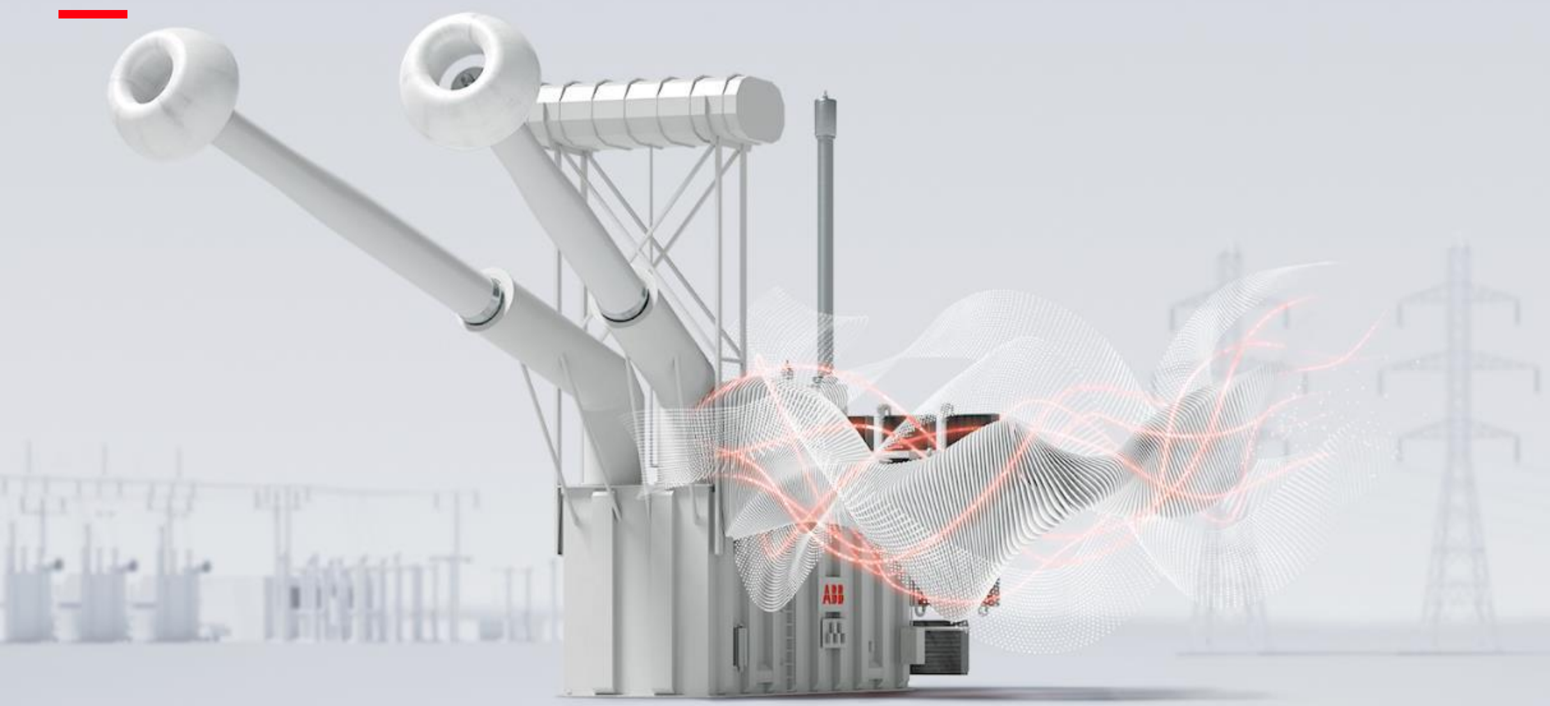
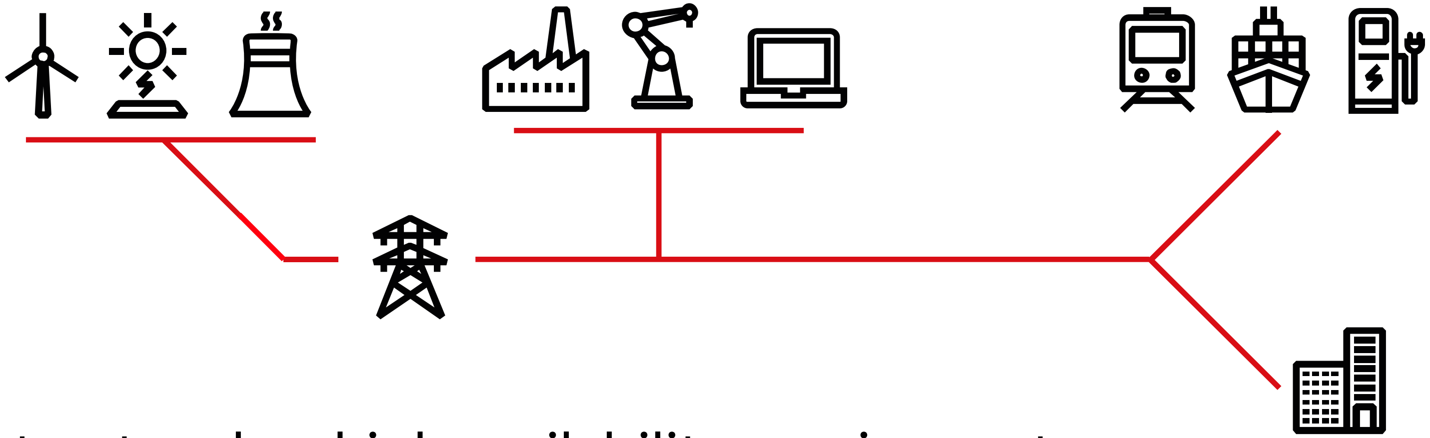


