Programmable Intelligent Networks: Opportunities and Challenges

Stefan Schmid

“We cannot direct the wind,
but we can adjust the sails.”
(Folklore)

Acknowledgements:

Programmable Intelligent Networks: Opportunities and Challenges

Stefan Schmid

“We cannot direct the wind,
but we can adjust the sails.”
(Folklore)

Two tales:
performance and
dependability

Acknowledgements:

Programmable Intelligent Networks: Opportunities and Challenges

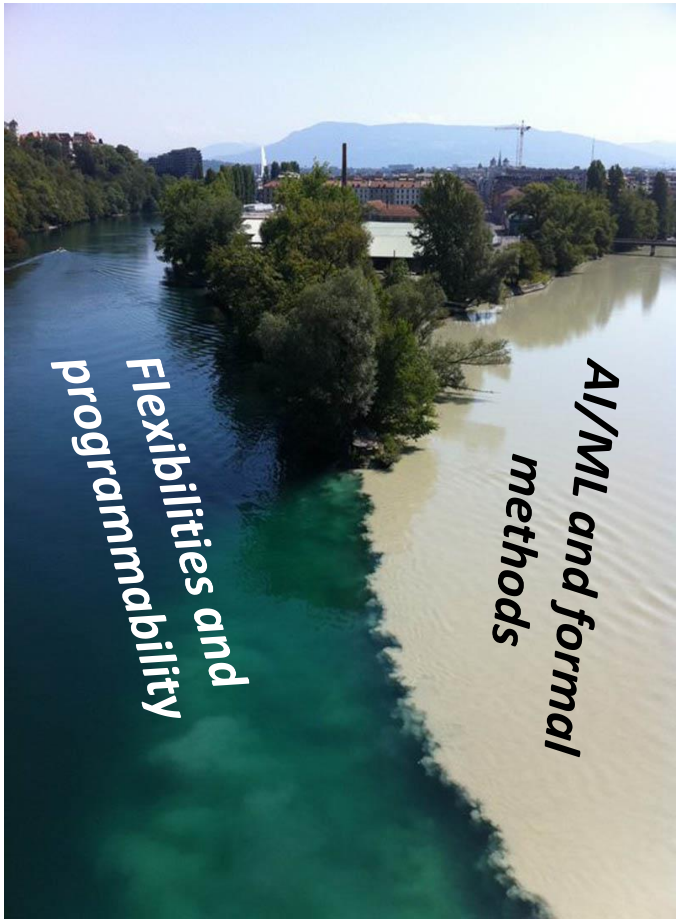
Stefan Schmid

"We cannot direct the wind,
but we can adjust the sails."
(Folklore)

Proudly hosting
IEEE NetSoft
2026 ☺

Acknowledgements:

It`s a Great Time to Be a IEEE NetSoft Researcher!



It`s a Great Time to Be a IEEE NetSoft Researcher!

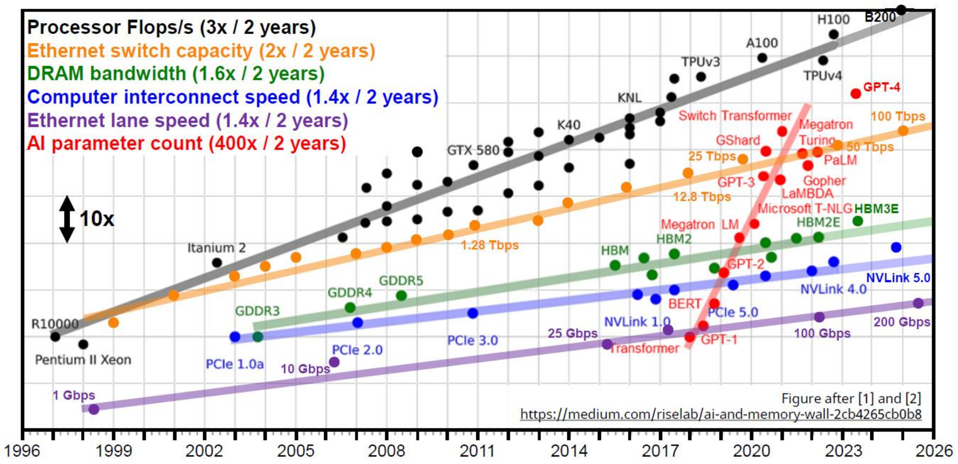
It`s a Great Time to Be a IEEE NetSoft Researcher!

Enables and motivates **self-driving networks**!

Time is right indeed

Network performance is critical

⇢ Increasing gap between network and compute



Credits: Nicola Calabretta

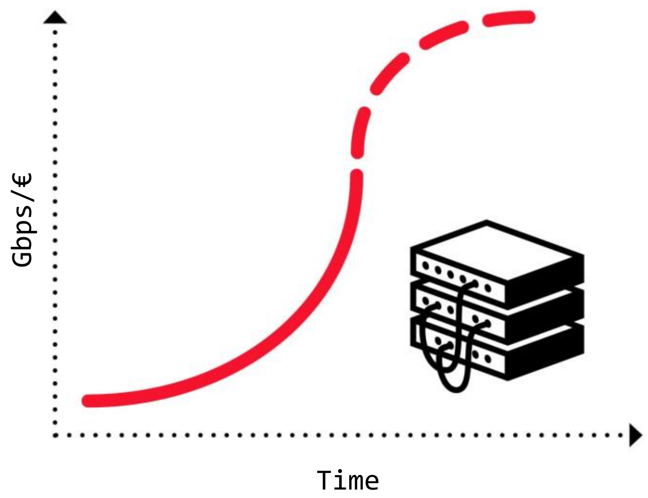
Time is right indeed

Network performance is critical

- In general: transistor density rates, power density rates are stalling

- Hence: more equipment, larger networks

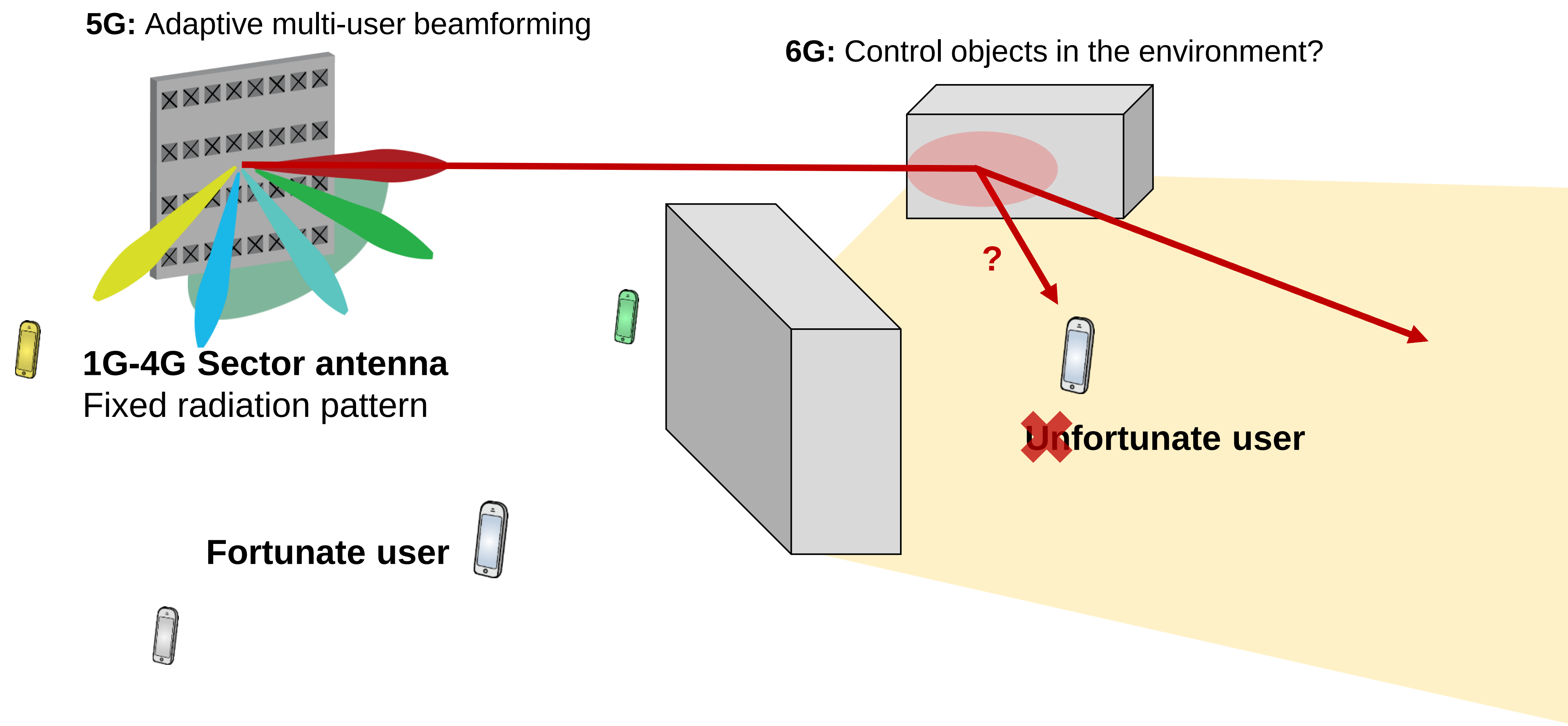
- Resource intensive and: **inefficient**



Emerging Flexibilities

From generation to generation more…

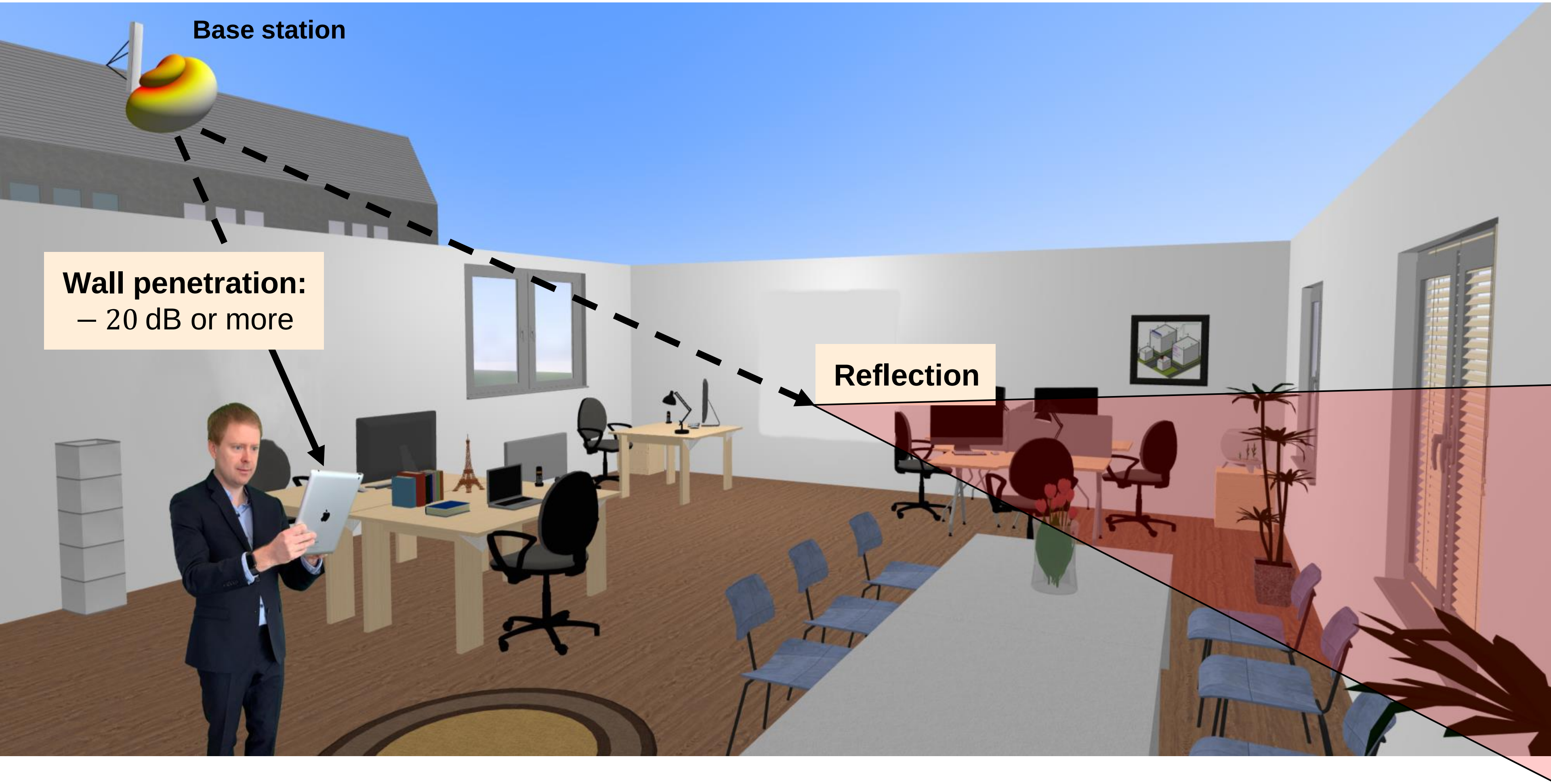
Flexibilities in Cellular



credit: Emil Björnson, Christos Liaskos

Traditionally limited by

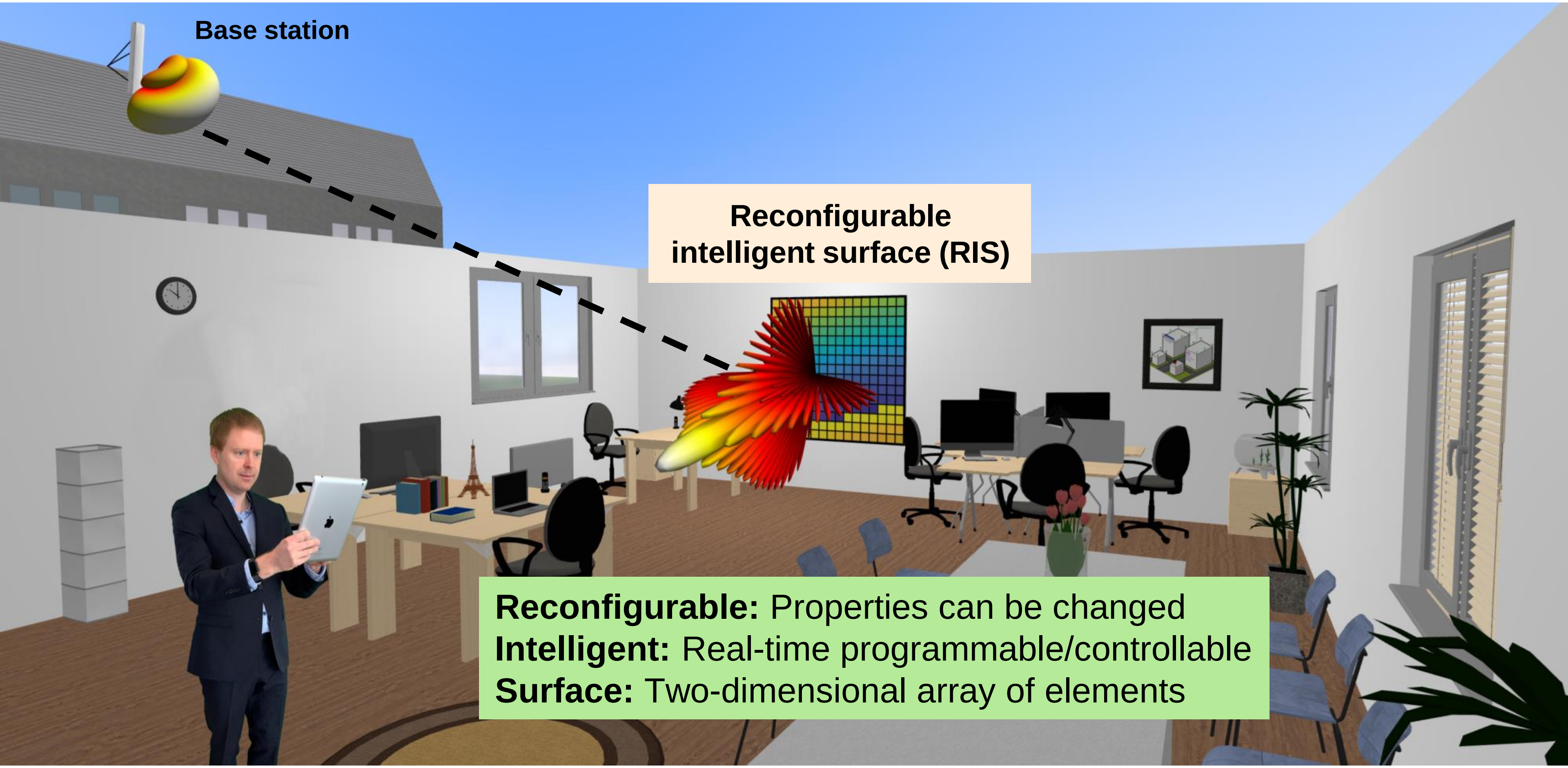
Line of Sight Only



credit: Emil Björnson

Beyond Line of Sight: Virtual LoS with

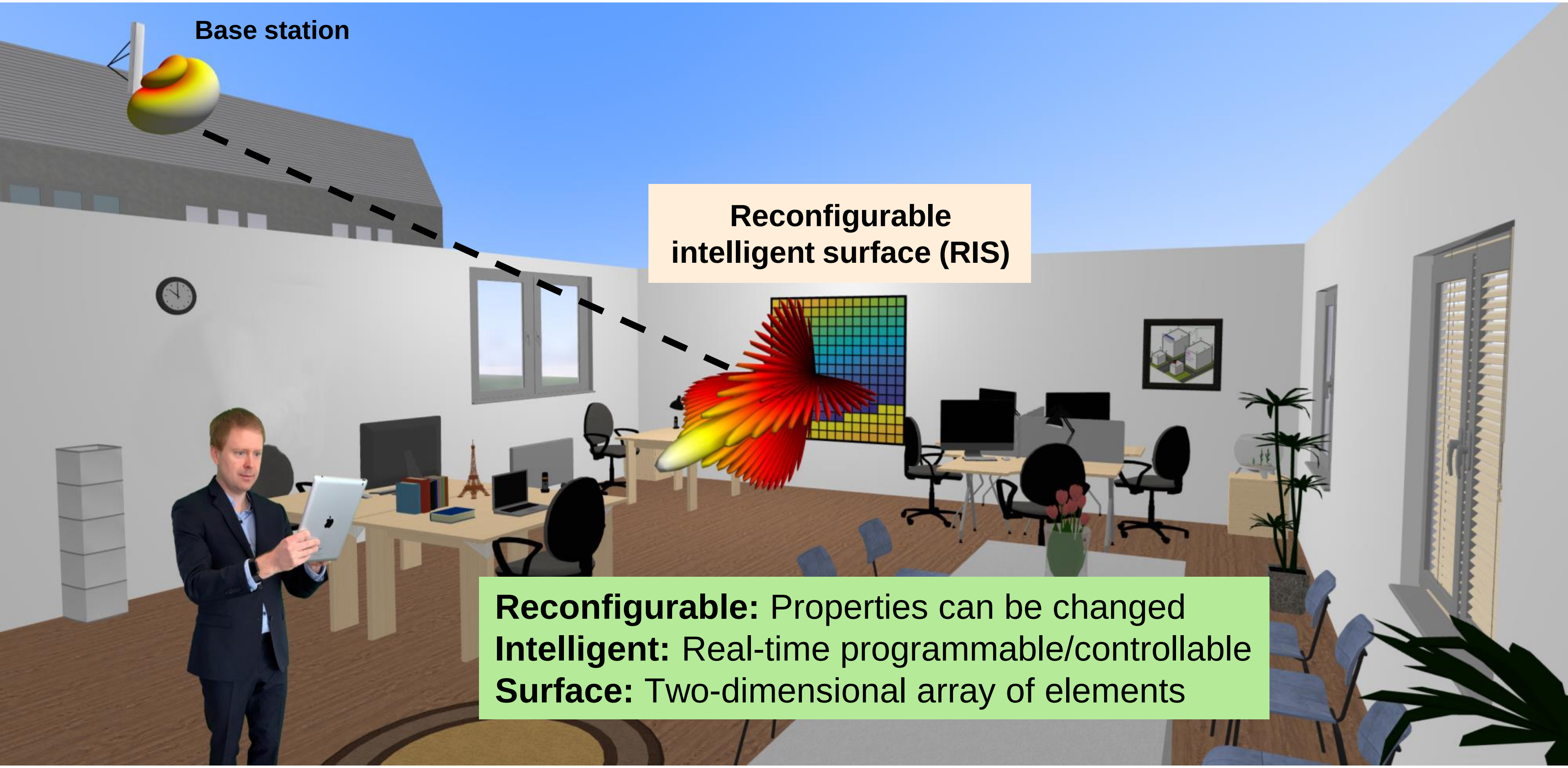
Programmable Surfaces



Reconfigurable: Properties can be changed
Intelligent: Real-time programmable/controllable
Surface: Two-dimensional array of elements

credit: Emil Björnson

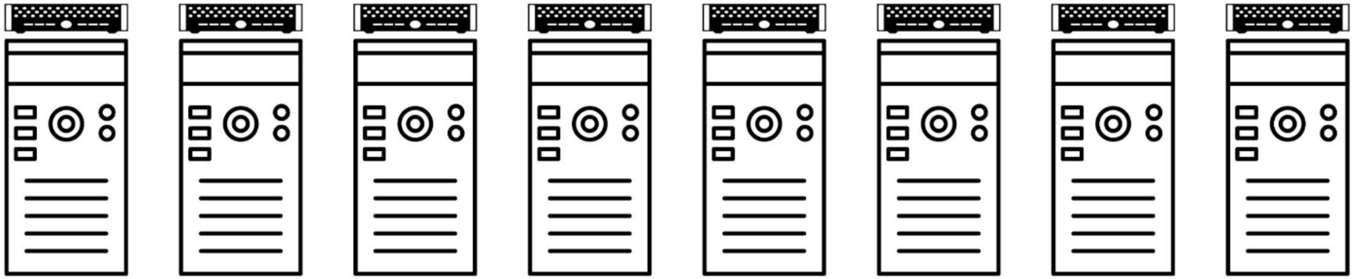
Beyond Line of Sight: Virtual LoS with Programmable Surfaces



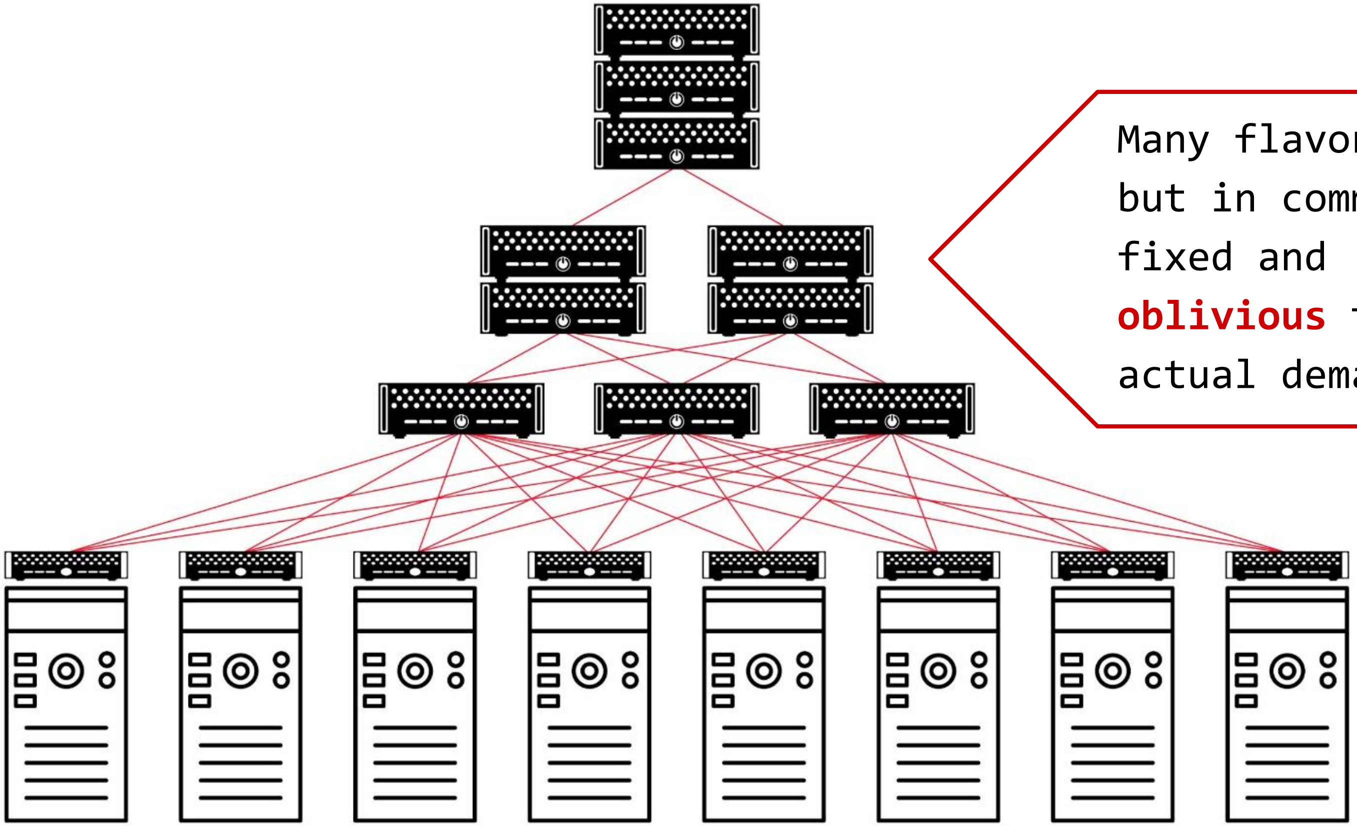
Literature: Software-Defined Reconfigurable Intelligent Surfaces: From Theory to End-to-End Implementation. Liaskos et al. Proceedings IEEE, 2022.

Another Example: Flexibilities with Topology Programming

How to interconnect?



Another Example: Flexibilities with Topology Programming

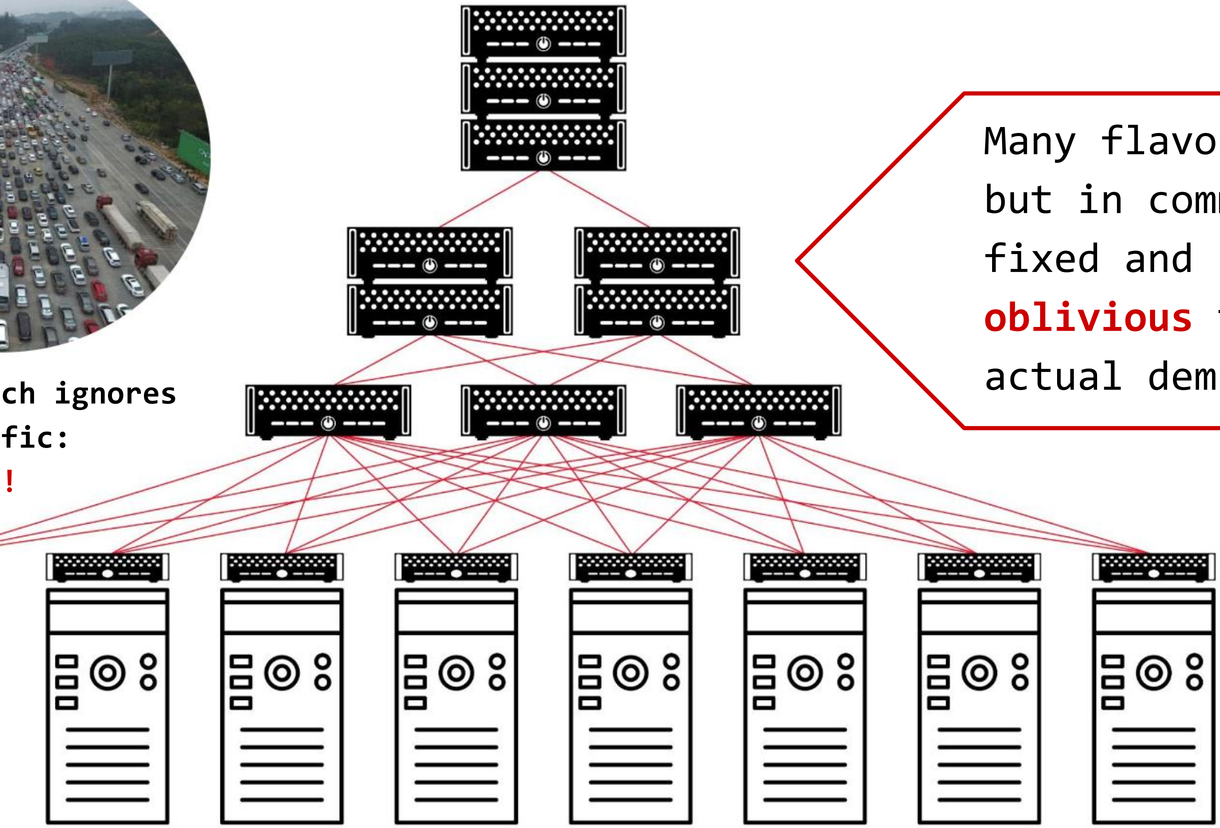


Many flavors, but in common: fixed and **oblivious** to actual demand.

Another Example: Flexibilities with

Topology Programming

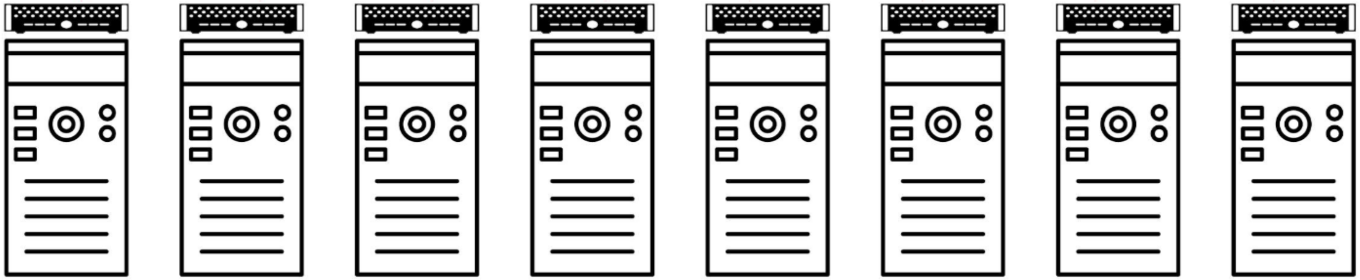
Highway which ignores actual traffic: **frustrating!**



Many flavors, but in common: fixed and **oblivious** to actual demand.

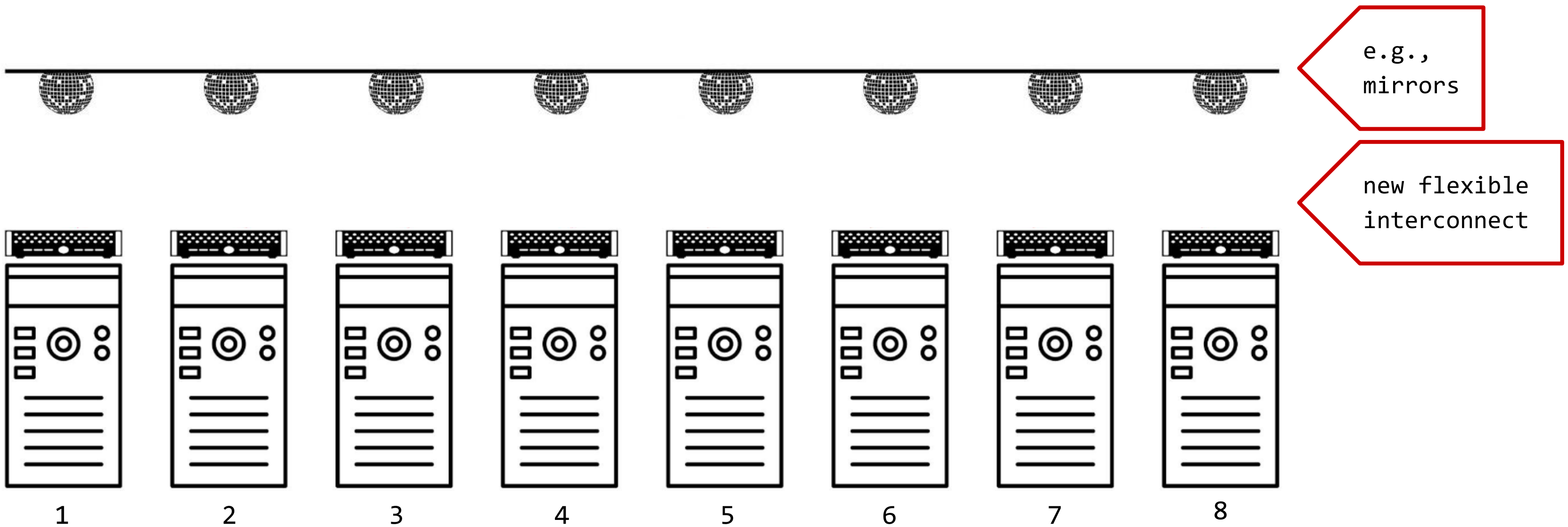
Another Example: Flexibilities with

Topology Programming



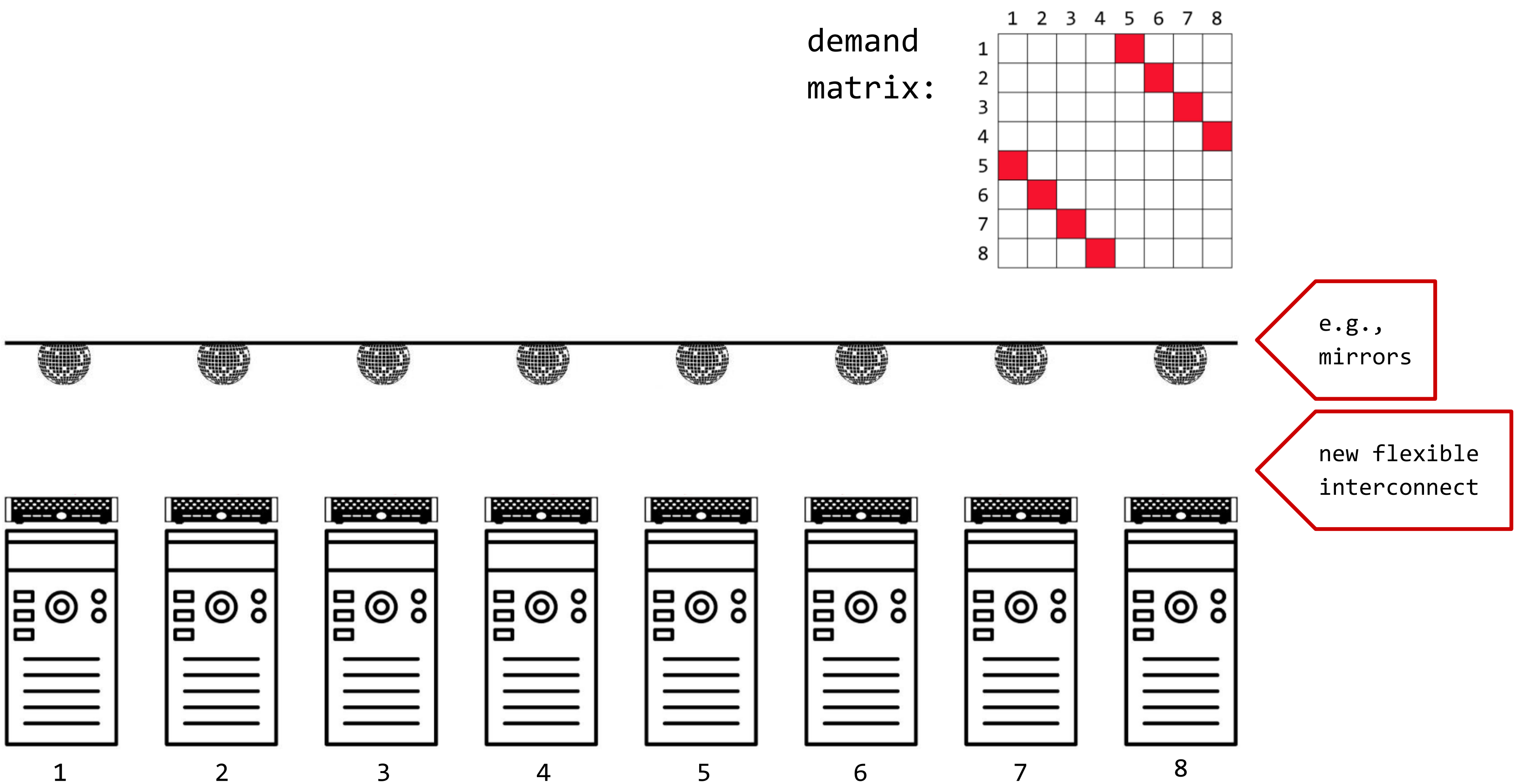
Another Example: Flexibilities with

Topology Programming



Another Example: Flexibilities with

Topology Programming

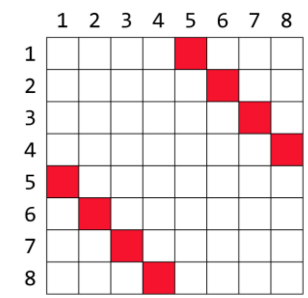


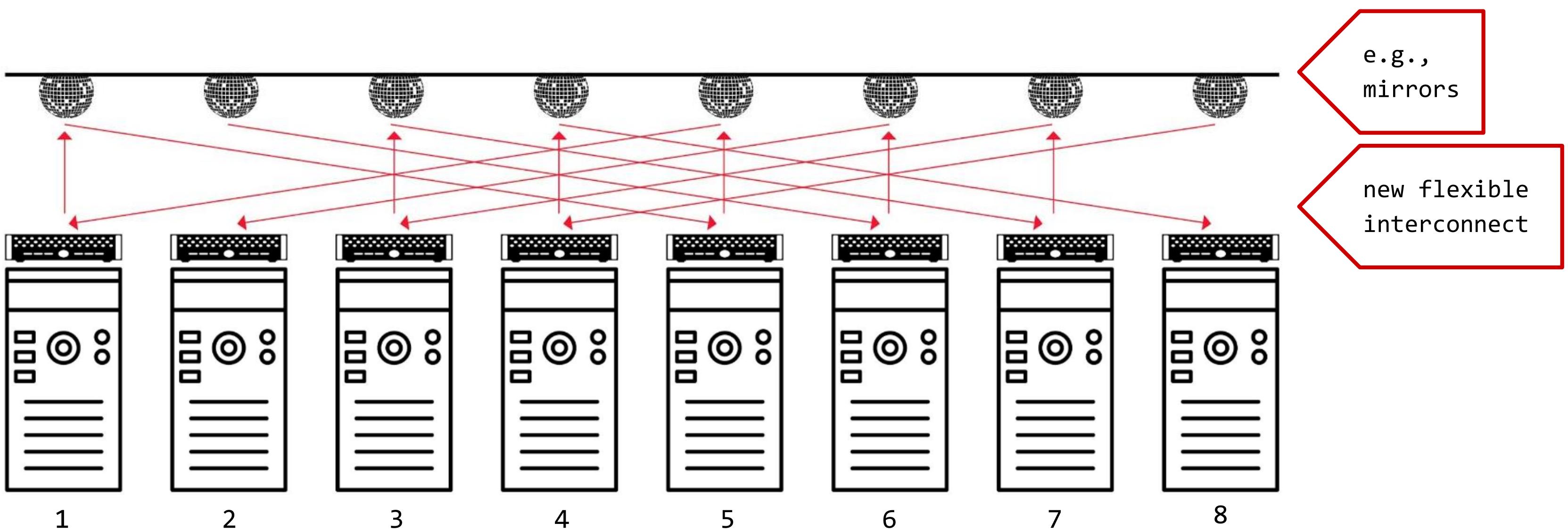
Another Example: Flexibilities with

Topology Programming

Matches demand

demand matrix:

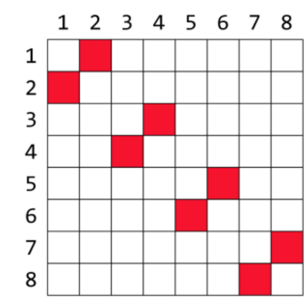


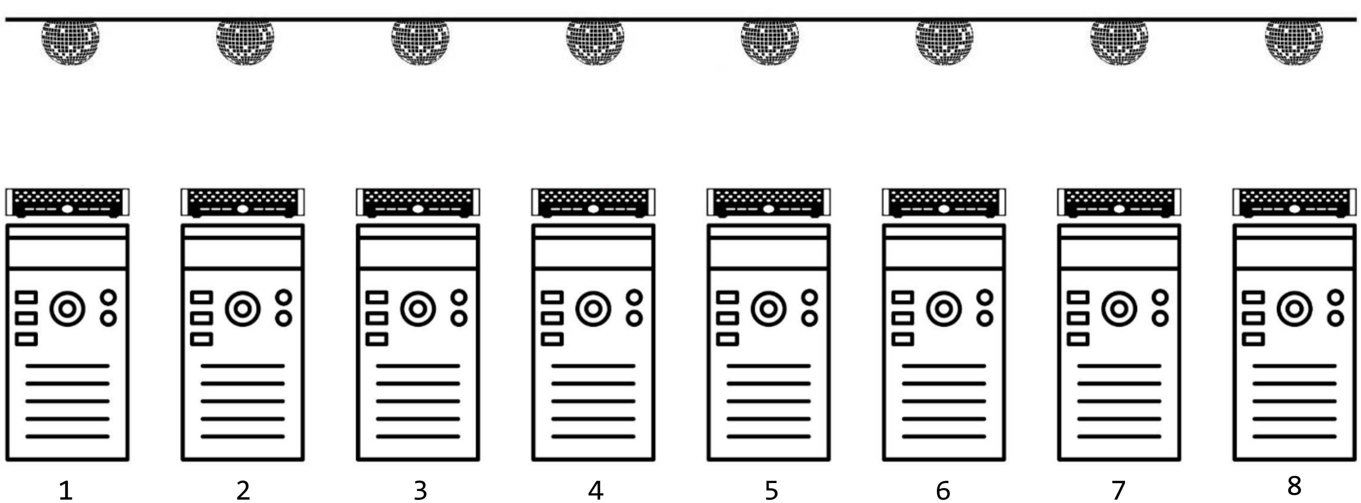


Another Example: Flexibilities with

Topology Programming

new demand:





e.g., mirrors

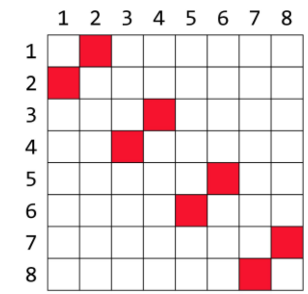
new flexible interconnect

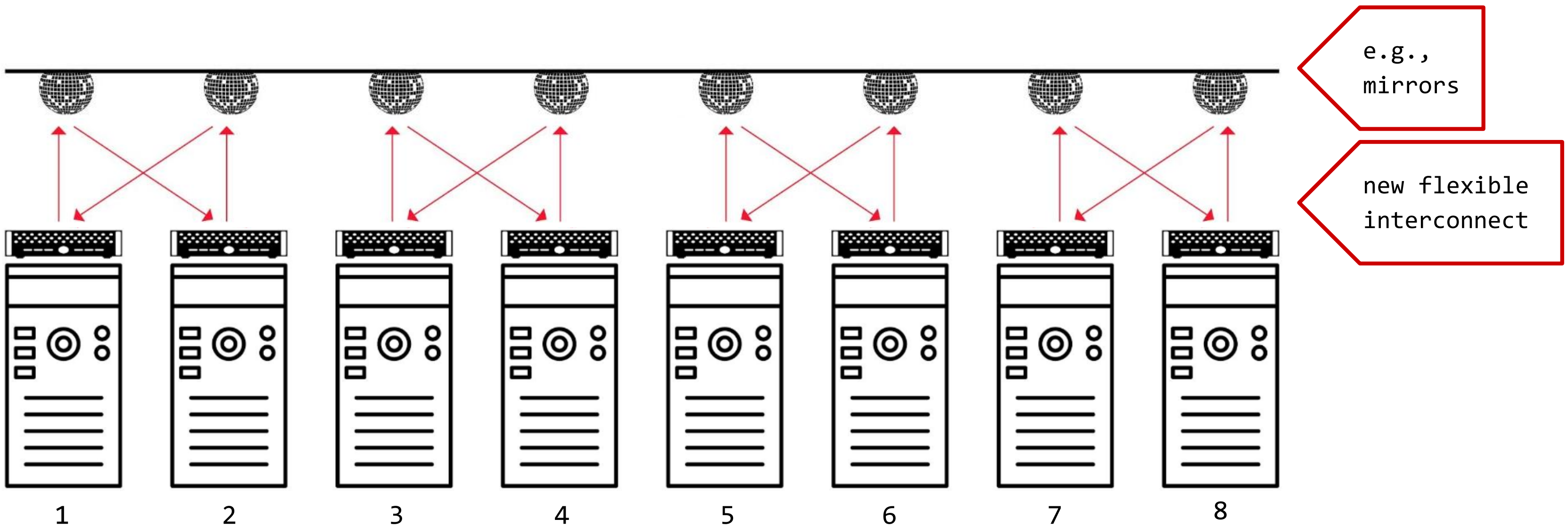
Another Example: Flexibilities with

Topology Programming

Matches demand

new demand:



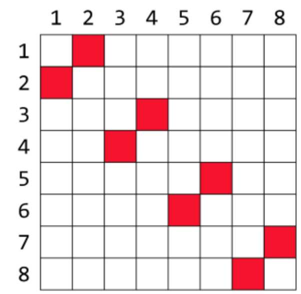


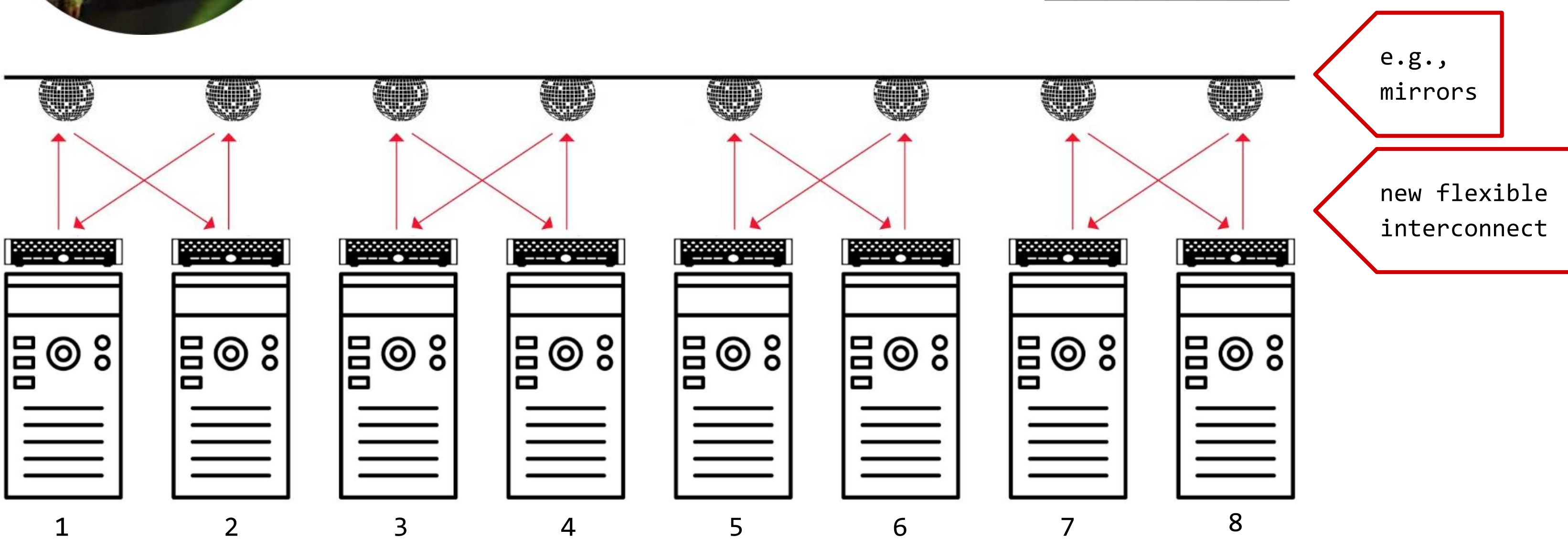
Another Example: Flexibilities with

Topology Programming

Self-Adjusting Networks

new demand:

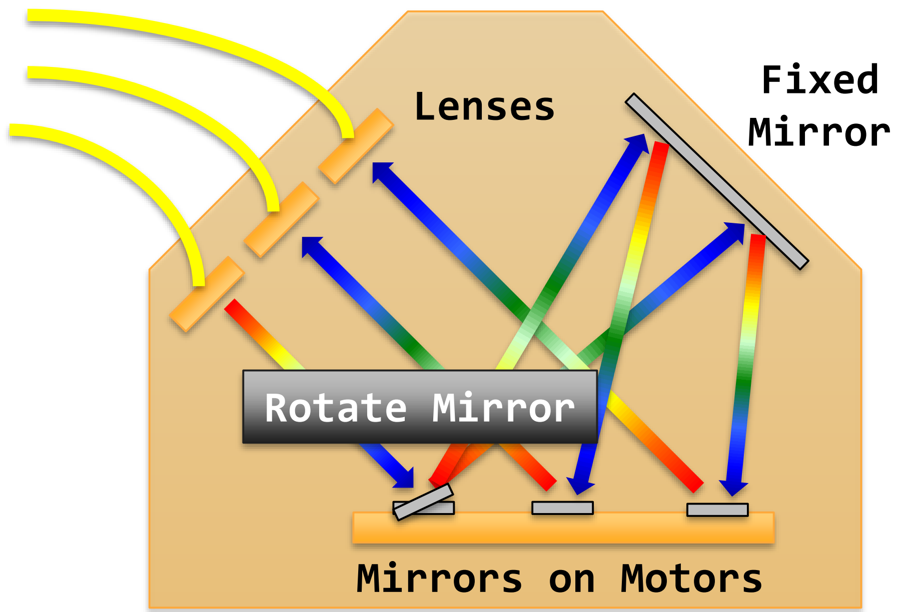




Reconfigurable Optics

Optical Circuit Switch

- Optical Circuit Switch rapid adaption of physical layer
  - Based on rotating mirrors



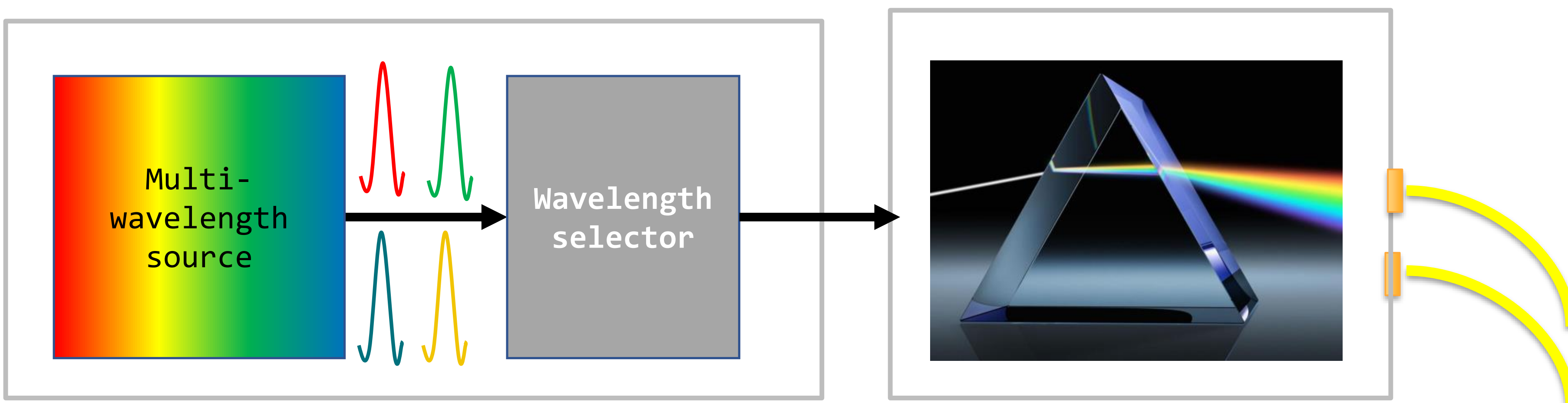
Optical Circuit Switch

By Nathan Farrington, SIGCOMM 2010

Another Example

Tunable Lasers (e.g., Microsoft's Sirius)

- Depending on wavelength, forwarded differently
- Optical switch is passive



Electrical switch with tunable laser

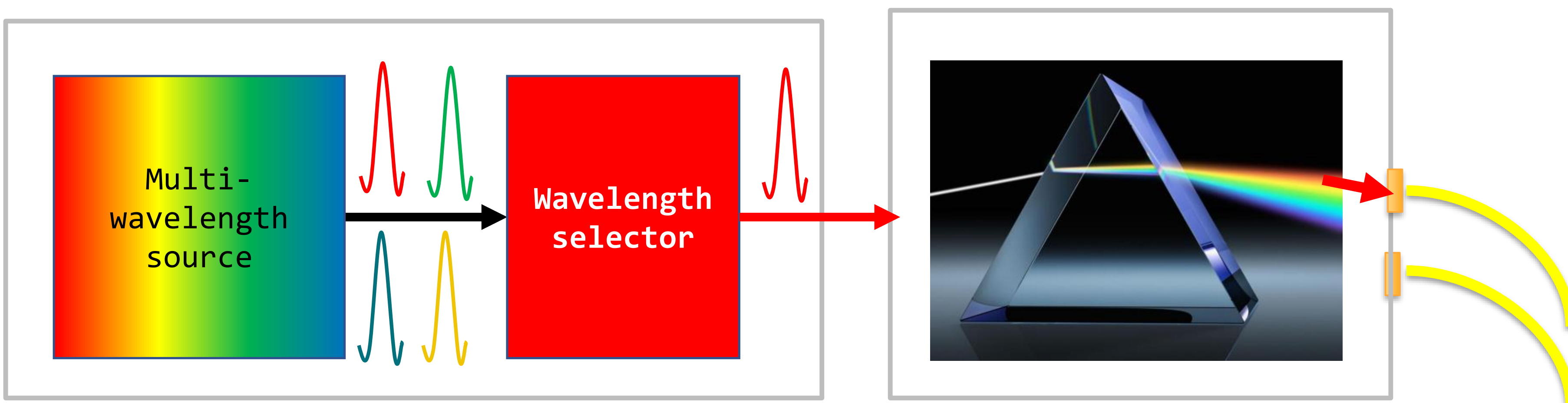
Optical switch
Passive

Ballani et al., Sirius, ACM SIGCOMM 2020.

Another Example

Tunable Lasers (e.g., Microsoft's Sirius)

⇢ Depending on wavelength, forwarded differently
⇢ Optical switch is passive



Electrical switch with tunable laser

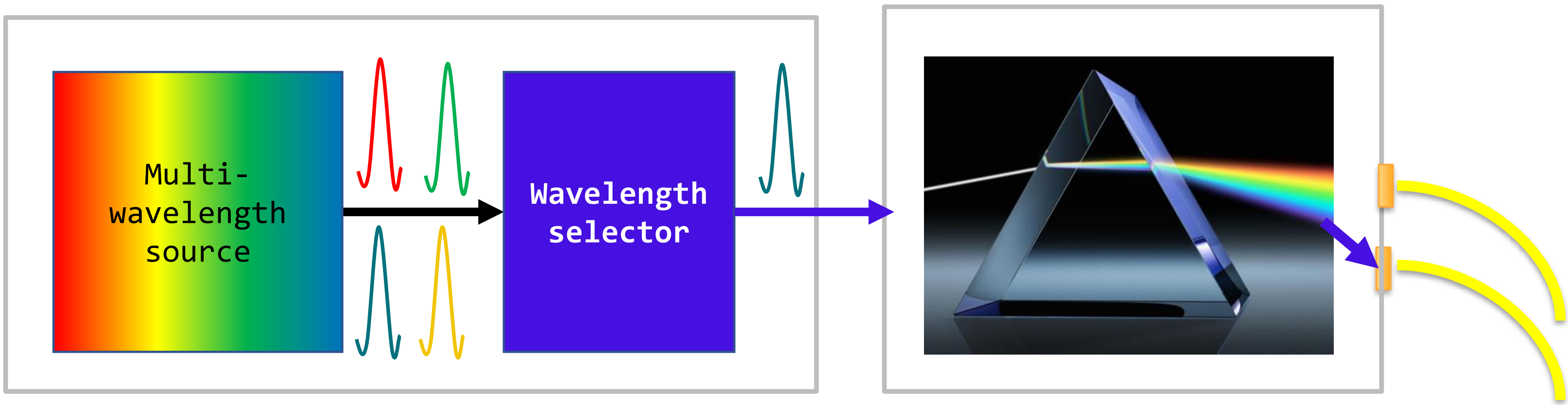
Optical switch
Passive

Ballani et al., Sirius, ACM SIGCOMM 2020.

Another Example

Tunable Lasers (e.g., Microsoft’s Sirius)

- ⇢ Depending on wavelength, forwarded differently
- ⇢ Optical switch is passive

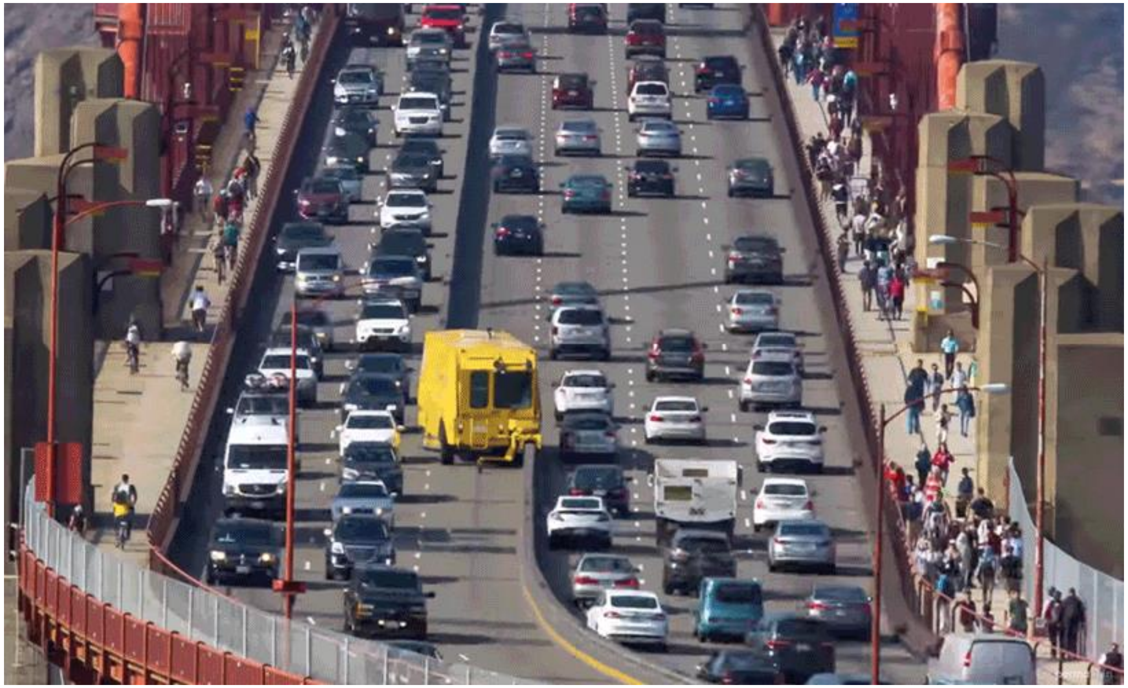


Electrical switch with tunable laser

Optical switch ***Passive***

Ballani et al., Sirius, ACM SIGCOMM 2020.

Analogy



Golden Gate Zipper

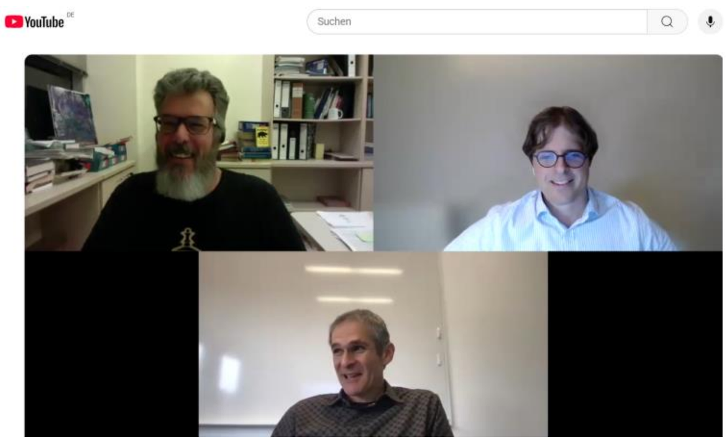
Many research avenues for dynamic networks:

Control and Network Stack

⇢ Scalable control plane such dynamic programmable networks?

⇢ Implications on other layers of the networking stack? How to do routing, congestion control, buffer management on dynamic networks?

See interview with Amin Vahdat, Google in June issue of CACM‘25: https://www.youtube.com/watch?v=IxcV1gu8ETA



Roadmap

Two tales:

- Traffic: structure in traffic = optimization opportunity for NetSoft researchers
- Dependability: Flexibility may introduce complexity, a case for ML and formal methods?

Roadmap

Two tales:

- **Traffic: structure in traffic = optimization opportunity for NetSoft researchers**

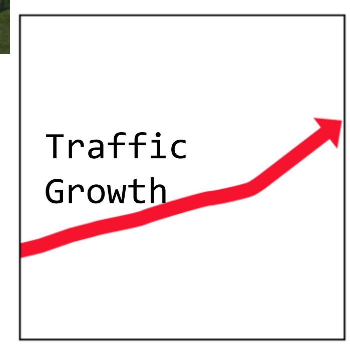
- Dependability: Flexibility may introduce complexity, a case for ML and formal methods?

Why Innovations Needed?

Explosive Traffic

Datacenters (“hyper-scale”)

+network



Interconnecting networks:
a **critical infrastructure**
of our digital society.

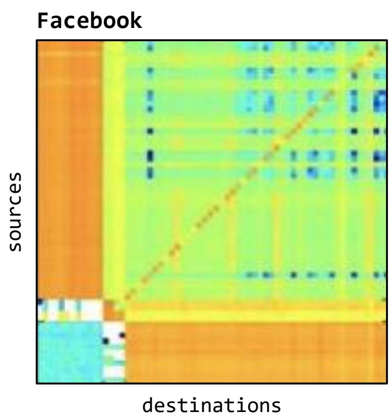
But good news: traffic also has

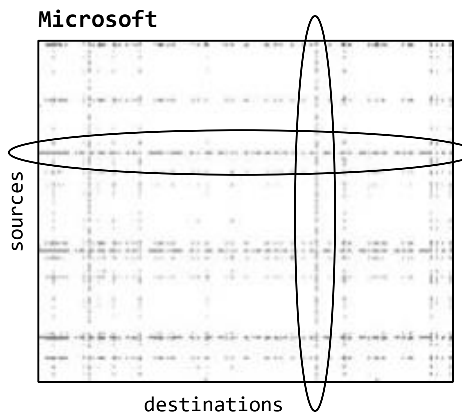
Much Structure

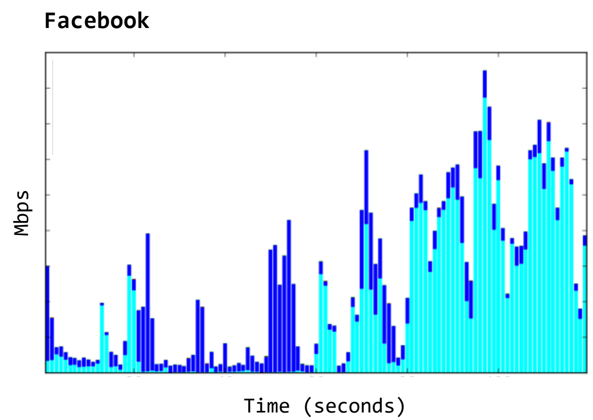
Empirical studies:

traffic matrices sparse and skewed

traffic bursty over time







Can we exploit this for optimization?

Be Aware of Your Application

Traffic Diversity

Diverse patterns:

→ Shuffling/Hadoop:
  all-to-all

→ Collective communications/All-reduce/ML: ring or tree traffic patterns

  → Elephant flows

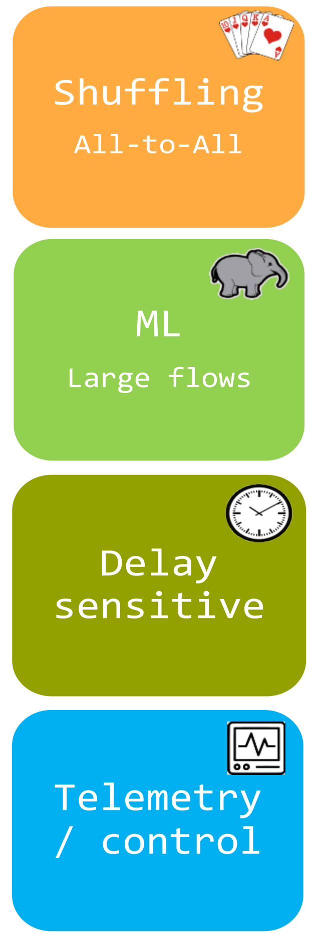
→ Query traffic: skewed

  → Mice flows

→ Control traffic: does not evolve but has non-temporal structure

Diverse requirements:

→ ML is bandwidth hungry, small flows are latency-sensitive

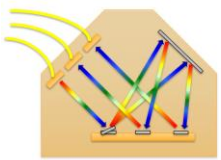


The big picture of

Self-Driving Networks

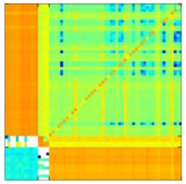
New!

Flexibility



More!

Structure



Self-Driving and Adaptive Networks

Performance, Sustainability, etc.

Now is the time!

A fundamental question:

How much structure is there? And how to measure and model structure in workloads?

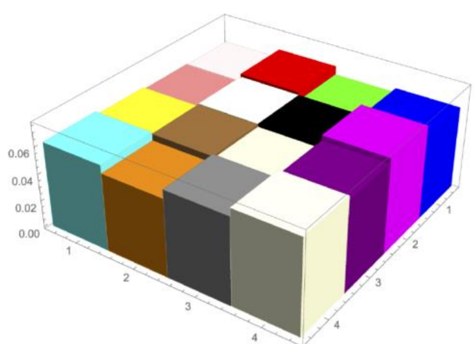
Intuition

Which demand has more structure?

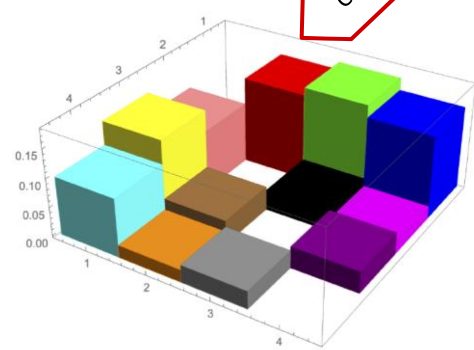
⇢ Traffic matrices of two different distributed ML applications

- ⇀ GPU-to-GPU

Color = communication pair



VS



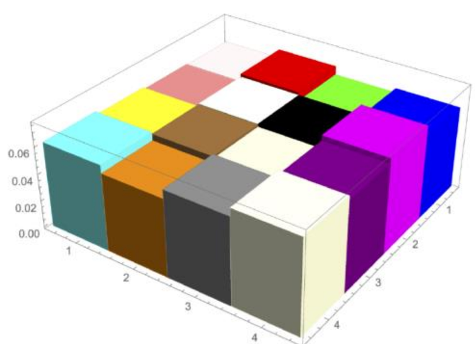
Intuition

Which demand has more structure?

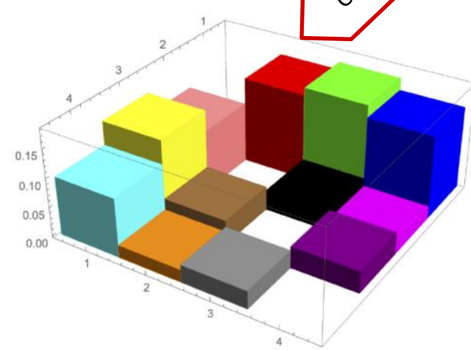
⇢ Traffic matrices of two different distributed ML applications

⇀ GPU-to-GPU

Color = communication pair



VS



More uniform

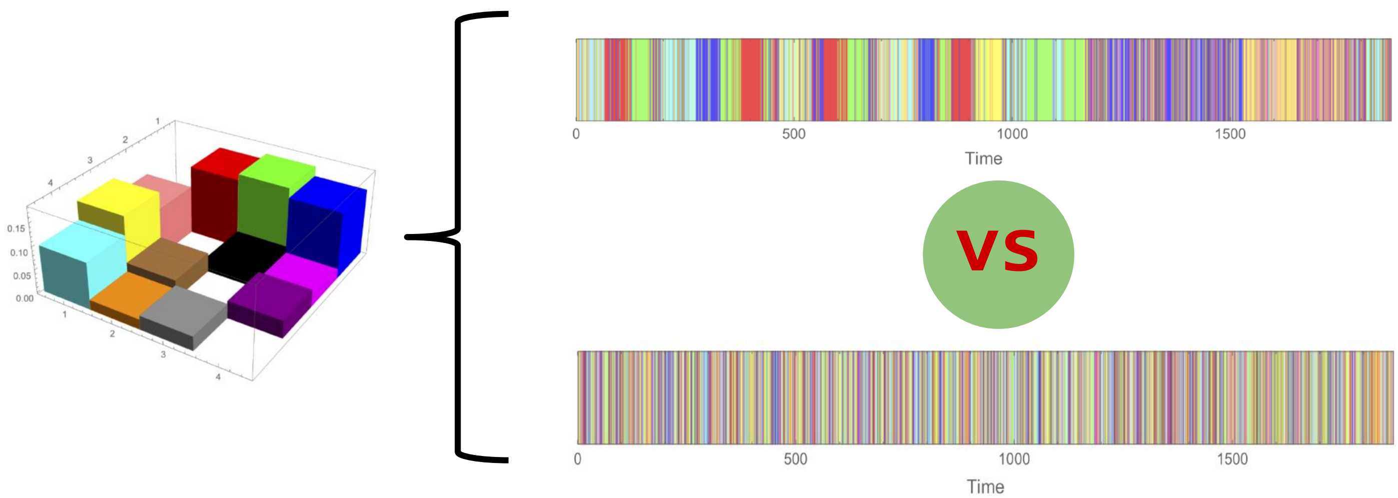
More structure

Intuition

Spatial vs temporal structure

- Two different ways to generate same traffic matrix:
  - Same non-temporal structure

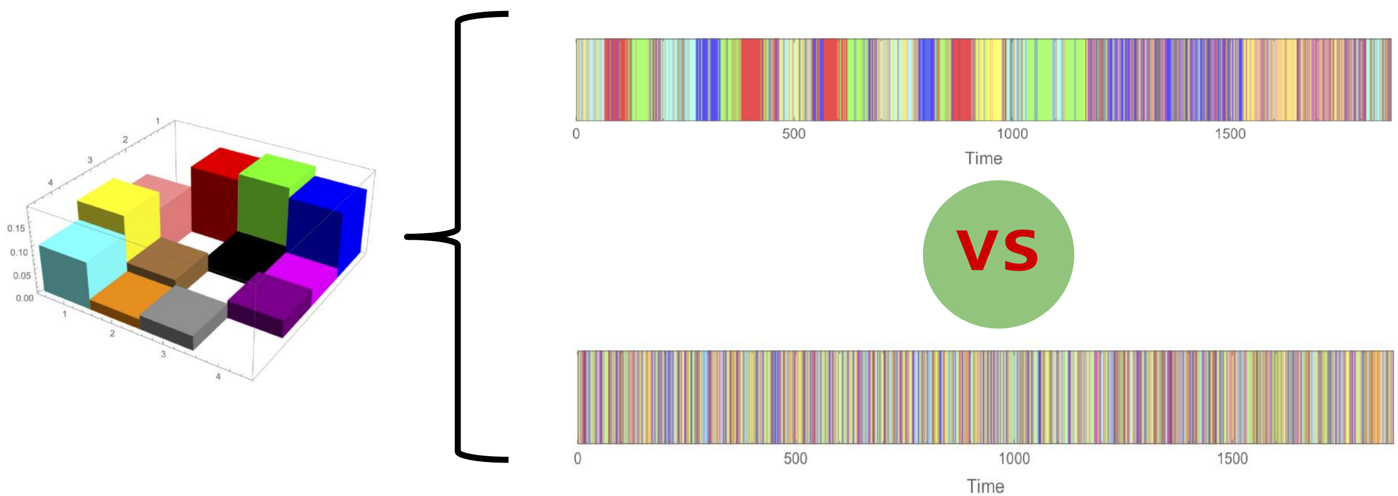
- Which one has more structure?



Intuition

Spatial vs temporal structure

- Two different ways to generate same traffic matrix:
  - Same non-temporal structure
- Which one has more structure?

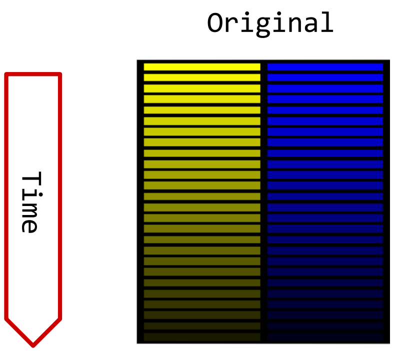


Systematically?

