Traffic engineering

CSE 561, Winter 2021

Ratul Mahajan

What we read

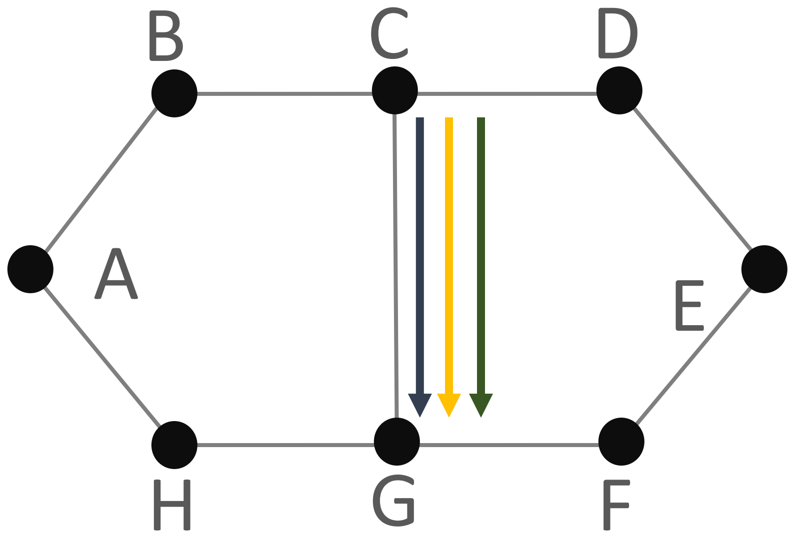
Techniques to get more out of your network

- SWAN: Centrally controlling your backbone traffic
- Edge Fabric: Centrally controlling egress traffic
- VLB: Optimal load balancing
- Network coding: Optimal throughput

Journey for optimizing network use (practice)

- SPF with load-based cost
- SPF with static cost
- CSPF (used in MPLS)
- Centralized control

Limitations of static-cost SPF



CSPF

Each ingress router measure traffic that it is sending to other routers

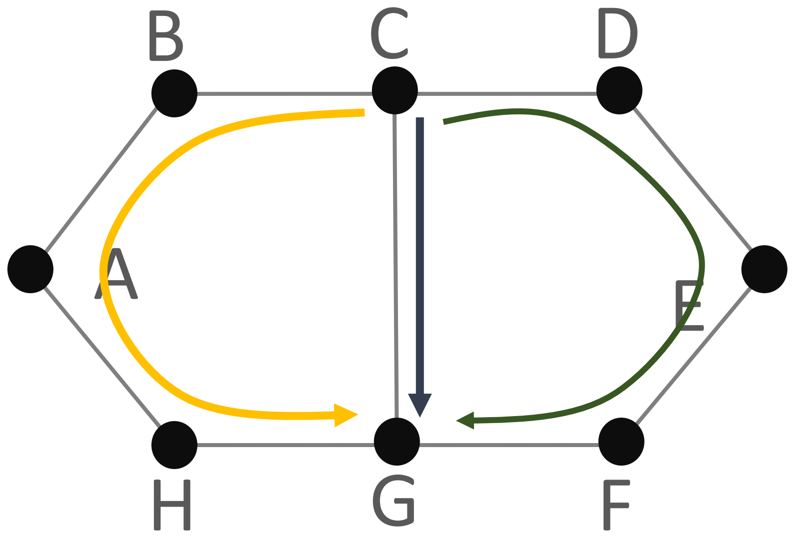
Ingress router finds paths that can accommodate its traffic

- Shortest path that meets the capacity constraint (CSPF)

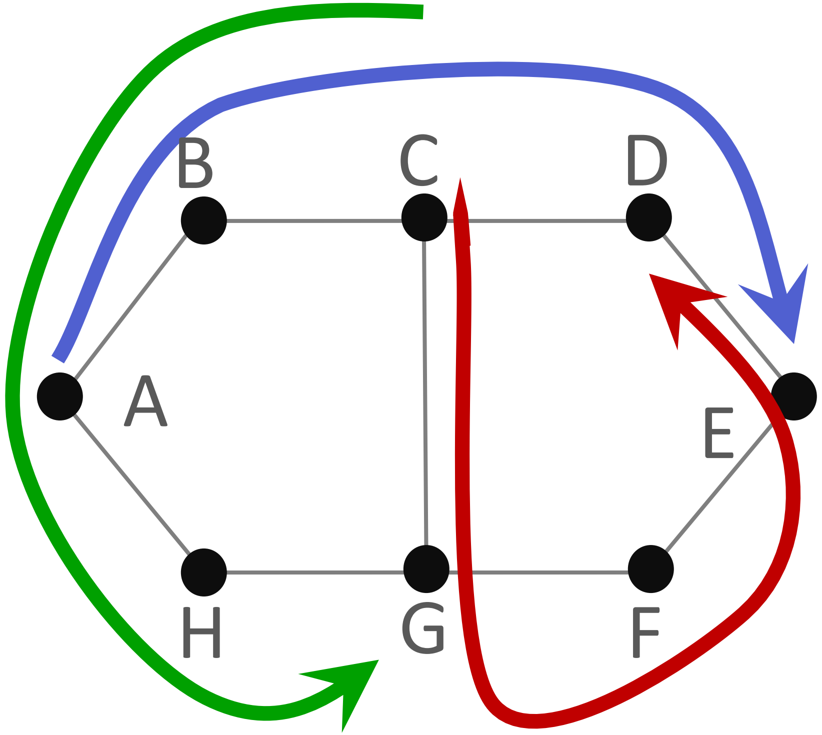
Ingress router asks other routers if they can use the path

- Necessary because all ingress routers are operating independently

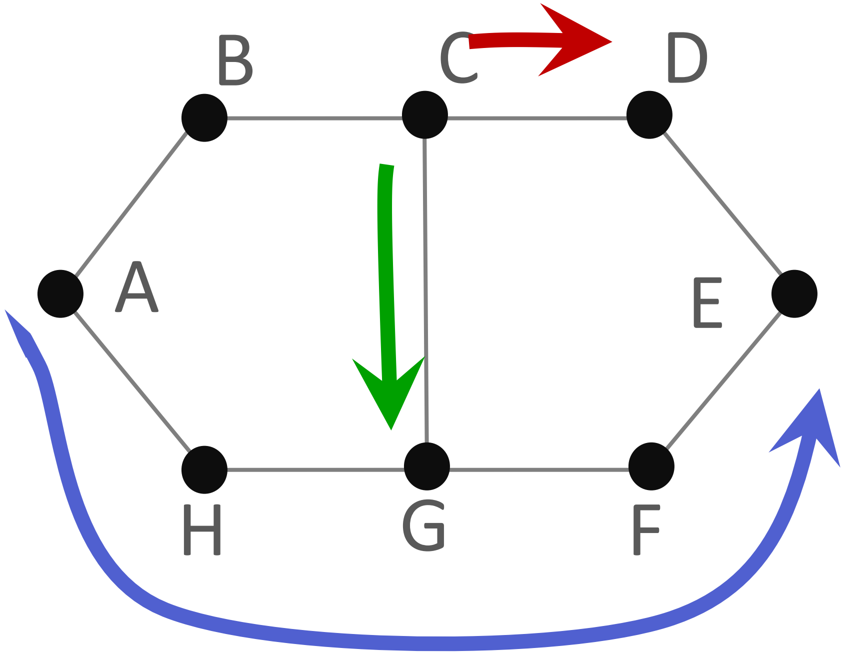
Same example with CSPF



But CSPF has issues too



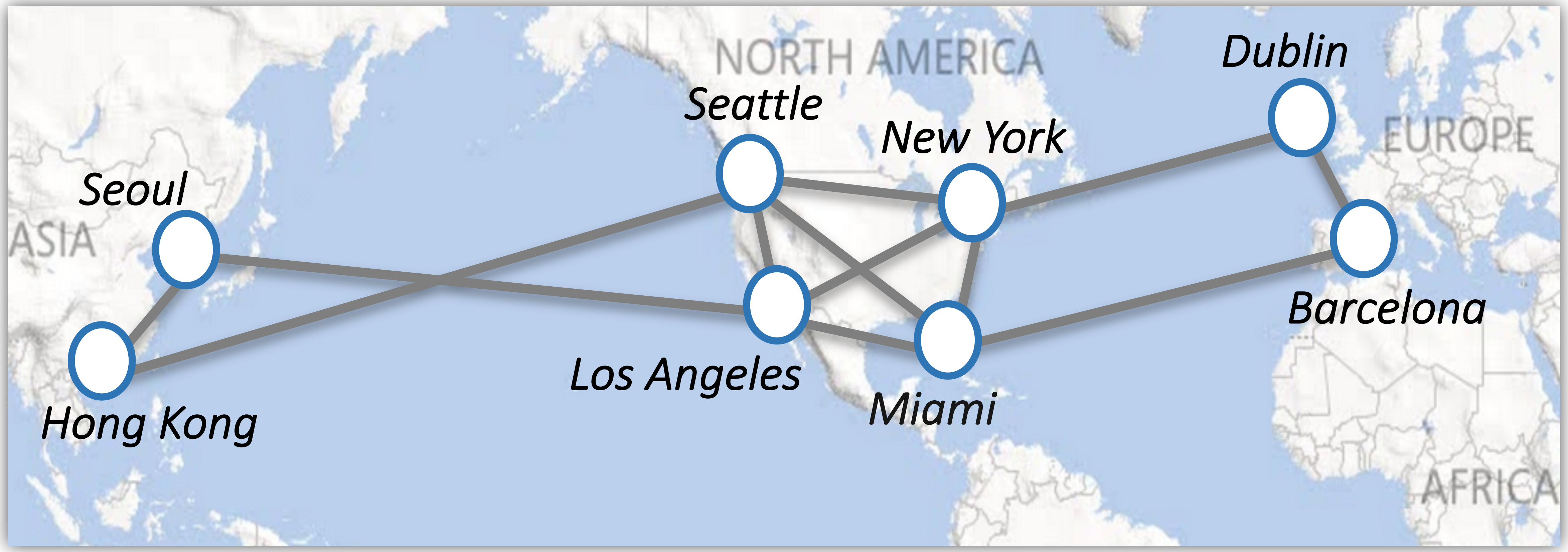
Local, greedy allocation
(Distributed CSPF)



