Failure Resilient Routing
Simple Failure Recovery with Load Balancing

Martin Suchara

in collaboration with:
D. Xu, R. Doverspike,
D. Johnson and J. Rexford

Acknowledgement

We gratefully acknowledge the support of the DARPA CORONET Program, Contract N00173-08-C-2011. The views, opinions, and/or findings contained in this article/presentation are those of the author/presenter and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

Special thanks to Olivier Bonaventure, Quynh Nguyen, Kostas Oikonomou, Rakesh Sinha, Robert Tarjan, Kobus van der Merwe, and Jennifer Yates.

Failure Recovery and Traffic Engineering in IP Networks

- Uninterrupted data delivery when links or routers fail

- Failure recovery essential for
  - Backbone network operators
  - Large datacenters
  - Local enterprise networks

- Major goal: re-balance the network load after failure

Overview

I. Failure recovery: *the challenges*

II. Architecture: *goals and proposed design*

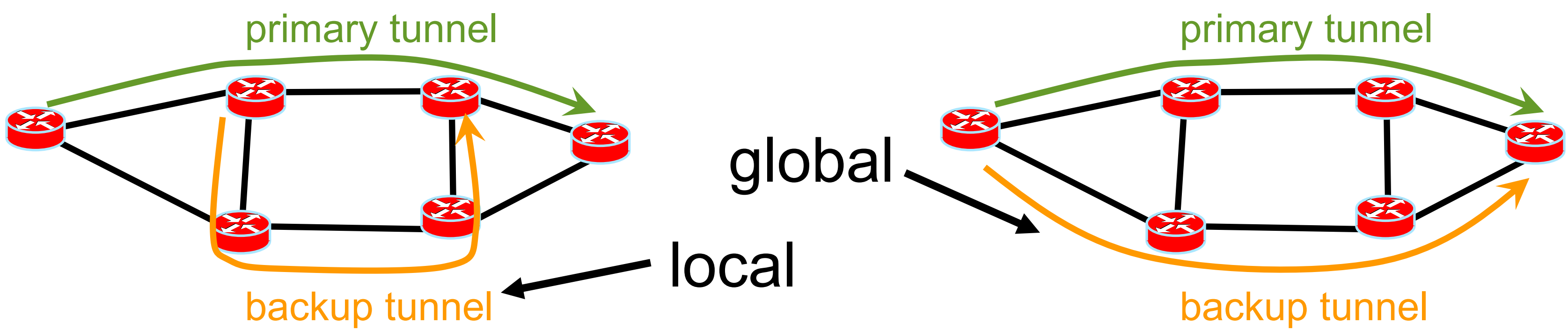
III. Optimizations: *of routing and load balancing*

IV. Evaluation: *using synthetic and realistic topologies*

V. Conclusion

Challenges of Failure Recovery

- Existing solutions reroute traffic to avoid failures
  - Can use, e.g., MPLS local or global protection



- Balance the traffic after rerouting
  - Challenging with local path protection
- Prompt failure detection
  - Global path protection is slow

Overview

I. Failure recovery: *the challenges*

II. Architecture: *goals and proposed design*

III. Optimizations: *of routing and load balancing*

IV. Evaluation: *using synthetic and realistic topologies*

V. Conclusion

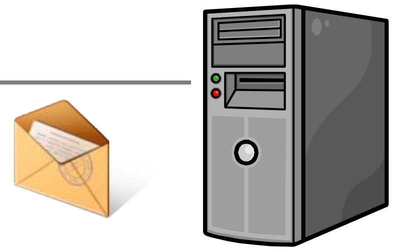
Architectural Goals

1. Simplify the network
   - Allow use of minimalist cheap routers
   - Simplify network management
2. Balance the load
   - Before, during, and after each failure
3. Detect and respond to failures quickly

The Architecture – Components

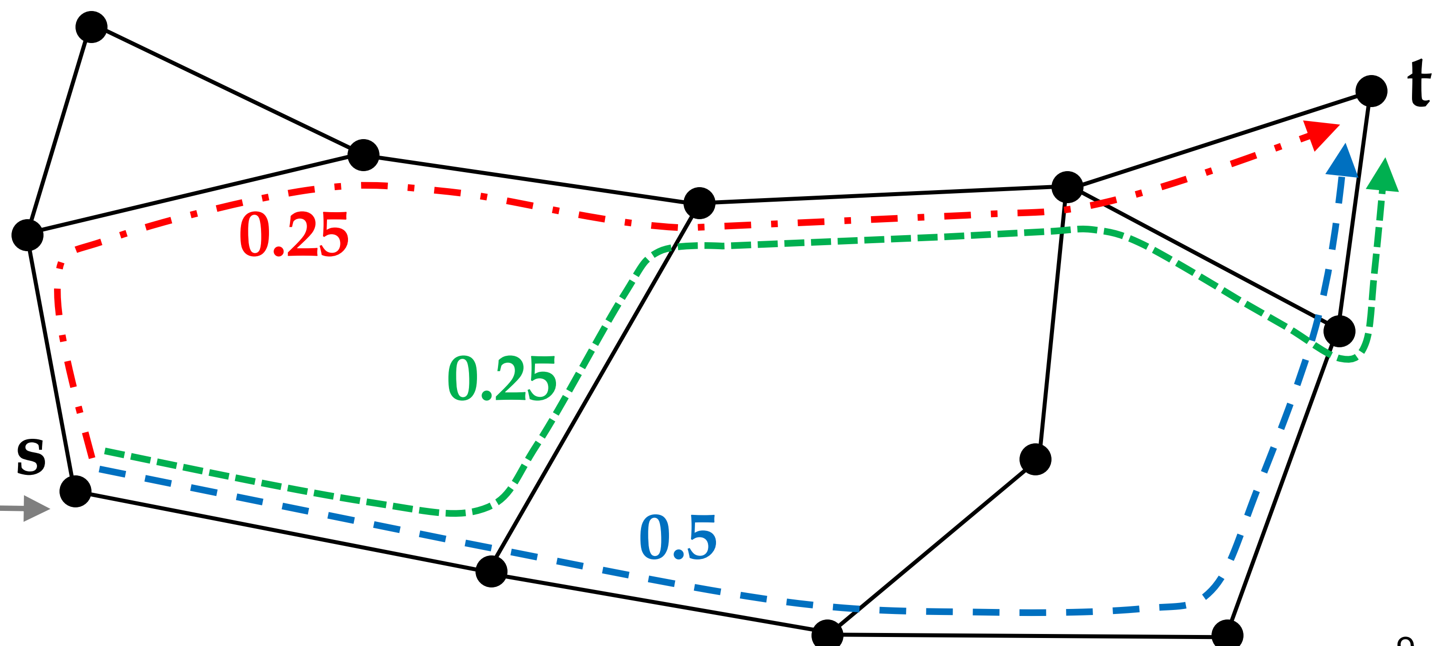
- Minimal functionality in routers
  - Path-level failure notification
  - Static configuration
  - No coordination with other routers
- Management system
  - Knows topology, approximate traffic demands, potential failures
  - Sets up multiple paths and calculates load splitting ratios