Trace Complexity

Information-Theoretic Approach

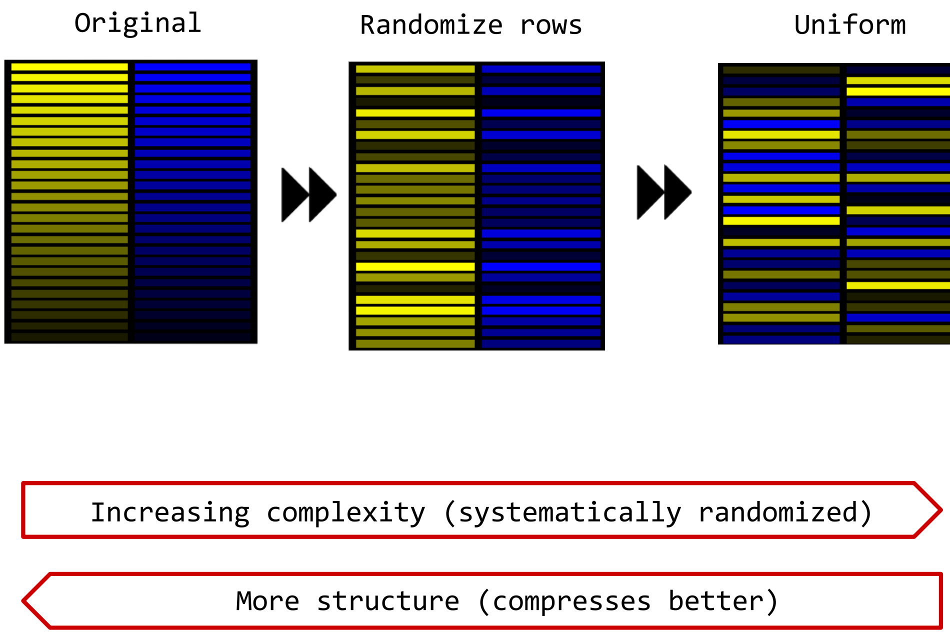
“Shuffle&Compress”



Trace Complexity

Information-Theoretic Approach

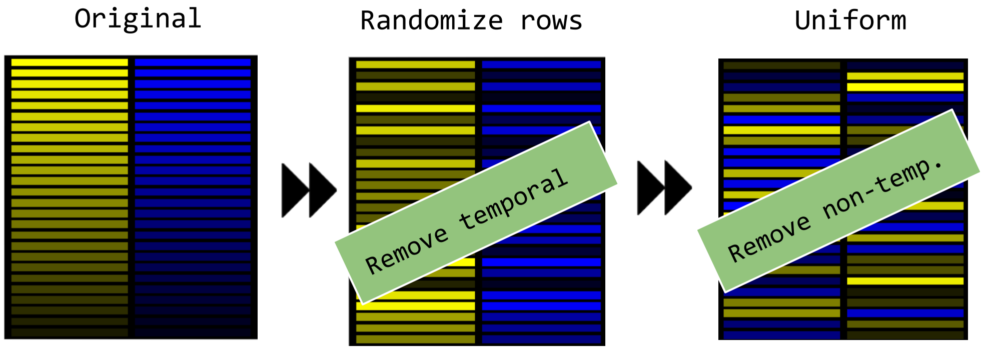
“Shuffle&Compress”



Trace Complexity

Information-Theoretic Approach

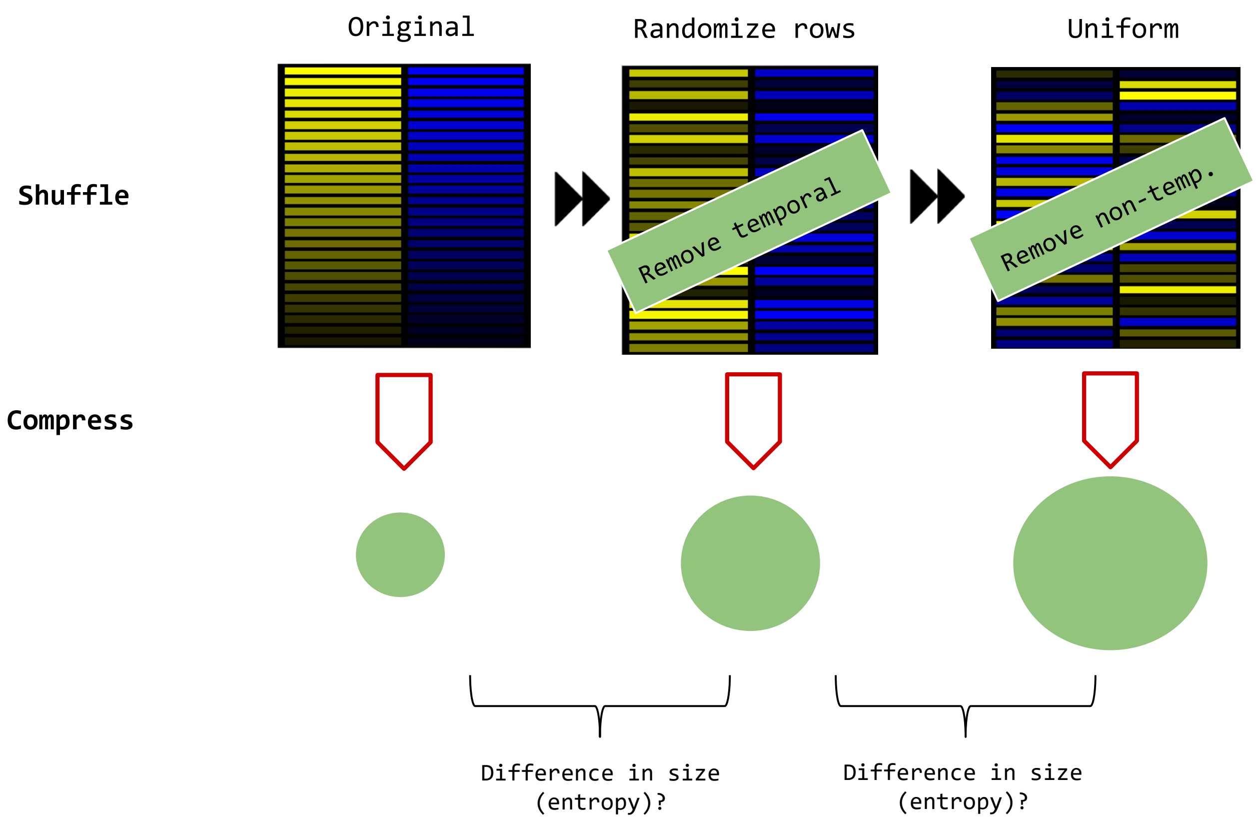
“Shuffle&Compress”



Trace Complexity

Information-Theoretic Approach

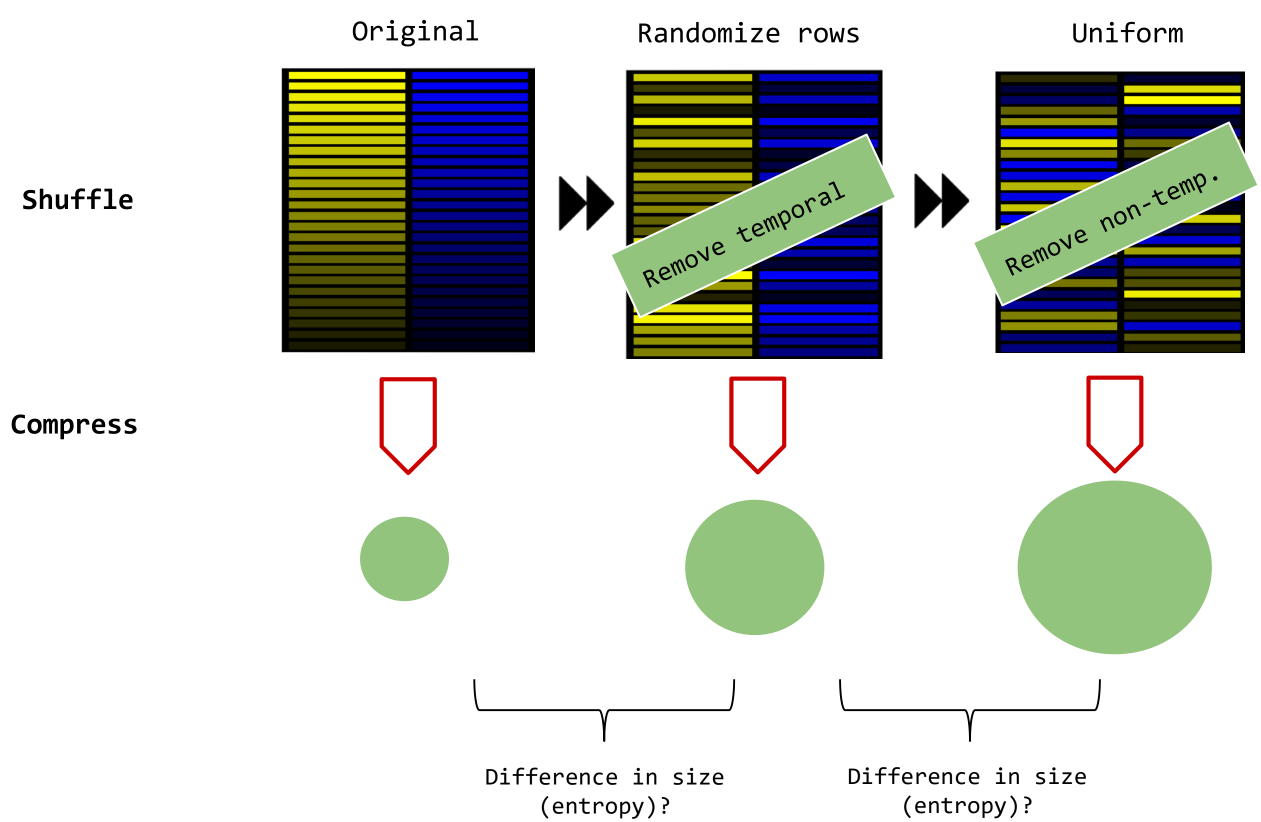
“Shuffle&Compress”



Trace Complexity

Information-Theoretic Approach

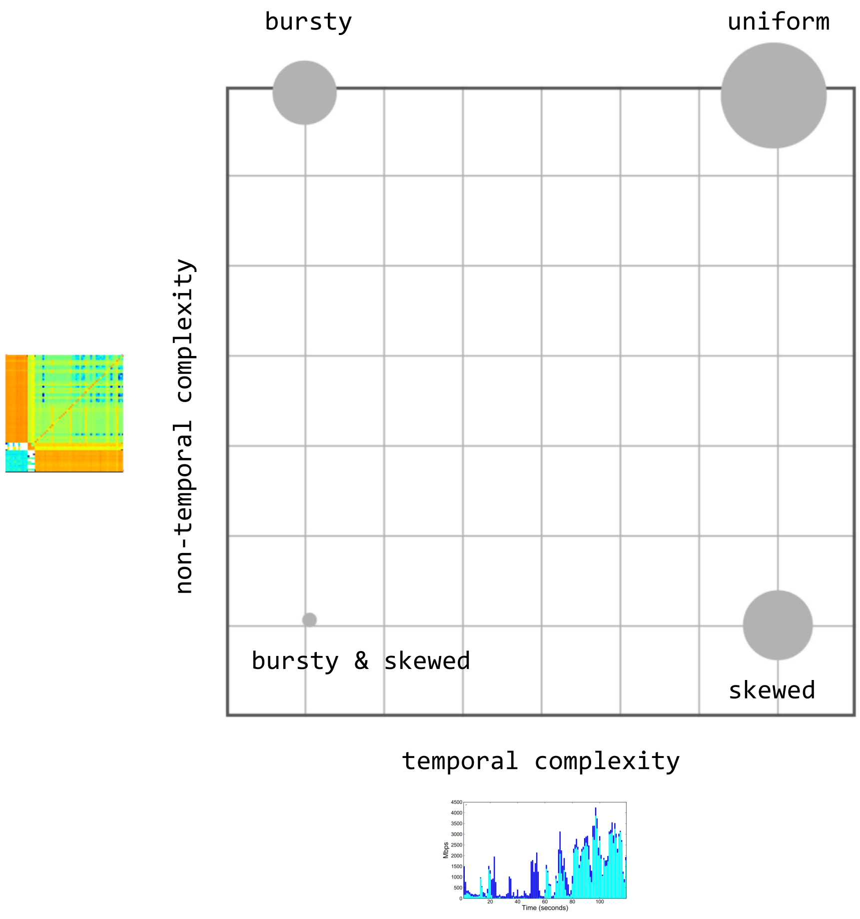
"Shuffle&Compress"



Can be used to define 2-dimensional complexity map!

Avin et al. (Sigmetrics’2020)

Complexity Map

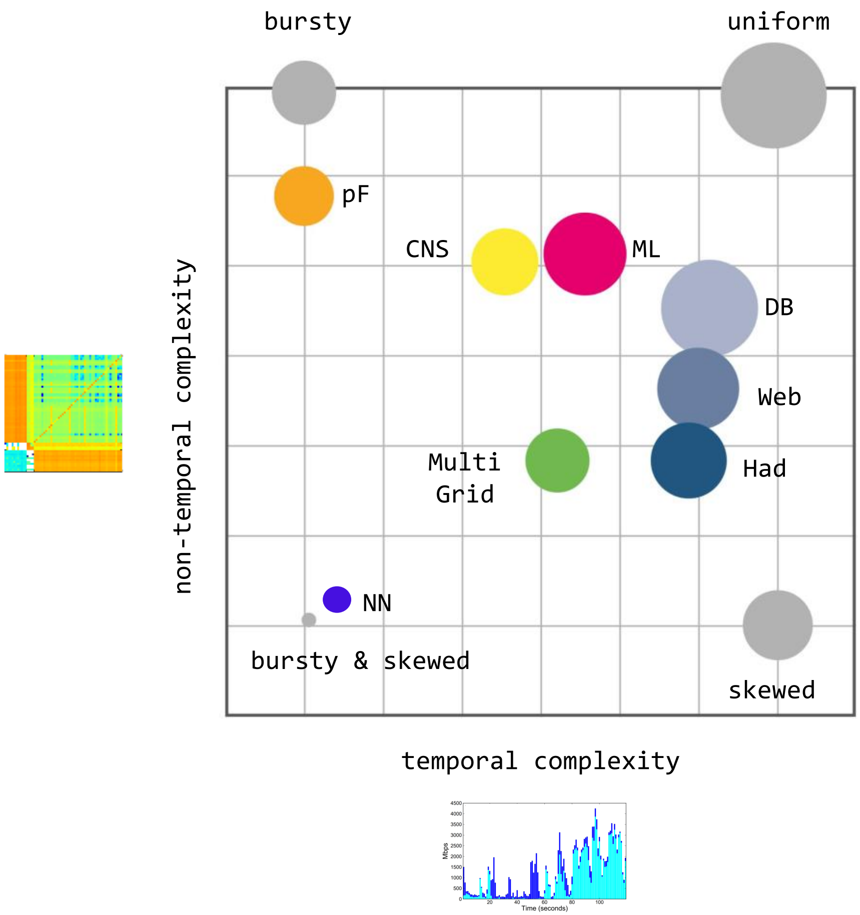


No structure

Our **approach:** iterative **randomization and compression** of trace to identify dimensions of structure.

Avin et al. (Sigmetrics'2020)

Complexity Map

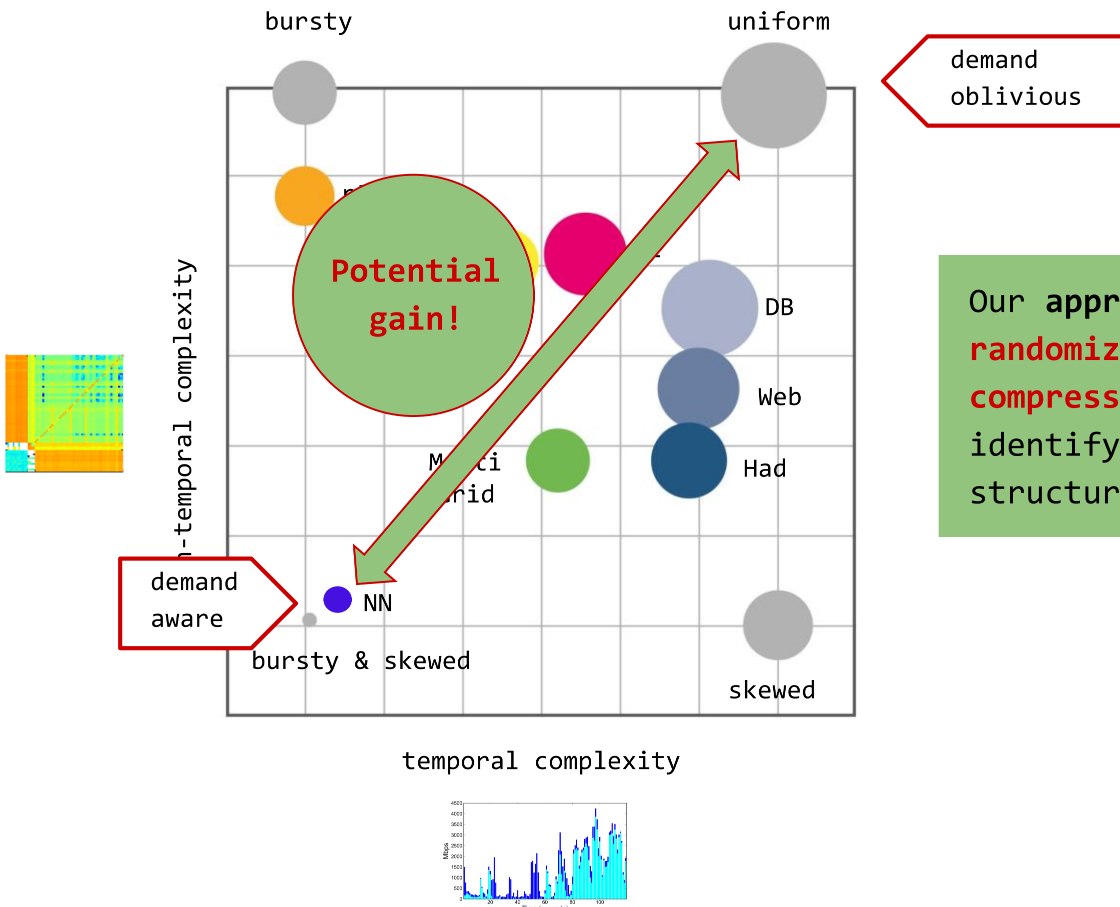


Our **approach**: iterative **randomization and compression** of trace to identify dimensions of structure.

Different structures!

Avin et al. (Sigmetrics’2020)

Complexity Map

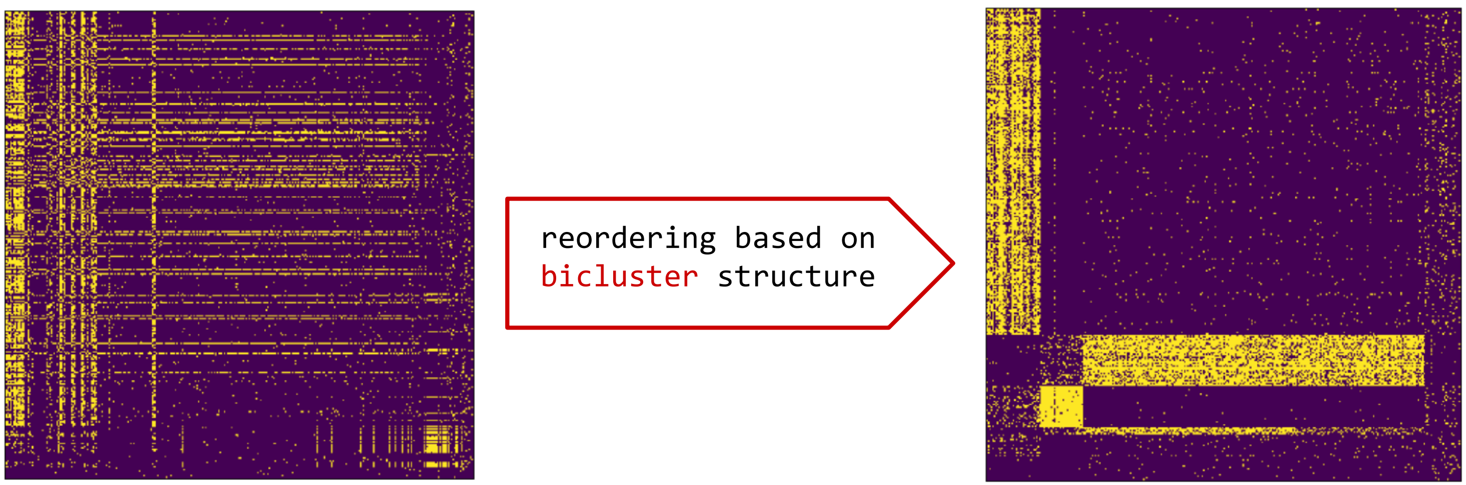


Our **approach**: iterative **randomization and compression** of trace to identify dimensions of structure.

Literature: On the Complexity of Traffic Traces and Implications. Avin et al., ACM SIGMETRICS, 2020.

Traffic is also clustered:

Small Stable Clusters



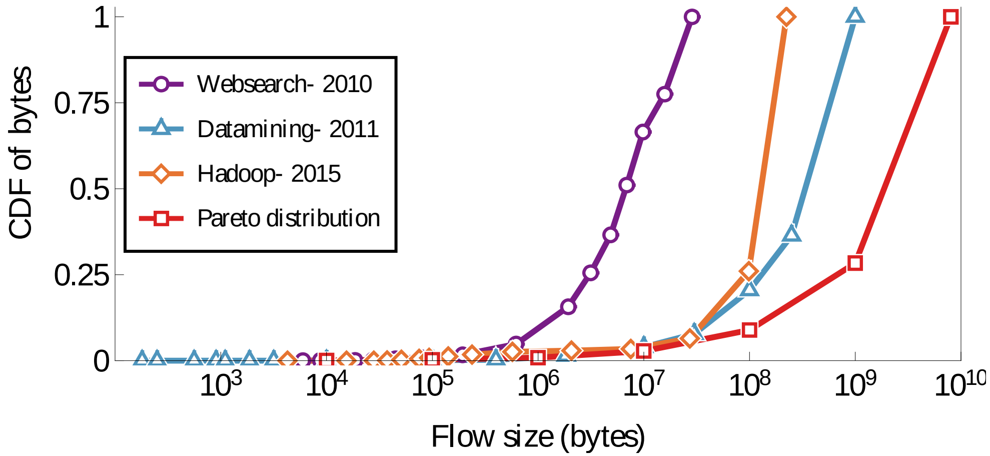
Opportunity: *exploit* with little reconfigurations!

Literature: Analyzing the Communication Clusters in Datacenters. Foerster et al. WWW Conference, 2023.

Even more structure:

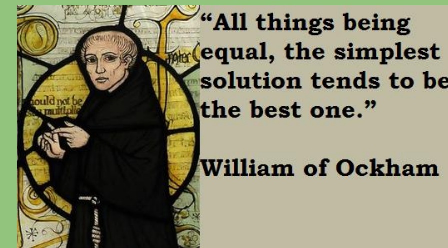
Flow Size Distribution

Flow transmission time (40Gbps)



- **Observation 1:** Different apps have different flow size distributions
- **Observation 2:** Most flows are small, most bytes in elephant flows

Synthesis for Researchers?



- We know properties but researchers have limited data currently.
- How to reproduce similar patterns synthetically? Can use Markov chains to „emulate" arbitrary points in complexity map!
- But what is "similar"? How different shall they be?
  - Similar = maps to same point in complexity map? Many more dimensions!
  - Is playing trace backward still similar?
  - How to generate similar traffic for larger networks?
- How to efficiently emulate application behavior? Use of „mini-apps" (no-op for compute)? Simulators like SimAI – efficient? Can we use LLMs?

Literature: On the Complexity of Traffic Traces and Implications. Avin et al., ACM SIGMETRICS, 2020.

How to exploit structure programmatically?

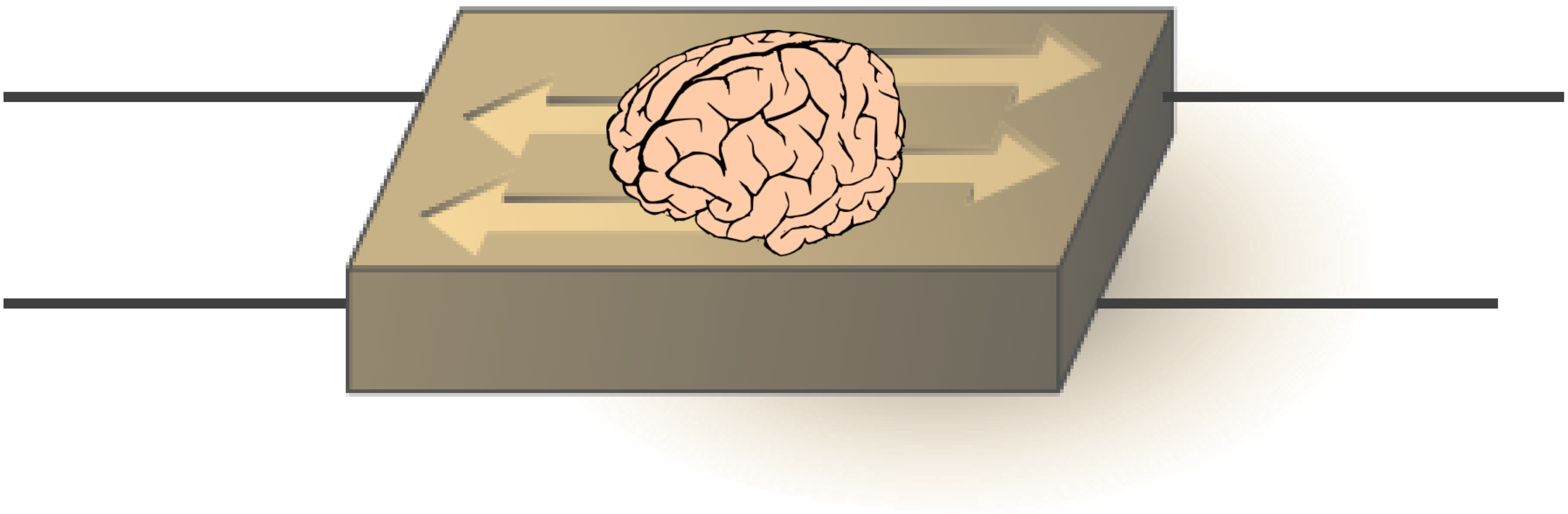
Example: Exploit Structure with

Smart Switches

Example: Exploit Structure with

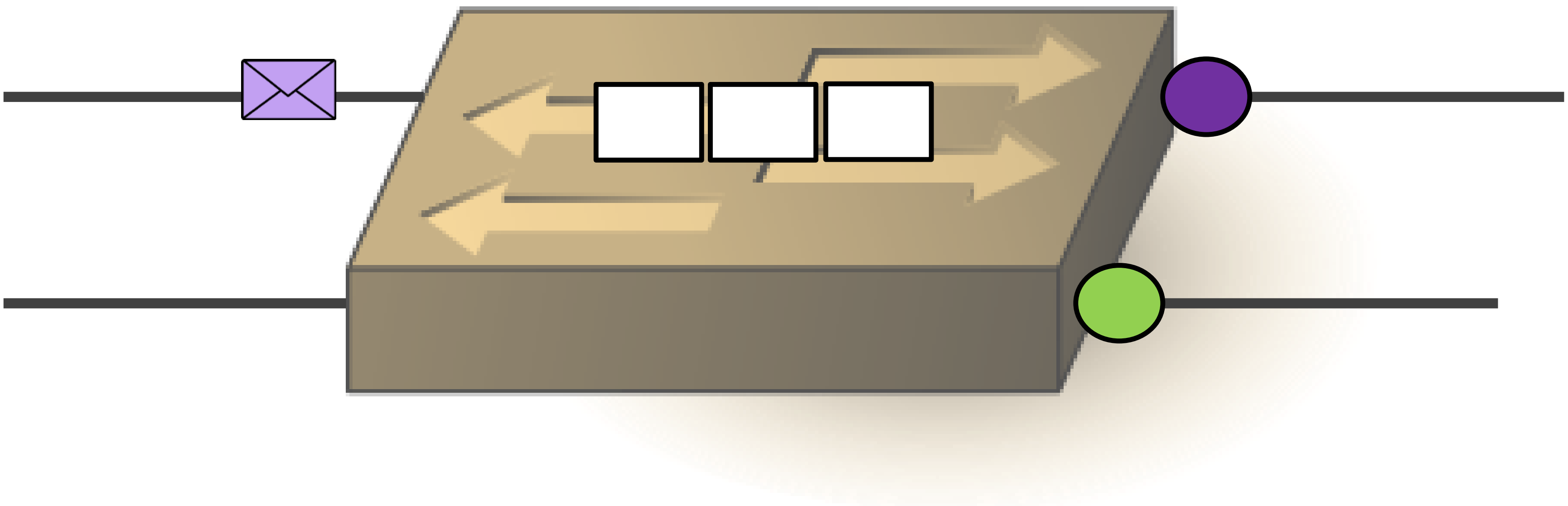
Smart Switches

- What if switches become smart?

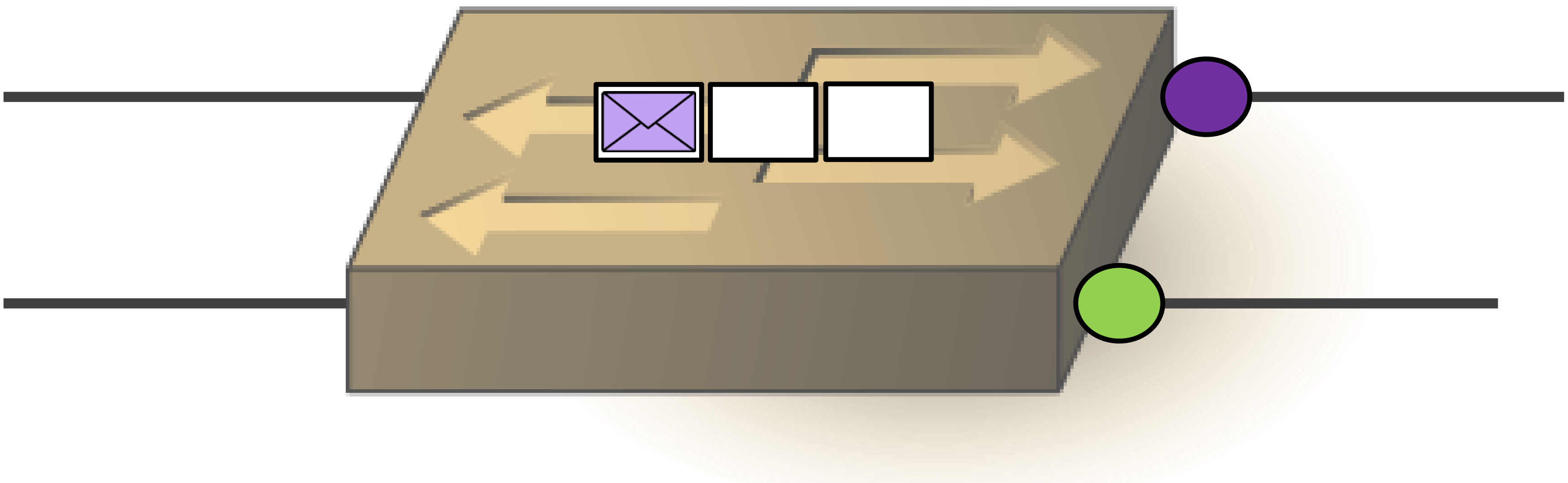


Scenario 1

Packet arrives
for violet port!



Scenario 1

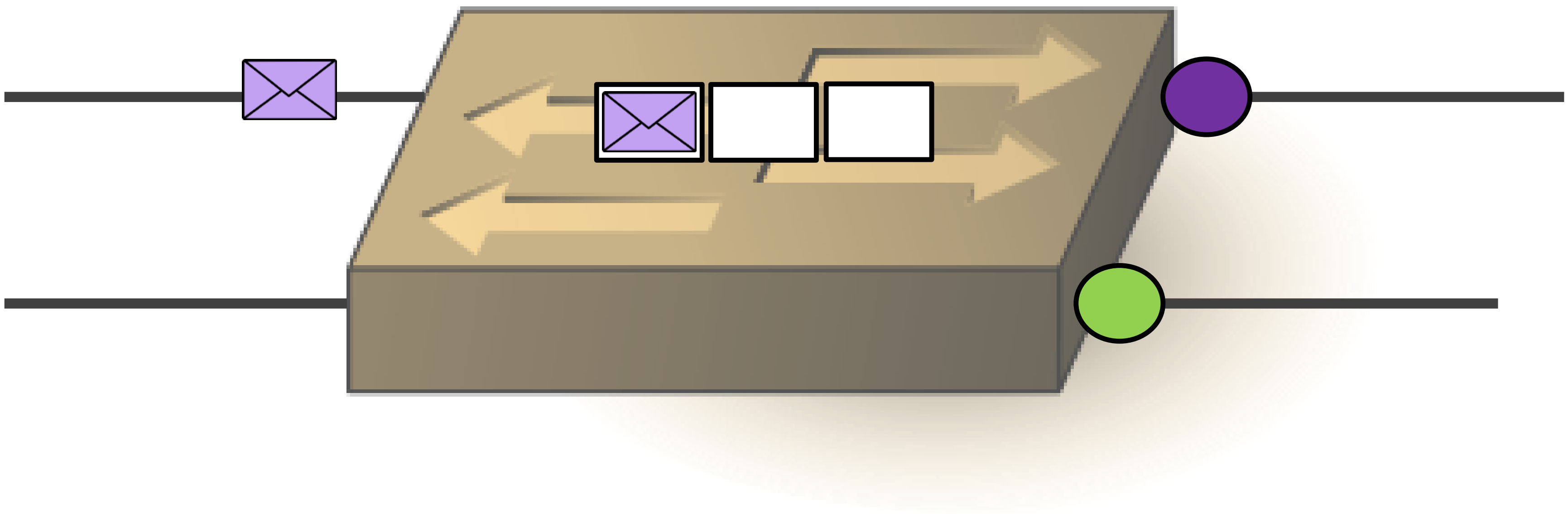


Admit to buffer!

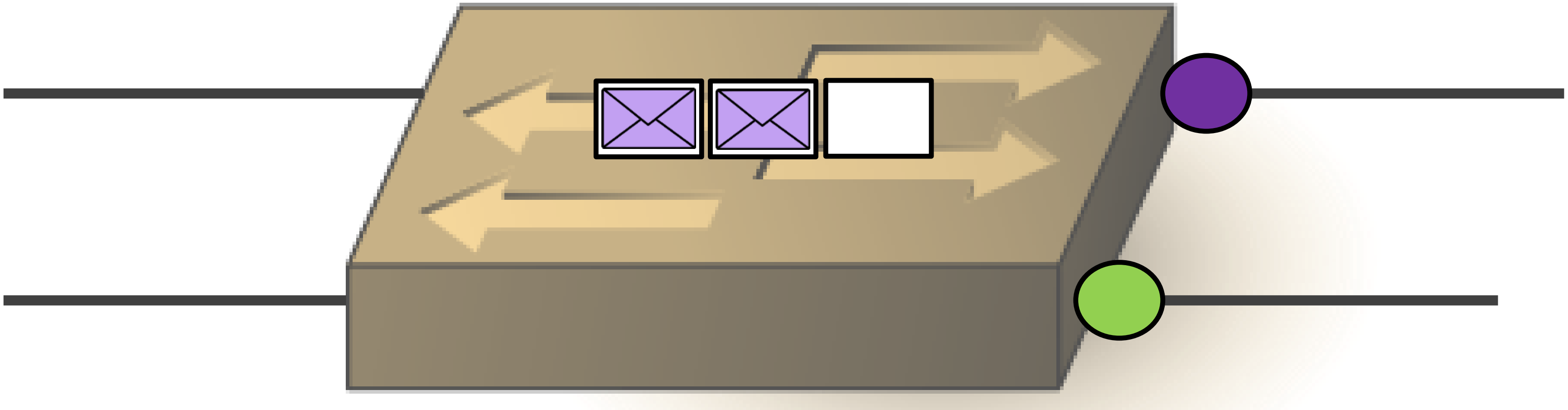
The Challenge: How to use shared memory?

Scenario 1

Packet arrives
for violet port!