Globally optimal allocation
(Centralized)

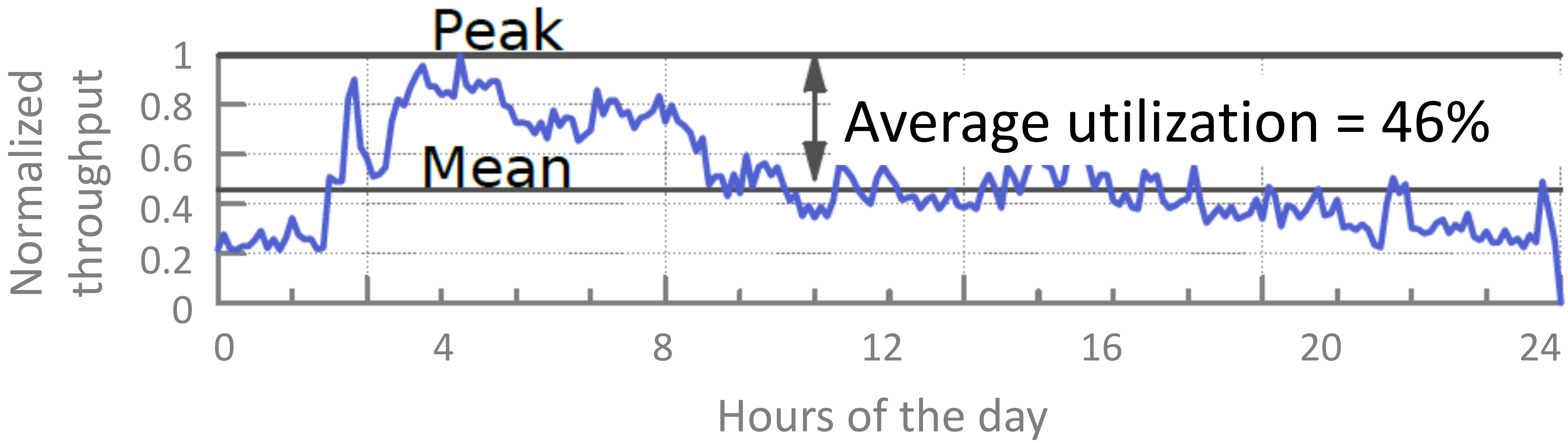
SWAN

Inter-DC WAN: A critical, expensive resource



But it is highly inefficient and inflexible!

Inefficiency of the inter-DC WAN



Normalized traffic on a busy link between data centers

Root cause: Service-level allocations

Operators configure individual services with maximum sending rate

	S1	S2	S3	…..
SEA → NYC (80)	10	15	5	….
SEA → CHI (100)	20	20	10	….
……				

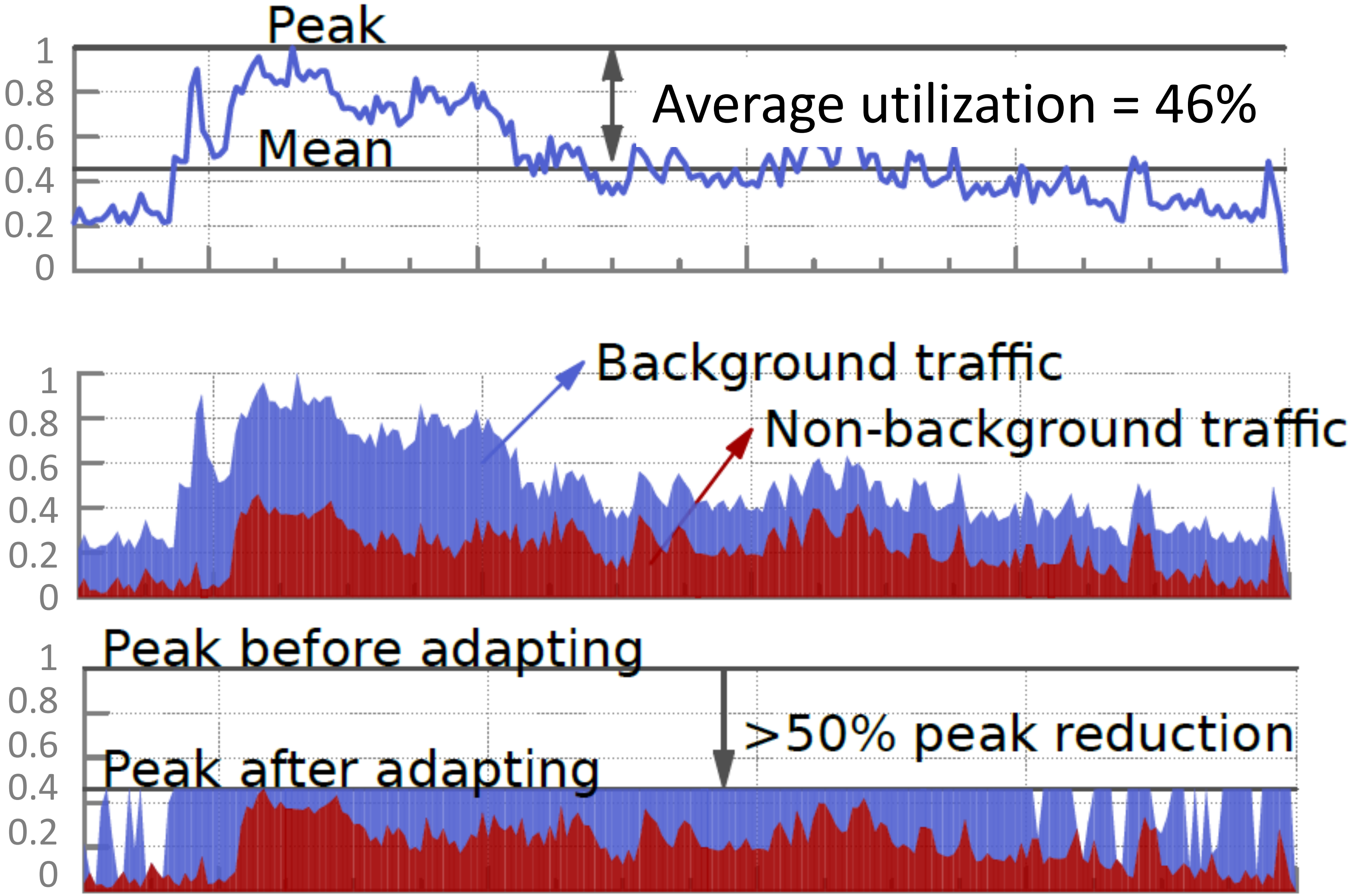
Inefficient: The combined maximum is uncommon

