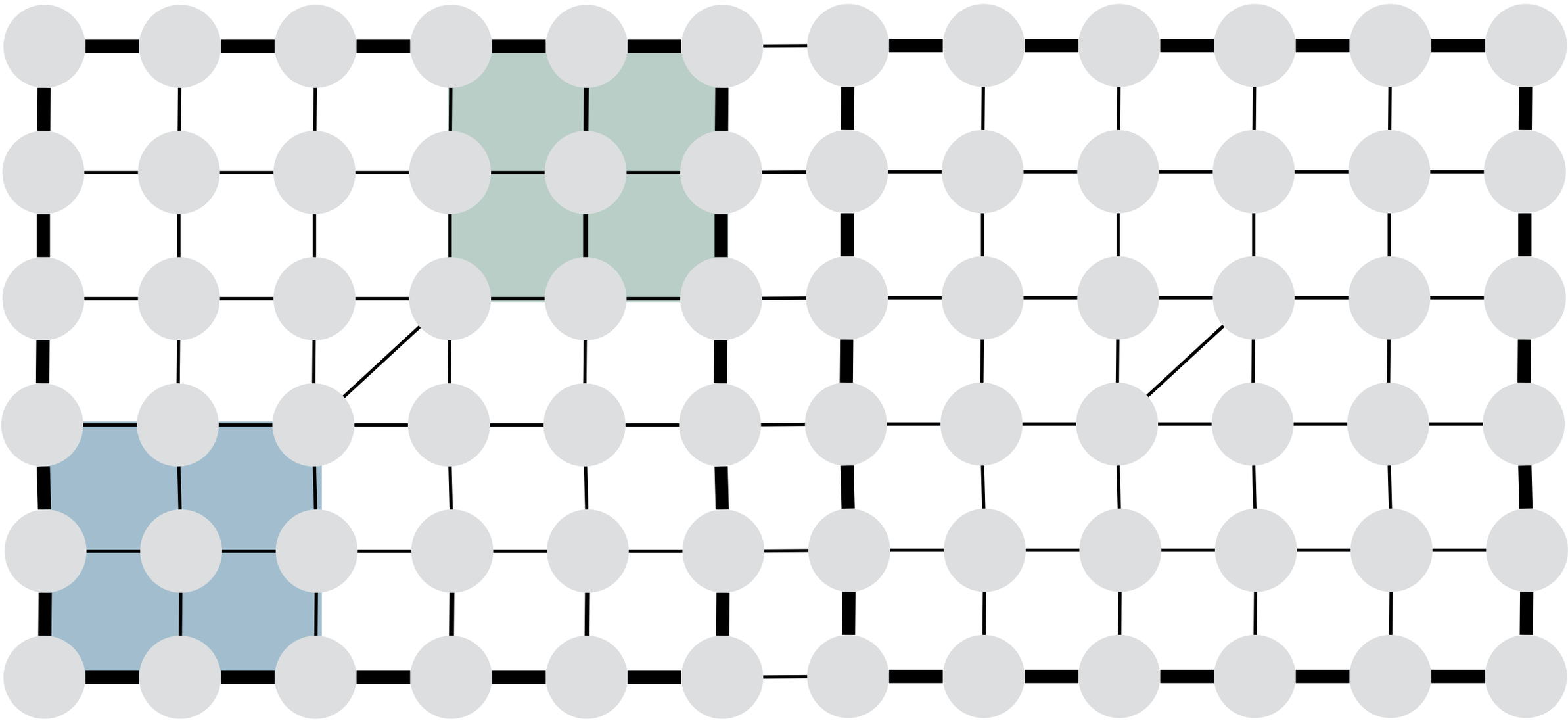
KULFI

Robust Traffic Engineering Using Semi-Oblivious Routing

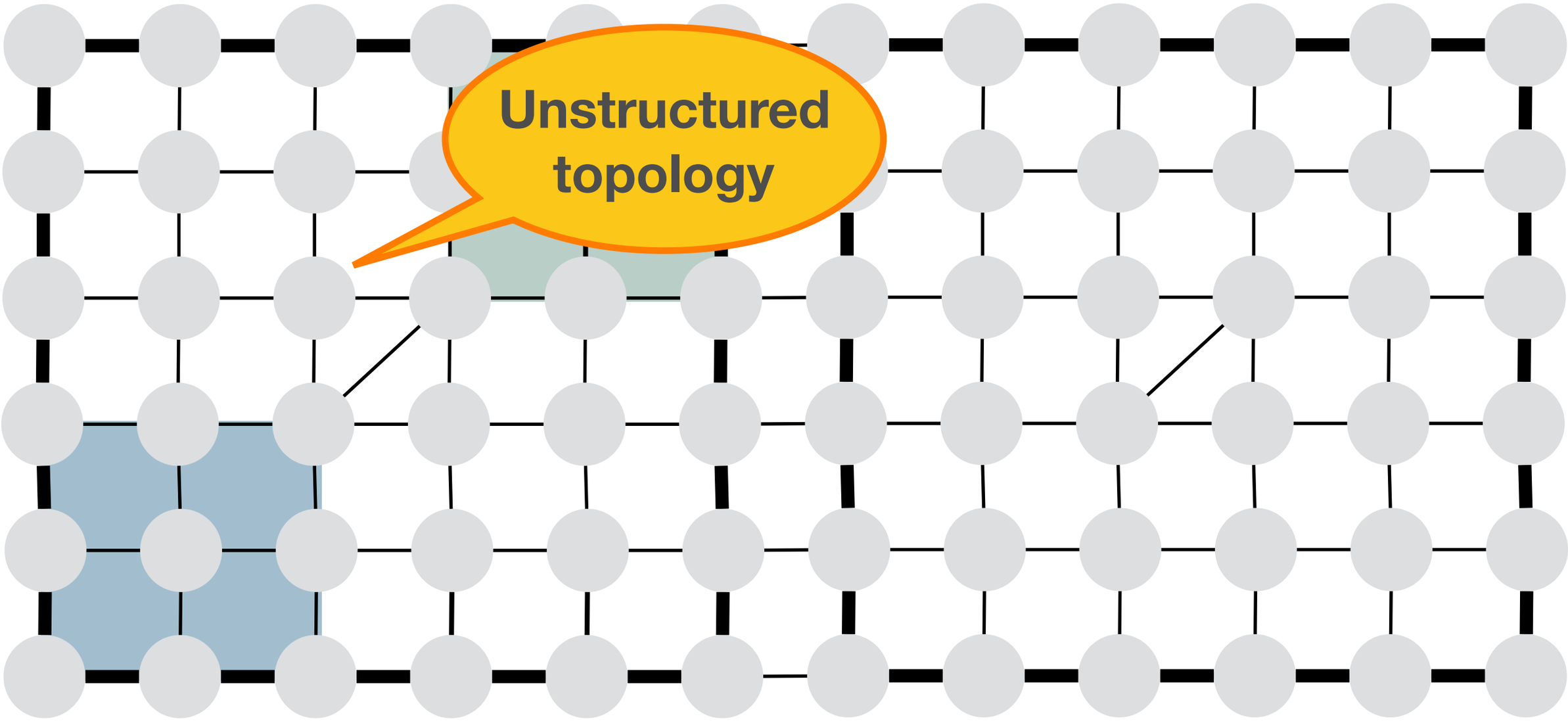
Praveen Kumar, Yang Yuan, Chris Yu,
Bobby Kleinberg, Robert Soulé, & Nate Foster

Cornell, Carnegie Mellon, Microsoft Research, & USI Lugano

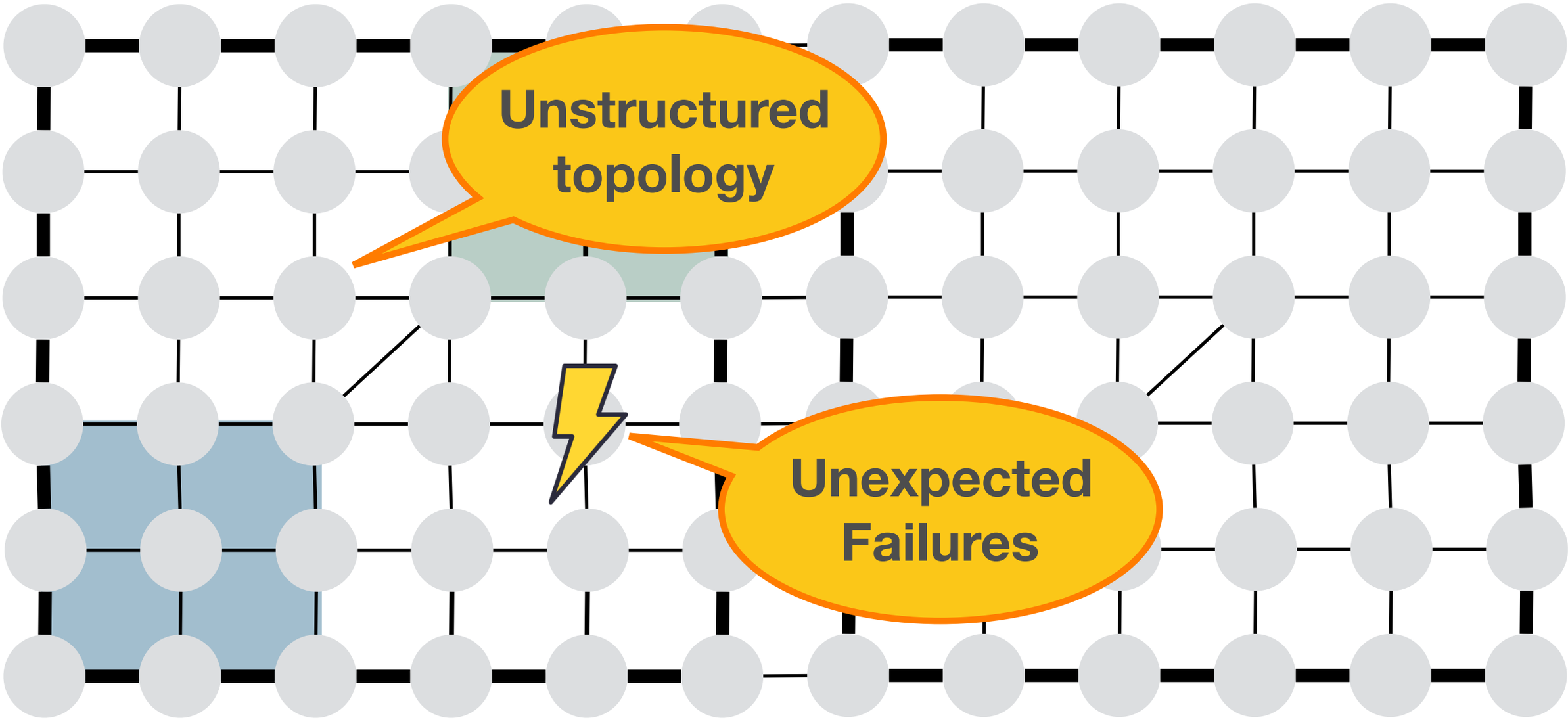
WIDE-AREA TRAFFIC ENGINEERING CHALLENGES



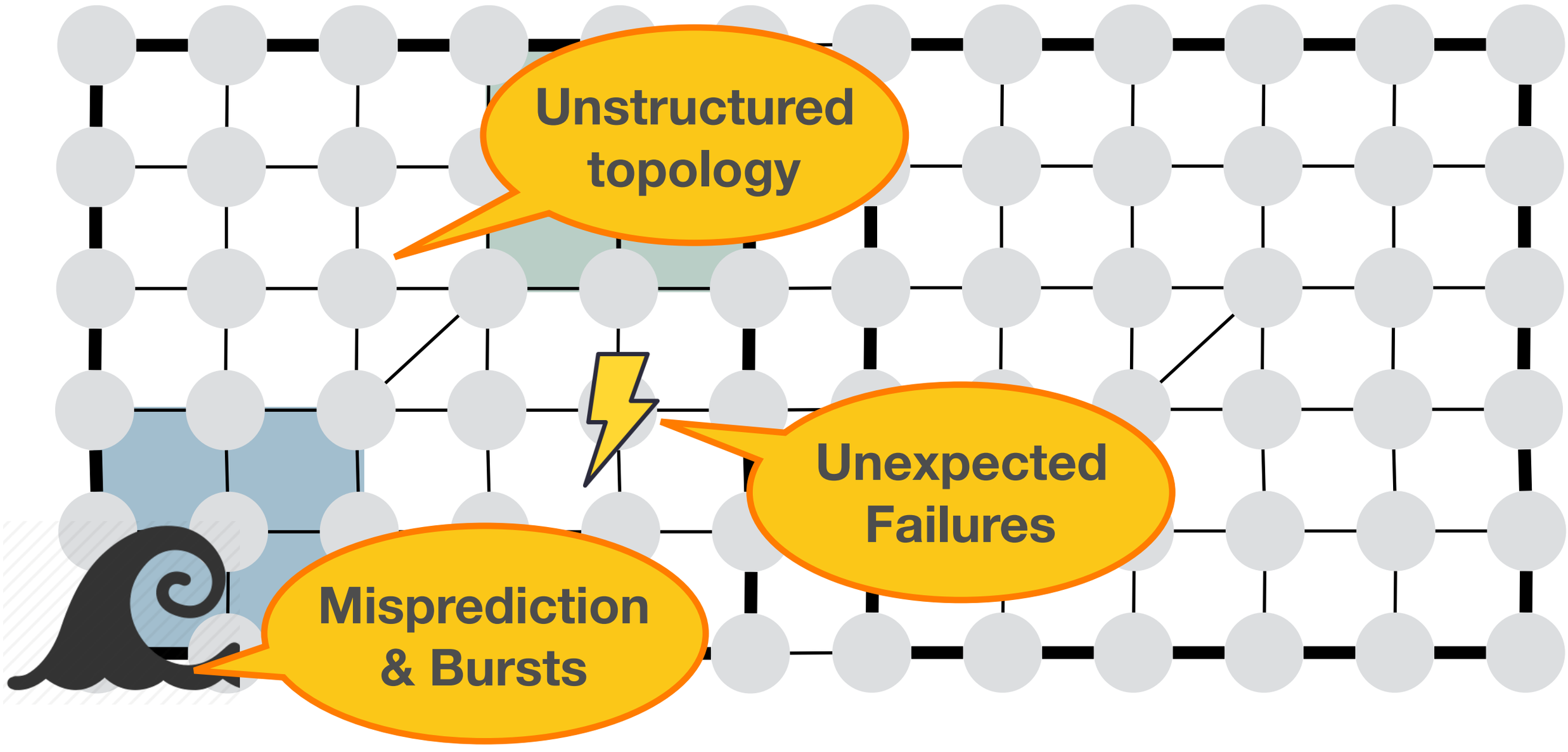
WIDE-AREA TRAFFIC ENGINEERING CHALLENGES



