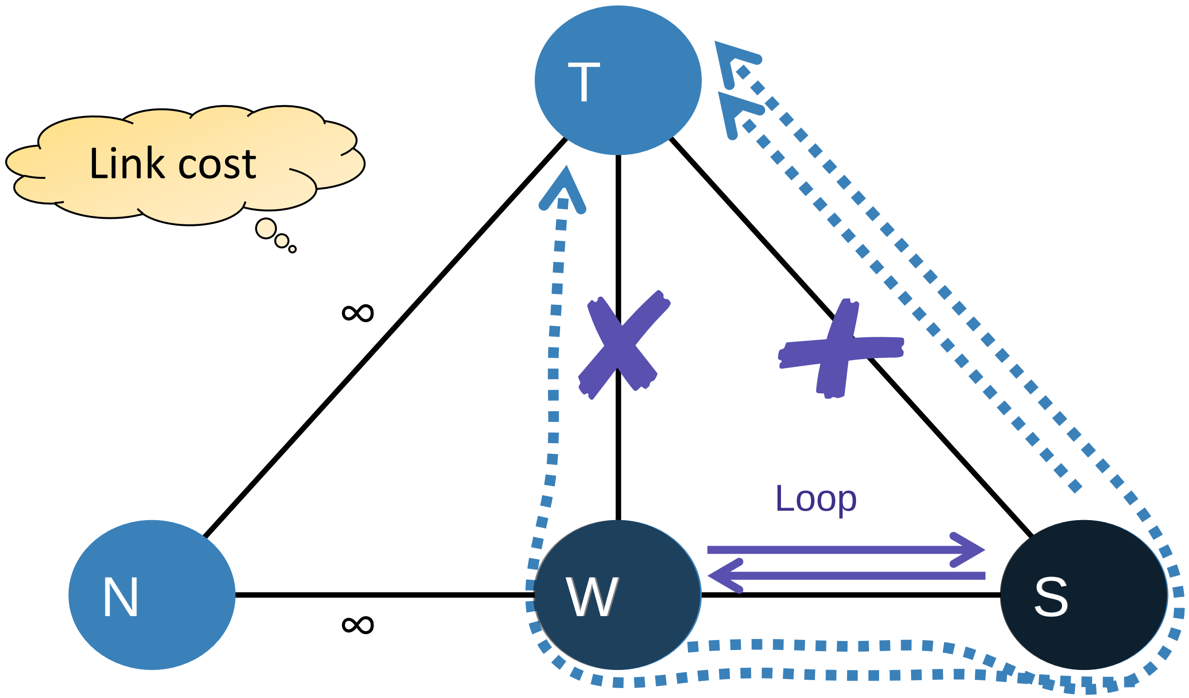


TI-LFA is provably robust to 1 failure!

What about 2 or more failures?

Not really…

TI-LFA Under Double Failure (✗✗)



Problem:

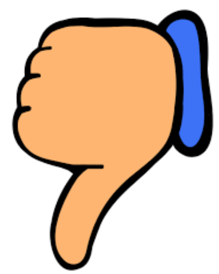
- If S pushes W to reroute…
- … but W also has a link failure and pushes S (only knows local failures)…
- … *we have a loop again!*

No longer TI!

A First Idea: Emulate FRR Based on Arborescences (Chiesa et al./Foerster et al.)

