universität wien

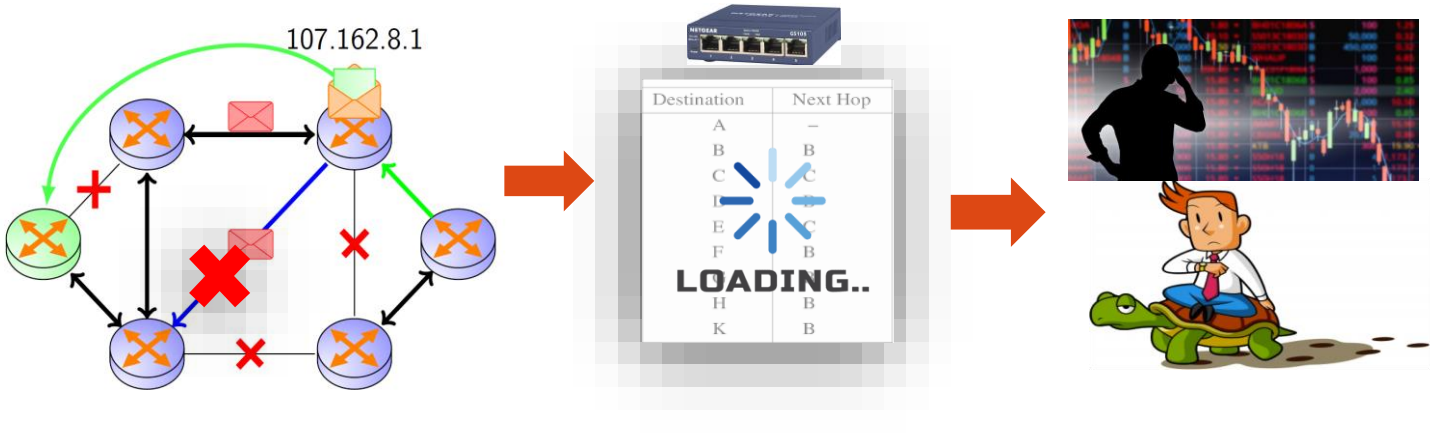
Slides by Wenkai Dai (University of Vienna)

A Tight Characterization of Fast Failover Routing: Resiliency to Two Link Failures is Possible

Wenkai Dai (University of Vienna, Austria), Klaus-Tycho Foerster (TU Dortmund, Germany), and Stefan Schmid (TU Berlin, Germany)

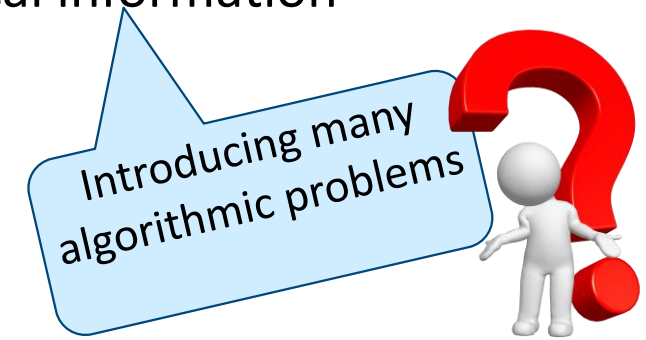
Motivation: Control-Plane Is Relatively Slow

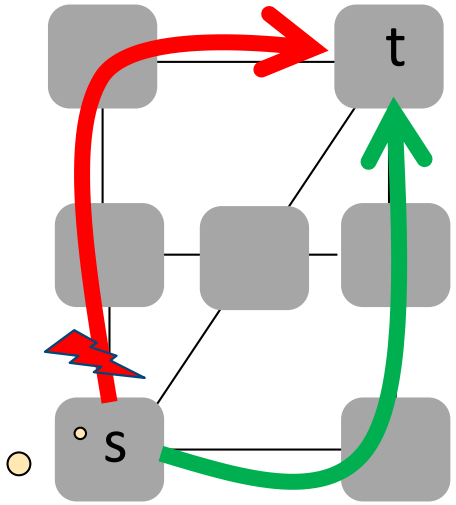
- Traditionally, control-plane triggers recomputation of routing tables to tackle link-failures
  - E.g., OSPF and IS-IS
  - Very slow (convergence time $\approx 100\ ms$), unacceptable for critical applications



Fast Rerouting (FRR) on Data-Plane

- Modern communication networks support fast reroute: local failover without invoking control plane, by pre-installing alternative paths
- Challenge: conditional forwarding rules can only depend on local information

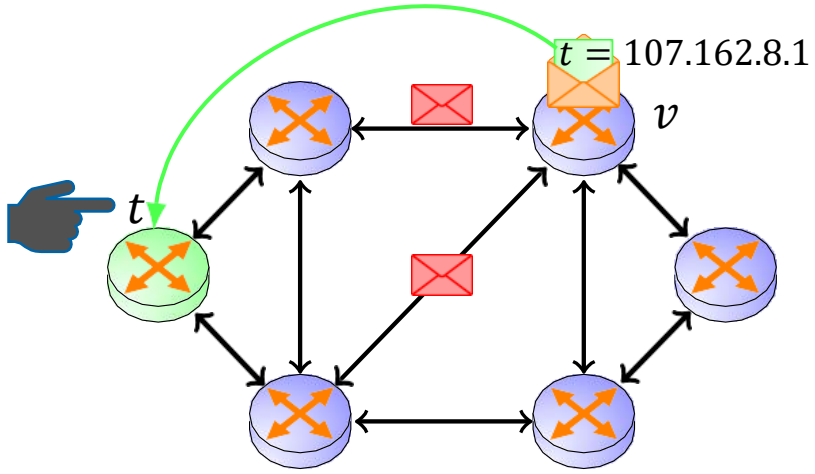




Preinstalled conditional failover rule

Model: Routing Functions Only Use Local Information

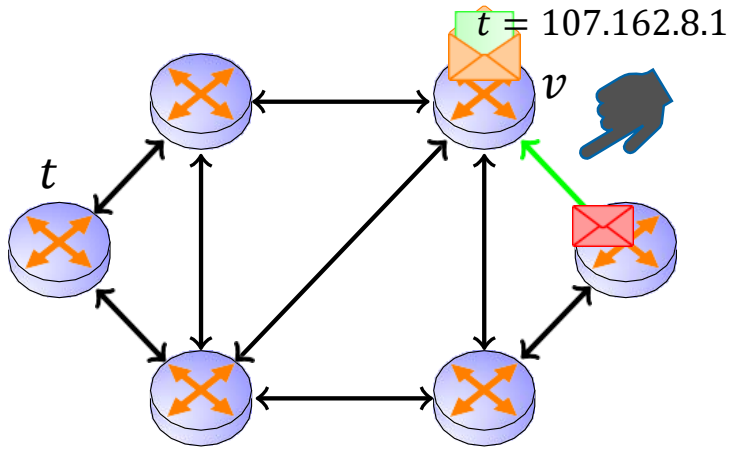
- Network is a connected undirected graph $G = (V, E)$
- Routing function at node v may only match on:
  1. Destination t



universität wien

Model: Routing Functions Only Use Local Information

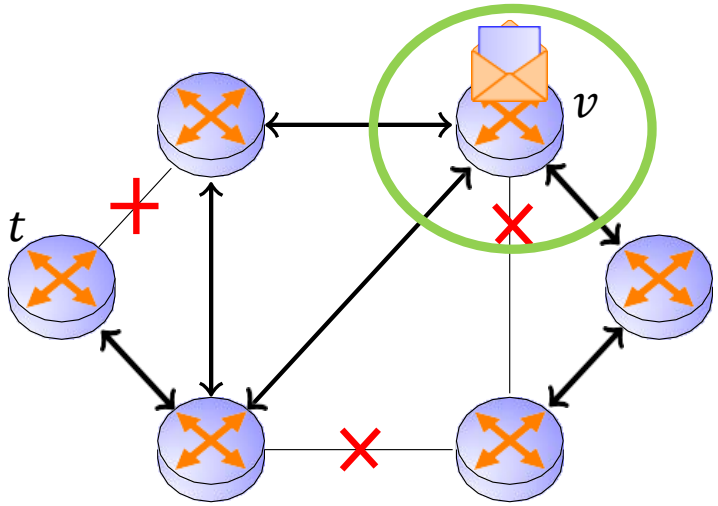
- Network is a connected undirected graph $G = (V, E)$
- Routing function at node v may only match on:
  1. Destination t
  2. Incoming port from $E(v) \cup \{\perp\}$



universität wien

Model: Routing Functions Only Use Local Information

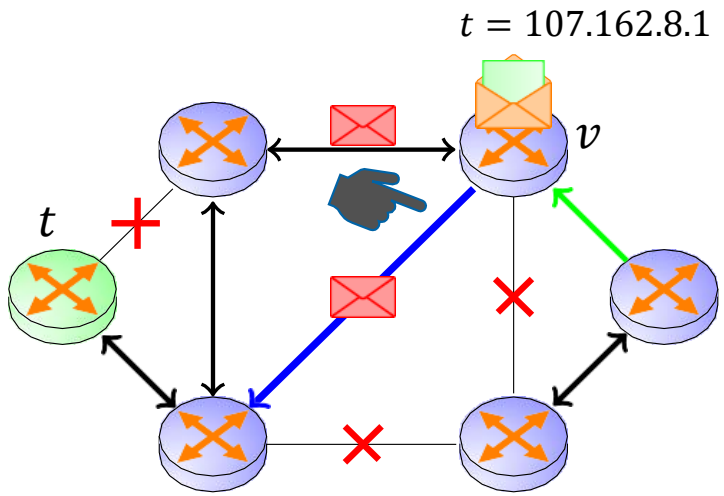
- Network is a connected undirected graph $G = (V, E)$
- Routing function at node v may only match on:
  1. Destination t
  2. Incoming port from $E(v) \cup \{\perp\}$
  3. Incident edge failures $F \cap E(v)$