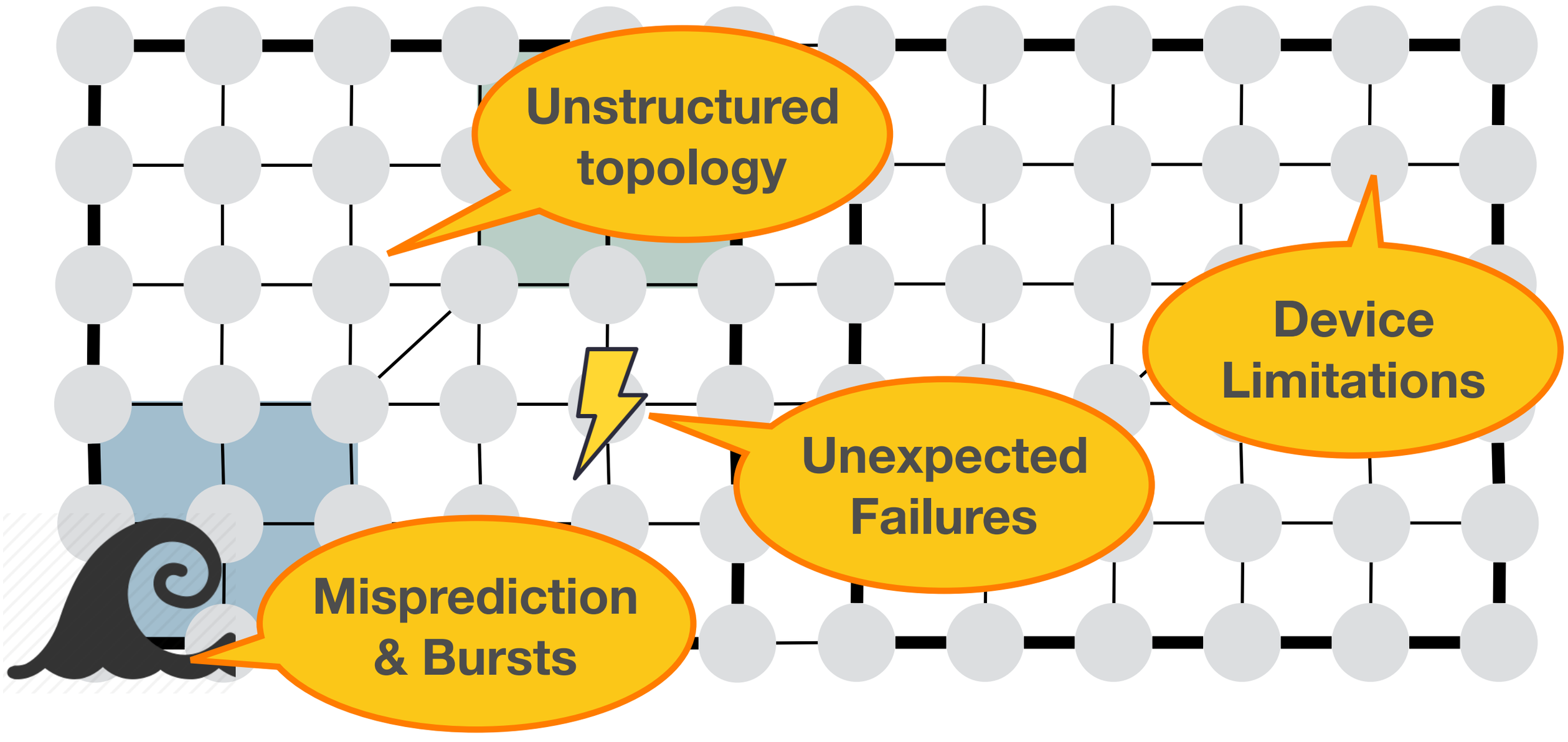
WIDE-AREA TRAFFIC ENGINEERING CHALLENGES



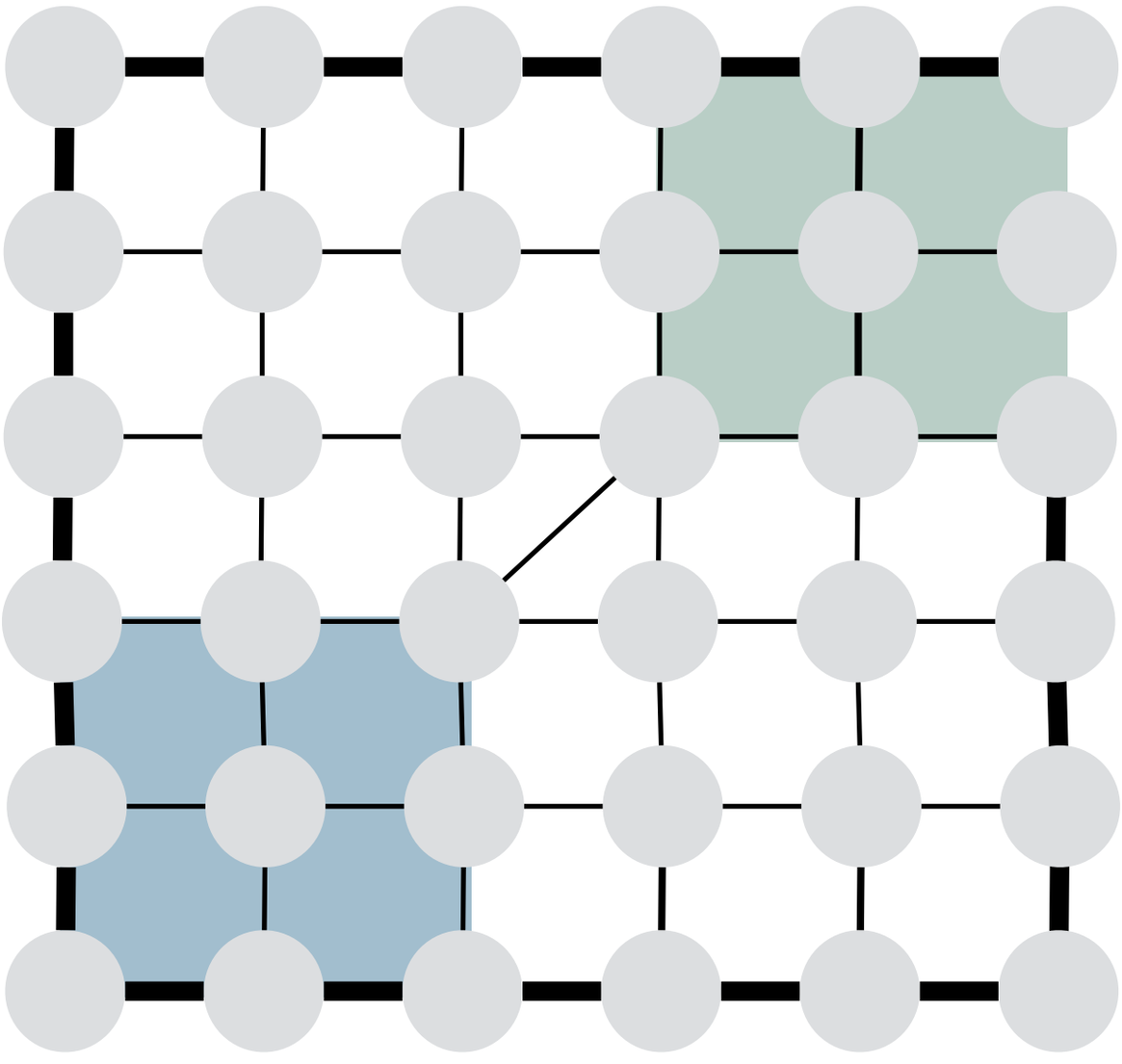
WIDE-AREA TRAFFIC ENGINEERING CHALLENGES



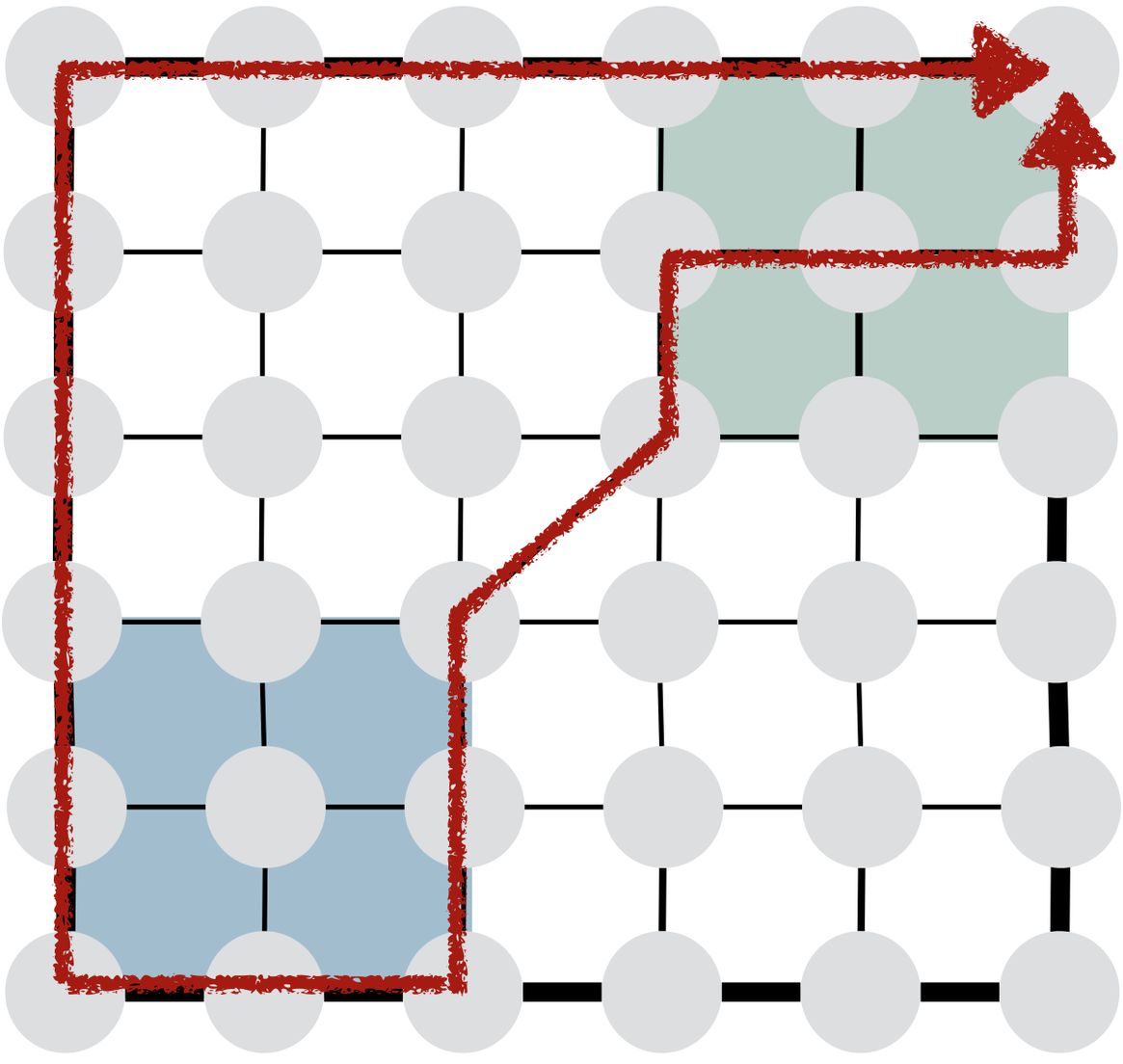
WIDE-AREA TRAFFIC ENGINEERING CHALLENGES



