SWAN: Software-driven wide area network

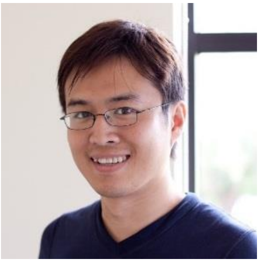
Ratul Mahajan

Microsoft Research

Partners in crime

Vijay Gill

Chi-Yao Hong



Srikanth Kandula

Ratul Mahajan

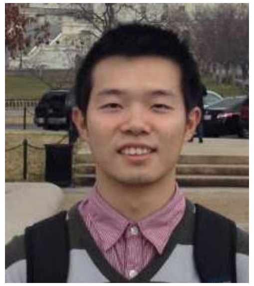
Mohan Nanduri

Ming Zhang

Roger Wattenhofer

Rohan Gandhi

Xin Jin



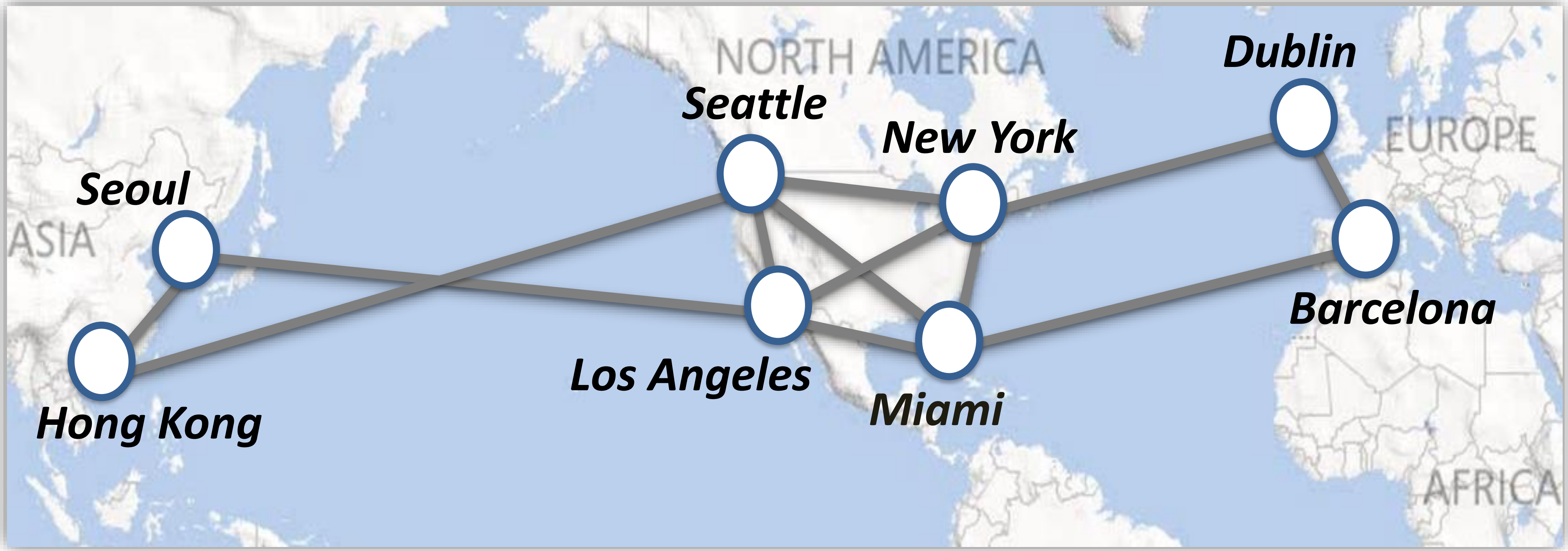
Harry Liu

Dave Maltz

Peng Sun

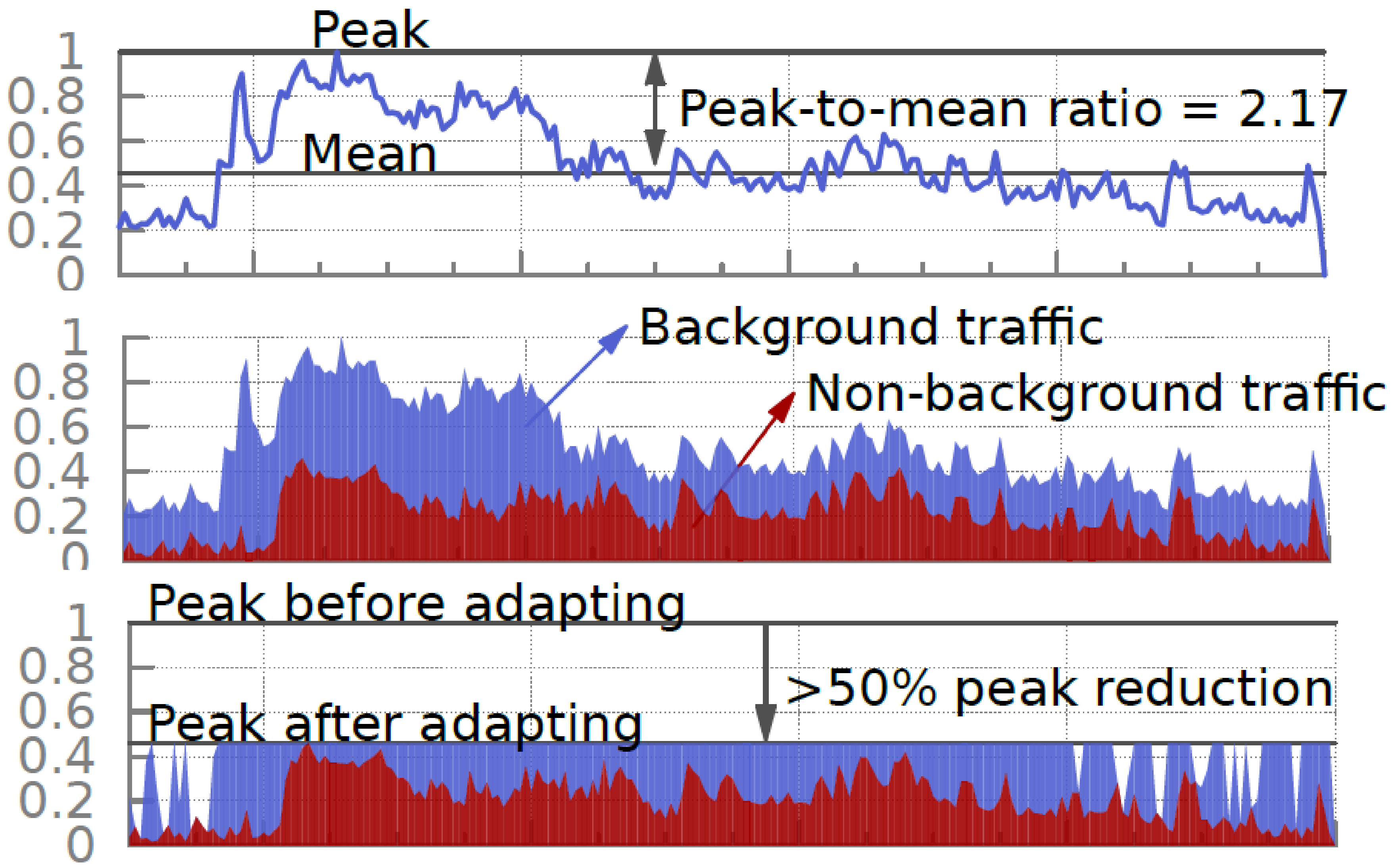
Lihua Yuan

Inter-DC WAN: A critical, expensive resource

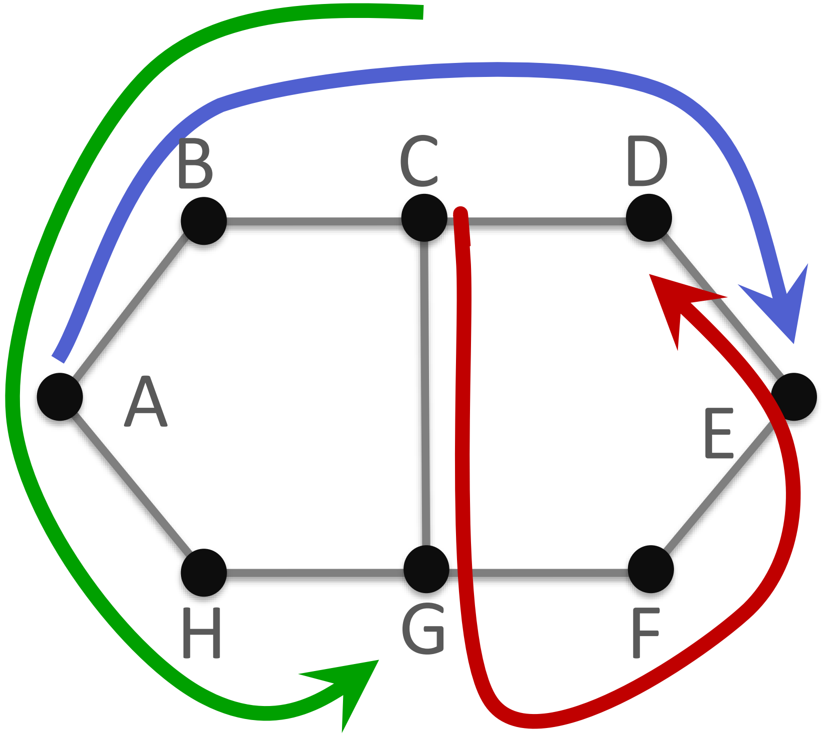