Model: Routing Functions Only Use Local Information

- Network is a connected undirected graph $G = (V, E)$
- Routing function at node v may only match on:
  1. Destination t
  2. Incoming port from $E(v) \cup \{\perp\}$
  3. Incident edge failures $F \cap E(v)$

To determine an outgoing link in $E(v) - F$

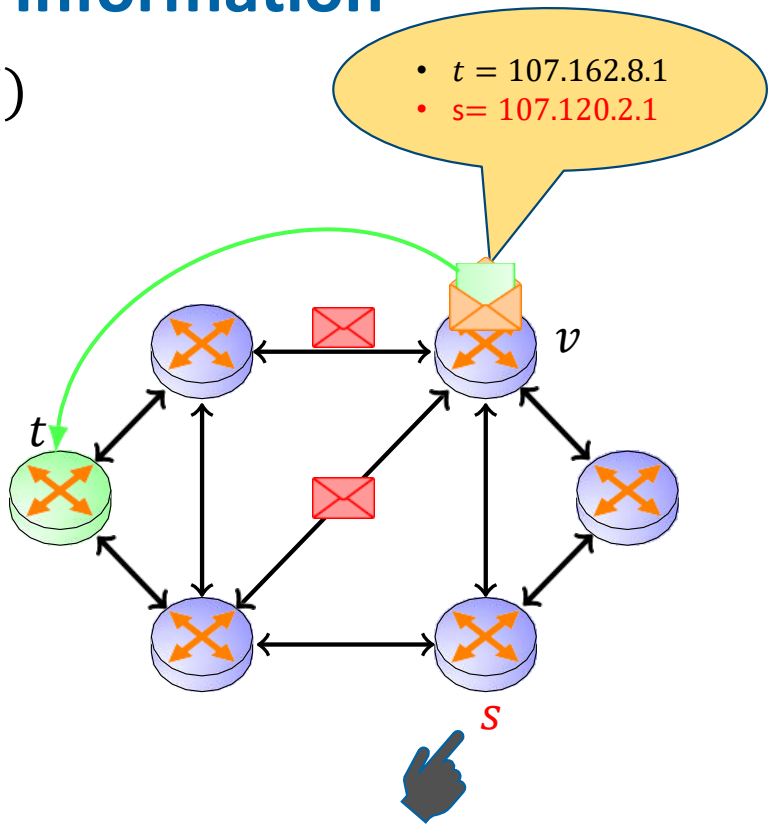


universität wien

Model: Routing Functions Only Use Local Information

- Network is a connected undirected graph $G = (V, E)$
- Routing function at node v may only match on:
  1. Destination t
  2. Incoming port from $E(v) \cup \{\perp\}$
  3. Incident edge failures $F \cap E(v)$
  4. *Optional: Source s (our model)*

  To determine an outgoing link in $E(v) - F$



universität wien

Model: Routing Functions Only Use Local Information

- Network is a connected undirected graph $G = (V, E)$
- Routing function at node v may only match on:
  1. Destination t
  2. Incoming port from $E(v) \cup \{\perp\}$
  3. Incident edge failures $F \cap E(v)$
  4. *Optional: Source s (our model)*

  To determine an outgoing link in $E(v) - F$
- Static routing tables, deterministic behavior, no rewriting bits in packets

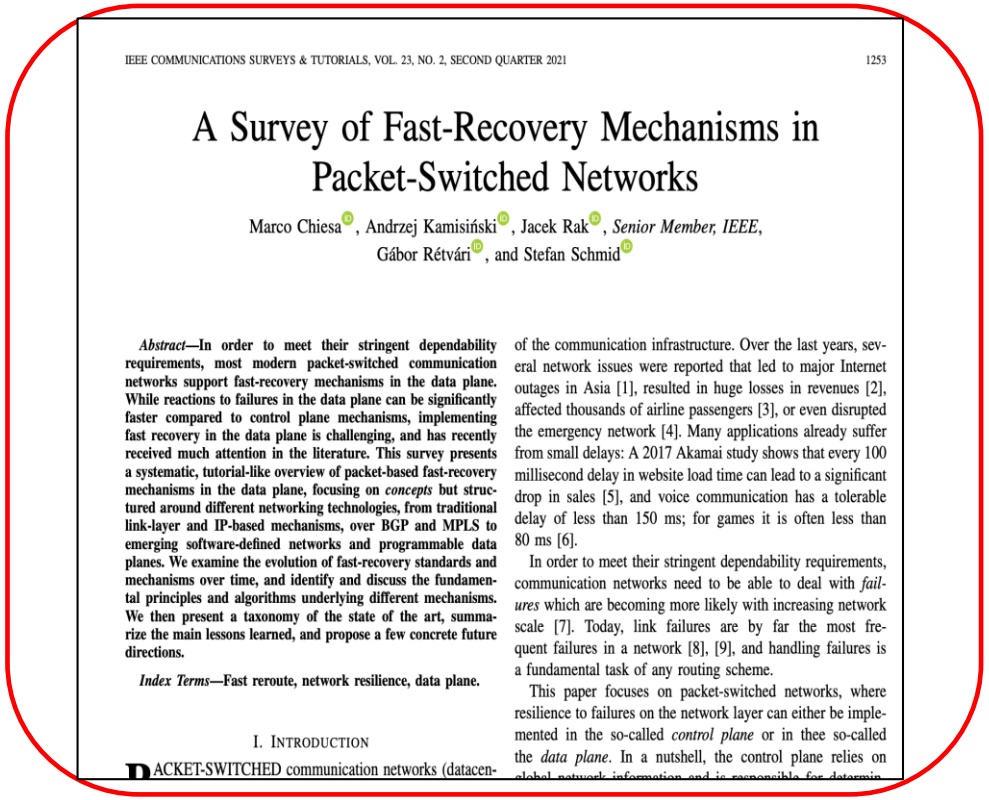
universität wien

Goal: Computing a k-Resilient Routing Scheme

- Given an undirected graph $G = (V, E)$ and a destination $t \in V$
- A routing scheme is a set of routing functions defined on V
- Let $F \subseteq E(G)$ denote k arbitrary failed edges in G
- A routing scheme is called k-**Resilient** ($1 \leq k \leq |E|$) :
  - any $v \in V$ can reach a destination $t \in V$
    - **as long as $v - t$ is still connected in $G - F$**
- Goal: Compute a k-resilient routing scheme for given $G = (V, E), t \in V$ and k

Related Works

- Fast failover is a widely studied area, see recent survey
  - Chiesa et al. (2021): *A Survey of Fast Recovery Mechanisms in Packet-Switched Networks*
- Randomized resilient routing has also been studied (not our model here)



IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 23, NO. 2, SECOND QUARTER 2021 1253

A Survey of Fast-Recovery Mechanisms in Packet-Switched Networks

Marco Chiesa, Andrzej Kamisiński, Jacek Rak, *Senior Member, IEEE*, Gábor Rétvári, and Stefan Schmid

***Abstract*—In order to meet their stringent dependability requirements, most modern packet-switched communication networks support fast-recovery mechanisms in the data plane. While reactions to failures in the data plane can be significantly faster compared to control plane mechanisms, implementing fast recovery in the data plane is challenging, and has recently received much attention in the literature. This survey presents a systematic, tutorial-like overview of packet-based fast-recovery mechanisms in the data plane, focusing on *concepts* but structured around different networking technologies, from traditional link-layer and IP-based mechanisms, over BGP and MPLS to emerging software-defined networks and programmable data planes. We examine the evolution of fast-recovery standards and mechanisms over time, and identify and discuss the fundamental principles and algorithms underlying different mechanisms. We then present a taxonomy of the state of the art, summarize the main lessons learned, and propose a few concrete future directions.**

***Index Terms*—Fast reroute, network resilience, data plane.**

I. Introduction

PACKET-SWITCHED communication networks (datacenter of the communication infrastructure. Over the last years, several network issues were reported that led to major Internet outages in Asia [1], resulted in huge losses in revenues [2], affected thousands of airline passengers [3], or even disrupted the emergency network [4]. Many applications already suffer from small delays: A 2017 Akamai study shows that every 100 millisecond delay in website load time can lead to a significant drop in sales [5], and voice communication has a tolerable delay of less than 150 ms; for games it is often less than 80 ms [6].