DSN, DENVER, 2017 – 06 - 28

Load-Optimal Local Fast Rerouting for Resilient Networks

Yvonne-Anne Pignolet, ABB Corporate Research
Stefan Schmid, University of Aalborg
Gilles Trédan, LAAS-CNRS, Toulouse

Motivation

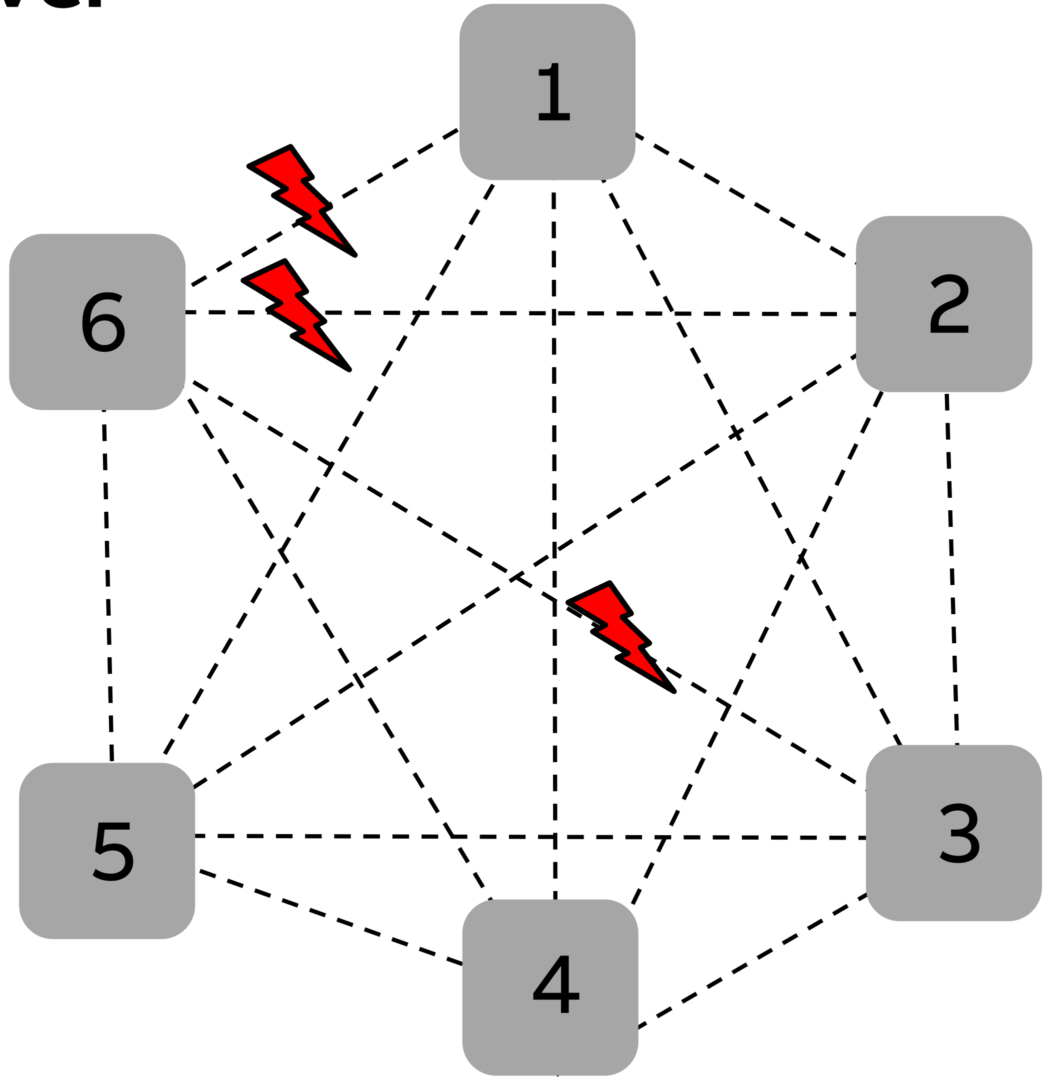


– Critical infrastructure has high availability requirements
– Industrial systems are more and more connected
– Hard real-time requirements

$\Rightarrow$ How to provide dependability guarantee despite link failures in networks?
$\Rightarrow$ Possible without communication between nodes? And low load?