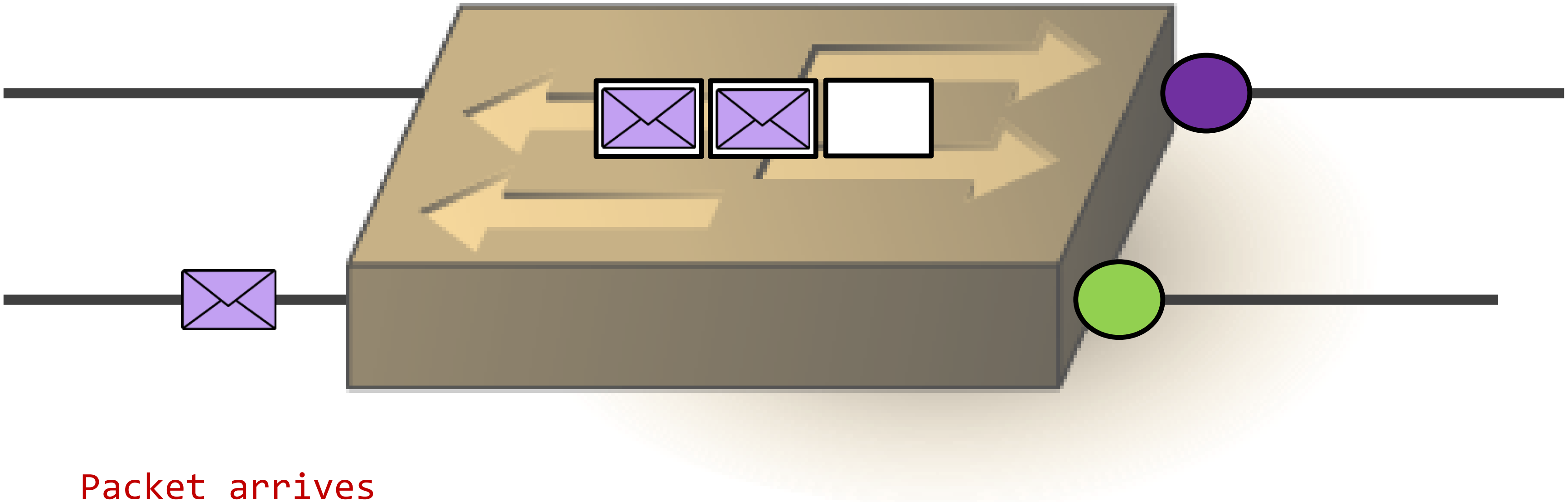
Scenario 1



Admit to buffer!

The Challenge: How to use shared memory?

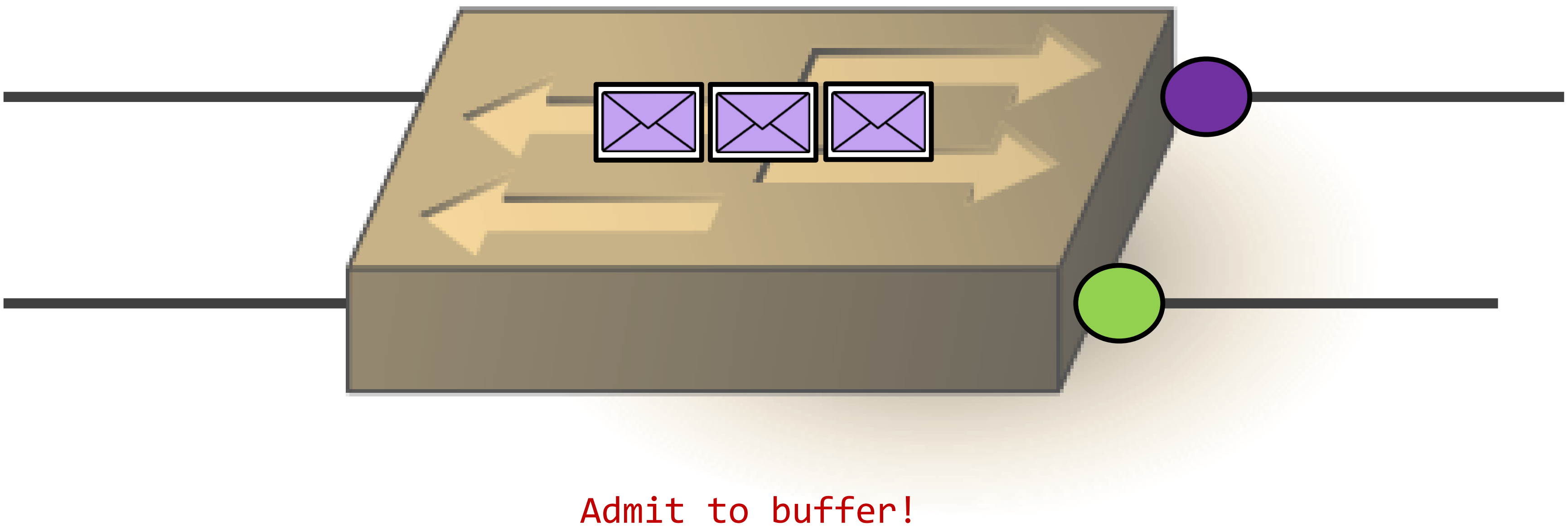
Scenario 1



Packet arrives
for violet port!

The Challenge: How to use shared memory?

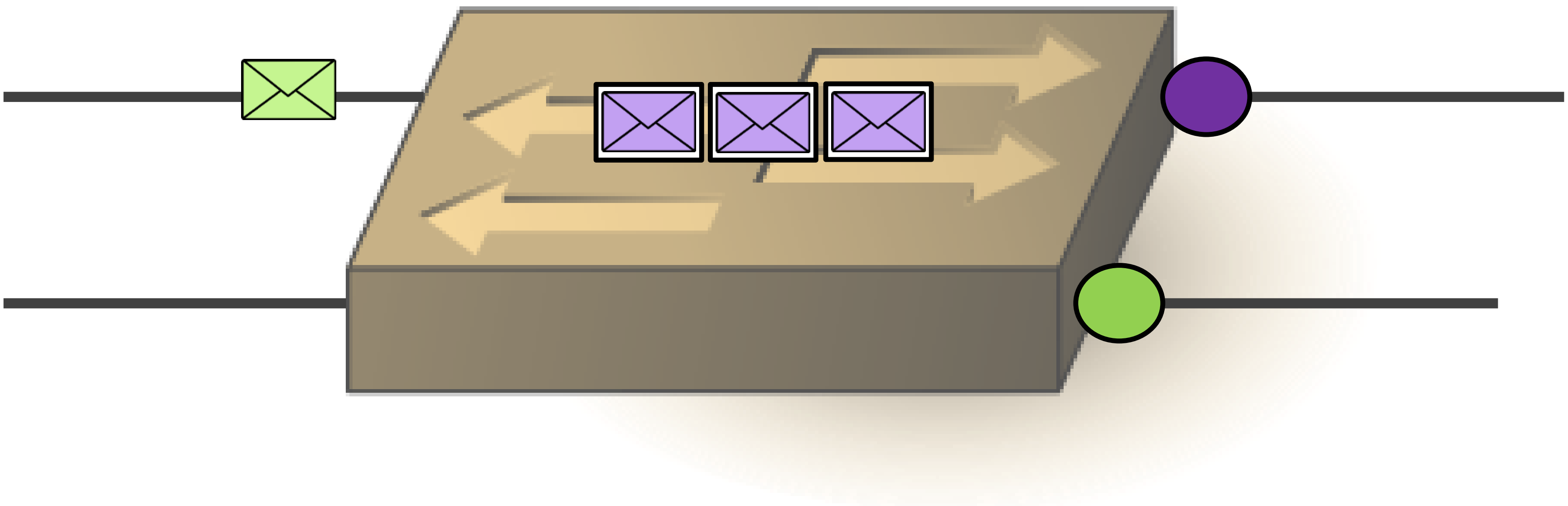
Scenario 1



Admit to buffer!

Scenario 1

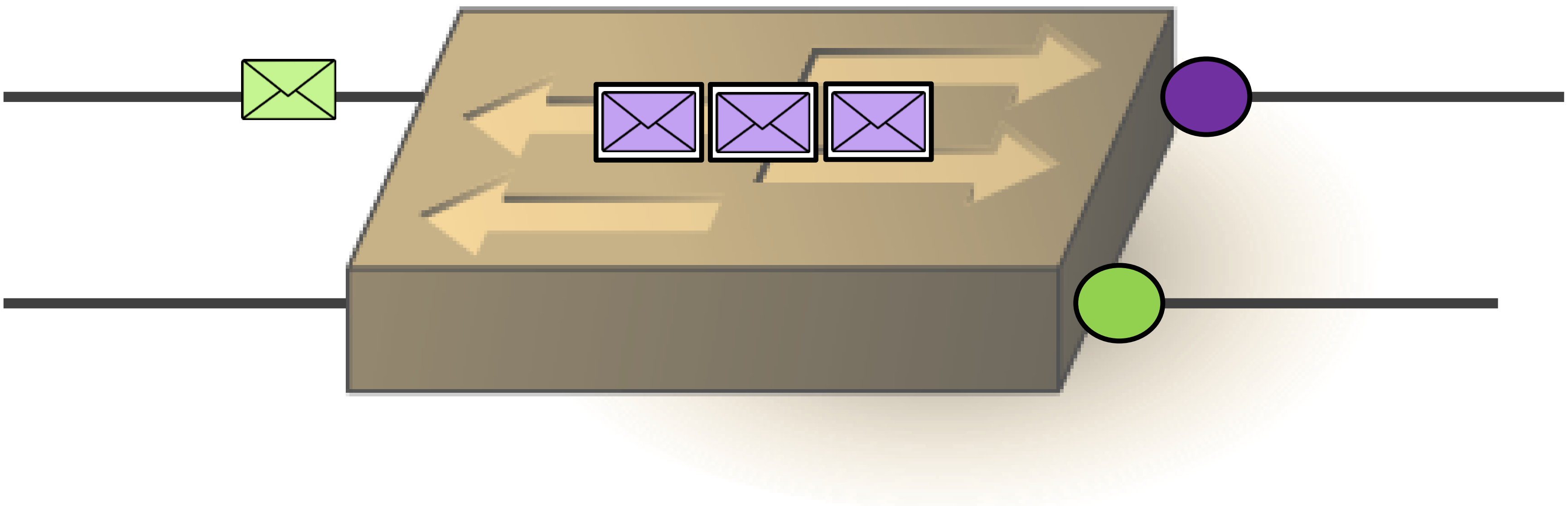
Packet arrives
for green port!



The Challenge: How to use shared memory?

Scenario 1

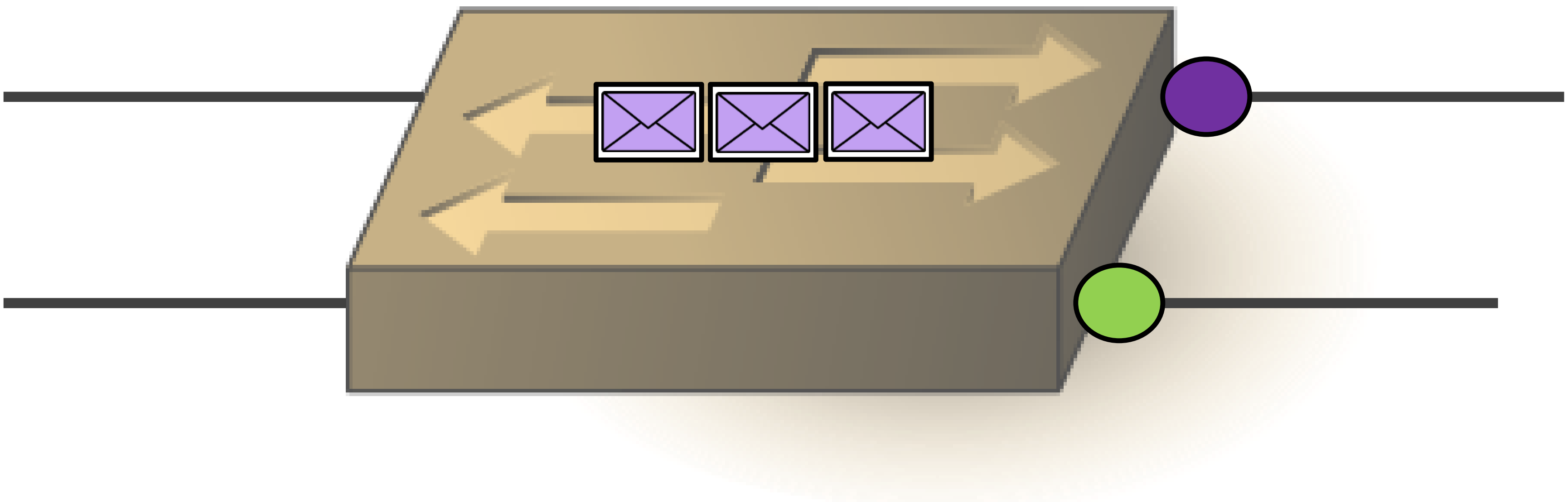
Need to drop: no
more buffer space!



The Challenge: How to use shared memory?

Scenario 1

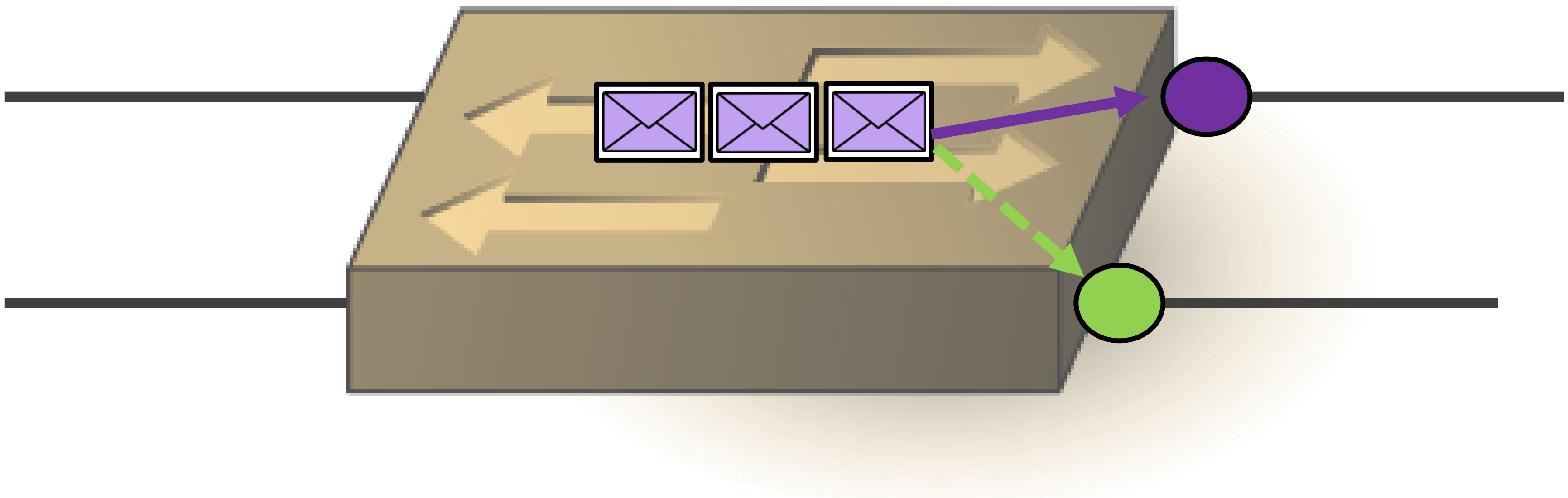
- The problem: missed opportunity for higher throughput
- With green packet can transmit packets in parallel on 2 ports



The Challenge: How to use shared memory?

Scenario 1

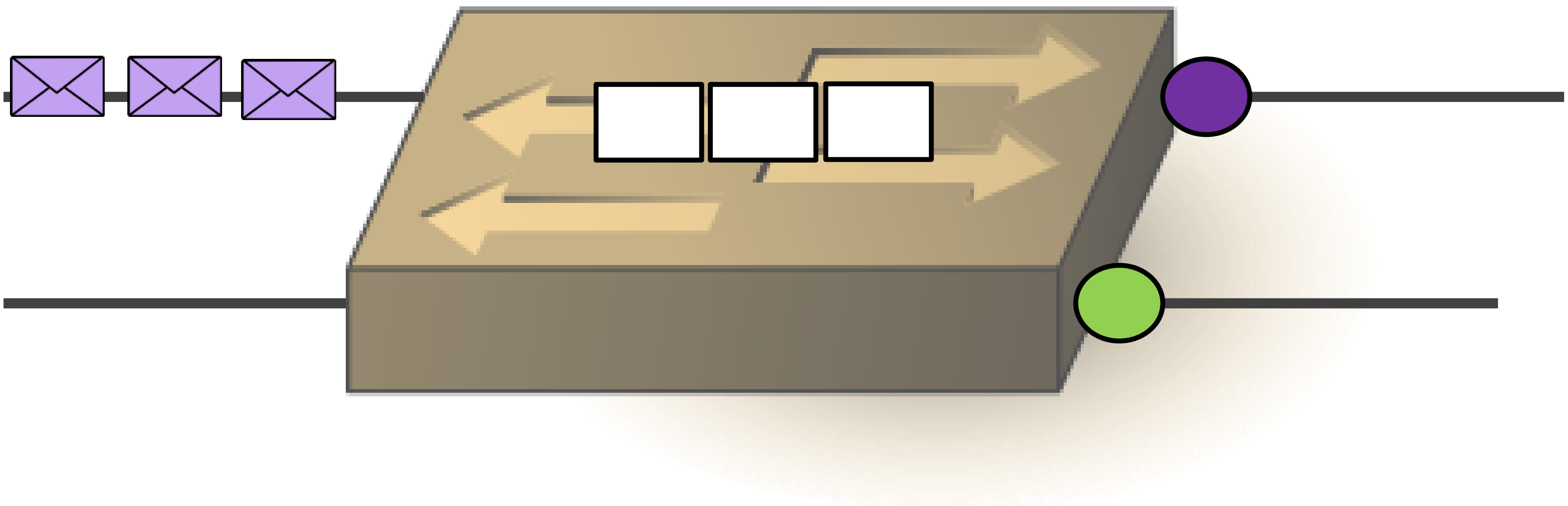
- The problem: missed opportunity for higher throughput
- With green packet can transmit packets in parallel on 2 ports



The Challenge: How to use shared memory?

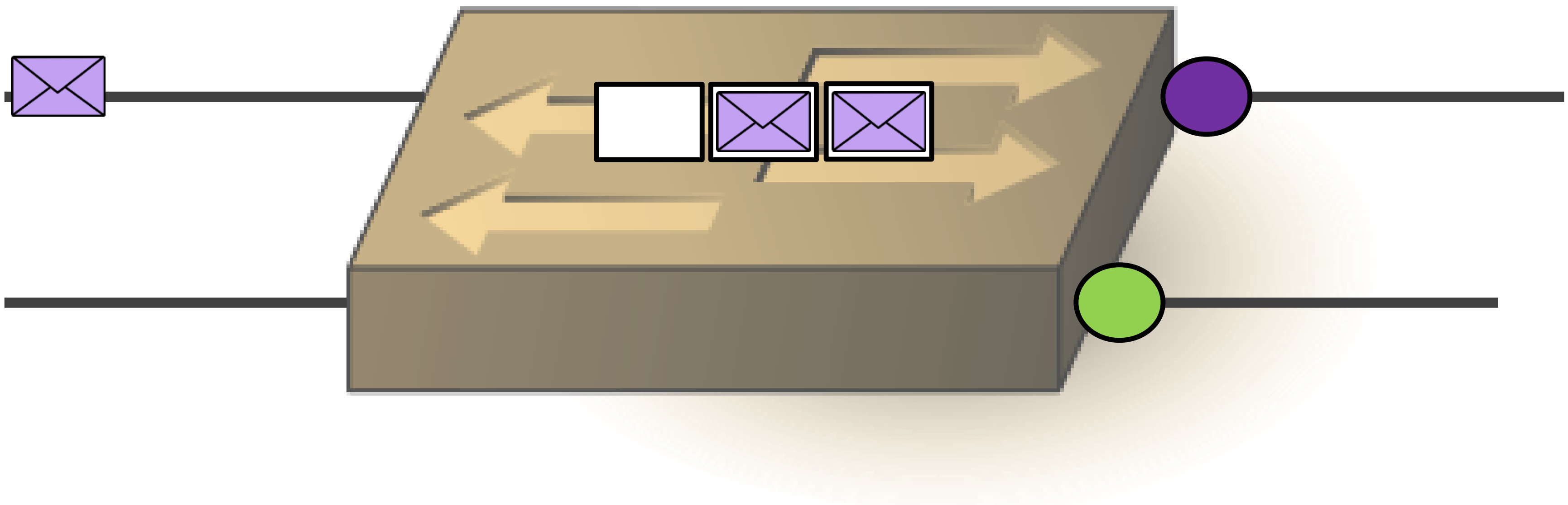
Scenario 2

3 packets arrive
for violet port!



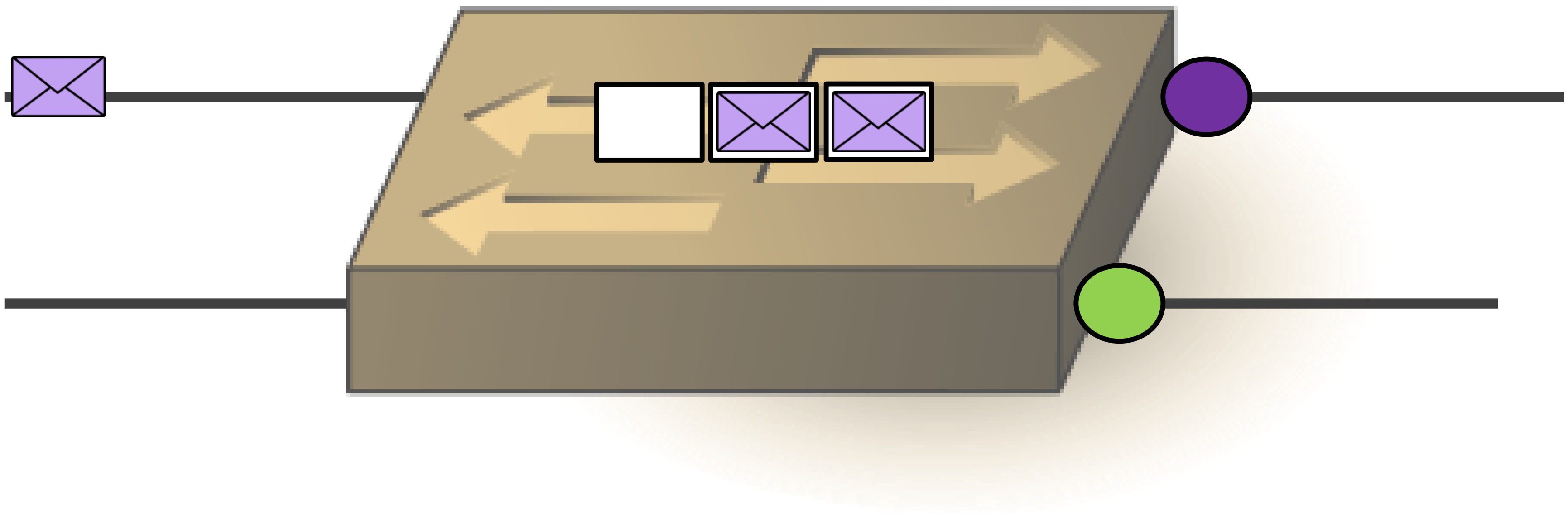
Scenario 2

Accept two of them! But safe one slot for green: potential for more throughput!



Scenario 2

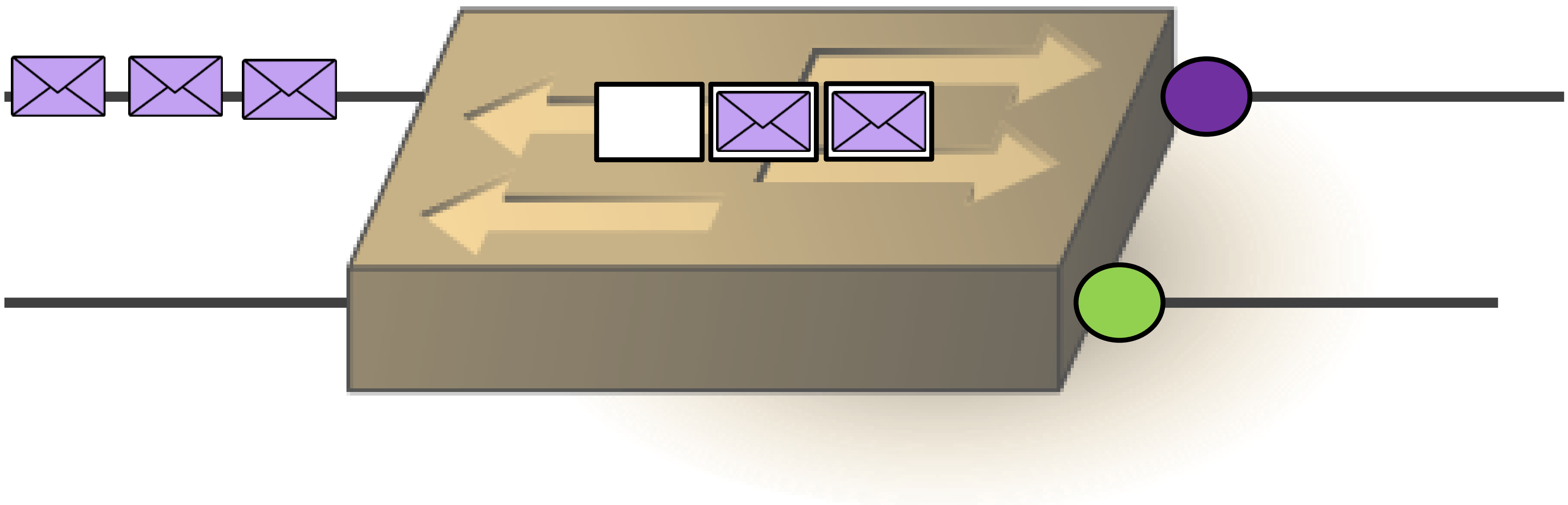
Accept two of them! But safe one slot for green: potential for more throughput!



The Challenge: How to use shared memory?

Scenario 2

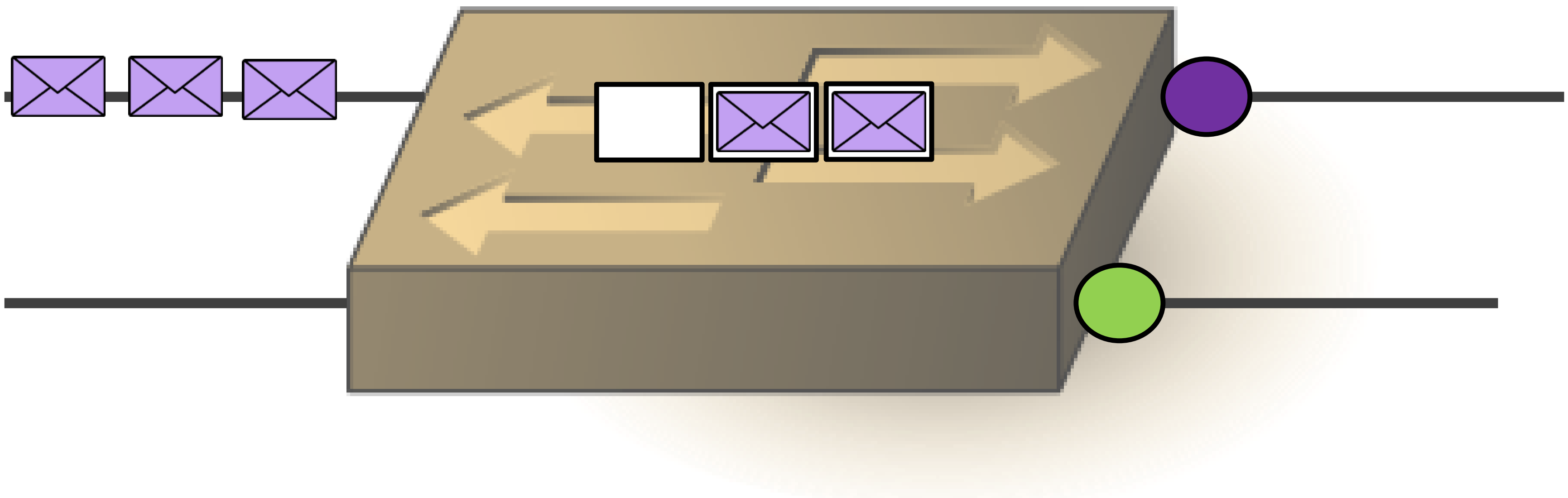
- The problem: what if many more violet packets arrive?
- Missed opportunity to use buffer!



The Challenge: How to use shared memory?

Scenario 2

- The problem: what if many more violet packets arrive?
- Missed opportunity to use buffer!



- Realm of online algorithms and competitive analysis: algorithms which perform well without knowing the future!

The Opportunity

Smart Buffer Management

- Idea: as traffic is often fairly predictable and has structure…
- … can we employ predictions for smarter buffer management?
- E.g., using random forests: feasible on programmable switches at line rate.

Smart Buffer Management

- Idea: as traffic is often fairly predictable and has structure…
- … can we employ predictions for smarter buffer management?
- E.g., using random forests: feasible on programmable switches at line rate.

How to evaluate online algorithms: algorithms which do not know the future?

Metrics

for Online Algorithms with Predictions

Classic goal of line algorithms:

⇢ *Perform (almost) like offline* algorithm

⇢ Minimize *competitive ratio*: CostON/CostOFF

Metrics

for Online Algorithms with Predictions

Classic goal of line algorithms:

⇢ *Perform (almost) like offline* algorithm

⇢ Minimize *competitive ratio*: CostON/CostOFF

With prediction:

⇢ If prediction is *true*: perform better than ON (*consistency*)

⇢ If prediction is *wrong*: don’t perform much worse (*robustness*)

Metrics

for Online Algorithms with Predictions

Classic goal of line algorithms:

⇢ *Perform (almost) like offline* algorithm

⇢ Minimize *competitive ratio*: CostON/CostOFF

With prediction:

⇢ If prediction is *true*: perform better than ON (*consistency*)

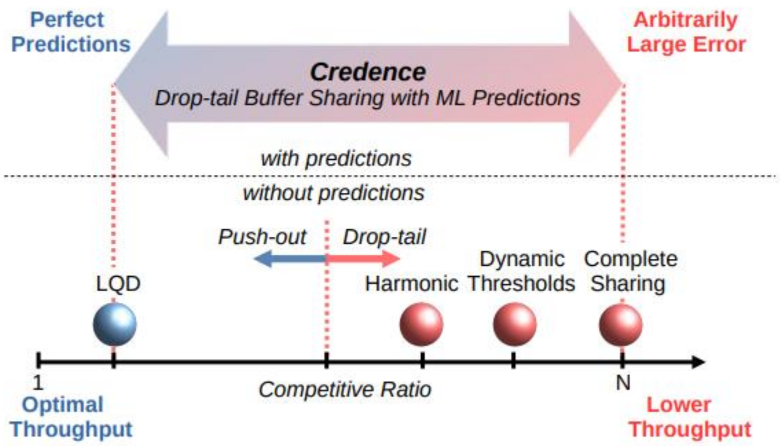
⇢ If prediction is *wrong*: don’t perform much worse (*robustness*)

Hot topic (so far) in theory

A first approach: Addanki et al. (NSDI 2024)

Credence

⇢ Predictions are powerful: allow simple drop-tail algorithm to perform as well as push-out algorithms



[Credence: Augmenting Datacenter Switch Buffer Sharing with ML Predictions]
Vamsi Addanki, Maciej Pacut, and Stefan Schmid.
21st USENIX Symposium on Networked Systems Design and Implementation (**NSDI**), 2024.

How to support such more dynamic networks?

Research Challenge

Stack for Dynamic Networks

- When some parts of networks become more dynamic, other layers may have to adapt too.
- Example: dynamic topology programming may challenge buffer management, routing performance and congestion control
- General ideas:
  - More local network control? Greedy routing can deal with dynamic topologies.
  - Make better use of visibility into the network: telemetry, INT
  - Lessons from other dynamic networks? P2P? Ad-hoc networks?

Case Study:

Congestion Control (CC)

Existing congestion control algorithms based on either

- State (“voltage”) like BDP, queue length, loss, e.g.:
  - DCTCP: uses ECN/loss
  - Swift: RTT
  - HPCC: inflight packets
- Gradient (“current”) like reaction to queue length change
  - Timely: RTT-gradient based

Case Study:

Congestion Control (CC)

Existing congestion control algorithms based on either

- State ("voltage") like BDP, queue length, loss, e.g.:
  - DCTCP: uses ECN/loss
  - Swift: RTT
  - HPCC: inflight packets

☺ Can achieve near-zero queue equilibrium

☹ Slow reaction

- Gradient ("current") like reaction to queue length change
  - Timely: RTT-gradient based

Case Study:

Congestion Control (CC)

Existing congestion control algorithms based on either

- State ("voltage") like BDP, queue length, loss, e.g.:
  - DCTCP: uses ECN/loss
  - Swift: RTT
  - HPCC: inflight packets
- Gradient ("current") like reaction to queue length change
  - Timely: RTT-gradient based

☺ Fast reaction

☹ No equilibrium

Case Study:

Congestion Control (CC)

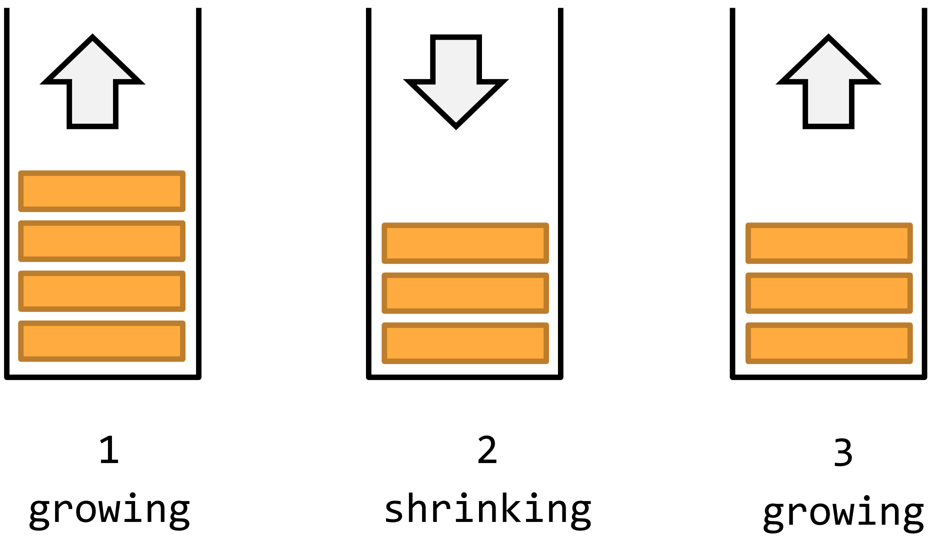
Existing congestion control algorithms based on either

- State ("voltage") like BDP, queue length, loss, e.g.:
  - DCTCP: uses ECN/loss
  - Swift: RTT
  - HPCC: inflight packets
- Gradient ("current") like reaction to queue length change
  - Timely: RTT-gradient based

Limitation: using only one of the two may miss useful information for fine-grained adaptions!