In order to meet their stringent dependability requirements, communication networks need to be able to deal with *failures* which are becoming more likely with increasing network scale [7]. Today, link failures are by far the most frequent failures in a network [8], [9], and handling failures is a fundamental task of any routing scheme.

This paper focuses on packet-switched networks, where resilience to failures on the network layer can either be implemented in the so-called *control plane* or in thee so-called the *data plane*. In a nutshell, the control plane relies on

universität wien

Related Works (Without Matching Source)

Per-destination	Per source	Incoming port	Incident links	k-edge-connected graph	Packet rewriting bits	Resiliency	Ref.
X			X			No	[Kwong et al. 2011]
X		X	X			1	[Feigenbaum et al. 2012]
X		X	X			No 2-resilient	[Chiesa et al. 2016]
X		X	X	$k \leq 5$		$(k-1)$	[Chiesa et al. 2016]
X		X	X	$k \to \infty$		$\lfloor k/2 \rfloor$	[Chiesa et al. 2016]
X		X	X	$k > 5$		?	Open
X		X	X	$k > 5$	3 bits	$(k-1)$	[Chiesa et al. 2016]
X		X	X	$k > 5$	$\log k$ bits	$(k-1)$	[Chiesa et al. 2016]

universität wien

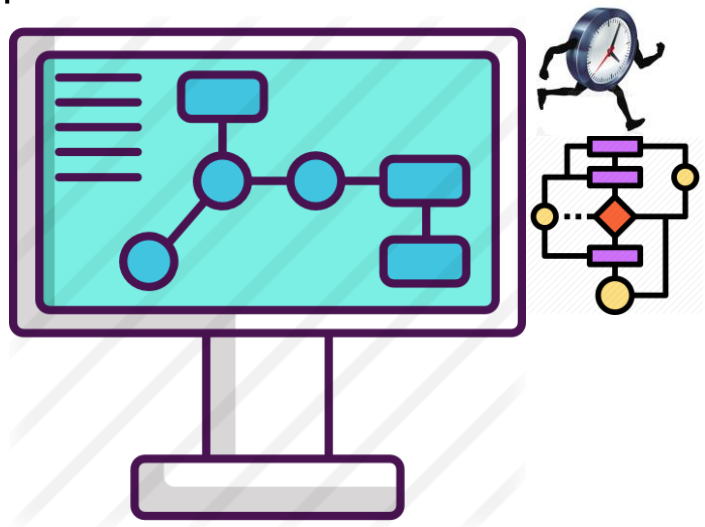
Related Works (Matching Source)

Per-destination	Per source	Incoming port	Incident links	k-edge-connected graph	Packet rewriting bits	Resiliency	Ref.
X	X	X	X	$k \to \infty$		$(k-1)$	[Foerster et al. 2019]
X	X	X	X			No ∞-resiliency	[Foerster et al. 2021]
X	X	X	X			1	[Feigenbaum et al. 2012]
X	X	X	X			$? = 2, 3, \ldots$	Open
X	X	X	X			2	[This paper]
X	X	X	X			No 3-resiliency	[This paper]

universität wien

Summary of Our Results

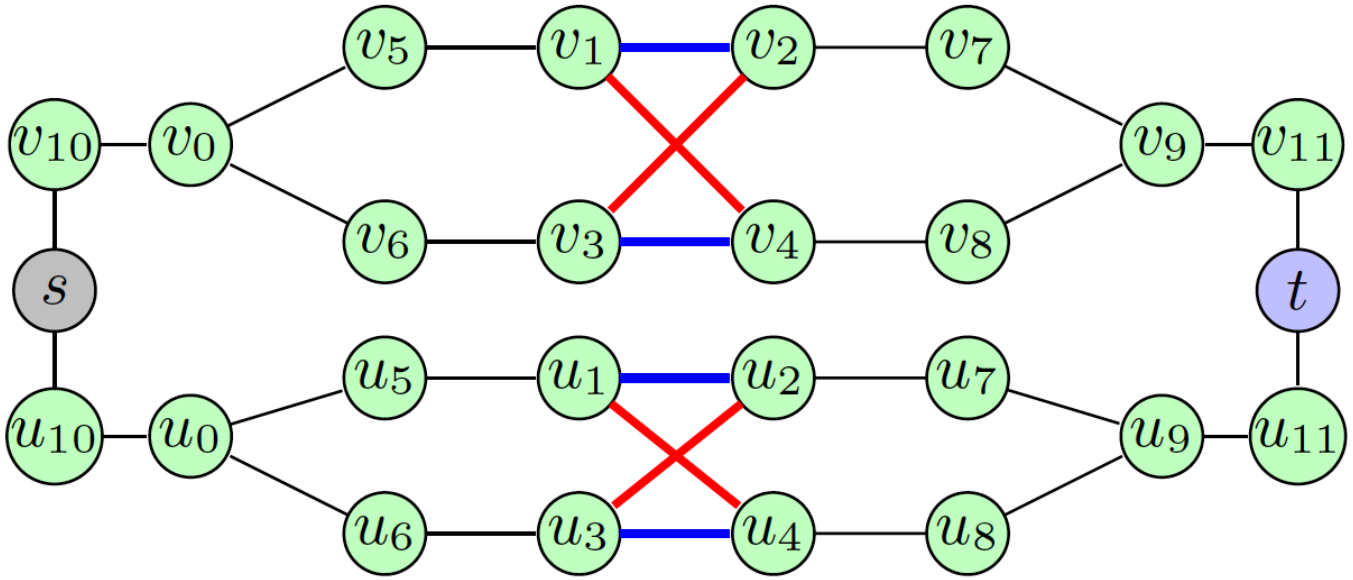
- A 2-resilient source-matched routing scheme is computable within $O(n \cdot m)$, where $m = |E|$ and n= $|V|$

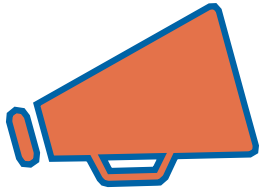


- No $\geq$ 3-resilient source-matched routing scheme in general graphs

3-Resilient Source-Matched Routing Is Impossible

- In the following network, the source s cannot reach t after **three** link failures even if $s - t$ is connected

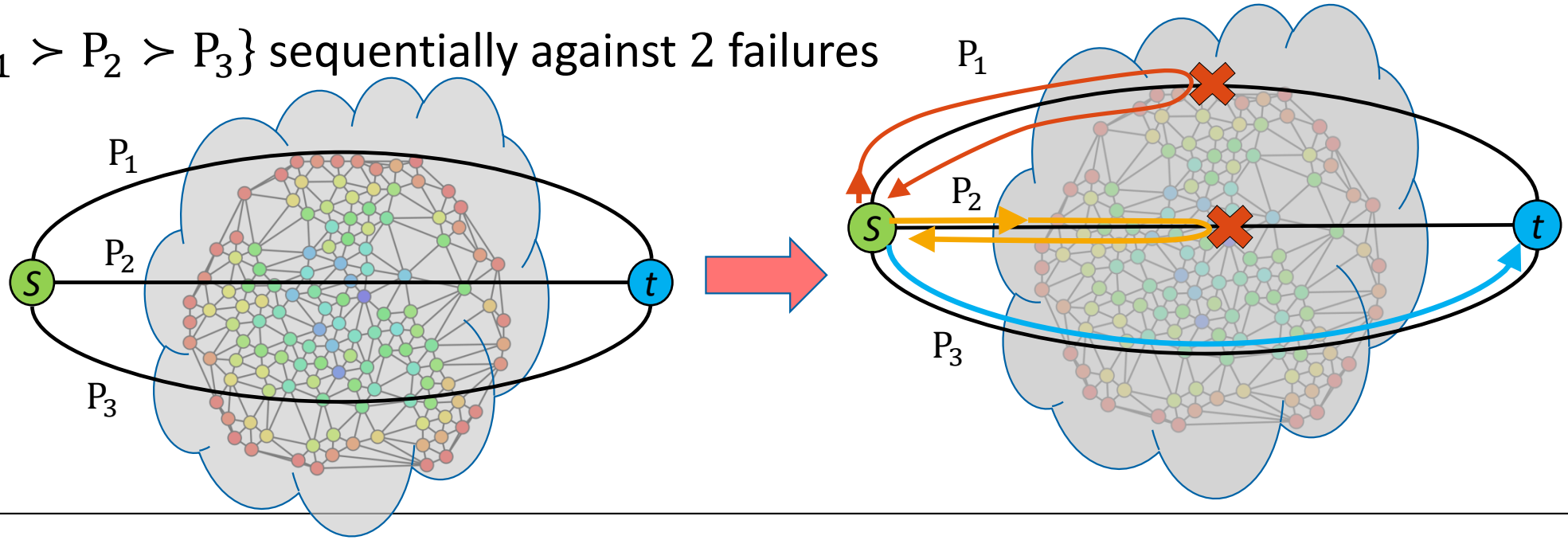




- From now on, k-***connected*** always means k-***edge-connected***

Computing a 2-Resilient Source-Matched Routing Scheme

- 2-resiliency is obvious if $s-t$ is at least 3-connected in G
- Find 3 edge-disjoint $s-t$ paths: $\{P_1, P_2, P_3\}$
- Routing on $\{P_1 \succ P_2 \succ P_3\}$ sequentially against 2 failures