Unreliable: Load can exceed capacity when failures occur

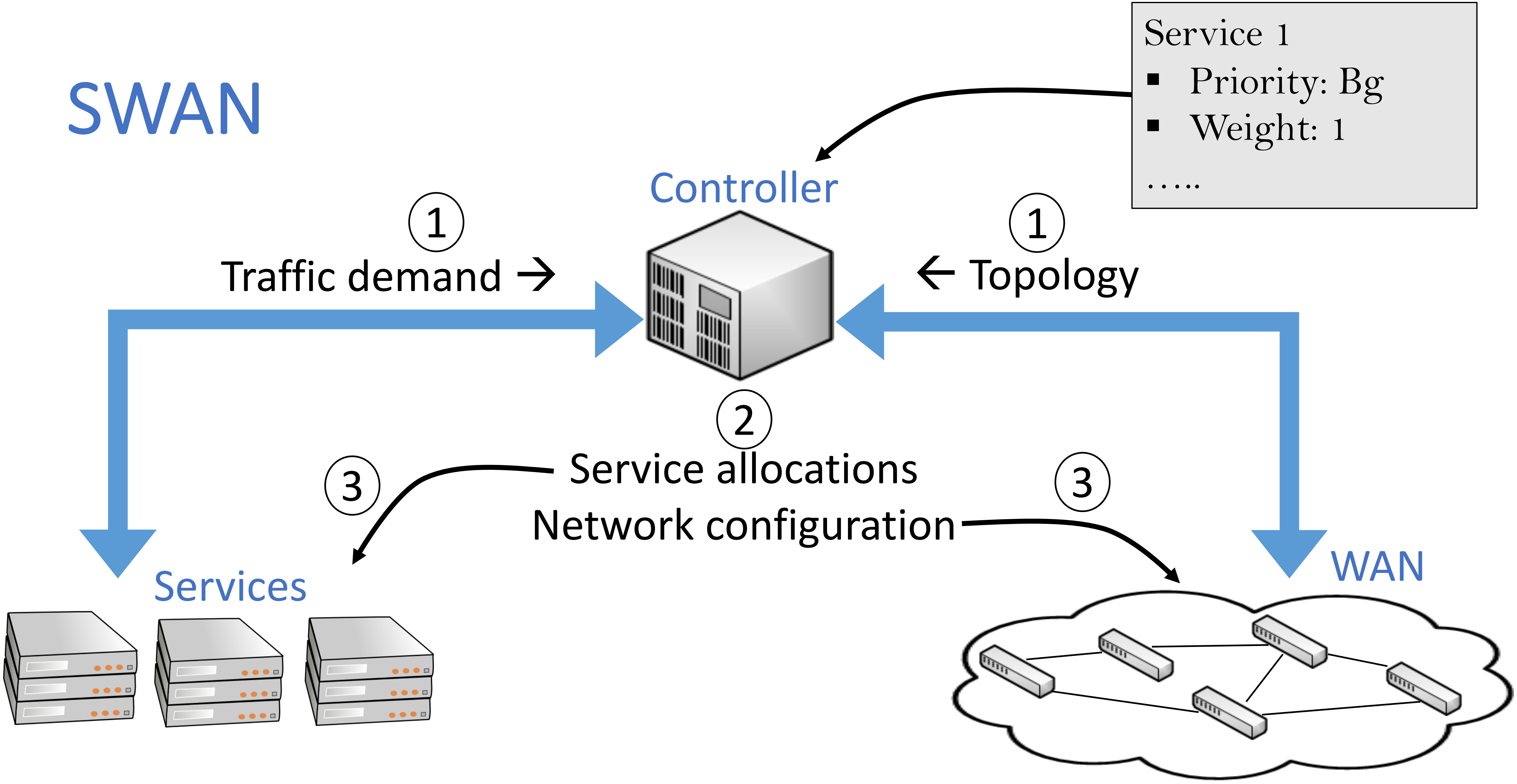
Slow to change: Must change all allocations to add services or network links

Centralized control can increase efficiency

```
Service 1
▪ Priority: Bg
▪ Weight: 1

Service 2
▪ Priority: Bg
▪ Weight: 2

Service 3
▪ Priority: Non-bg
▪ Weight: 1

……
```



SWAN



Service 1
▪ Priority: Bg
▪ Weight: 1
…..
Controller
1
Traffic demand →
1
← Topology
2
Service allocations
Network configuration
3
3
Services
WAN

Centralized control presents new challenges

- Computational scalability → Approximation algorithm with provable bounds
- Congestion due to updates → Congestion-free updates
- Limited switch memory → Maintain the “working set” in memory

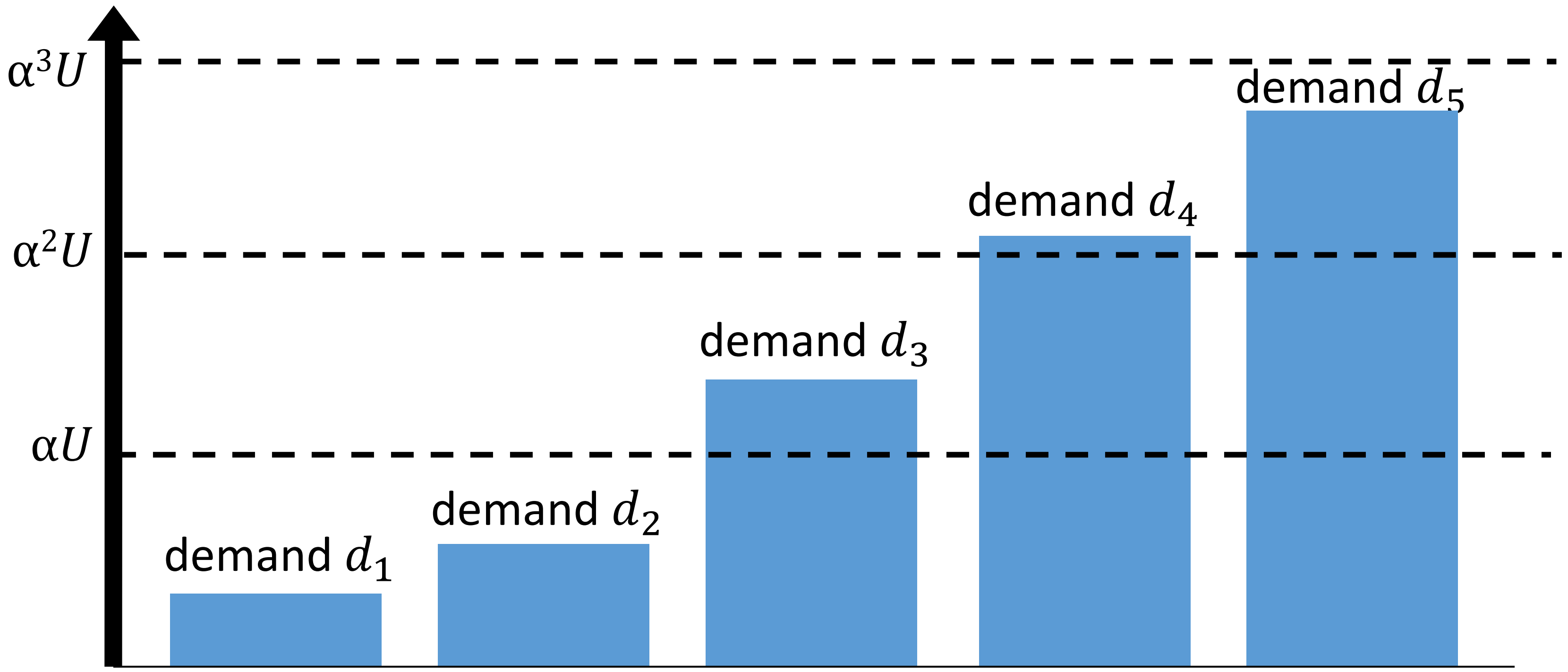
Scalably computing allocation

Goal: Prefer higher-priority traffic and max-min fair within a class

Challenge: Network-wide fairness requires many MCFs

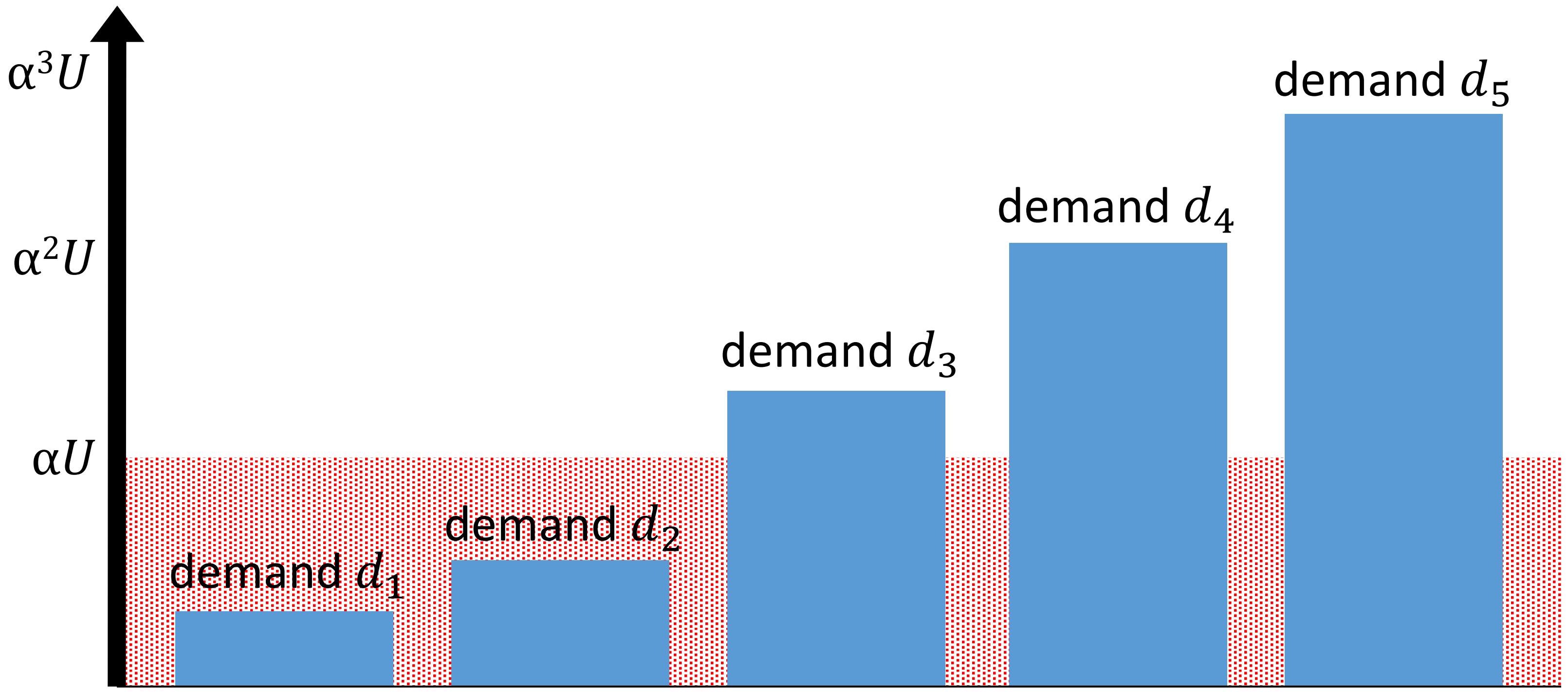
Approach: Bounded max-min fairness (fixed number of MCFs)