The Architecture

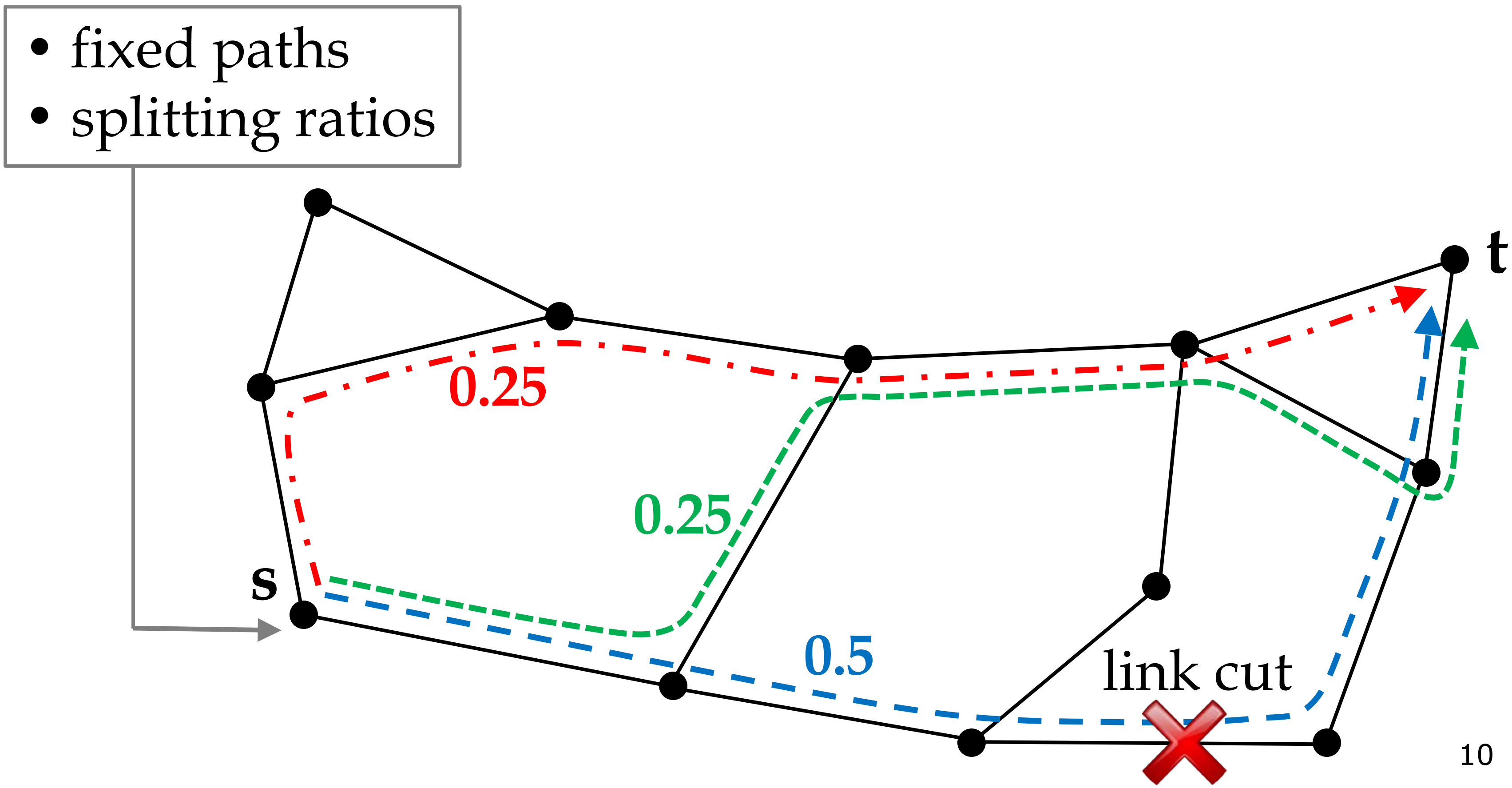


- topology design
- list of shared risks
- traffic demands

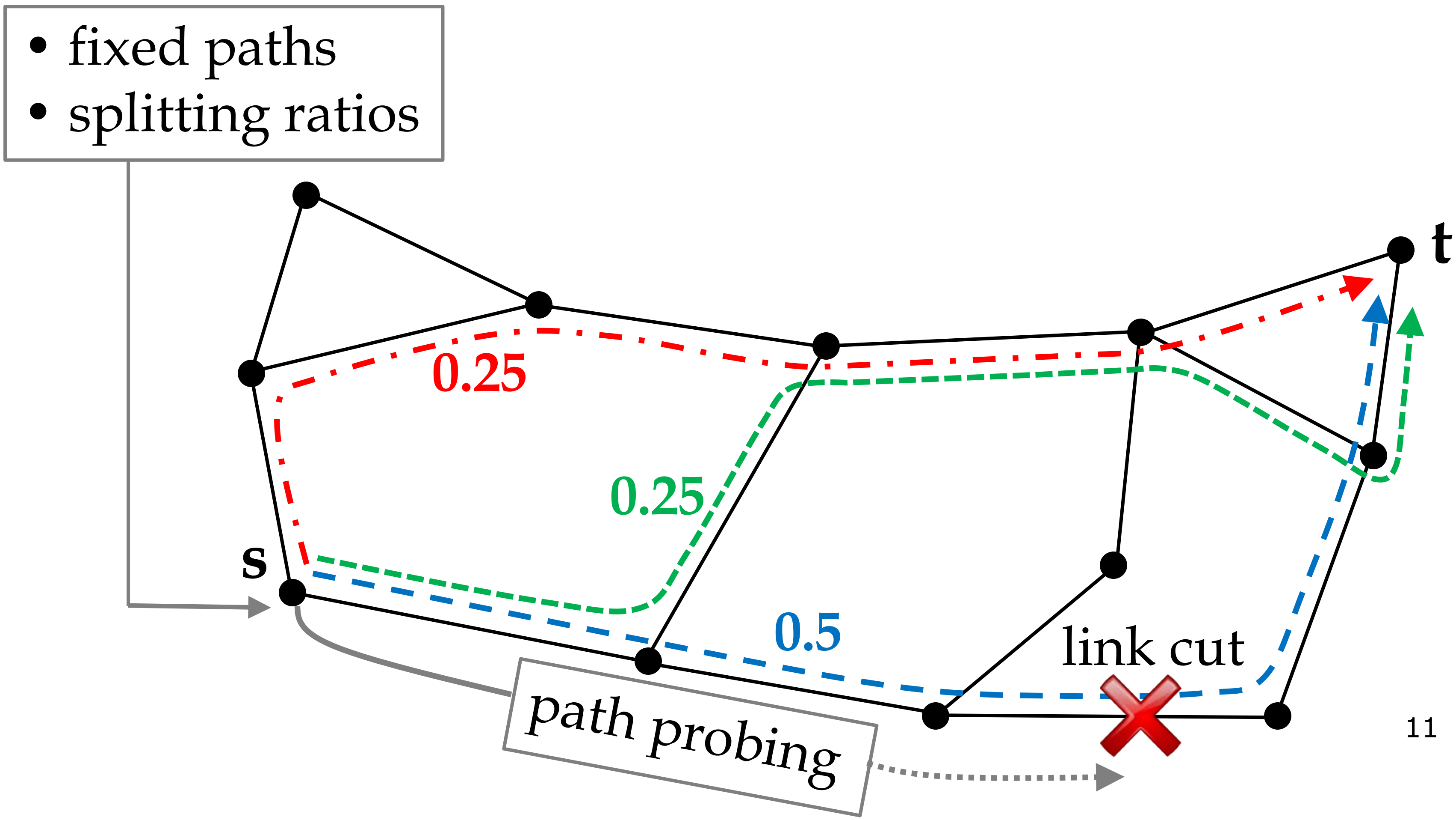
- fixed paths
- splitting ratios



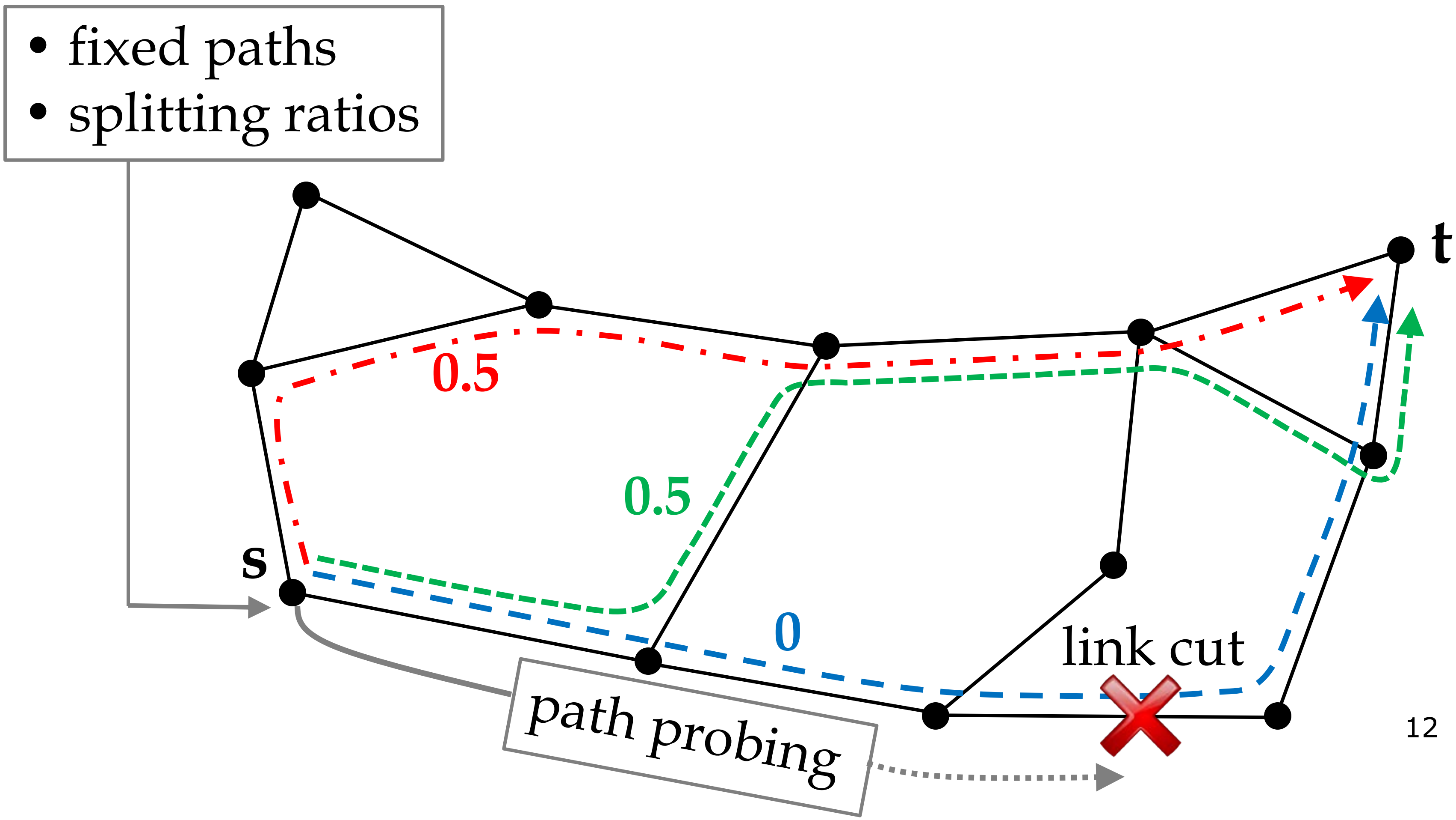
The Architecture



The Architecture



The Architecture



The Architecture: Summary

1. Offline optimizations
2. Load balancing on end-to-end paths
3. Path-level failure detection

How to calculate the paths and splitting ratios?