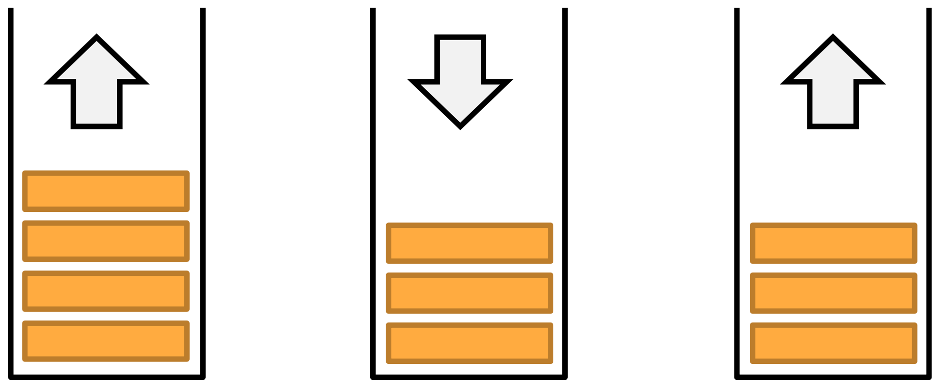
Limitation of SOTA

⇢ Consider a queue which may be in three different states:



Limitation of SOTA

⇢ Consider a queue which may be in three different states:



1
growing

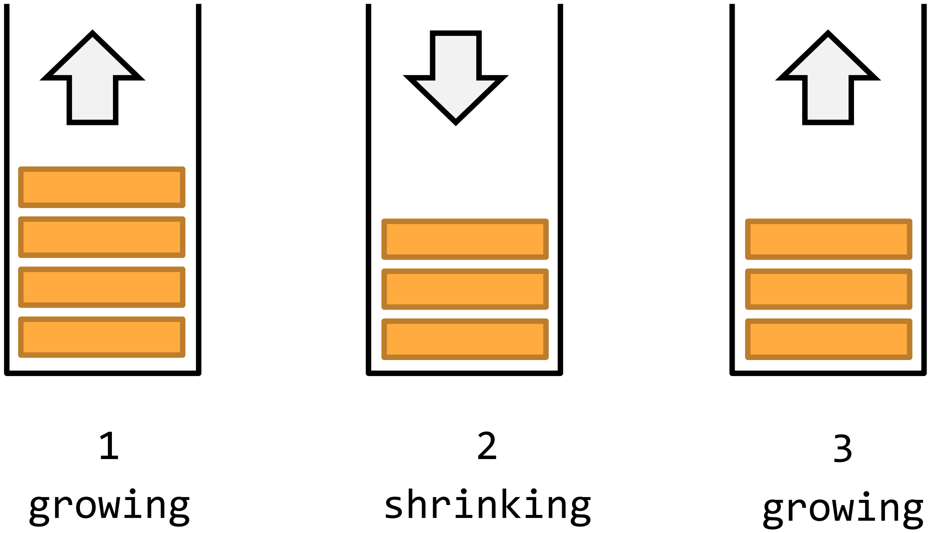
2
shrinking

3
growing

2 and 3: impossible to distinguish for voltage-based CCA

Limitation of SOTA

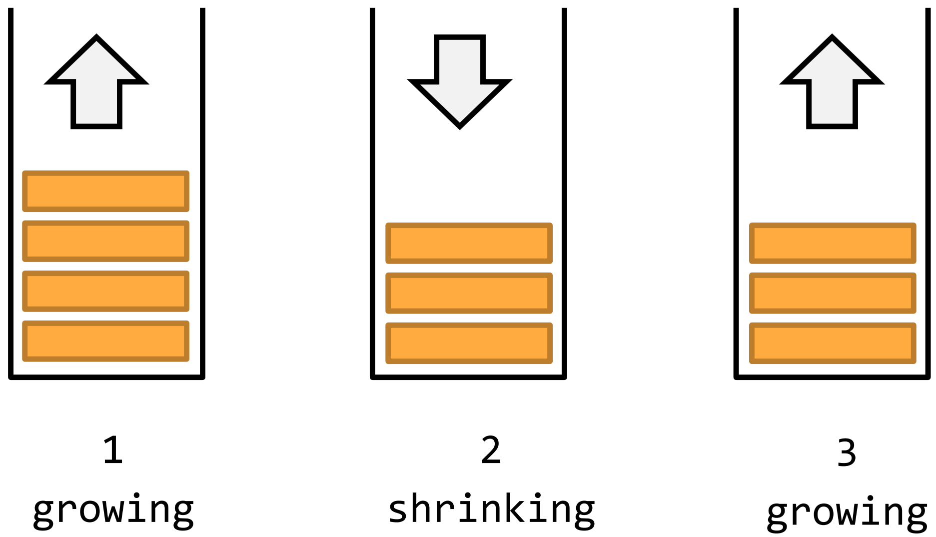
⇢ Consider a queue which may be in three different states:



1 and 3: impossible to distinguish for current-based CC

Limitation of SOTA

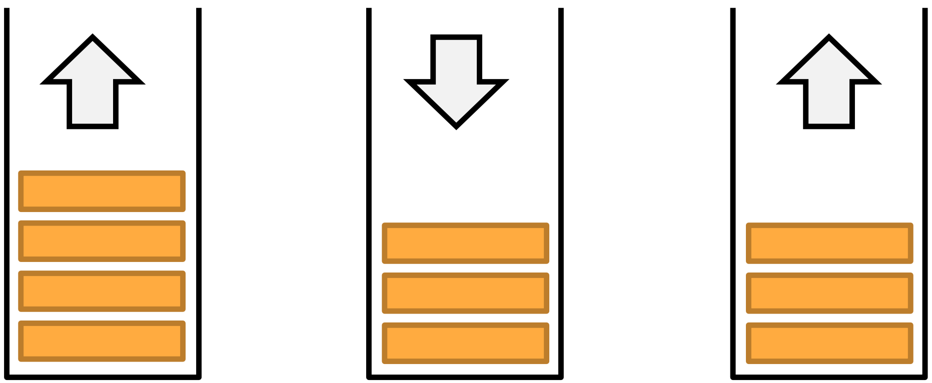
⇢ Consider a queue which may be in three different states:



We need both: Power (Voltage x Current)

Limitation of SOTA

⇢ Consider a queue which may be in three different states:



1 growing | 2 shrinking | 3 growing

We need both: Power (Voltage x Current)

Inspired: POWERTCP

Improving Performance Further with

Telemetry Powered CC

- Telemetry provides opportunities to further improve CC, but so far limited to switches
- Would be nice to enable telemetry-based congestion control in the kernel without changing end-host
- First proofs-of-concepts* show that using eBPF we can run CC algorithms that execute different control laws
- Promising: TCP incast workloads experience less queuing, faster convergence and better fairness

* TCP's Third Eye: Leveraging eBPF for Telemetry-Powered Congestion Control. Jörn-Thorben Hinz, Vamsi Addanki, Csaba Györgyi, Theo Jepsen, and Stefan Schmid. SIGCOMM Workshop on eBPF and Kernel Extensions (eBPF), 2023.

Looking Forward

- It would be nice to see further telemetry-based protocols at end-hosts
  - e.g. for routing storage traffic, path load balancing, flow scheduling
- With future support for offloading eBPF to hardware they could even run directly in the NIC
- Would be nice: standardize use of INT at lower-level protocols—like IP header options. Feature support from the eBPF community?

Roadmap

Two tales:

- Traffic: structure in traffic = optimization opportunity for NetSoft researchers
- Dependability: Flexibility may introduce complexity, a case for ML and formal methods?

Roadmap

Two tales:

- ⇢ Traffic: structure in traffic = optimization opportunity for NetSoft researchers
- ⇢ **Dependability: Flexibility may introduce complexity, a case for ML and formal methods?**

Networks:

Critical Infrastructure

- If networks break, it can have knock-on effects
- For example, Facebook outage in 2021: not only took down their social networking site, but also Instagram, WhatsApp, …
- … and their own internal systems, which manage the doors: engineers had to break into their own buildings to bring the network back up

The New York Times

Gone in Minutes, Out for Hours: Outage Shakes Facebook

When apps used by billions of people worldwide blinked out, lives were disrupted, businesses were cut off from customers — and some Facebook employees were locked out of their offices.

Share full article 884

Facebook's internal communications platform, Workplace, was also taken out, leaving most employees unable to do their jobs. Kelsey McClellan for The New York Times

Credits: Nate Foster

Human Errors

Countries disconnected

Data Centre ▸ Networks

Google routing blunder sent Japan's Internet dark on Friday

Another big BGP blunder

By Richard Chirgwin 27 Aug 2017 at 22:35 40 SHARE ▼

Last Friday, someone in Google fat-thumbed a border gateway protocol (BGP) advertisement and sent Japanese Internet traffic into a black hole.

The trouble began when The Chocolate Factory "leaked" a big route table to Verizon, the result of which was traffic from Japanese giants like NTT and KDDI was sent to Google on the expectation it would be treated as transit.

Passengers stranded

British Airways' latest Total Inability To Support Upwardness of Planes* caused by Amadeus system outage

Stuck on the ground awaiting a load sheet? Here's why

By Gareth Corfield 19 Jul 2018 at 11:16 109 SHARE ▼

Even 911 affected

Officials: Human error to blame in Minn. 911 outage

According to a press release, CenturyLink told department of public safety that human error by an employee of a third party vendor was to blame for the outage

Aug 16, 2018

Duluth News Tribune

SAINT PAUL, Minn. — The Minnesota Department of Public Safety Emergency Communication Networks division was told by its 911 provider that an Aug. 1 outage was caused by human error.

Even tech-savvy companies struggle:

GoDaddy.com github SOCIAL CODING United Airlines amazon web services

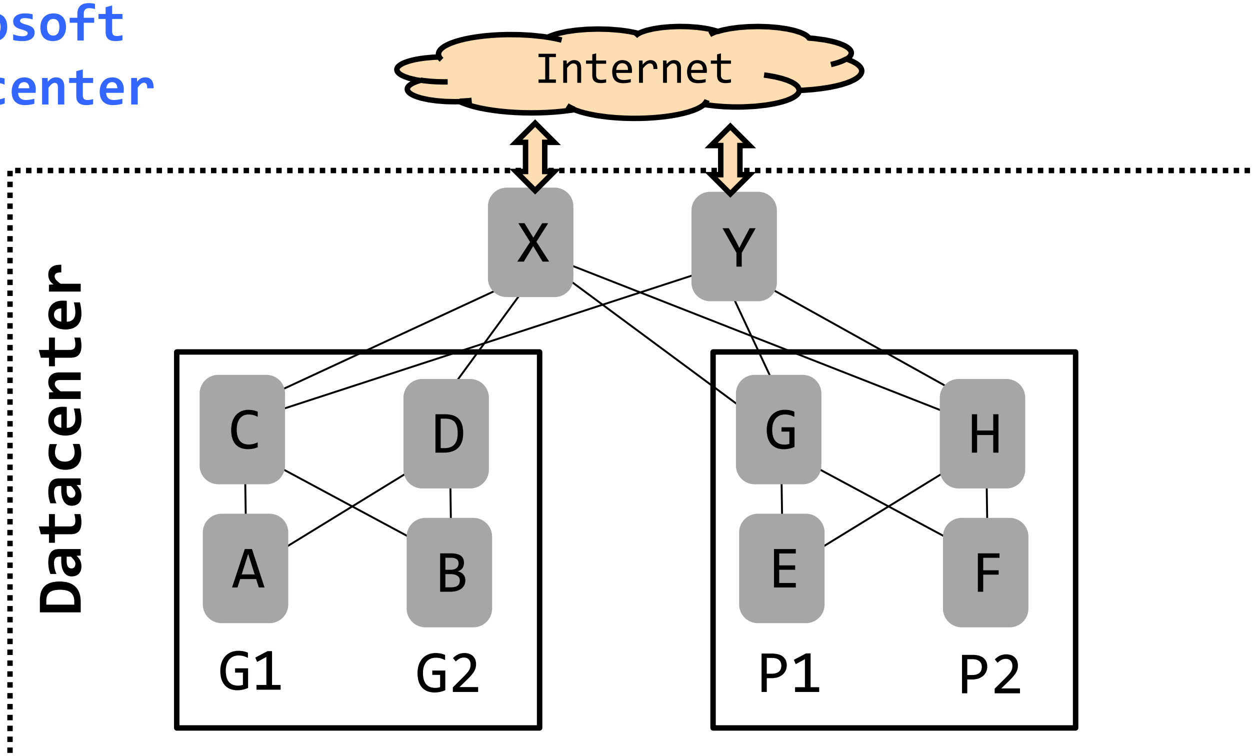
Mainly: human errors!

Slide credits: Nate Foster and Laurent Vanbever

A Reason: Complexity

Especially Under Failures (Policy Compliance)

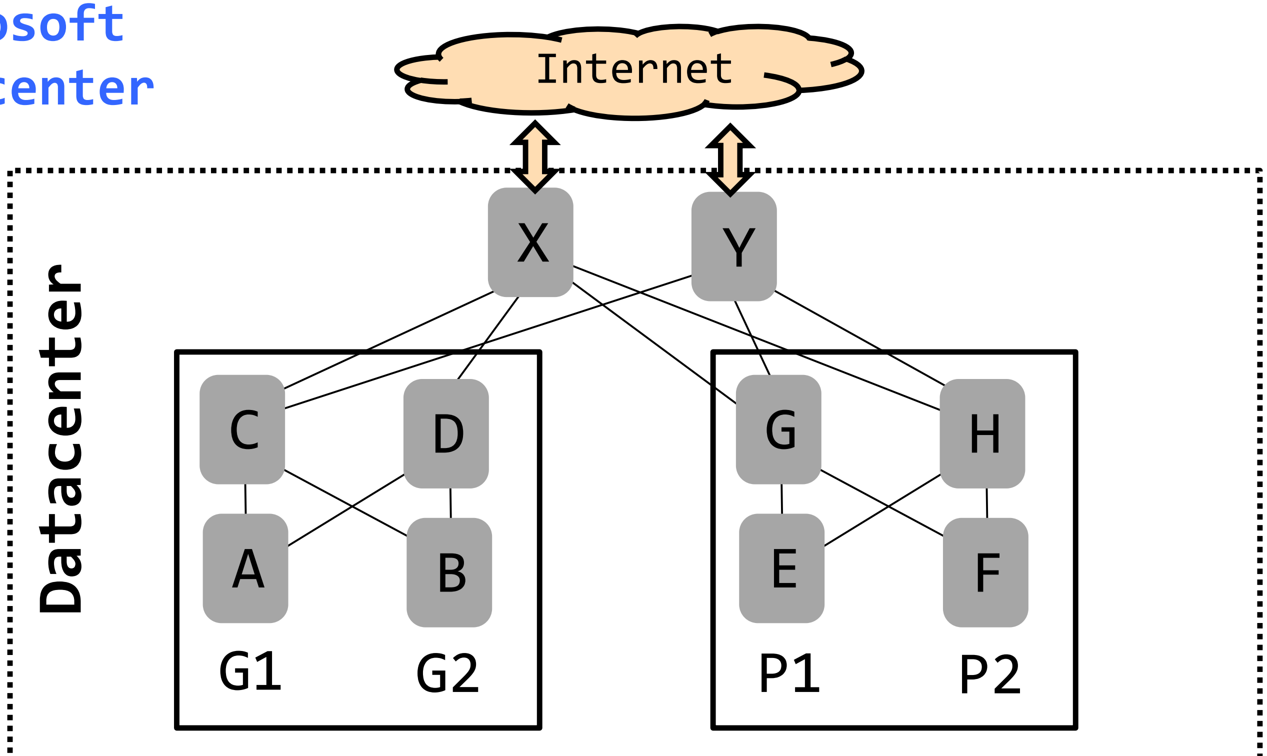
Example: BGP in **Microsoft datacenter**



A Reason: Complexity

Especially Under Failures (Policy Compliance)

Example: BGP in **Microsoft datacenter**



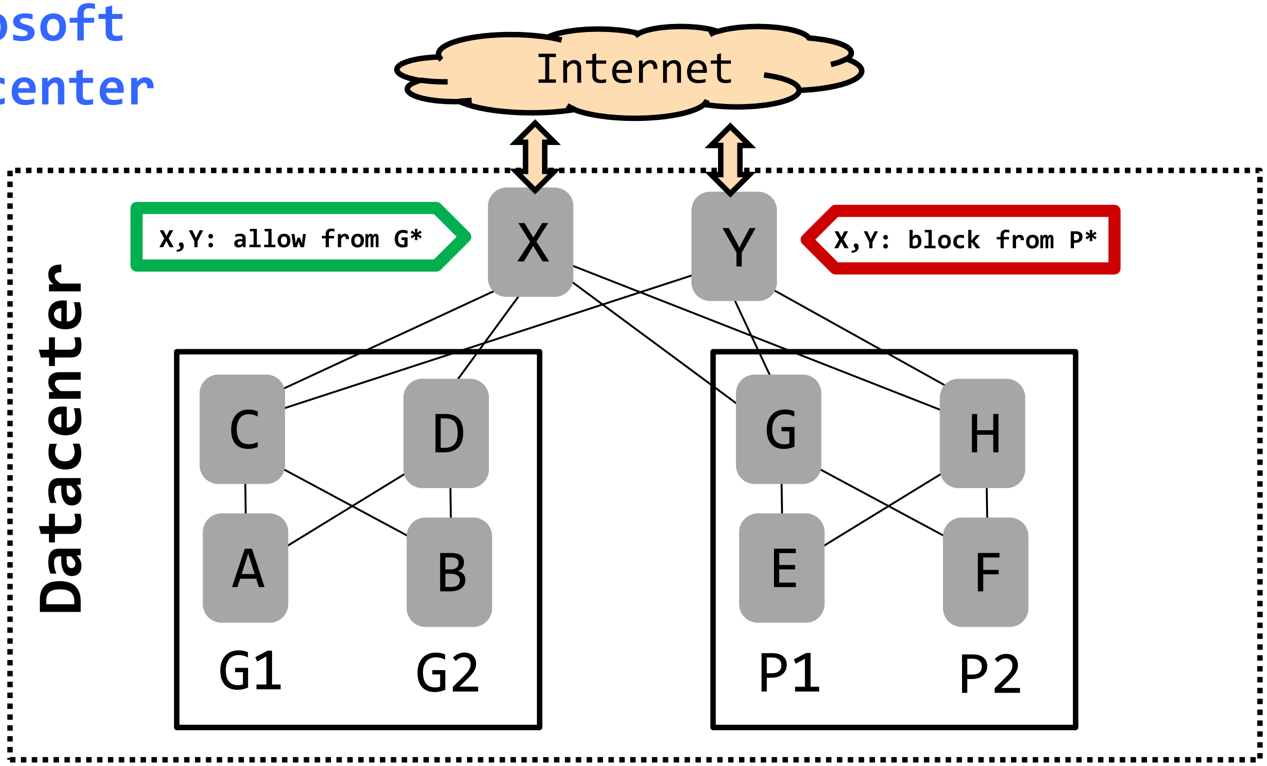
Cluster with globally reachable services

Cluster with internally accessible services

A Reason: Complexity

Especially Under Failures (Policy Compliance)

Example: BGP in **Microsoft datacenter**



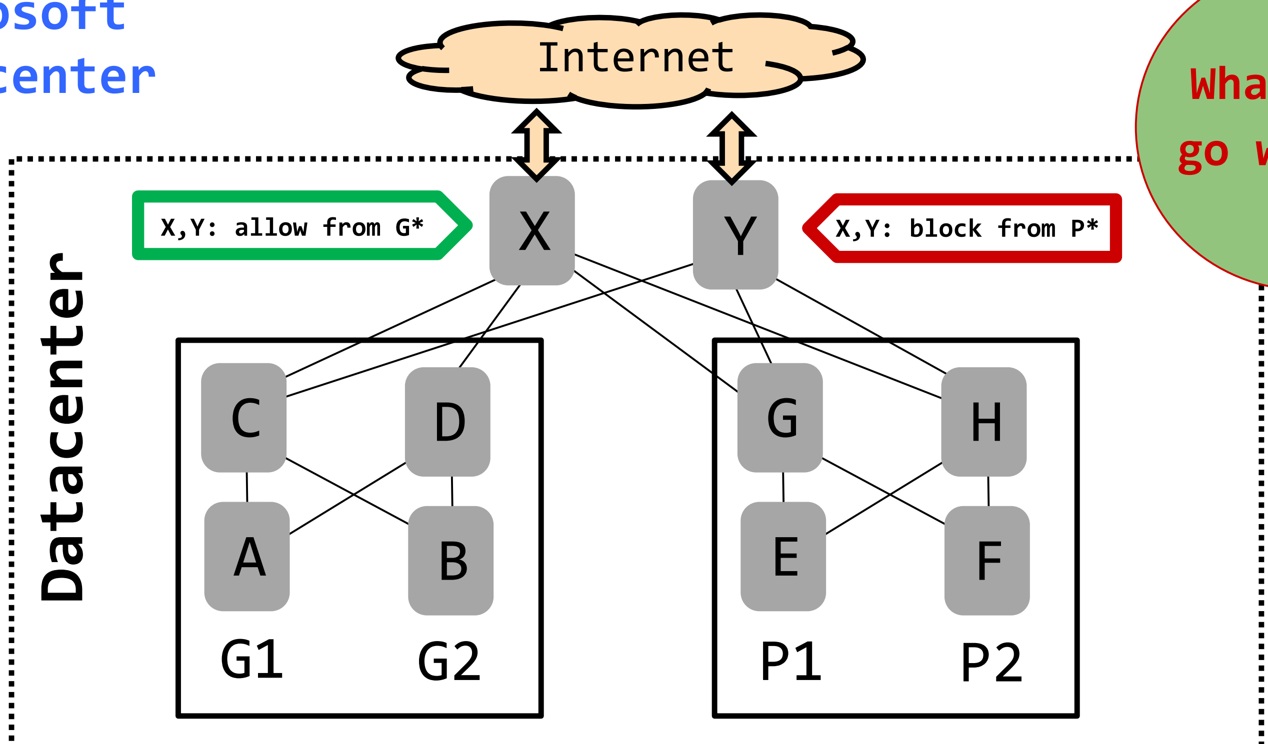
Cluster with globally reachable services

Cluster with internally accessible services

A Reason: Complexity

Especially Under Failures (Policy Compliance)

Example: BGP in **Microsoft datacenter**



What can go wrong?

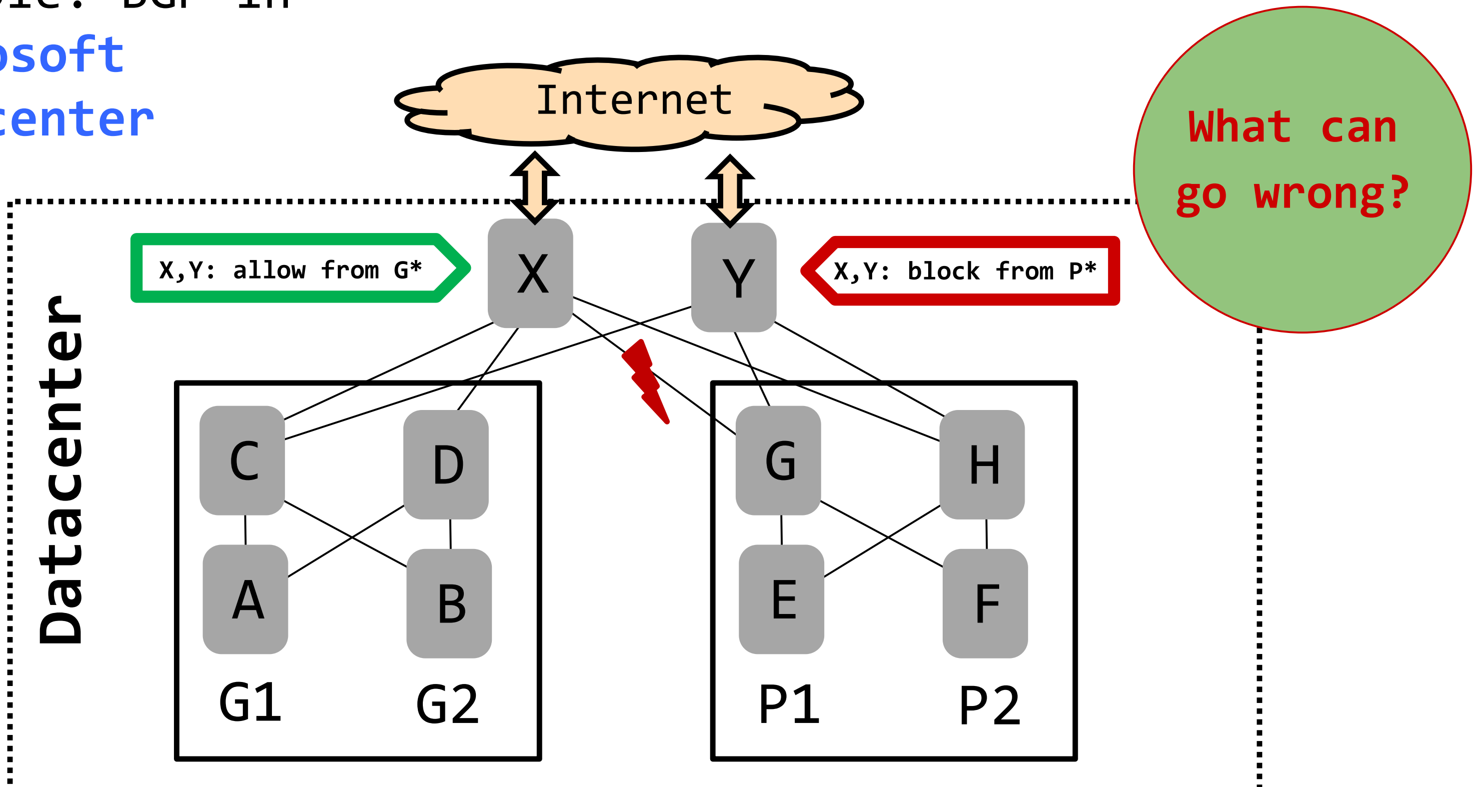
Cluster with globally reachable services

Cluster with internally accessible services

A Reason: Complexity

Especially Under Failures (Policy Compliance)

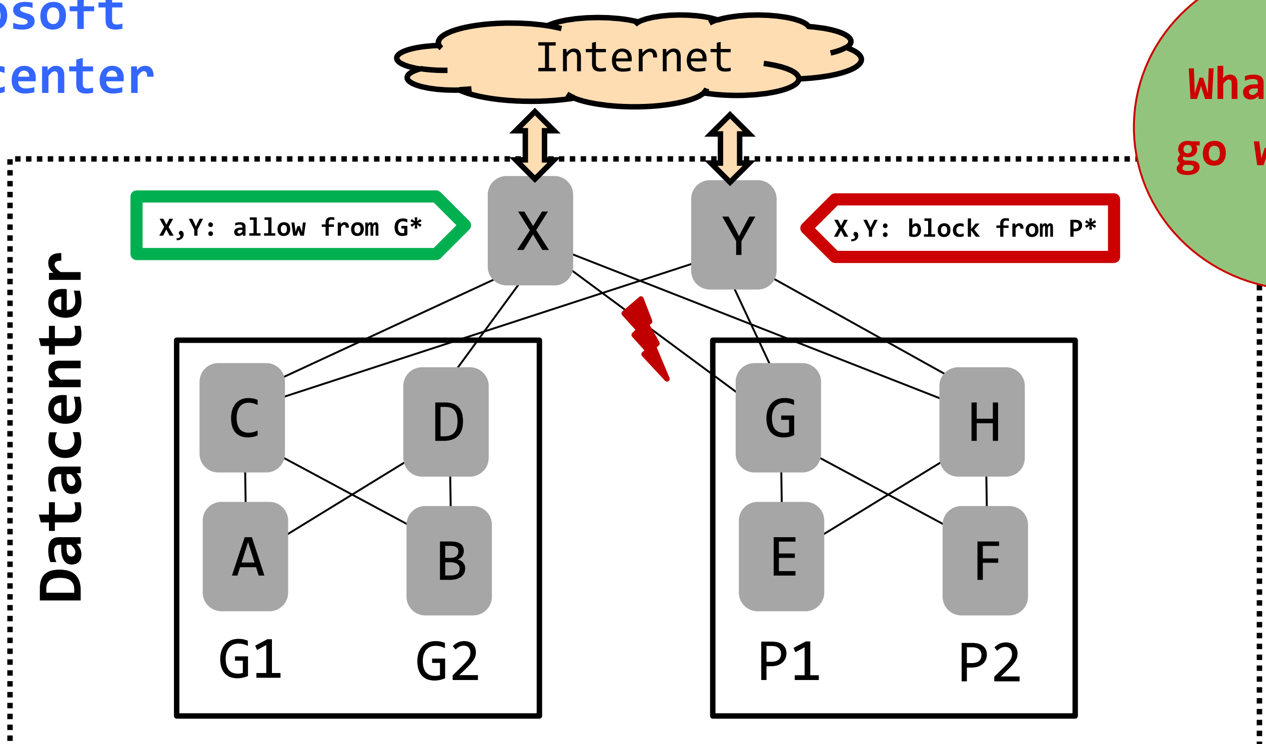
Example: BGP in **Microsoft datacenter**



A Reason: Complexity

Especially Under Failures (Policy Compliance)

Example: BGP in **Microsoft datacenter**



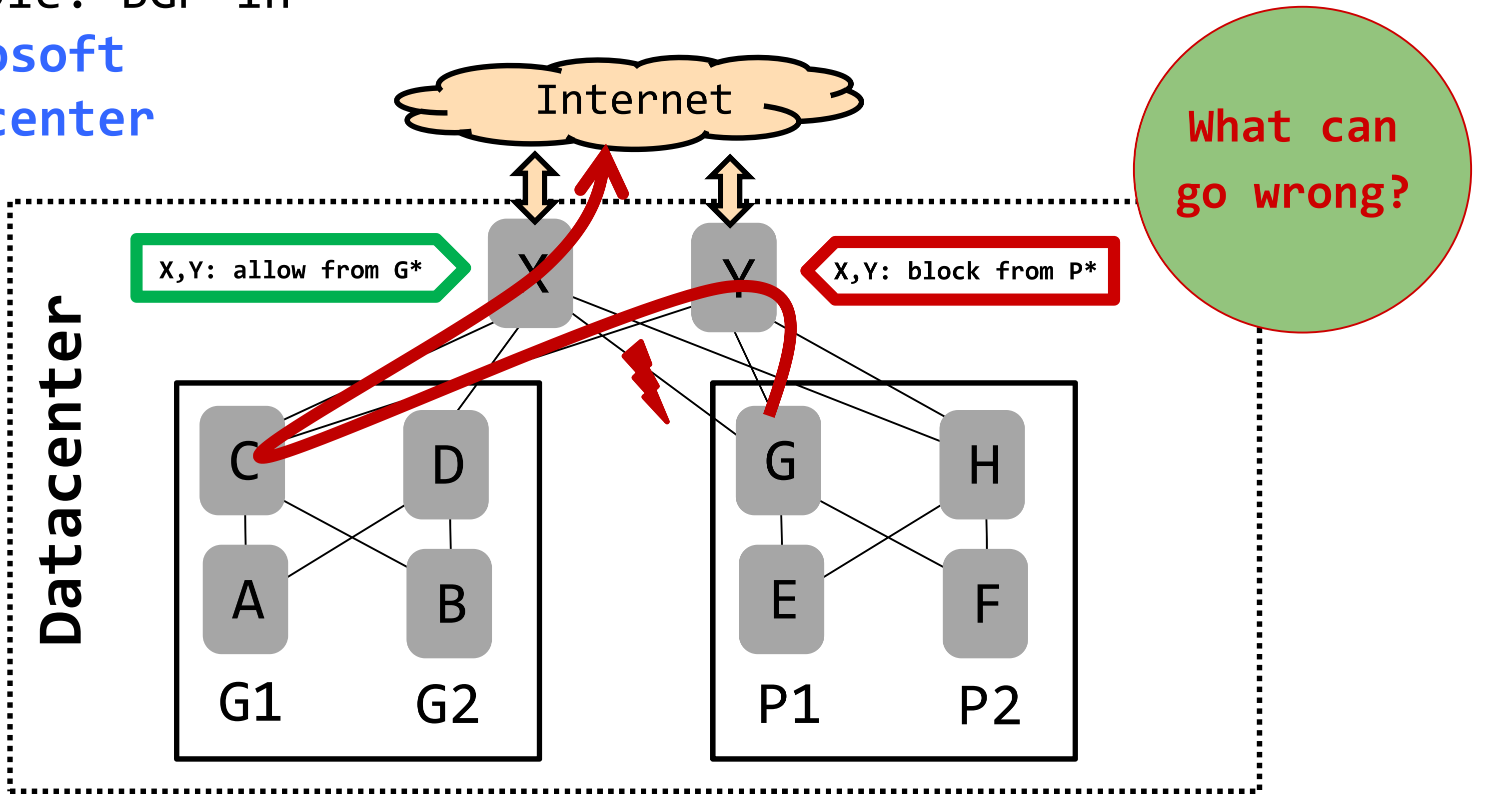
What can go wrong?

If link (G,X) fails and traffic from G is rerouted via Y and C to X:
X announces (does not block) G and H as it comes from C. (Note: BGP.)

A Reason: Complexity

Especially Under Failures (Policy Compliance)

Example: BGP in **Microsoft datacenter**



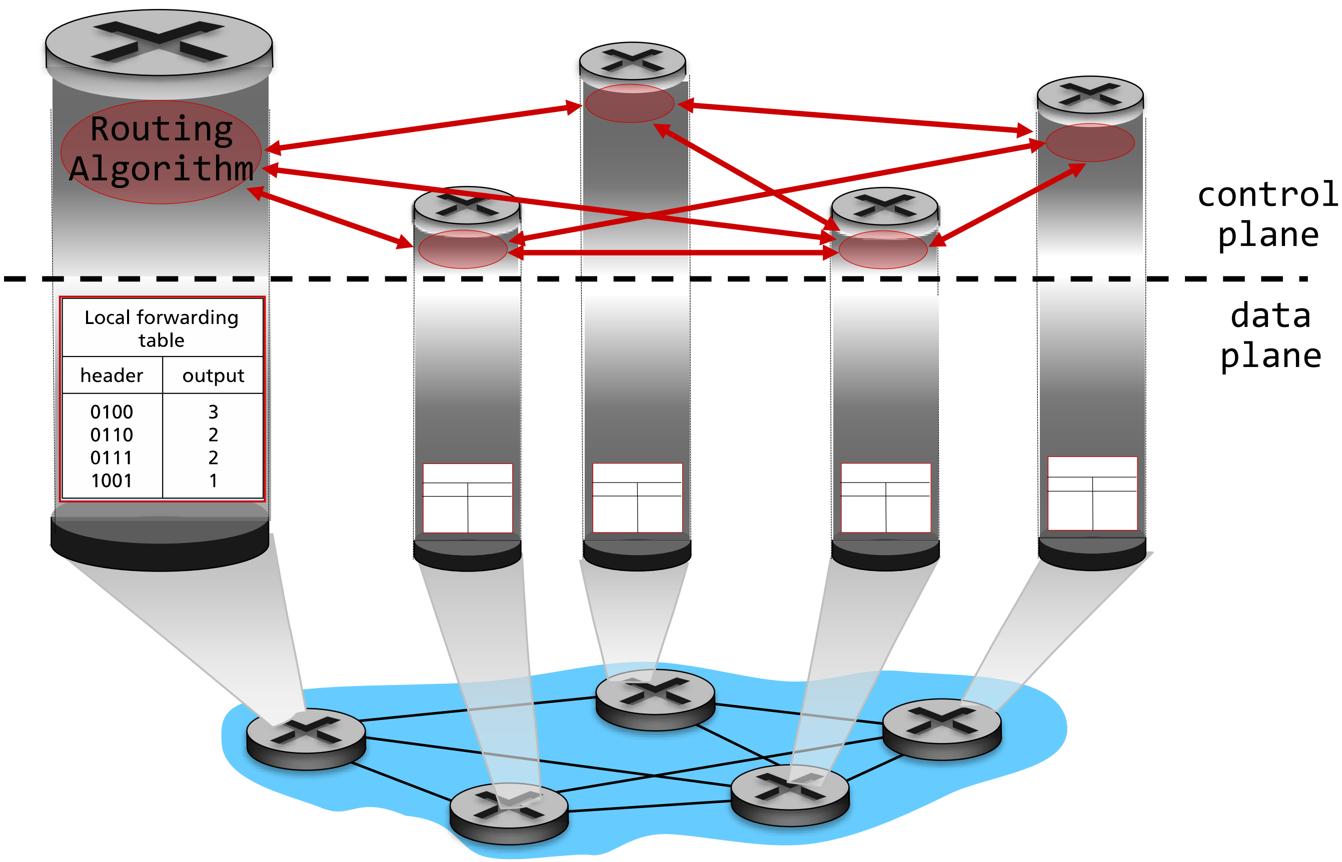
If link (G,X) fails and traffic from G is rerouted via Y and C to X:
X announces (does not block) G and H as it comes from C. (Note: BGP.)