But it is highly inefficient

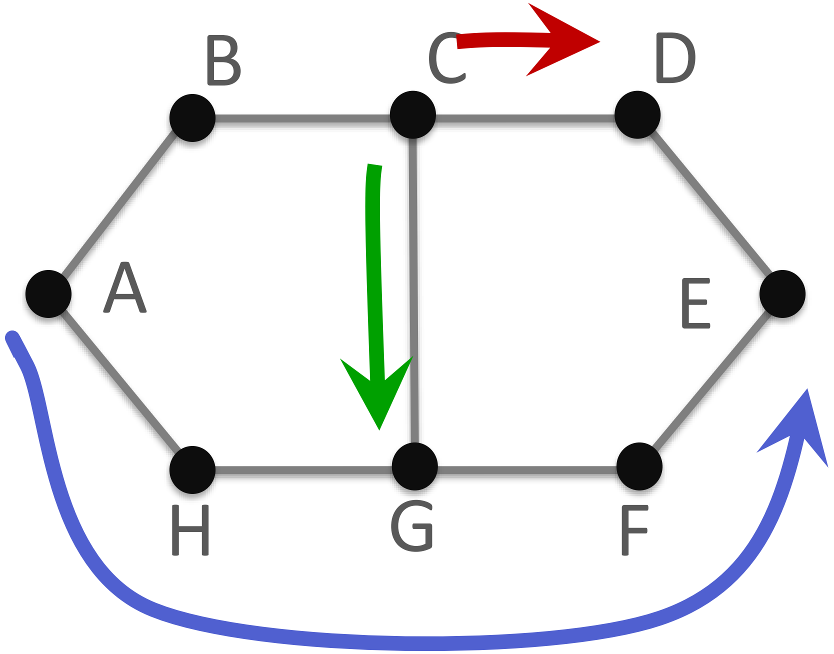
One cause of inefficiency: Lack of coordination



Another cause of inefficiency: Local, greedy resource allocation



Local, greedy allocation



Globally optimal allocation

SWAN: Software-driven WAN

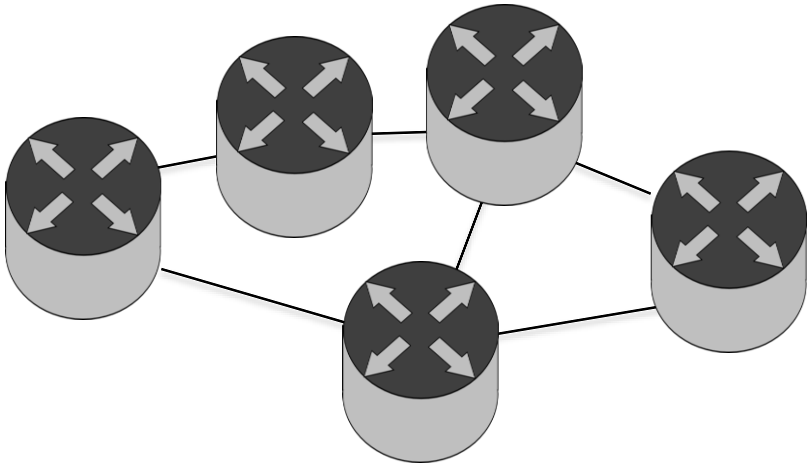
Goals:

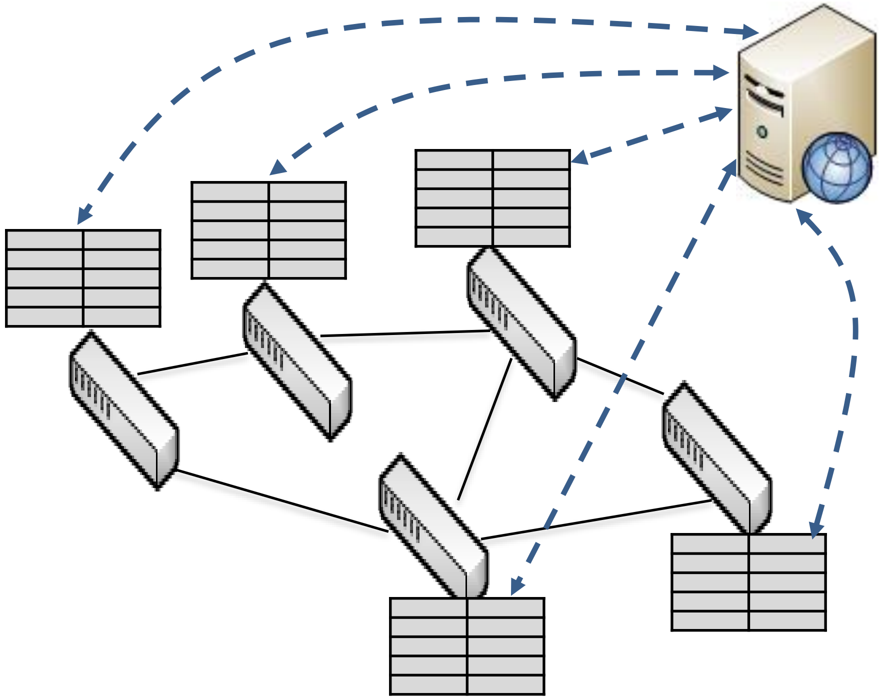
- Highly efficient WAN
- Support flexible sharing policies
  - Strict priority classes
  - Max-min fairness within a class