Overview

I. Failure recovery: *the challenges*

II. Architecture: *goals and proposed design*

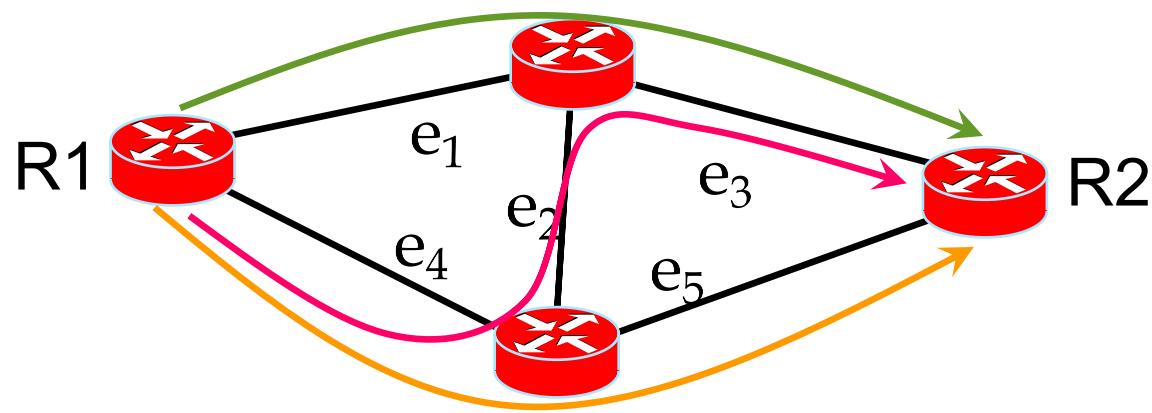
III. Optimizations: *of routing and load balancing*

IV. Evaluation: *using synthetic and realistic topologies*

V. Conclusion

Goal I: Find Paths Resilient to Failures

- A working path needed for each allowed failure state (shared risk link group)



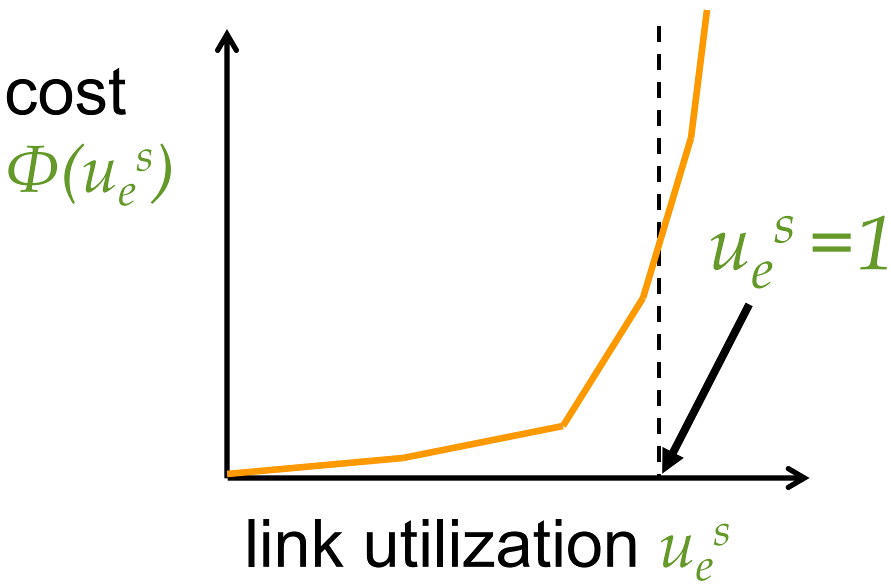
Example of failure states:

$S = \{e_1\}, \{ e_2\}, \{ e_3\}, \{ e_4\}, \{ e_5\}, \{e_1, e_2\}, \{e_1, e_5\}$

Goal II: Minimize Link Loads

links indexed by e

failure states indexed by s



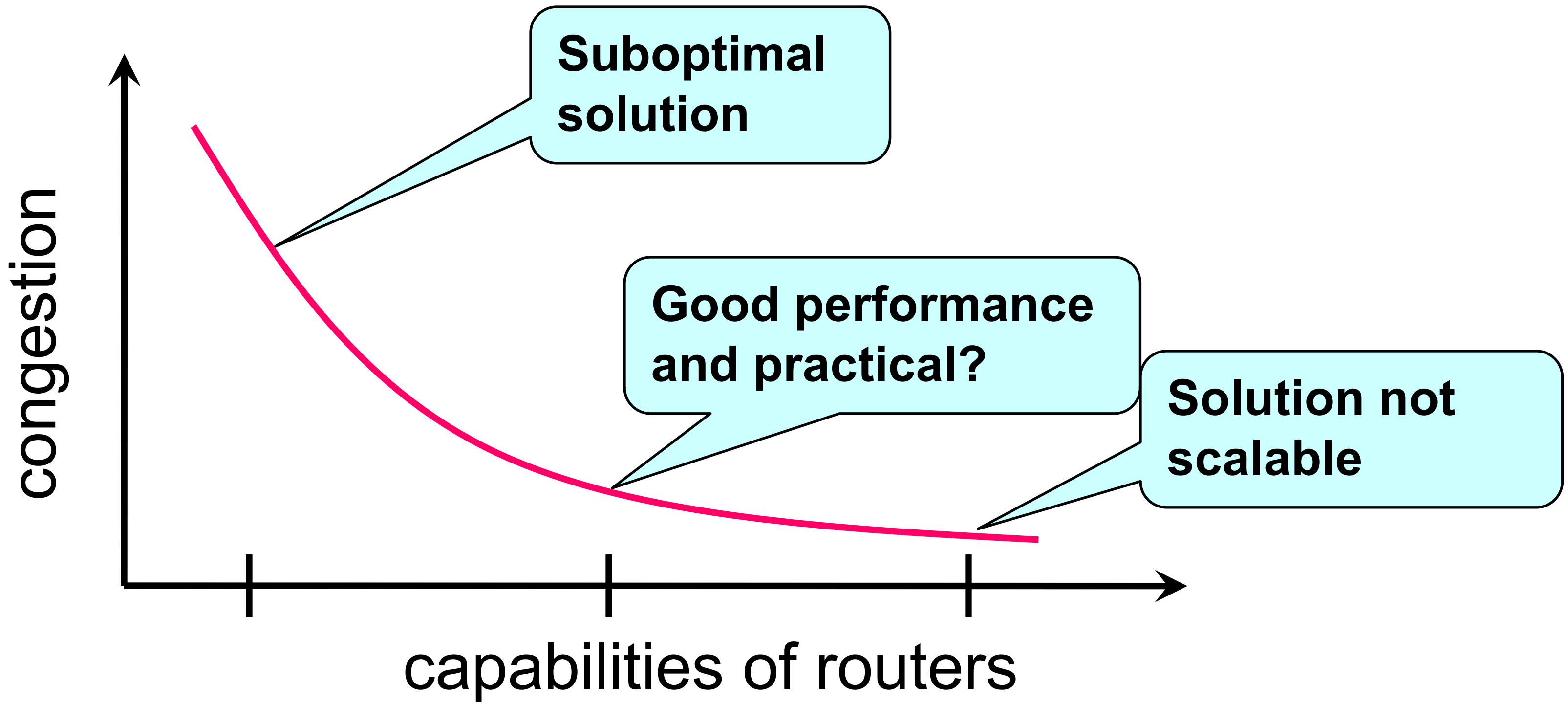
aggregate congestion cost weighted for all failures:

minimize $\sum_s w^s \sum_e \Phi(u_e^s)$ while routing all traffic

failure state weight

Cost function is a penalty for approaching capacity

Our Three Solutions



- Too simple solutions do not do well
- Diminishing returns when adding functionality