Local Fast Failover

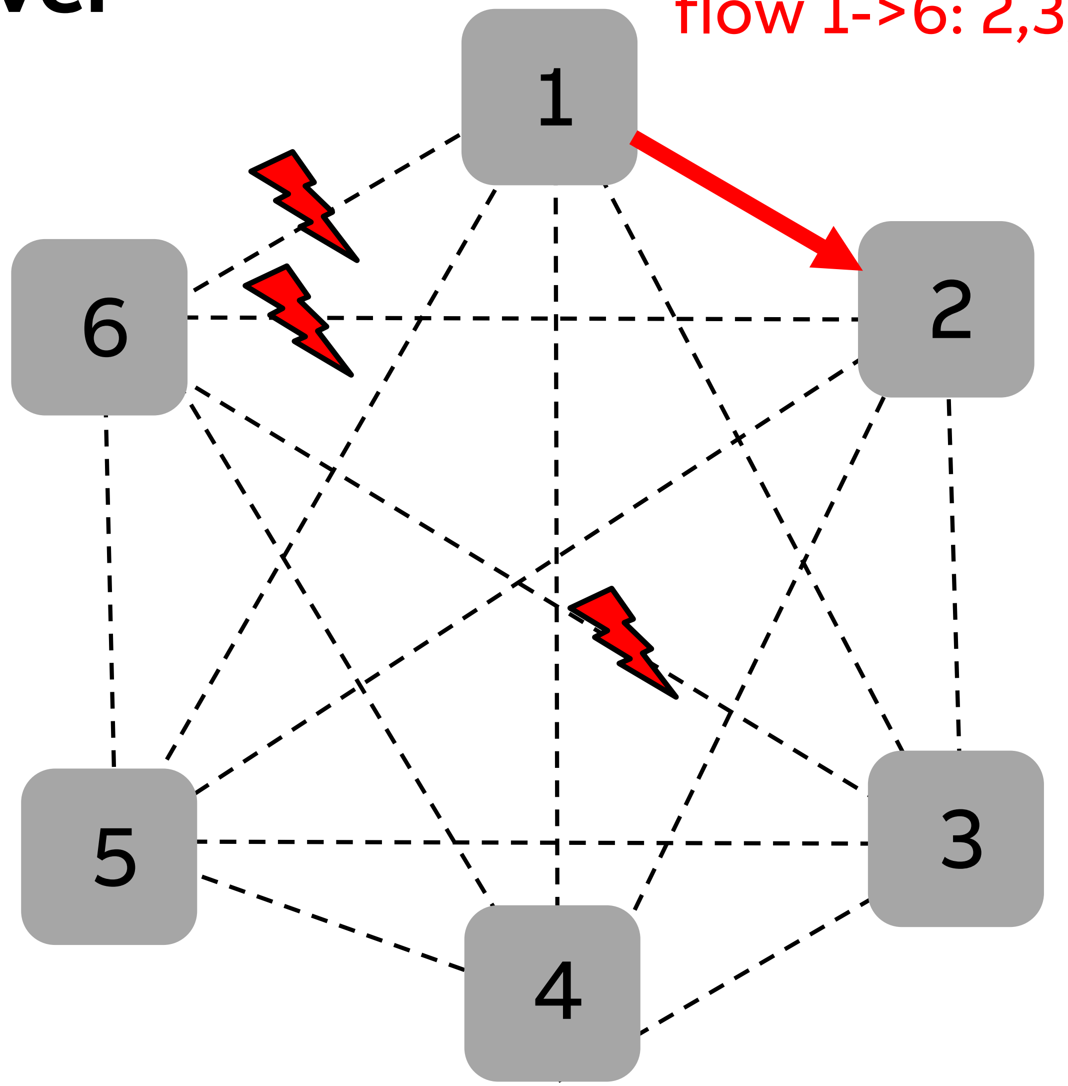
Traffic demand:
{1,2,3}->6



Local failover @1:
Does not know
failures downstream!