Bounded max-min fairness



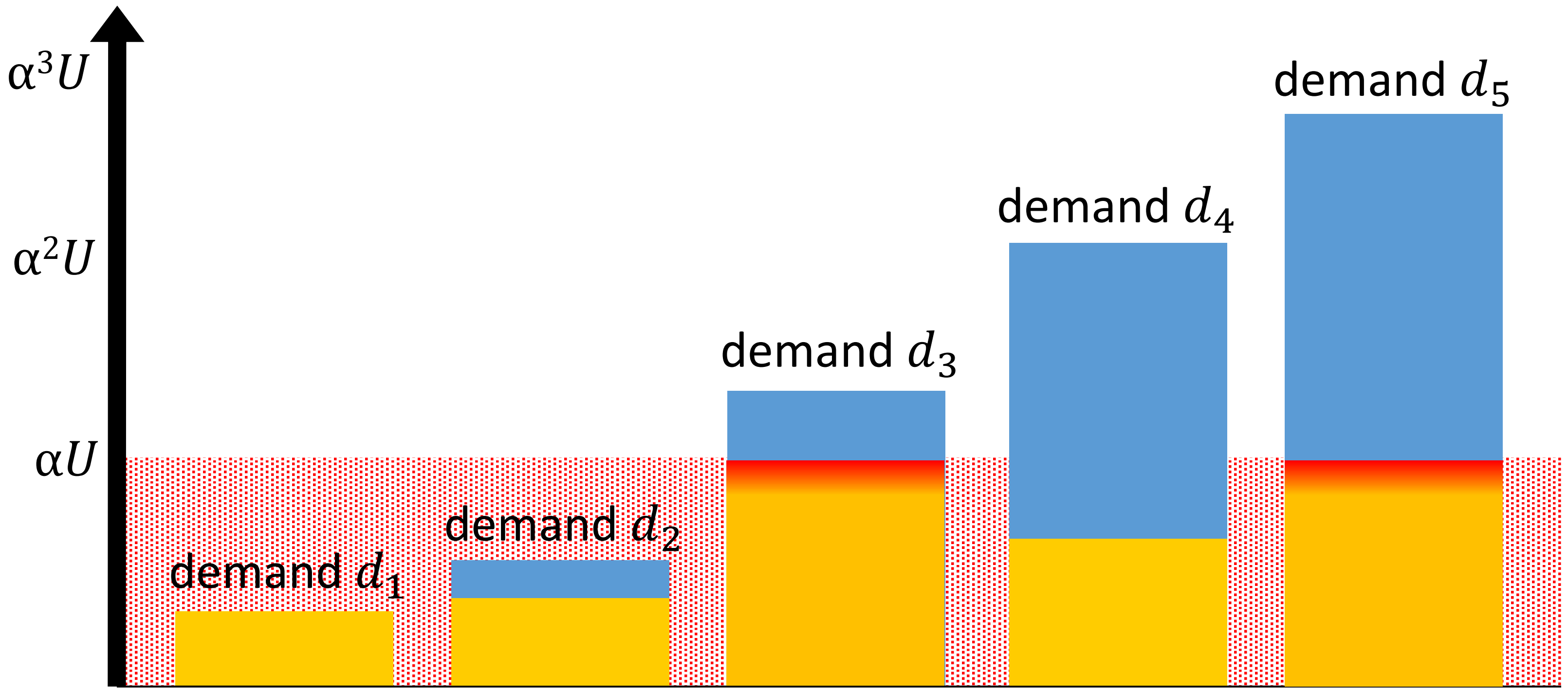
Geometrically partition the demand space with parameters α and U

Bounded max-min fairness



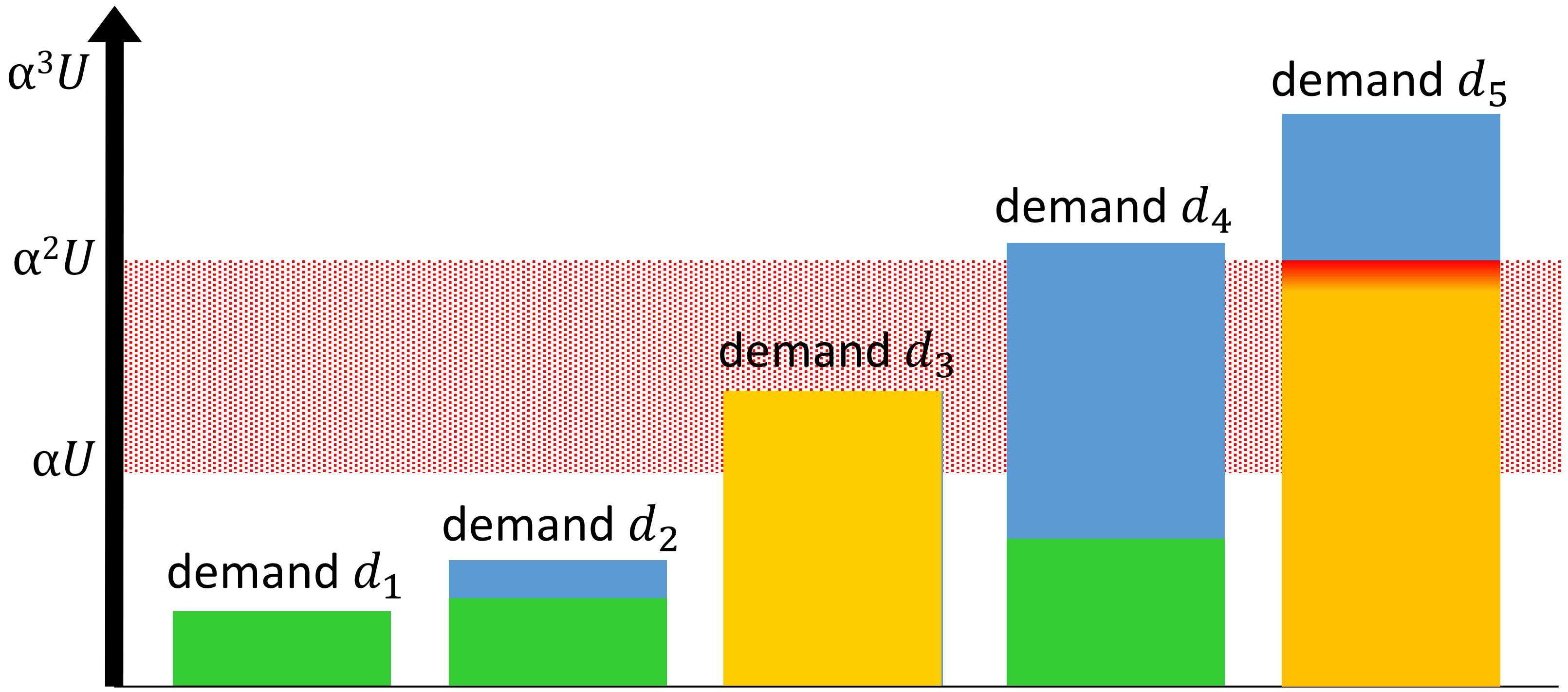
Maximize throughput while limiting all allocations below αU