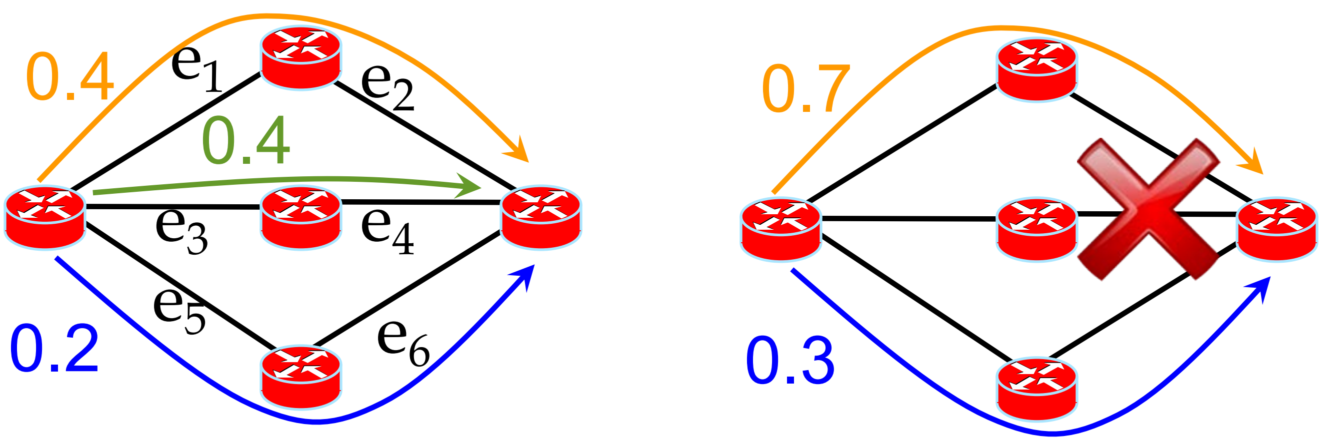
1. Optimal Solution: State Per Network Failure

- Edge router must learn which *links* failed
- Custom splitting ratios for each failure

configuration:

Failure	Splitting Ratios
-	0.4, 0.4, 0.2
e_4	0.7, 0, 0.3
e_1 & e_2	0, 0.6, 0.4
…	…

one entry per failure



1. Optimal Solution: State Per Network Failure

- Solve a classical multicommodity flow for each failure case s:

  min *load balancing objective*
  s.t. *flow conservation*
  demand satisfaction
  edge flow non-negativity

- Decompose edge flow into paths and splitting ratios
- Does not scale with number of potential failure states

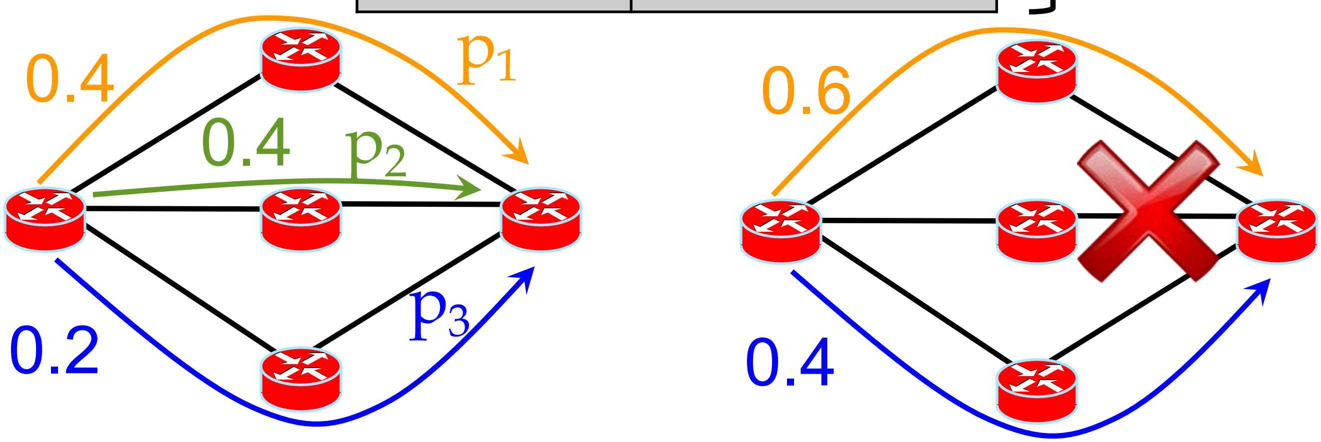
2. State-Dependent Splitting: Per Observable Failure

- Edge router observes which *paths* failed
- Custom splitting ratios for each observed combination of failed paths
- NP-hard unless paths are fixed

configuration:

Failure	Splitting Ratios
-	0.4, 0.4, 0.2
p2	0.6, 0, 0.4
…	…

at most $2^{\#paths}$ entries



2. State-Dependent Splitting: Per Observable Failure

- Heuristic: use the same paths as the optimal solution

- If paths fixed, can find optimal splitting ratios:

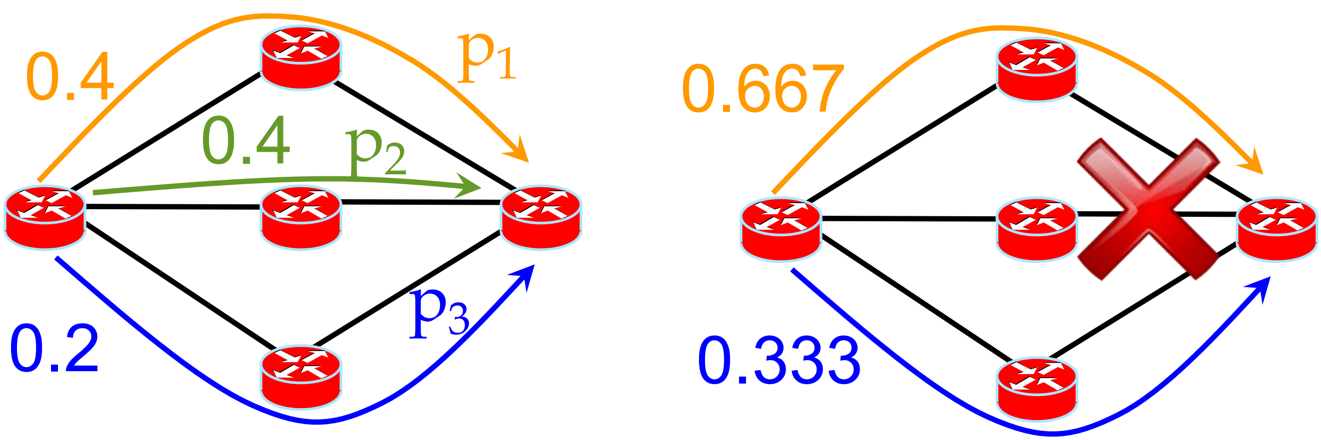
min *load balancing objective*
s.t. *flow conservation*
demand satisfaction
path flow non-negativity

3. State-Independent Splitting: Across All Failure Scenarios

- Edge router observes which *paths* failed
- Fixed splitting ratios for all observable failures
- Non-convex optimization even with fixed paths

configuration:

p1, p2, p3:
0.4, 0.4, 0.2



3. State-Independent Splitting: Across All Failure Scenarios

- Heuristic to compute paths
  - The same paths as the optimal solution
- Heuristic to compute splitting ratios
  - Use averages of the optimal solution weighted by all failure case weights

$$r_i = \sum_s w^s r_i^s$$

fraction of traffic on the i-th path

Our Three Solutions

1. Optimal solution
2. State-dependent splitting
3. State-independent splitting

How well do they work in practice?

Overview

I. Failure recovery: *the challenges*

II. Architecture: *goals and proposed design*

III. Optimizations: *of routing and load balancing*

IV. Evaluation: *using synthetic and realistic topologies*

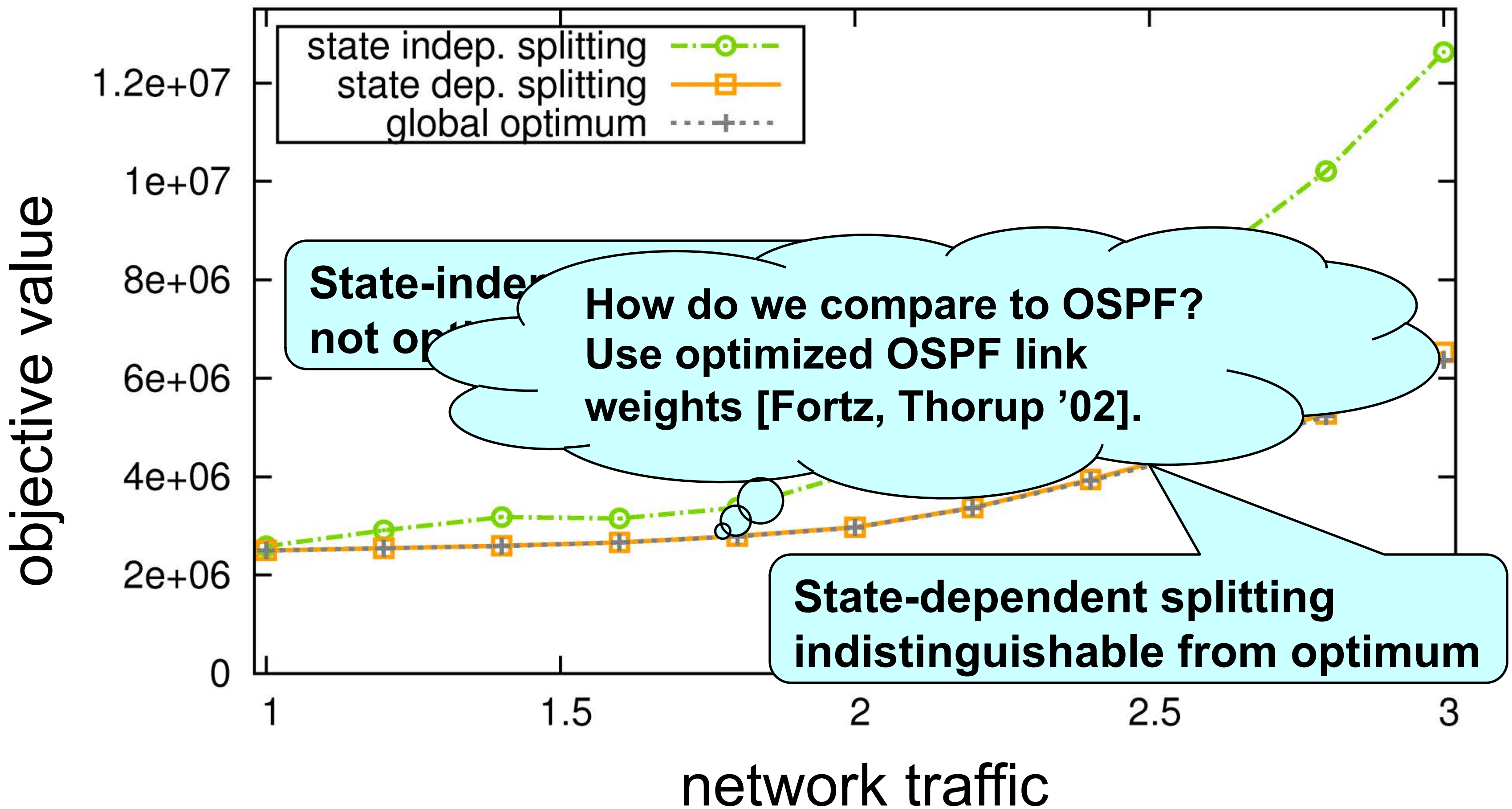
V. Conclusion

Simulations on a Range of Topologies

Topology	Nodes	Edges	Demands
Tier-1	50	180	625
Abilene	11	28	253
Hierarchical	50	148 - 212	2,450
Random	50 - 100	228 - 403	2,450 – 9,900
Waxman	50	169 - 230	2,450