Particularly Difficult

Fast Rerouting

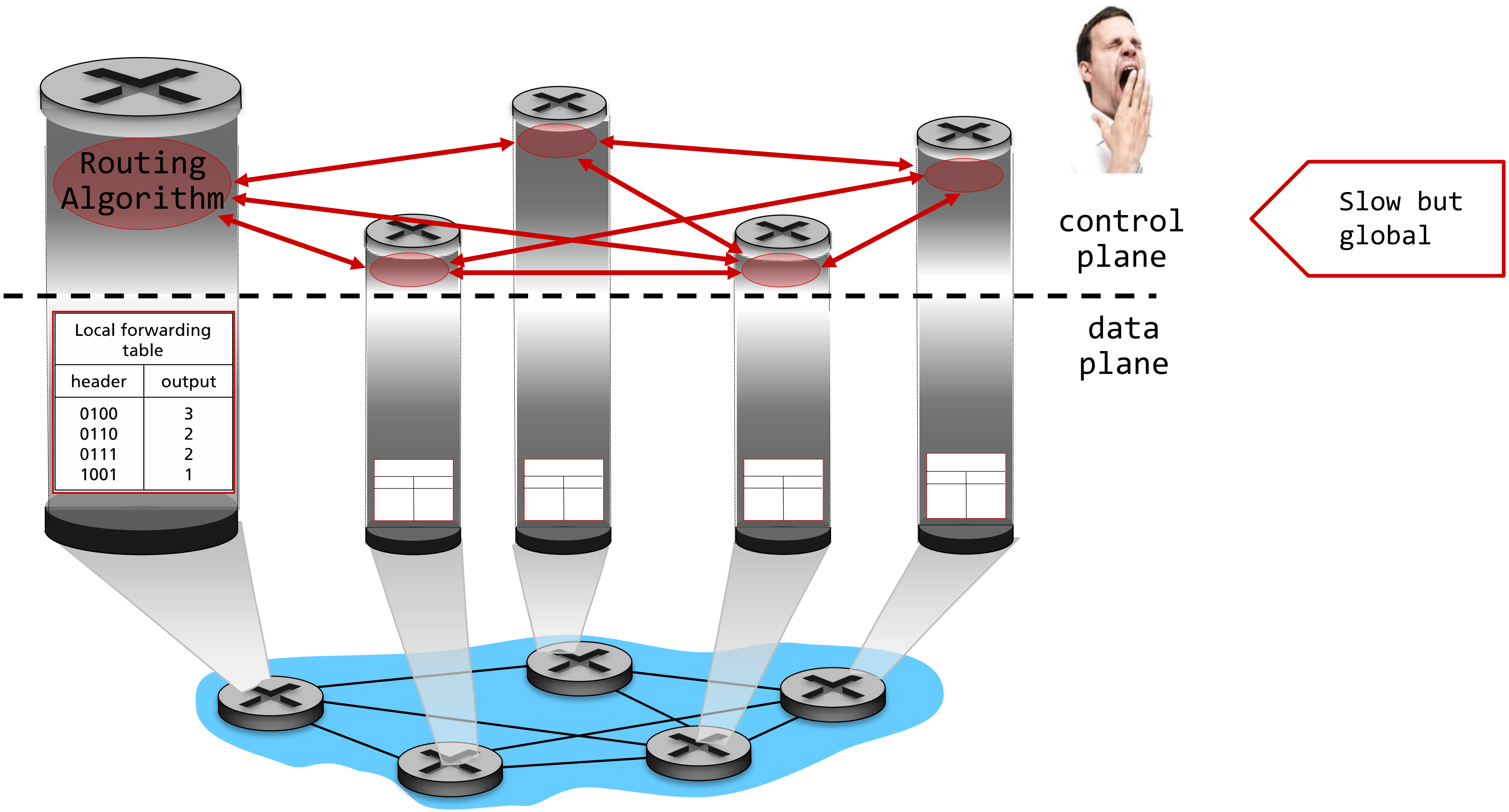
Particularly Difficult

Fast Rerouting



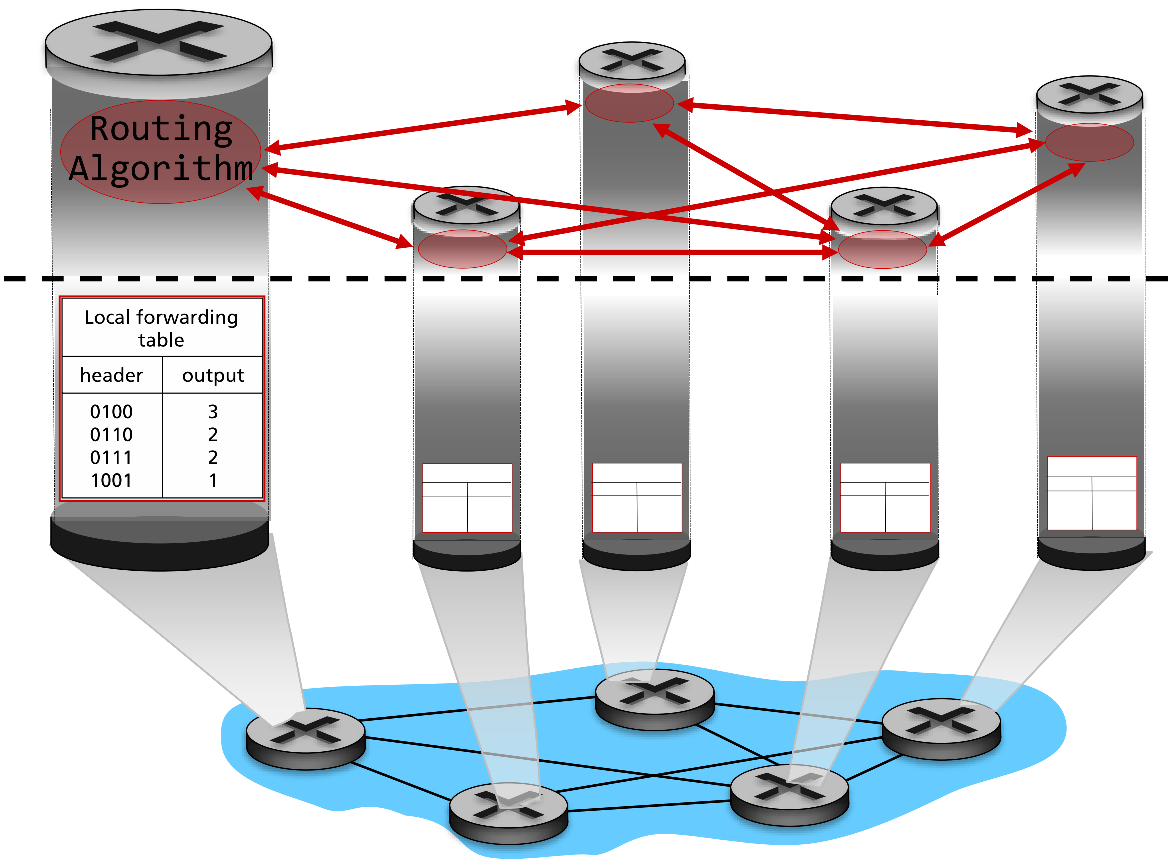
Particularly Difficult

Fast Rerouting



Particularly Difficult

Fast Rerouting



Fast but local

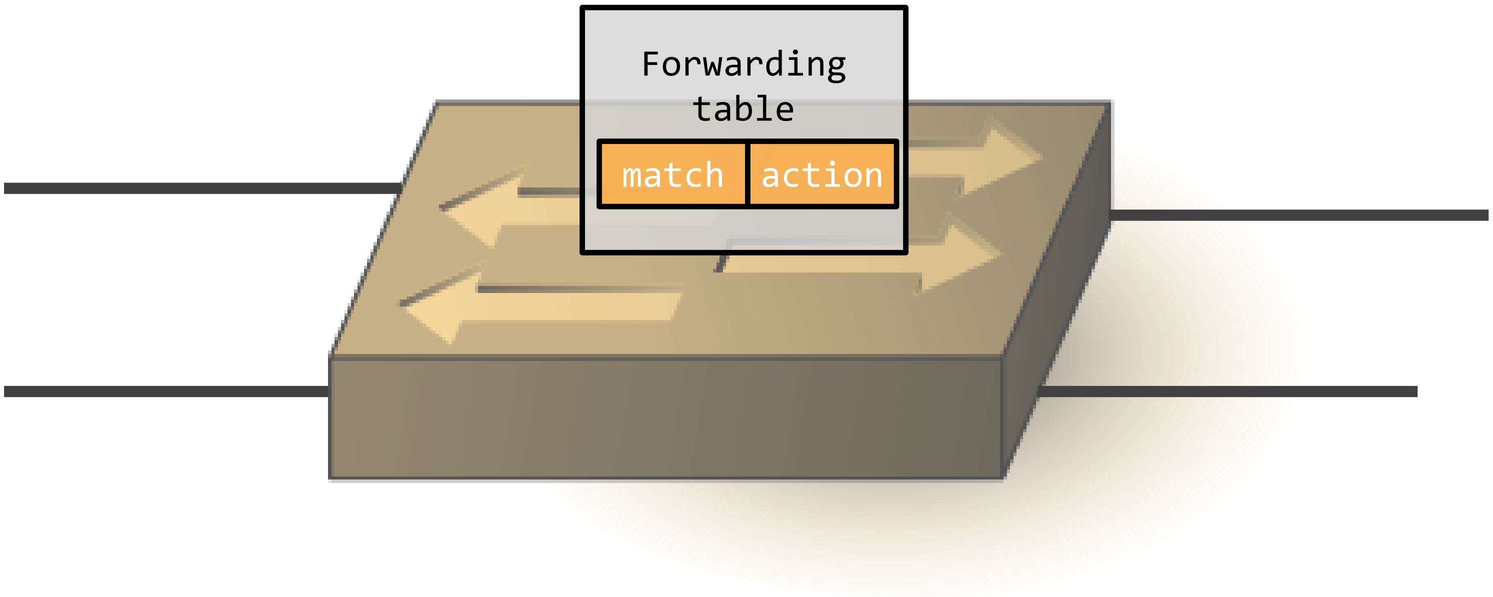
Information at Switch for

Local Decision Making?

Information at Switch for

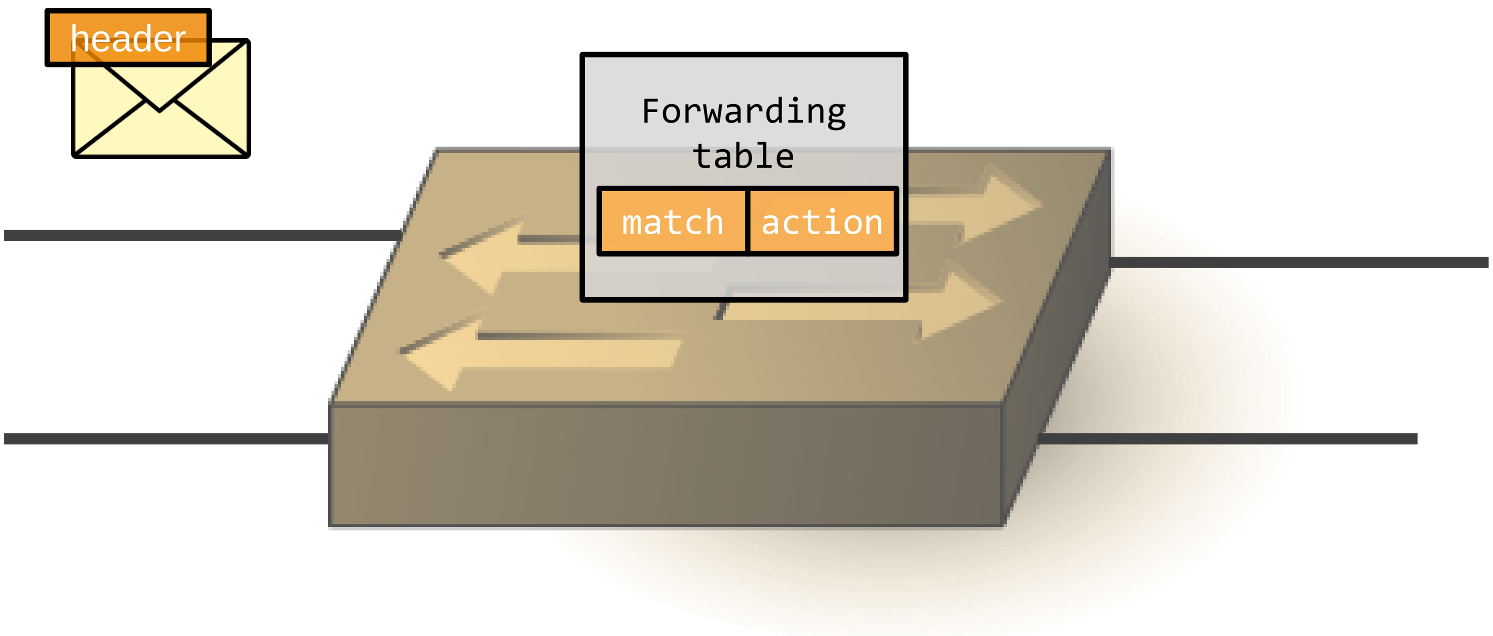
Local Decision Making?

- Nodes locally store a forwarding Match -> Action table



Information at Switch for Local Decision Making?

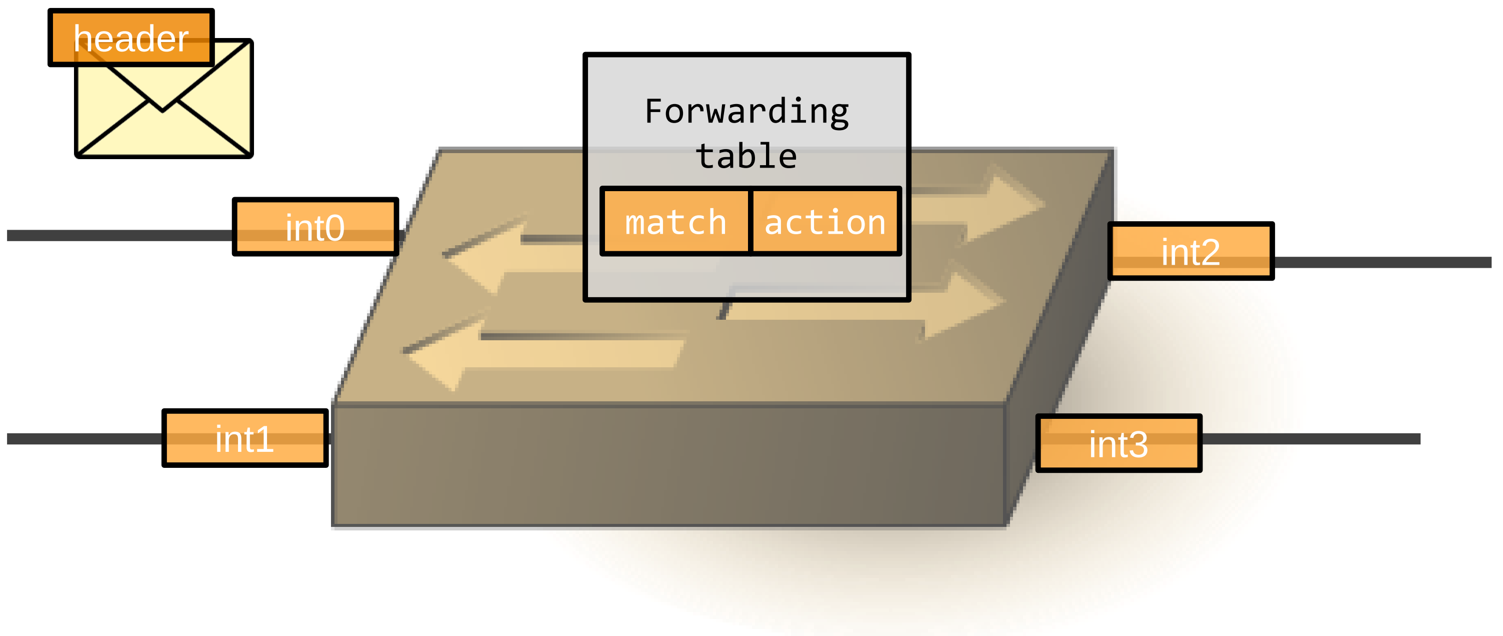
⇢ The Packet Header (e.g., source, destination)



Information at Switch for

Local Decision Making?

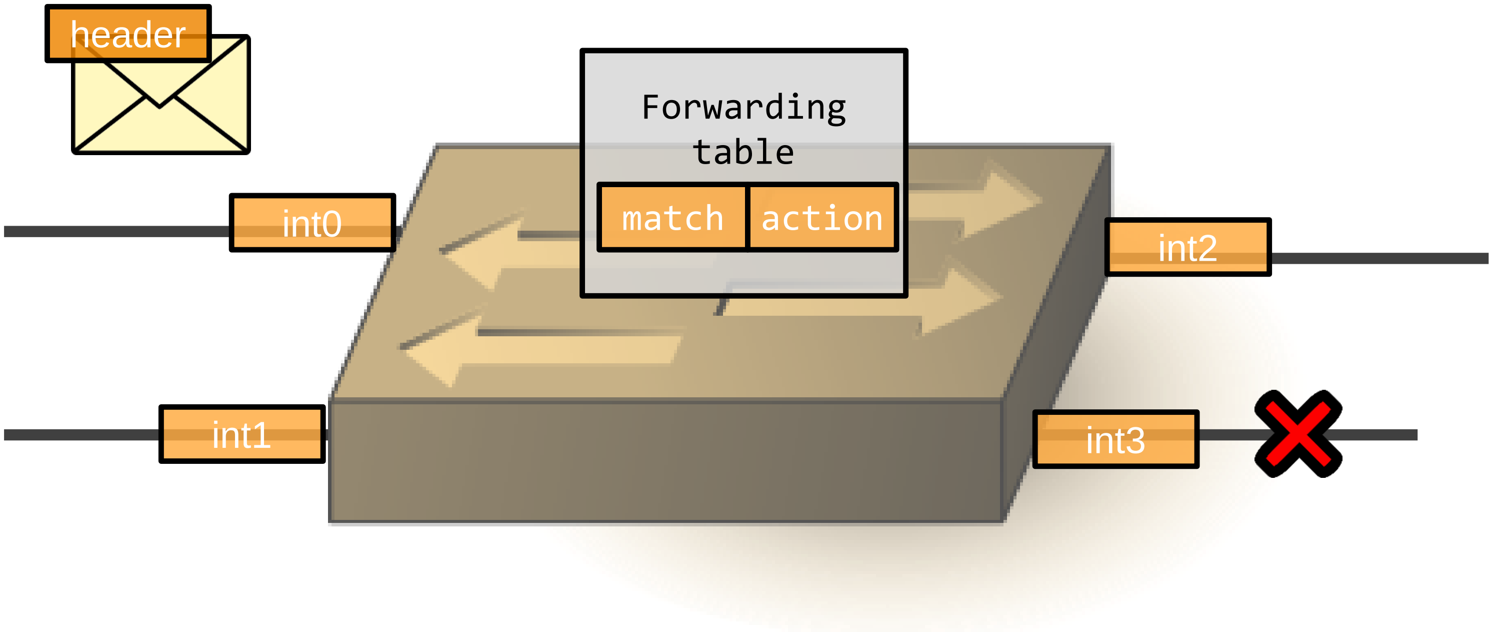
⇢ The Inport of the received packet



Information at Switch for

Local Decision Making?

⇢ Which incident links failed



Objective

What-if Analysis & Synthesis

- … for robust networks tolerating many link failures.
- Verification: Are the current forwarding rules policy compliant (reachability, waypoint traversal) even under failures?
- Synthesis: Can we pre-install local fast failover rules which ensure reachability under multiple failures?
- In general: How many failures can be tolerated by static forwarding tables?

Objective

What-if Analysis & Synthesis

- … for robust networks tolerating many link failures.
- Verification: Are the current forwarding rules policy compliant (reachability, waypoint traversal) even under failures?
- Synthesis: Can we pre-install local fast failover rules which ensure reachability under multiple failures?
- In general: How many failures can be tolerated by static forwarding tables?

Imagine SDN model where we can directly program the dataplane.

Two fundamental

Notions of Resilience

Ideal resilience

Given a *k*-connected graphs, fast reroute can tolerate *any k-1 link failures.*

Perfect resilience

Fast reroute can tolerate any failures as long as the unterlying network is *physically connected*.

⇢ What is the difference? Which is stronger?

A big open challenge

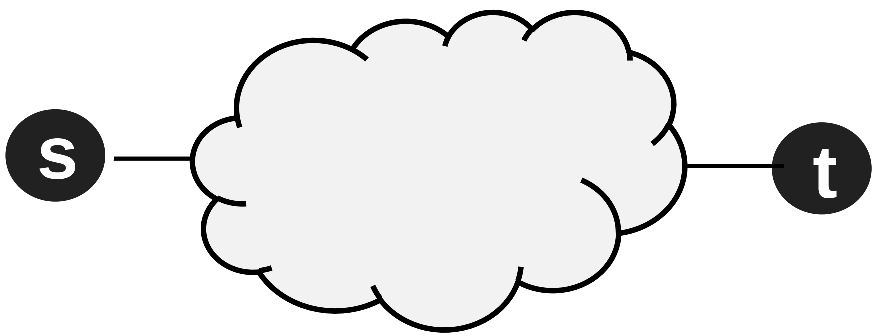
Ideal Resilience

- Given a k-connected network: how many link failures can a fast re-routing mechanism tolerate? Conjecture: k-1.
- Assume: cannot change header, but can match inport, src and dst

A big open challenge

Ideal Resilience

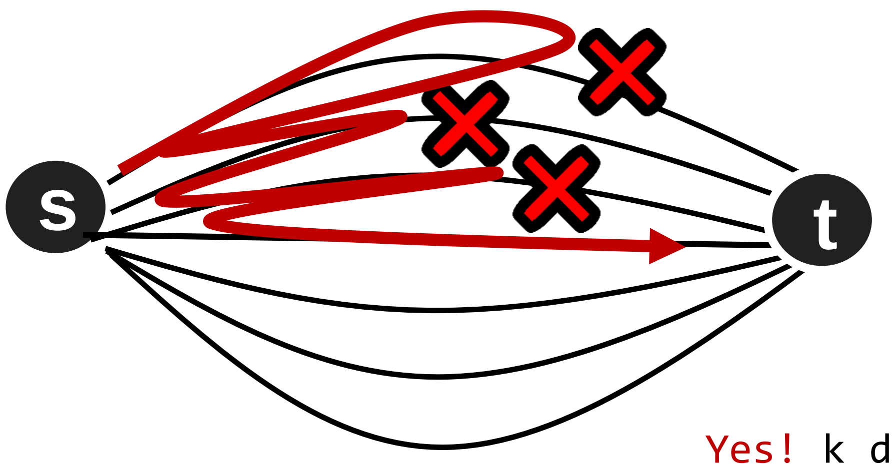
- Given a k-connected network: how many link failures can a fast re-routing mechanism tolerate? Conjecture: k-1.
- Assume: cannot change header, but can match inport, src and dst



A big open challenge

Ideal Resilience

- Given a k-connected network: how many link failures can a fast re-routing mechanism tolerate? Conjecture: k-1.
- Assume: cannot change header, but can match inport, src and dst



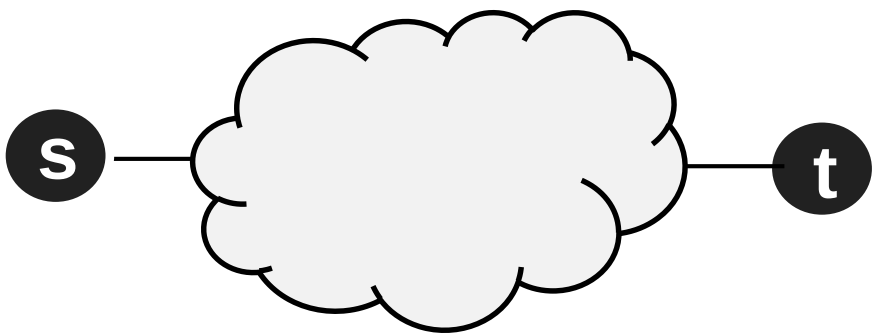
Yes! k disjoint paths: try one after the other, routing *back to source* each time.

A big open challenge

Ideal Resilience

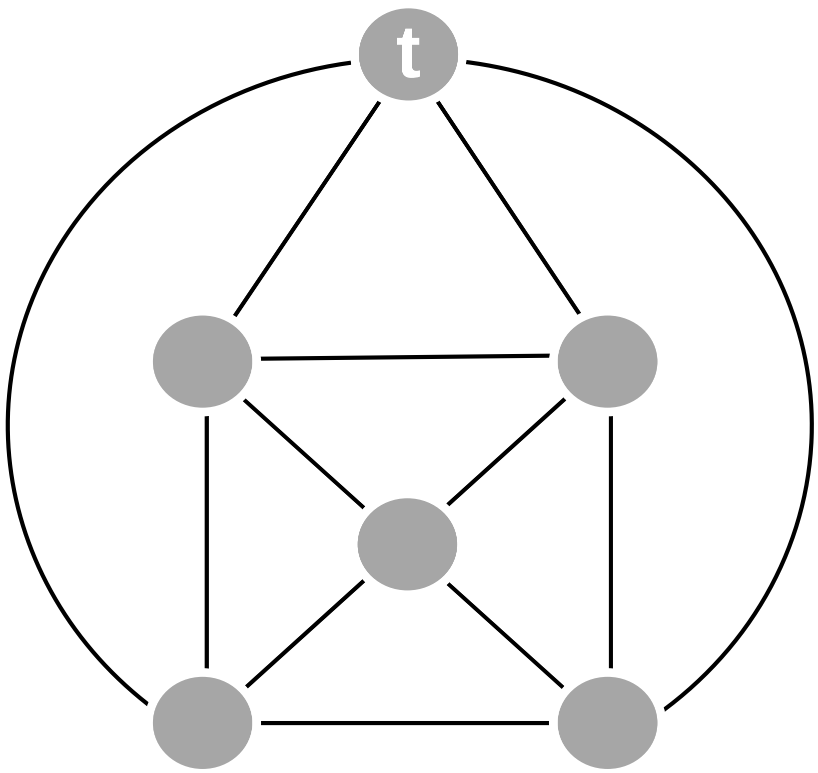
- Given a k-connected network: how many link failures can a fast re-routing mechanism tolerate? Conjecture: k-1.
- Assume: cannot change header, but can match inport, and dst

What if I cannot match source?! Open conjecture.



State-of-the-Art Approach for Ideal Resilience

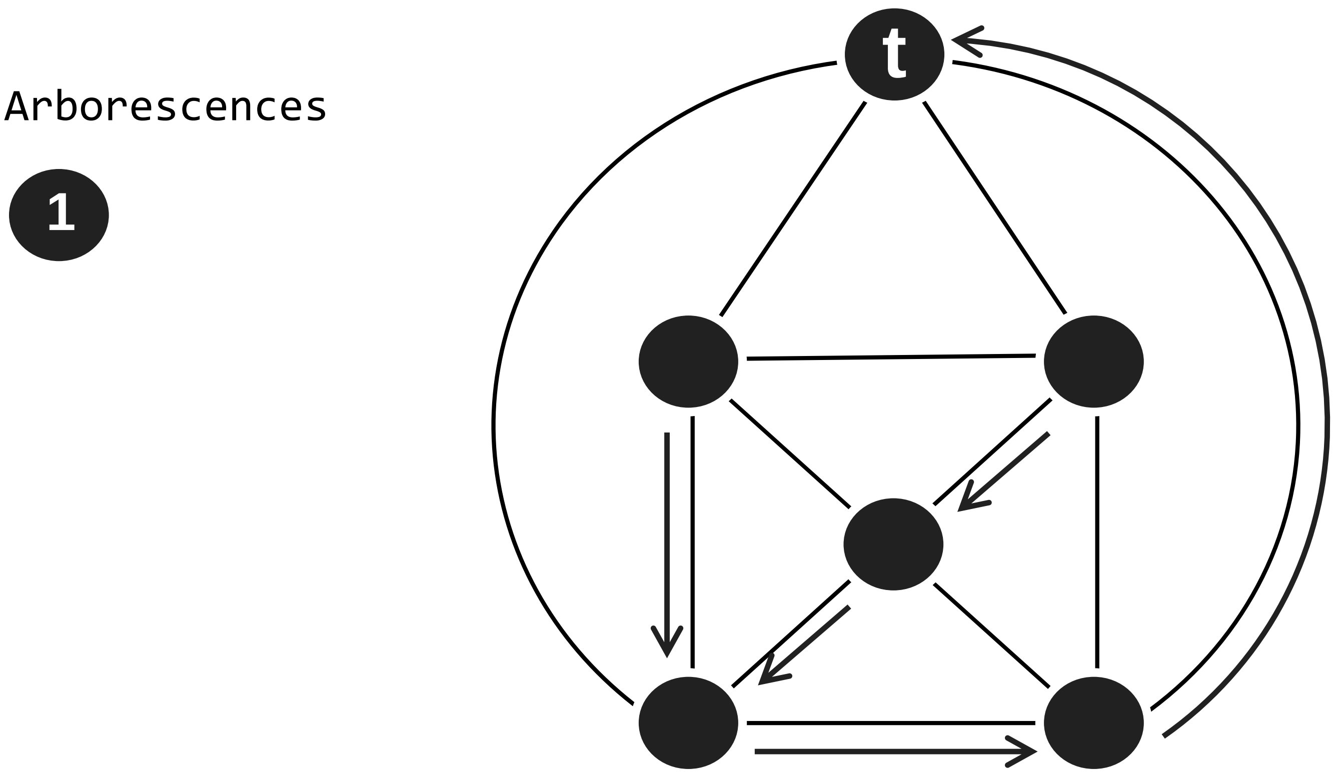
Spanning Arborescences



- Fact: k-connected network has k-arborescence decomposition
- Basically disjoint spanning trees directed to destination

Spanning Arborescences

Arborescences

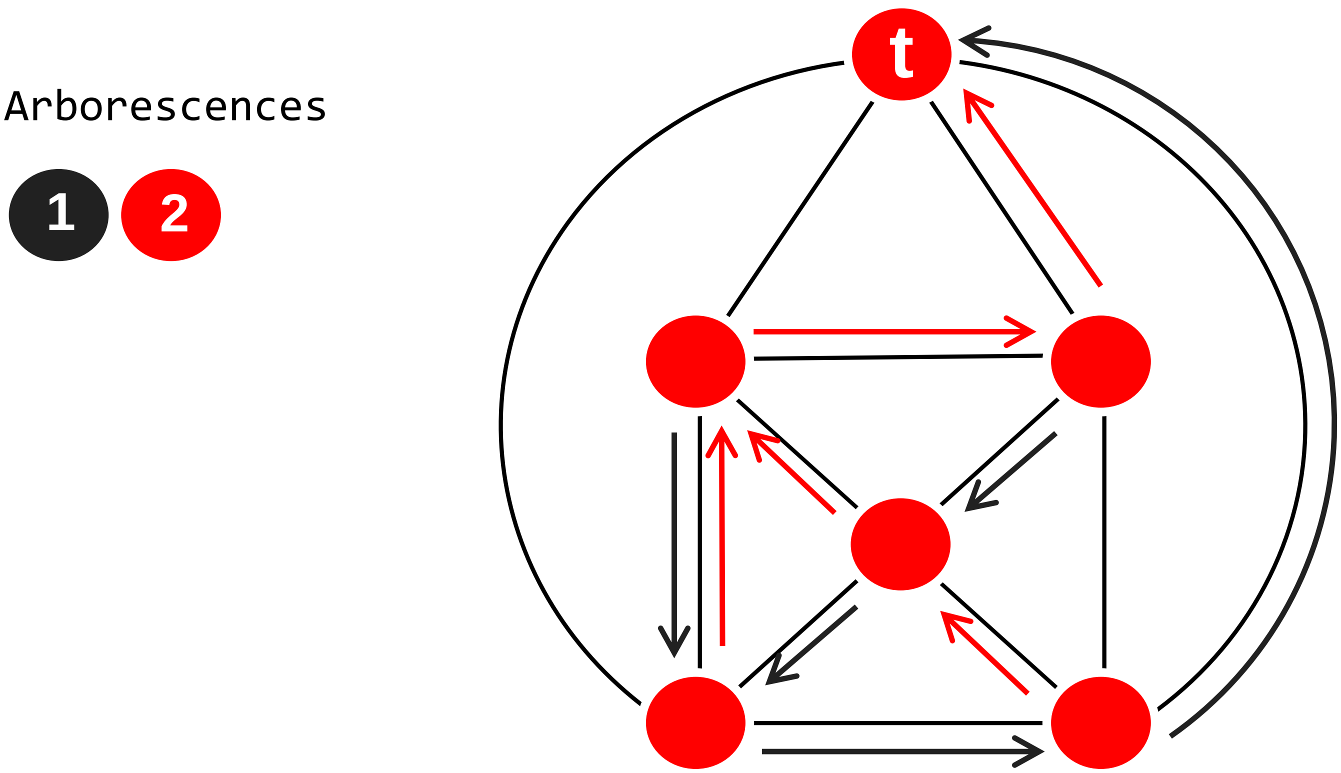


⇢ Fact: k-connected network has k-arborescence decomposition

⇢ Basically disjoint spanning trees directed to destination

Spanning Arborescences

Arborescences



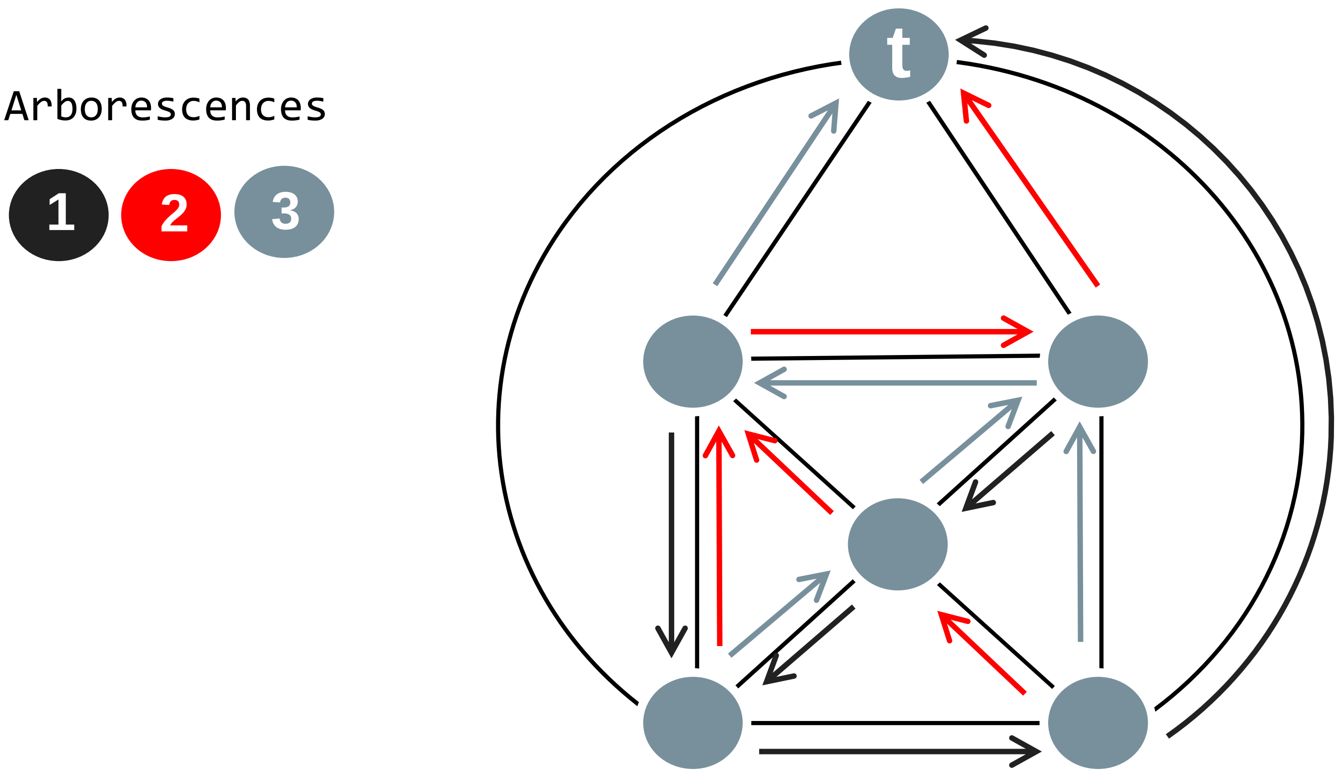
⇢ Fact: k-connected network has k-arborescence decomposition

⇢ Basically disjoint spanning trees directed to destination

State-of-the-Art Approach for Ideal Resilience

Spanning Arborescences

Arborescences



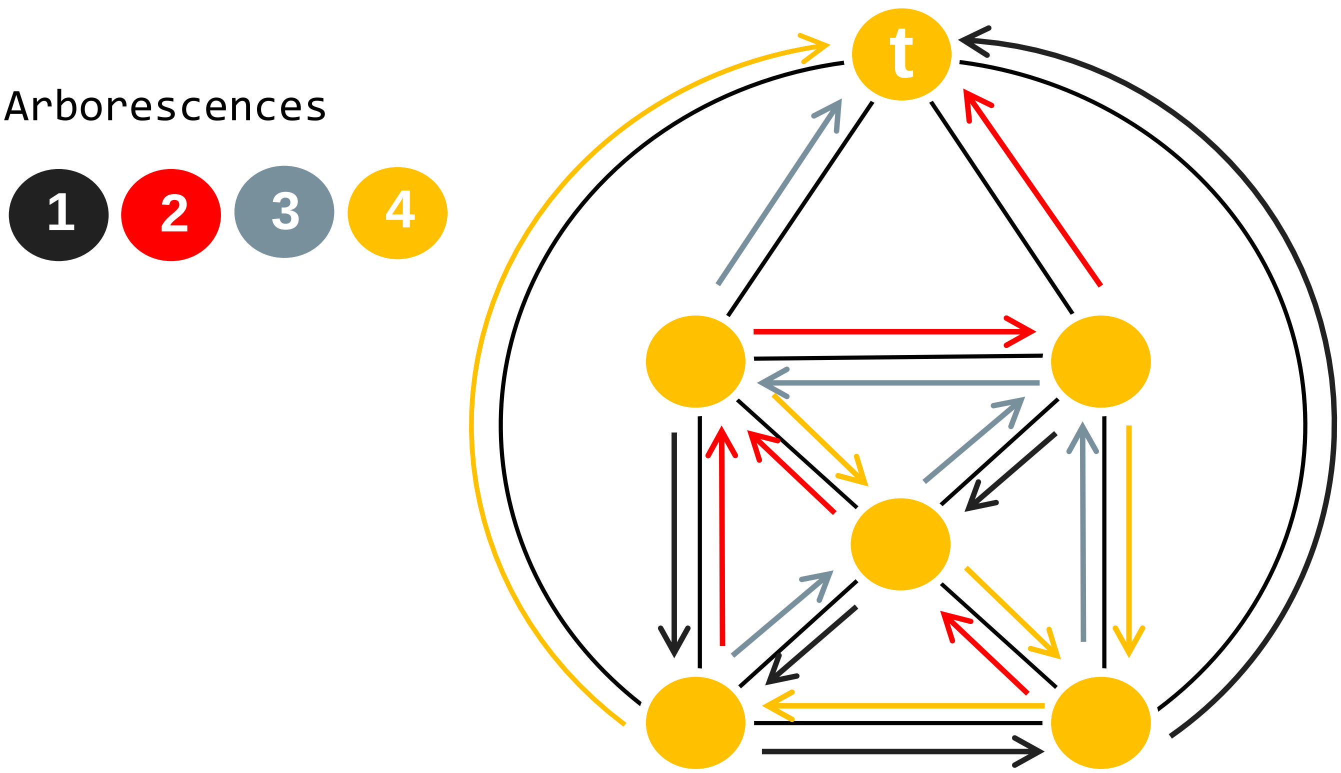
⇢ Fact: k-connected network has k-arborescence decomposition