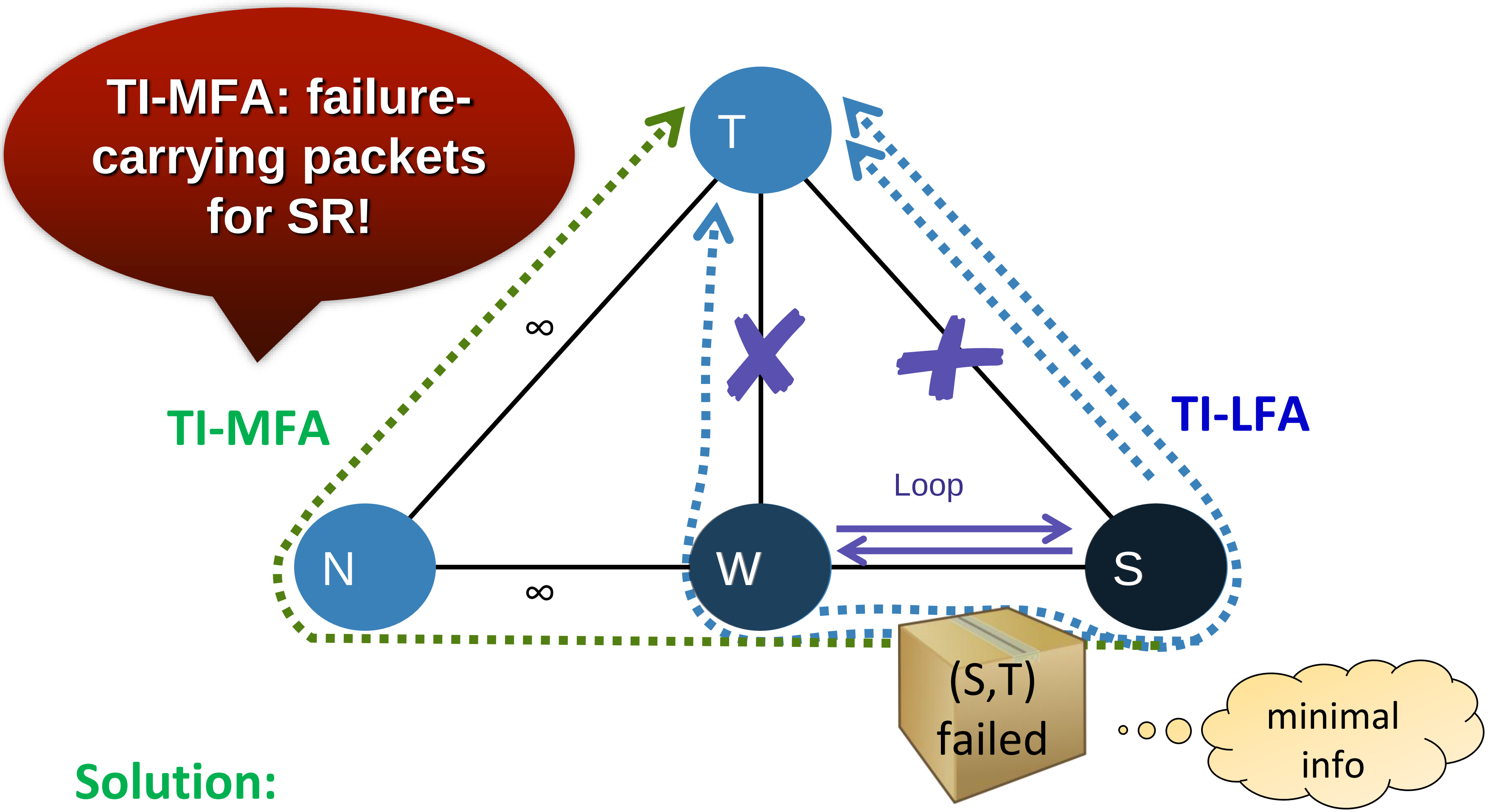
In principle, one can **emulate** FRR based on arborescences (Chiesa et al., Foerster et al.):

- high resiliency



- Need inport matching
- Need to force one link, hop-by-hop: many (forcing) rules!
- Goes against idea of SR
- Paths can be long

TI-LFA Under Double Failure (✗✗)



Solution:

- The **packet could tell** *W* about the failure of ST: W in this case **sees** and **pushes N**
- Rerouting **through 3 segments** would avoid both failures: *SW, WN, NT*

TI-MFA: Topology-Independent Multi-Failure Alternate

From the viewpoint of the node *S* where the packet hits another failed link:

1. **Flush** the label stack except for the destination *T*

2. Based on all **link failure info** stored in the packet header, compute the segments necessary to reach *T* and the labels accordingly

3. Find the last node on *ShortestPath(S,T)* that a packet can reach from *S* without hitting known failed link (**"repeated TI-LFA on subgraph"**)

   a. Let ***V1*** be this node followed by the link ***(V1,V2)*** on this path

   b. Set the top of **label stack** as *(V1, (V1,V2),…*

   c. Repeat the same for *V2* as the start of next segment and keep repeating until the segment that ends with *T*

4. **Dispatch** the packet (it will reach *T* unless it hits a failure disconnecting the network)

TI-MFA: Topology-Independent Multi-Failure Alternate

From the viewpoint of the node *S* where the packet hits another failed link:

1. **Flush** the label stack except for the destination *T*

2. Based on all failur stored in the pac ry to reach *T*
   and the labels acc

3. Find the l n
   failed link

   a. Let ***V1*** be this node followed by the link this path

   b. Set the top of **label stack** as *(V1, (V1,V2),…*

   c. Repeat the same for *V2* as the start of next segment and keep repeating until the segment that ends with *T*

4. Dispatch the packet (it will reach *T* unless it hits a failure disconnecting the network)

We also consider a variant ***without flushing***: we force to strictly route around each failed link, before continuing toward destination. Can also extend TI-LFA like this…

TI-MFA Under Many Failures (xxxxxx)