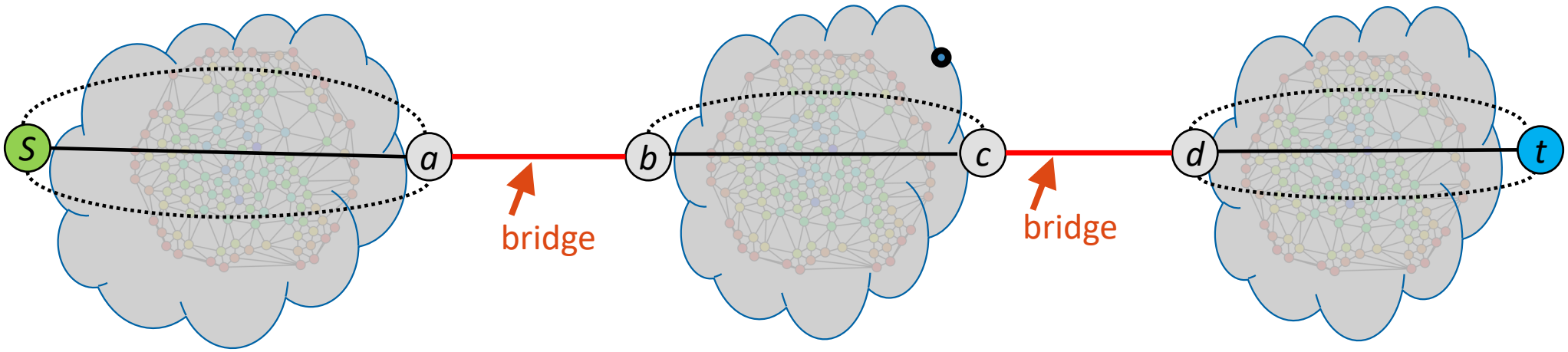


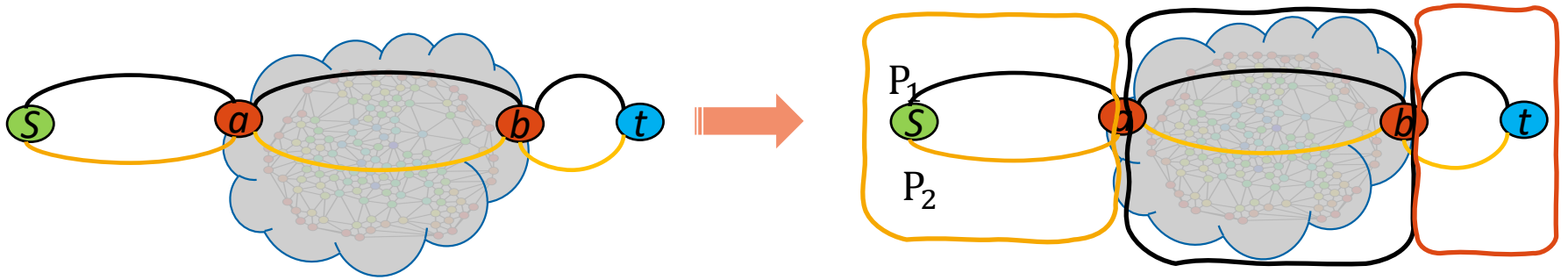
universität wien

Reduced to sub-instances:
Source and destination are exactly 2-connected

Reduction to Instances Where $s - t$ Is Exactly 2-Connected

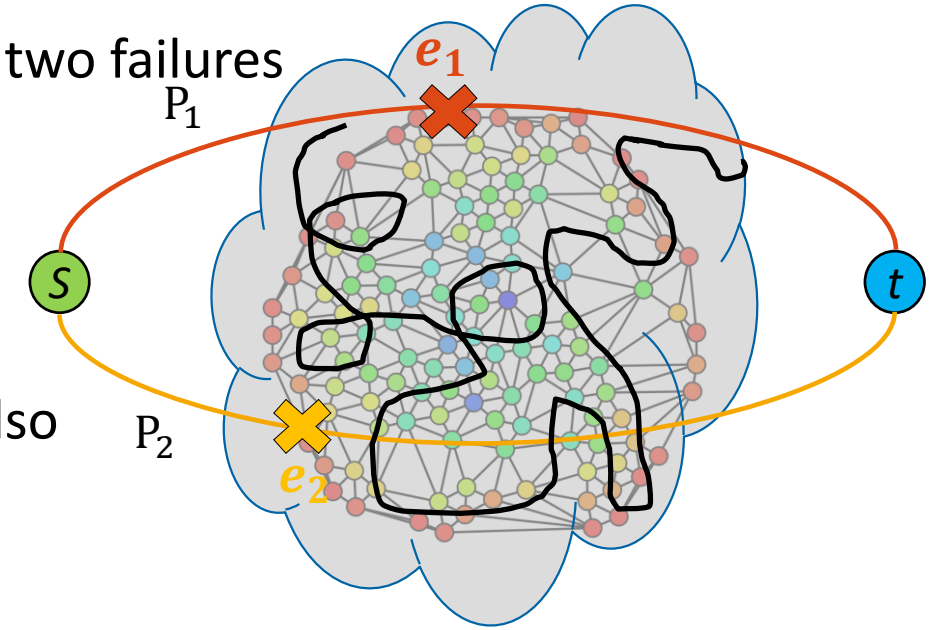
- If $s - t$ is 1-connected in G, by removing all bridges, $s - t$ in each connected component is at least 2-connected
- Obvious when $s - t$ is 3-connected





Still Challenging When $s - t$ Is Exactly 2-Connected in G

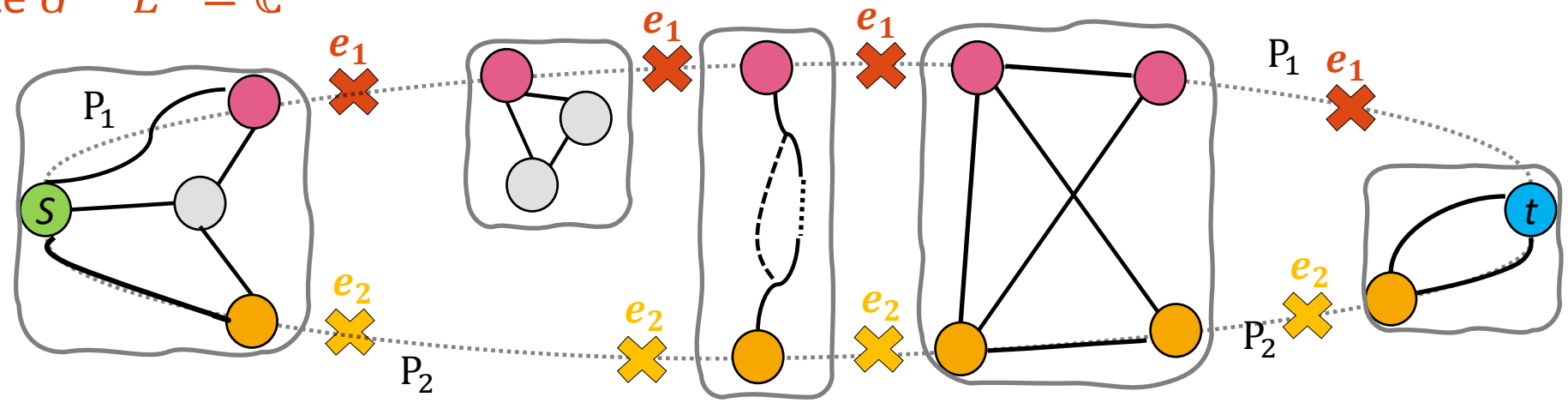
- Two edge-disjoint $s - t$ paths P_1 and P_2 are not enough for two failures
- A path avoiding failures might exist
- Hard to determine useful structure in general graphs
- Further reduction:
  - Assume two edge-disjoint $s - t$ paths P_1 and P_2 in G are also node-disjoint (splitting articulation points)