⇢ Basically disjoint spanning trees directed to destination

State-of-the-Art Approach for Ideal Resilience

Spanning Arborescences

Arborescences

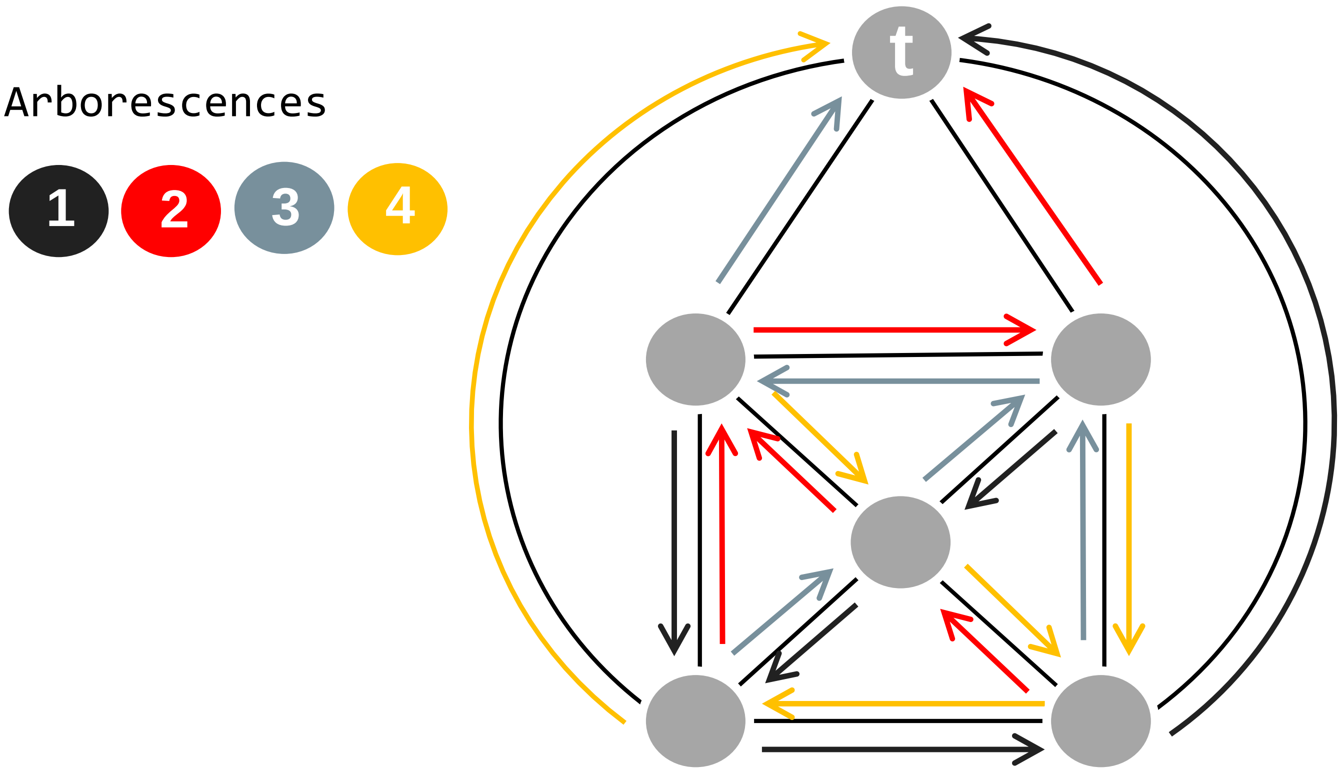


⇢ Fact: k-connected network has k-arborescence decomposition

⇢ Basically disjoint spanning trees directed to destination

Spanning Arborescences

Arborescences



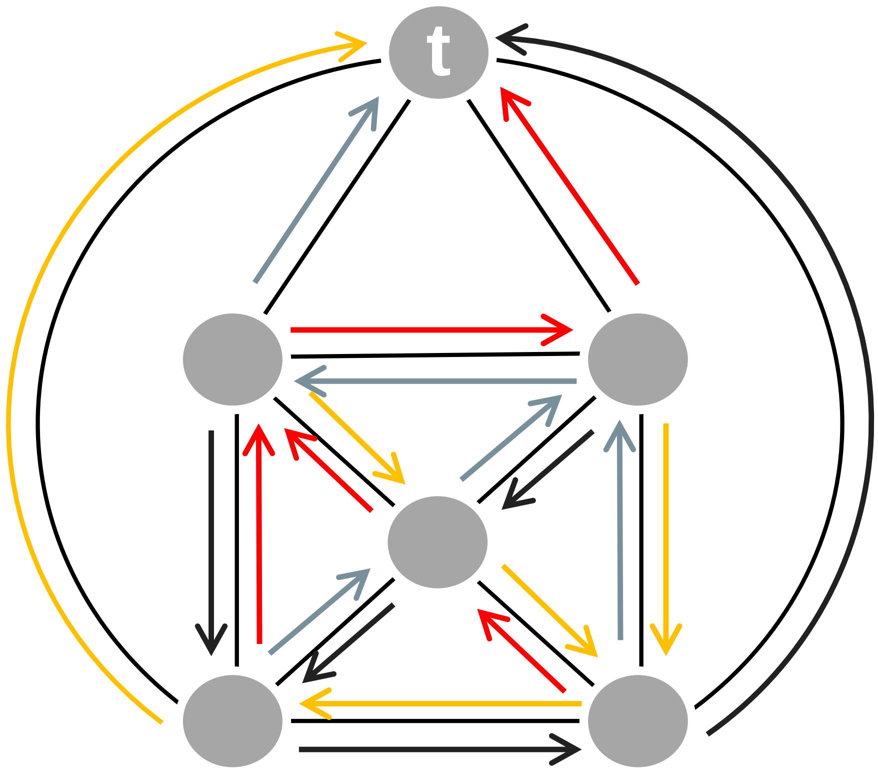
⇢ Fact: k-connected network has k-arborescence decomposition

⇢ Basically disjoint spanning trees directed to destination

(k/2-1)-resilient with circular

Arborescence Routing

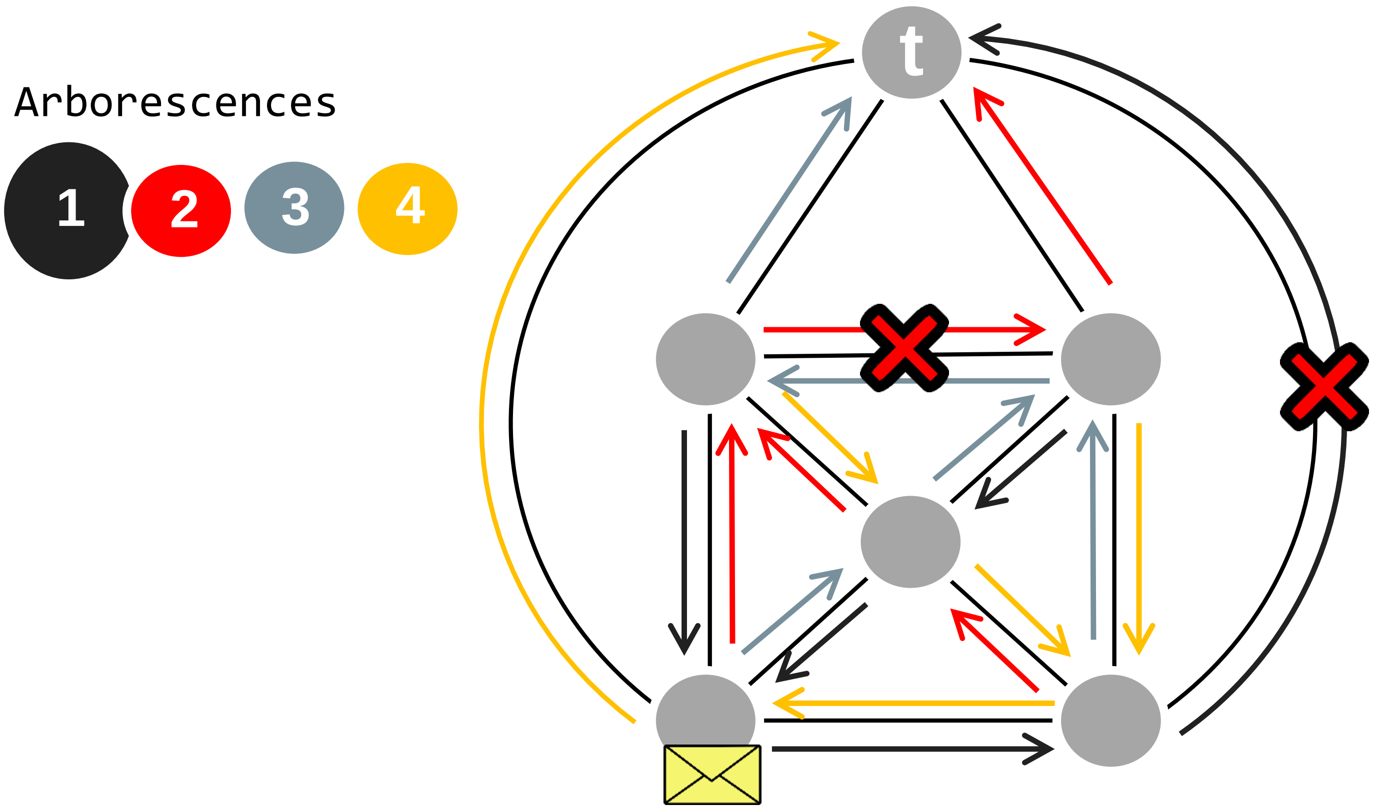
Arborescences



(k/2-1)-resilient with circular

Arborescence Routing

Arborescences

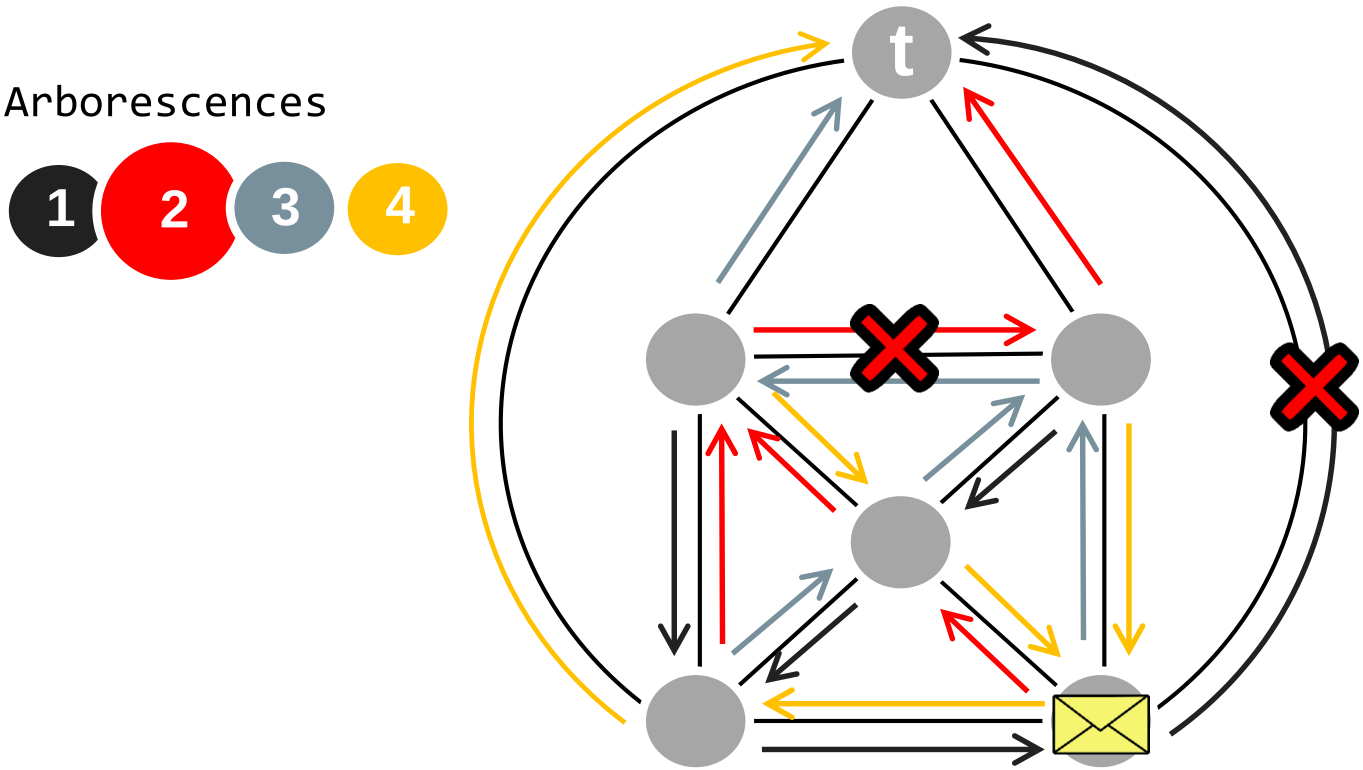


⇢ Try arborescences in order

(k/2-1)-resilient with circular

Arborescence Routing

Arborescences

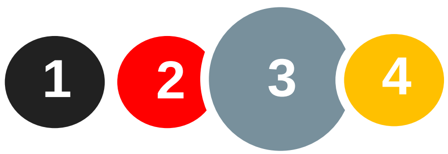


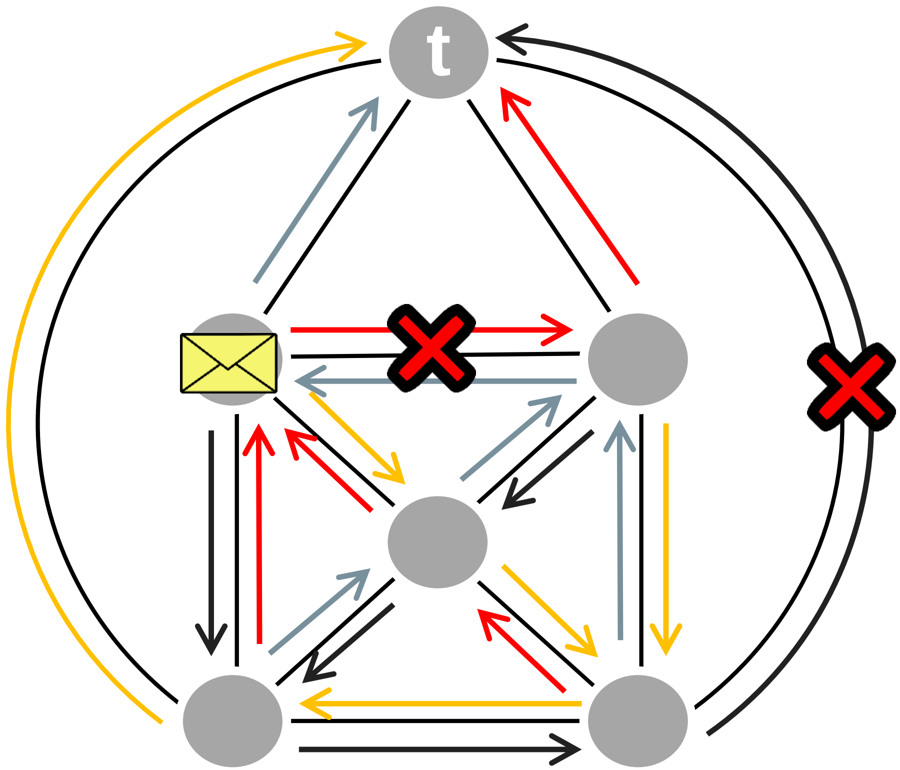
⇢ Try arborescences in order

(k/2-1)-resilient with circular

Arborescence Routing

Arborescences



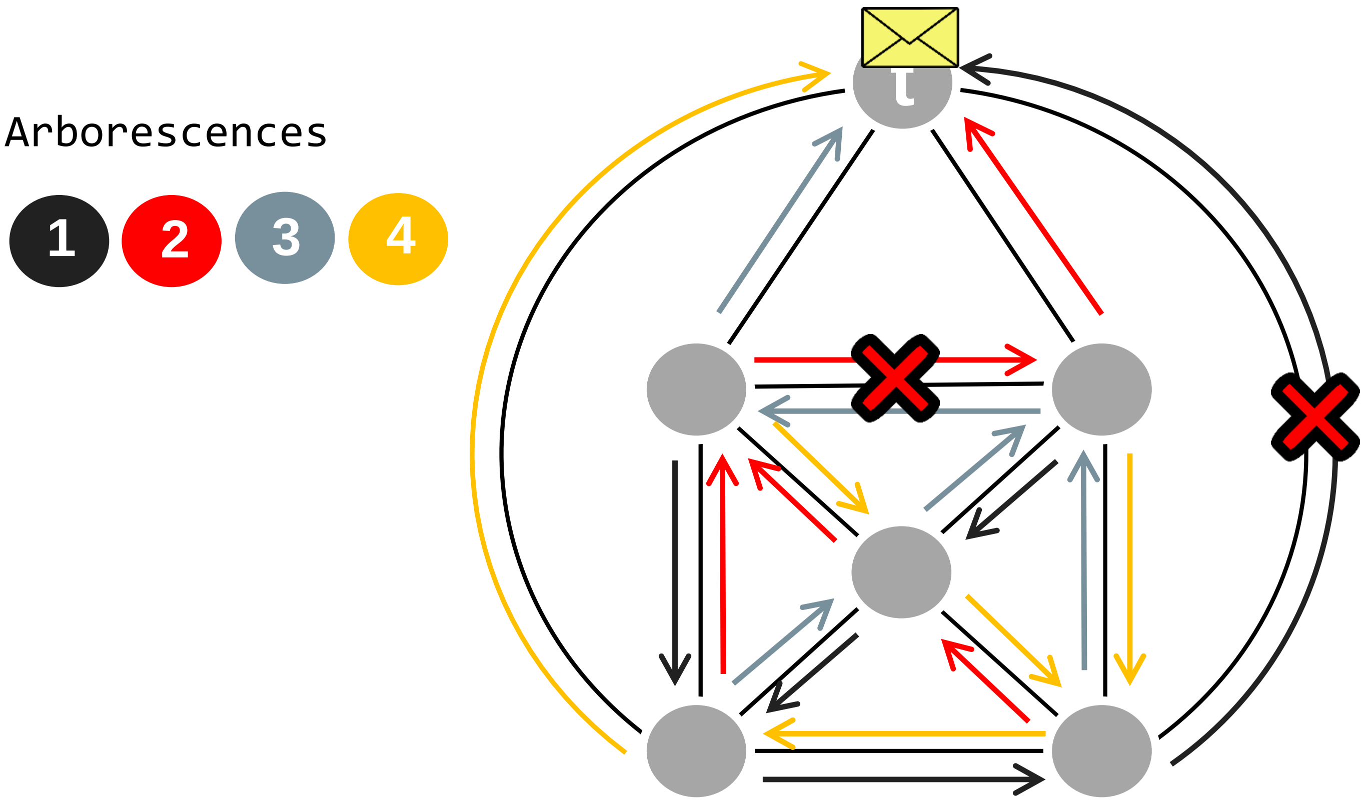


⇢ Try arborescences in order

(k/2-1)-resilient with circular

Arborescence Routing

Arborescences

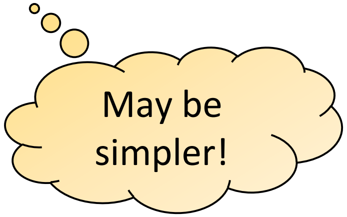


⇢ Try arborescences in order

⇢ k/2-1 resilient: link failure affects at most 2 arborescences

Research Challenges

- Complexity of verifying resilience and policy-compliance?
- Algorithms for synthesizing resilient fast reroute mechanisms?
- Application to specific protocols, like MPLS or Segment Routing?



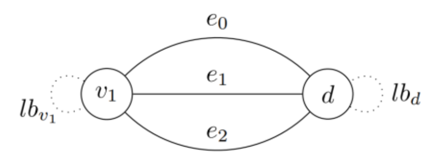
A General Solution: Automation

Synthesis with BDDs

- Binary decision diagrams (BDDs) allow us to synthesize resilient routings
  - … or to repair
- Attractive: all solutions, compactly represented
  - Supports operator preferences!
  - Better alternative to e.g. ILPs
- Still somewhat slow

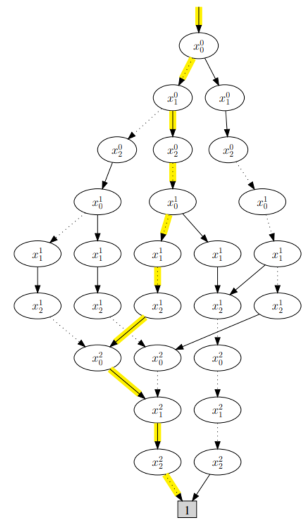
Synthesis with BDDs

Network:



- Binary decision diagrams (BDDs) allow us to synthesize resilient routings
  - … or to repair
- Attractive: all solutions, compactly represented
  - Supports operator preferences!
  - Better alternative to e.g. ILPs
- Still somewhat slow

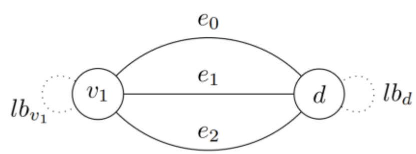
BDD 2-resilient routing:s



A General Solution: Automation

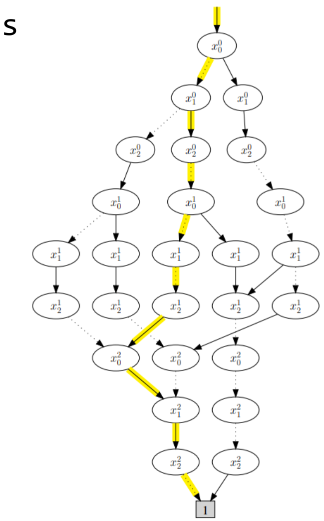
Synthesis with BDDs

Network:



- Binary decision diagrams (BDDs) allow us to synthesize resilient routings
  - … or to repair
- Attractive: all solutions, compactly represented
  - Supports operator preferences!
  - Better alternative to e.g. ILPs
- Still somewhat slow

BDD 2-resilient routing:s

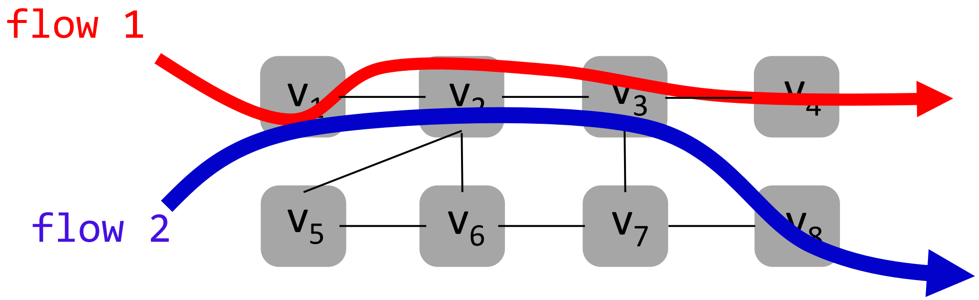


For specific protocols we can be faster!

Faster for specific protocol:

MPLS Fast Reroute (FRR)

⇢ Forwarding based on **top label** of label **stack**

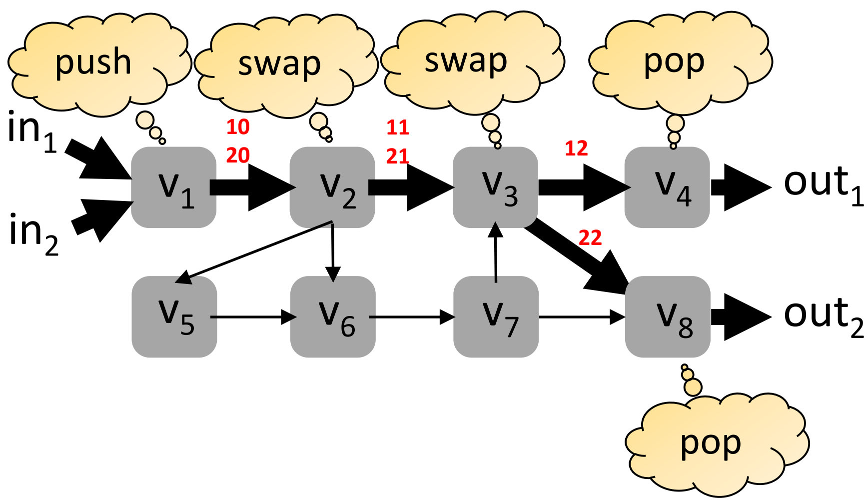


Default routing of two flows

Faster for specific protocol:

MPLS Fast Reroute (FRR)

⇢ Forwarding based on **top label** of label **stack**

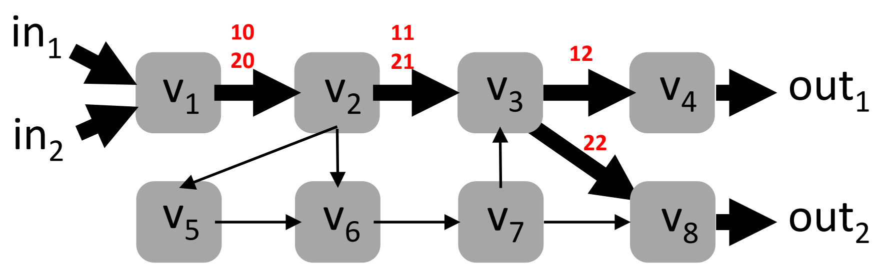


Default routing of two flows

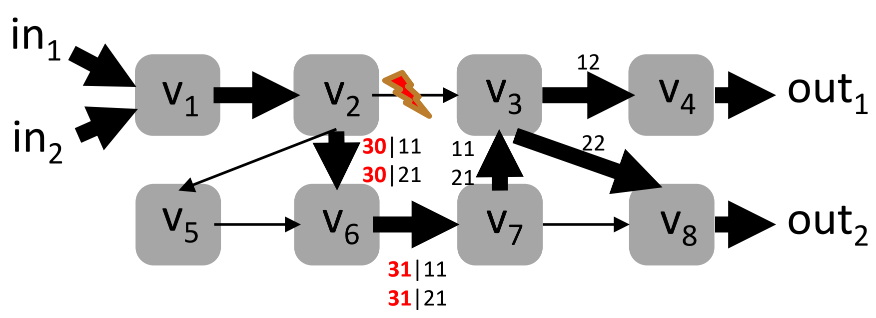
Faster for specific protocol:

MPLS Fast Reroute (FRR)

⇢ Forwarding based on **top label** of label **stack**



Default routing of two flows

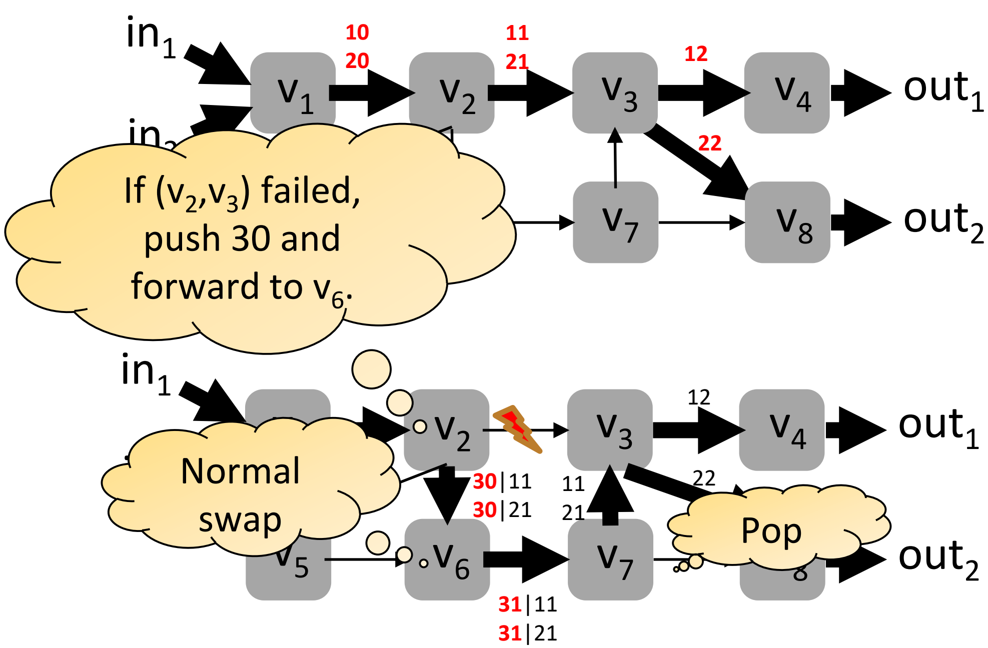


One failure: push 30: route around (v_2,v_3)

Faster for specific protocol:

MPLS Fast Reroute (FRR)

⇢ Forwarding based on **top label** of label **stack**



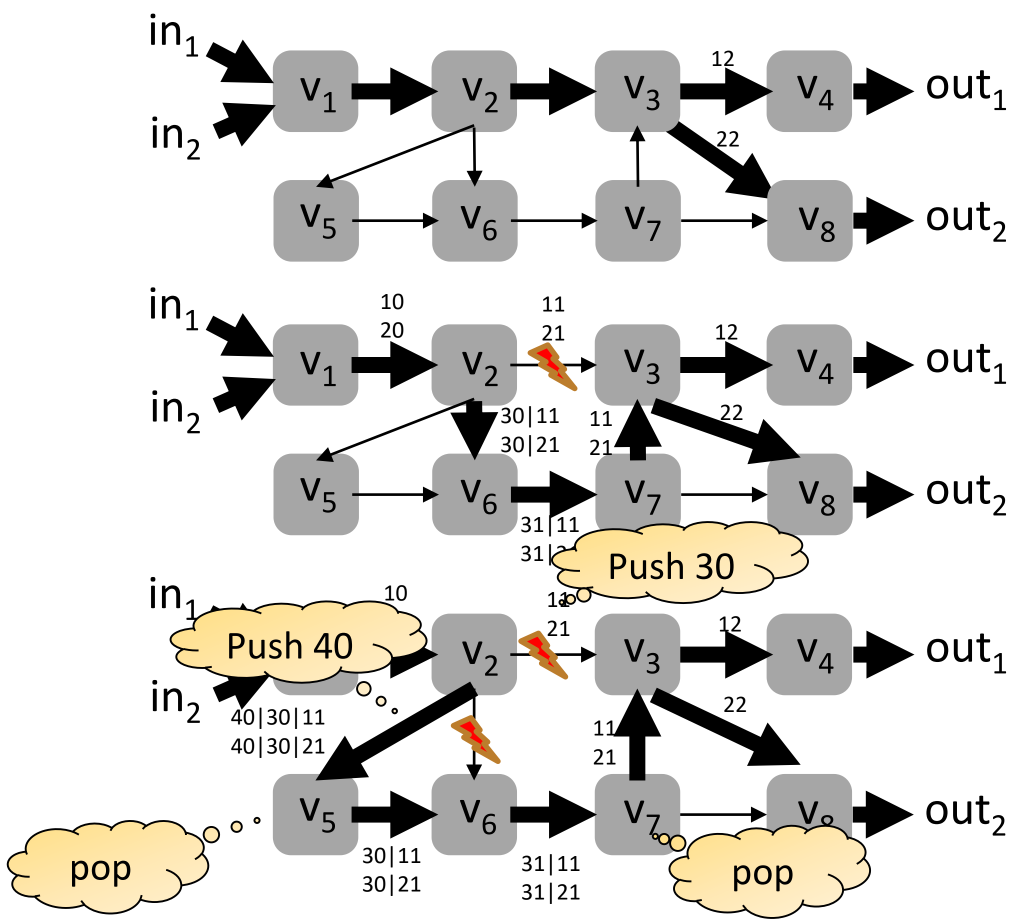
Default routing of two flows

One failure: push 30: route around (v_2,v_3)

Faster for specific protocol:

MPLS Fast Reroute (FRR)

⇢ Multiple link failures: simply recursive



Original Routing

One failure: push 30: route around (v_2,v_3)

Two failures: first push 30: route around (v_2,v_3)

Push recursively 40: route around (v_2,v_6)

Faster for specific protocol:

MPLS Fast Reroute (FRR)

- Specific structure of MPLS networks can be exploited for fast what-if analysis: it‘s a stack machine
- Can use the result by Büchi: set of all reachable configurations of pushdown automaton is regular set
- We hence simply use Nondeterministic Finite Automata when reasoning about the pushdown automata
- The resulting regular operations are all polynomial time

Julius Richard Büchi

1924-1984

Swiss logician

Faster for specific protocol:

MPLS Fast Reroute (FRR)

- Specific structure of MPLS networks can be exploited for fast what-if analysis: it's a stack machine
- Can use the result by Büchi: set of all reachable configurations of pushdown automaton is regular set
- We hence simply use Nondeterministic Finite Automata when reasoning about the pushdown automata
- The resulting regular operations are all polynomial time

Julius Richard Büchi

1924-1984

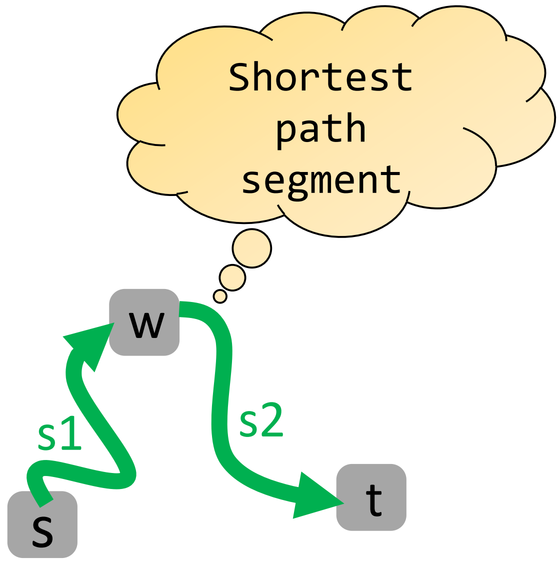
Swiss logician

What about complexity of other special networks?

Example:

Segment Routing FRR

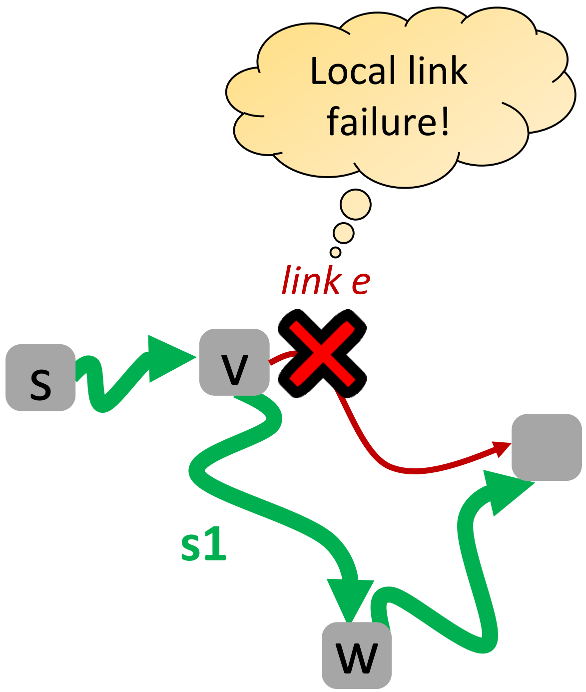
- Segment routing (SR): shortest path routing on segments (between waypoints)
- Waypoints can perform functions (also NFVs), like pushing another waypoint to header
- A little bit like Valiant Routing
- Waypoint "stack" can be used for fast reroute



Example:

How to Re-Route in SR?