ROUTING SCHEME

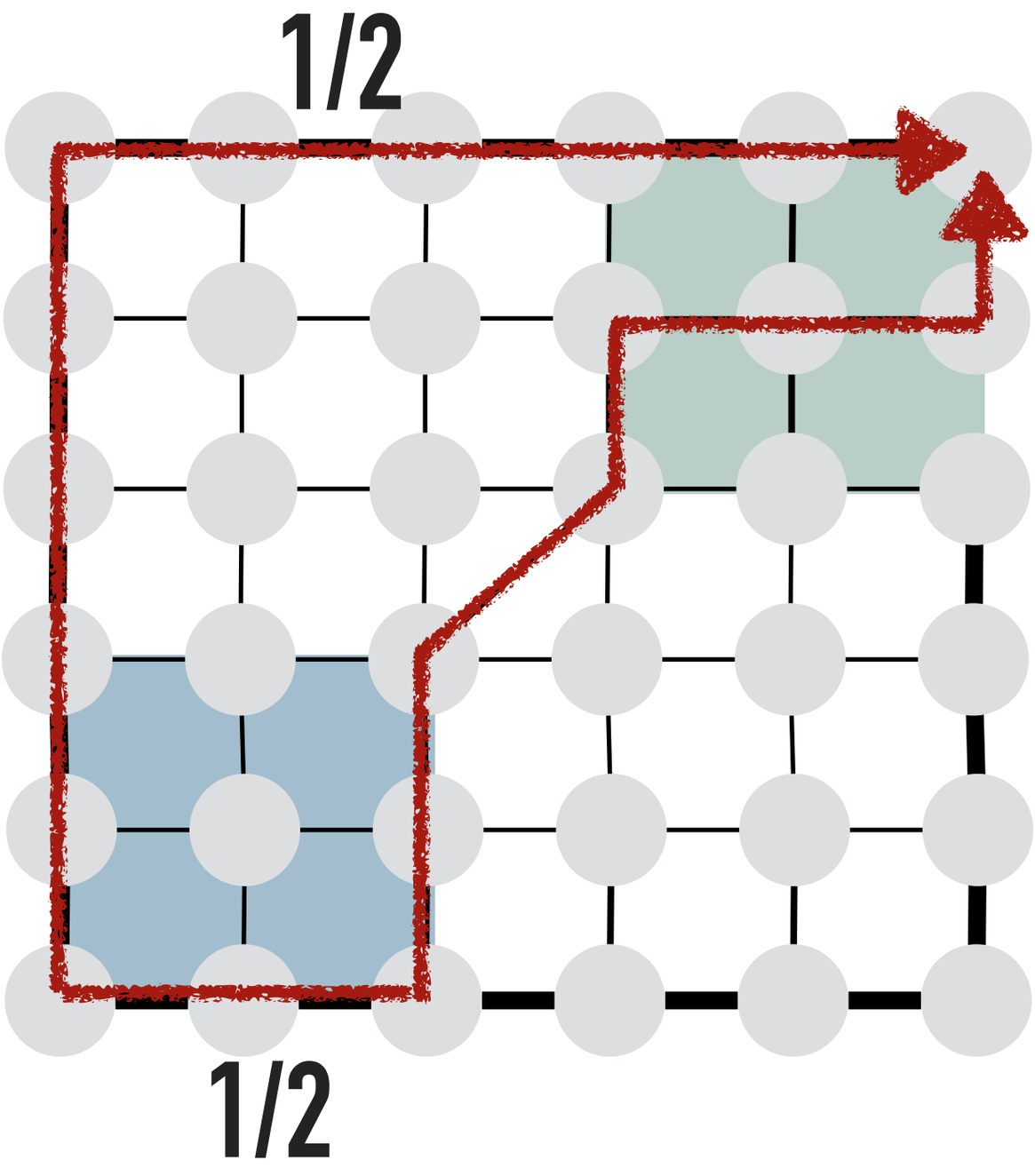


ROUTING SCHEME



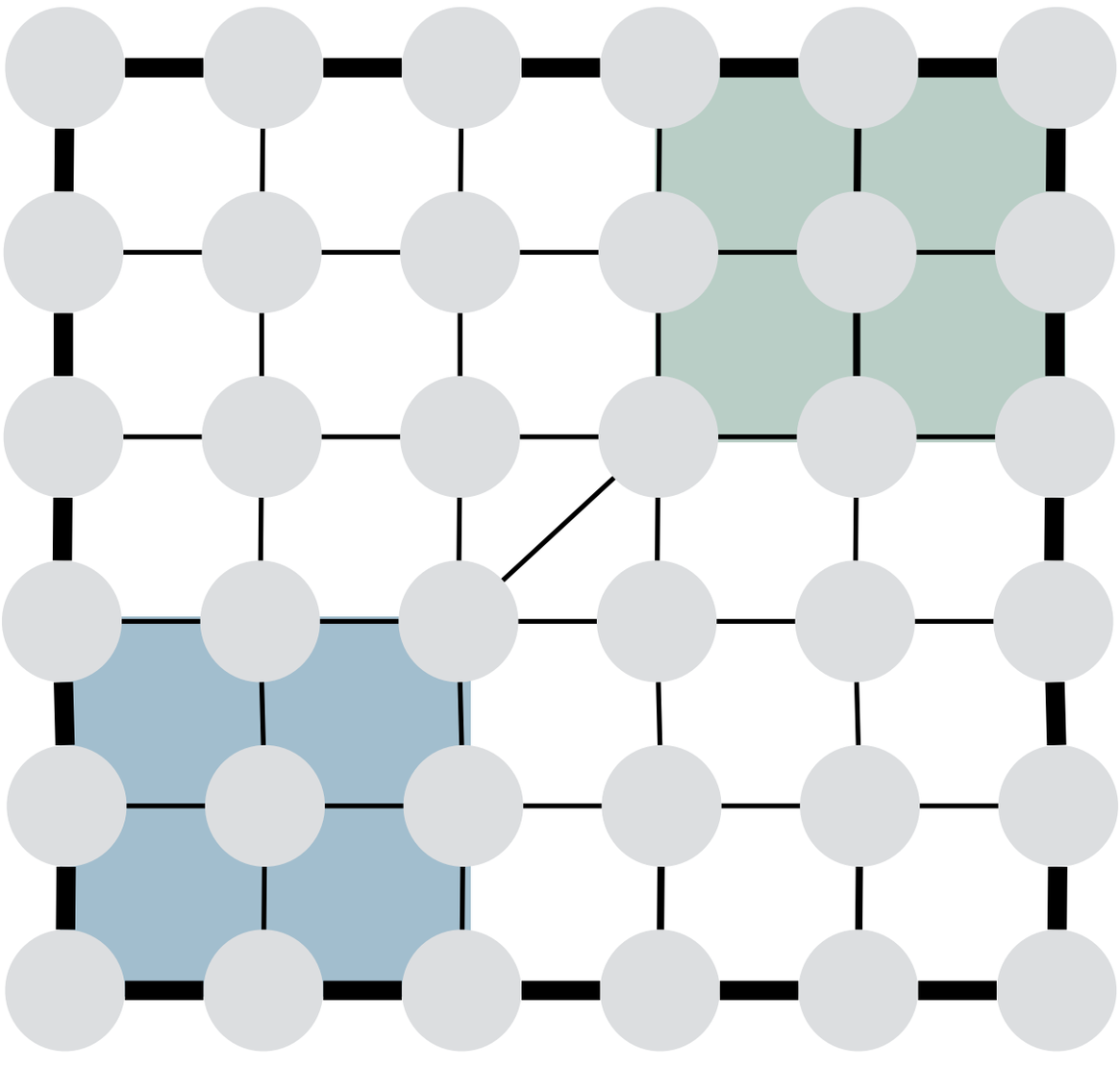
1. Which forwarding paths to use send traffic from sources to destinations?

ROUTING SCHEME



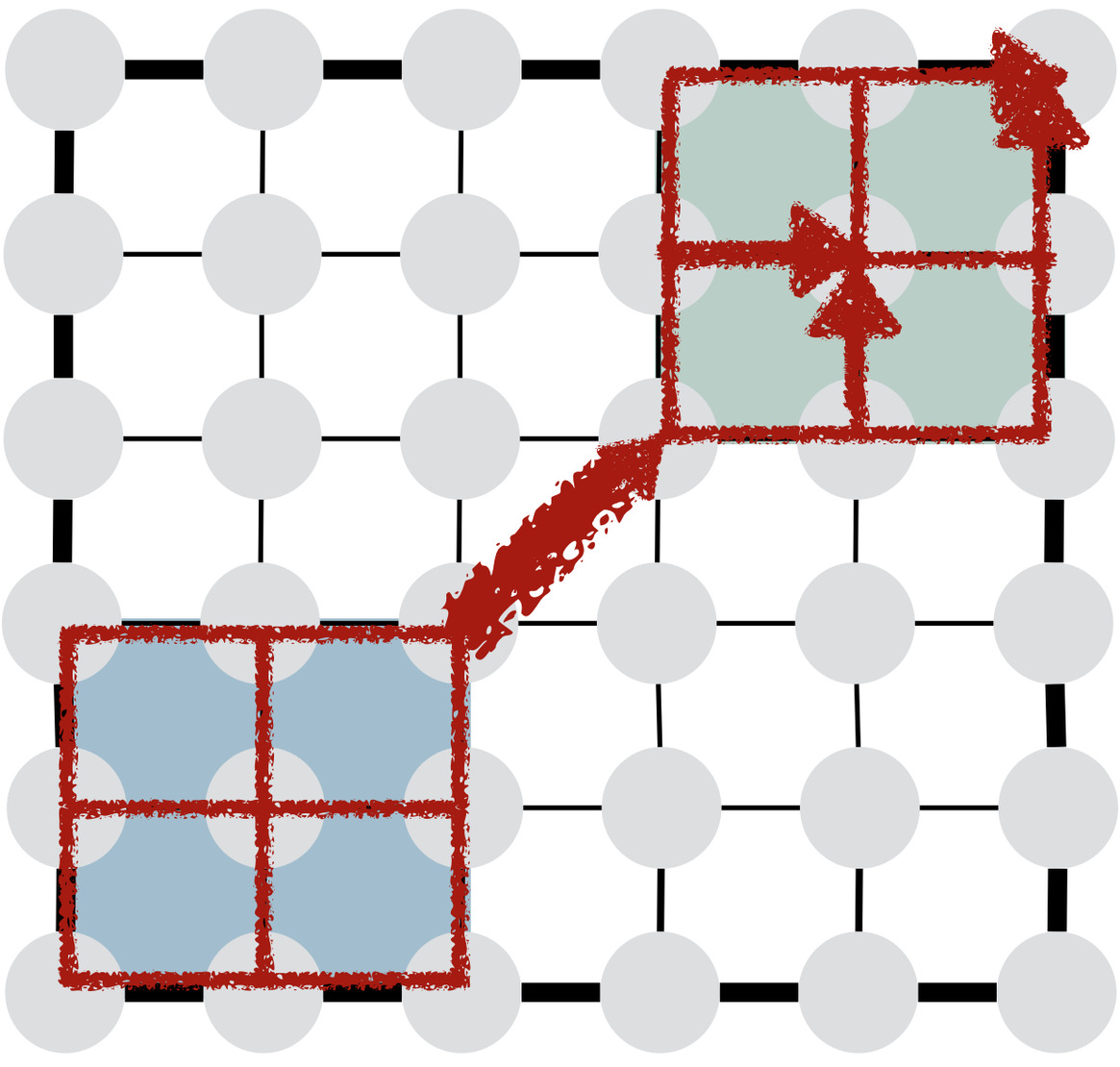
1. Which forwarding paths to use send traffic from sources to destinations?
2. How to map incoming traffic flows onto multiple forwarding paths?

ECMP



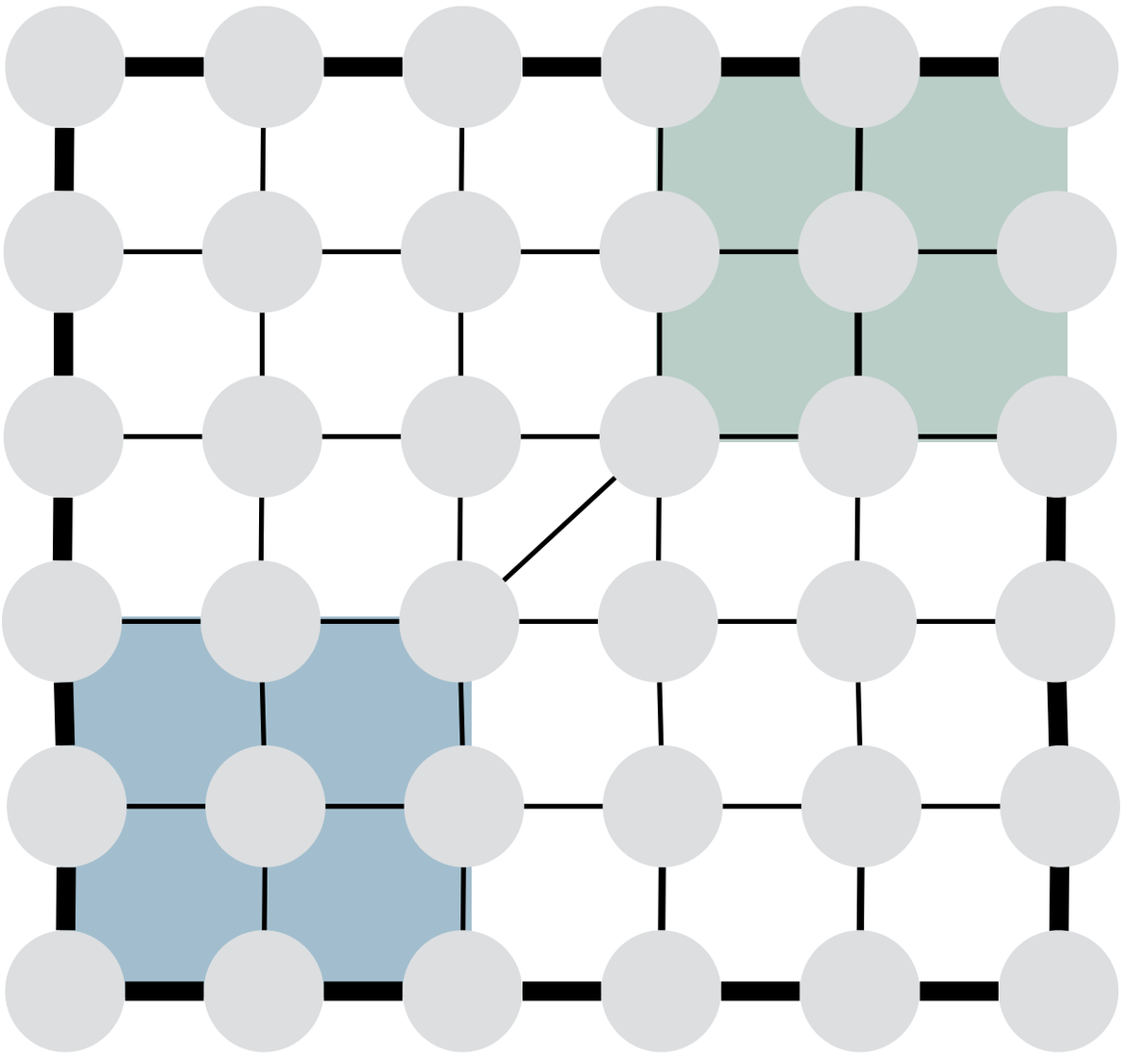
1. Pre-compute a set of least-cost paths
2. Identify flows by hashing packet header fields
3. Randomly forward along least cost paths

ECMP

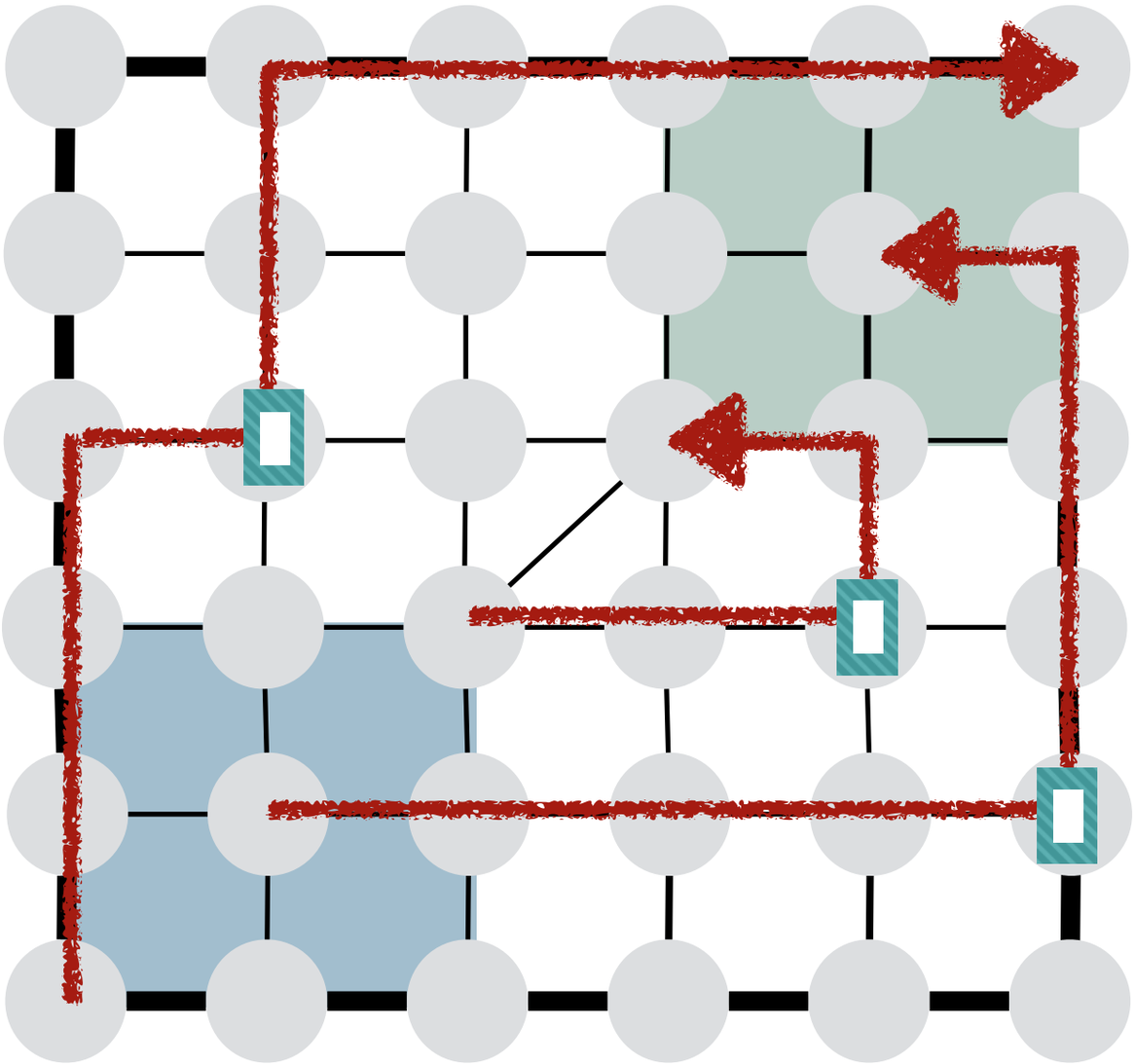


1. Pre-compute a set of least-cost paths
2. Identify flows by hashing packet header fields
3. Randomly forward along least cost paths

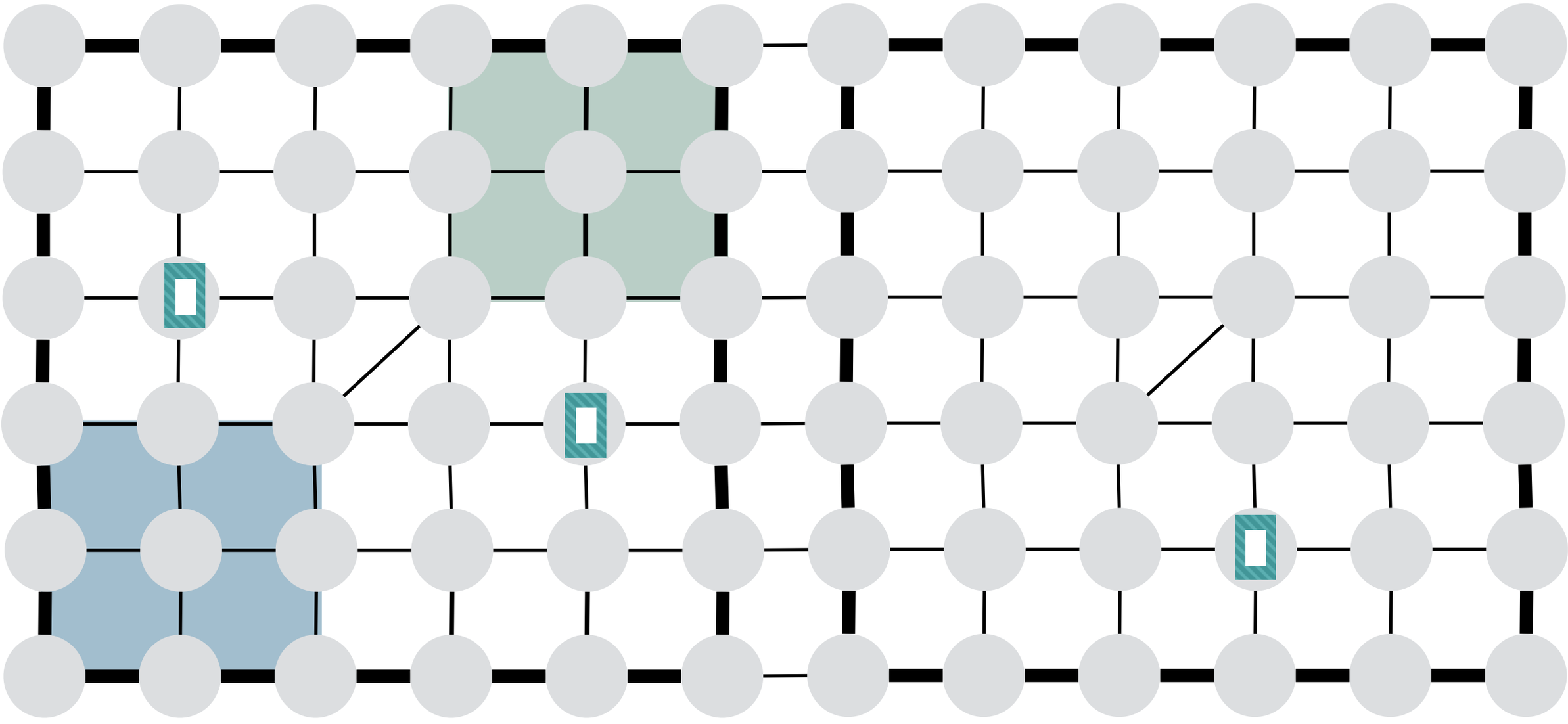
VALIANT LOAD BALANCING



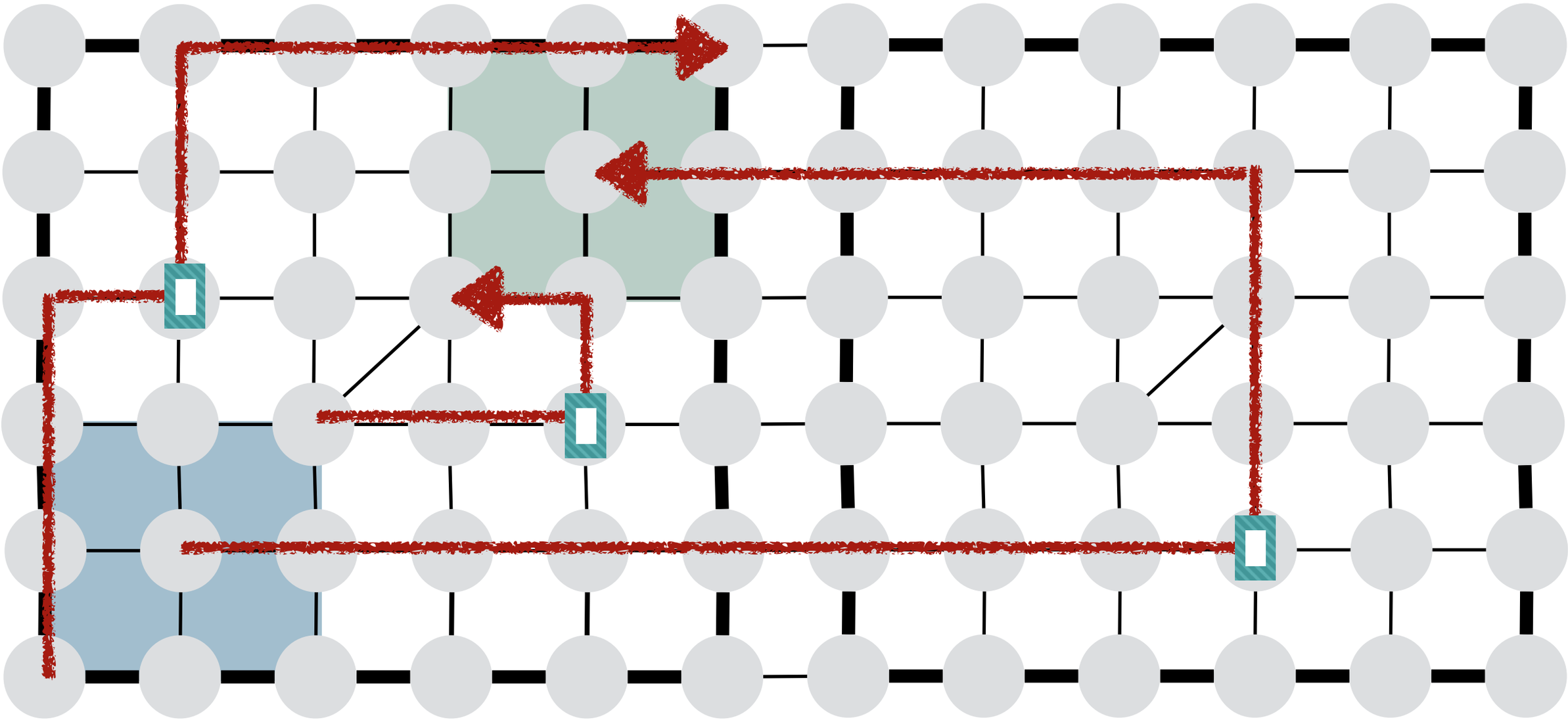
VALIANT LOAD BALANCING



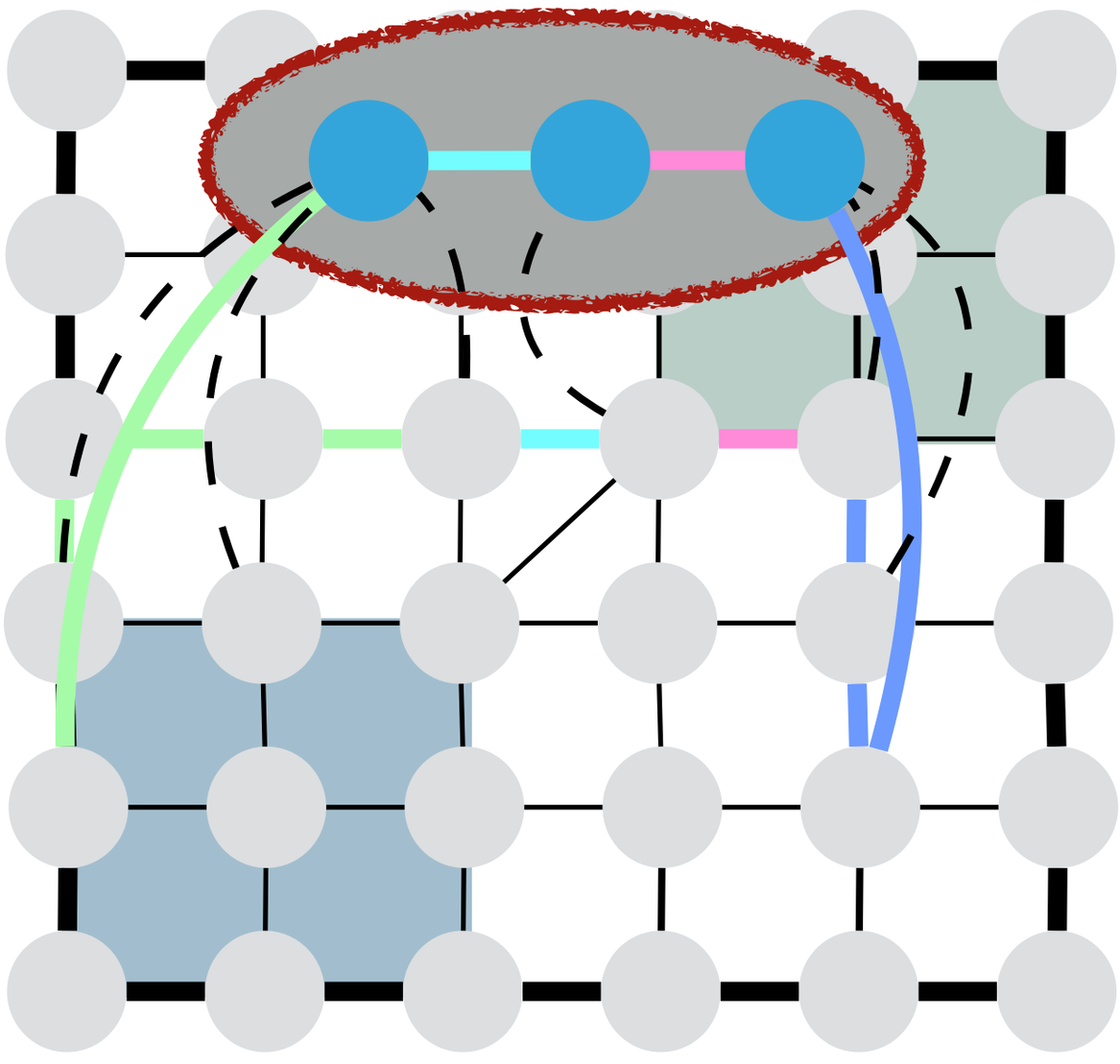
VALIANT LOAD BALANCING



VALIANT LOAD BALANCING



OBLIVIOUS ROUTING (RÄCKE '08)



- Idea: if we can somehow limit ourselves to shorter paths can we get low-congestion routing schemes?
- A *randomized routing tree* is probability distribution over routing trees
- We need a way to construct routing trees that have low stretch...

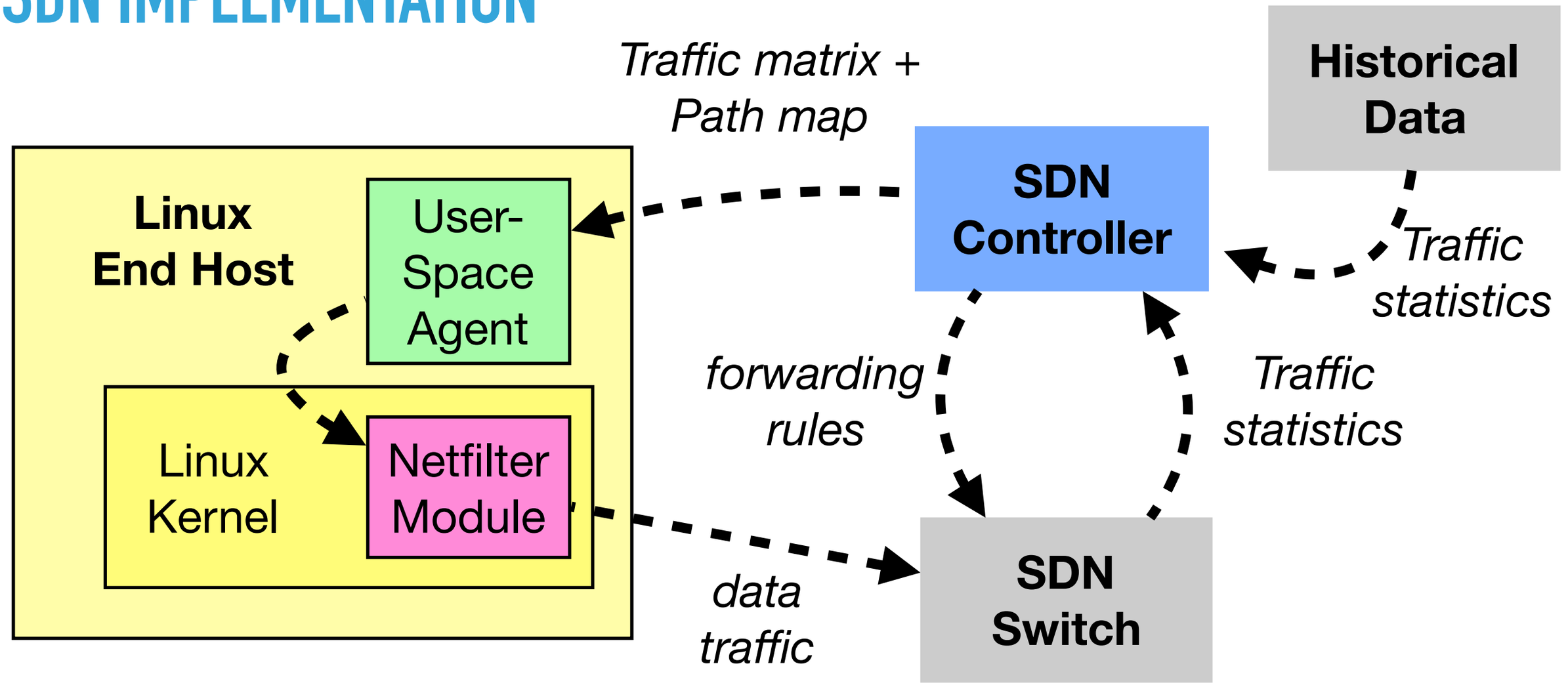
RÄCKE'S ALGORITHM + SEMI-OBLIVIOUS ROUTING

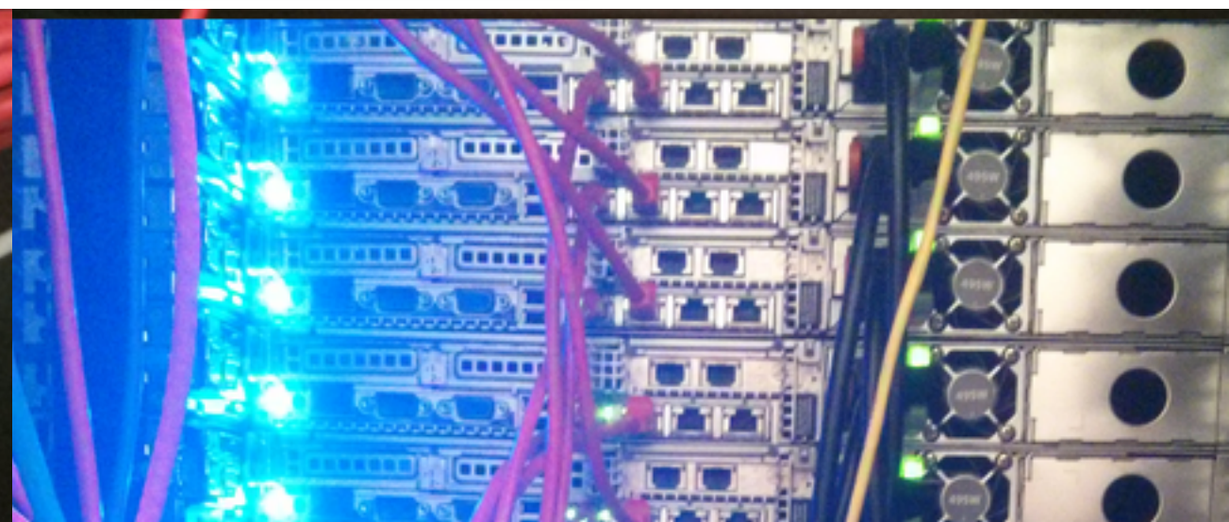
- Iteratively constructs a randomized routing tree
- At each iteration, it penalizes edges that have been heavily utilized in previous trees
- Achieves a poly-logarithmic competitive ratio with respect to optimal **regardless of the demand matrix**—i.e. it is *oblivious!*
- *Semi-oblivious routing* combines Räcke's oblivious routing with dynamic rate adaptation and local failure recovery
  - Compute forwarding paths *statically*
  - Adjust sending rates *dynamically*
- Performance of semi-oblivious routing in practice not known...

IMPLEMENTATION AND EVALUATION

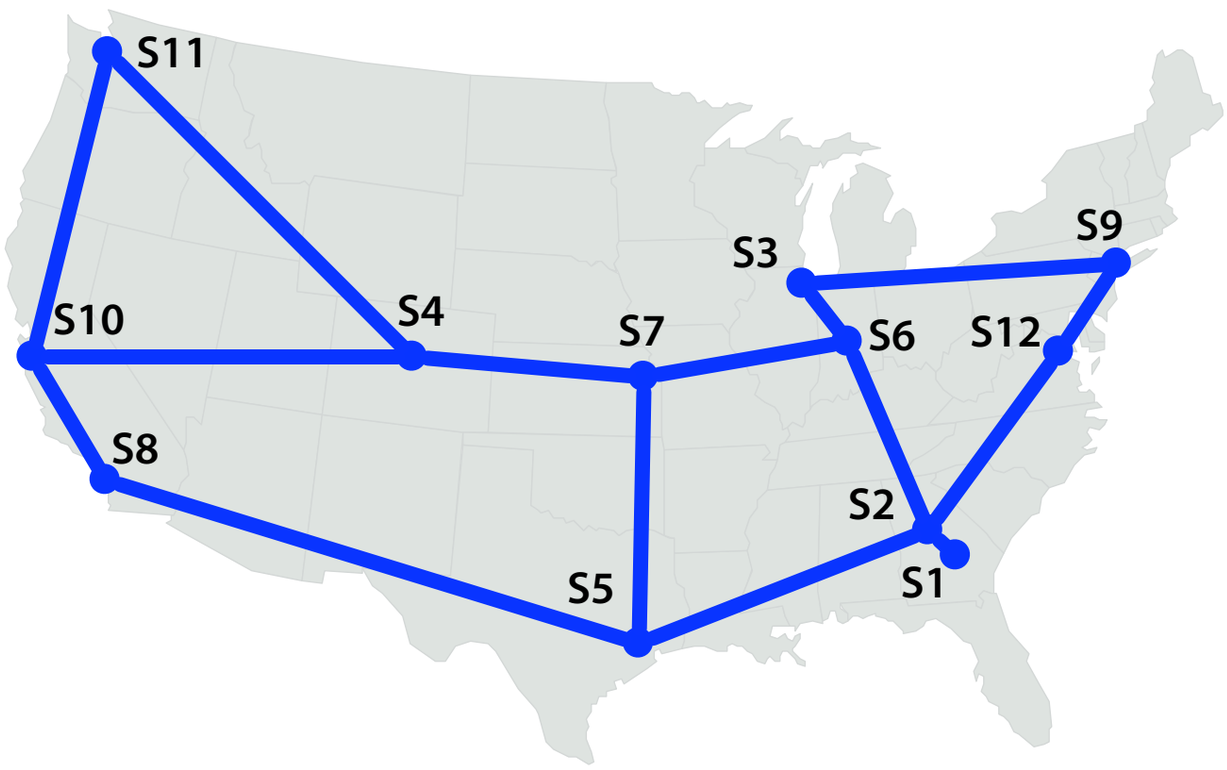
* HARDWARE TESTBED
* SIMULATION

SDN IMPLEMENTATION



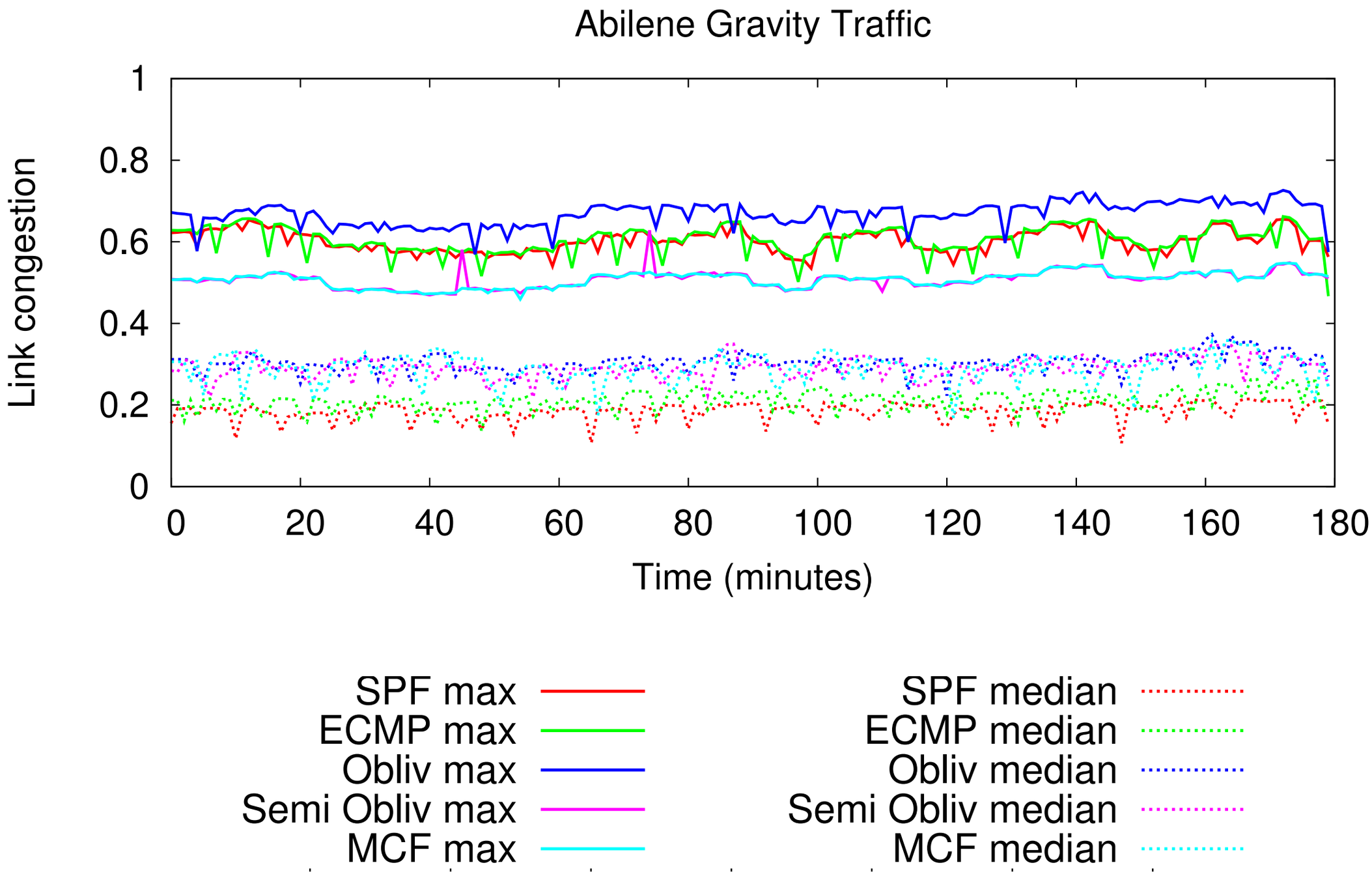


ABILENE TOPOLOGY



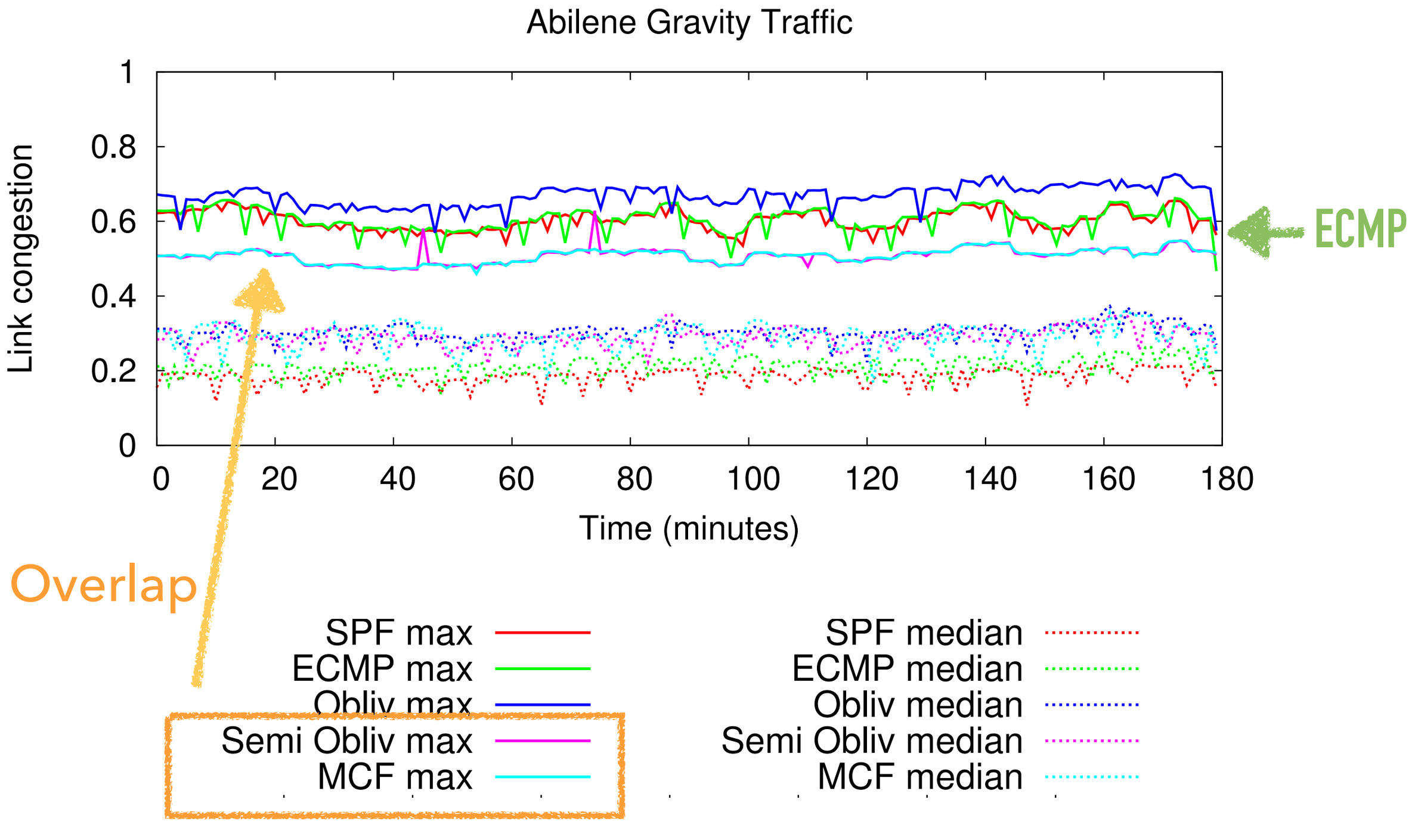
- Emulated Abilene topology in hardware test bed
- Used real-world and worst case traffic scenarios
- Compared shortest-path, ECMP, MCF, oblivious, and semi-oblivious

ABILENE TOPOLOGY: SIMULATED WORKLOAD

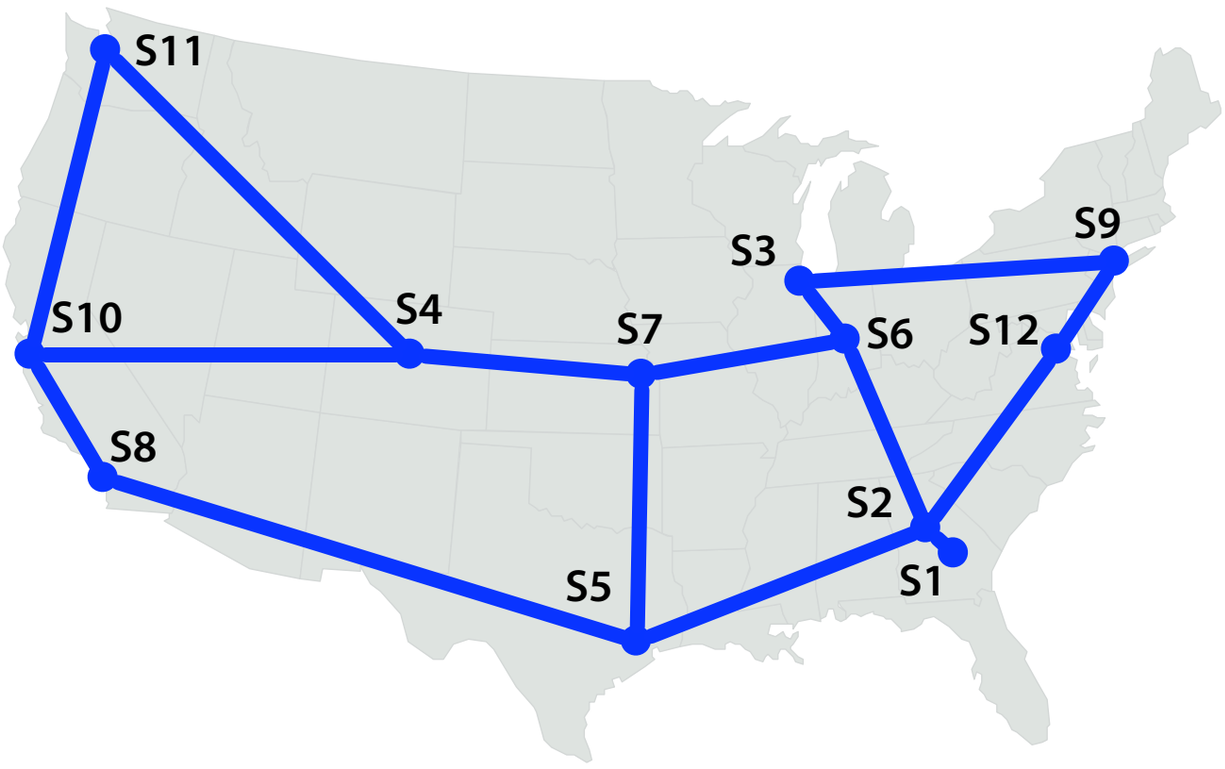


ECMP

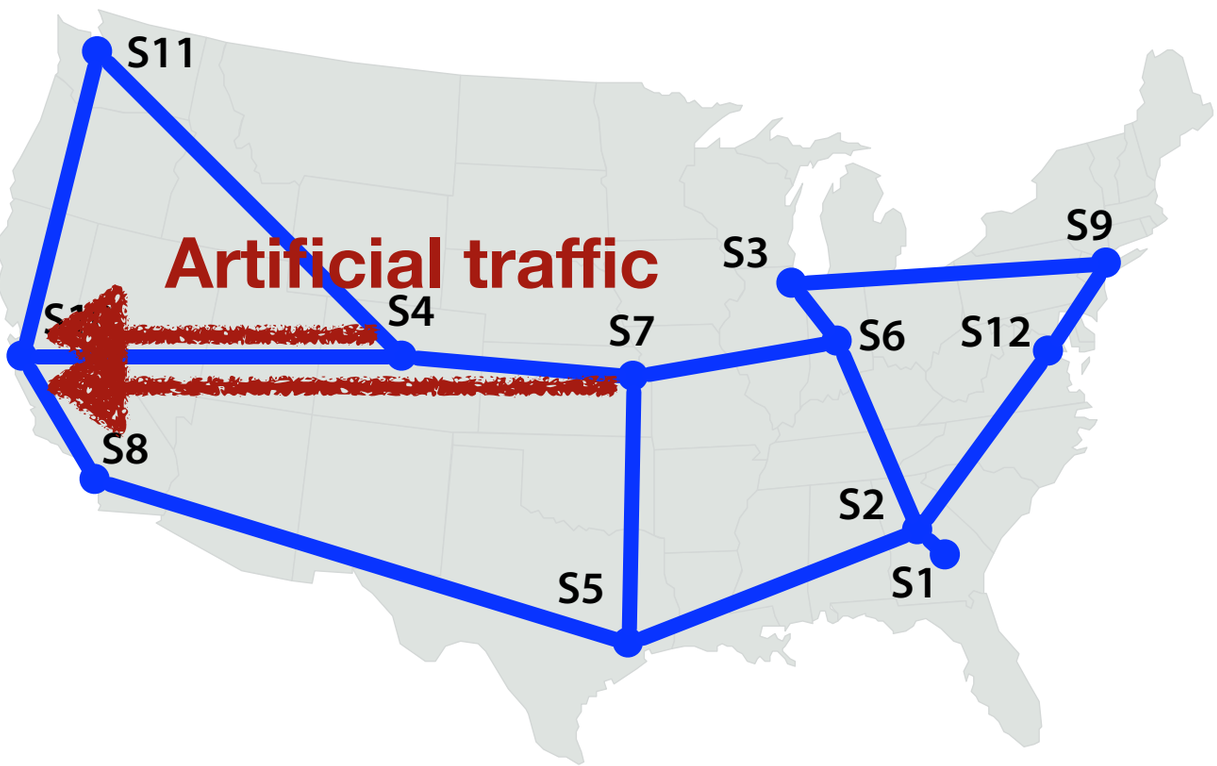
ABILENE TOPOLOGY: SIMULATED WORKLOAD



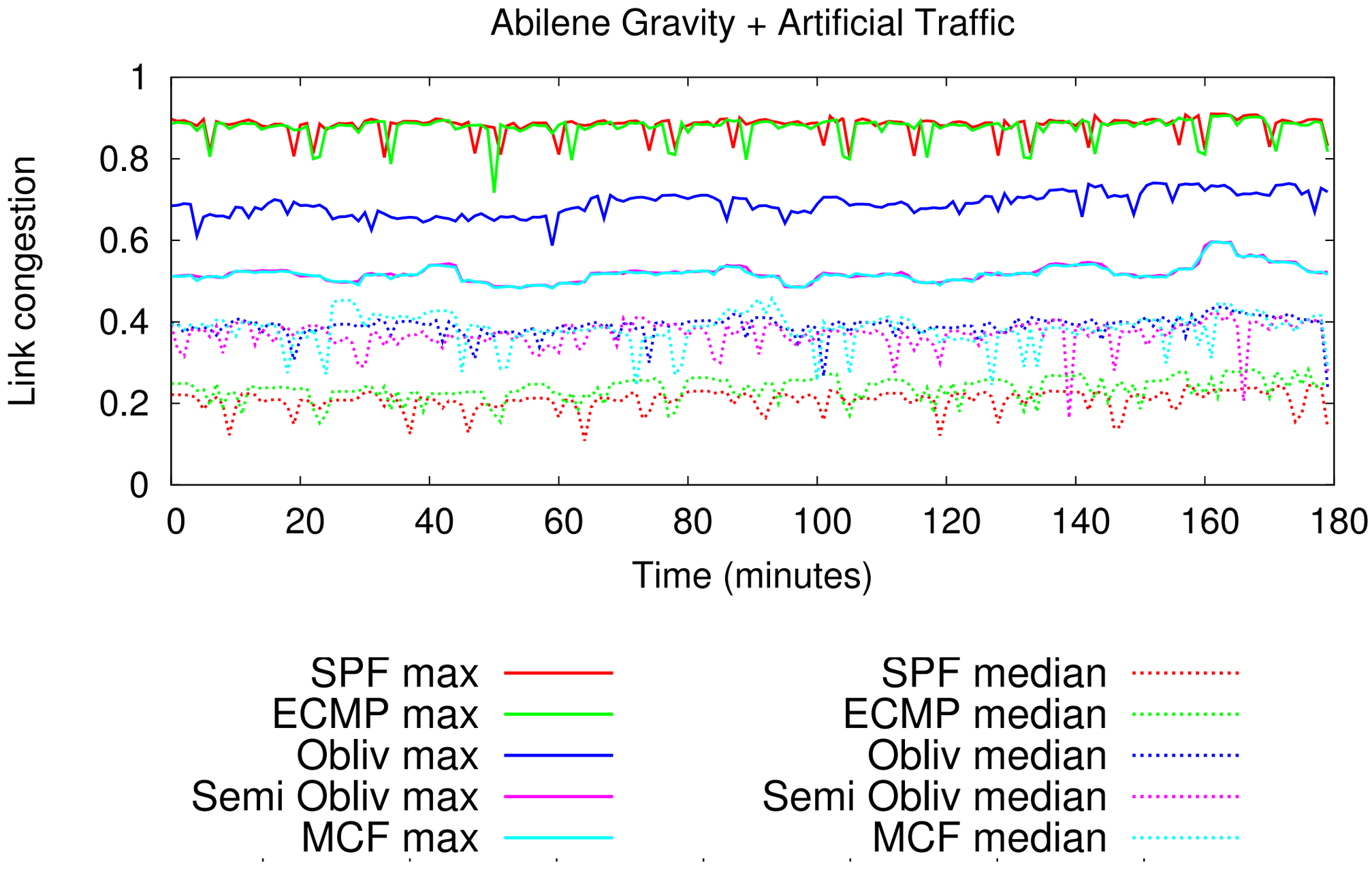
ABILENE TOPOLOGY: SIMULATED WORKLOAD



ABILENE TOPOLOGY: SIMULATED WORKLOAD

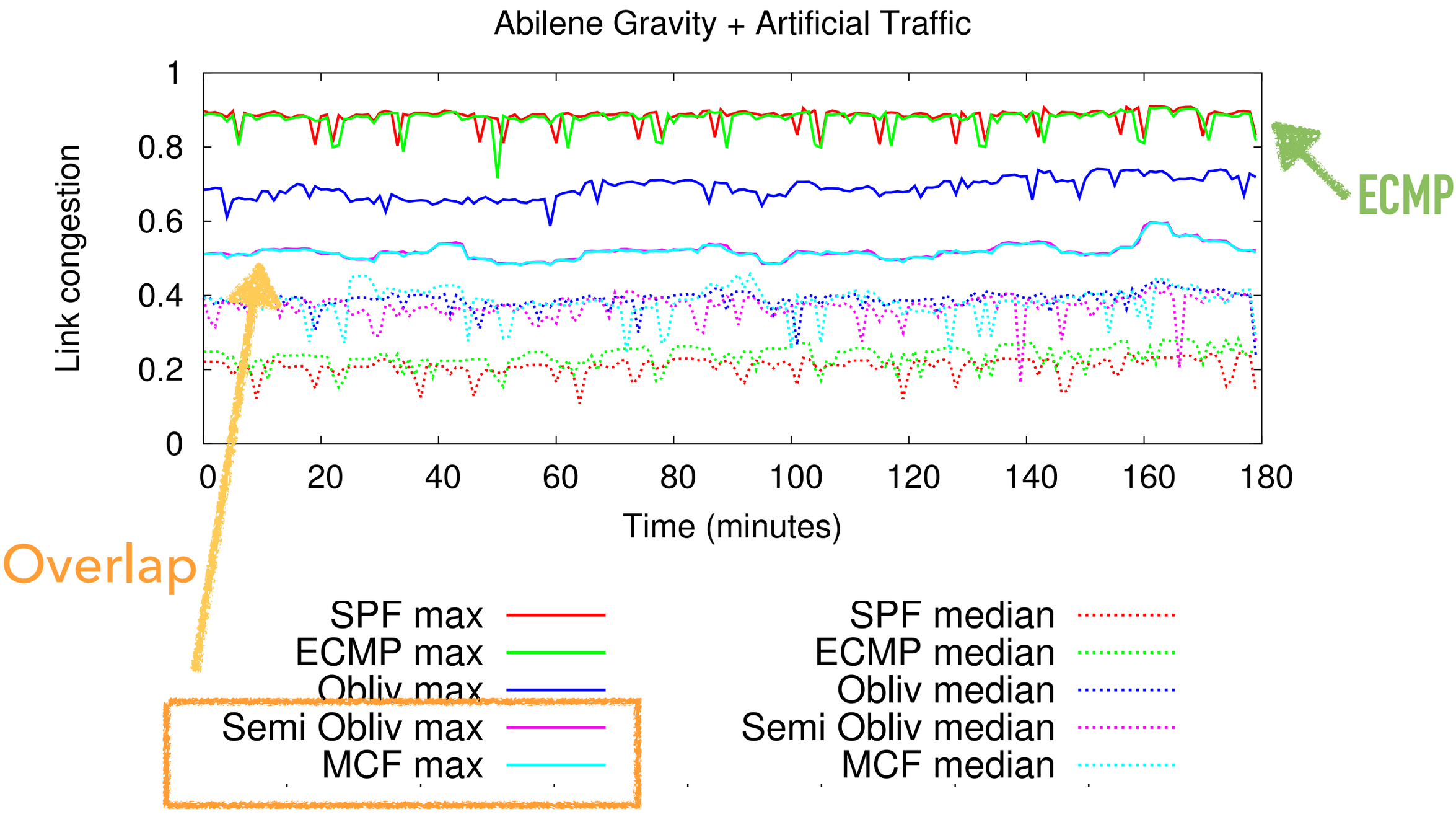


ABILENE TOPOLOGY: SIMULATED WORKLOAD



ECMP

ABILENE TOPOLOGY: SIMULATED WORKLOAD



KULFI FRAMEWORK

Routing Algorithm	Description	Type	Path Diversity	Max Congestion	Overheads Churn	Overheads Recovery
MCF	Multi-Commodity Flow solved with LP [20]	conscious	medium	least	high	slow
MW	Multi-Commodity Flow solved with Multiplicative Weights [17]	conscious	medium	least	high	slow
SPF	Shortest Path First	oblivious	least	high	none	none
ECMP	Equal-Cost, Multi-Path	oblivious	low	high	none	fast
KSP	K-Shortest Paths	oblivious	medium	medium	none	fast
Räcke	Räcke [38]	oblivious	high	low	none	fast
VLB	Valiant Load Balancing [44]	oblivious	high	medium	none	fast
SemiMCF-MCF	MCF for paths MCF for weights	semi-oblivious	medium	least	none	fast
SemiMCF-ECMP	ECMP for paths MCF for weights	semi-oblivious	low	medium	none	fast
SemiMCF-KSP	KSP for paths MCF for weights	semi-oblivious	medium	medium	none	fast
SemiMCF-Räcke	Räcke for paths MCF for weights	semi-oblivious	high	low	none	fast
SemiMCF-VLB	VLB for paths MCF for weights	semi-oblivious	high	medium	none	fast
SemiMCF-MCF-Env	MCF over demand envelope for paths MCF for weights [43]	semi-oblivious	medium	low	none	fast
SemiMCF-MCF-FT-Env	Multiple MCF-Env considering failures MCF for weights [43]	semi-oblivious	high	medium	none	fast

- Implemented over a dozen different traffic engineering schemes
- Measure performance in simulator and hardware testbed with a variety of demands and failures
- Used “local” failure recovery

KULFI FRAMEWORK

Routing Algorithm	Description	Type	Path Diversity	Max Congestion	Overheads Churn	Overheads Recovery
MCF	Multi-Commodity Flow solved with LP [20]	conscious	medium	least	high	slow
MW	Multi-Commodity Flow solved with Multiplicative Weights [17]	conscious	medium	least	high	slow
SPF	Shortest Path First	oblivious	least	high	none	none
ECMP	Equal-Cost, Multi-Path	oblivious	low	high	none	fast
KSP	K-Shortest Paths	oblivious	medium	medium	none	fast
Räcke	Räcke [38]	oblivious	high	low	none	fast
VLB	Valiant Load Balancing [44]	oblivious	high	medium	none	fast
SemiMCF-MCF	MCF for paths MCF for weights	semi-oblivious	medium	least	none	fast
SemiMCF-ECMP	ECMP for paths MCF for weights	semi-oblivious	low	medium	none	fast
SemiMCF-KSP	KSP for paths MCF for weights	semi-oblivious	medium	medium	none	fast
SemiMCF-Räcke	Räcke for paths MCF for weights	semi-oblivious	high	low	none	fast
SemiMCF-VLB	VLB for paths MCF for weights	semi-oblivious	high	medium	none	fast
SemiMCF-MCF-Env	MCF over demand envelope for paths MCF for weights [43]	semi-oblivious	medium	low	none	fast
SemiMCF-MCF-FT-Env	Multiple MCF-Env considering failures MCF for weights [43]	semi-oblivious	high	medium	none	fast

SWAN

- Implemented over a dozen different traffic engineering schemes
- Measure performance in simulator and hardware testbed with a variety of demands and failures
- Used “local” failure recovery

KULFI FRAMEWORK

Routing Algorithm	Description	Type	Path Diversity	Max Congestion	Overheads Churn	Overheads Recovery
MCF	Multi-Commodity Flow solved with LP [20]	conscious	medium	least	high	slow
MW	Multi-Commodity Flow solved with Multiplicative Weights [17]	conscious	medium	least	high	slow
SPF	Shortest Path First	oblivious	least	high	none	none
ECMP	Equal-Cost, Multi-Path	oblivious	low	high	none	fast
KSP	K-Shortest Paths	oblivious	medium	medium	none	fast
Räcke	Räcke [38]	oblivious	high	low	none	fast
VLB	Valiant Load Balancing [44]	oblivious	high	medium	none	fast
SemiMCF-MCF	MCF for paths MCF for weights	semi-oblivious	medium	least	none	fast
SemiMCF-ECMP	ECMP for paths MCF for weights	semi-oblivious	low	medium	none	fast
SemiMCF-KSP	KSP for paths MCF for weights	semi-oblivious	medium	medium	none	fast
SemiMCF-Räcke	Räcke for paths MCF for weights	semi-oblivious	high	low	none	fast
SemiMCF-VLB	VLB for paths MCF for weights	semi-oblivious	high	medium	none	fast
SemiMCF-MCF-Env	MCF over demand envelope for paths MCF for weights [43]	semi-oblivious	medium	low	none	fast
SemiMCF-MCF-FT-Env	Multiple MCF-Env considering failures MCF for weights [43]	semi-oblivious	high	medium	none	fast

SWAN

- Implemented over a dozen different traffic engineering schemes
- Measure performance in simulator and hardware testbed with a variety of demands and failures
- Used “local” failure recovery

KULFI FRAMEWORK

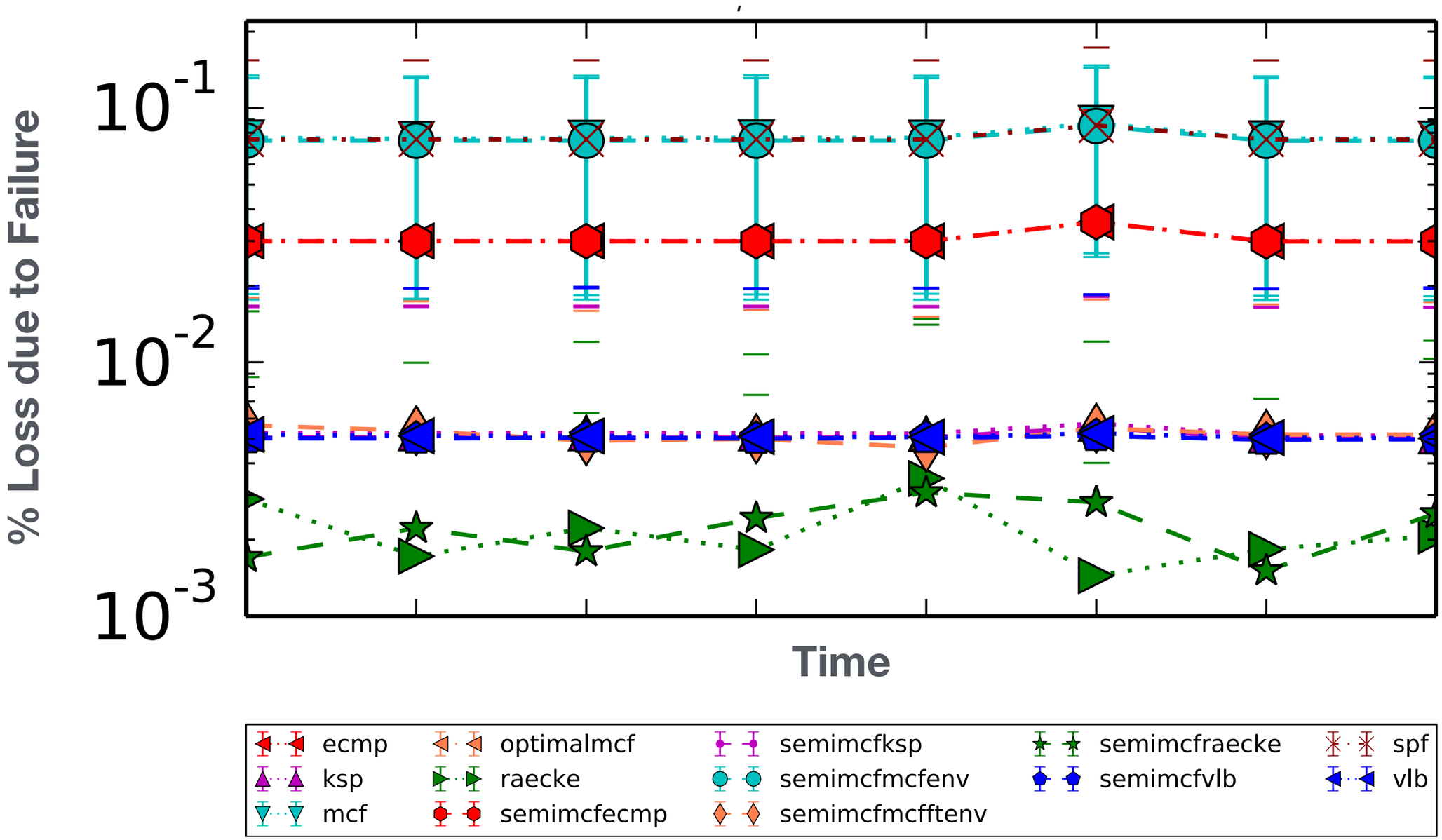
Routing Algorithm	Description	Type	Path Diversity	Max Congestion	Overheads Churn	Overheads Recovery
MCF	Multi-Commodity Flow solved with LP [20]	conscious	medium	least	high	slow
MW	Multi-Commodity Flow solved with Multiplicative Weights [17]	conscious	medium	least	high	slow
SPF	Shortest Path First	oblivious	least	high	none	none
ECMP	Equal-Cost, Multi-Path	oblivious	low	high	none	fast
KSP	K-Shortest Paths	oblivious	medium	medium	none	fast
Räcke	Räcke [38]	oblivious	high	low	none	fast
VLB	Valiant Load Balancing [44]	oblivious	high	medium	none	fast
SemiMCF-MCF	MCF for paths MCF for weights	semi-oblivious	medium	least	none	fast
SemiMCF-ECMP	ECMP for paths MCF for weights	semi-oblivious	low	medium	none	fast
SemiMCF-KSP	KSP for paths MCF for weights	semi-oblivious	medium	medium	none	fast
SemiMCF-Räcke	Räcke for paths MCF for weights	semi-oblivious	high	low	none	fast
SemiMCF-VLB	VLB for paths MCF for weights	semi-oblivious	high	medium	none	fast
SemiMCF-MCF-Env	MCF over demand envelope for paths MCF for weights [43]	semi-oblivious	medium	low	none	fast
SemiMCF-MCF-FT-Env	Multiple MCF-Env considering failures MCF for weights [43]	semi-oblivious	high	medium	none	fast

SWAN

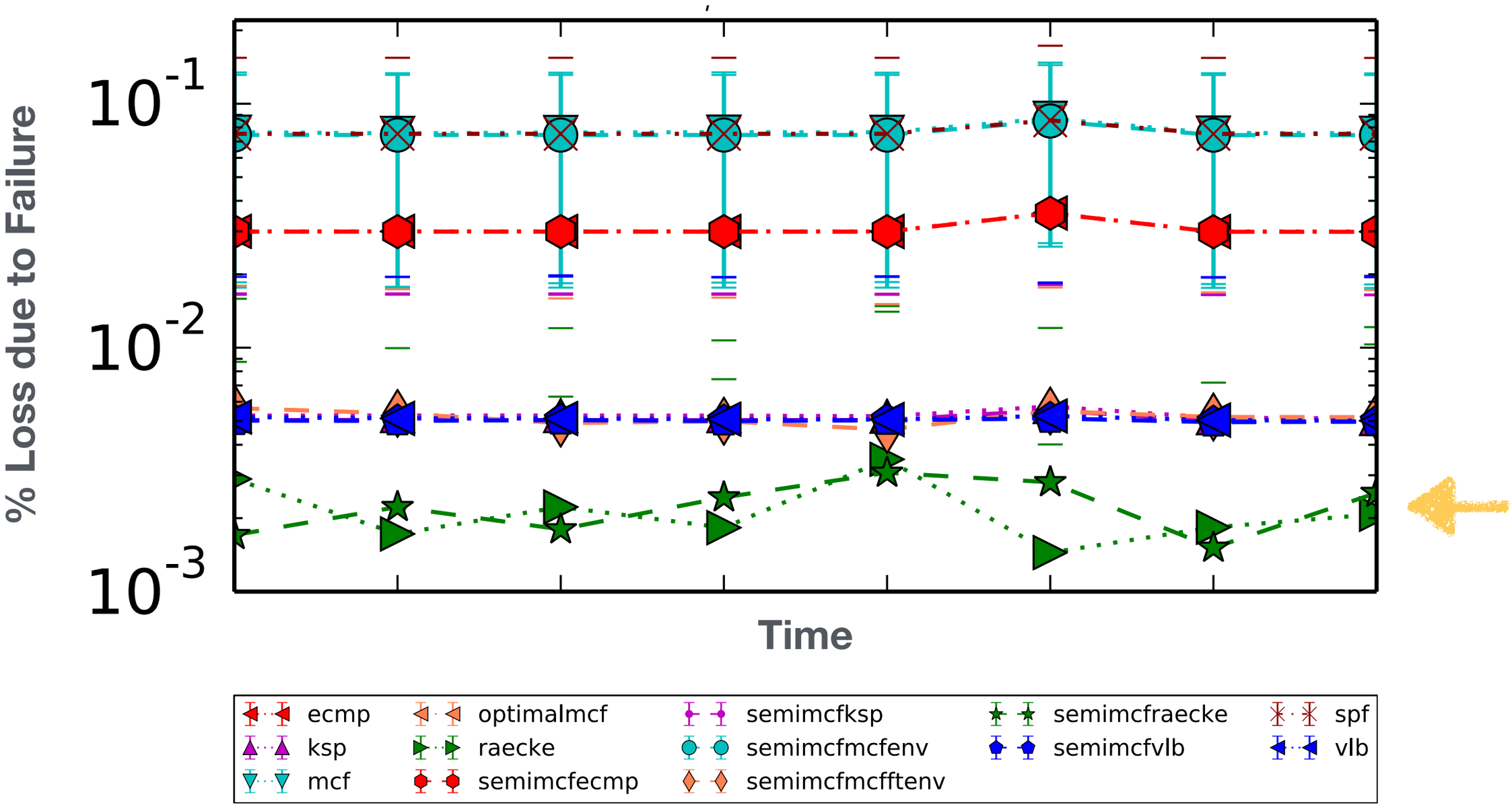
Suchara et al SIGMETRICS ‘11

- Implemented over a dozen different traffic engineering schemes
- Measure performance in simulator and hardware testbed with a variety of demands and failures
- Used “local” failure recovery

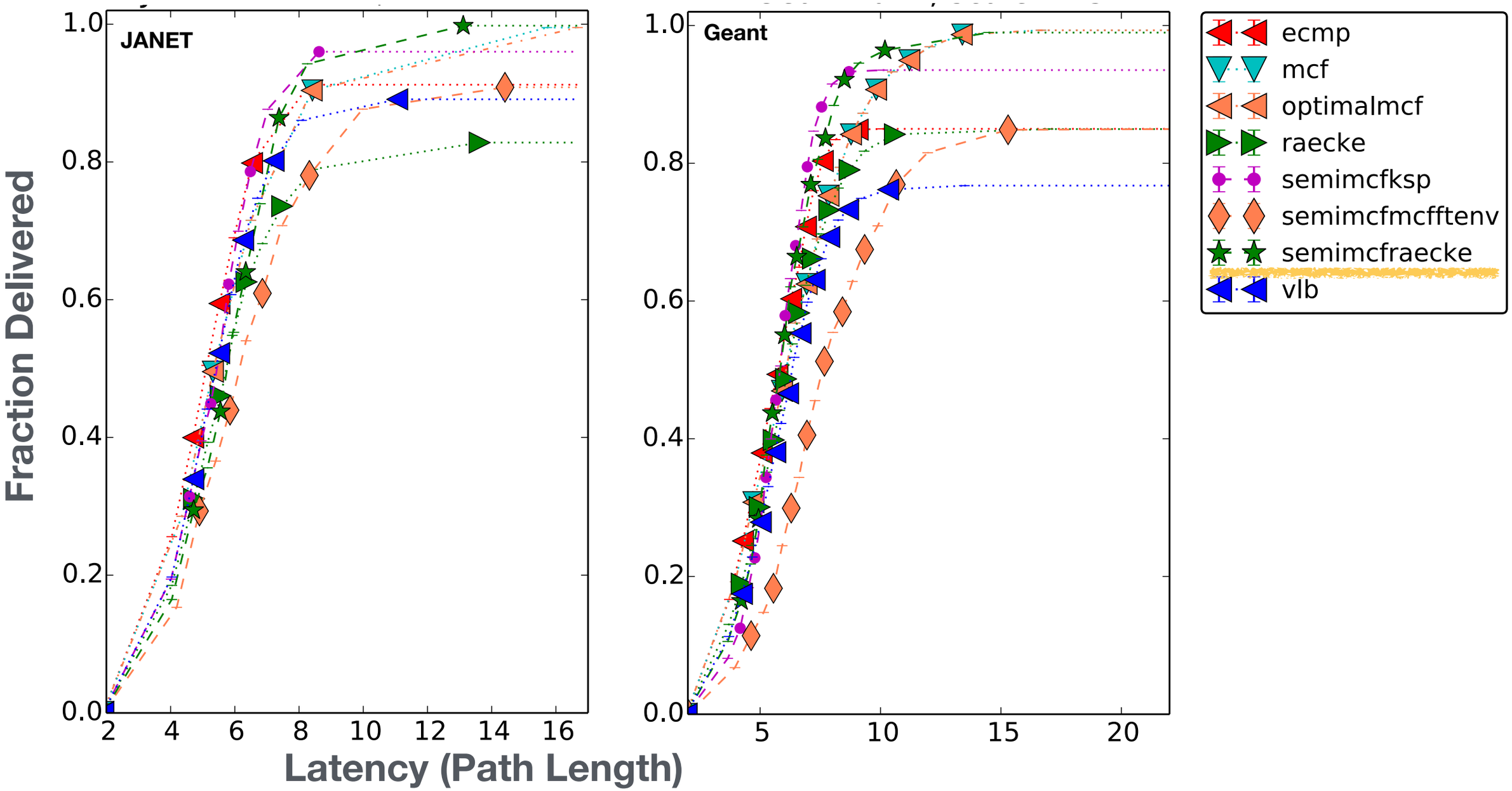
TOPOLOGY ZOO: FAULT TOLERANCE



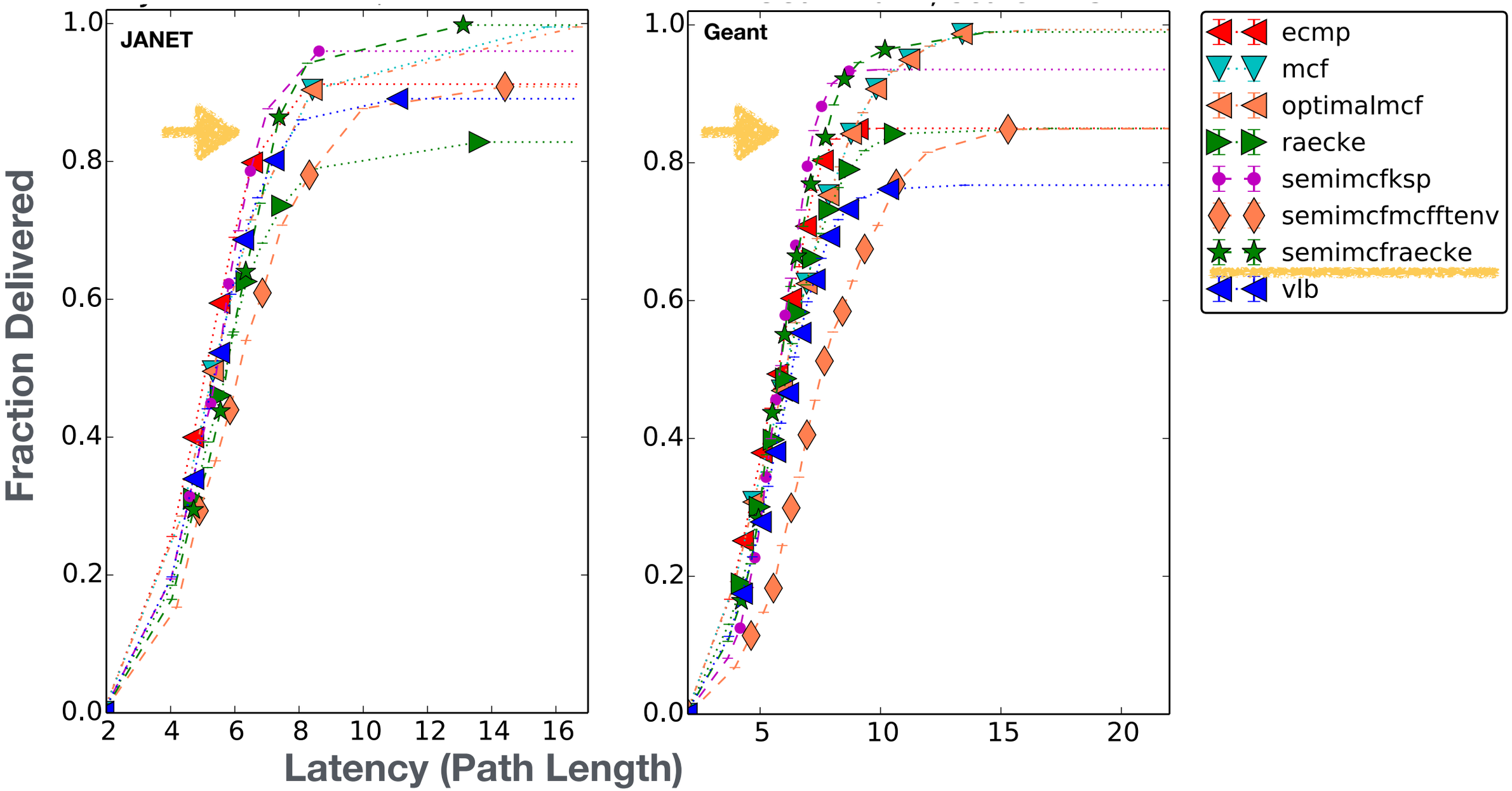
TOPOLOGY ZOO: FAULT TOLERANCE



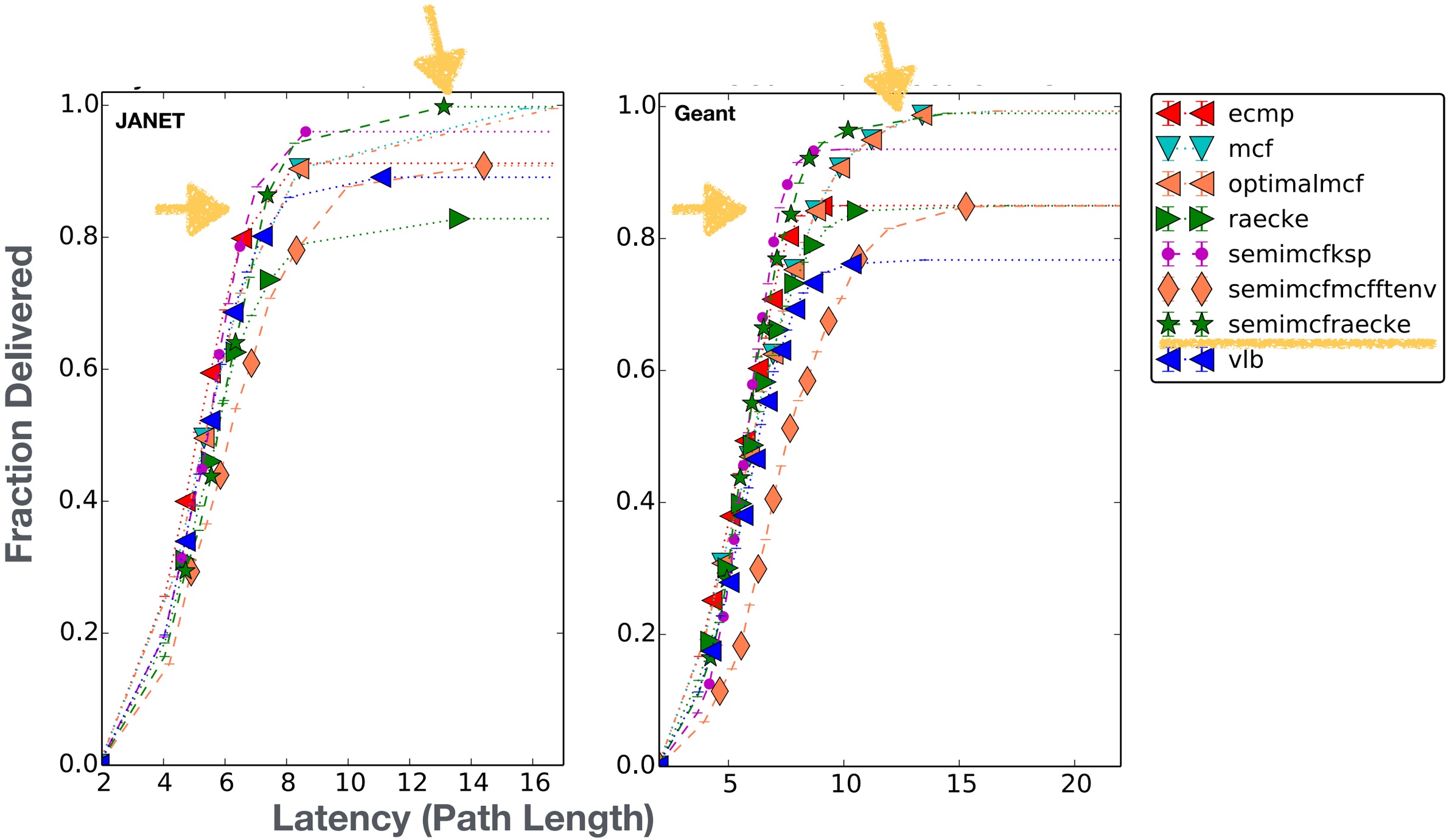
TOPOLOGY ZOO: LATENCY



TOPOLOGY ZOO: LATENCY



TOPOLOGY ZOO: LATENCY



CONCLUSIONS

- Randomization can dramatically simplify traffic engineering while balancing competing objectives
- Oblivious routing performs much better in practice than expected, avoids problems associated with churn, and load-balances better
- Semi-oblivious routing provides near-optimal performance in real-world scenarios, even in the presence of demand mis-prediction, traffic bursts, and failures
- Ongoing work: working with large ISP and content provider to further refine and evaluate Kulfi

https://github.com/merlin-lang/kulfi