Bounded max-min fairness



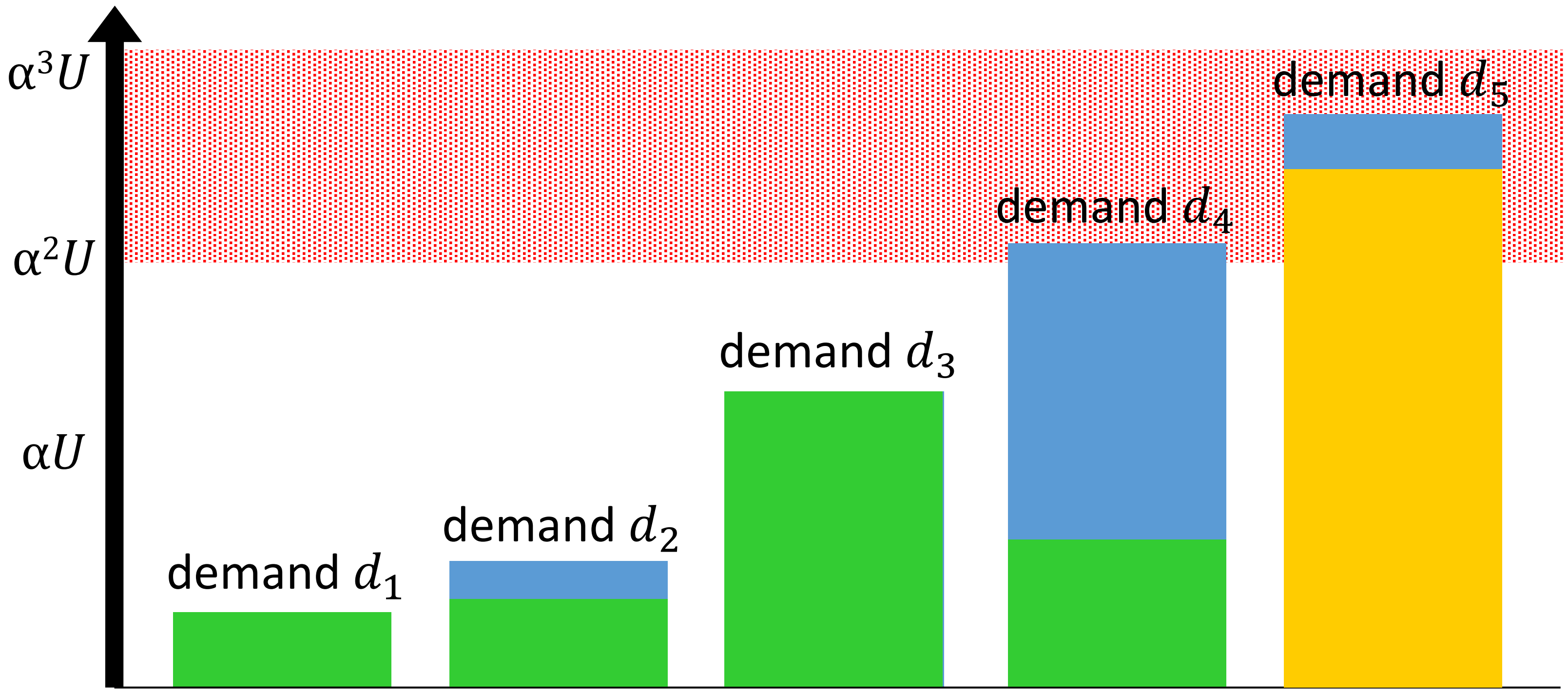
Maximize throughput while limiting all allocations below αU

Bounded max-min fairness



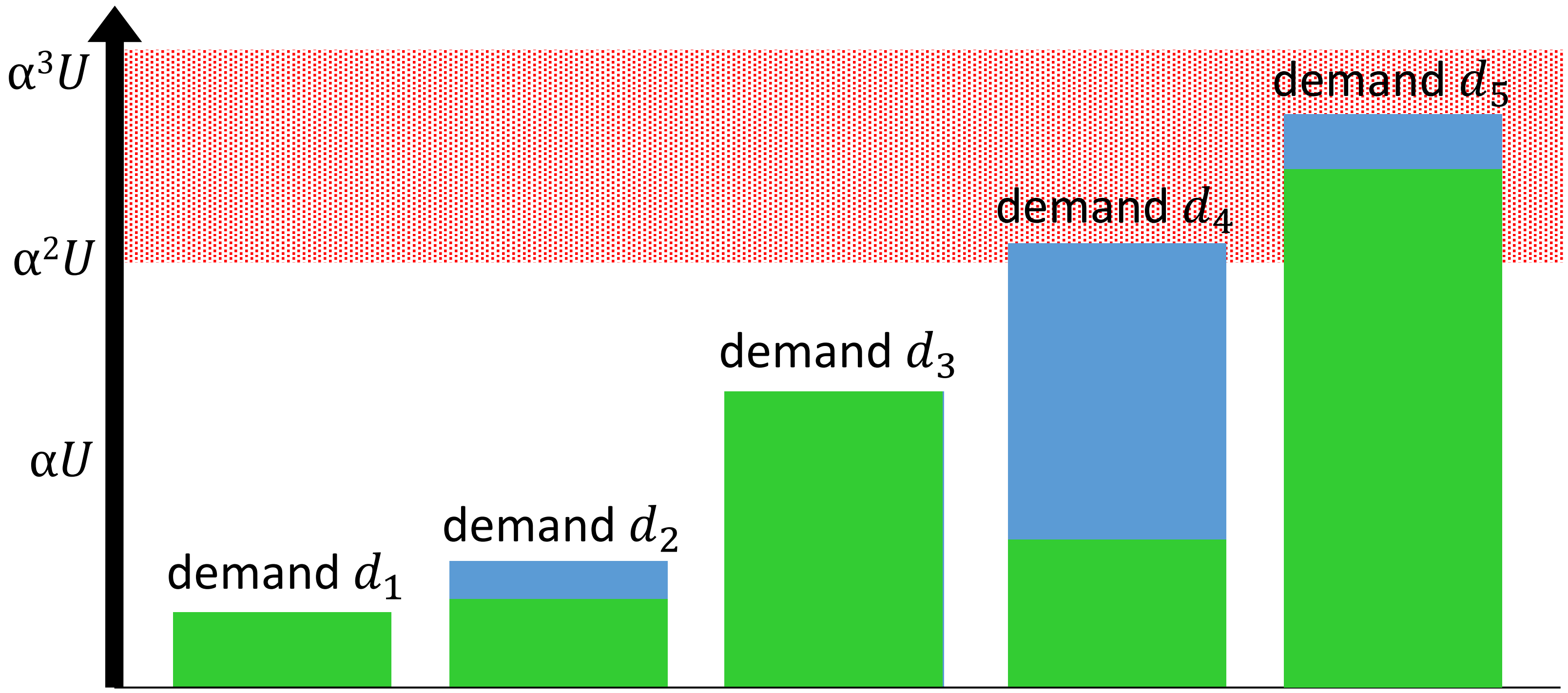
Fix the allocation for smaller flows

Bounded max-min fairness



Continue until all flows fixed

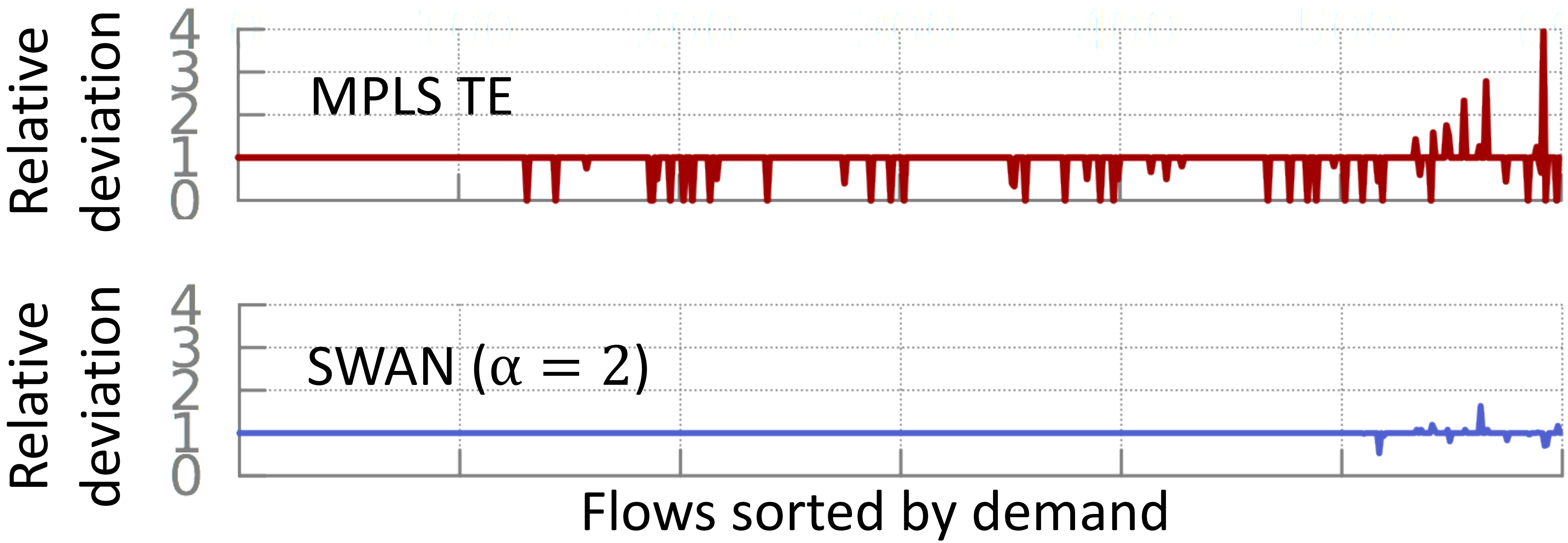
Bounded max-min fairness



Fairness bound: Each flow is within $\left[\frac{1}{\alpha}, \alpha\right]$ of its fair rate