Local Fast Failover

Failover matrix:
flow 1->6: 2,3,4,5,…

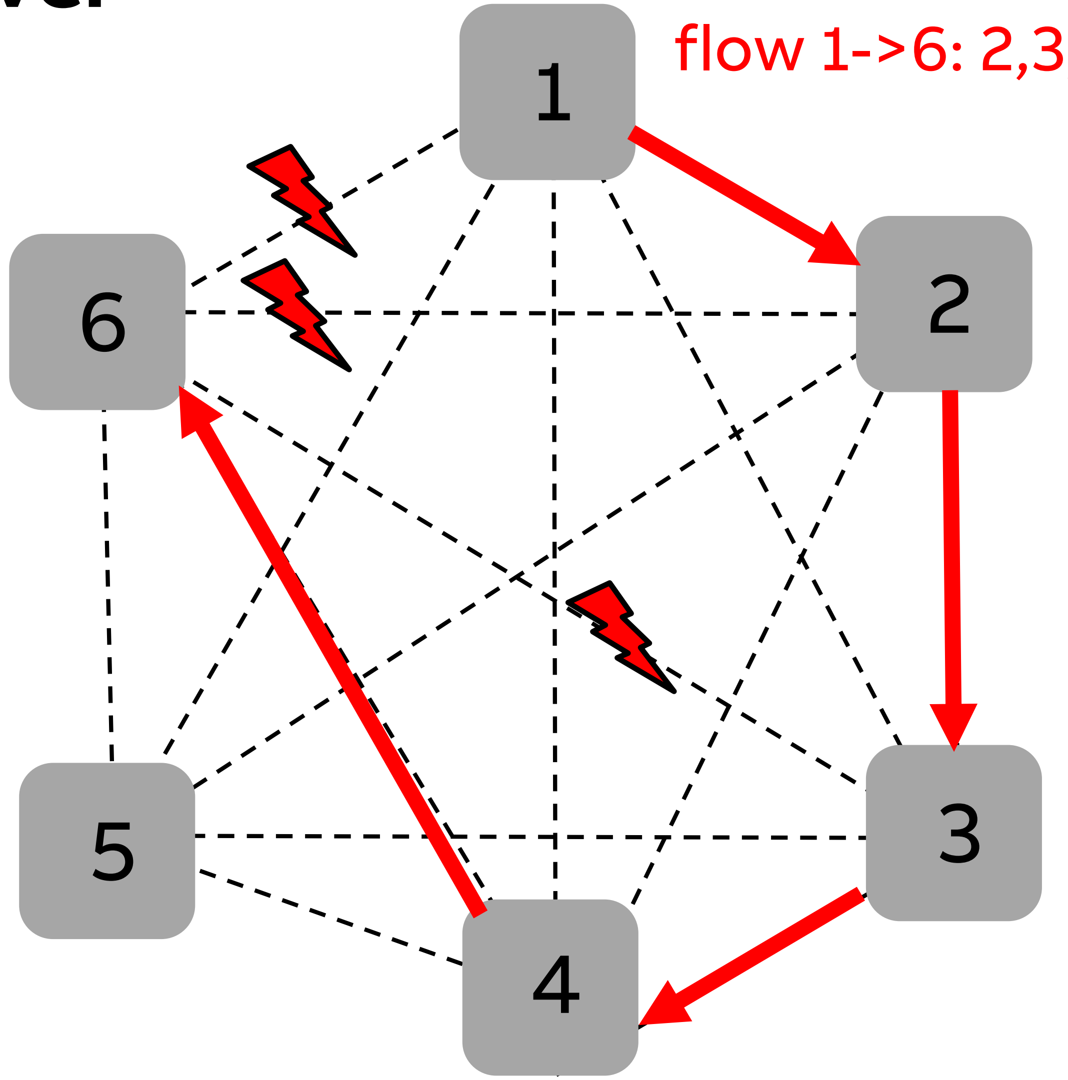


Local failover @1:
Reroute to 2!

Traffic demand:
{1,2,3}->6

Local Fast Failover

Failover matrix:
flow 1->6: 2,3,4,5,...



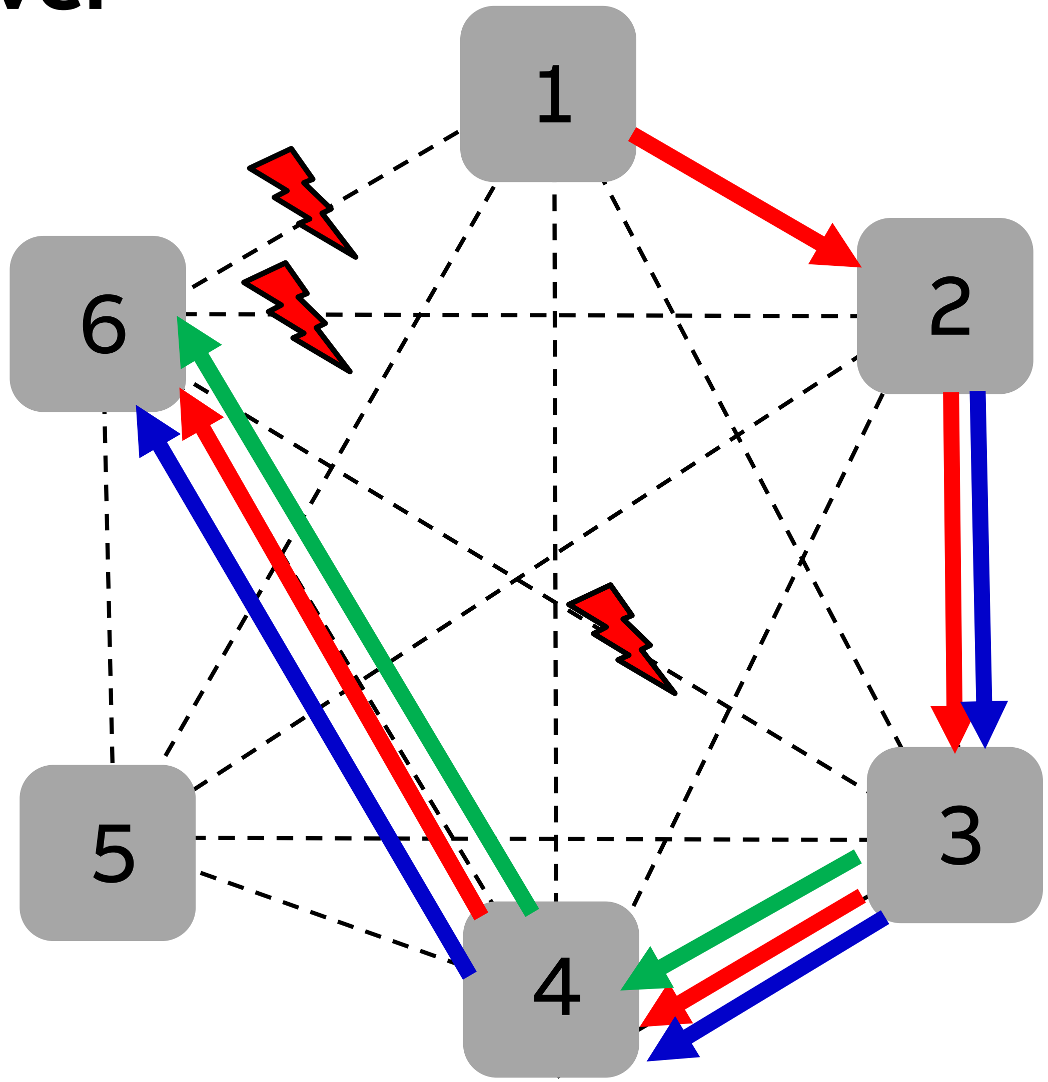
Traffic demand:
{1,2,3}->6

Local failover @1:
Reroute to 2!

But also from 2:
6 not reachable.
Next: 3.