universität wien

Obtain Connected Components of G after Two Failures

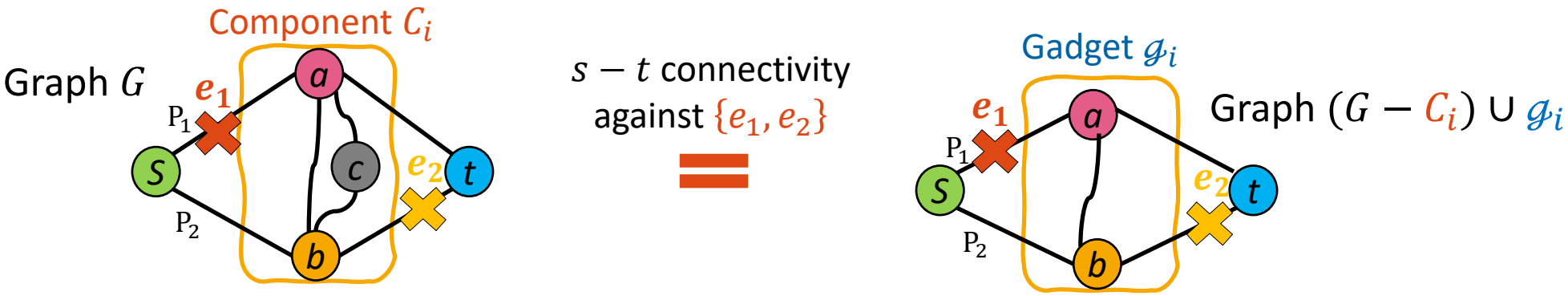
- For any $\{e_1, e_2\} \subset E$, if $s - t$ disconnected in $G - \{e_1, e_2\}$, let $E^c = \biguplus\{e_1, e_2\}$
- Obtain a graph $G - E^c$ containing a set of connected components $\mathbb{C}$
- Denote $G - E^c = \mathbb{C}$



universität wien

Finding Gadgets for Each Connected Component $C_i \in \mathbb{C}$

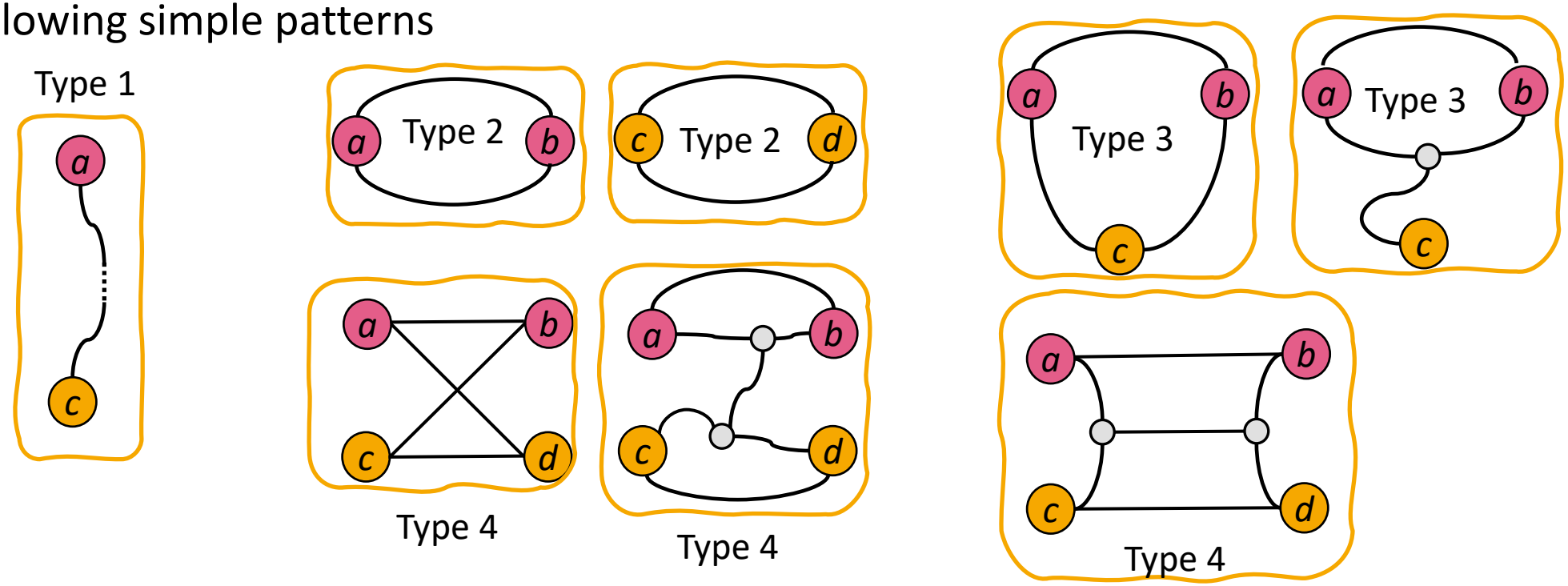
- For each $C_i \in \mathbb{C}$ in G, our algorithm can find a ***gadget*** $\mathcal{g}_i$ of C_i:
  - A gadget $\mathcal{g}_i$ is a ***subgraph*** of $C_i \in \mathbb{C}$, i.e., $\mathcal{g}_i \subseteq C_i$
  - By replacing C_i with $\mathcal{g}_i$ in G, the $\boldsymbol{s-t}$ ***connectivity*** remains **unchanged** against two failures $\{e_1, e_2\} \subset E$



universität wien

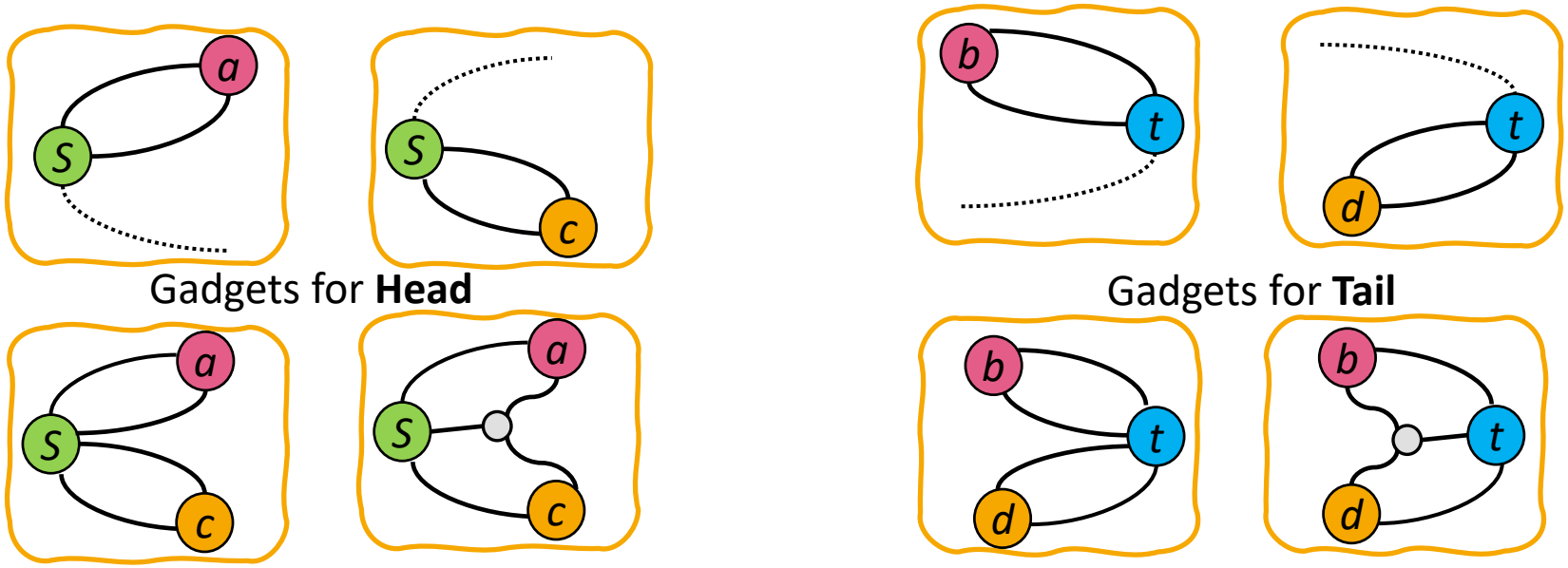
Representing Gadgets for Components

- Gadgets for all components $C_i \in \mathbb{C}$ can be represented by the following simple patterns



Representing Gadgets for Components (2)