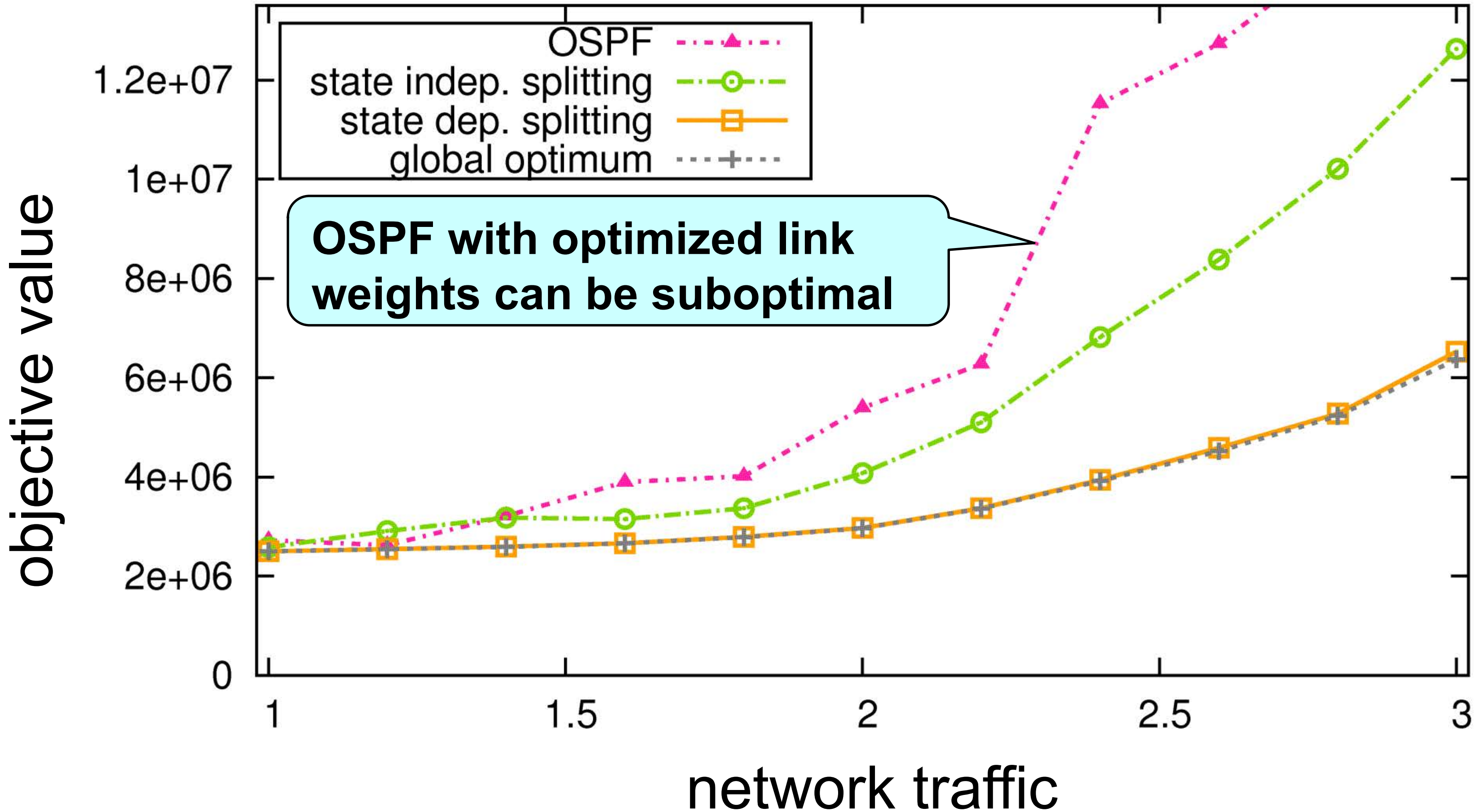
- Shared risk failures for the tier-1 topology
  - 954 failures, up to 20 links simultaneously
- Single link failures

Congestion Cost – Tier-1 IP Backbone with SRLG Failures



Additional router capabilities improve performance up to a point

Congestion Cost – Tier-1 IP Backbone with SRLG Failures



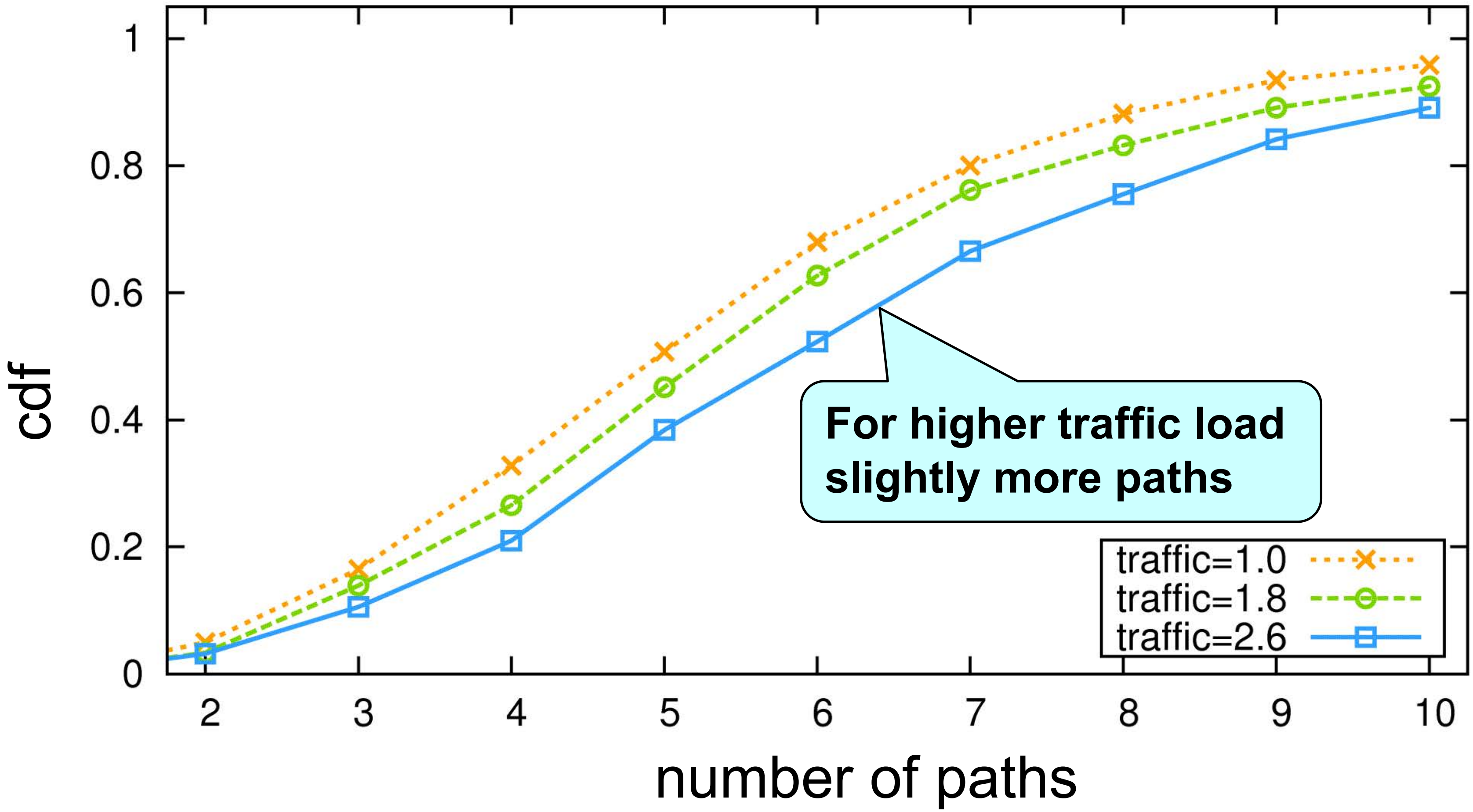
OSPF uses equal splitting on shortest paths. This restriction makes the performance worse.