Key design elements:

- Coordinate the sending rate of services
- Centralized resource allocation

SDN primer





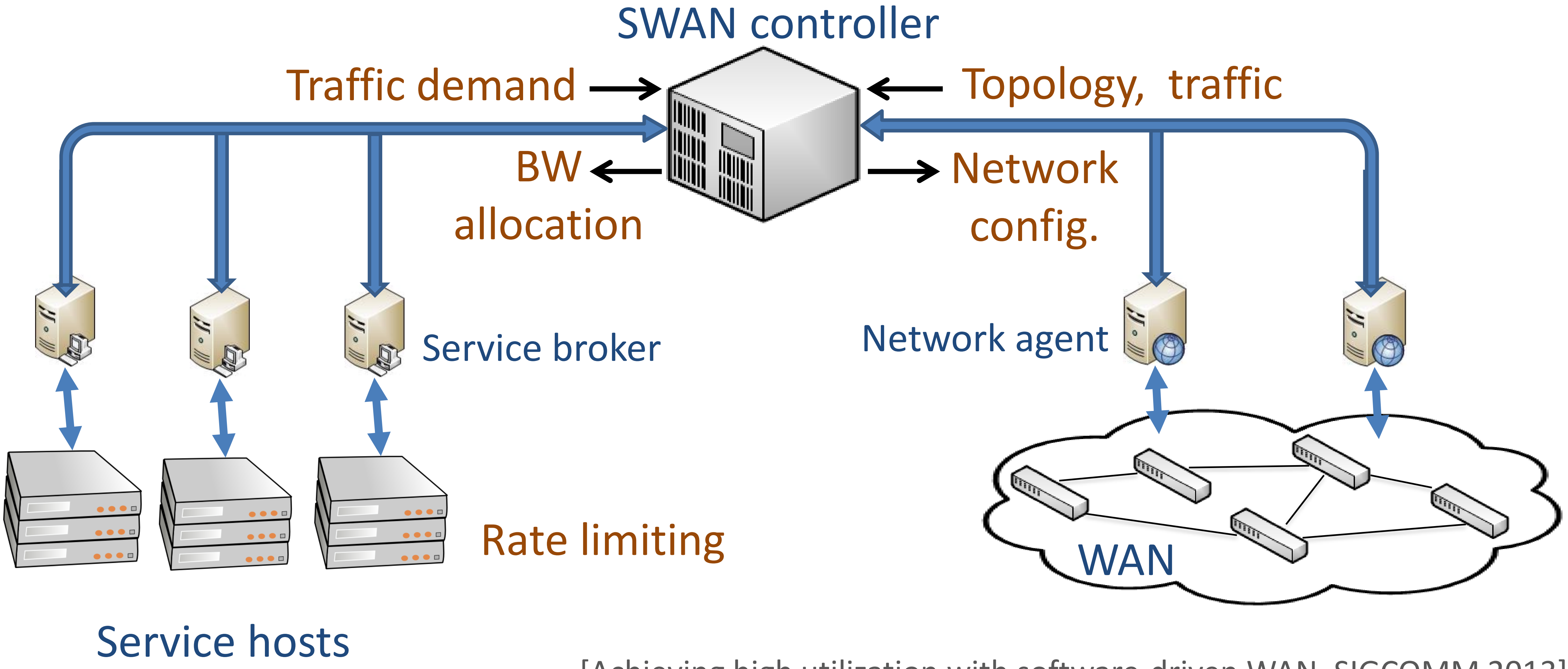
Networks today

- Beefy routers
- Control plane: distributed, on-board
- Data plane: indirect configuration

SDNs

- Streamlined switches
- Control plane: centralized, off-board
- Data plane: direct configuration

SWAN overview



[Achieving high utilization with software-driven WAN, SIGCOMM 2013]

Key design challenges

Scalably computing BW allocations and network config

Avoiding congestion during network updates

Working with limited switch memory

Scalably computing allocation

Path-constrained, multi-commodity flow problem

- Allocate higher-priority traffic first
- Fair within a class (weighted, max-min)

Solve at the granularity of DCs

- Split DC-level allocation fairly among services
- Derive switch configuration by leveraging network symmetry

Achieving Max-Min Fairness

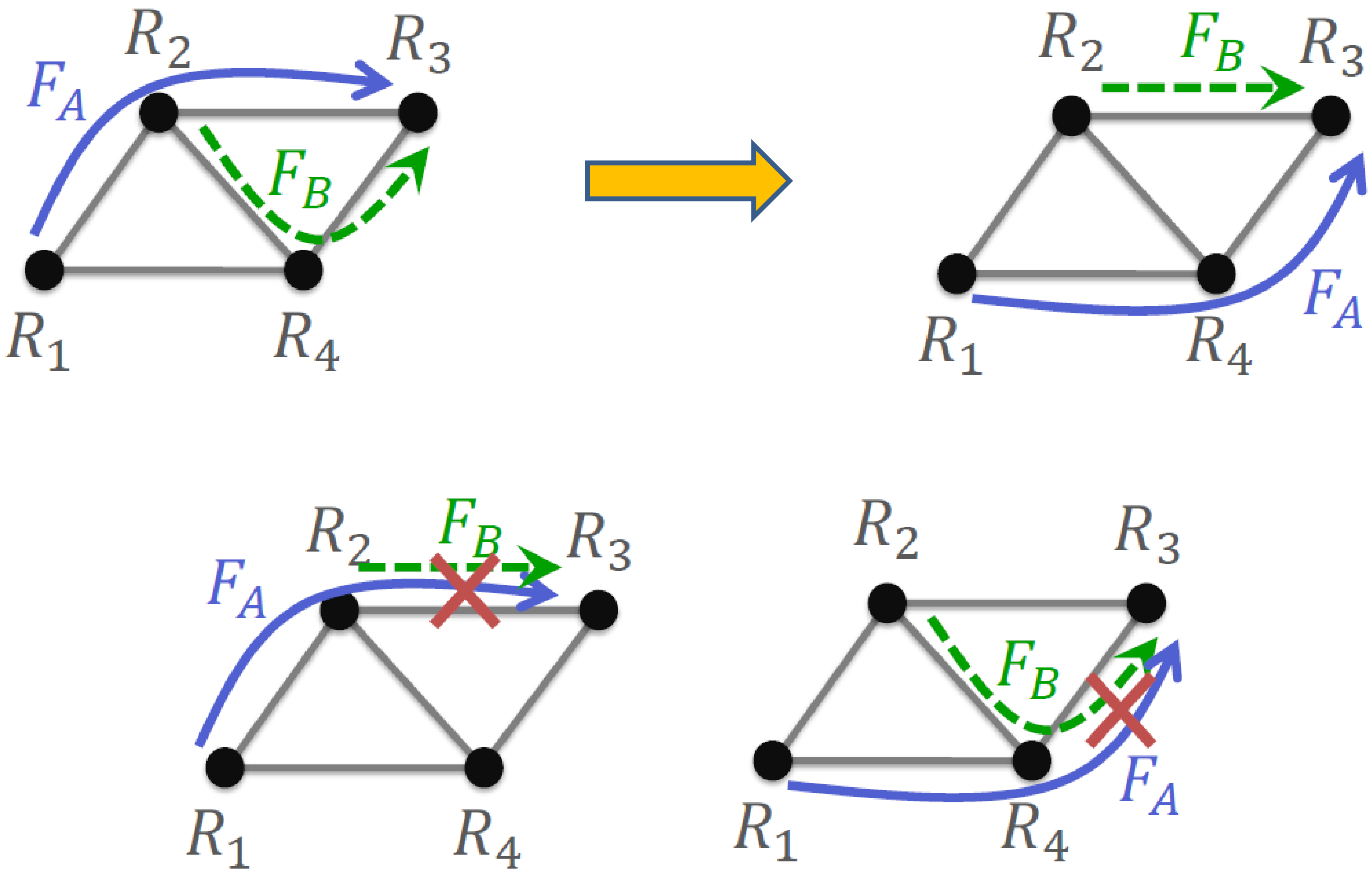
Why is network-wide max-min fairness hard?