Local Fast Failover

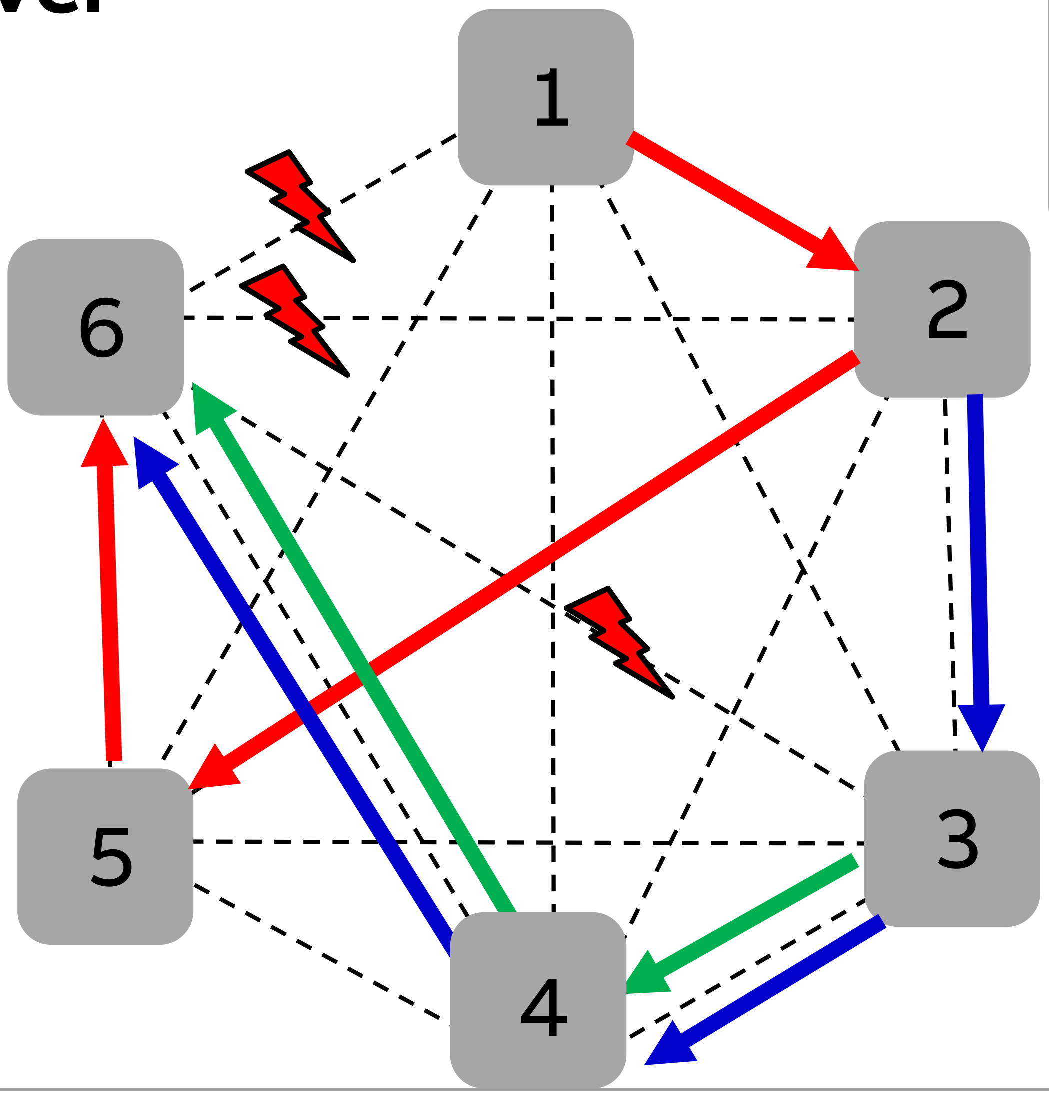
Traffic demand:
{1,2,3}->6



Failover matrix:
flow 1->6: 2,3,4,5,…
flow 2->6: 3,4,5,…
flow 3->6: 4,5,…

Max load:
3 ☹

Local Fast Failover



Statically defined, no global knowledge and no communication!

Failover matrix:

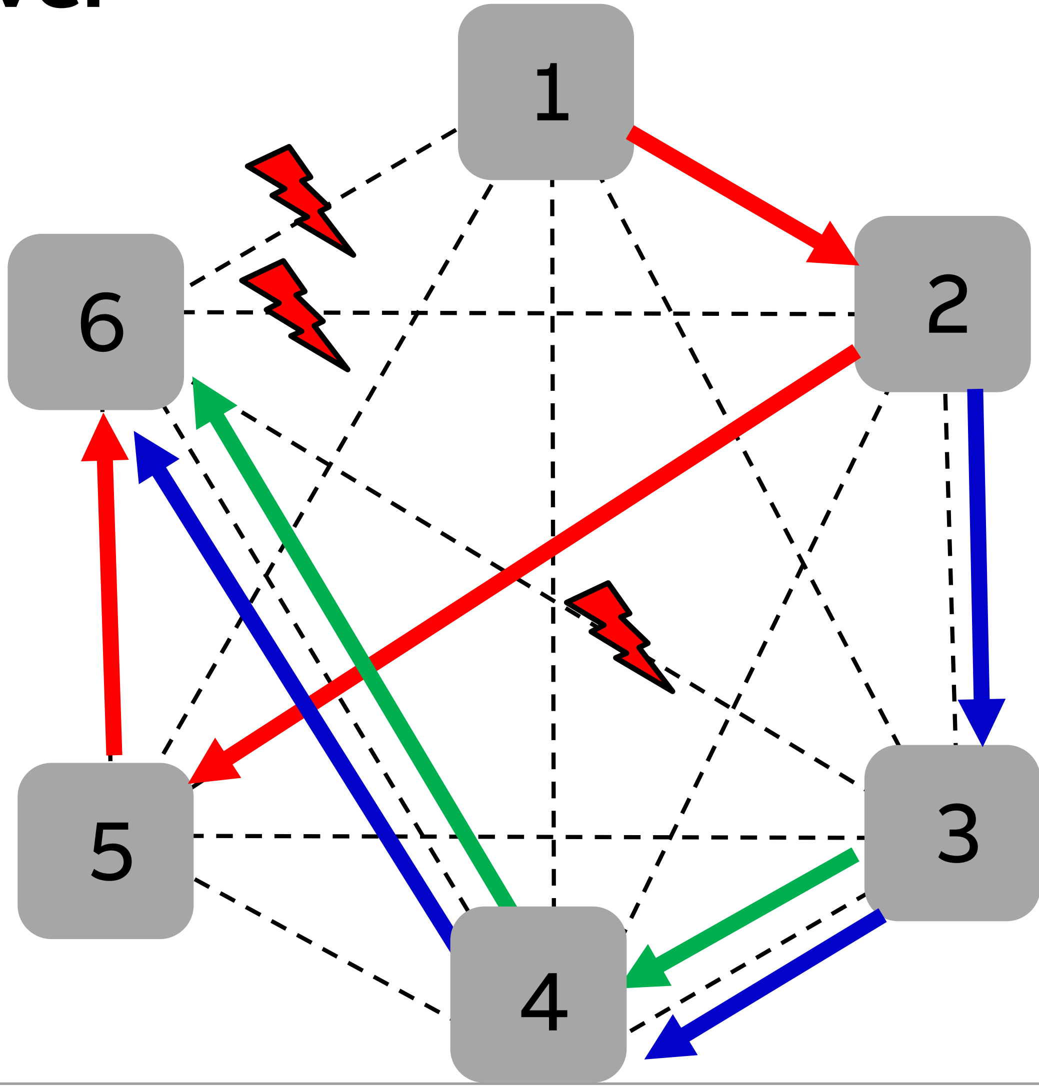
flow 1->6: 2,5, …

flow 2->6: 3,4,5,…

flow 3->6: 4,5,…

A better solution: load 2 ☺

Local Fast Failover



For load balance the prefixes should differ

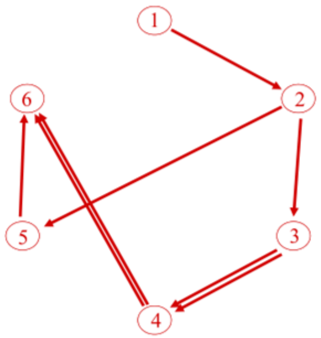
Failover matrix:
flow 1->6: 2,5, …
flow 2->6: 3,4,5,…
flow 3->6: 4,5,…

A better solution: load 2 ☺

Problem statement

Find a failover matrix M that needs many link failures for a high load

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 5 & 1 & 3 & 4 & 6 \\ 3 & 4 & 5 & 1 & 2 & 6 \\ 4 & 1 & 2 & 5 & 3 & 6 \\ 5 & 3 & 4 & 2 & 1 & 6 \\ 5 & 1 & 2 & 3 & 4 & 6 \end{bmatrix}$$



Row i used for flow i, each row is a permutation, source and destination are ignored

```
Upon receiving a packet of flow i at node v
If v != destination:
      If (v,destination) available: forward to d
j = index of v in ith row,      /*m_i,j = v*/
While m_i,j = source or (v,m_i,j) unavailable
      j = j+1
Forward to m_i,j
```

Good and bad news [BS,Opodis 2013]

Lower bound:

High load unavoidable even in well-connected residual networks:
 # failures φ can lead to load at least √φ, even in highly connected networks

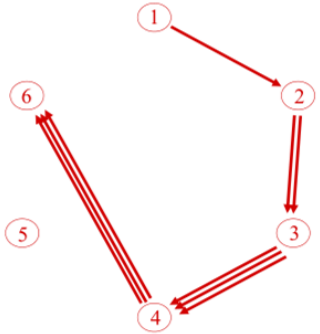
Example: All-to-One Traffic

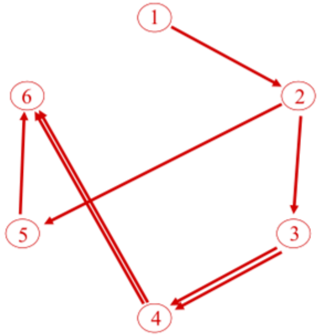
Upper bound:

Load √φ generated with a failover matrix where each row is a random permutation needs at least Omega(φ /log n) failures.

Deterministic Failover Matrices

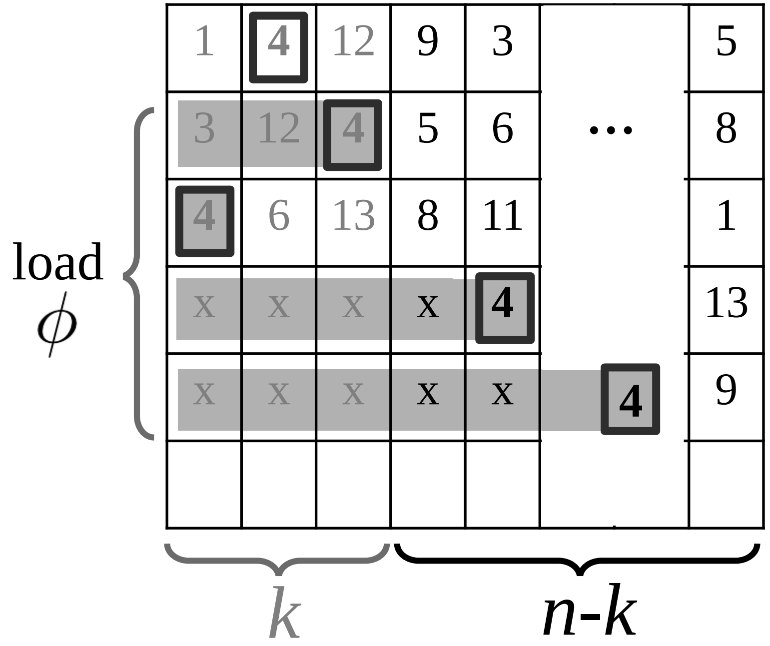
$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 3 & 4 & 5 & 1 & 6 \\ 3 & 4 & 5 & 1 & 2 & 6 \\ 4 & 5 & 1 & 2 & 3 & 6 \\ 5 & 1 & 2 & 3 & 4 & 6 \\ 5 & 1 & 2 & 3 & 4 & 6 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 5 & 1 & 3 & 4 & 6 \\ 3 & 4 & 5 & 1 & 2 & 6 \\ 4 & 1 & 2 & 5 & 3 & 6 \\ 5 & 3 & 4 & 2 & 1 & 6 \\ 5 & 1 & 2 & 3 & 4 & 6 \end{bmatrix}$$