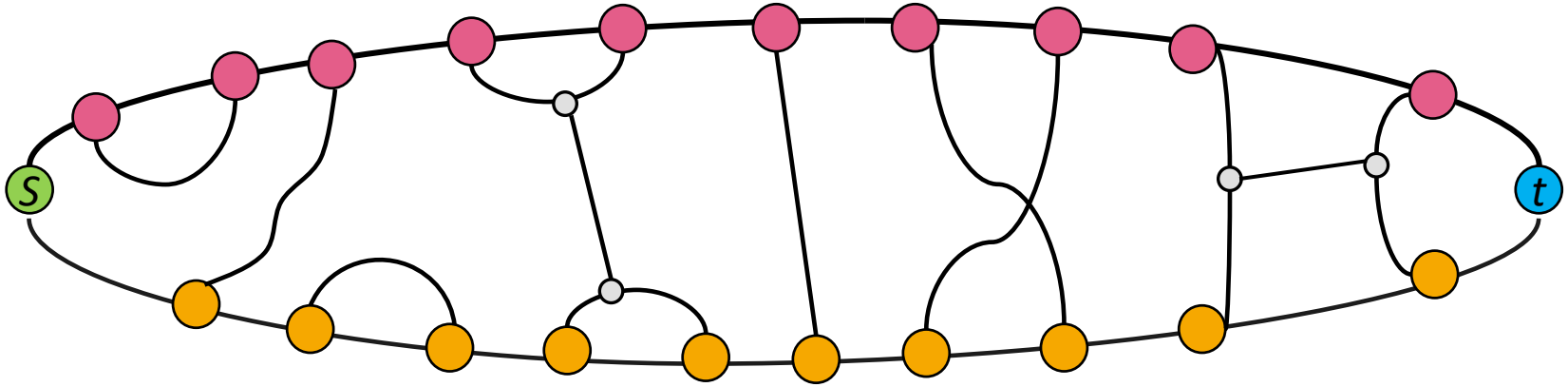
- If $C_i \in \mathbb{C}$ contains s or t, its gadget can be represented by these patterns



universität wien

Obtain a Kernel Graph Based on Gadgets

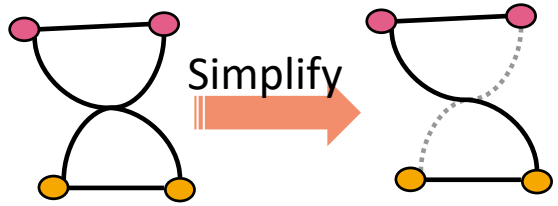
- In a graph G, replacing each component $C_i \in \mathbb{C}$ by its gadget $\mathcal{g}_i$
- Obtain a ***kernel graph*** $\mathcal{G} \coloneqq \bigcup_{C_i \in \mathbb{C}} \mathcal{g}_i \cup E^c$, where $\mathcal{G} \subseteq G$
- **Fact:** ***The $s-t$ connectivity remains the same in $\mathcal{G}$ as G after any two failures***

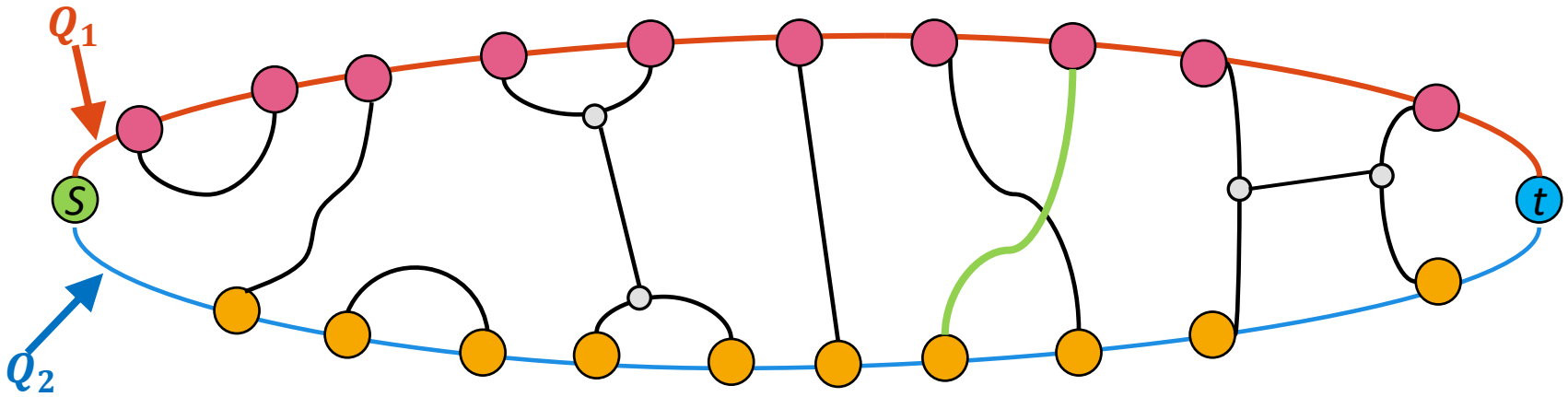


universität wien

Determine 2-Resilient Source-Matched Routing in Kernel Graph

- If a gadget has ***crossing paths***, simplify it by ignoring one path
- Now, simplified kernel graph is a planar graph, where paths Q_1 ∪ Q_2 are the ***boundary*** of the ***outer face*** on the drawing.

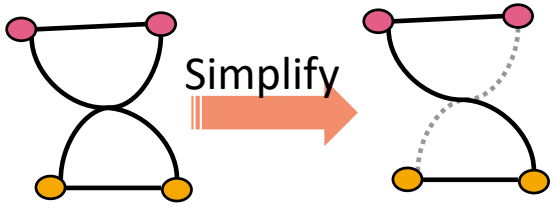


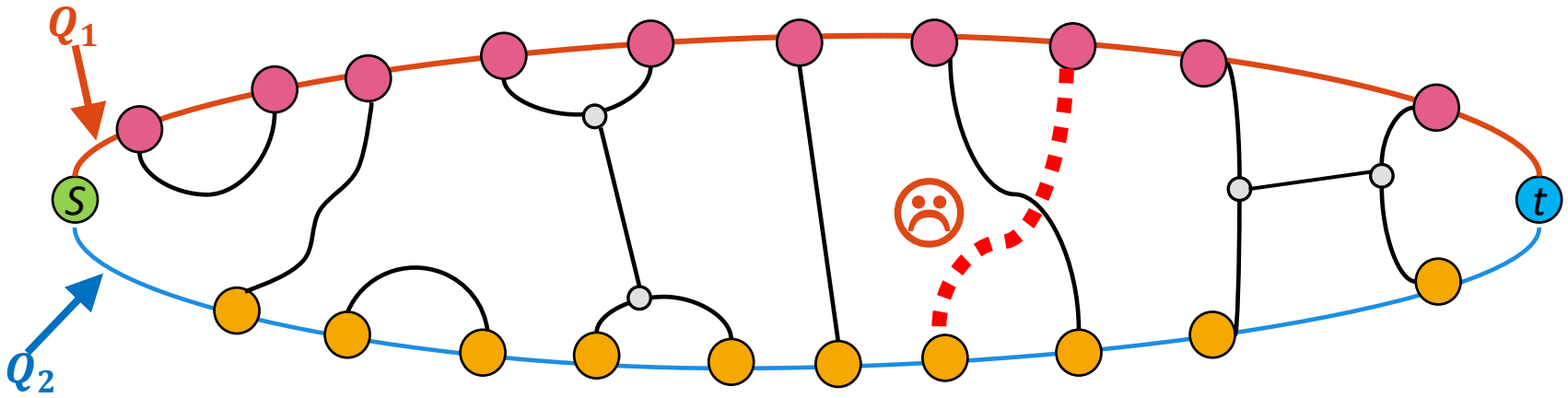


universität wien

Determine 2-Resilient Source-Matched Routing in Kernel Graph

- If a gadget has ***crossing paths***, simplify it by ignoring one path
- Now, simplified kernel graph is a planar graph, where paths Q_1 ∪ Q_2 are the ***boundary*** of the ***outer face*** on the drawing.