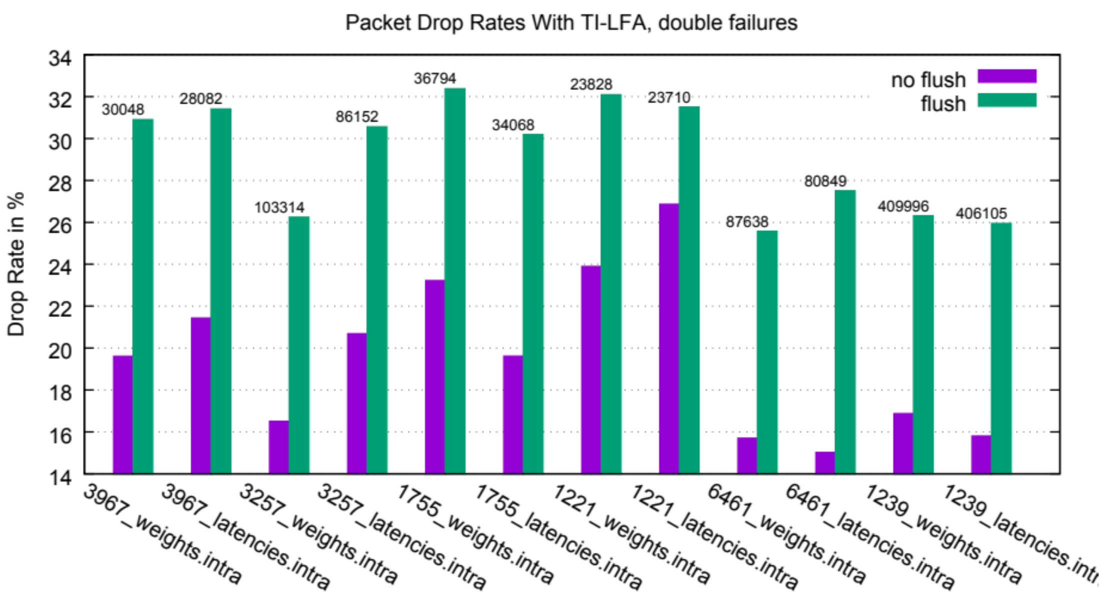
Theorem: TI-MFA tolerates k failures in k-connected network!

Proof:

- **Invariant:** by construction, previously hit failures won’t be hit again
- *k* failures: by construction the backup path will not use any failed link **seen previously**
- Hence, the packet either hits all the *k* failures or reaches its destination early

Experimental Results

- Simulations on **Rocketfuel** topologies, over 5 million scenarios
- Recorded **connectivity**, maximum **header sizes**, and path **lengths**



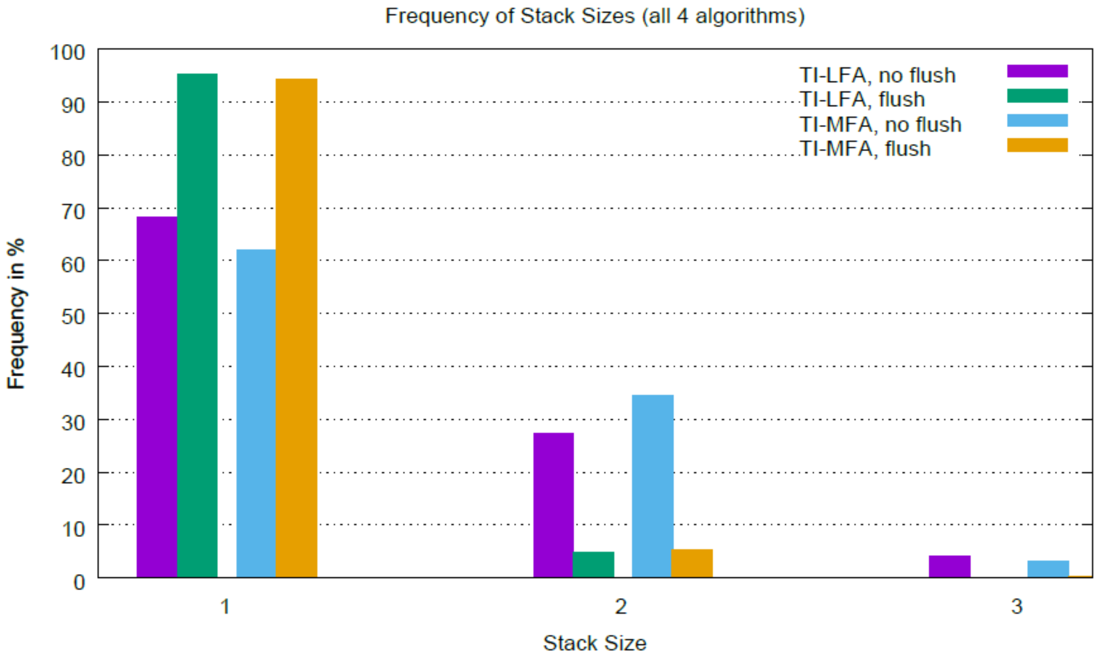
TI-LFA fails to deal with 2 failures in many cases (and not only in the worst case).

Surprisingly, TI-LFA cannot benefit from flushing!

Experimental Results

- Simulations on **Rocketfuel** topologies, over 5 million scenarios
- Recorded **connectivity**, maximum **header sizes**, and path **lengths**

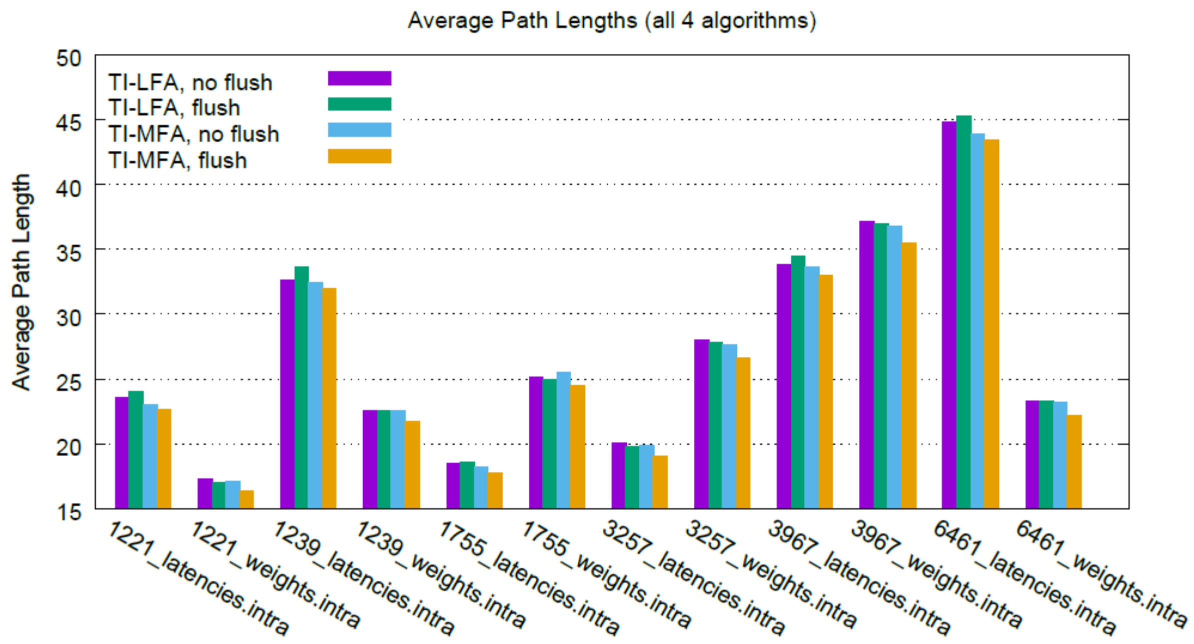
Stacks are usually small (especially with flush of course)



Experimental Results

- Simulations on **Rocketfuel** topologies, over 5 million scenarios
- Recorded **connectivity**, maximum **header sizes**, and path **lengths**

Path lengths of the algorithms are comparable (TI-MFA, especially with flush shorter, as expected)



More Results in the Paper

Theorem: There is a fundamental tradeoff efficiency vs robustness of failover (if packets cannot carry failures). Any failover scheme for SR which tolerates at least two failures, can be forced to use very costly routes even in the presence of a single failure.

Summary

- **Fast rerouting** important but not well-understood
- Interesting algorithmic problem, many open questions
- First look at **segment routing**
  - Limitations of TI-LFA
  - Robust to many failures with MI-LFA
- Future work: yes ☺

Further Reading

- Local Fast Failover Routing With Low Stretch
  Klaus-Tycho Foerster, Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Tredan.
  ACM SIGCOMM Computer Communication Review (**CCR**), 2018.
- Load-Optimal Local Fast Rerouting for Dependable Networks
  Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Tredan.
  47th IEEE/IFIP International Conference on Dependable Systems and Networks (**DSN**), Denver, Colorado, USA, June 2017.
- Polynomial-Time What-If Analysis for Prefix-Manipulating MPLS Networks
  Stefan Schmid and Jiri Srba.
  37th IEEE Conference on Computer Communications (**INFOCOM**), Honolulu, Hawaii, USA, April 2018.

Thank you! Questions?