- When a node v on route from s to t *locally detects* failure on link e, it can *push a waypoint* w.
- Rule: v should push a w such that the shortest path s1 (from v to w) and the shortest path s2 (from w to t) *does not include e again*! So can route around failed link.
- Which waypoint w should fast reroute push?



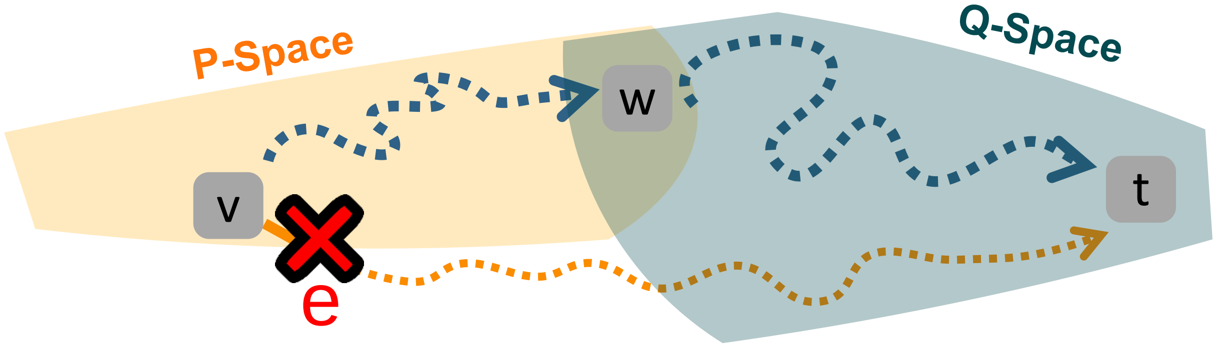
Example:

How to Re-Route in SR?

We need two definitions:

⇢ *P-Space*: nodes which v reaches on shortest paths without e

⇢ *Q-Space*: nodes which reach t on shortest paths without e



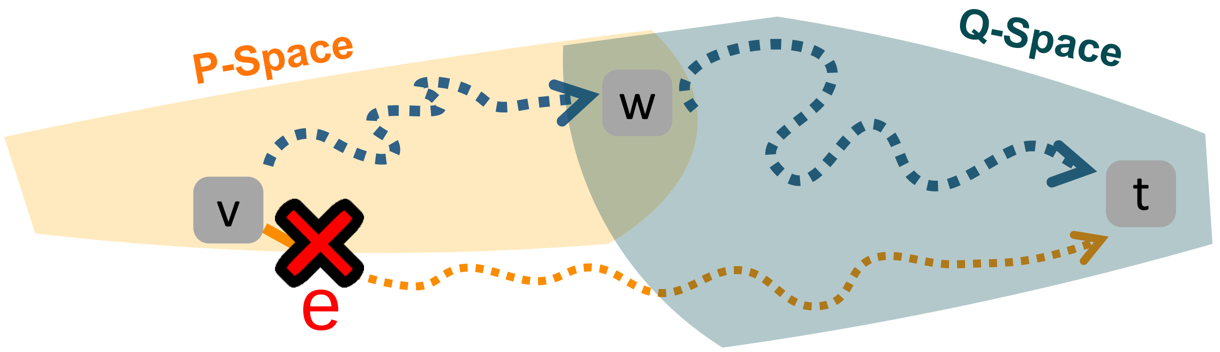
Example:

How to Re-Route in SR?

We need two definitions:

⇢ *P-Space*: nodes which v reaches on shortest paths without e

⇢ *Q-Space*: nodes which reach t on shortest paths without e

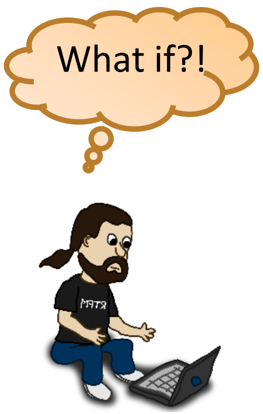


⇢ Choose any waypoint w *at intersection** for rerouting!

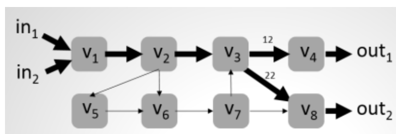
*If intersection empty, spaces must be adjacent and there is also a (different) solution.

Opportunity: Fast reroute and robust networks with

Automation



FT	In-I	In-Label	Out-I	op
τ_{v_1}	in_1	$\perp$	(v_1, v_2)	$push(10)$
	in_2	$\perp$	(v_1, v_2)	$push(20)$
τ_{v_2}	(v_1, v_2)	10	(v_2, v_3)	$swap(11)$
	(v_1, v_2)	20	(v_2, v_3)	$swap(21)$
τ_{v_3}	(v_2, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_2, v_3)	21	(v_3, v_8)	$swap(22)$
	(v_7, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_7, v_3)	21	(v_3, v_8)	$swap(22)$
τ_{v_4}	(v_3, v_4)	12	out_1	pop
τ_{v_5}	(v_2, v_5)	40	(v_5, v_6)	pop
τ_{v_6}	(v_2, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	61	(v_6, v_7)	$swap(62)$
	(v_5, v_6)	71	(v_6, v_7)	$swap(72)$
τ_{v_7}	(v_6, v_7)	31	(v_7, v_3)	pop
	(v_6, v_7)	62	(v_7, v_3)	$swap(11)$
	(v_6, v_7)	72	(v_7, v_8)	$swap(22)$
τ_{v_8}	(v_3, v_8)	22	out_2	pop
	(v_7, v_8)	22	out_2	pop

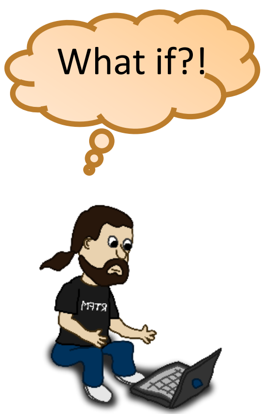


local FFT	Out-I	In-Label	Out-I	op
τ_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$push(30)$
	(v_2, v_3)	21	(v_2, v_6)	$push(30)$
	(v_2, v_6)	30	(v_2, v_5)	$push(40)$
global FFT	**Out-I**	**In-Label**	**Out-I**	**op**
τ'_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$swap(61)$
	(v_2, v_3)	21	(v_2, v_6)	$swap(71)$
	(v_2, v_6)	61	(v_2, v_5)	$push(40)$
	(v_2, v_6)	71	(v_2, v_5)	$push(40)$

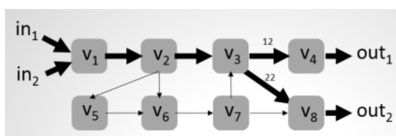
Router **configurations**
(Cisco, Juniper, etc.)

Opportunity: Fast reroute and robust networks with

Automation

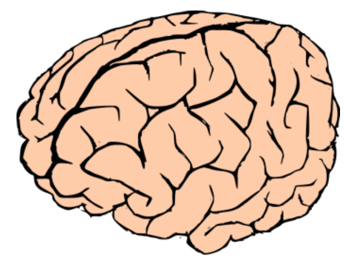


FT	In-I	In-Label	Out-I	op
τ_{v_1}	in_1	$\perp$	(v_1, v_2)	$push(10)$
	in_2	$\perp$	(v_1, v_2)	$push(20)$
τ_{v_2}	(v_1, v_2)	10	(v_2, v_3)	$swap(11)$
	(v_1, v_2)	20	(v_2, v_3)	$swap(21)$
τ_{v_3}	(v_2, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_2, v_3)	21	(v_3, v_8)	$swap(22)$
	(v_7, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_7, v_3)	21	(v_3, v_8)	$swap(22)$
τ_{v_4}	(v_3, v_4)	12	out_1	pop
τ_{v_5}	(v_2, v_5)	40	(v_5, v_6)	pop
τ_{v_6}	(v_2, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	61	(v_6, v_7)	$swap(62)$
	(v_5, v_6)	71	(v_6, v_7)	$swap(72)$
τ_{v_7}	(v_6, v_7)	31	(v_7, v_3)	pop
	(v_6, v_7)	62	(v_7, v_3)	$swap(11)$
	(v_6, v_7)	72	(v_7, v_8)	$swap(22)$
τ_{v_8}	(v_3, v_8)	22	out_2	pop
	(v_7, v_8)	22	out_2	pop



local FFT	Out-I	In-Label	Out-I	op
τ_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$push(30)$
	(v_2, v_3)	21	(v_2, v_6)	$push(30)$
	(v_2, v_6)	30	(v_2, v_5)	$push(40)$
global FFT	**Out-I**	**In-Label**	**Out-I**	**op**
τ'_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$swap(61)$
	(v_2, v_3)	21	(v_2, v_6)	$swap(71)$
	(v_2, v_6)	61	(v_2, v_5)	$push(40)$
	(v_2, v_6)	71	(v_2, v_5)	$push(40)$

Router **configurations**
(Cisco, Juniper, etc.)

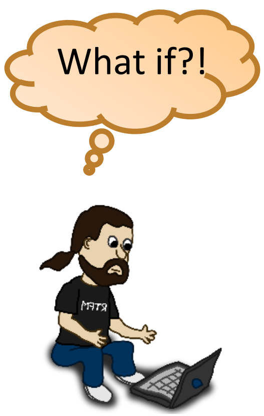


Formal methods?
Machine learning?

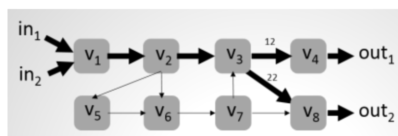
9

Opportunity: Fast reroute and robust networks with

Automation

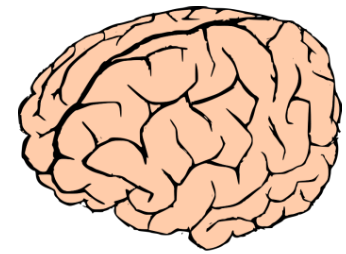


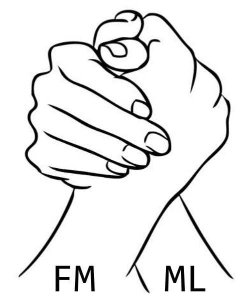
FT	In-I	In-Label	Out-I	op
τ_{v_1}	in_1	$\perp$	(v_1, v_2)	$push(10)$
	in_2	$\perp$	(v_1, v_2)	$push(20)$
τ_{v_2}	(v_1, v_2)	10	(v_2, v_3)	$swap(11)$
	(v_1, v_2)	20	(v_2, v_3)	$swap(21)$
τ_{v_3}	(v_2, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_2, v_3)	21	(v_3, v_8)	$swap(22)$
	(v_7, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_7, v_3)	21	(v_3, v_8)	$swap(22)$
τ_{v_4}	(v_3, v_4)	12	out_1	pop
τ_{v_5}	(v_2, v_5)	40	(v_5, v_6)	pop
τ_{v_6}	(v_2, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	61	(v_6, v_7)	$swap(62)$
	(v_5, v_6)	71	(v_6, v_7)	$swap(72)$
τ_{v_7}	(v_6, v_7)	31	(v_7, v_3)	pop
	(v_6, v_7)	62	(v_7, v_3)	$swap(11)$
	(v_6, v_7)	72	(v_7, v_8)	$swap(22)$
τ_{v_8}	(v_3, v_8)	22	out_2	pop
	(v_7, v_8)	22	out_2	pop



local FFT	Out-I	In-Label	Out-I	op
τ_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$push(30)$
	(v_2, v_3)	21	(v_2, v_6)	$push(30)$
	(v_2, v_6)	30	(v_2, v_5)	$push(40)$
global FFT	**Out-I**	**In-Label**	**Out-I**	**op**
τ'_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$swap(61)$
	(v_2, v_3)	21	(v_2, v_6)	$swap(71)$
	(v_2, v_6)	61	(v_2, v_5)	$push(40)$
	(v_2, v_6)	71	(v_2, v_5)	$push(40)$

Router configurations
(Cisco, Juniper, etc.)



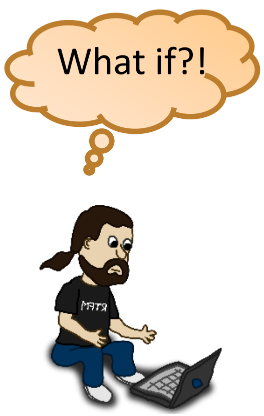


Both!

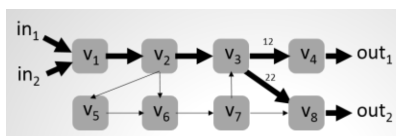
9

Example:

MPLS and Segment Routing

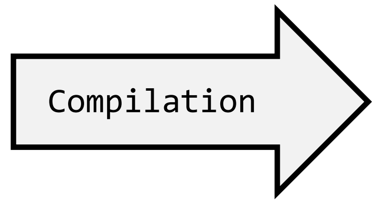


FT	In-I	In-Label	Out-I	op
τ_{v_1}	in_1	$\perp$	(v_1, v_2)	$push(10)$
	in_2	$\perp$	(v_1, v_2)	$push(20)$
τ_{v_2}	(v_1, v_2)	10	(v_2, v_3)	$swap(11)$
	(v_1, v_2)	20	(v_2, v_3)	$swap(21)$
τ_{v_3}	(v_2, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_2, v_3)	21	(v_3, v_8)	$swap(22)$
	(v_7, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_7, v_3)	21	(v_3, v_8)	$swap(22)$
τ_{v_4}	(v_3, v_4)	12	out_1	pop
τ_{v_5}	(v_2, v_5)	40	(v_5, v_6)	pop
τ_{v_6}	(v_2, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	61	(v_6, v_7)	$swap(62)$
	(v_5, v_6)	71	(v_6, v_7)	$swap(72)$
τ_{v_7}	(v_6, v_7)	31	(v_7, v_3)	pop
	(v_6, v_7)	62	(v_7, v_3)	$swap(11)$
	(v_6, v_7)	72	(v_7, v_8)	$swap(22)$
τ_{v_8}	(v_3, v_8)	22	out_2	pop
	(v_7, v_8)	22	out_2	pop

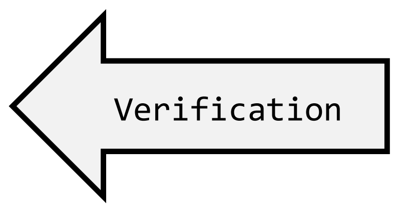


local FFT	Out-I	In-Label	Out-I	op
τ_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$push(30)$
	(v_2, v_3)	21	(v_2, v_6)	$push(30)$
	(v_2, v_6)	30	(v_2, v_5)	$push(40)$

global FFT	Out-I	In-Label	Out-I	op
τ'_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$swap(61)$
	(v_2, v_3)	21	(v_2, v_6)	$swap(71)$
	(v_2, v_6)	61	(v_2, v_5)	$push(40)$
	(v_2, v_6)	71	(v_2, v_5)	$push(40)$



$$
\begin{aligned}
pX &\Rightarrow qXX \\
pX &\Rightarrow qYX \\
qY &\Rightarrow rYY \\
rY &\Rightarrow r \\
rX &\Rightarrow pX
\end{aligned}
$$

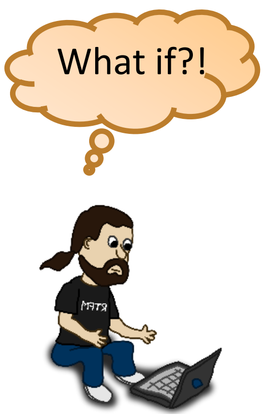


Formal language
which supports
automated analysis

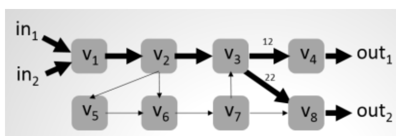
- ⇢ Verification fast: MPLS+SR networks are pushdown automata
- ⇢ Many alternatives: automata theory, binary decision diagrams (BDDs), games (e.g., Stackelberg, Petri nets), SMTs, ILPs …

Example:

MPLS and Segment Routing

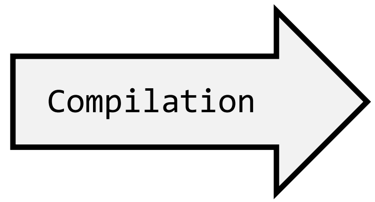


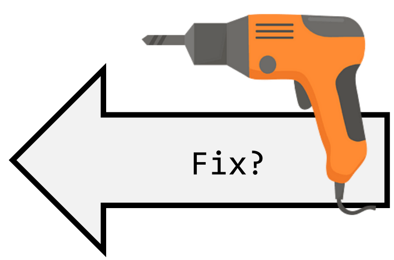
FT	In-I	In-Label	Out-I	op
τ_{v_1}	in_1	$\perp$	(v_1, v_2)	$push(10)$
	in_2	$\perp$	(v_1, v_2)	$push(20)$
τ_{v_2}	(v_1, v_2)	10	(v_2, v_3)	$swap(11)$
	(v_1, v_2)	20	(v_2, v_3)	$swap(21)$
τ_{v_3}	(v_2, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_2, v_3)	21	(v_3, v_8)	$swap(22)$
	(v_7, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_7, v_3)	21	(v_3, v_8)	$swap(22)$
τ_{v_4}	(v_3, v_4)	12	out_1	pop
τ_{v_5}	(v_2, v_5)	40	(v_5, v_6)	pop
τ_{v_6}	(v_2, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	61	(v_6, v_7)	$swap(62)$
	(v_5, v_6)	71	(v_6, v_7)	$swap(72)$
τ_{v_7}	(v_6, v_7)	31	(v_7, v_3)	pop
	(v_6, v_7)	62	(v_7, v_3)	$swap(11)$
	(v_6, v_7)	72	(v_7, v_8)	$swap(22)$
τ_{v_8}	(v_3, v_8)	22	out_2	pop
	(v_7, v_8)	22	out_2	pop



local FFT	Out-I	In-Label	Out-I	op
τ_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$push(30)$
	(v_2, v_3)	21	(v_2, v_6)	$push(30)$
	(v_2, v_6)	30	(v_2, v_5)	$push(40)$

global FFT	Out-I	In-Label	Out-I	op
τ'_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$swap(61)$
	(v_2, v_3)	21	(v_2, v_6)	$swap(71)$
	(v_2, v_6)	61	(v_2, v_5)	$push(40)$
	(v_2, v_6)	71	(v_2, v_5)	$push(40)$





$$\mathrm{pX} \Rightarrow \mathrm{qXX}$$
$$pX \Rightarrow qYX$$
$$qY \Rightarrow rYY$$
$$rY \Rightarrow r$$
$$rX \Rightarrow pX$$

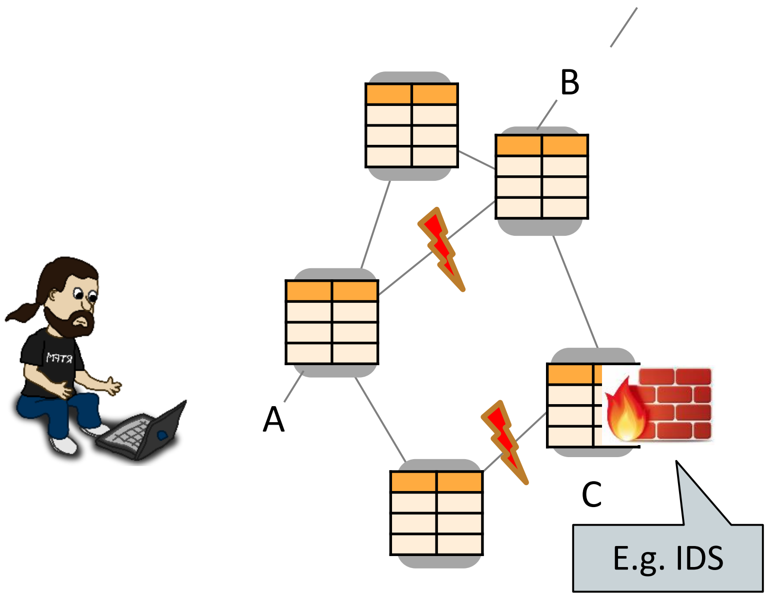
Formal language
which supports
automated analysis

⇢ But synthesis slow: a case for machine learning?

Fast Synthesis: FM+ML

- ***Ideally ML+FM***: guarantees from formal methods, performance from ML
- For example: synthesize with ML then verify with formal methods
- Examples: DeepMPLS, DeepBGP, …
- ***Self-driving networks!***

Can cover many policies!

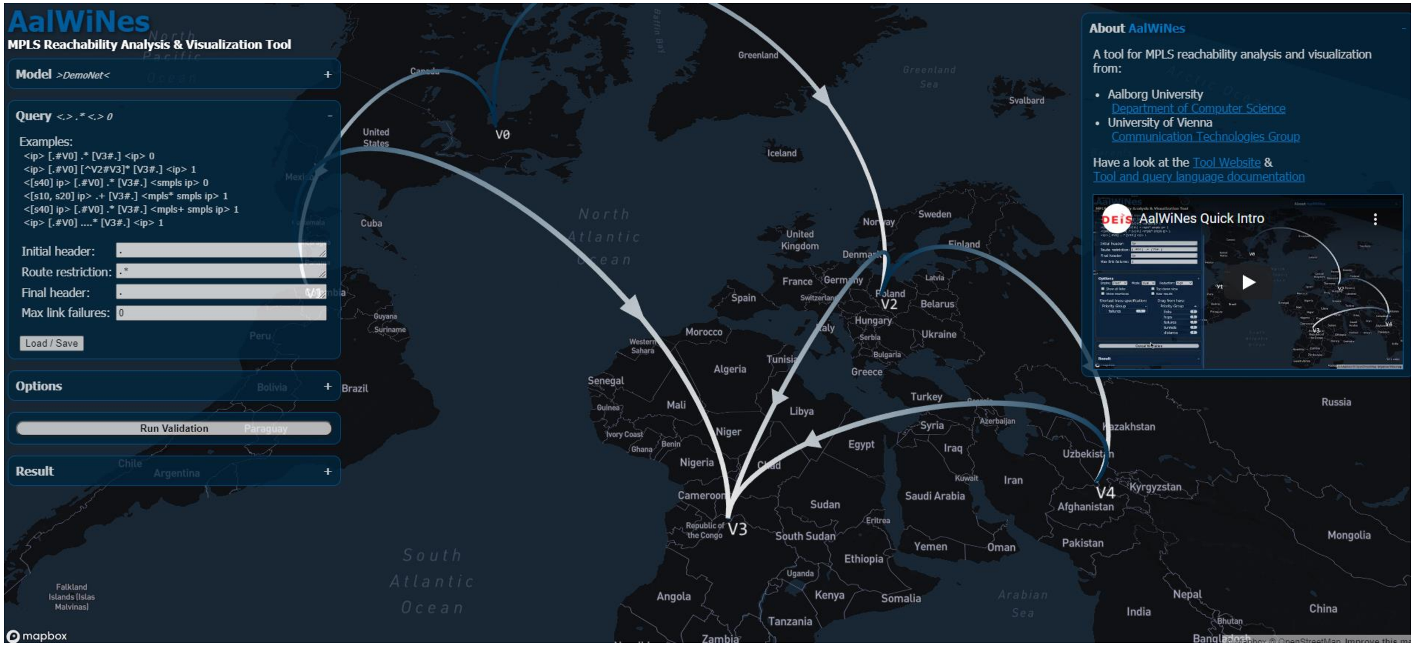


Sysadmin responsible for:

- **Reachability:** Can traffic from ingress port A reach egress port B?
- **Loop-freedom:** Are the routes implied by the forwarding rules loop-free?
- **Policy:** Is it ensured that traffic from A to B never goes via C?
- **Waypoint enforcement:** Is it ensured that traffic from A to B is always routed via a node C (e.g., intrusion detection system or a firewall)?

… and everything under multiple failures!

Example: AalWiNes Tool



Tool: https://demo.aalwines.cs.aau.dk/
Youtube: https://www.youtube.com/watch?v=mvXAn9i7_Q0

Summary

- ⇢ Opportunity: *adaptable networks* and *structure* in demand
- ⇢ Opportunity: *AI/ML* for performance and *formal methods* for dependability
- ⇢ Enables *self-driving networks*
- ⇢ Requires: models and automated, computer-driven designs
- ⇢ Great research opportunities ahead!

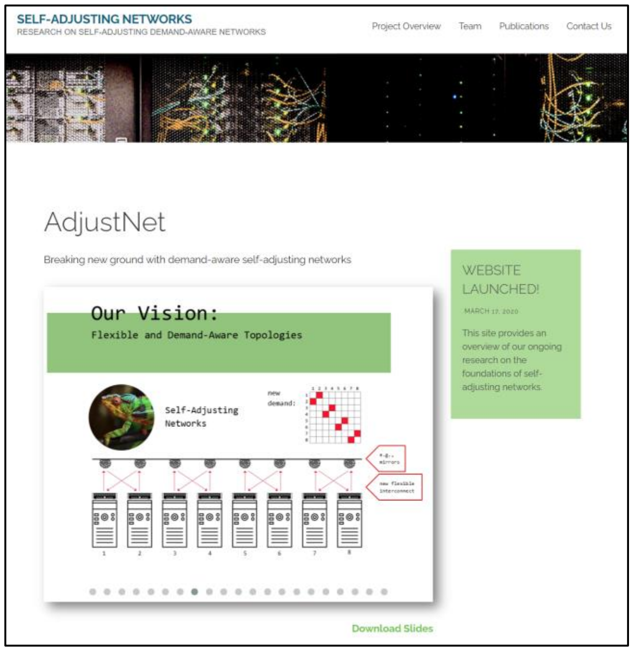
Online Video Course

YouTube Interview & CACM

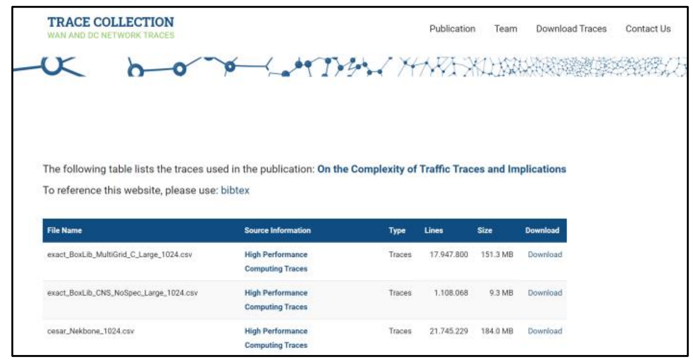
Check out our **YouTube interviews**
on Reconfigurable Datacenter Networks:

[Revolutionizing Datacenter Networks via Reconfigurable Topologies](#)
Chen Avin and Stefan Schmid.
Communications of the ACM (**CACM**), 2025.
Watch here: https://www.youtube.com/@self-adjusting-networks-course

Websites



http://self-adjusting.net/
Project website



https://trace-collection.net/
Trace collection website

June Issue CACM'25

Revolutionizing Datacenter Networks via Reconfigurable Topologies

CHEN AVIN, is a Professor at Ben-Gurion University of the Negev, Beersheva, Israel
STEFAN SCHMID, is a Professor at TU Berlin, Berlin, Germany

With the popularity of cloud computing and data-intensive applications such as machine learning, datacenter networks have become a critical infrastructure for our digital society. Given the explosive growth of datacenter traffic and the slowdown of Moore's law, significant efforts have been made to improve datacenter network performance over the last decade. A particularly innovative solution is reconfigurable datacenter networks (RDCNs): datacenter networks whose topologies dynamically change over time, in either a demand-oblivious or a demand-aware manner. Such dynamic topologies are enabled by recent optical switching technologies and stand in stark contrast to state-of-the-art datacenter network topologies, which are fixed and oblivious to the actual traffic demand. In particular, reconfigurable demand-aware and "self-adjusting" datacenter networks are motivated empirically by the significant spatial and temporal structures observed in datacenter communication traffic. This paper presents an overview of reconfigurable datacenter networks. In particular, we discuss the motivation for such reconfigurable architectures, review the technological enablers, and present a taxonomy that classifies the design space into two dimensions: static vs. dynamic and demand-oblivious vs. demand-aware. We further present a formal model and discuss related research challenges. Our article comes with complementary video interviews in which three leading experts, Manya Ghobadi, Amin Vahdat, and George Papen, share with us their perspectives on reconfigurable datacenter networks.

KEY INSIGHTS

- **Datacenter networks have become a critical infrastructure** for our digital society, serving explosively growing communication traffic.
- **Reconfigurable datacenter networks (RDCNs)** which can adapt their topology dynamically, based on innovative **optical switching technologies**, bear the potential to improve datacenter network performance, and to simplify datacenter planning and operations.
- Demand-aware dynamic topologies are particularly interesting because of the **significant spatial and temporal structures** observed in real-world traffic, e.g., related to distributed machine learning.
- The study of RDCNs and self-adjusting networks raises many **novel technological and research challenges** related to their design, control, and performance.

References (1)

On the Complexity of Traffic Traces and Implications
Chen Avin, Manya Ghobadi, Chen Griner, and Stefan Schmid.
ACM **SIGMETRICS** and ACM Performance Evaluation Review (**PER**), Boston, Massachusetts, USA, June 2020.

Toward Demand-Aware Networking: A Theory for Self-Adjusting Networks (Editorial)
Chen Avin and Stefan Schmid.
ACM SIGCOMM Computer Communication Review (**CCR**), October 2018.

Revolutionizing Datacenter Networks via Reconfigurable Topologies
Chen Avin and Stefan Schmid.
Communications of the ACM (**CACM**), 2025.

Cerberus: The Power of Choices in Datacenter Topology Design (A Throughput Perspective)
Chen Griner, Johannes Zerwas, Andreas Blenk, Manya Ghobadi, Stefan Schmid, and Chen Avin.
ACM **SIGMETRICS** and ACM Performance Evaluation Review (**PER**), Mumbai, India, June 2022.

AalWiNes: A Fast and Quantitative What-If Analysis Tool for MPLS Networks
Peter Gjøl Jensen, Morten Konggaard, Dan Kristiansen, Stefan Schmid, Bernhard Clemens Schrenk, and Jiri Srba.
16th ACM International Conference on emerging Networking EXperiments and Technologies (**CoNEXT**), Barcelona, Spain, December 2020.

Latte: Improving the Latency of Transiently Consistent Network Update Schedules
Mark Glavind, Niels Christensen, Jiri Srba, and Stefan Schmid.
38th International Symposium on Computer Performance, Modeling, Measurements and Evaluation (**PERFORMANCE**) and ACM Performance Evaluation Review (**PER**), Milan, Italy, November 2020.

Model-Based Insights on the Performance, Fairness, and Stability of BBR (IRTF Applied Networking Research Prize)
Simon Scherrer, Markus Legner, Adrian Perrig, and Stefan Schmid.
ACM Internet Measurement Conference (**IMC**), Nice, France, October 2022.

Credence: Augmenting Datacenter Switch Buffer Sharing with ML Predictions
Vamsi Addanki, Maciej Pacut, and Stefan Schmid.
21st USENIX Symposium on Networked Systems Design and Implementation (**NSDI**), Santa Clara, California, USA, April 2024.

References (2)

Mars: Near-Optimal Throughput with Shallow Buffers in Reconfigurable Datacenter Networks
Vamsi Addanki, Chen Avin, and Stefan Schmid.
ACM **SIGMETRICS** and ACM Performance Evaluation Review (**PER**), Orlando, Florida, USA, June 2023.

Duo: A High-Throughput Reconfigurable Datacenter Network Using Local Routing and Control
Johannes Zerwas, Csaba Györgyi, Andreas Blenk, Stefan Schmid, and Chen Avin.
ACM **SIGMETRICS** and ACM Performance Evaluation Review (**PER**), Orlando, Florida, USA, June 2023.

SyPer: Synthesis of Perfectly Resilient Local Fast Rerouting Rules for Highly Dependable Networks
Csaba Györgyi, Kim G. Larsen, Stefan Schmid, and Jiri Srba.
IEEE Conference on Computer Communications (**INFOCOM**), Vancouver, Canada, May 2024.

Demand-Aware Network Design with Minimal Congestion and Route Lengths
Chen Avin, Kaushik Mondal, and Stefan Schmid.
IEEE/ACM Transactions on Networking (**TON**), 2022.

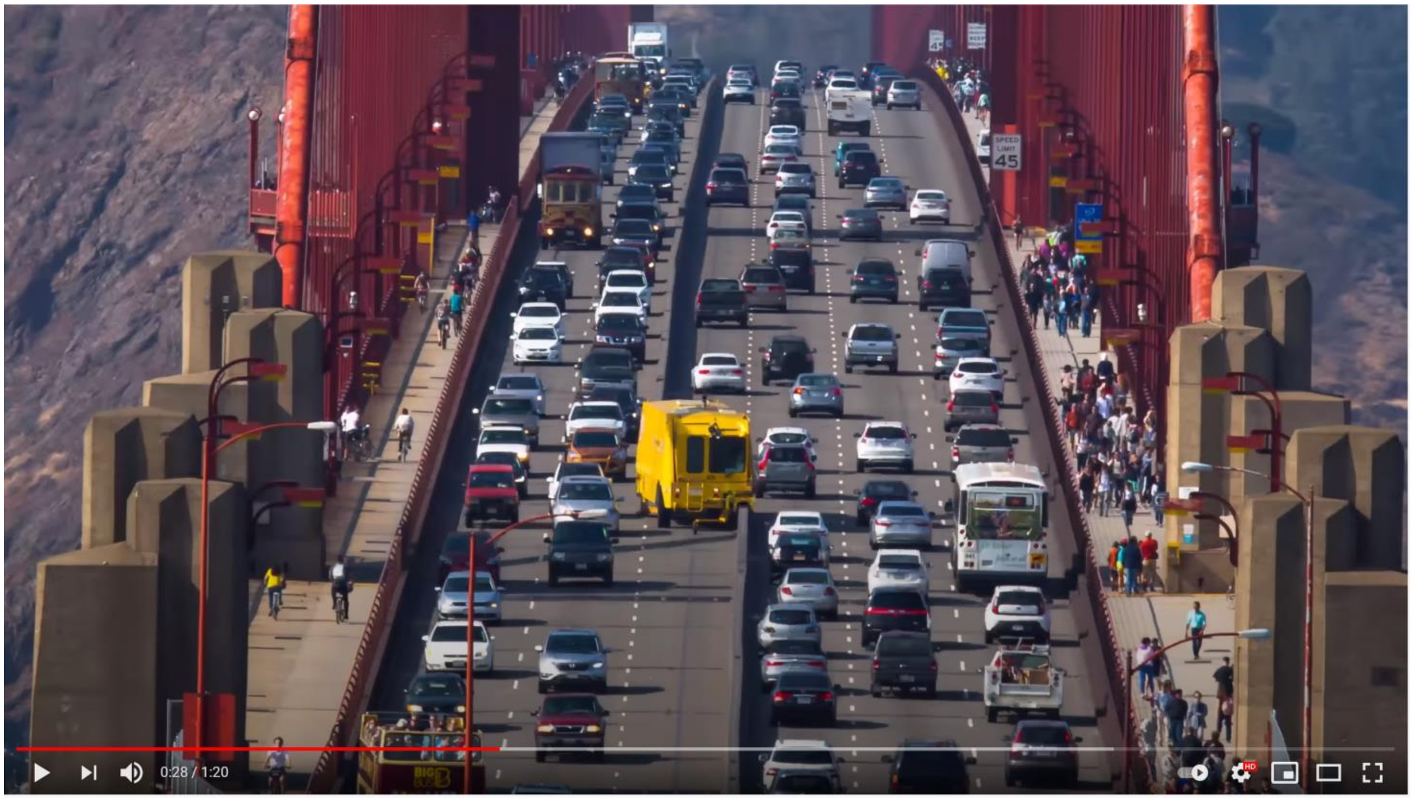
A Survey of Reconfigurable Optical Networks
Matthew Nance Hall, Klaus-Tycho Foerster, Stefan Schmid, and Ramakrishnan Durairajan.
Optical Switching and Networking (**OSN**), Elsevier, 2021.

SplayNet: Towards Locally Self-Adjusting Networks
Stefan Schmid, Chen Avin, Christian Scheideler, Michael Borokhovich, Bernhard Haeupler, and Zvi Lotker.
IEEE/ACM Transactions on Networking (**TON**), Volume 24, Issue 3, 2016.

TCP's Third Eye: Leveraging eBPF for Telemetry-Powered Congestion Control
Jörn-Thorben Hinz, Vamsi Addanki, Csaba Györgyi, Theo Jepsen, and Stefan Schmid.
SIGCOMM Workshop on eBPF and Kernel Extensions (**eBPF**), Columbia University, New York City, New York, USA, September 2023.

PowerTCP: Pushing the Performance Limits of Datacenter Networks
Vamsi Addanki, Oliver Michel, and Stefan Schmid.
19th USENIX Symposium on Networked Systems Design and Implementation (**NSDI**), Renton, Washington, USA, April 2022.
.

Questions?



Slides available here: