Local Fast Segment Rerouting on Hypercubes

Klaus-Tycho Foerster
University of Vienna, Austria

Mahmoud Parham
University of Vienna, Austria

Stefan Schmid
University of Vienna, Austria

Tao Wen
UESTC, China

Fast Rerouting (FRR)

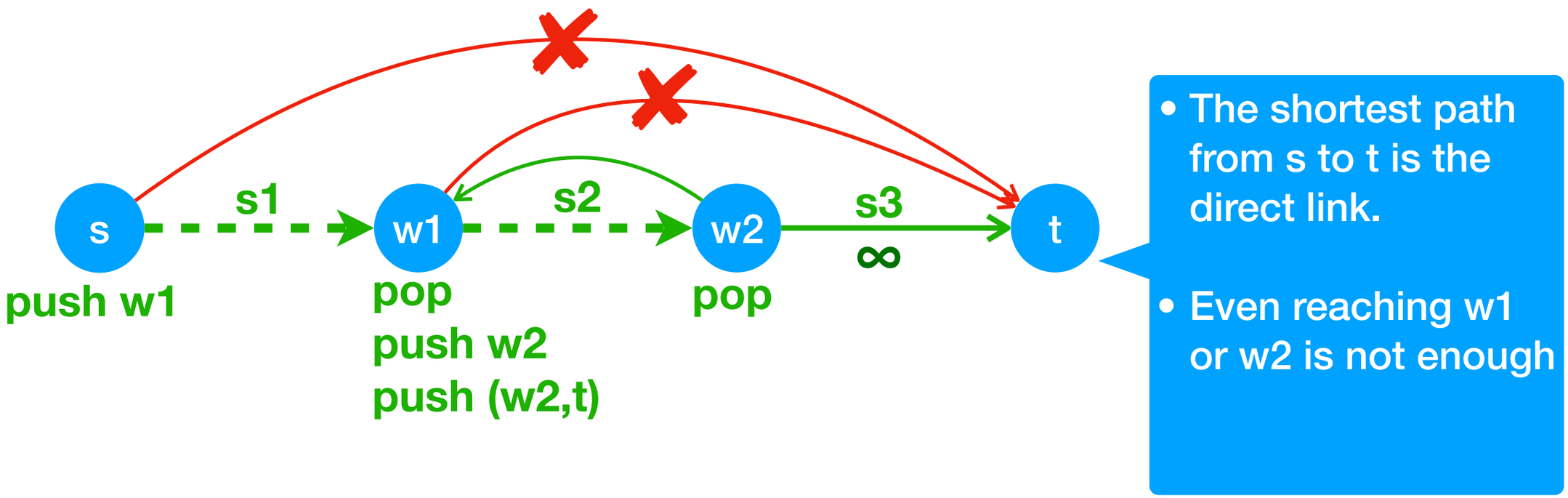
- Realtime traffic requires sub-50ms restoration
- Local failover without invoking control plane, **no reconvergence**
- Pre-computed **backup paths** instead of on-the-fly computing
- Carrying failures is not desired (i.e. exponential number of rules)
- Desired: little computation overhead on routers, reasonable data overhead on packets

Segment Routing

- SR as a tool for enforcing a specific path or a bundle of paths
- A path can be divided into **segments**
- Specified by intermediate destinations: **waypoint**, **tunnel link**
- Within segments **shortest path** routing is performed (e.g., IGP)

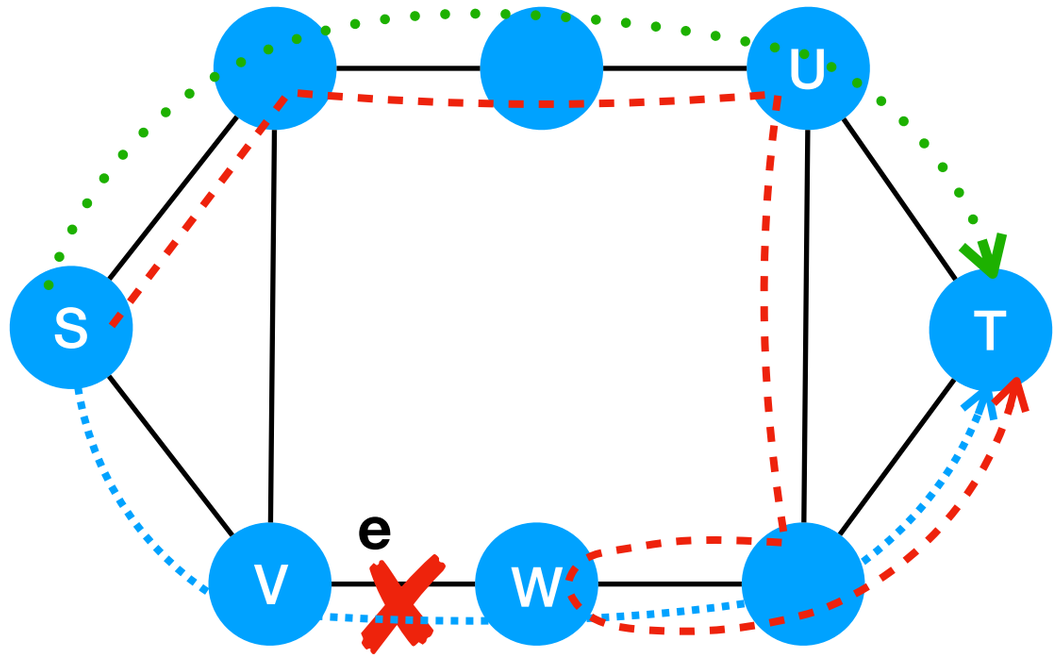
Segment Routing (1)

- A segment is specified by a **label**, be it waypoint or tunnel link
- Packet can carry information about segments it should traverse, in its header (label **stack**)
- Nodes can **push** labels, e.g., for rerouting upon failure



Single Failure (TI-LFA)

push U
push W
push T

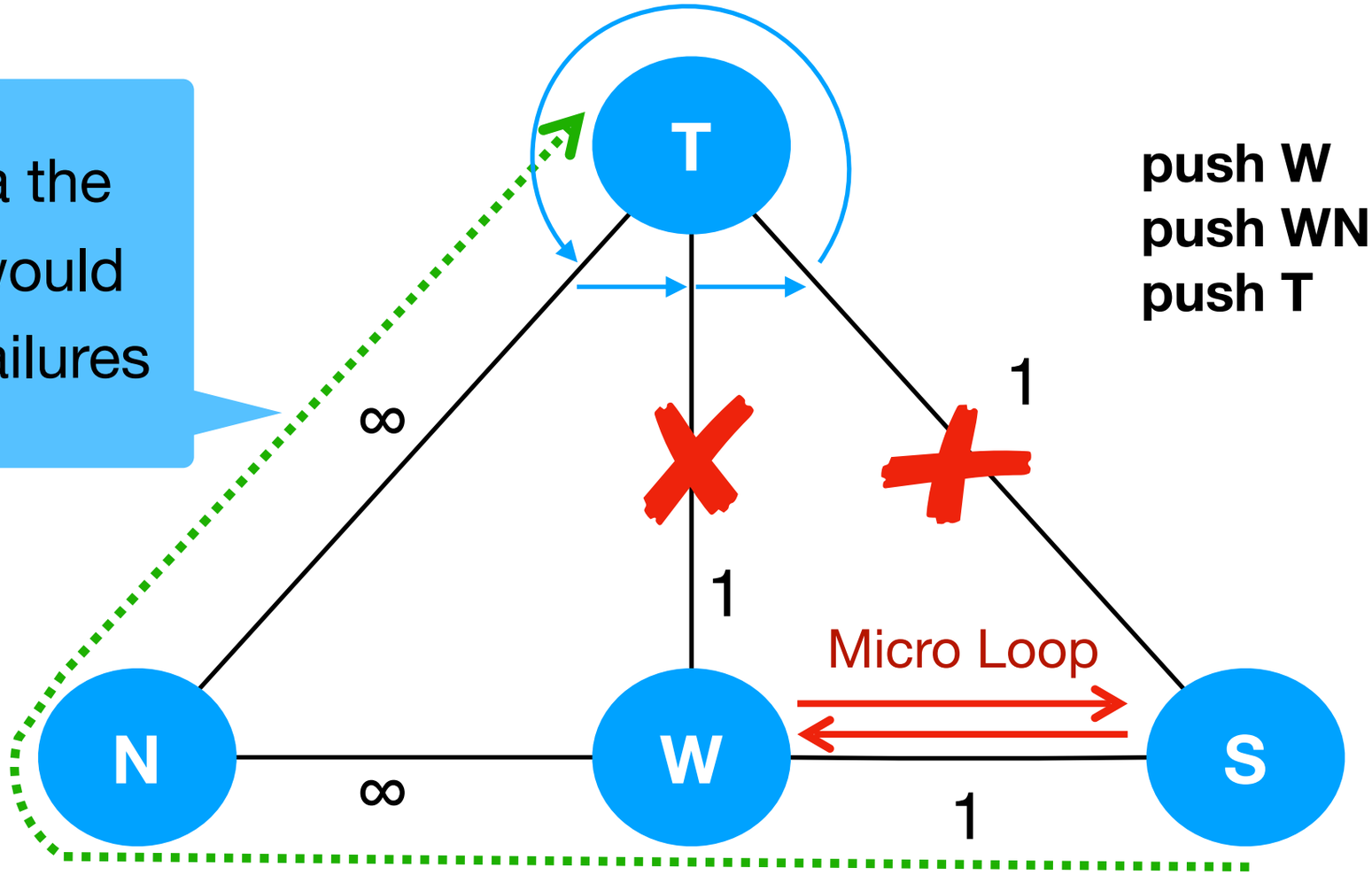


Circumventing the failed link using 3 segments, 2 additional labels

Double Failures

Dependency
Link
Path

Rerouting via the green path would avoid both failures

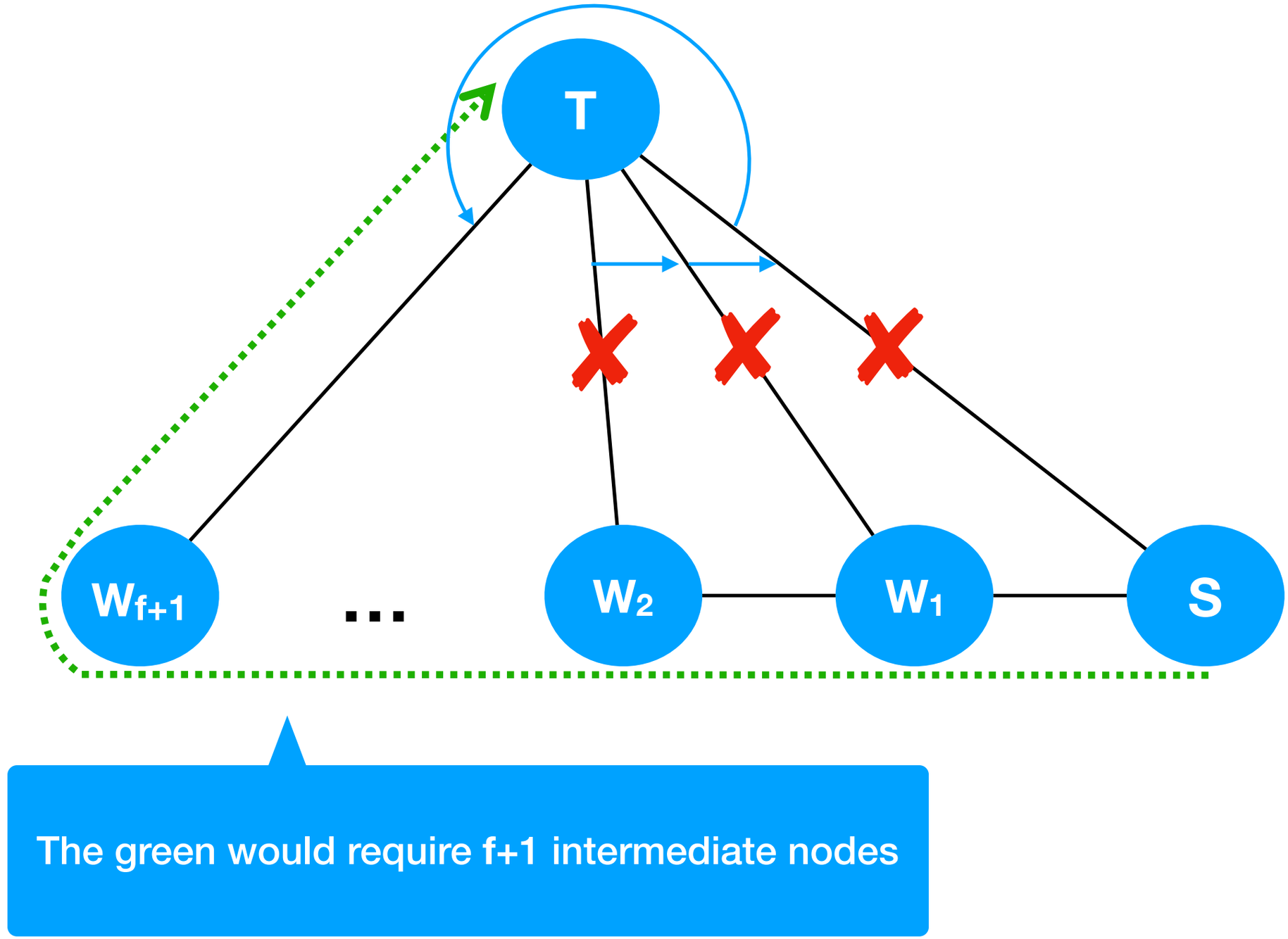


push W
push WN
push T

Tunnel link

S reroutes to W, W reroutes back to S 👉 Loop

f Failures



The green would require f+1 intermediate nodes

Challenges

- Fixed choice of route under arbitrary sets of failures
- Based on **local** failures => unknown past/future failures
- Maximum resiliency => reroute whenever possible
- Number of segments per route must be limited
- Preserving visits to waypoints

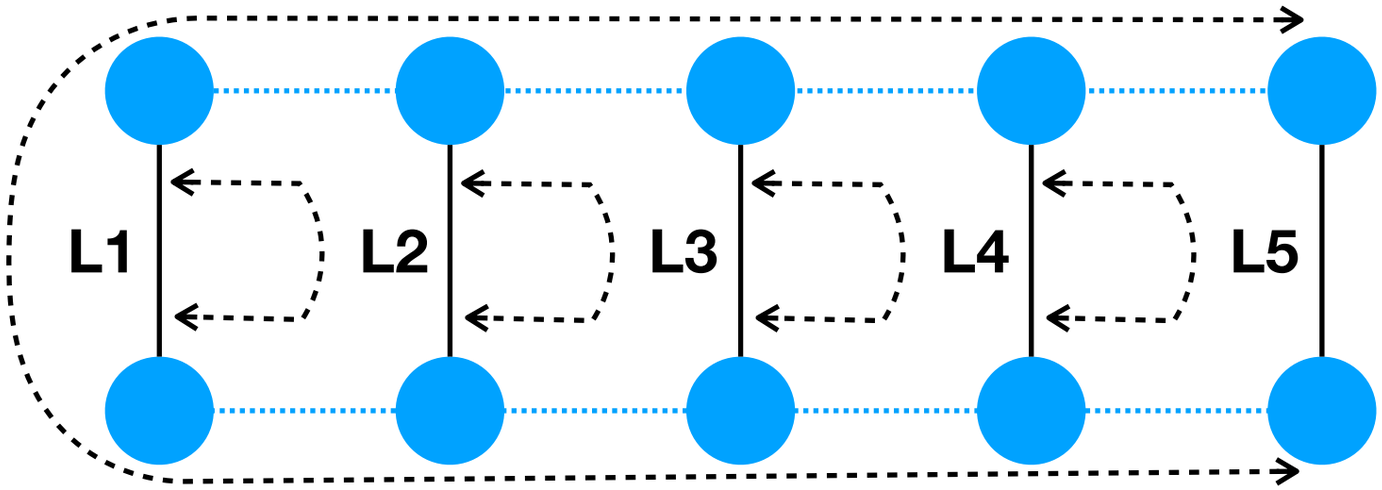
Model

- A single backup path for each link: **backup path scheme**
- On hitting a failed link, its BP activates to bypass the link
- No knowledge of up/down stream failures, only the incident one
- Guaranteed delivery via pre-computed BPs

Preliminaries

- Link L1 **depends** on L2 iff L2 lies on the BP of L1
- Denoted by dependency arc: L1 → L2
- A BP scheme induces lots of dependencies
- Cycles of such dependencies determine resiliency
- E.g., if L1 and L2 are mutually dependent we have a cycle of length two => 1-resilient
- A scheme is **f-resilient** iff the shortest cycle has at least **f+1** arcs

Example



L1 → L2 → L3 → L4 → L5 → L1

A cycle of dependencies over 5 links => 4-resilient

The Problem

Given a (f+1)-connected graph,

find a backup path allocation that is

1. *f*-resilient

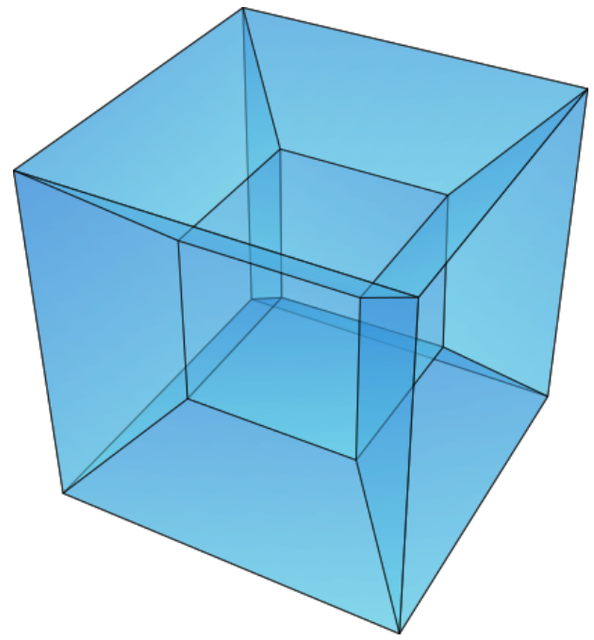
2. can be enforced using few segments

(f+1)-edge-connectivity is a necessary condition.
Also sufficient? We don’t know!

Similar Works

- Based on arc-disjoint spanning trees, spanning **arborescences**
- Rooted at a fixed destination **d**
- **f+1** arborescences in a cyclic order
- On failure, reroute to the next arborescence 👉 **f-resilient**
- The pre-computation repeats for every destination d
- Does not guarantee waypoint visits
- Our approach: independent of destinations, one entry per link

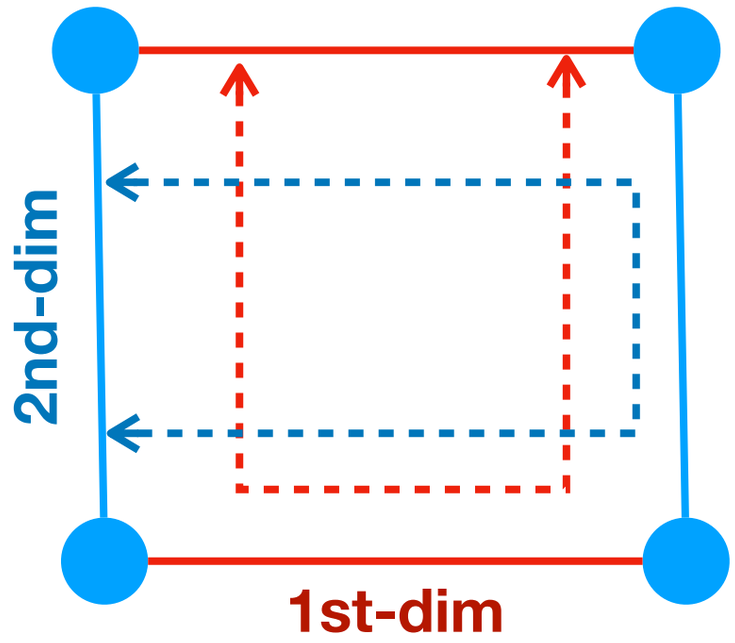
Hypercube



- Easy, well structured
- A step-stone towards more general graphs
- The 1-cube is 0-resilient trivially
- The *k* dimensional HC is (*k*-1)-resilient

2-cube

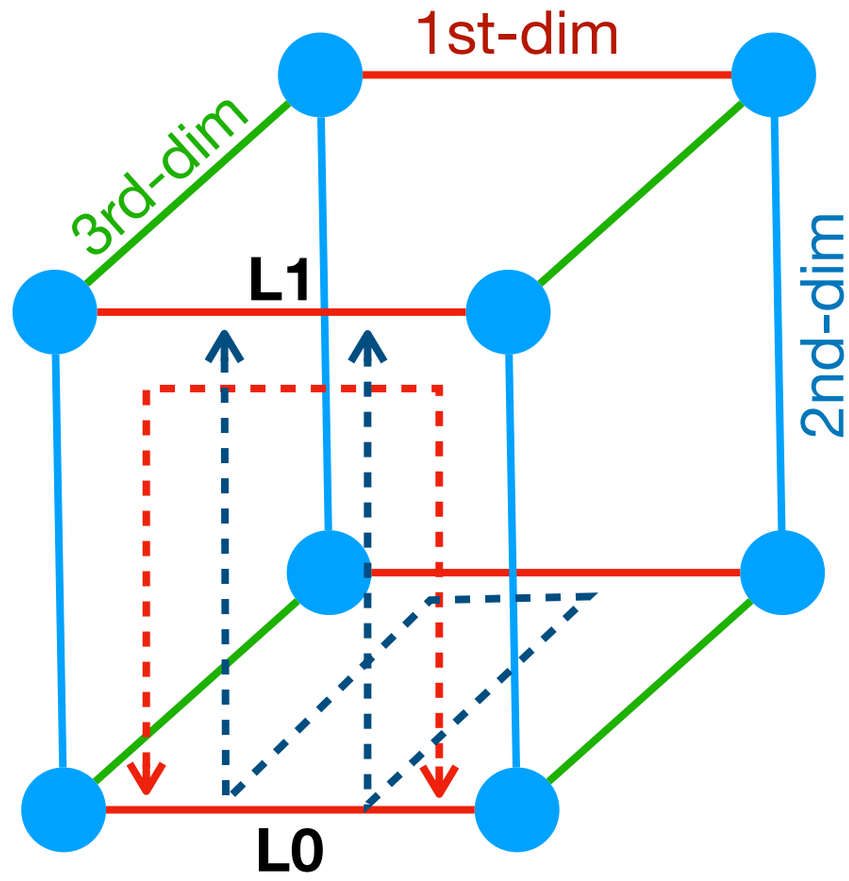
- 1st dimension uses 2nd dimension
- 2nd dimension uses 1st dimension
- 1-resilient scheme



1-resilient

k-cube

- BP of a d-dim link traverses higher dimensions sequentially
- d-dim links are indexed: i<j ⇔ (L_i is closer to the origin)
- Process d-dim links in the ascending order of i
- BP of the *i*th d-dim link includes the (i+1)th d-dim link
- Next dimension links in **pairs**, when detour is necessary
- A 1st-dim link uses 2nd-dim, then 3rd-dim and so on
- *d*-dim link uses (*d+1*)-dim up to *(d+⌈log (k-1)⌉)*-dim

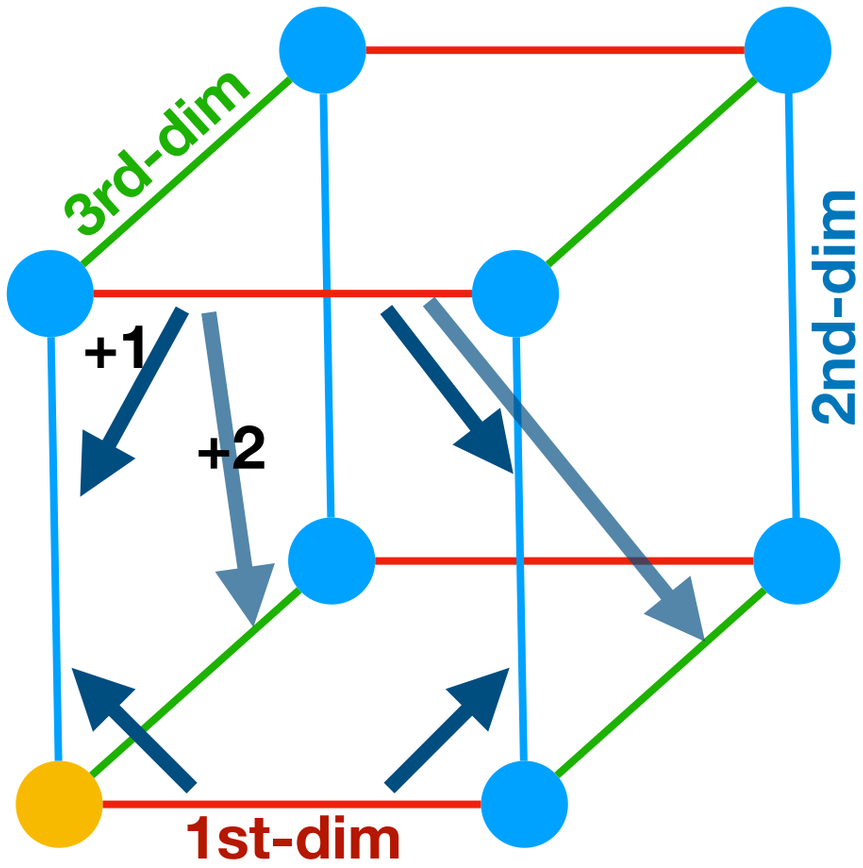


2-resilient,
≤1 detour

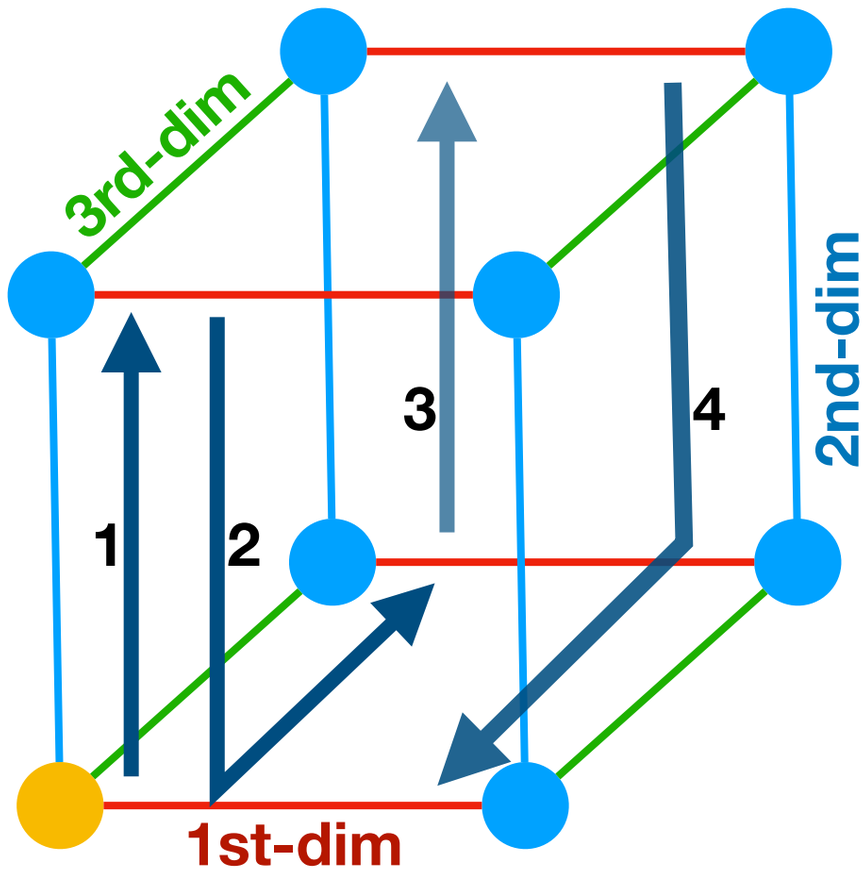
Path length increases with each detour, up to 3+2⌈*log k*⌉

Analysis

- A cycle of dependencies (**CoD**) must not be shorter than *k*
- CoDs over same-dimension links are long by construction
- Cross-dim CoDs do not always traverse dimensions sequentially: e.g., 1st-dim→3rd-dim→1st-dim (i.e., +2 then +1)



cross-dimension dependencies



same-dimension dependencies

Analysis (1)

- A cycle of dependencies (CoD) must not be shorter than *k*
- CoDs over same-dimension links are long, by construction
- Cross CoD traverses all dimensions an even number of times, ≥ 2k traversals (1)
- Direction away from the origin: **uphill**, otherwise **downhill**
- Equal number of uphill and downhill traversals in any CoD (2)
- A dependency can take many downhills, but at most one uphill (3)
- (1,2,3)⇒ at least **k** uphills in a CoD⇒ at least k dependency arcs in the CoD
- ⇒ the scheme is (k-1)-resilient

General Solver

$$\textbf{Maximize} \sum_{\ell \in E} \mathcal{I}_\ell \tag{2}$$

number of protected links

$$\mathcal{SP}_\ell^z = \begin{cases} 1 & \ell \in SP(u,z) \\ 0 & else \end{cases} \qquad \forall \ell = (u,v) \in E, z \in V \tag{3}$$

$$\mathcal{D}_{\ell\ell'}, \mathcal{X}^v_{\ell\ell'}, \mathcal{T}_{\ell\ell'}, \mathcal{W}^v_\ell, \mathcal{I}_\ell, \in \{0,1\} \qquad \forall \ell, \ell' \in E, \forall v \in V \tag{4}$$

$$\mathcal{D}_{\ell\ell} = 0 \qquad \forall \ell \in E \tag{5}$$

BP maker

$$\sum_{\ell_2=(v,*)} \mathcal{D}_{\ell_1\ell_2} - \sum_{\ell_2=(*,v)} \mathcal{D}_{\ell_1\ell_2} = \begin{cases} \mathcal{I}_{\ell_1} & v = s \\ -\mathcal{I}_{\ell_1} & v = t \\ 0 & \text{else} \end{cases} \qquad \forall \ell_1 = (s,t) \in E, v \in V \tag{6}$$

segment assignment

$$\mathcal{X}^v_{\ell_1\ell_2} \le \mathcal{SP}^v_{\ell_2}, \sum_{\ell \in E, \ell \ni v} \mathcal{D}_{\ell_1\ell} \qquad \forall \ell_1, \ell_2 \in E, v \in V \tag{7}$$

$$\mathcal{D}_{\ell_1\ell_2} \le \mathcal{SP}^t_{\ell_2} + \mathcal{T}_{\ell_1\ell_2} + \sum_{v \in V} \mathcal{X}^v_{\ell_1\ell_2} \qquad \forall \ell_1 = (s,t), \ell_2 \in E \tag{8}$$

dependency cycle check

$$d_{\ell_1\ell_2} \ge 0, d_{\ell_1\ell_1} = 0 \qquad \forall \ell_1, \ell_2 \in E \tag{9}$$

$$d_{\ell_1\ell_3} \le d_{\ell_1\ell_2} + 1 + (1 - \mathcal{D}_{\ell_2\ell_3}) \times \infty \qquad \forall \ell_1, \ell_2, \ell_3 \in E \tag{10}$$

$$d_{\ell_1\ell_2} + d_{\ell_2\ell_1} \ge f + 1 \qquad \forall \ell_1, \ell_2 \in E \tag{11}$$

$$\mathcal{W}^v_{\ell_1} \ge \mathcal{X}^v_{\ell_1\ell_2} \qquad \forall \ell_1\ell_2 \in E, v \in V \tag{12}$$

$$\sum_{v \in V} \mathcal{W}^v_\ell + \sum_{\ell' \in E} \mathcal{T}_{\ell\ell'} \le \text{LABELS} \qquad \forall \ell \in E \tag{13}$$

number of labels

MIP overview

1. Picks a backup path **P** for every link (s,t), whenever feasible

2. For every link **L**=(u,v)∈P that is not on SP(u,t):

   2.1. Finds an intermediate node **w**∈P s.t. SP(u,w) includes L

   2.2. else, make it a tunnel link

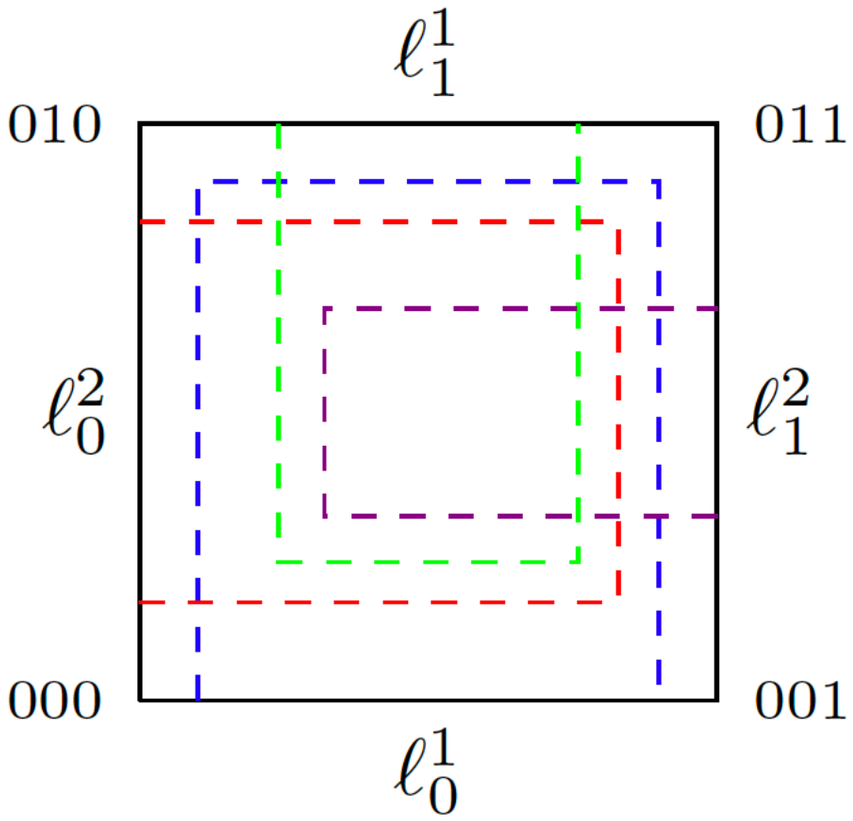
3. Keeps track of dependencies, disallows cycles shorter than f+1

4. Restricts the number of segments per BP

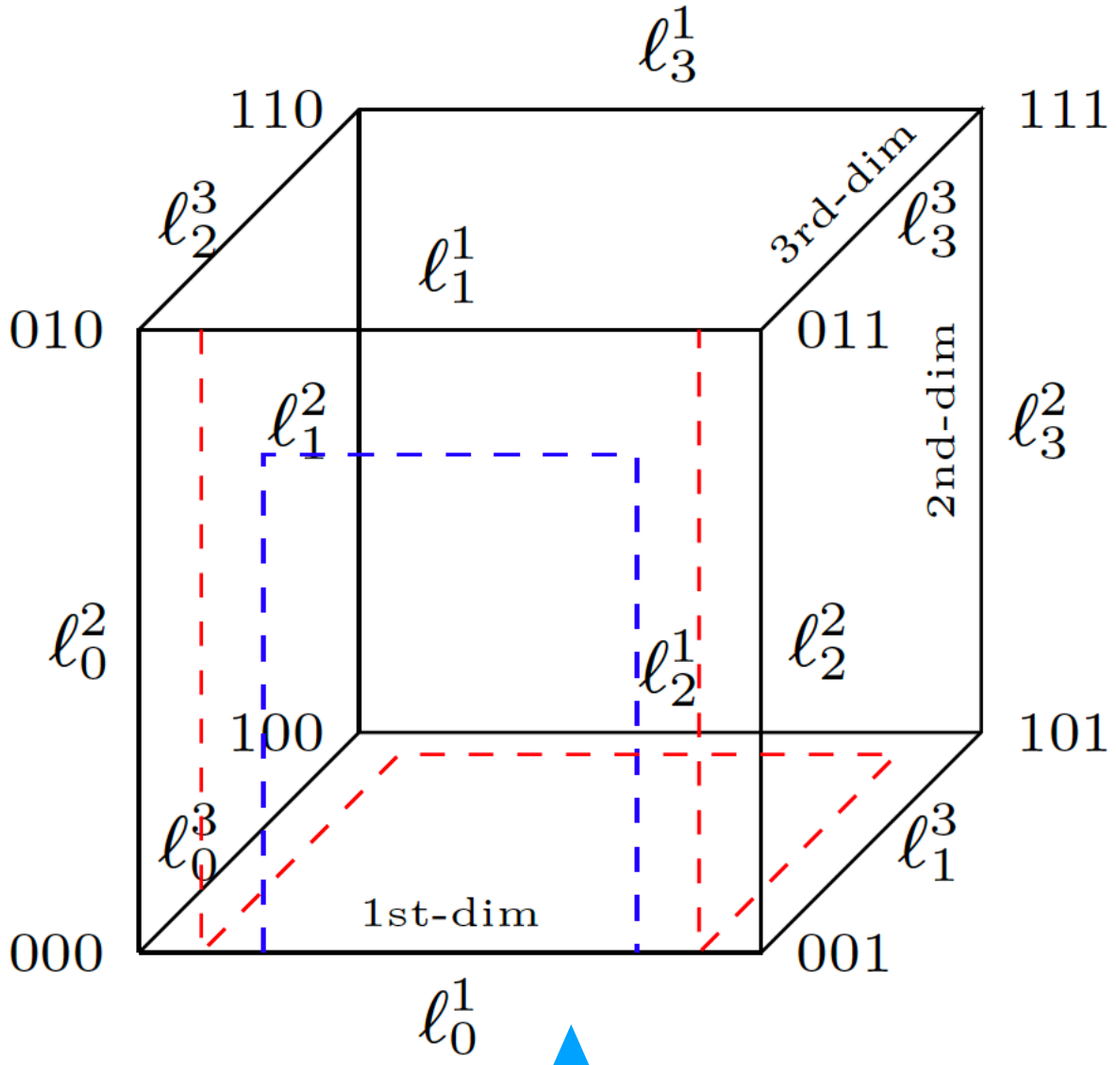
5. Maximizes the number of BP allocations

Future Works

- Backup path allocation: algorithms for more general graphs
- Complexity of the general problem, NP-Hard?
- Optimizing number of segments

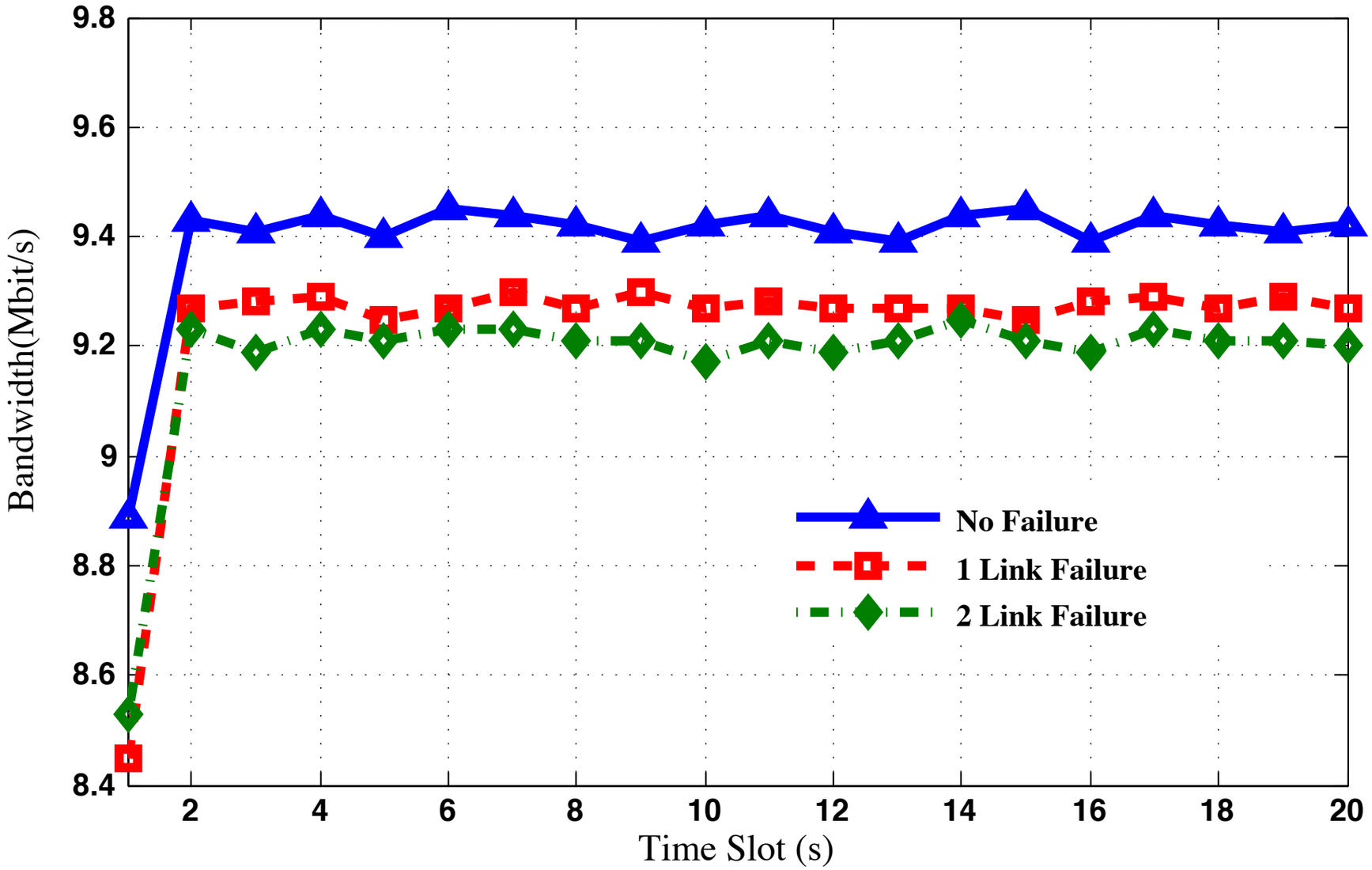


1-resilient



2-resilient

Experiments



TCP throughput of iperf3 under 0, 1, and 2 link failures in Nanonet, using an adapted version of the topology and Segment Routing rules from Figure 2.