Number of MCFs: $\log_\alpha \frac{\max(di)}{U}$

SWAN computes fair allocations



In practice, only 4% of the flows deviate more than 5%

Centralized control presents new challenges

Computational scalability → Approximation algorithm with provable bounds

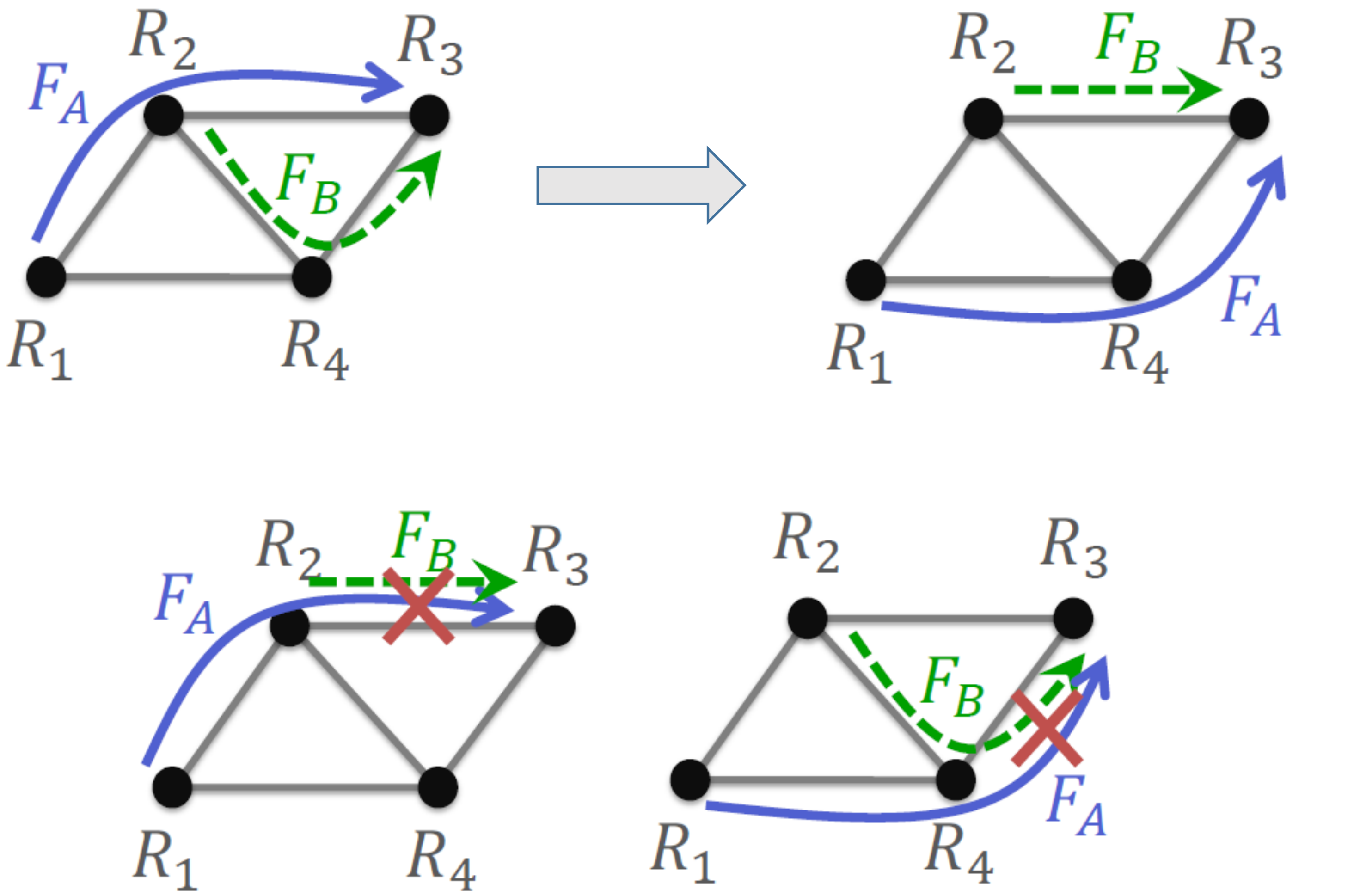
Congestion due to updates → Congestion-free updates

Limited switch memory → Maintain the “working set” in memory

Challenge: Congestion during network updates

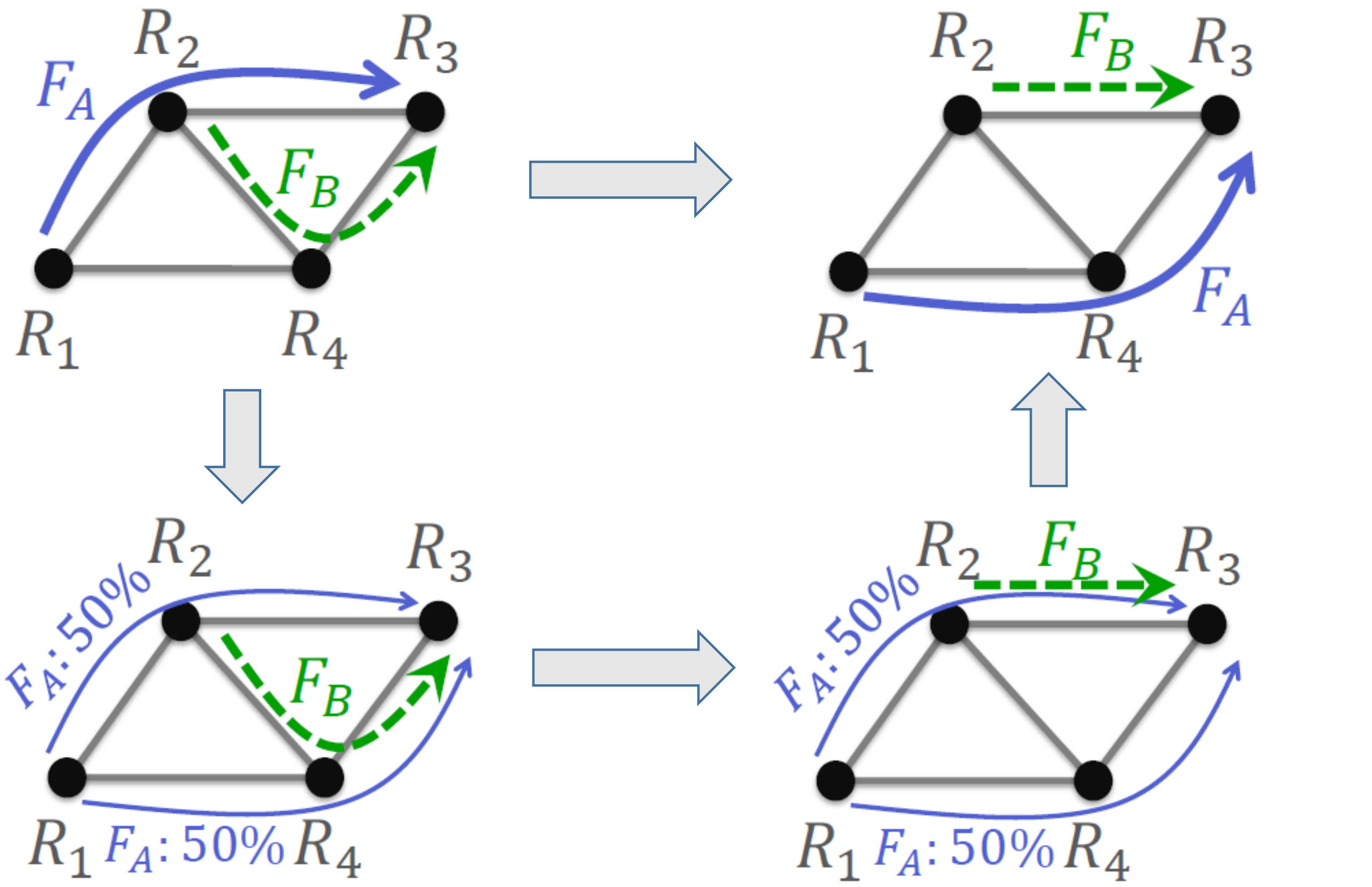
Link capacity: 10

Flow size: 6.6



Solution: Congestion-free update plans

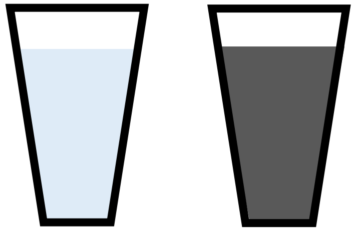
Link capacity: 10
Flow size: 6.6



Computing congestion-free update plans

Leave scratch capacity s on each link

- Guarantees a plan with at most $\left\lceil \frac{1}{s} \right\rceil - 1$ steps