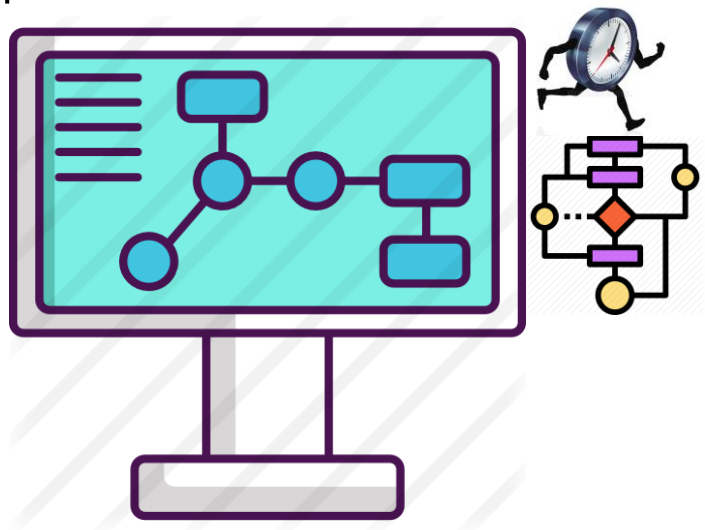




- **Routing on the outer face is easy ☺**
- **But we removed some edges..? ☹**
  - **Fix in the paper ☺**

Summary

- A 2-resilient source-matched routing scheme is computable within $O(n \cdot m)$, where $m = |E|$ and n= $|V|$



- No $\geq$ 3-resilient source-matched routing scheme in general graphs

universität wien

Thank you! ☺

A Tight Characterization of Fast Failover Routing: Resiliency to Two Link Failures is Possible

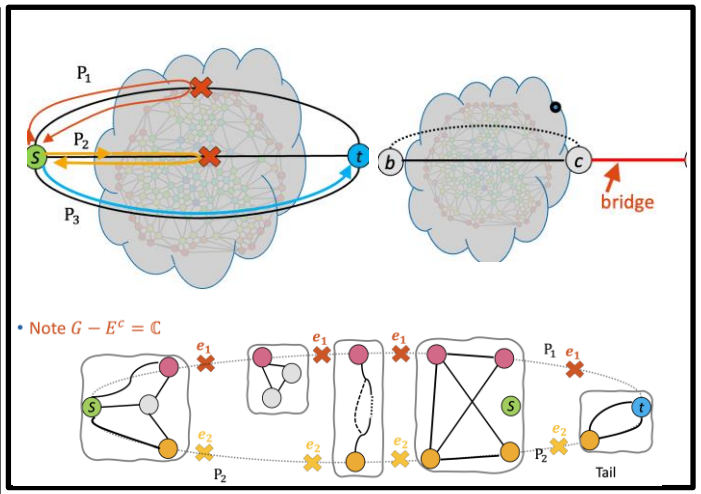
Wenkai Dai (University of Vienna, Austria), Klaus-Tycho Foerster (TU Dortmund, Germany), and Stefan Schmid (TU Berlin, Germany)

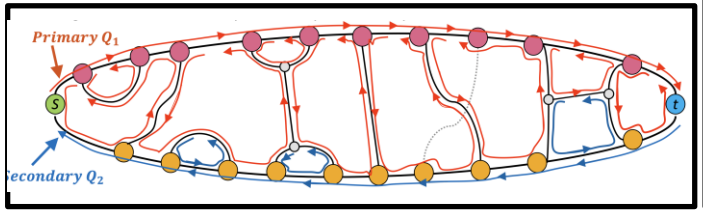
Model: Routing Functions only Using Local Information

- Network is a connected undirected graph $G = (V, E)$
- Basic routing function at node v may only match on:
  1. Destination t
  2. Incoming port from $E(v) \cup \{\perp\}$
  3. Incident edge failures $F \cap E(v)$
  4. *Optional: Source s (our model)*

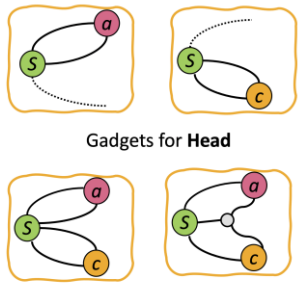
  To determine an outgoing link in $E(v) - F$
- Static routing tables, deterministic behavior (no rewriting bits in packets)
- Routing functions are computed *ahead of time*
  - Very slow (converge time $\approx 100\ ms$), unacceptable for critical

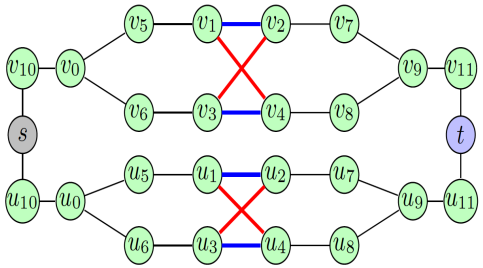
Fast reroute on data-plane by pre-installing alternative paths





- A 2-resilient source-matched routing scheme is computable within $O(n \cdot m)$, where $m = |E|$ and $n = |V|$
- No $\geq$ 3-resilient source-matched routing scheme in general graphs



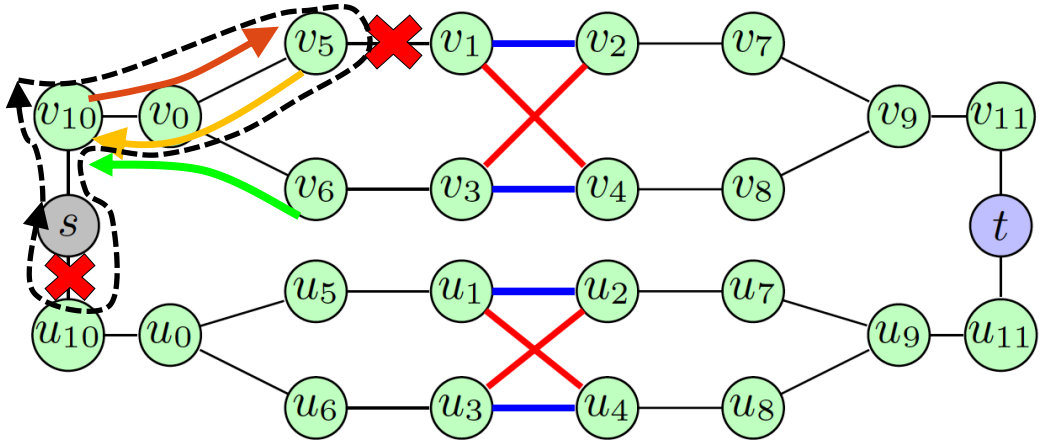


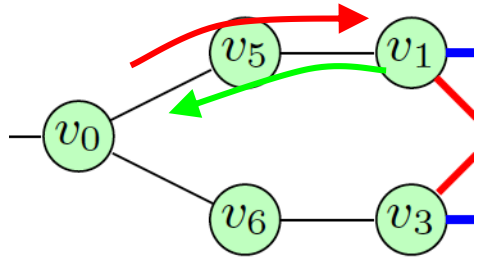
Slides by Wenkai Dai (University of Vienna)

universität wien

Intuitions for Devising 2-Resilient Routing

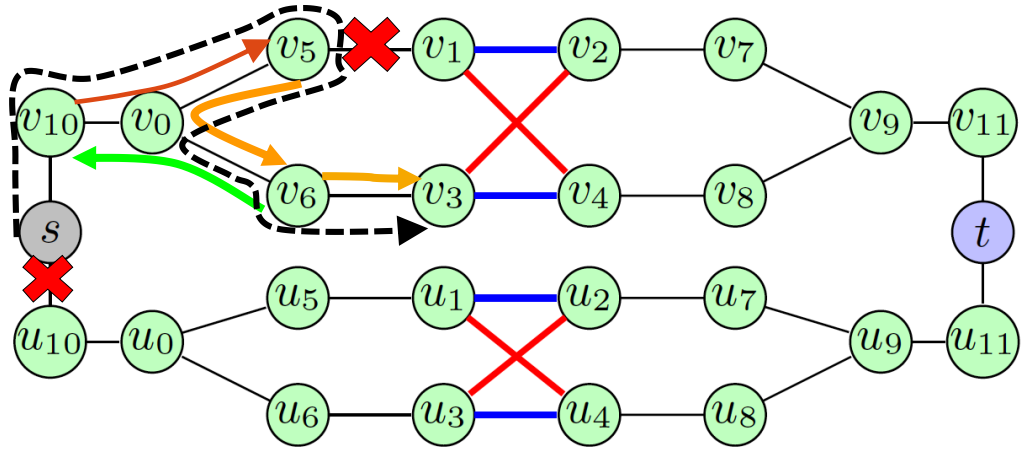
- E.g., define routing function at v_0 as follows:
  - Then it cannot be 2-resilient since a routing loop occurs

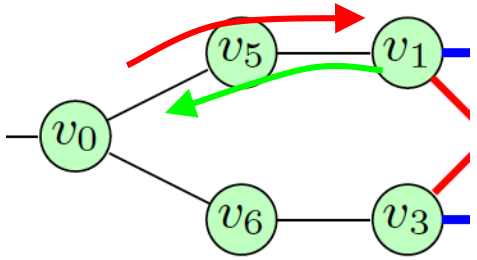




Intuitions for Devising 2-Resilient Routing (2)