- Requires progressive water filling
- Freeze rates whenever a link becomes congested

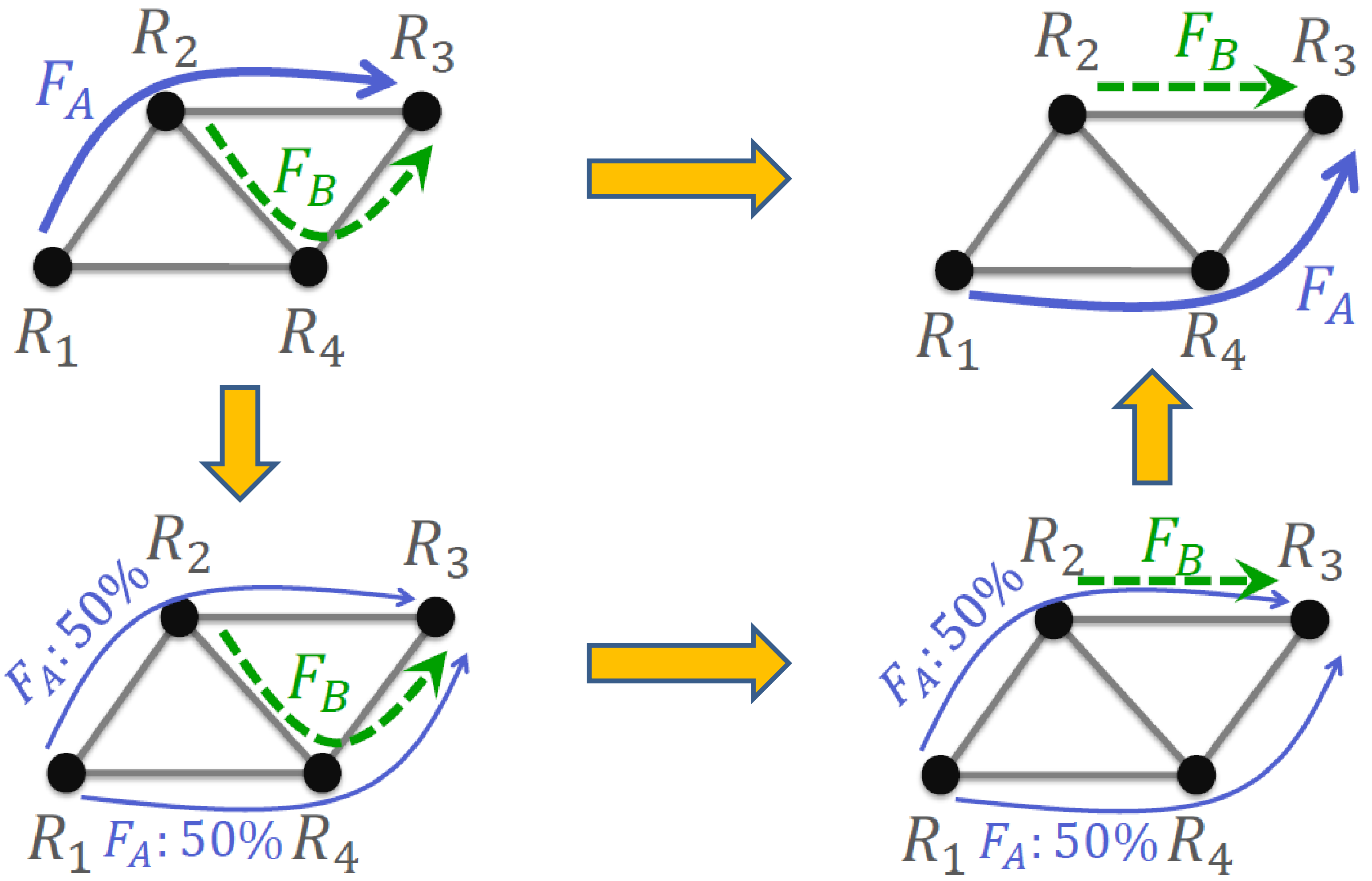
Our approach

- Geometrically partitions the rate space with param α
- At i'th step, classes receive rate up to $\alpha^i U$
- If class gets lower rate, then its rate is held fixed in subsequent iterations
- We prove that rates within $[1/\alpha, \alpha]$ of fair rate

Congestion during network updates



Congestion-free network updates



Computing congestion-free update plans

Leave scratch capacity s on each link

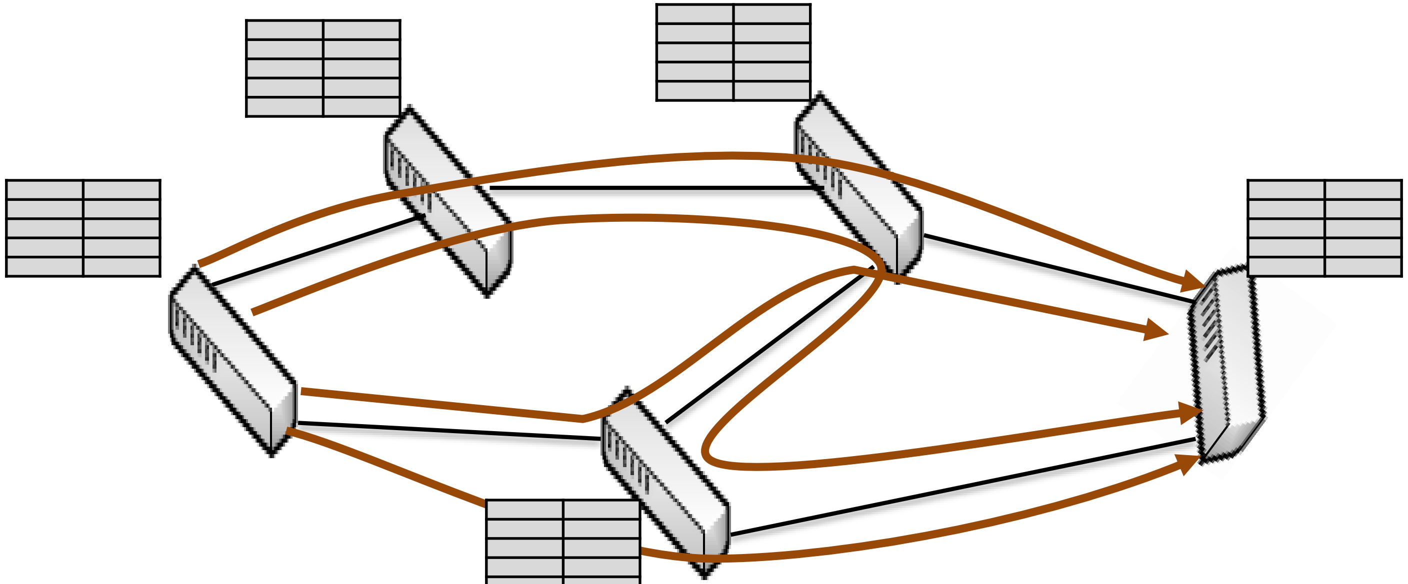
- Ensures a plan with at most $\left\lceil \frac{1}{s} \right\rceil - 1$ steps

Find a plan with minimal number of steps using an LP

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

Use scratch capacity for background traffic

Working with limited switch memory



Use tunnel-based forwarding

Install only the "working set" of tunnels

- Efficient mechanisms to update the set

Updating the set of tunnels

Challenge:

- Must add before remove

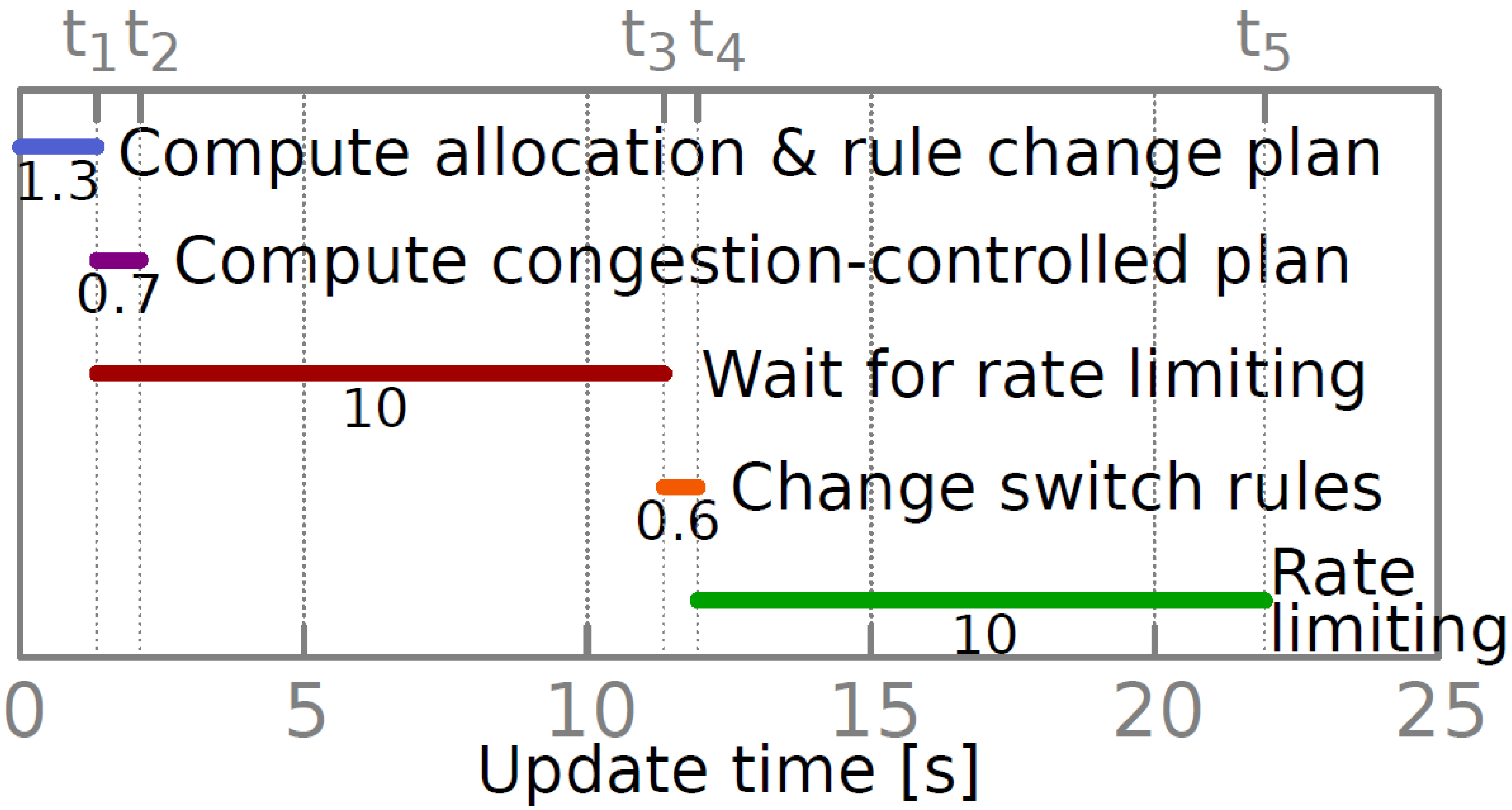
Our approach:

- Leave scratch rule capacity of λ
- Compute a multi-step transition plan
  - Add and remove $\lambda.M$ tunnels in each step
  - Max number of steps is $\left\lceil \frac{1}{\lambda} \right\rceil - 1$

Workflow in each epoch

1. Compute bw allocation, network config.
2. Compute rule change plan
3. Compute bounded-congestion plan
4. Notify services with lower allocation
5. Update the network
6. Notify services with higher allocation

Workflow in each epoch



Prototype

16 OpenFlow switches

- Mix of Blades and Aristas

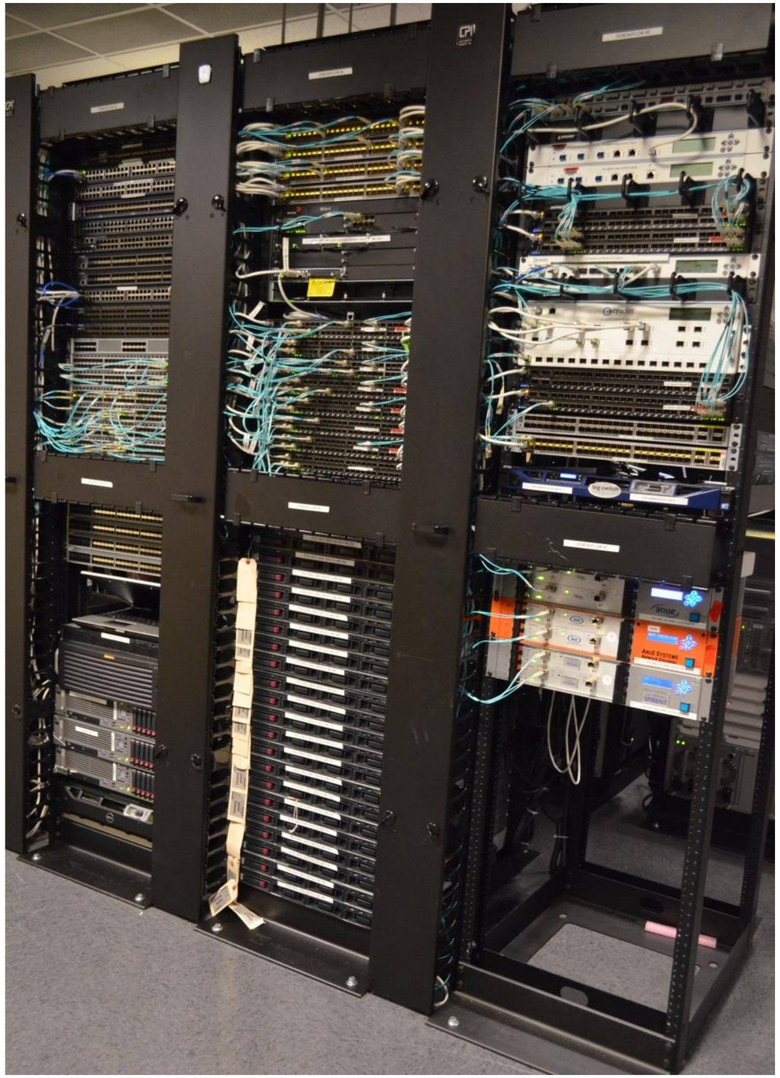
BigSwitch OpenFlow controller

32 servers as traffic sources

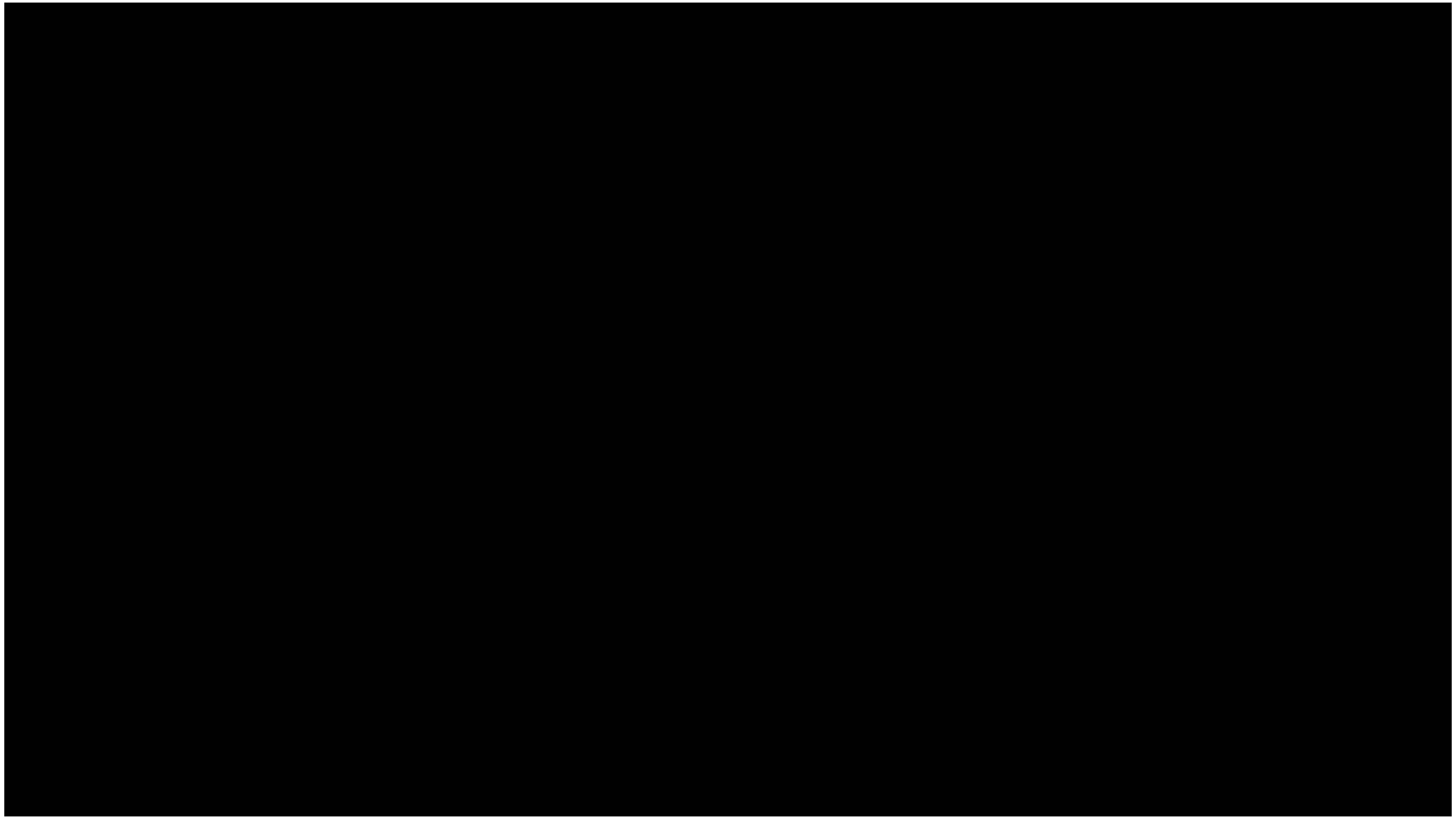
- 25 virtual hosts per server

8 routers (L3)

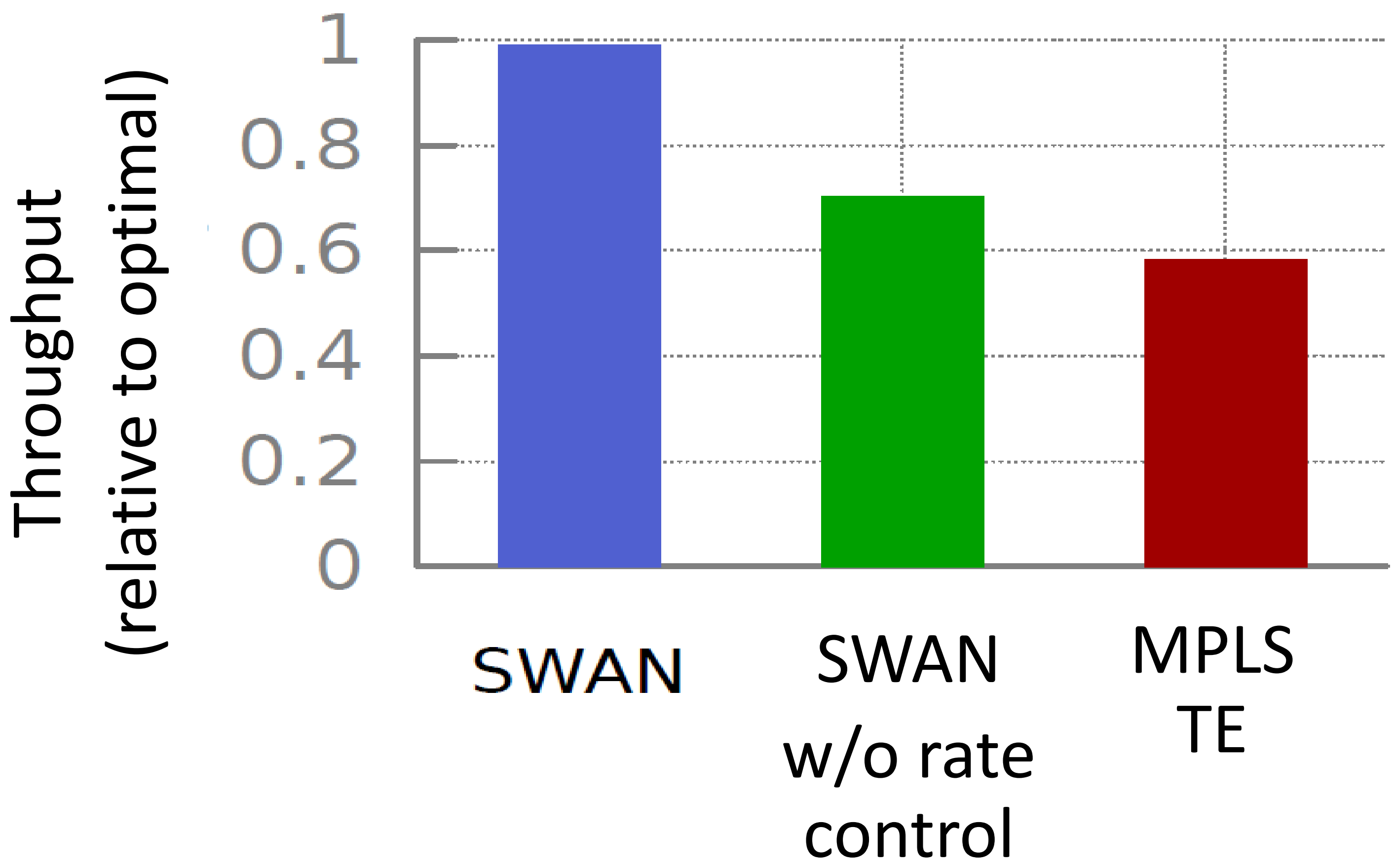
- Mix of Cisco and Juniper



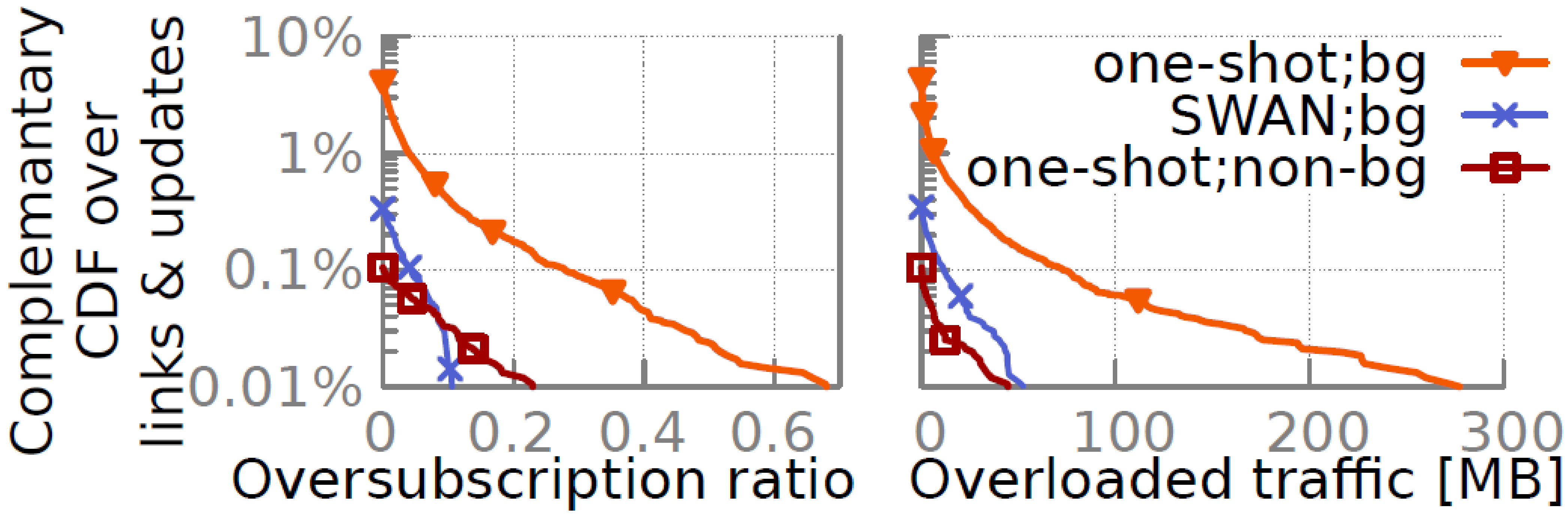
Demo



SWAN comes close to optimal



Network updates: SWAN provides congestion-controlled updates

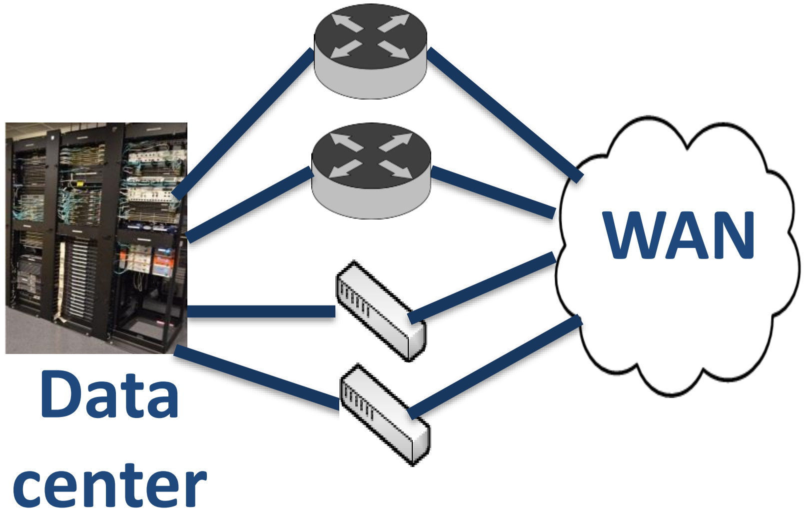


Ongoing work

Wide-area pilot

Resilience to failures and uncertainty

- Algorithms for local failure recovery
- Fast application of updates
- Robust switch software



Summary

SWAN yields a highly efficient and flexible WAN

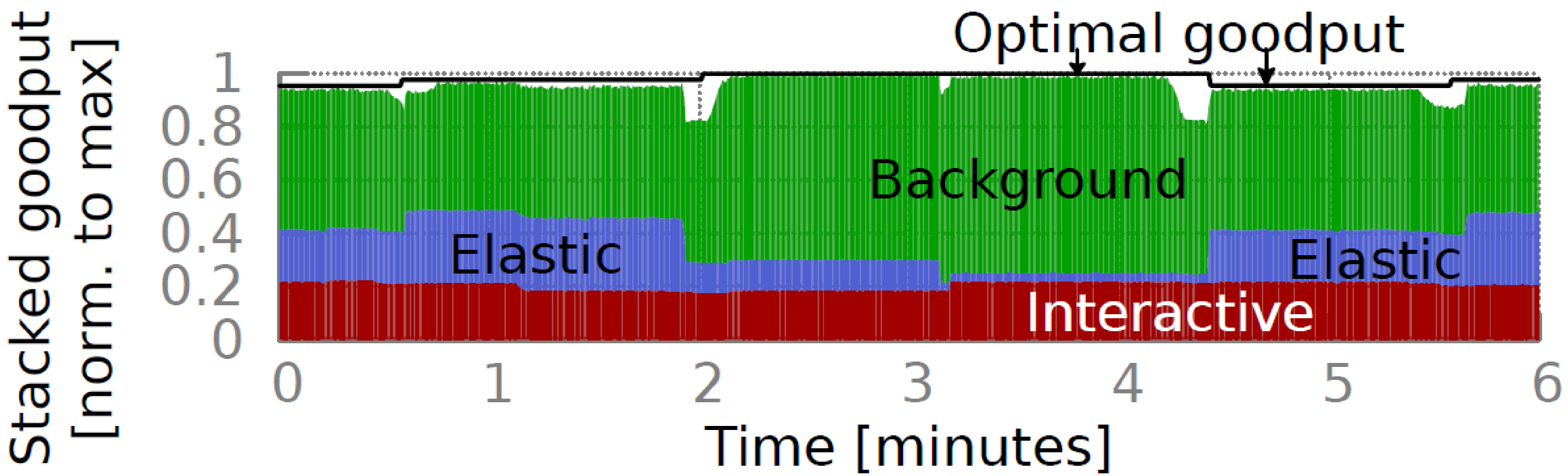
- Coordinates transmissions of services
- Allocates resources centrally
- Manages transitions by using scratch link and memory capacity

High efficiency is key to cost-effective cloud services

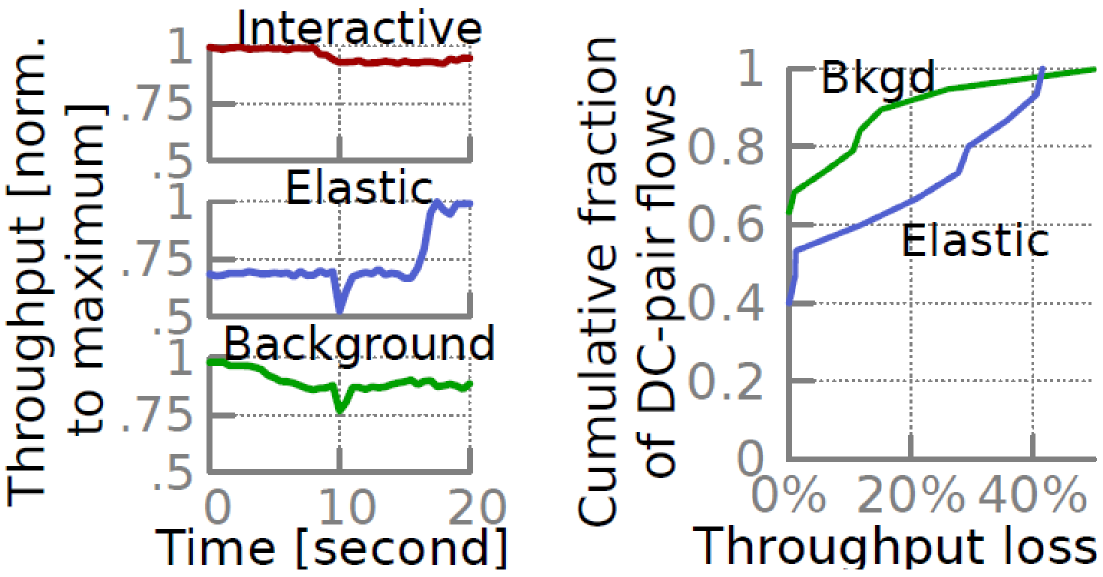
- Many avenues for impactful research
- Opportunity to be "clean slate"

Backup

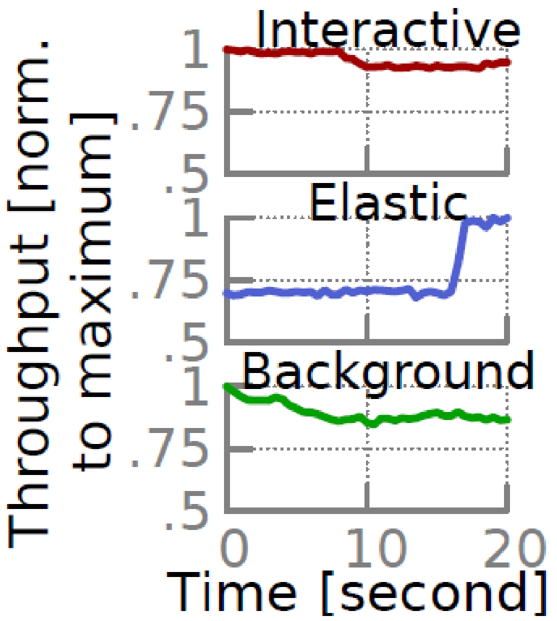
SWAN comes close to optimal (testbed)



No transient congestion during updates with SWAN

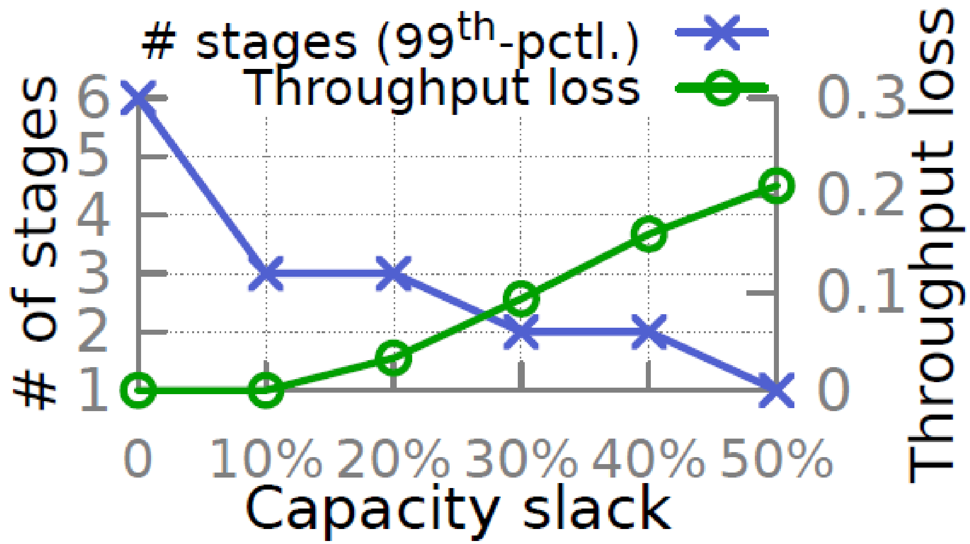


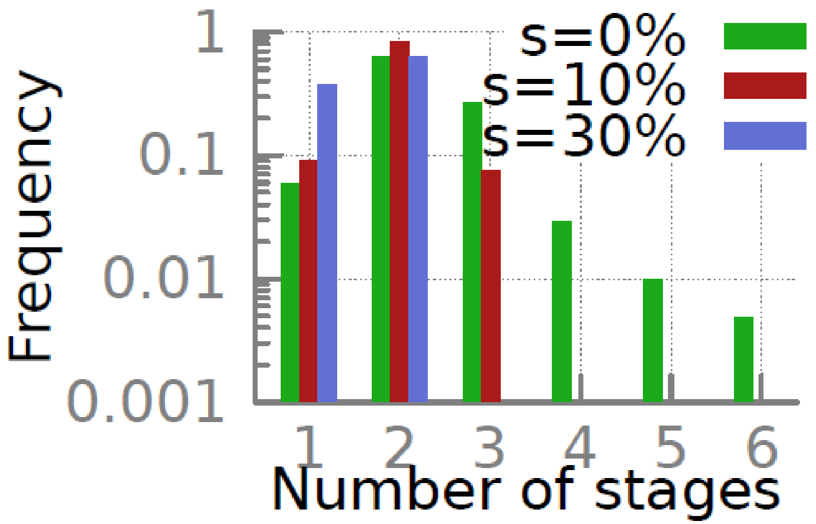
One shot updates



SWAN

Network updates: Impact of *s*





s = ~10% leads to quick updates and little throughput loss

SWAN's dynamic tunnel management needs little memory and is nimble