Find a plan with minimum number of steps using an LP

- Search for a feasible plan with 1, 2, ... max steps

Use scratch capacity for background traffic

- Bound its experienced congestion

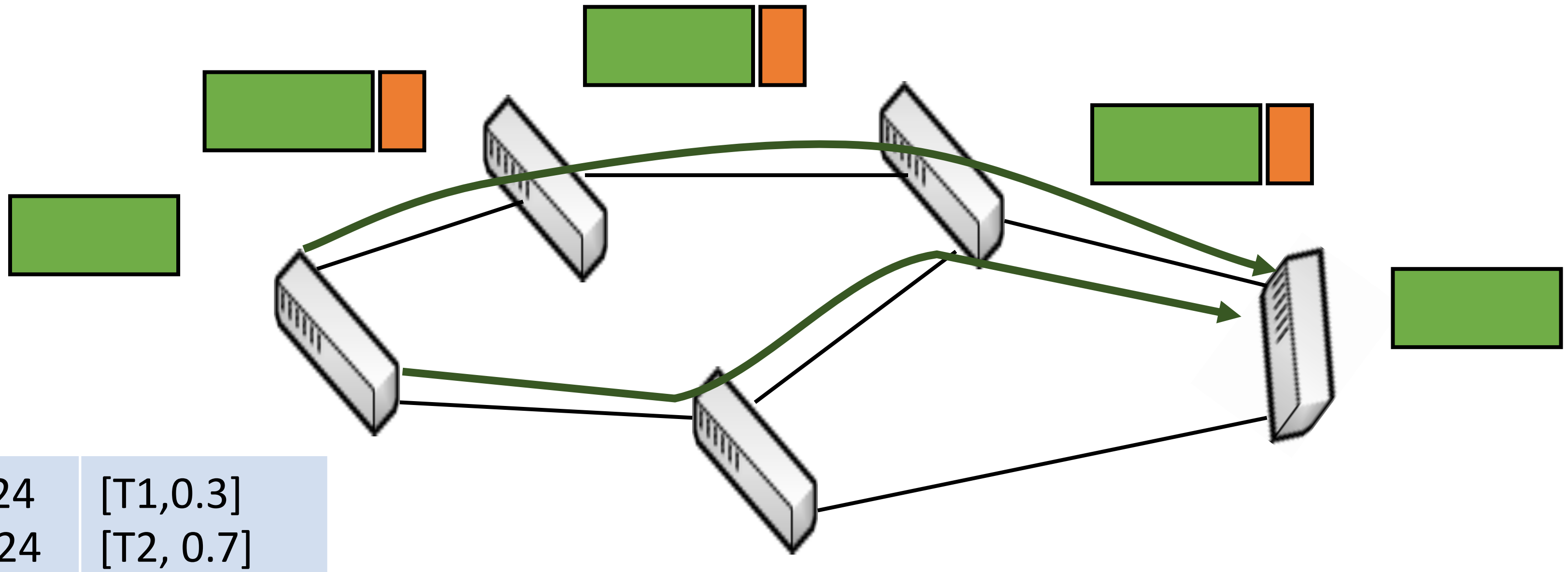
Centralized control presents new challenges

Computational scalability → Approximation algorithm with provable bounds

Congestion due to updates → Congestion-free updates

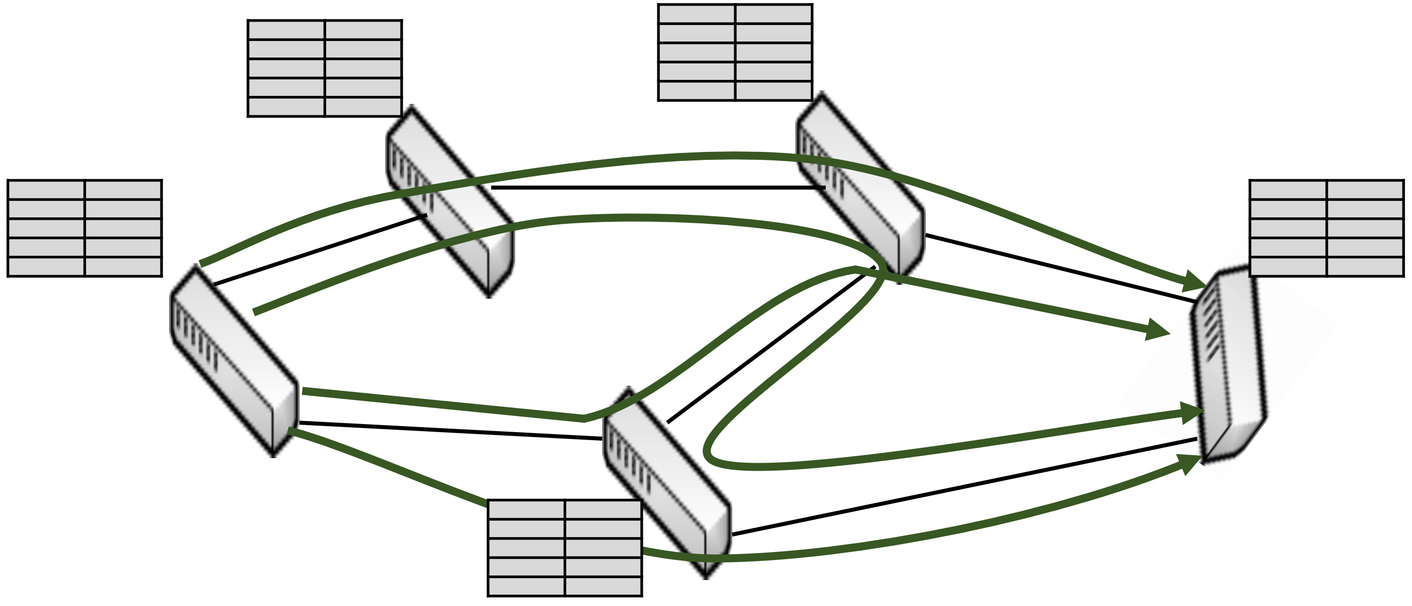
Limited switch memory → Maintain the “working set” in memory

How tunnels work

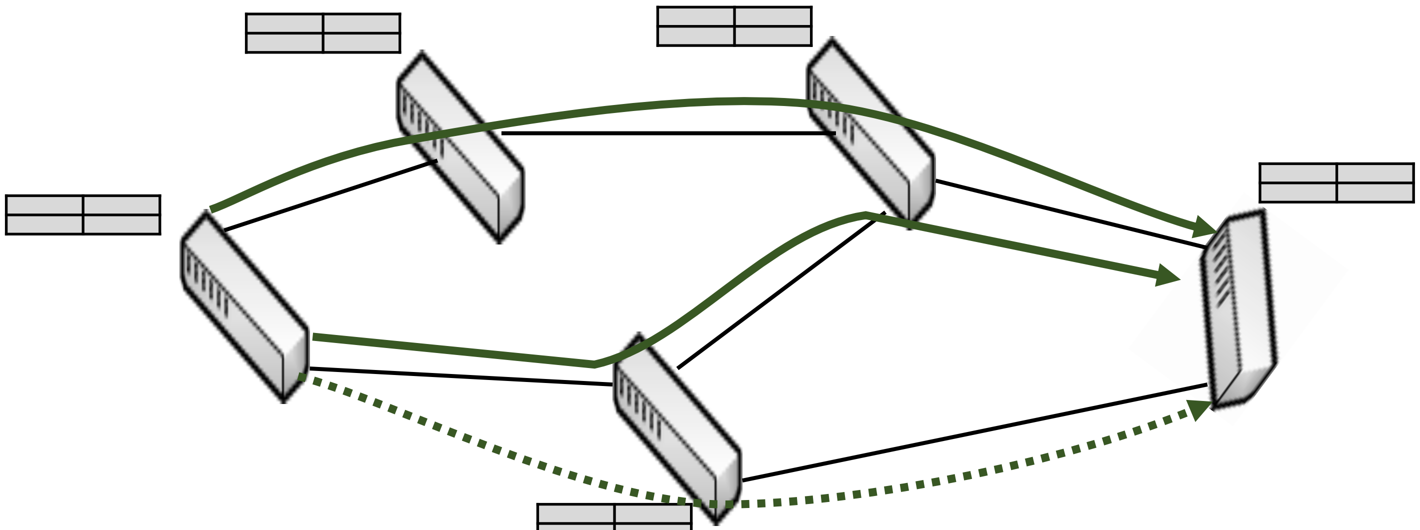


Src: 1.2.3.0/24 **Dst**: 5.6.7.0/24 **DSCP**: 3	[T1,0.3] [T2, 0.7]

