

SMORE: Semi-Oblivious Traffic Engineering

Praveen Kumar*

Yang Yuan*

Chris Yu†

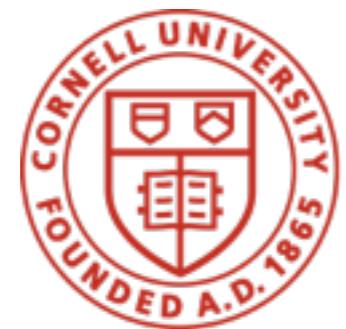
Nate Foster*

Robert Kleinberg*

Petr Lapukhov#

Chiun Lin Lim#

Robert Soulé §



* Cornell



† CMU



Facebook

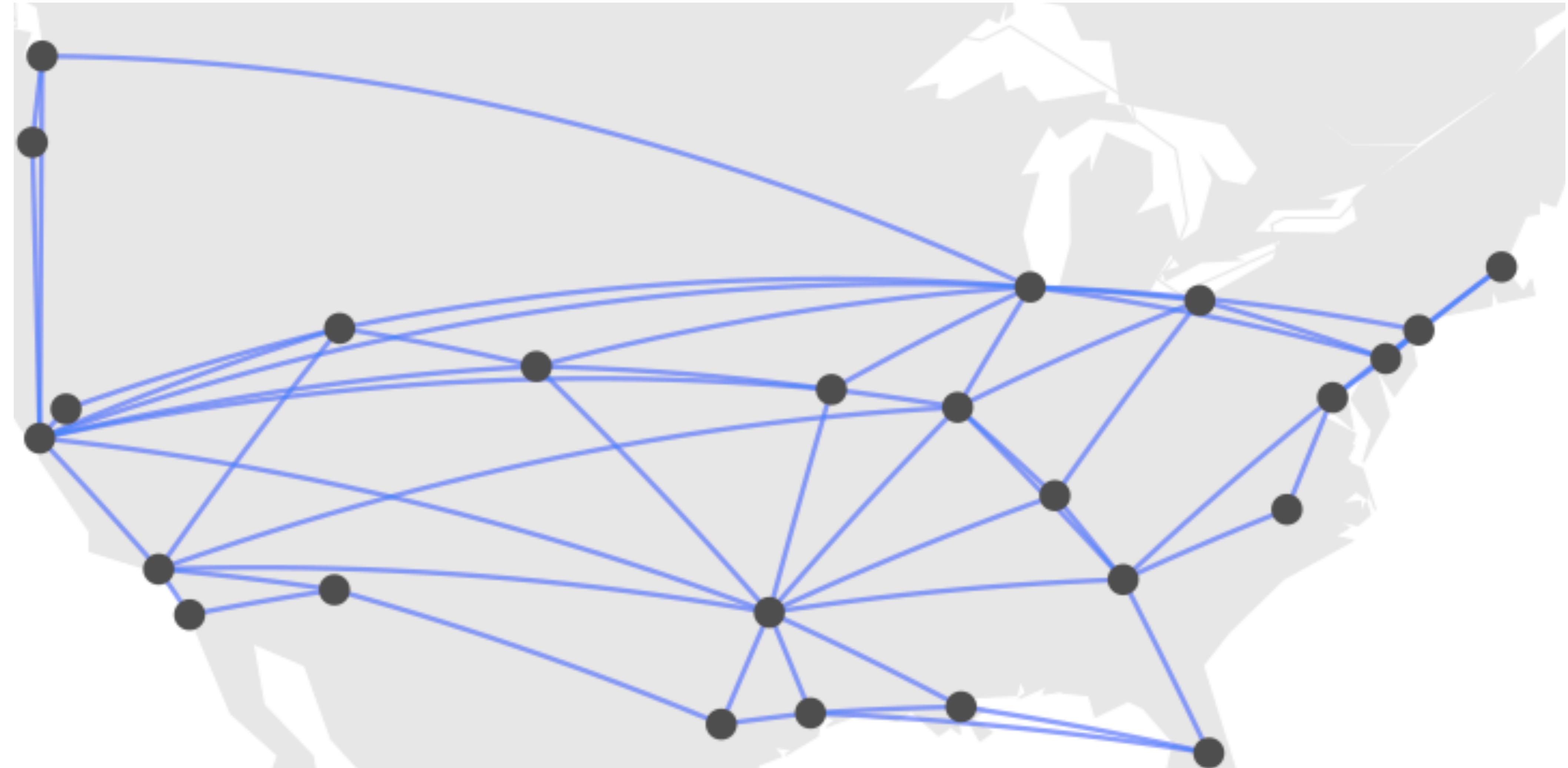


§ USI Lugano



2018

WAN Traffic Engineering



WAN Traffic Engineering

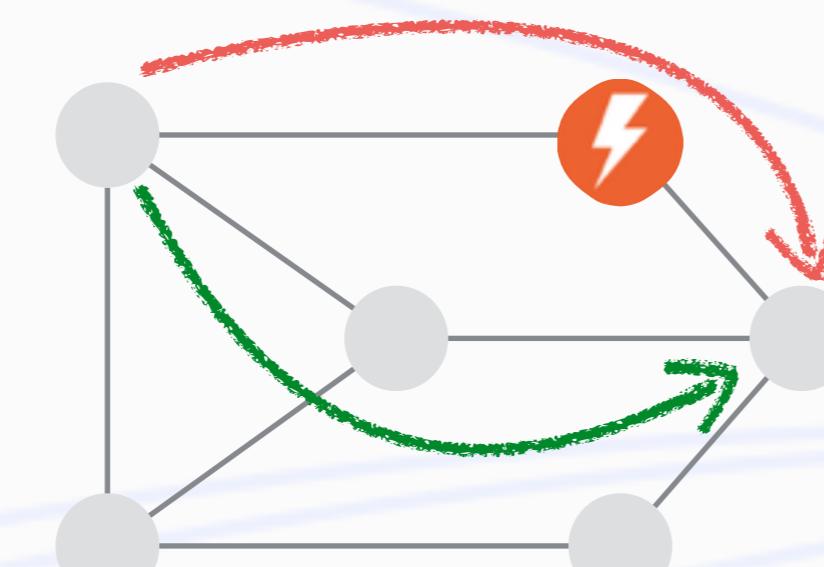
Objectives



Performance



Latency



Robustness



Operational simplicity

Challenges



WAN Traffic Engineering

Objectives



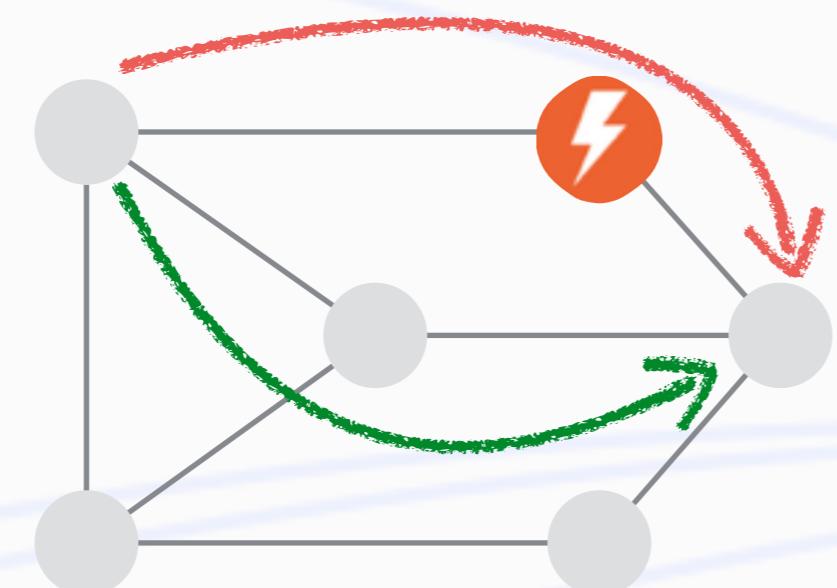
Performance



Latency



Operational simplicity



Robustness

Challenges

Unstructured topology

Heterogeneous capacity

Unexpected failures

Misprediction & Traffic Bursts

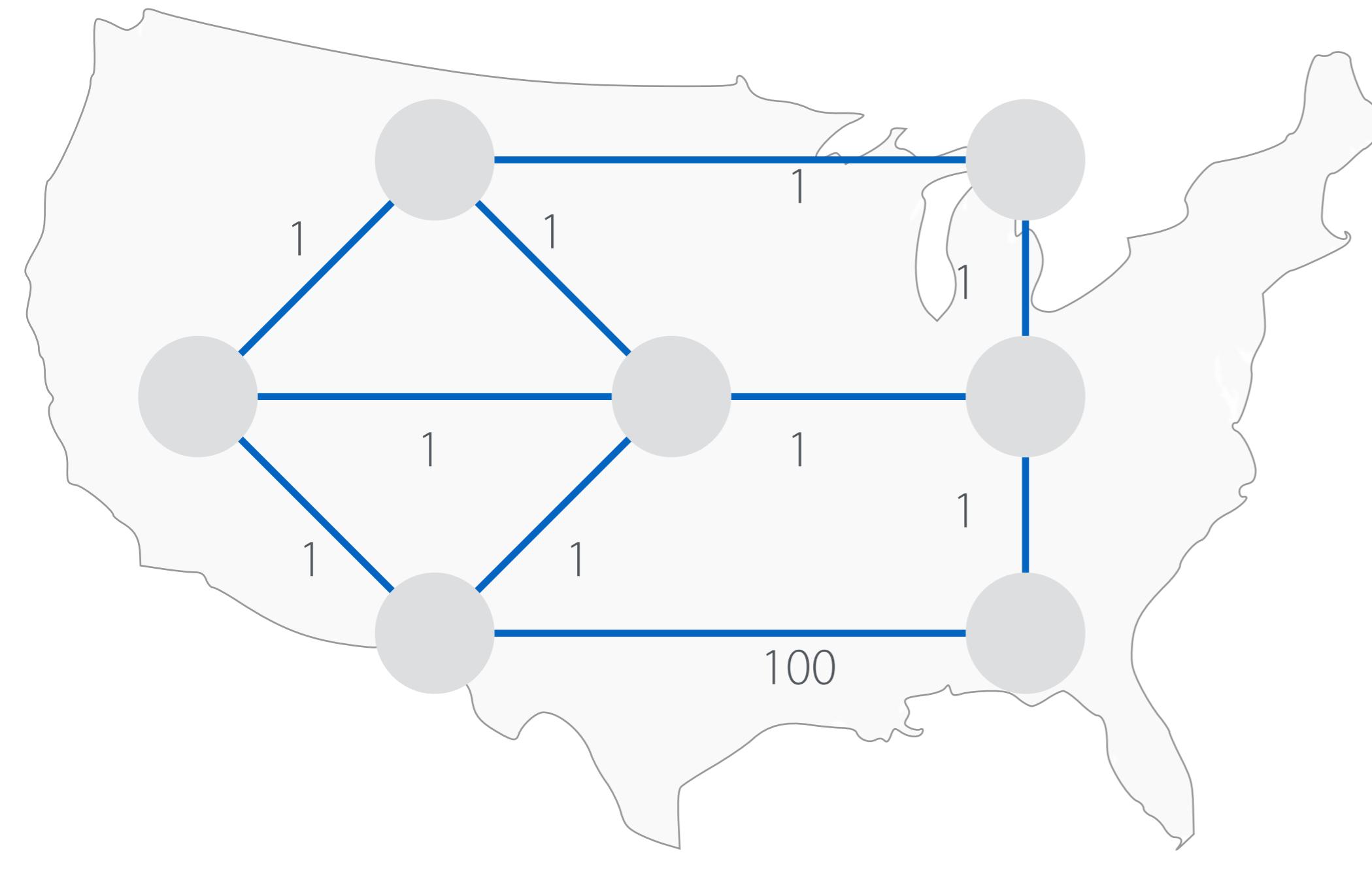
Device limitations

Update overheads

TE Approaches

Traditional
Distributed

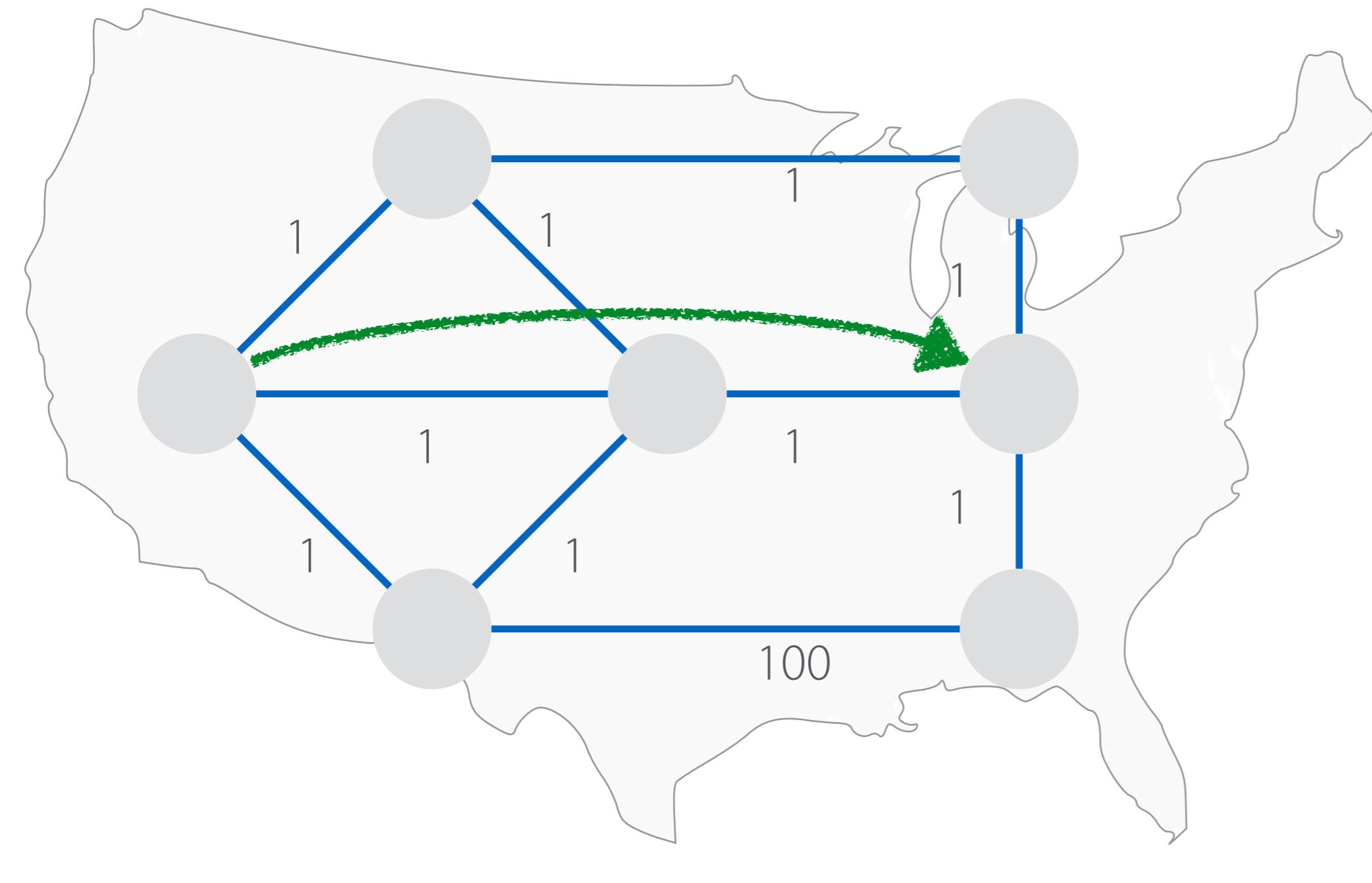
SDN-Based
Centralized



TE Approaches

Traditional
Distributed

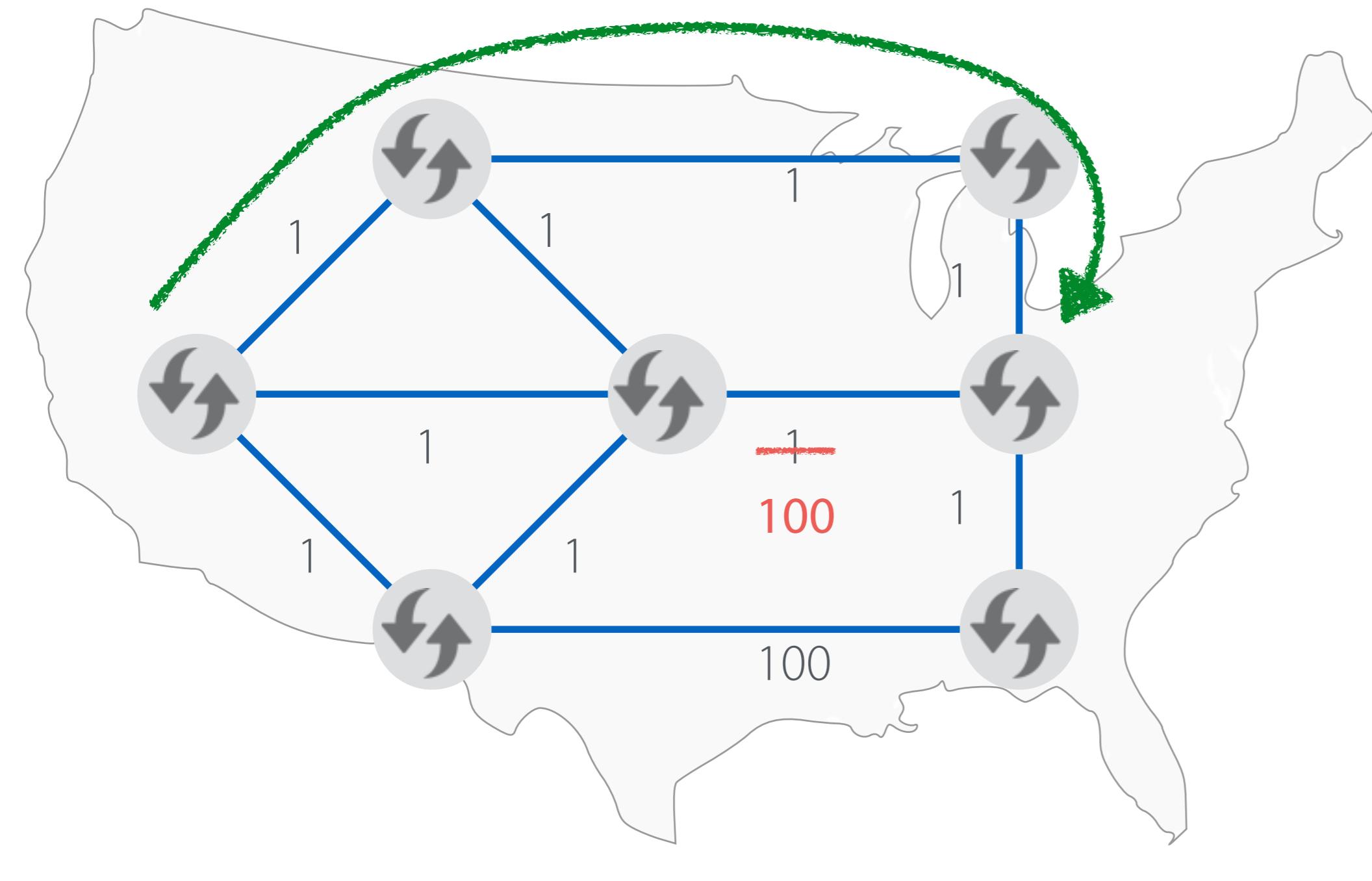
SDN-Based
Centralized



TE Approaches

Traditional
Distributed

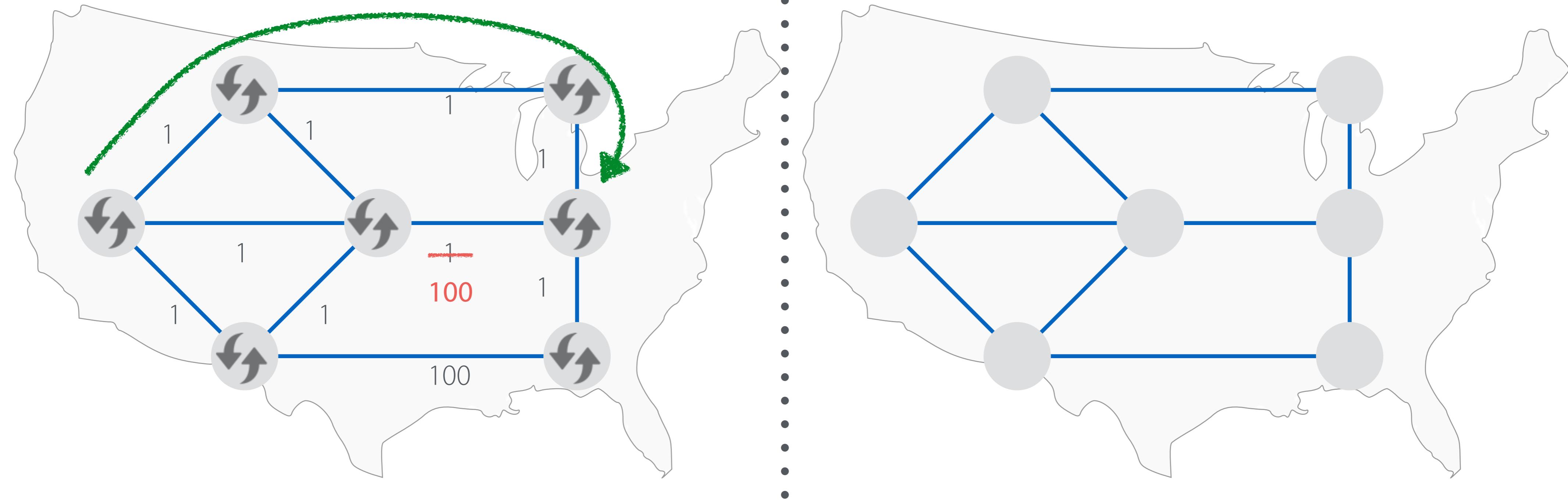
SDN-Based
Centralized



TE Approaches

Traditional
Distributed

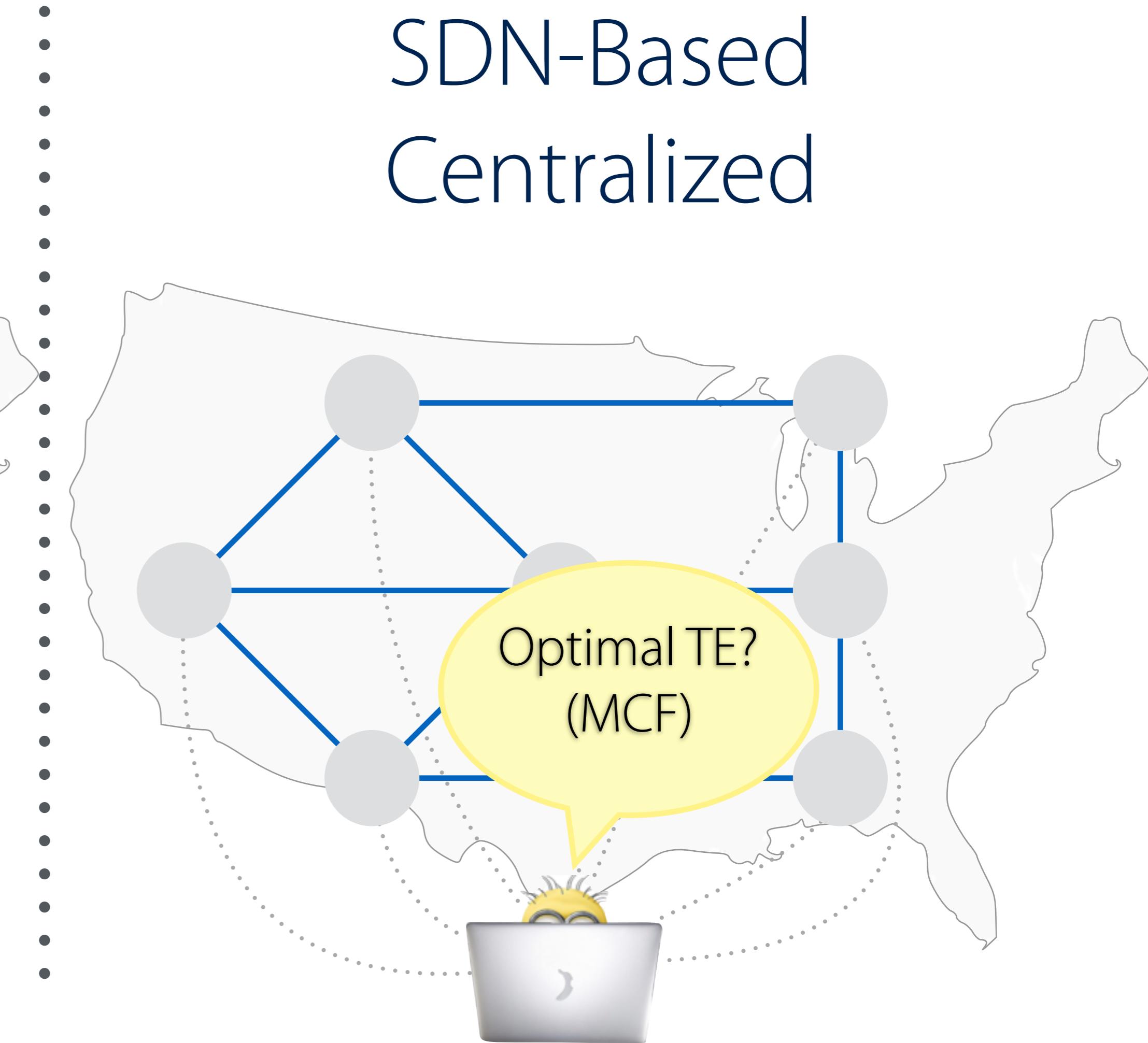
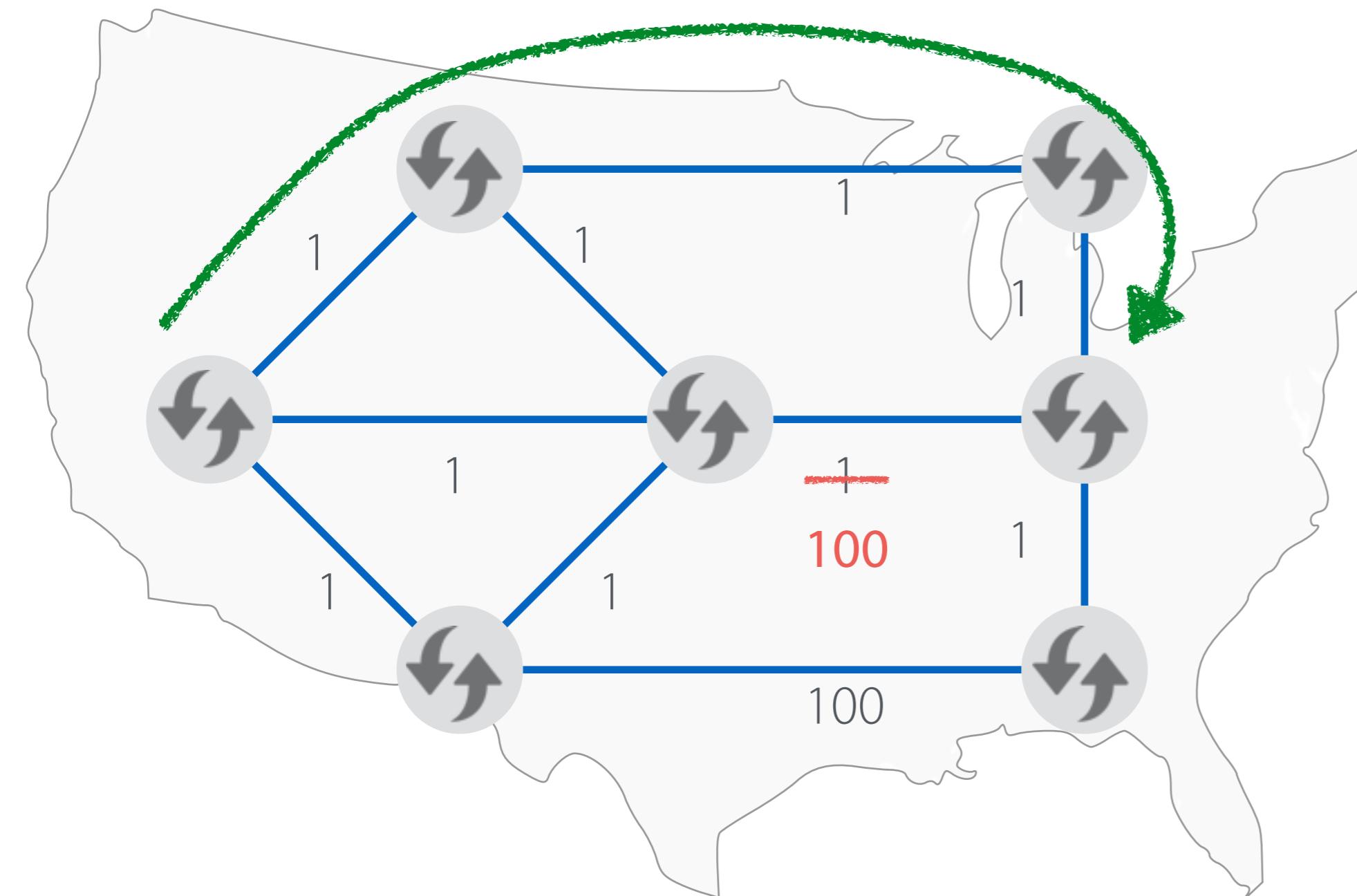
SDN-Based
Centralized



TE Approaches

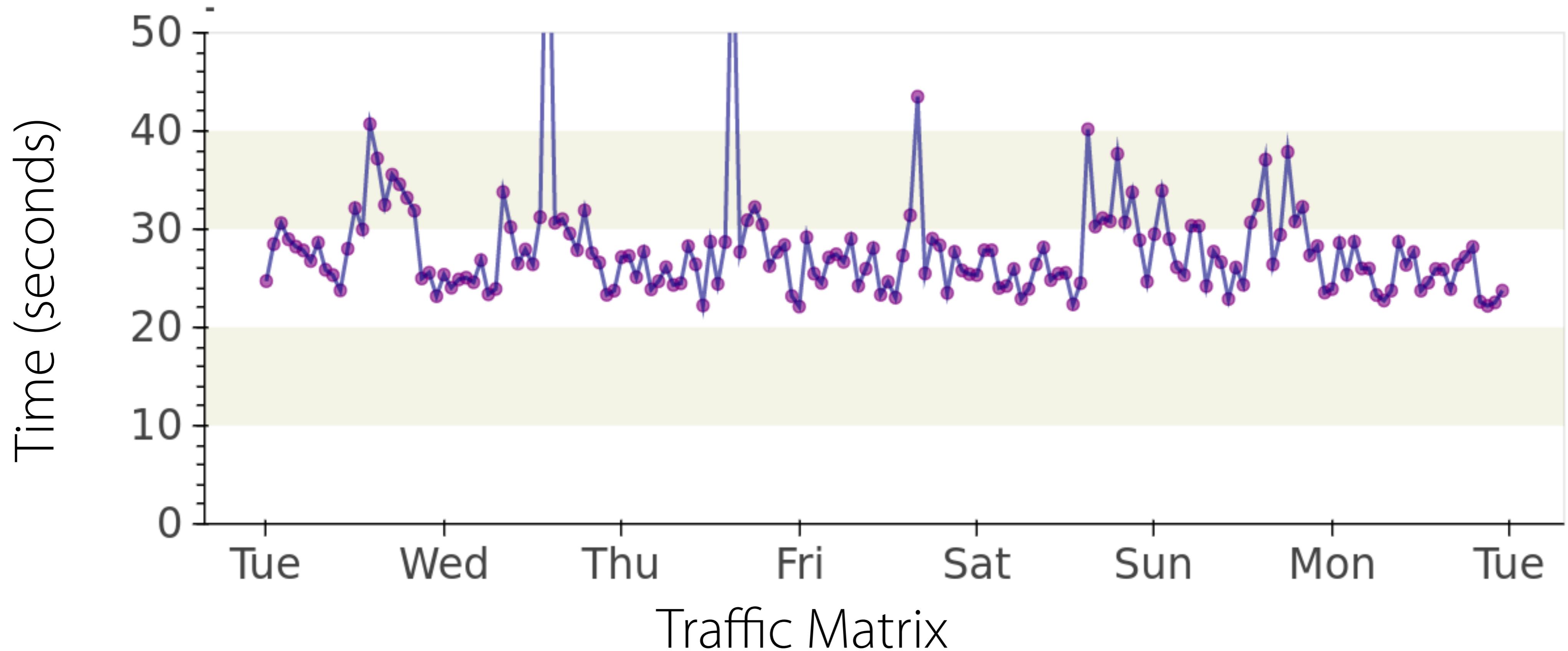
Traditional
Distributed

SDN-Based
Centralized



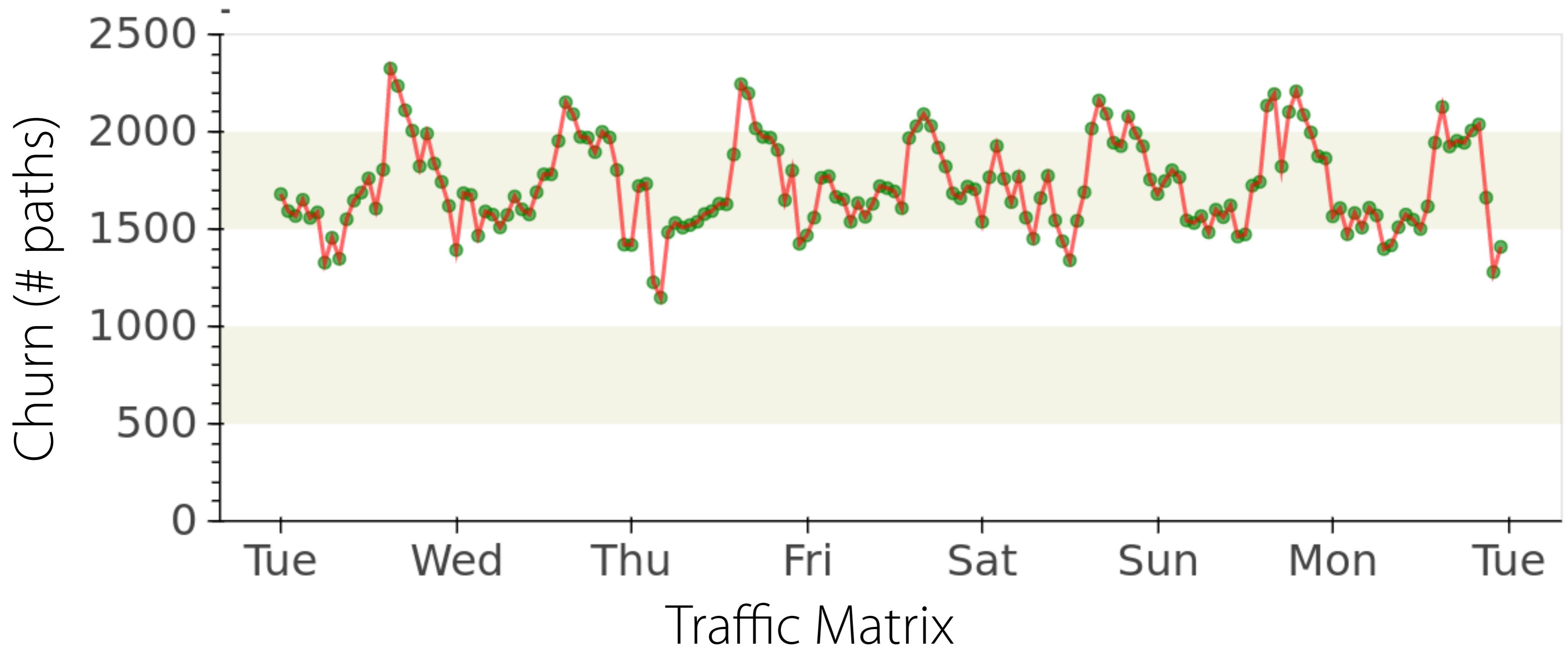
Operational Cost of Optimality

Solver Time



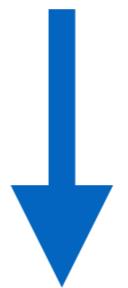
Operational Cost of Optimality

Path Churn



Towards a Practical Model

Topology
(+ demands)



Path
Selection

1

Paths

Demands

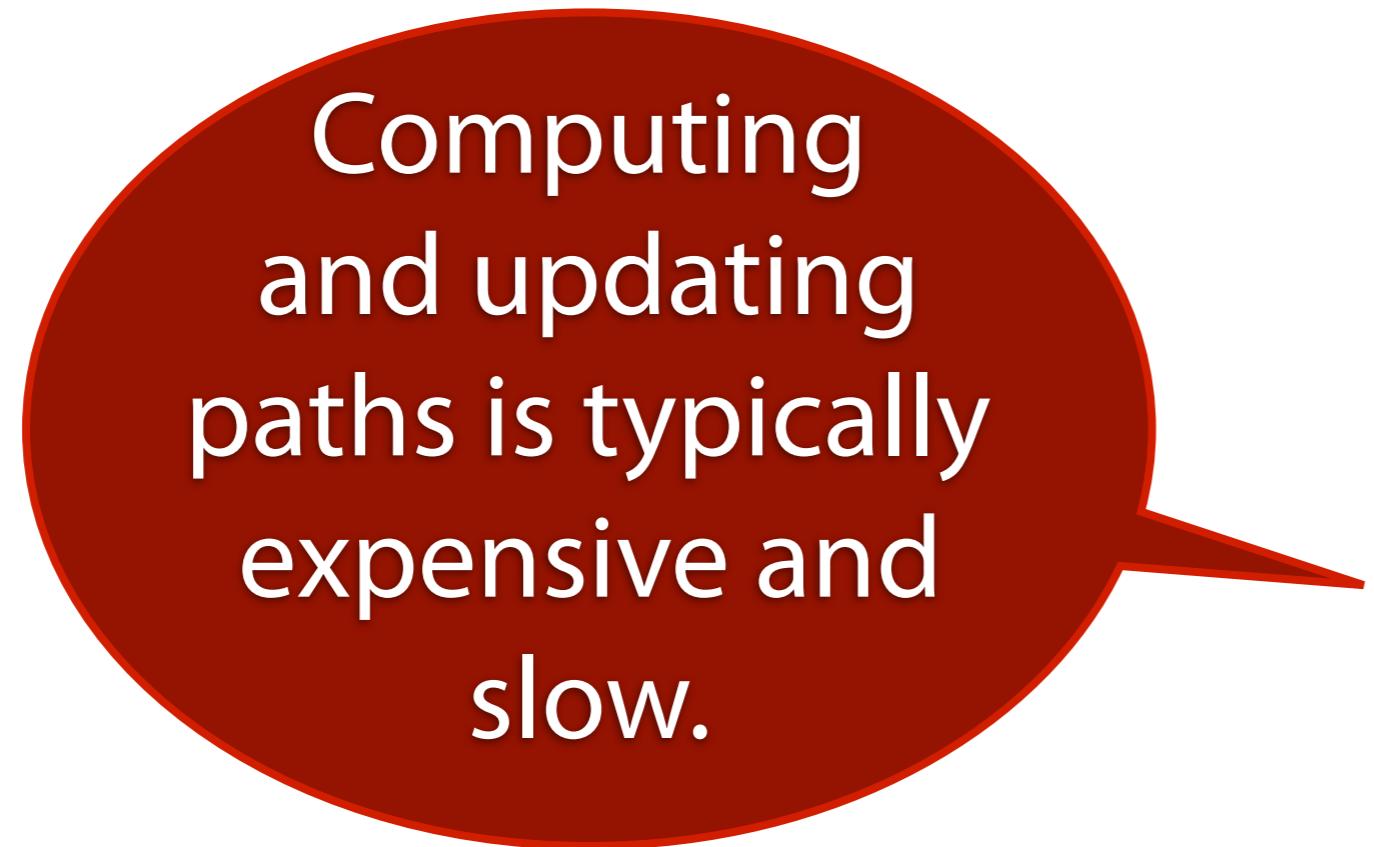


Rate
Adaptation

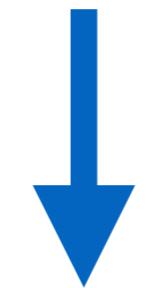
2

Splitting Ratio

Towards a Practical Model

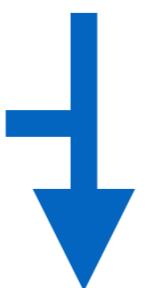


Topology
(+ demands)



Paths

Demands



Splitting Ratio

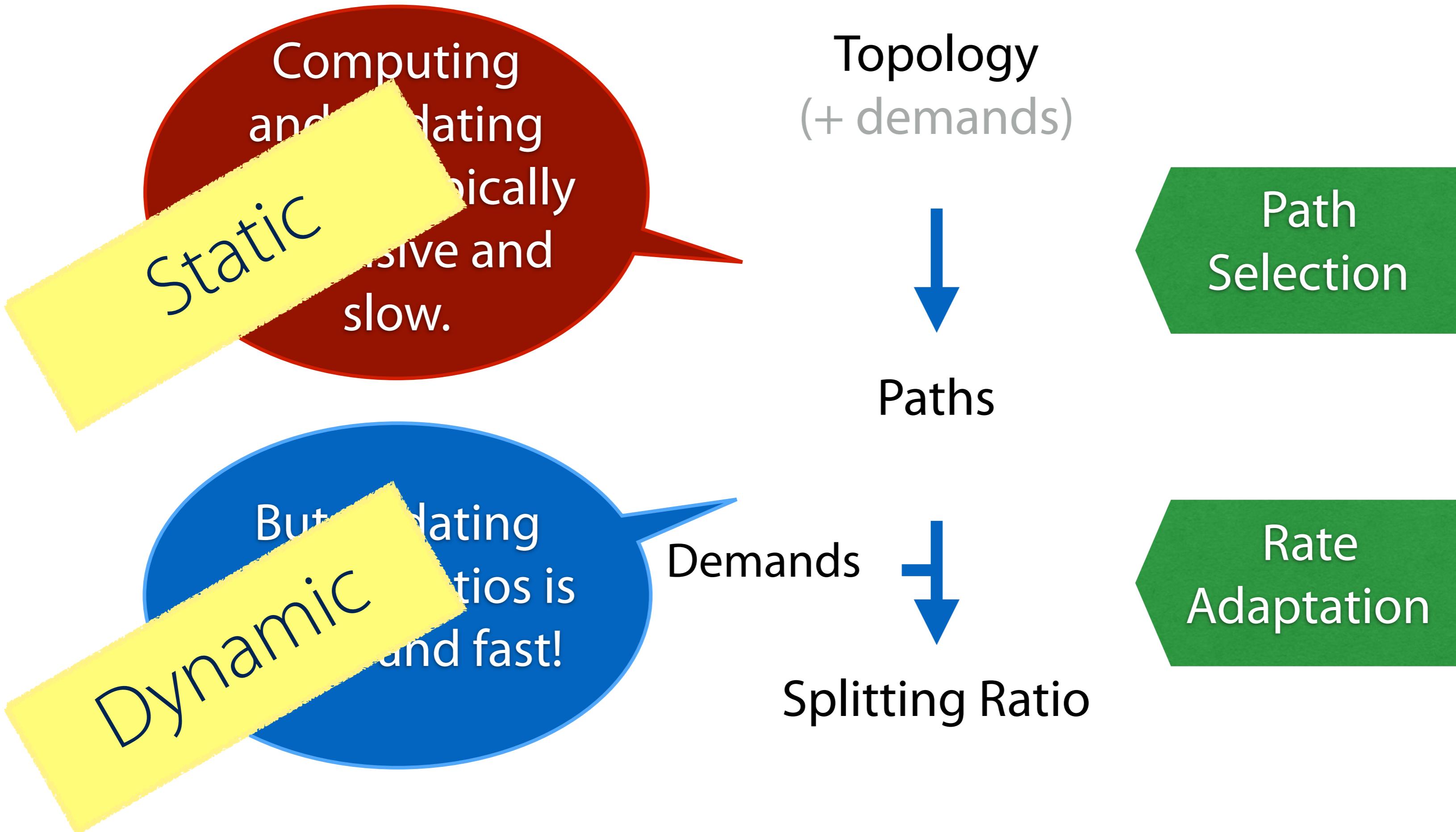
Path
Selection

1

Rate
Adaptation

2

Towards a Practical Model



Path Selection Challenges

- Selecting a good set of paths is tricky!
- Route the demands (ideally, with competitive latency)
- React to changes in demands (diurnal changes, traffic bursts, etc.)
- Be robust under mis-prediction of demands
- Have sufficient extra capacity to route demands in presence of failures
- and more ...

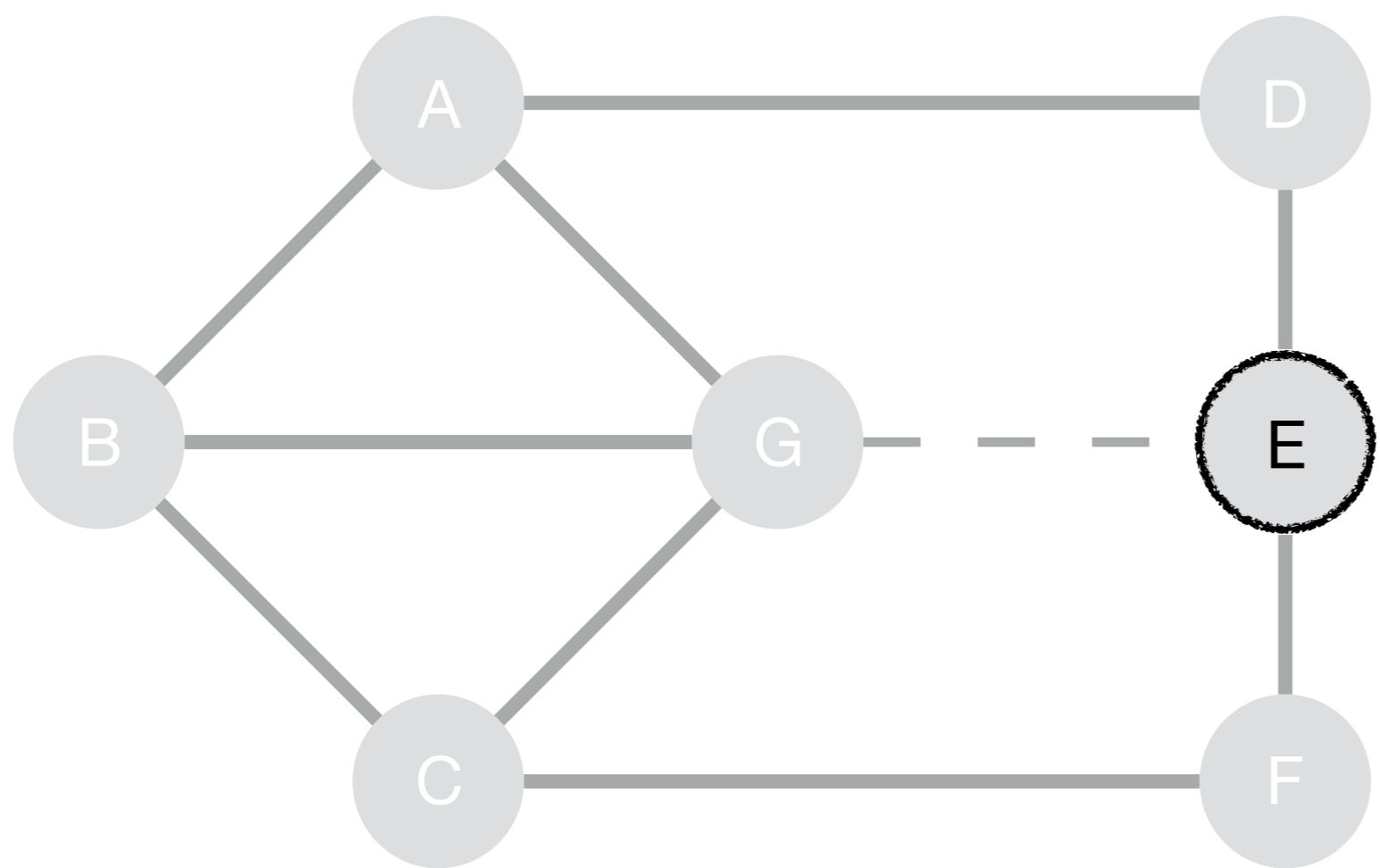
Approach

A **static** set of cleverly-constructed paths can provide near-optimal performance and robustness!

Desired path properties:

- ***Low stretch*** for minimizing latency
 - ***High diversity*** for ensuring robustness
 - ***Good load balancing*** for performance
- 
 - Capacity aware
 - Globally optimized

Path Properties: Capacity Aware



- Traditional approaches to routing based on shortest paths (e.g., ECMP, KSP) are generally not capacity aware

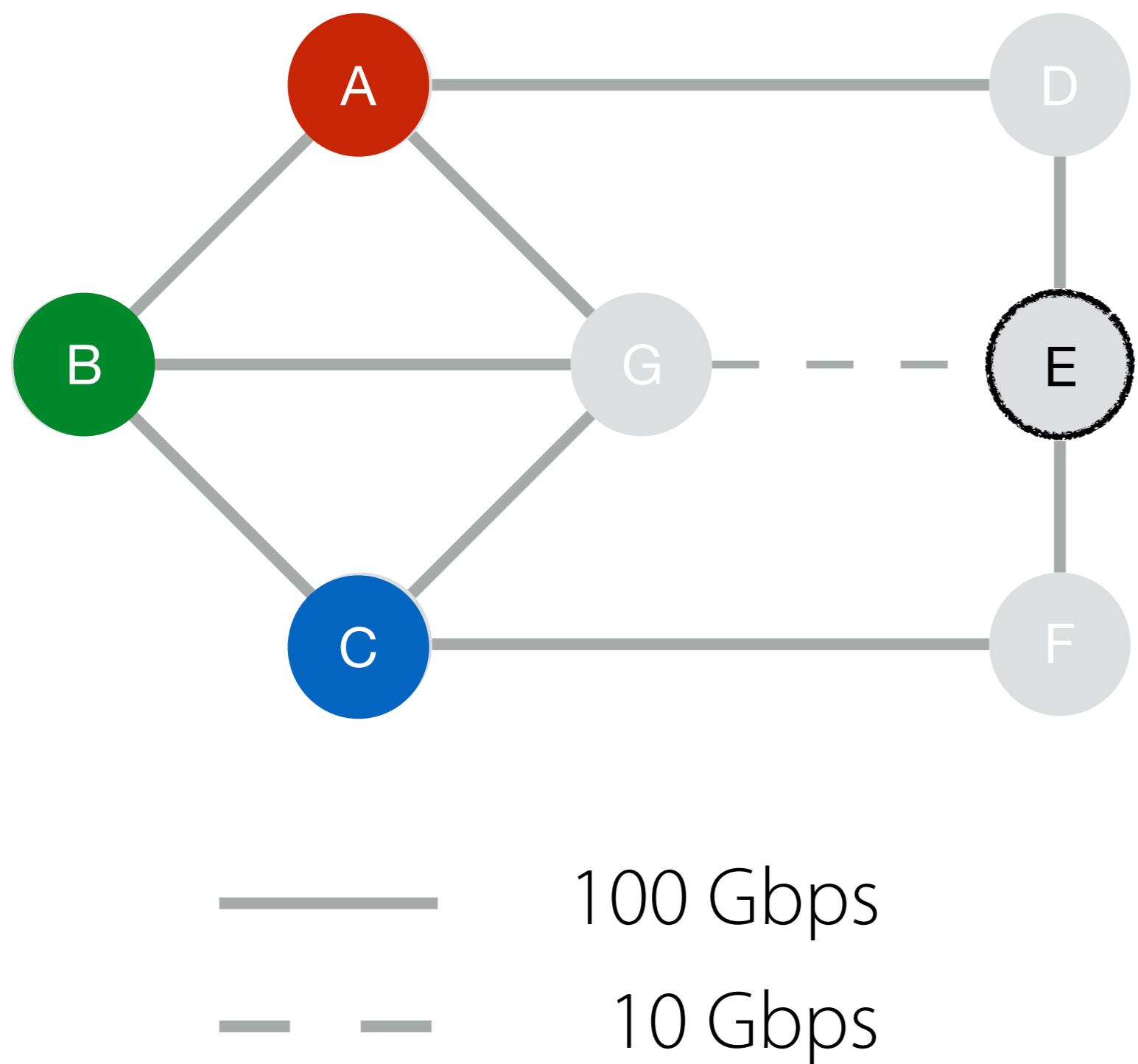


100 Gbps



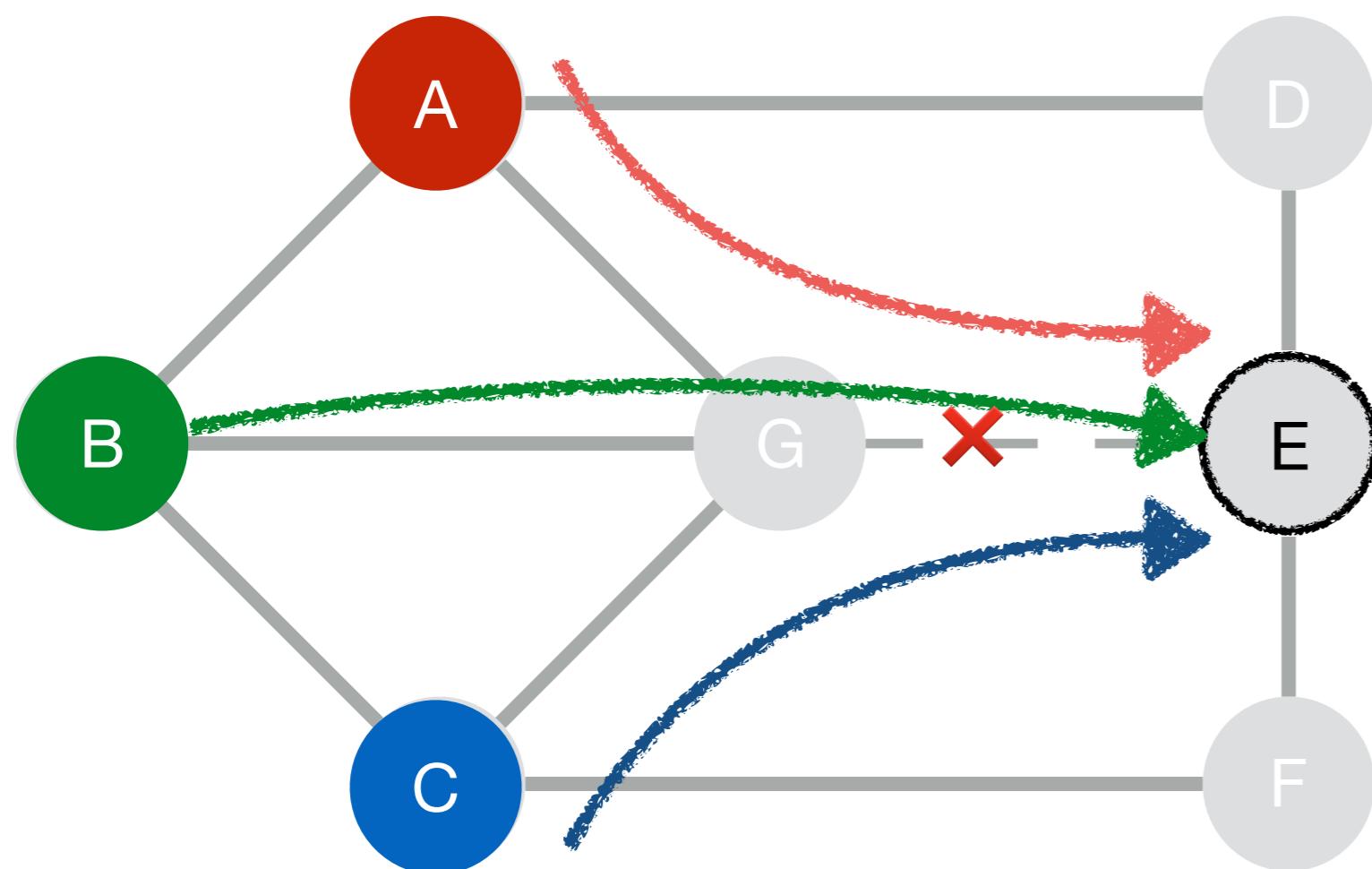
10 Gbps

Path Properties: Capacity Aware



- Traditional approaches to routing based on shortest paths (e.g., ECMP, KSP) are generally not capacity aware

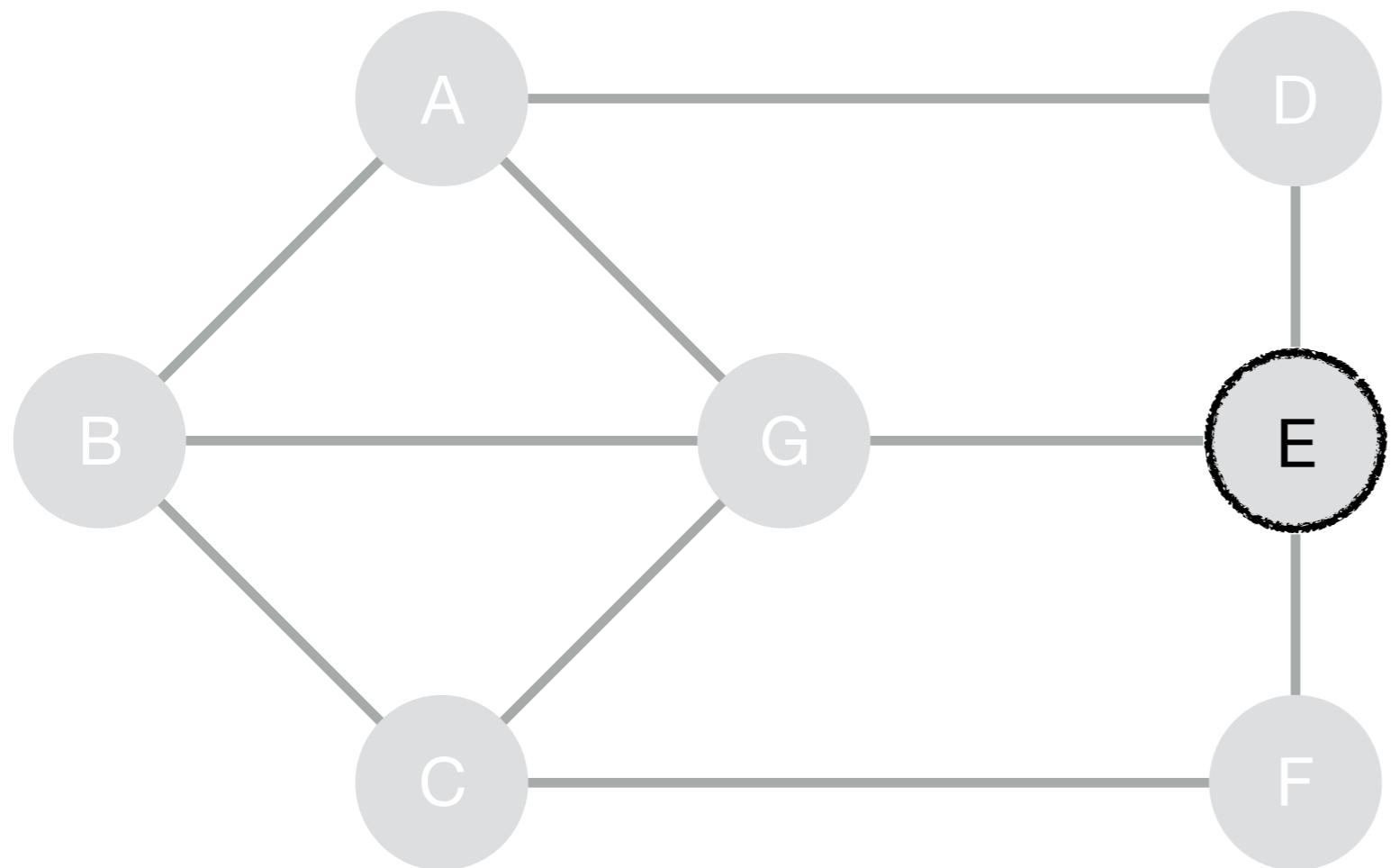
Path Properties: Capacity Aware



- Traditional approaches to routing based on shortest paths (e.g., ECMP, KSP) are generally not capacity aware

Path Properties: Globally Optimal

Other approaches based on greedy algorithms are capacity aware, but are still not globally optimal



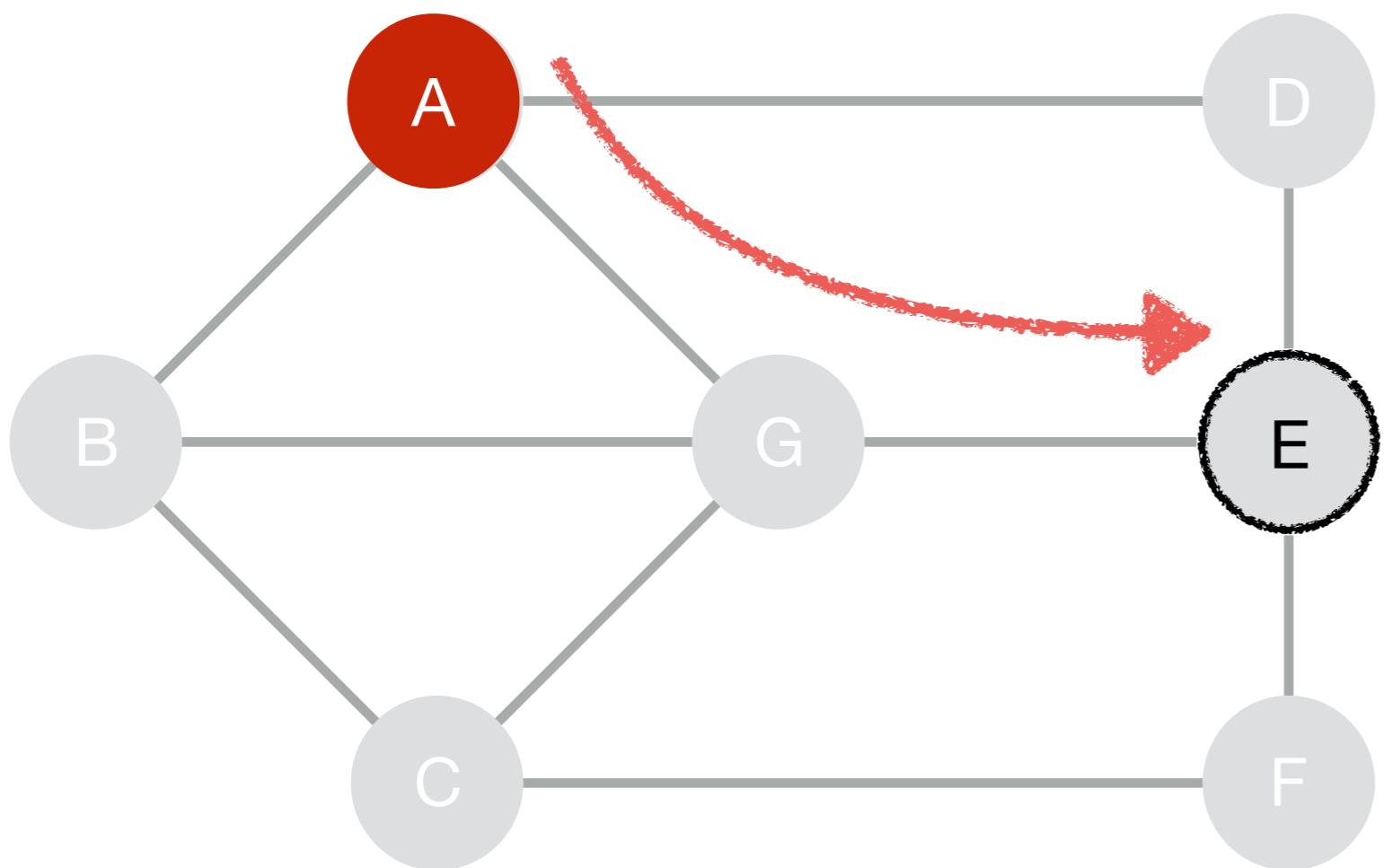
CSPF

•
•
•
•
•
•
•
•
•
•

Globally optimal

Path Properties: Globally Optimal

Other approaches based on greedy algorithms are capacity aware, but are still not globally optimal

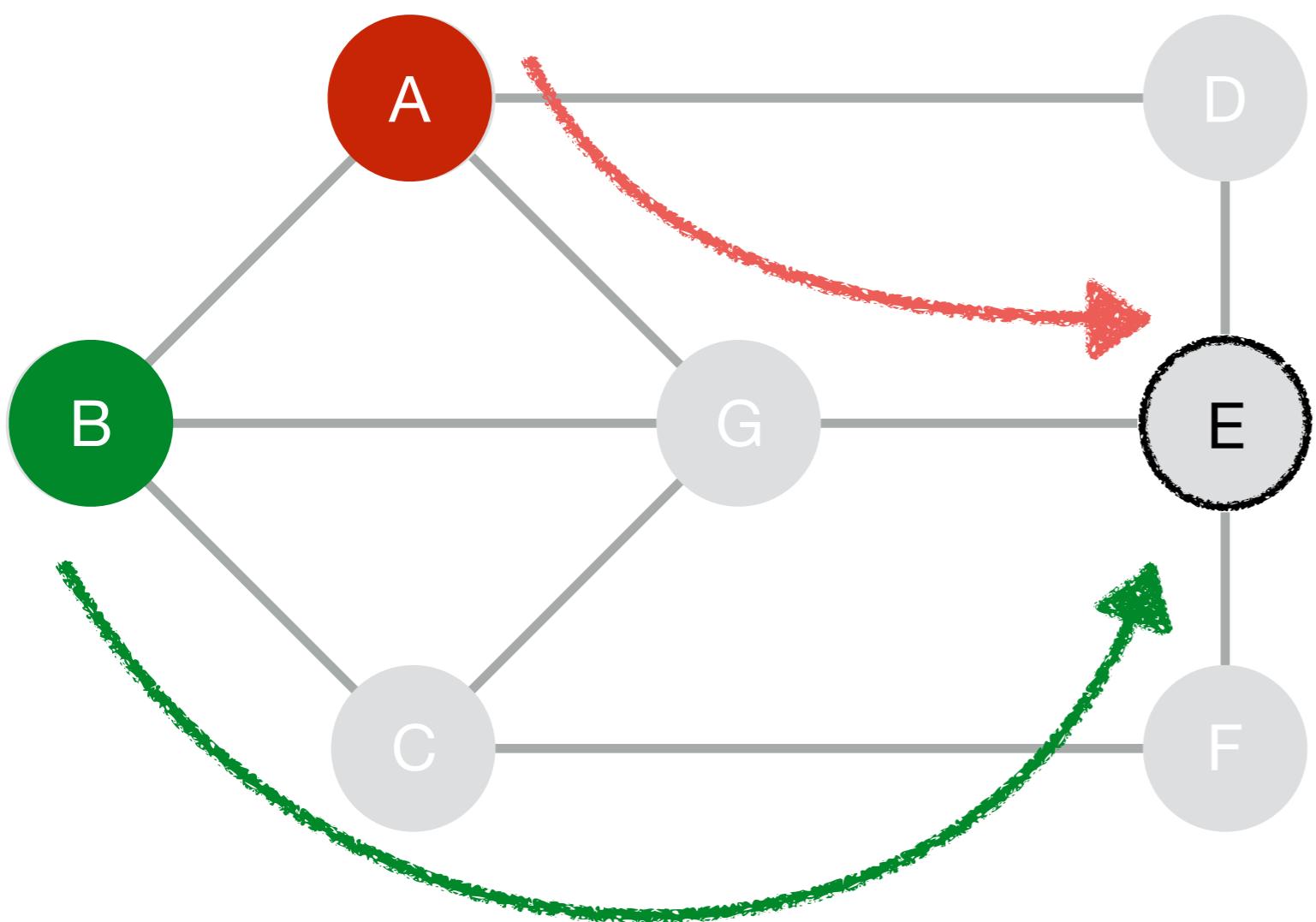


CSPF

Globally optimal

Path Properties: Globally Optimal

Other approaches based on greedy algorithms are capacity aware, but are still not globally optimal

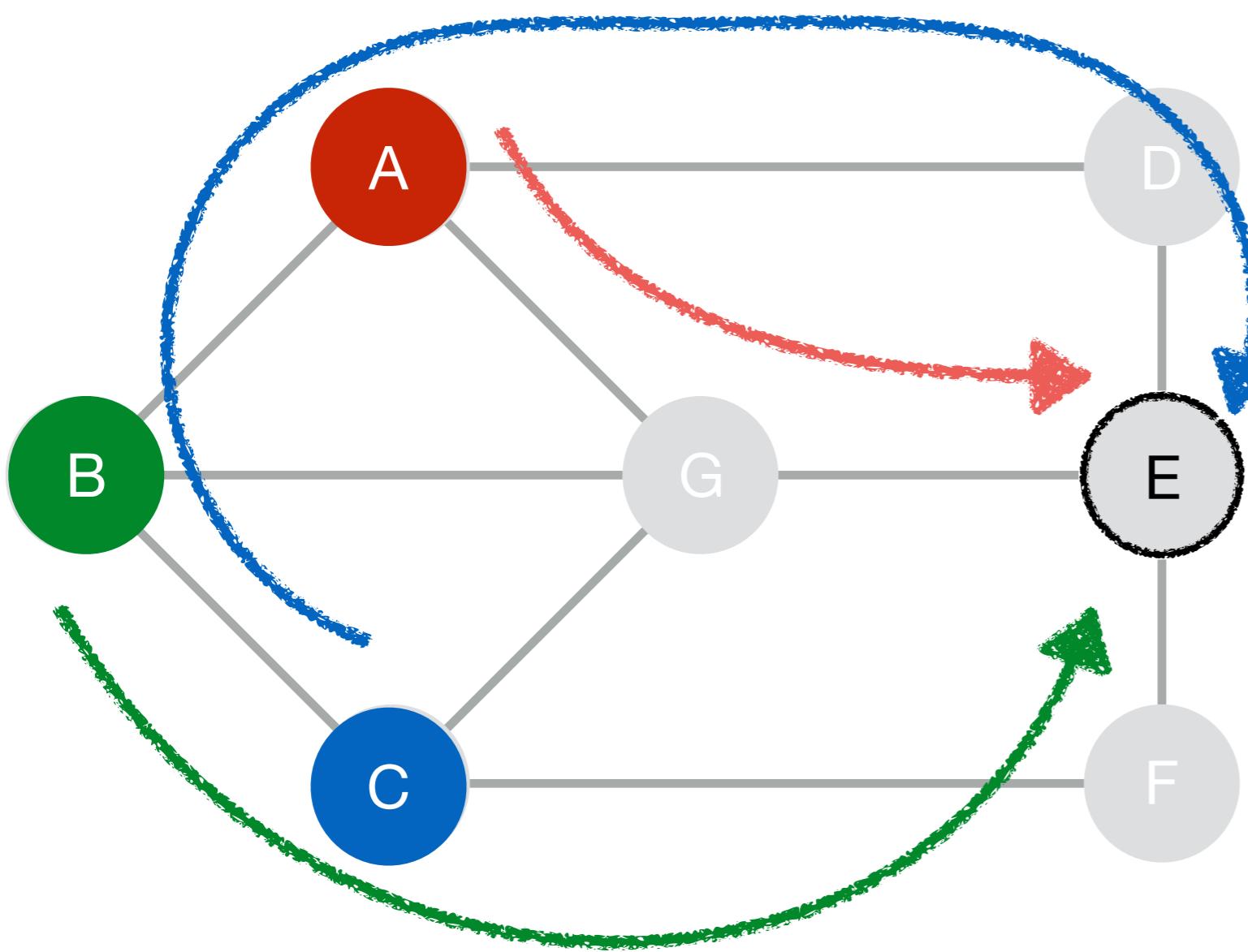


CSPF

Globally optimal

Path Properties: Globally Optimal

Other approaches based on greedy algorithms are capacity aware, but are still not globally optimal

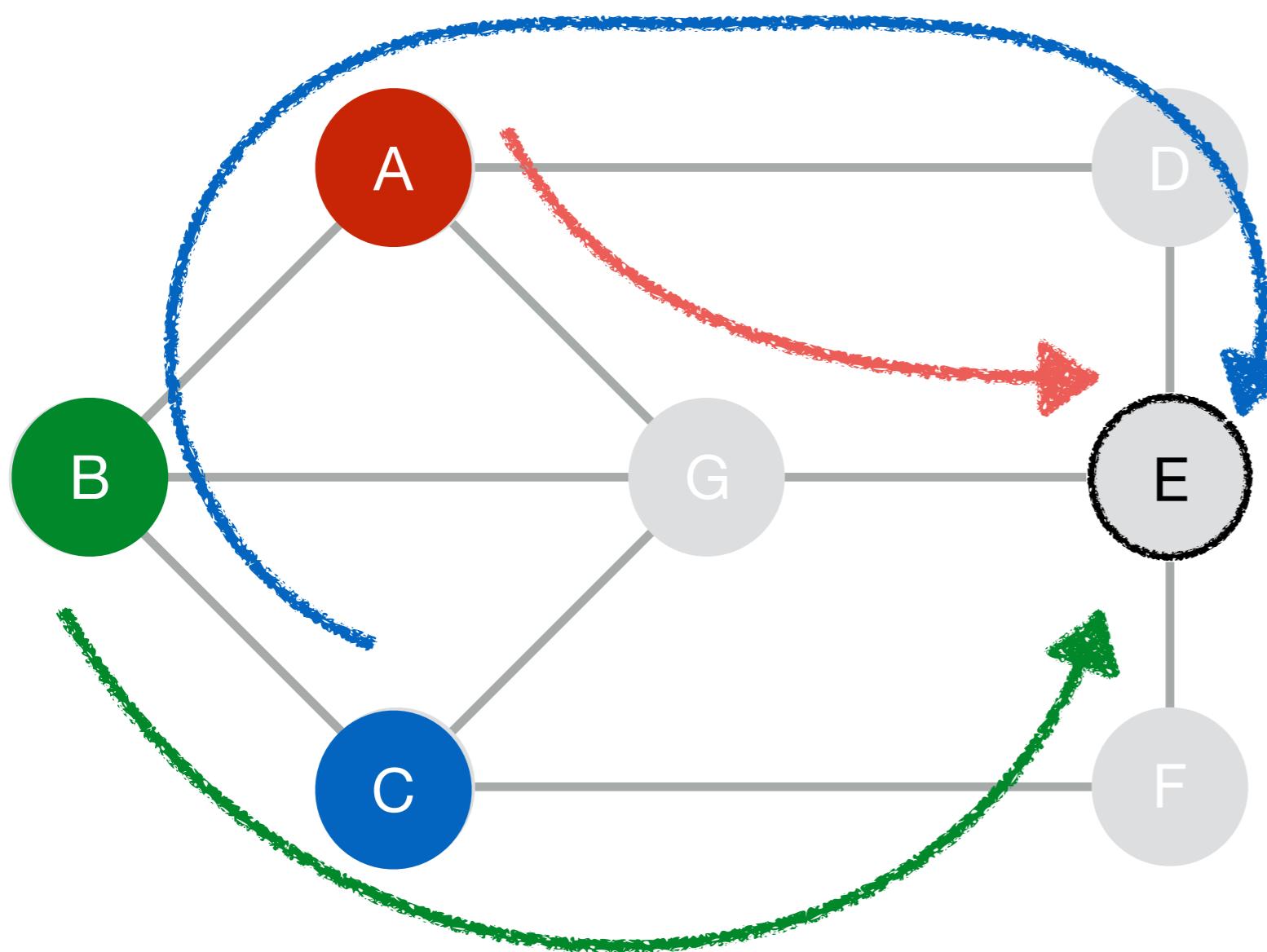


CSPF

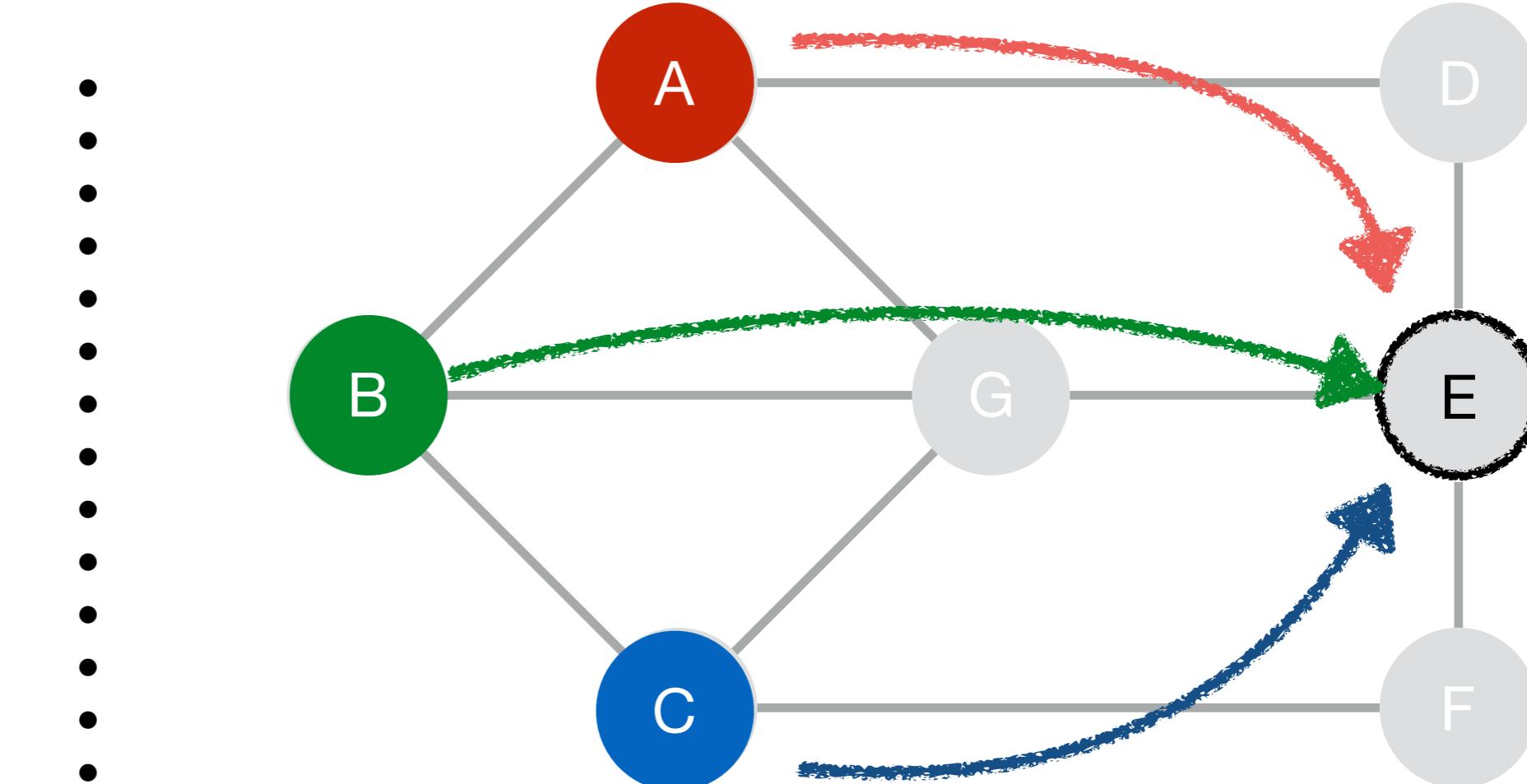
Globally optimal

Path Properties: Globally Optimal

Other approaches based on greedy algorithms are capacity aware, but are still not globally optimal



CSPF



Globally optimal

Path Selection

Algorithm	Load balanced			
	Capacity aware	Globally Optimized	Diverse	Low-stretch
SPF / ECMP	✗	✗	✗	✓
CSPF	✓	✗	✗	✓
k-shortest paths	✗	✗	?	✓
Edge-disjoint KSP	✗	✗	✓	✓
MCF	✓	✓	✗	✗
VLB	✗	✗	✓	✗
B4	✓	✓	✗	?

? - Difficult to generalize

Path Selection

Algorithm	Load balanced			
	Capacity aware	Globally Optimized	Diverse	Low-stretch
SPF / ECMP	✗	✗	✗	✓
CSPF	✓	✗	✗	✓
k-shortest paths	✗	✗	?	✓
Edge-disjoint KSP	✗	✗	✓	✓
MCF	✓	✓	✗	✗
VLB	✗	✗	✓	✗
B4	✓	✓	✗	?

? - Difficult to generalize

Path Selection

Algorithm	Load balanced			
	Capacity aware	Globally Optimized	Diverse	Low-stretch
SPF / ECMP	✗	✗	✗	✓
CSPF	✓	✗	✗	✓
k-shortest paths	✗	✗	?	✓
Edge-disjoint KSP	✗	✗	✓	✓
MCF	✓	✓	✗	✗
VLB	✗	✗	✓	✗
B4	✓	✓	✗	?

? - Difficult to generalize

Path Selection

Algorithm	Load balanced			
	Capacity aware	Globally Optimized	Diverse	Low-stretch
SPF / ECMP	✗	✗	✗	✓
CSPF	✓	✗	✗	✓
k-shortest paths	✗	✗	?	✓
Edge-disjoint KSP	✗	✗	✓	✓
MCF	✓	✓	✗	✗
VLB	✗	✗	✓	✗
B4	✓	✓	✗	?

? - Difficult to generalize

Path Selection

Algorithm	Load balanced			
	Capacity aware	Globally Optimized	Diverse	Low-stretch
SPF / ECMP	✗	✗	✗	✓
CSPF	✓	✗	✗	✓
k-shortest paths	✗	✗	?	✓
Edge-disjoint KSP	✗	✗	✓	✓
MCF	✓	✓	✗	✗
VLB	✗	✗	✓	✗
B4	✓	✓	✗	?

? - Difficult to generalize

Path Selection

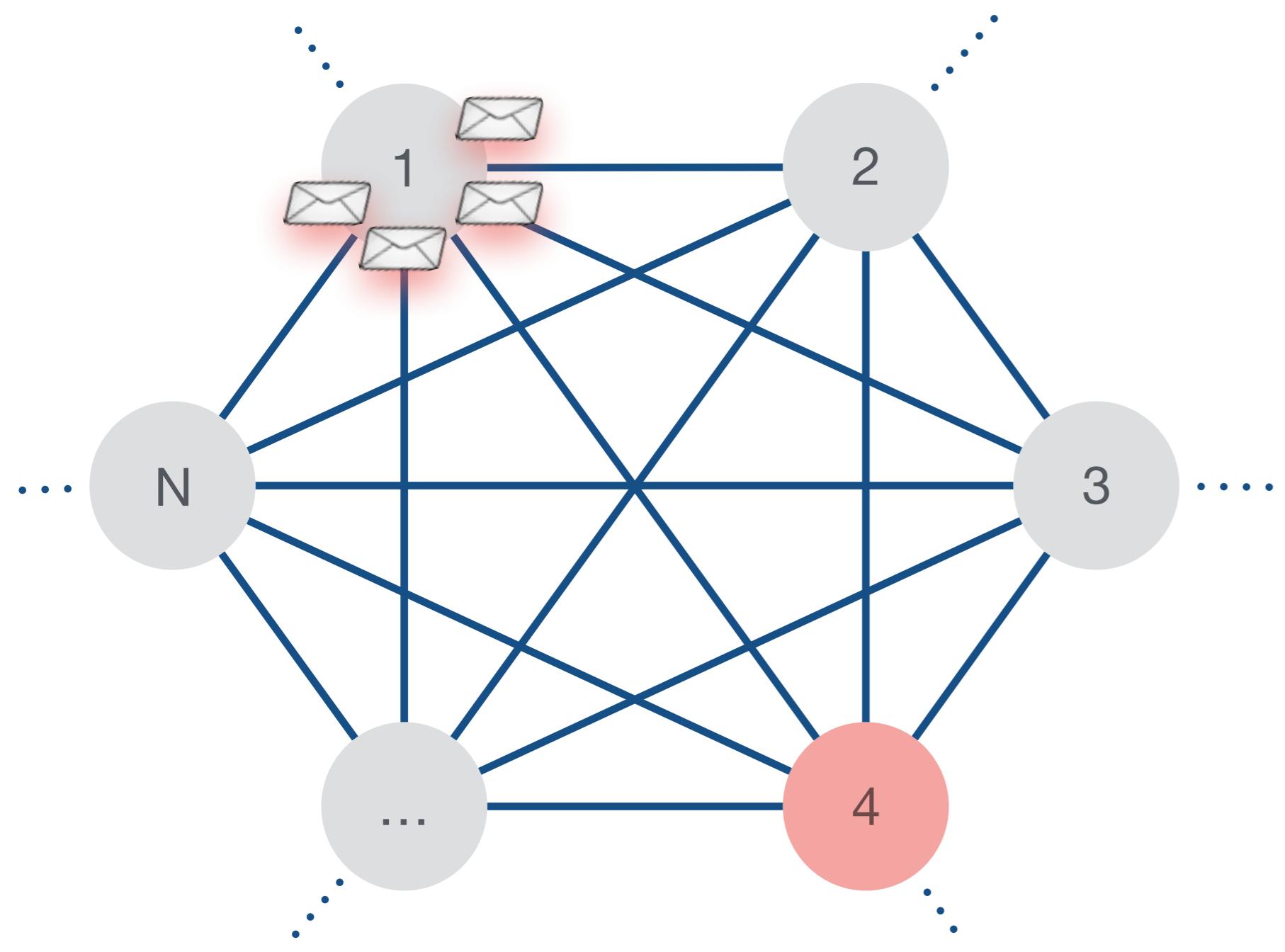
Algorithm	Load balanced			
	Capacity aware	Globally Optimized	Diverse	Low-stretch
SPF / ECMP	✗	✗	✗	✓
CSPF	✓	✗	✗	✓
k-shortest paths	✗	✗	?	✓
Edge-disjoint KSP	✗	✗	✓	✓
MCF	✓	✓	✗	✗
VLB	✗	✗	✓	✗
B4	✓	✓	✗	?

? - Difficult to generalize

Oblivious Routing

VLB

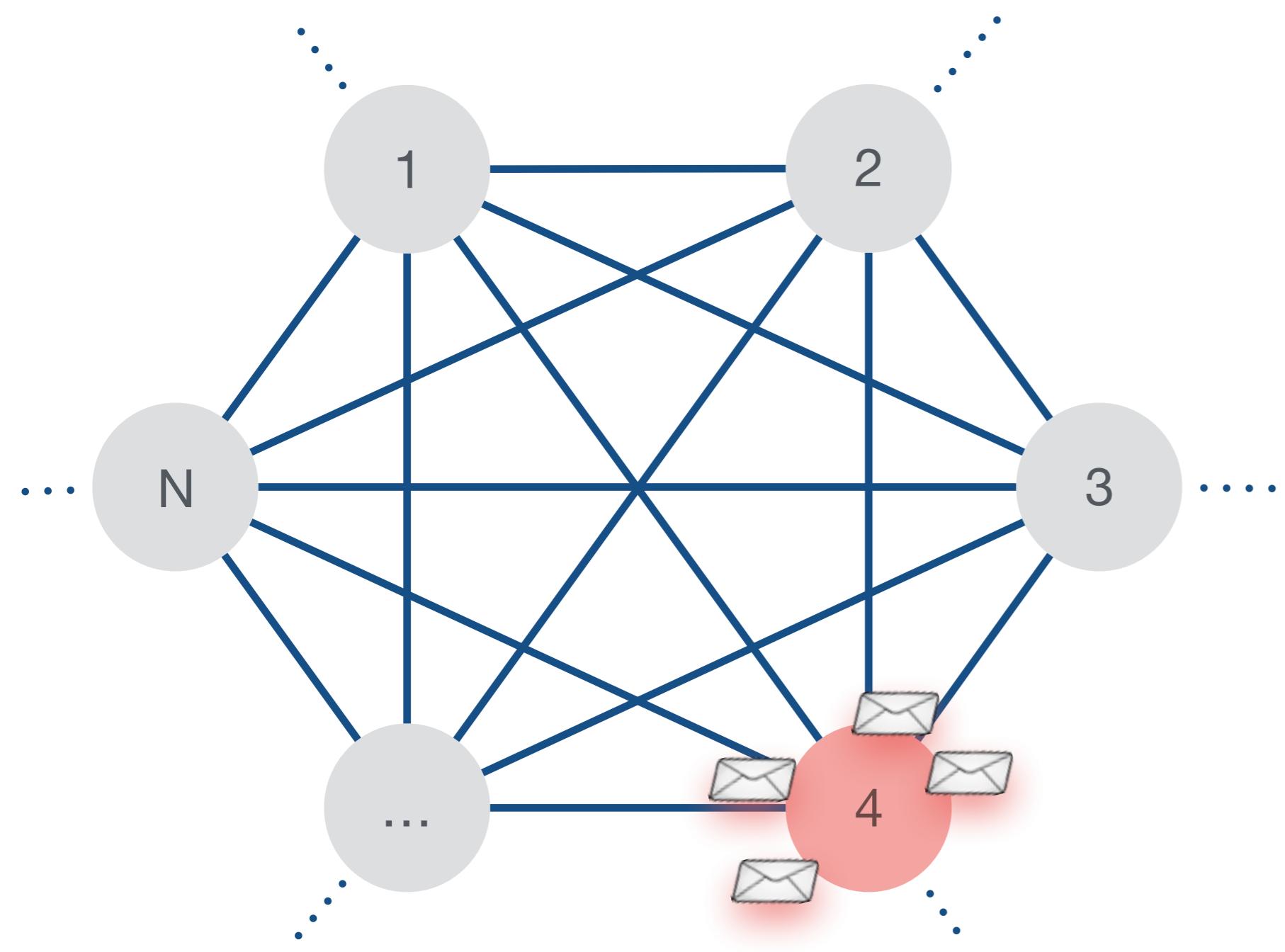
Mesh



- Route through **random** intermediate node
- Works well for **mesh** topologies
- WANs are not mesh-like
 - Good resilience
 - Poor performance & latency

VLB

Mesh



- Route through **random** intermediate node
- Works well for **mesh** topologies
- WANs are not mesh-like
 - Good resilience
 - Poor performance & latency

VLB

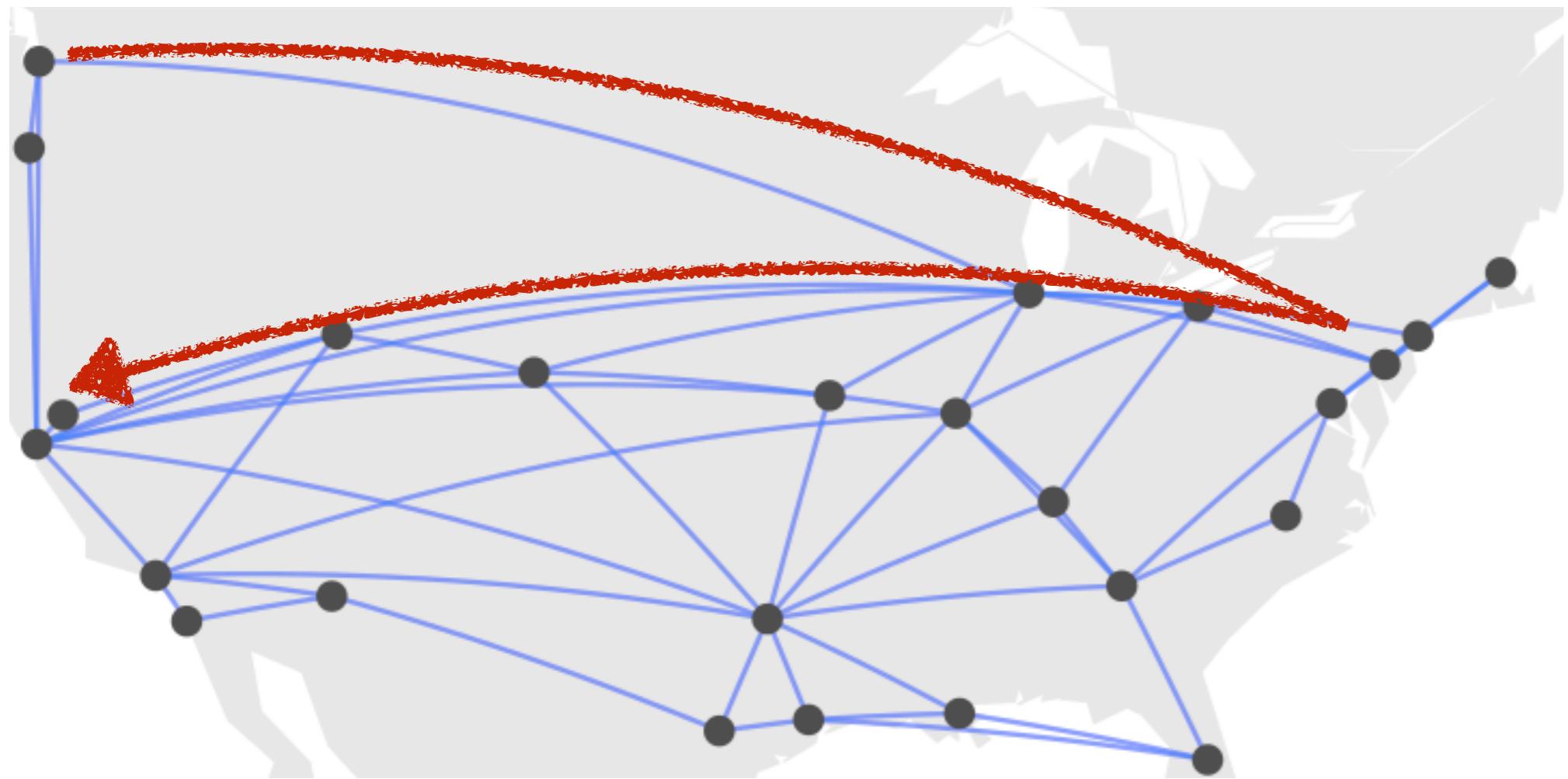
Not Mesh



- Route through random intermediate node
- Works well for mesh topologies
- WANs are not mesh-like
 - Good resilience
 - Poor performance & latency

VLB

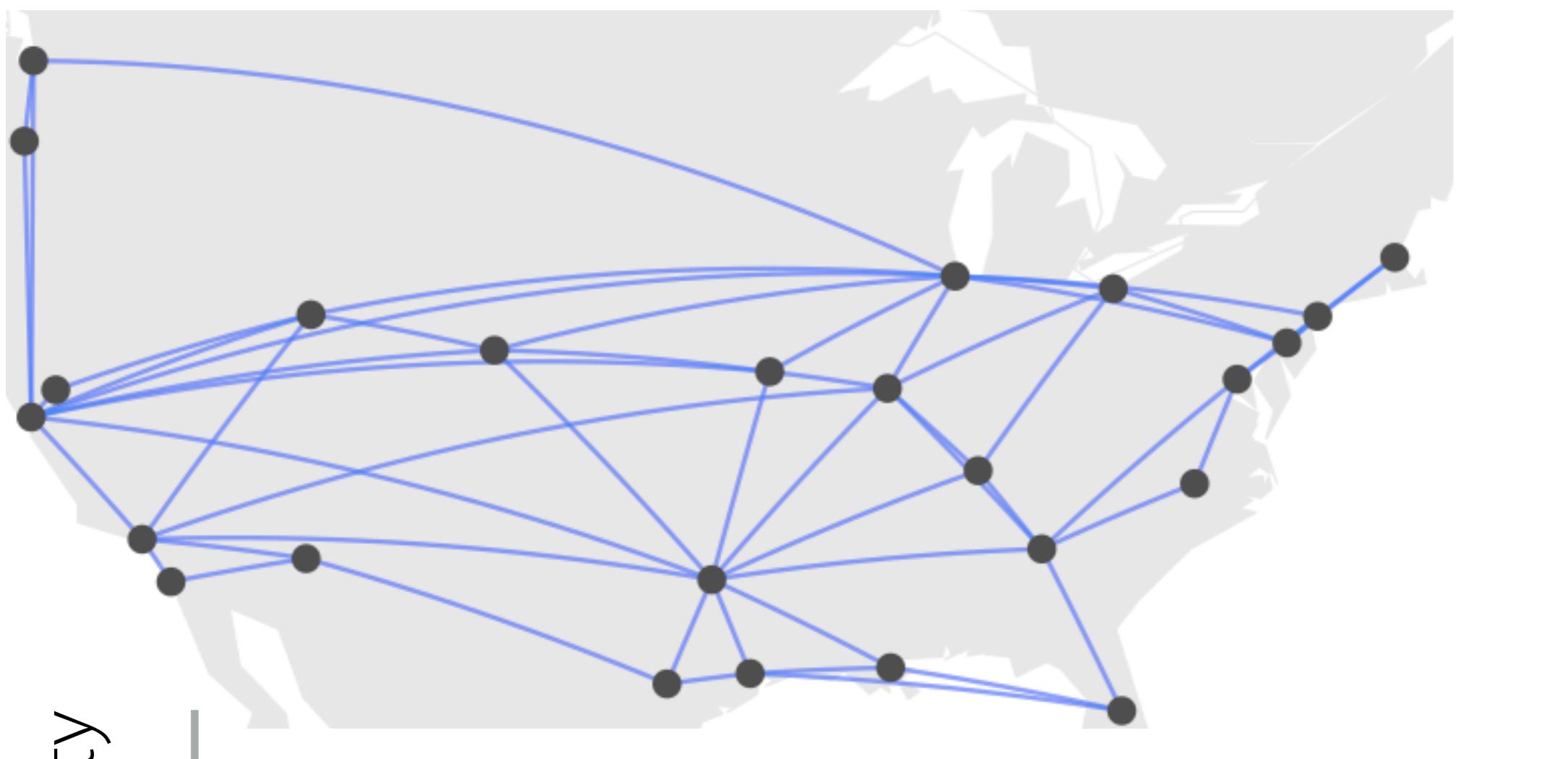
Not Mesh



- Route through random intermediate node
- Works well for mesh topologies
- WANs are not mesh-like
 - Good resilience
 - Poor performance & latency

Oblivious [Räcke '08]

Not Mesh



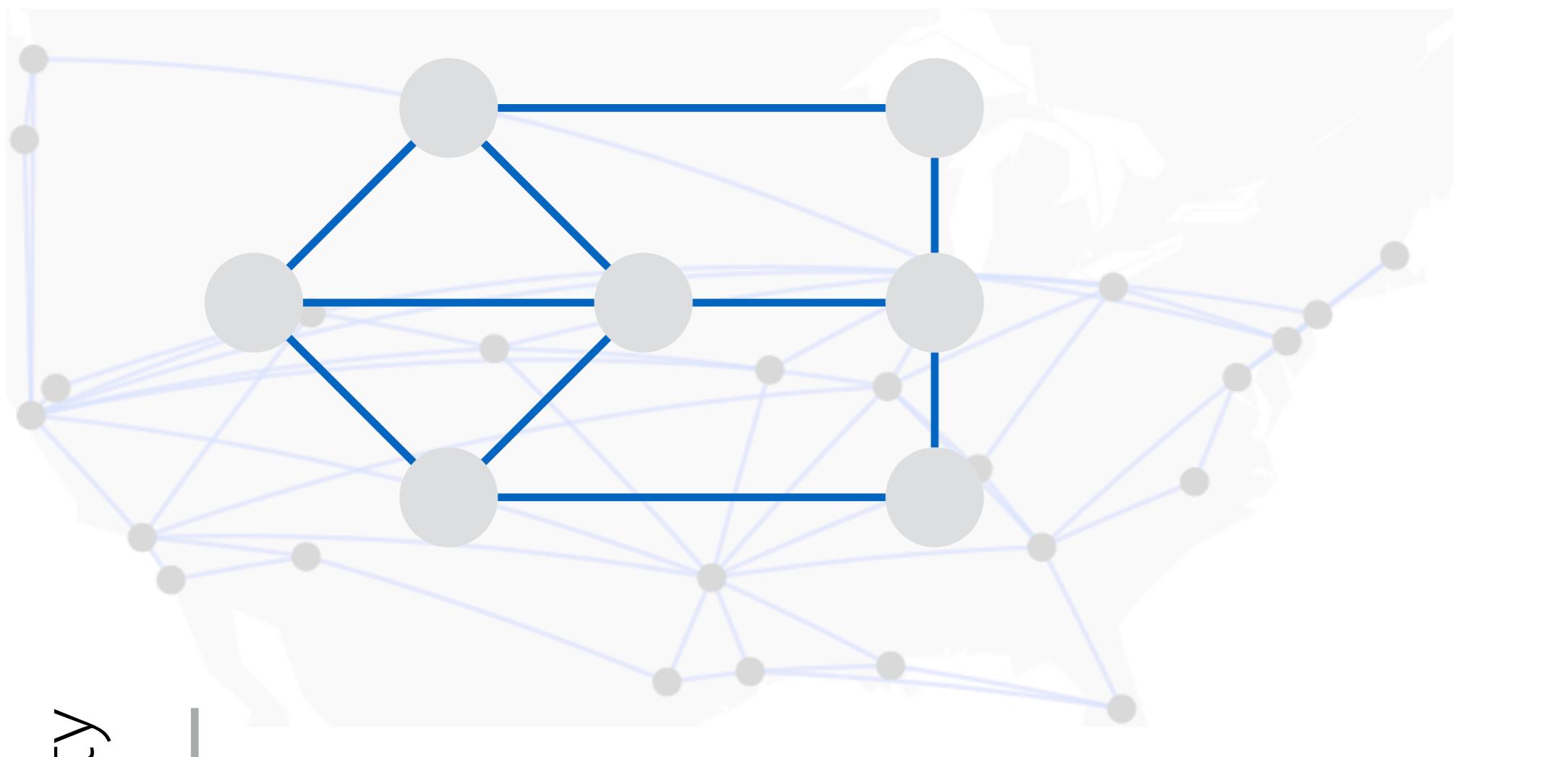
Probability

Low-stretch routing trees

- Generalizes VLB to non-mesh
- Distribution over routing trees
 - Approximation algorithm for low-stretch trees [FRT '04]
 - Penalize links based on usage
- $O(\log n)$ competitive

Oblivious [Räcke '08]

Not Mesh



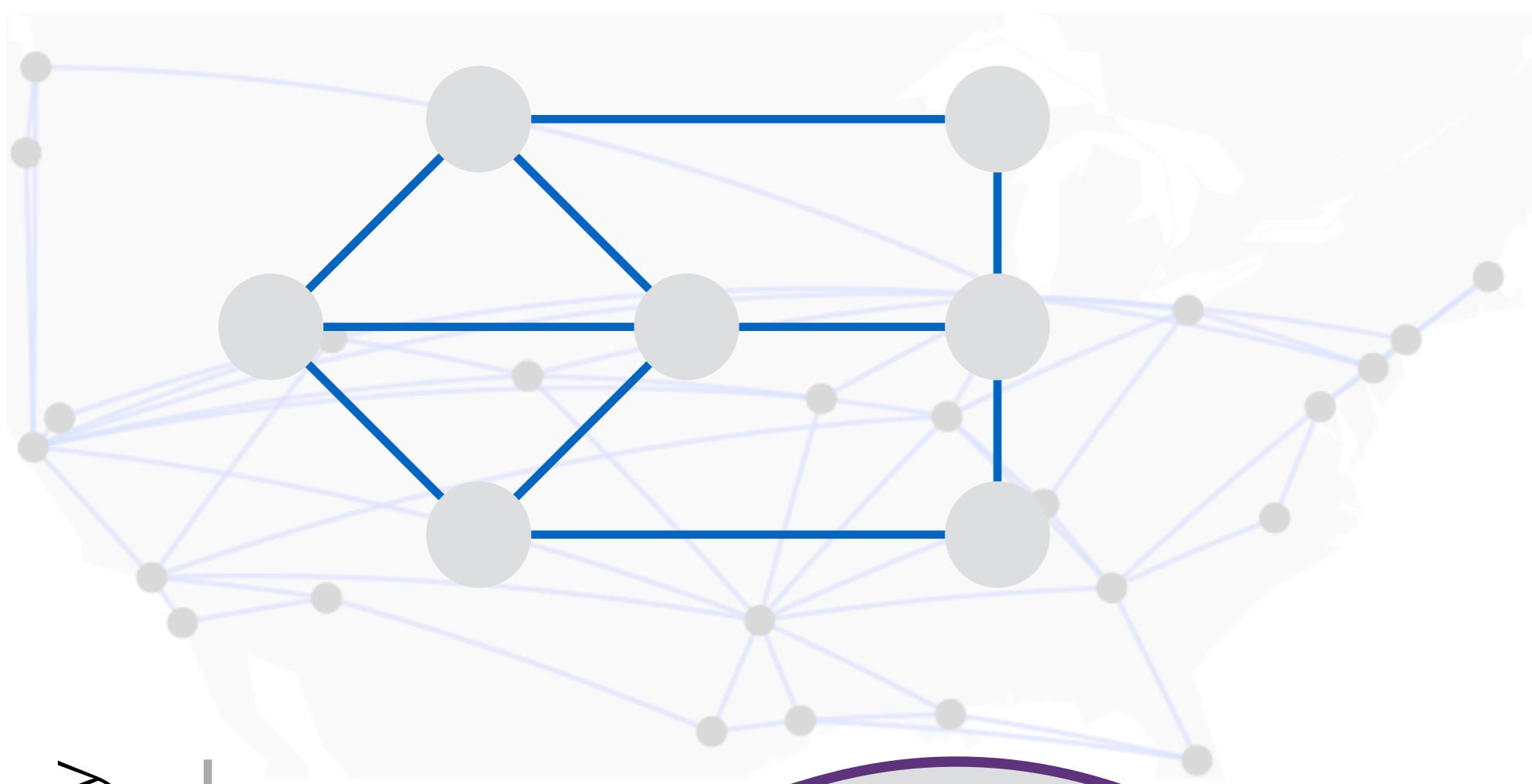
Probability

Low-stretch routing trees

- Generalizes VLB to non-mesh
- Distribution over routing trees
 - Approximation algorithm for low-stretch trees [FRT '04]
 - Penalize links based on usage
- $O(\log n)$ competitive

Oblivious [Räcke '08]

Not Mesh



Probability

Low-stretch routing trees

- Generalizes VLB to non-mesh
- Distribution over routing trees
 - Approximation algorithm for low-stretch trees [FRT '04]
- Penalize links based on usage
- $O(\log n)$ competitive

Path Selection

Algorithm	Load balanced			
	Capacity aware	Globally Optimized	Diverse	Low-stretch
SPF / ECMP	✗	✗	✗	✓
CSPF	✓	✗	✗	✓
k-shortest paths	✗	✗	?	✓
Edge-disjoint KSP	✗	✗	✓	✓
MCF	✓	✓	✗	✗
VLB	✗	✗	✓	✗
B4	✓	✓	✗	?
SMORE / Oblivious	✓	✓	✓	✓

SMORE: Semi-Oblivious Routing

Oblivious Routing computes a set of paths which are low-stretch, robust and have good load balancing properties

Path Selection

LP Optimizer balances load by dynamically adjusting splitting ratios used to map incoming traffic flows to paths

Rate Adaptation

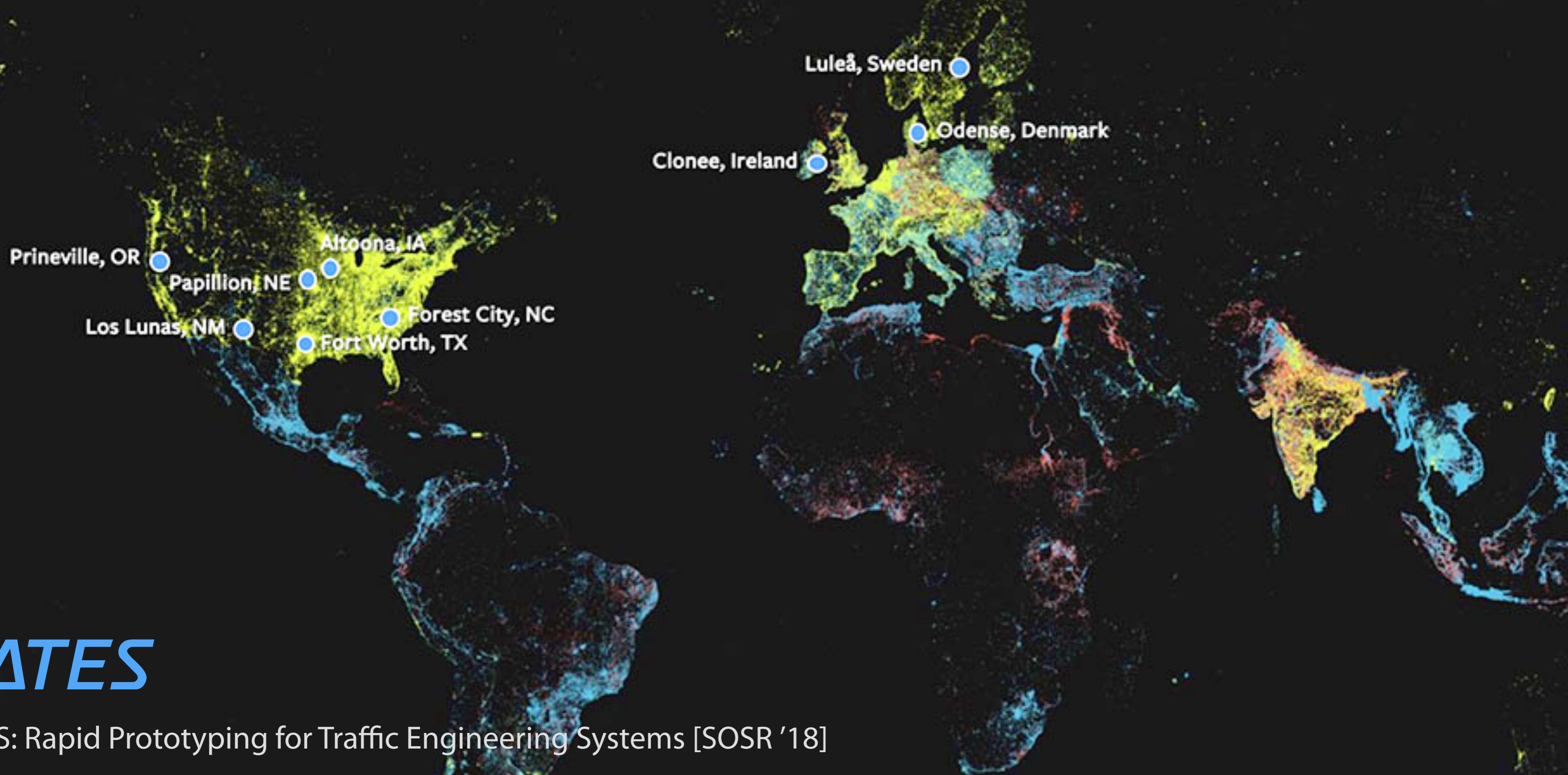
Semi-Oblivious Routing in Practice?

-  Previous work [Hajiaghayi et al.] established a worst-case competitive ratio that is not much better than oblivious routing: $\Omega(\log(n)/\log \log(n))$
-  But the real-world does not typically exhibit worst-case scenarios
-  Implicit correlation between demands and link capacities

Question: How well does semi-oblivious routing perform in practice?

Evaluation

Facebook's Backbone Network

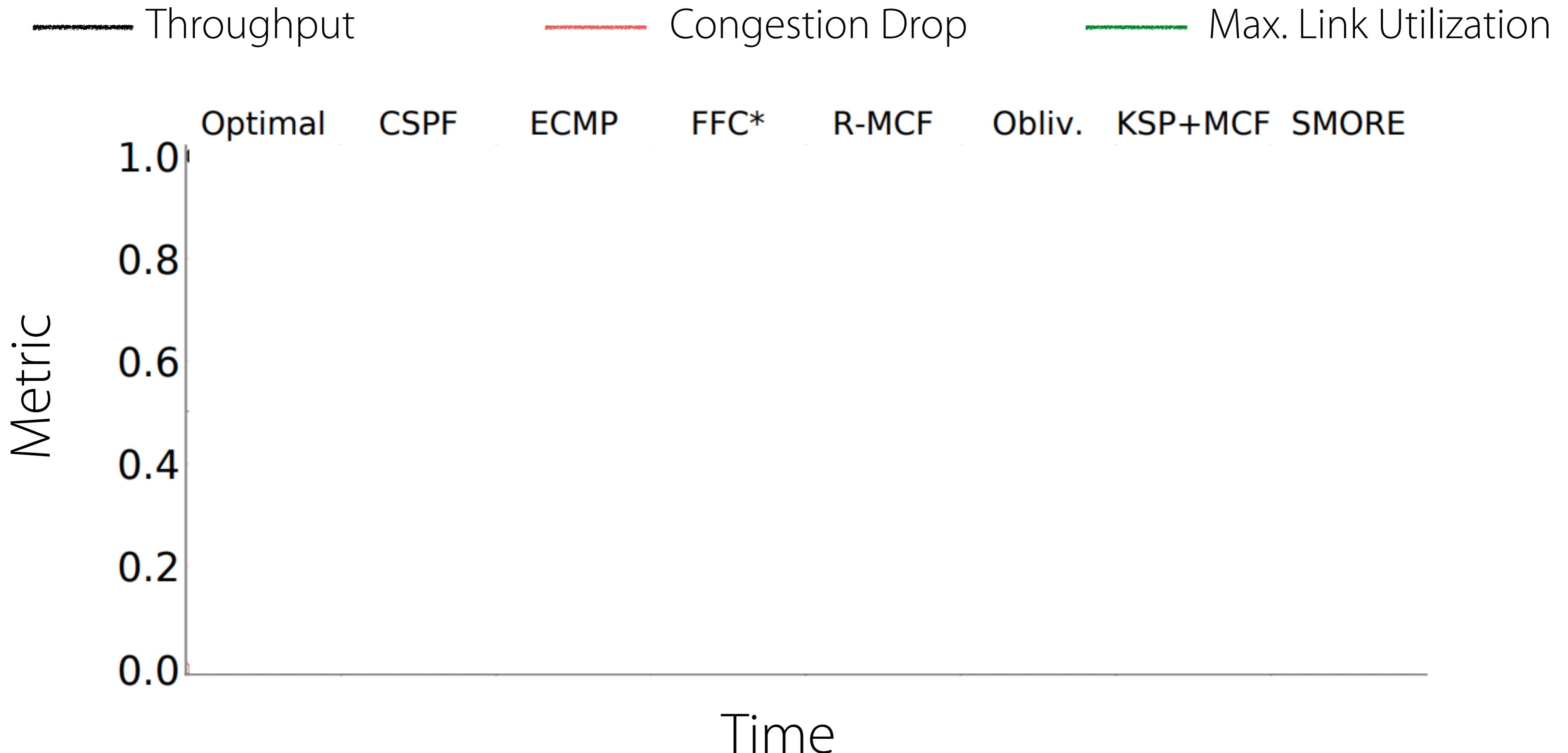


YATES

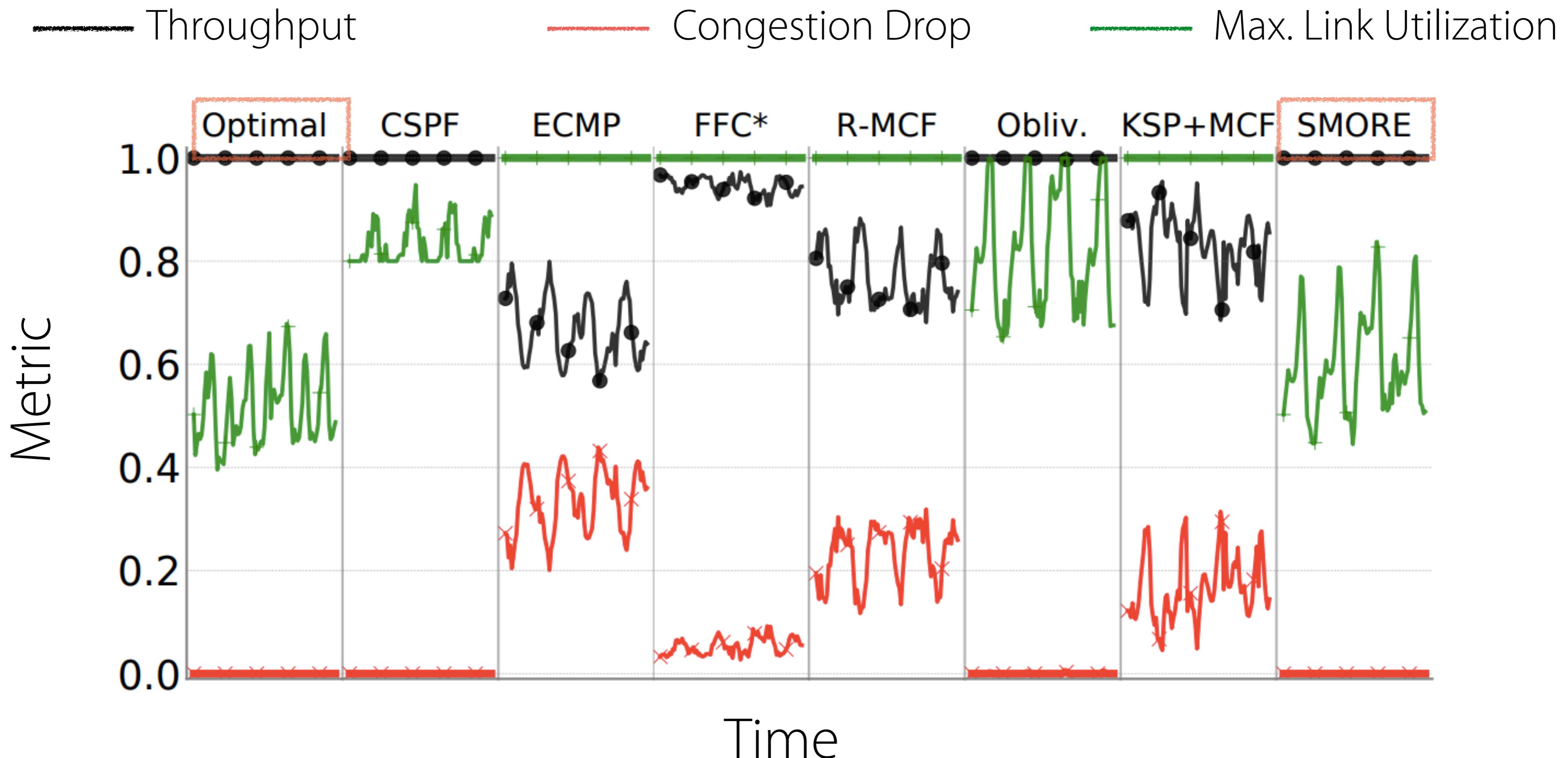
YATES: Rapid Prototyping for Traffic Engineering Systems [SOSR '18]

Source: <https://research.fb.com/robust-and-efficient-traffic-engineering-with-oblivious-routing/>

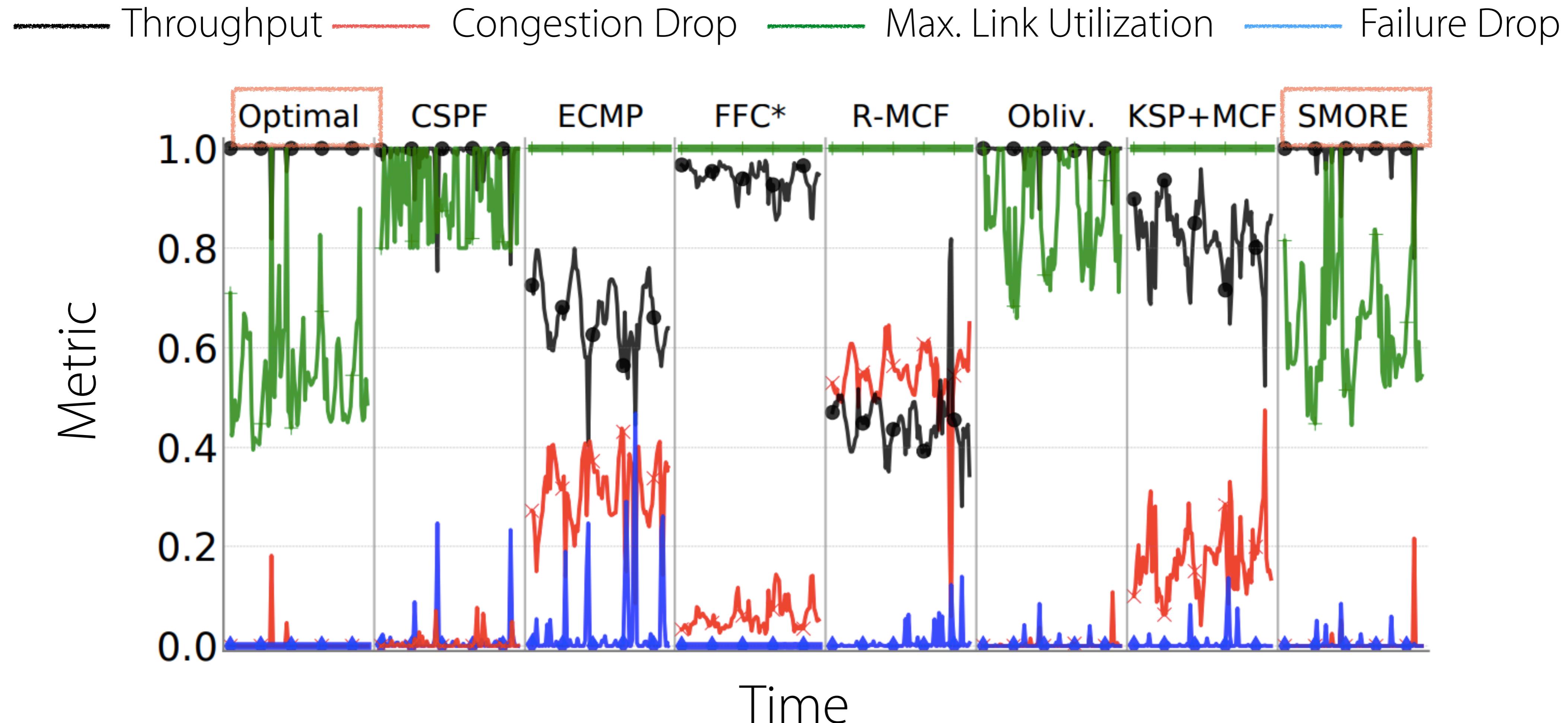
Performance



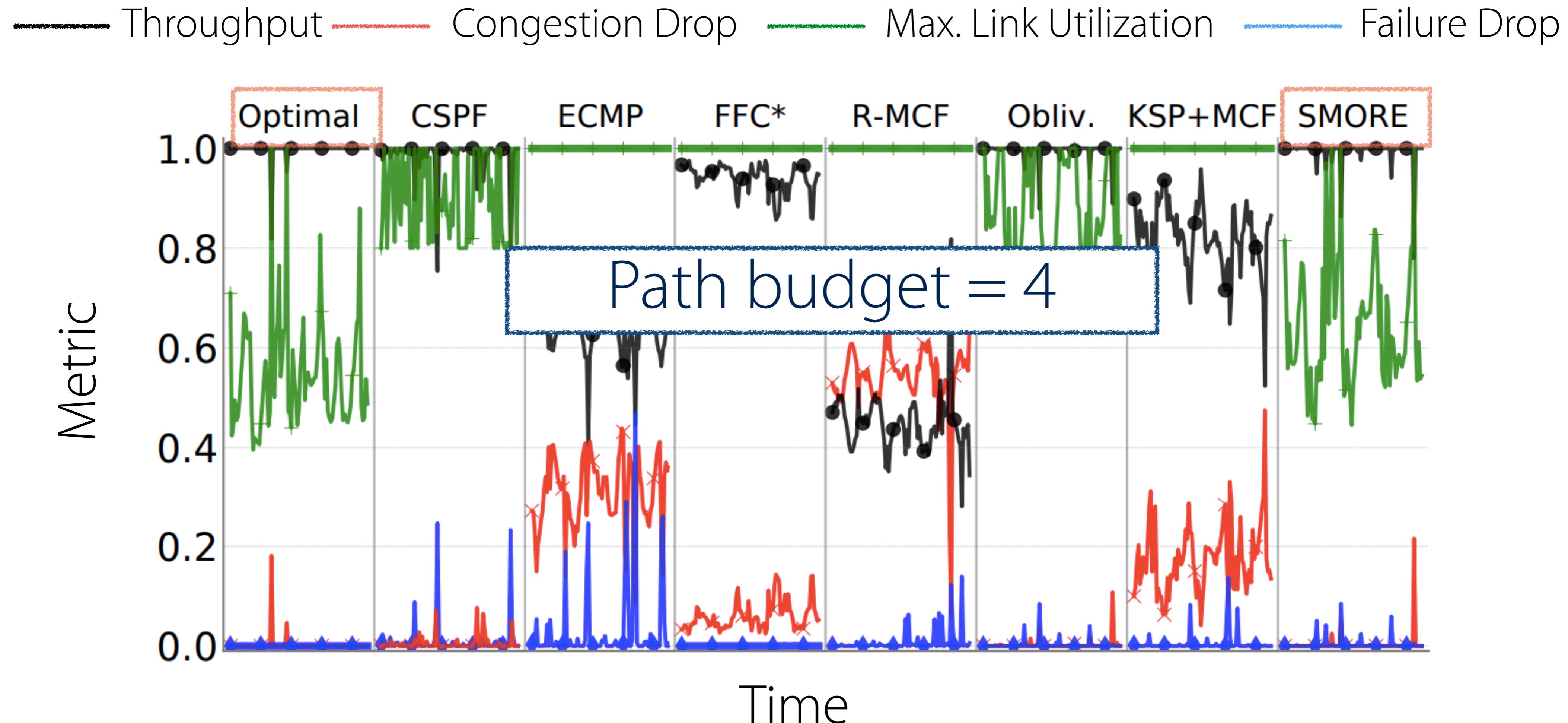
Performance



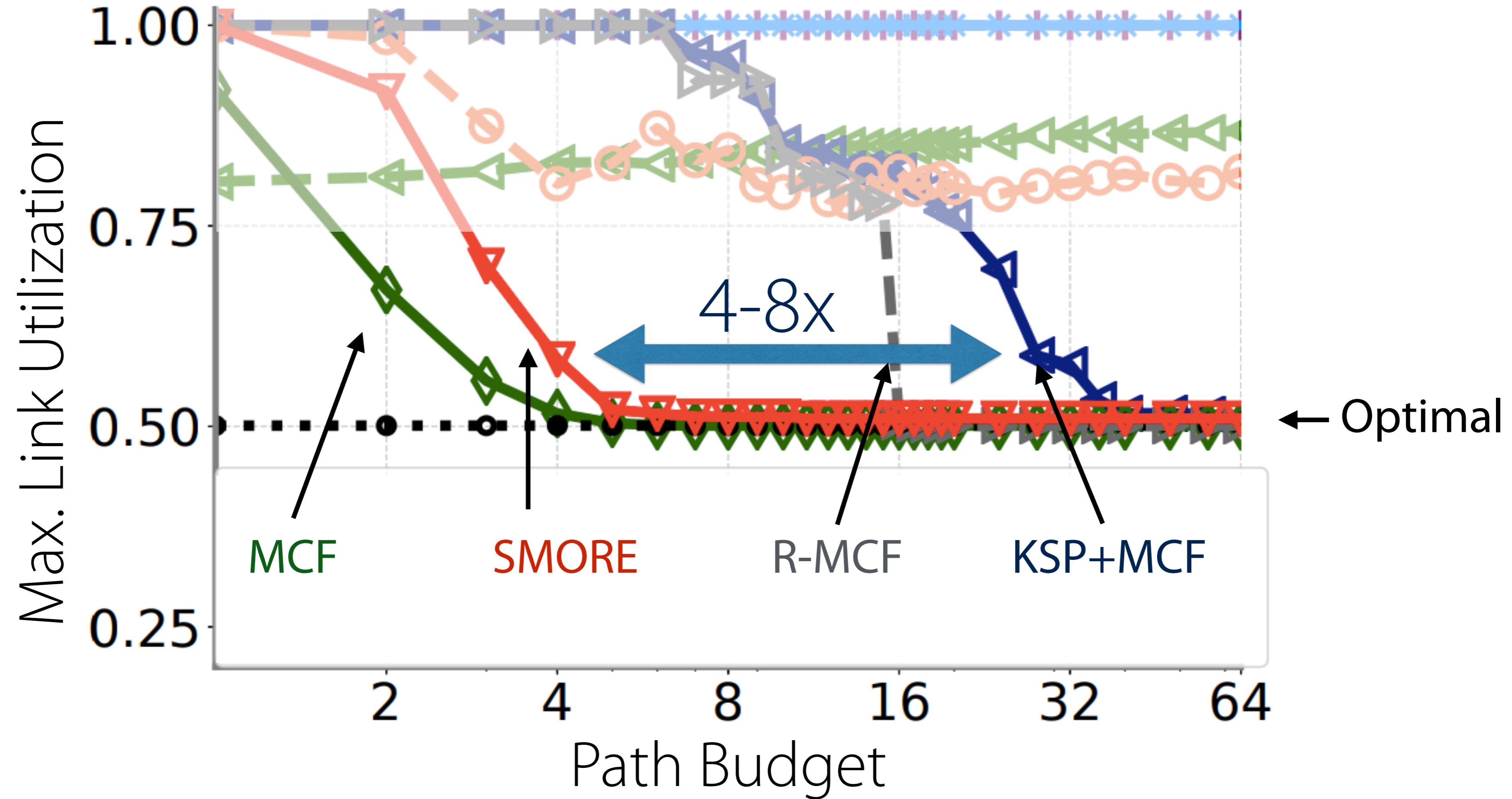
Robustness



Robustness



Operational Constraints - Path Budget



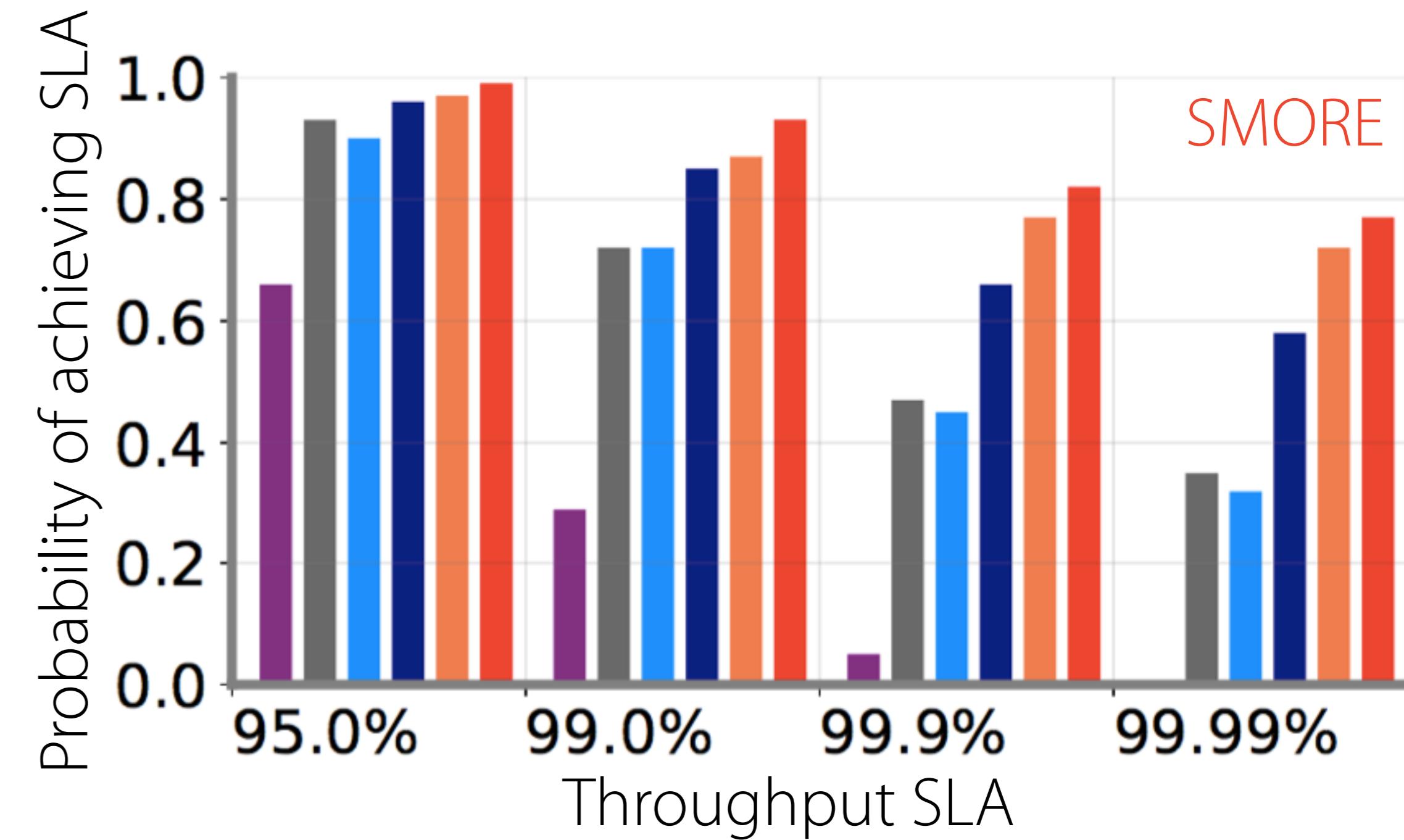
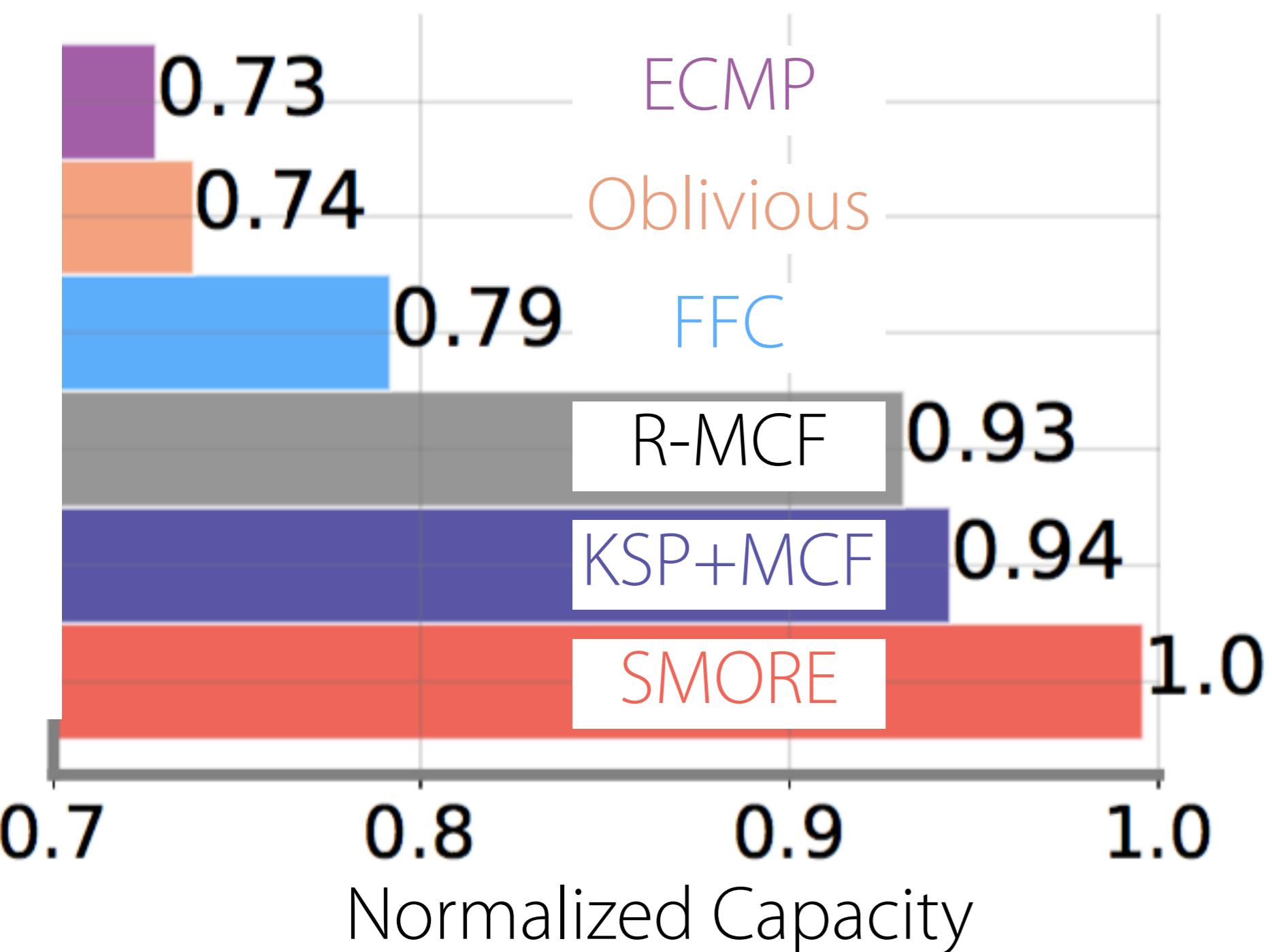
Large Scale Simulations



- Conducted larger set of simulations on Internet Topology Zoo
- 30 topologies from ISPs and content providers
- Multiple traffic matrices (gravity model), failure models and operational conditions

Do these results generalize?

Yes*



Takeaways

- **Path selection** plays an outsized role in the performance of TE systems
- **Semi-oblivious TE** meets the competing objectives of performance and robustness in modern networks
 - **Oblivious routing** for path selection + **Dynamic load-balancing**
- Ongoing and future-work:
 - Apply to other networks (e.g. non-Clos DC topologies)
 - SR-based implementations and deployments

Thank You!

SMORE: Oblivious routing + Dynamic rate adaptation



Yang Yuan
Cornell



Chris Yu
CMU



Nate Foster
Cornell



Bobby Kleinberg
Cornell



Petr Lapukhov
Facebook



Chiun Lin Lim
Facebook



Robert Soule
Lugano

Semi-Oblivious Traffic Engineering: The Road Not Taken

Praveen Kumar Cornell Yang Yuan CMU Chris Yu Cornell Nate Foster Cornell Robert Kleinberg Cornell
Petr Lapukhov Facebook Chiun Lin Lim Facebook Robert Soule Università della Svizzera italiana

Abstract
Networks are expected to provide reliable performance under a wide range of operating conditions, but existing traffic engineering (TE) solutions optimize for performance or robustness, but not both. A key factor that impacts the quality of a TE system is the set of paths used to carry traffic. Some systems rely on shortest paths, which leads to excessive congestion in topologies with bottleneck links, while others use paths that minimize congestion, which are brittle and prone to failure. This paper presents a system that uses a set of paths composed using RACK's oblivious routing algorithm, as well as a contraction controller to dynamically adapt sending rates. Although oblivious routing and contracted TE have been studied previously in isolation, their combination is novel and powerful. We build a software framework to model TE solutions and compare them experimentally across a large variety of topologies and scenarios including a producer-consumer pair, a content provider and an ISP. Our results show that semi-oblivious routing provides better performance and is more robust than state-of-the-art.

1 Introduction
*Two roads diverged in a wood, and I—
I took the one less traveled by,
And that has made all the difference.* —Robert Frost

Networks are expected to provide good performance even in the presence of unexpected traffic shifts and outright failures. But while there is extensive literature on how to best route traffic through a network while optimizing for objectives such as minimizing congestion [3, 9, 13, 14, 15, 22, 24, 26, 47], current traffic engineering (TE) solutions can perform poorly under operating conditions far from the expected [32, 42]. The tension between performance and reliability is not merely a hypothetical concern. Leading technology companies such as Google [18, 24] and Microsoft [22, 32] have

identified these properties as critical issues for their private networks. For example, a central goal of Google's B4 system is to drive link utilization to 100%, but doing this means that packet loss is "inevitable" when failures occur [34]. Meanwhile a different study of availability at Google identified "no more than a few minutes of downtime per month" as a goal, where downtime is defined as packet loss above 0.1%–2% [18].

Stepping back, one can see that there are two fundamental choices in the design of any TE system: (i) which forwarding paths to use to carry traffic from sources to destination, and (ii) which sending rates to use to balance incoming traffic flows among those paths. Any TE solution can be viewed in terms of these choices, but there are also practical considerations that limit the kinds of systems that can be deployed. For example, setting TE solutions and contract TE rates requires a large amount of topology and scenario information, producing a complex set of constraints for a controller and an ISP. Our results show that semi-oblivious routing provides better performance and is more robust than state-of-the-art.

Code: github.com/cornell-netlab/yates

NSDI '18

Learn more: www.cs.cornell.edu/~praveenk/smore/