Construction goal:
Low intersection in short prefixes!

Latin Squares with Low Intersection



If k < √n a latin square failover matrix where the intersection of two k-prefixes is at most 1 has load φ < k with Omega(φ^2).

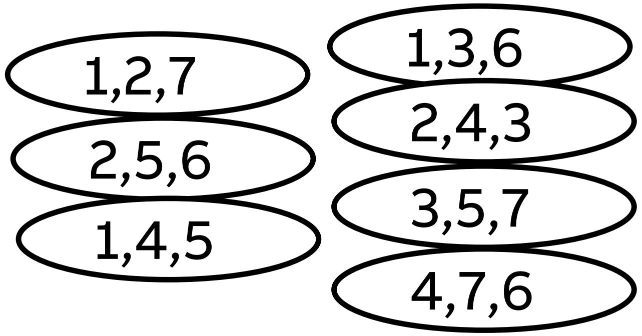
Can we construct such matrices?

Ingredients: Design Theory and Graph Theory

Symmetric Balanced Incomplete Block Designs

An (n, k, λ)-BIBD consists of

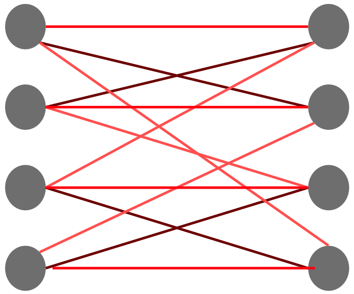
- Set X with n elements $\{1,\ldots,n\}$
- Blocks $A1, \ldots, An$, containing k elements of X
- $|Ai \cap Aj| = \lambda$



(7,3,1)-BIBD

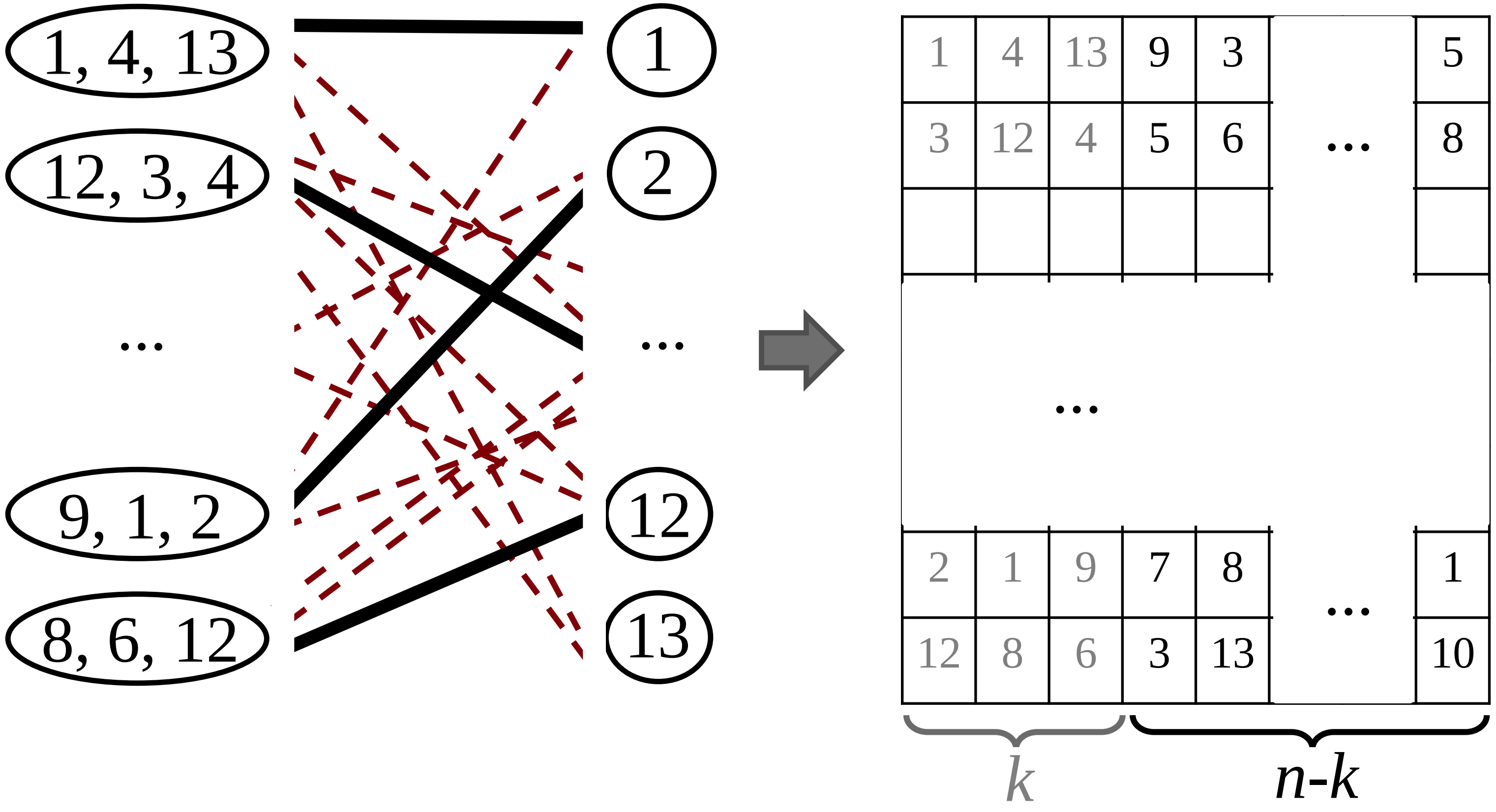
Hall's Marriage Theorem:

A d-regular graph contains d disjoint perfect matchings



3-regular bipartite graph

Construction of k-prefix with low intersection



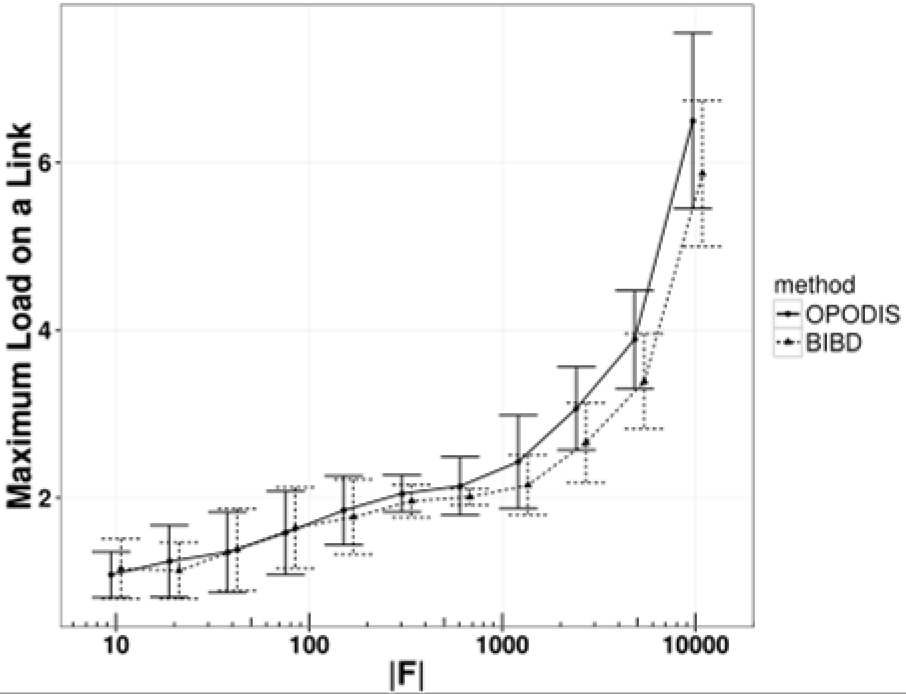
Results

Theory:

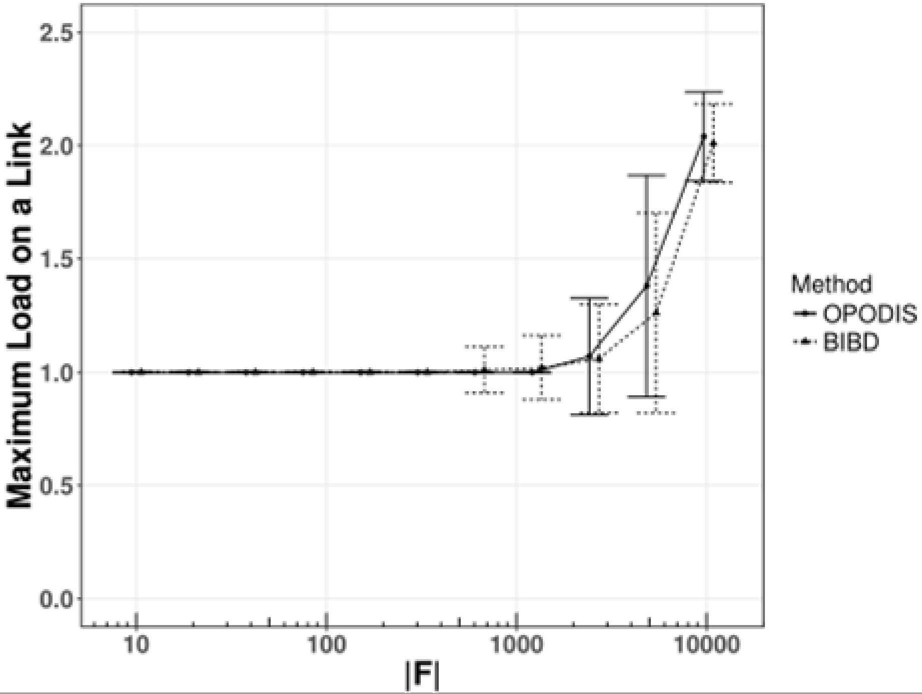
Deterministic BIBD-Failover Matrix achieves asymptotically optimal load

Experiments:

All-to-one routing and random failures



Permutation routing and random failures



Conclusion and Future Work

- Deterministic failover with **load guarantees**

  applying latin squares, BIBDs, matchings

- BIBDs are a tool that can probably be used in many other contexts

Next:

Algorithms and improved bounds for sparse communication networks

ABB