- Another routing function can easily avoid the loop



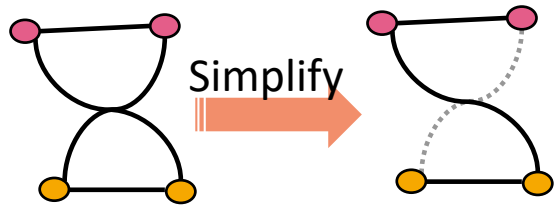


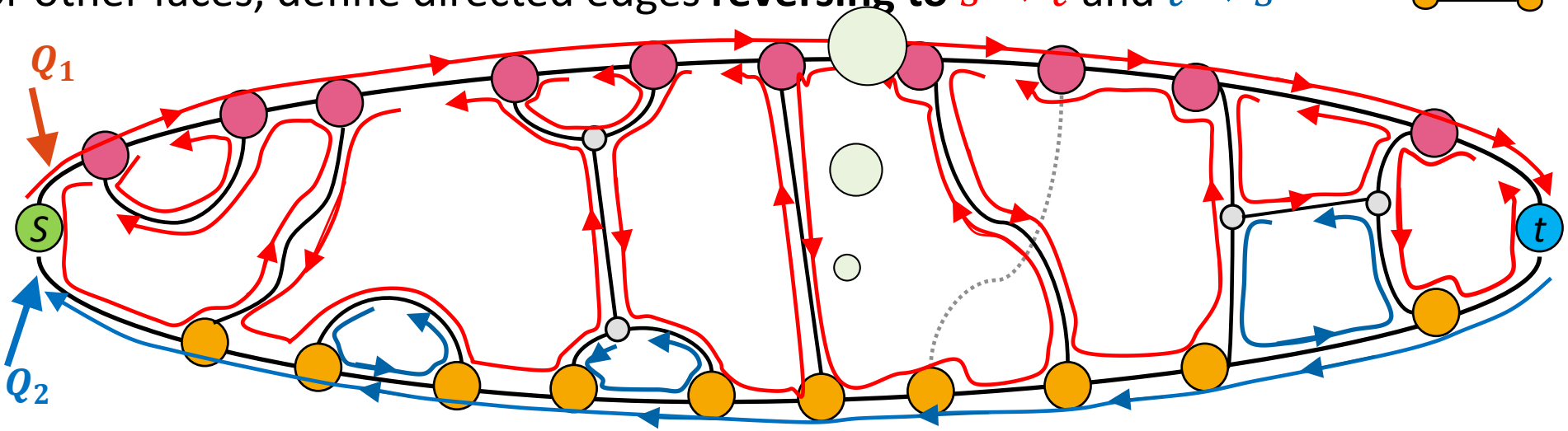
universität wien

- Routing functions defined by the directed paths and cycles
- Starting at s, first try Q_1
- If facing a failure, switch to the directed path using this failure as in-coming link

2-Resilient Sour…

- In a *simplified kernel grap…* *…ed:*
  - For the outer face $Q_1 \cup Q_2$, define directed … and then $t \to s$
  - For other faces, define directed edges **reversing to** $s \to t$ and $t \to s$

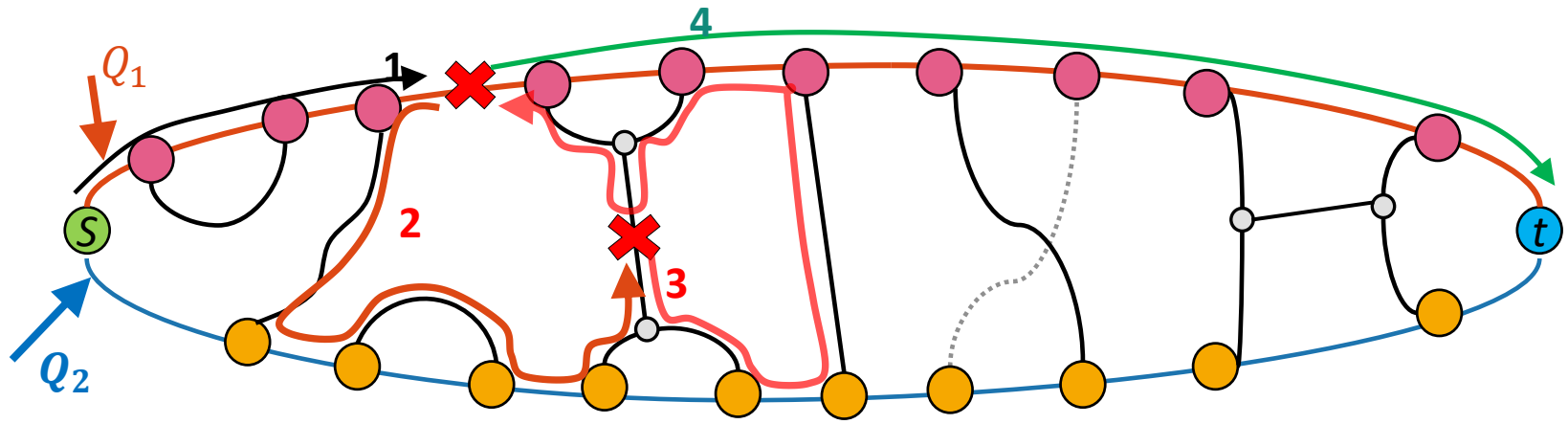




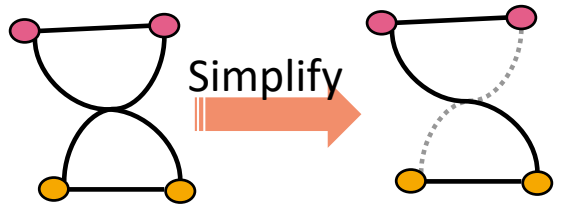
universität wien

Correctness of 2-Resilient Routing in Simplified Kernel Graph

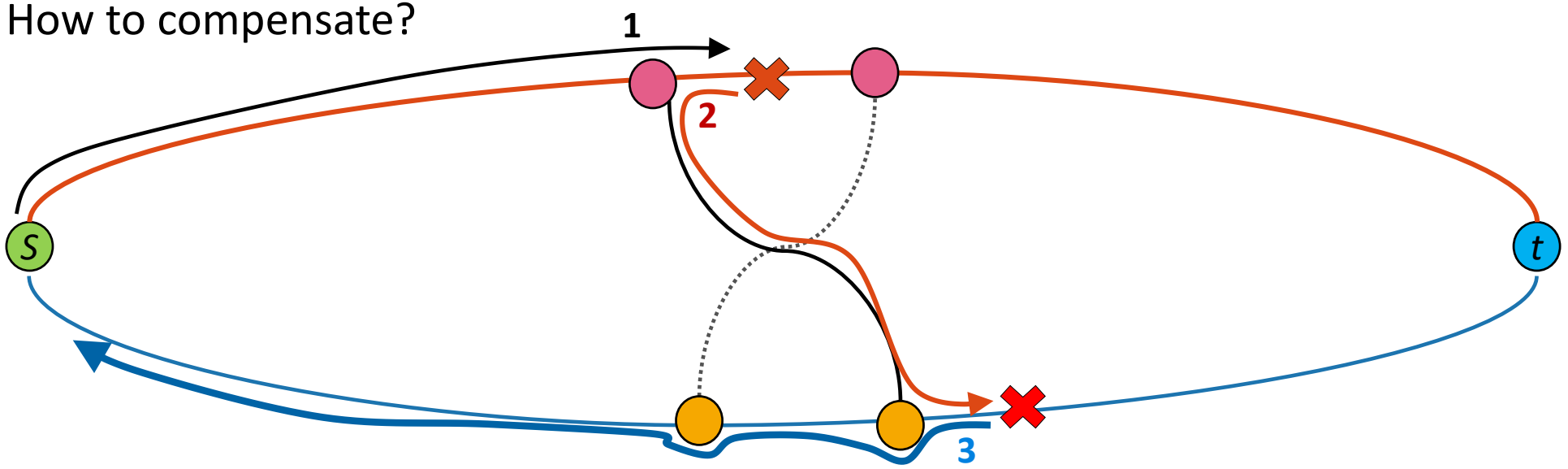
- **Proof of Correctness:**
  - Starting at s, always walking on the outer face $s \rightarrow t$ of the residual graph up to 2 failures
  - Each failure makes the detouring paths being new outer face of residual graph



Problem of Simplified Kernel Graph

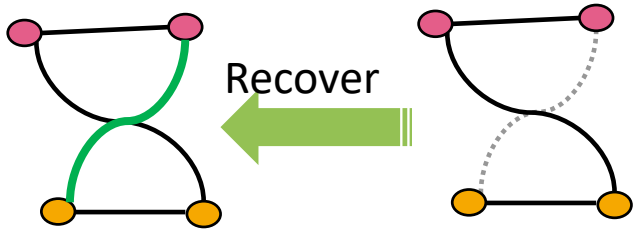


- Drawback of simplification:
  - $s - t$ is still connected in the example, but routing cannot reach t anymore
- How to compensate?



universität wien

Additional Routing Rule for Simplified Gadgets



- First, recover the ignored crossing paths
- **Additional Rule:** add one directed path as shown in the example (green path)
- **Correctness:** only used after seeing two failures, and directed path cannot introduce a loop