Working with limited switch memory



Working with limited switch memory

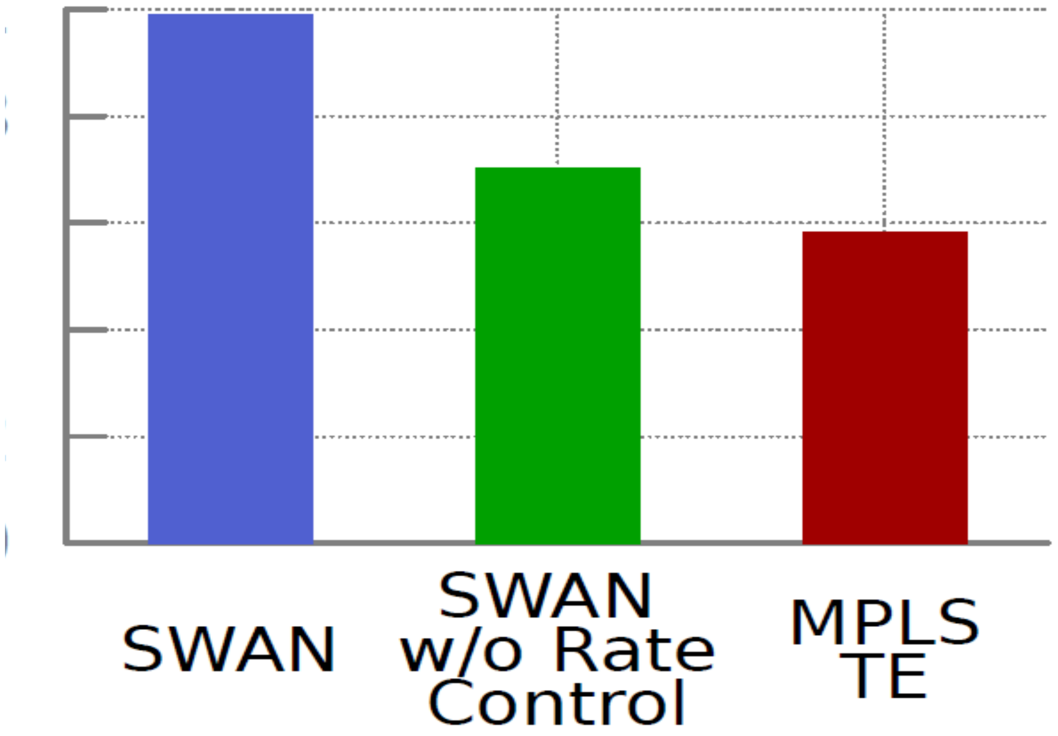


Install only the “working set” of paths

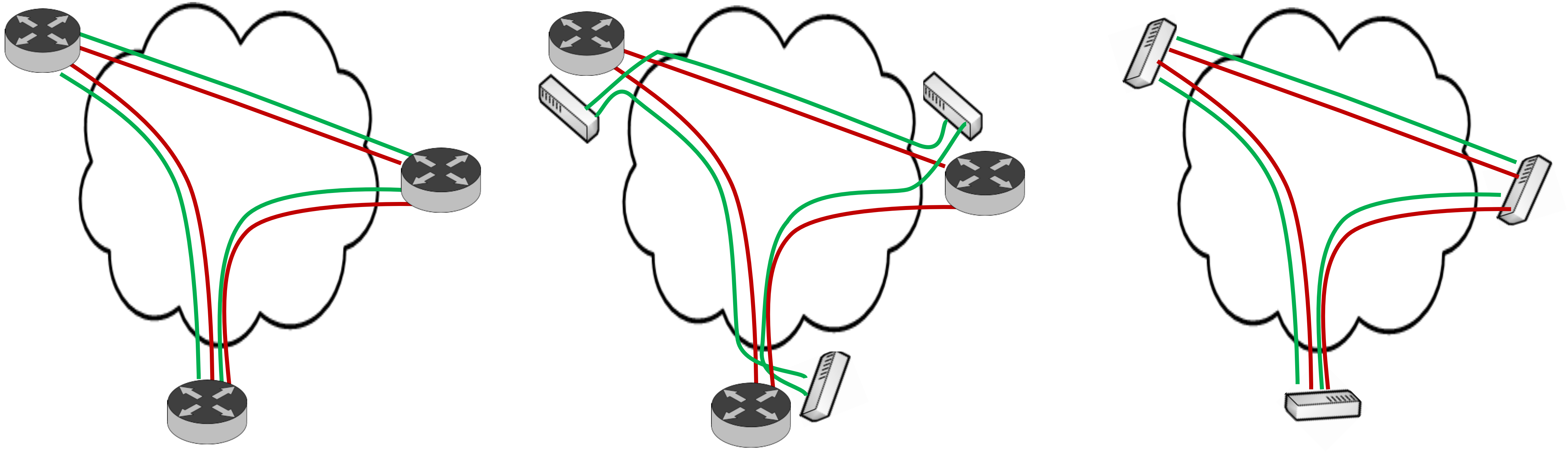
Use scratch capacity to enable disruption-free updates to the set

Efficiency improvement with SWAN

Throughput
(relative to optimal)



Deploying SWAN



Before

Partial deployment

Full deployment

Centralized vs distributed control

Centralized

- More efficient, flexible allocation policies

Distributed

- More fault tolerant
- Easy to get started?

Both can have poor transients

- But centralized offers a way to carefully manage them

VLB

SWAN is doing a lot of work to be efficient

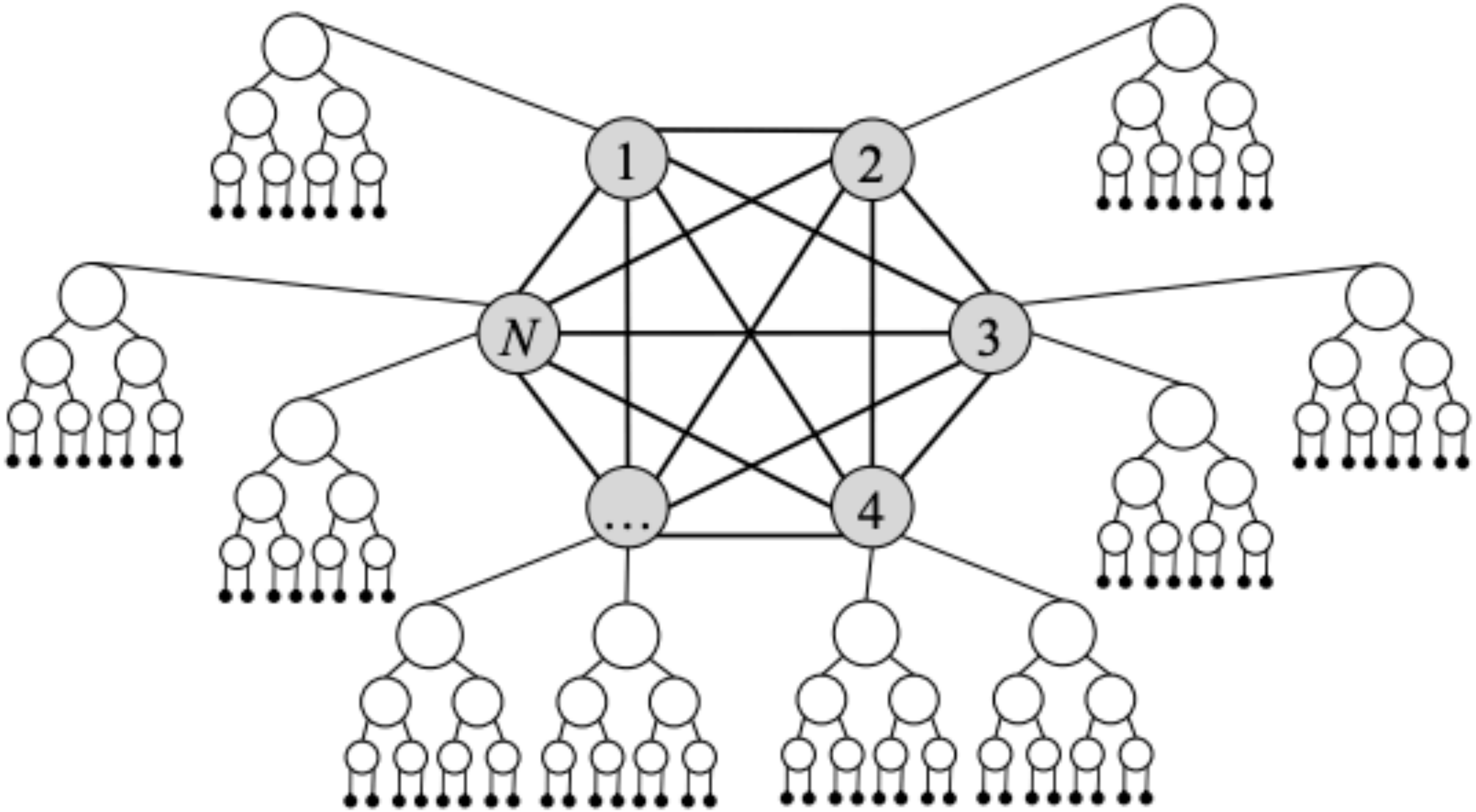
Estimate traffic matrix

Carefully plan traffic paths

Rate limit services

What if all this were not possible?

Borrowing from VLB



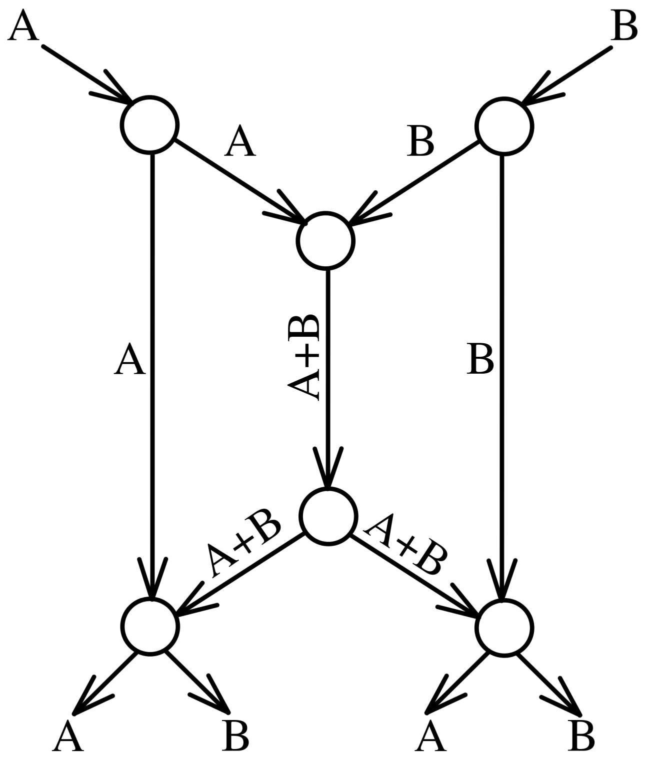
Good idea? Bad idea?

The good	The bad
Simple operation	High latency
Can work with any valid matrix	Sub-optimal for any matrix

Where does it make sense?

Network coding

Routing vs coding



Where does coding make sense?

Next lecture

Today: How can routing help with resource allocation in the network

Next: How can forwarding-time logic help?