Average Traffic Propagation Delay in Tier-1 Backbone

- Service Level Agreements guarantee 37 ms mean traffic propagation delay
- Need to ensure mean delay doesn’t increase much

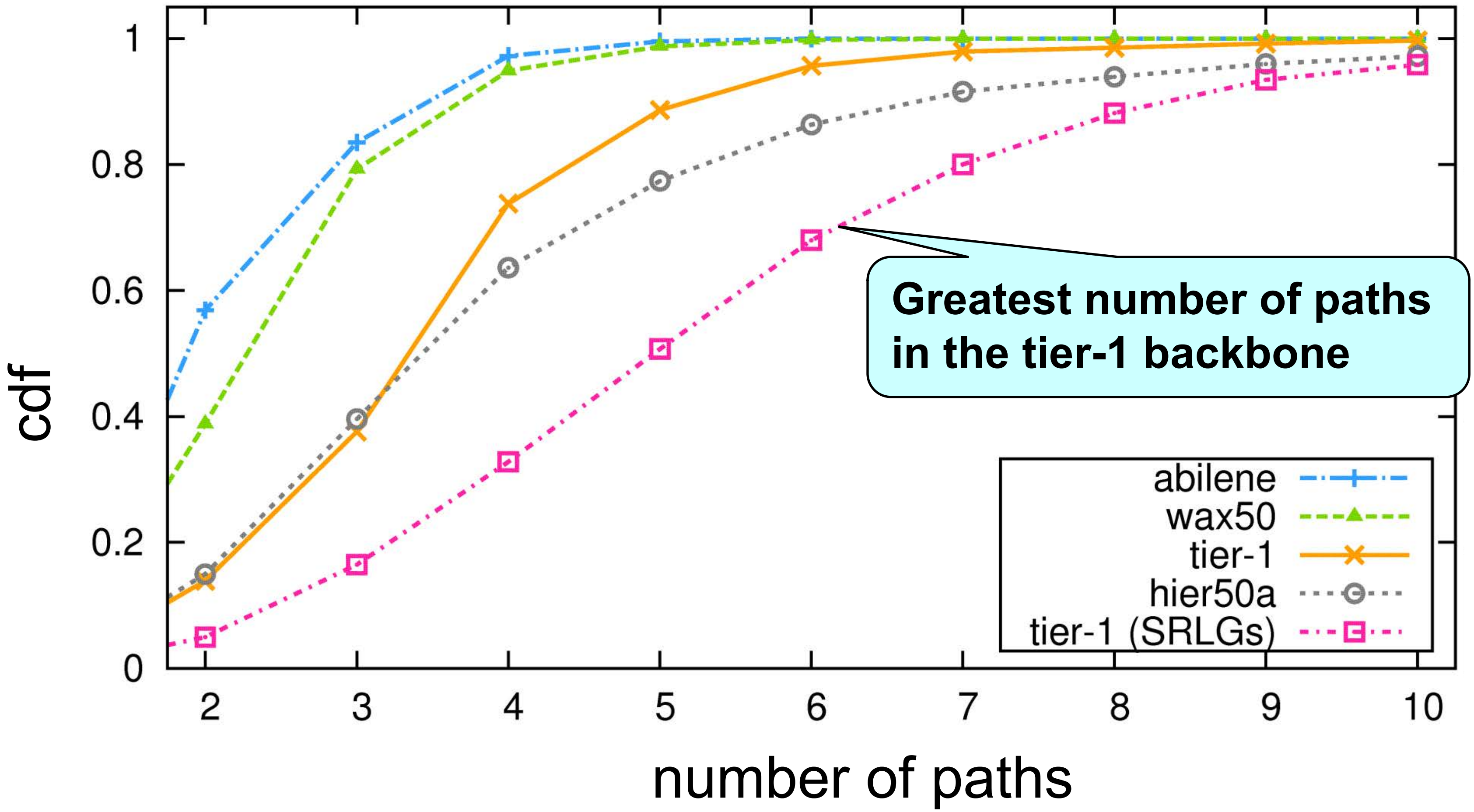
Algorithm	Delay (ms)	Stdev
OSPF (current)	28.49	0.00
Optimum	31.03	0.22
State dep. splitting	30.96	0.17
State indep. splitting	31.11	0.22

Number of Paths– Tier-1 IP Backbone with SRLG Failures



Number of paths almost independent of the load

Number of Paths – Various Topologies



More paths for larger and more diverse topologies

Overview

I. Failure recovery: *the challenges*

II. Architecture: *goals and proposed design*

III. Optimizations: *of routing and load balancing*

IV. Evaluation: *using synthetic and realistic topologies*

V. Conclusion

Conclusion

- Simple mechanism combining path protection and traffic engineering
- Favorable properties of state-dependent splitting algorithm:

(i)	Simplifies network design
(ii)	Near optimal load balancing
(iii)	Small number of paths
(iv)	Delay comparable to current OSPF

- Path-level failure information is just as good as complete failure information

Thank You!