Demand-Aware and Self-Adjusting Networks

Stefan Schmid (University of Vienna)

Our research vision:
Self-* networks!

Self-observing, self-adjusting, self-repairing, "self-driving", …

Enabler of Our Vision: Flexibilities

Passau, Germany

Inn, Donau, Ilz

Enabler of Our Vision: Flexibilities

Passau, Germany

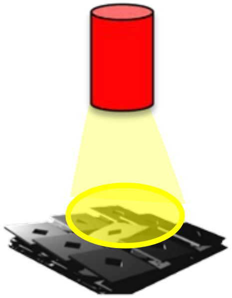
Inn, Donau, Ilz

Enabler of Our Vision: Flexibilities

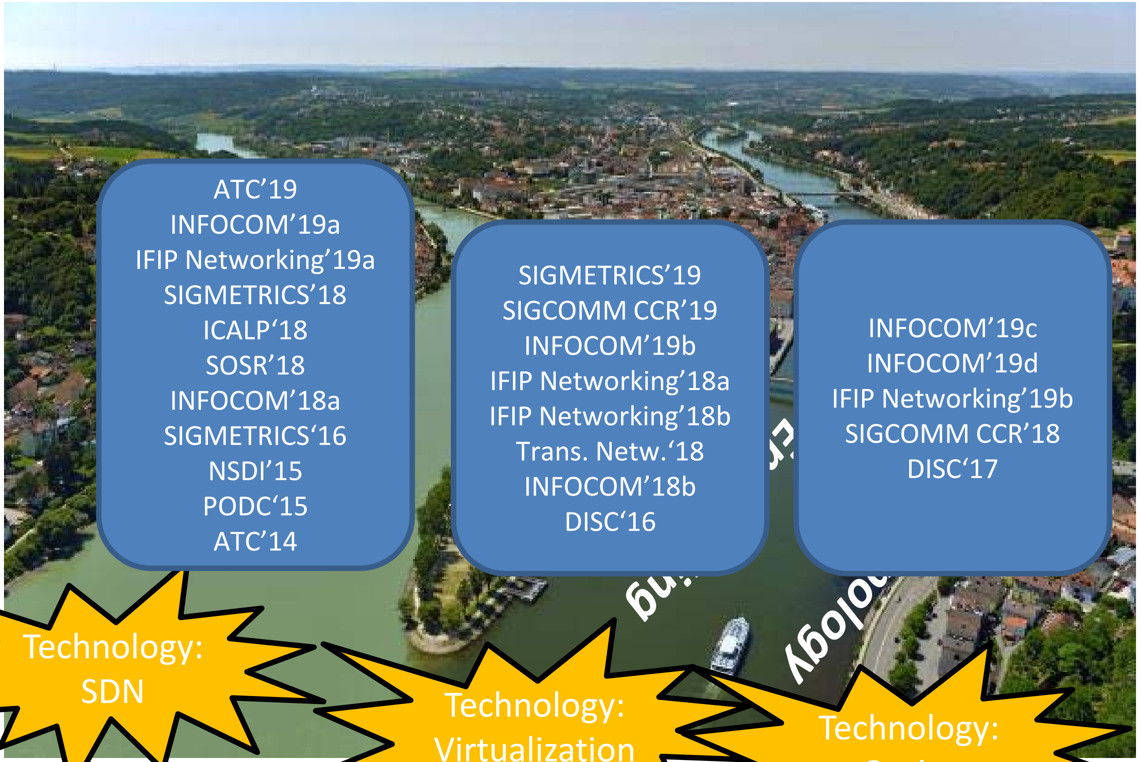
Enabler of Our Vision: Flexibilities

sa… …rmany

…nn, Donau, Ilz



Enabler of Our Vision: Flexibilities



ATC'19
INFOCOM'19a
IFIP Networking'19a
SIGMETRICS'18
ICALP'18
SOSR'18
INFOCOM'18a
SIGMETRICS'16
NSDI'15
PODC'15
ATC'14

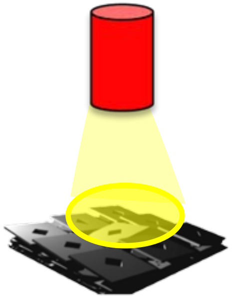
SIGMETRICS'19
SIGCOMM CCR'19
INFOCOM'19b
IFIP Networking'18a
IFIP Networking'18b
Trans. Netw.'18
INFOCOM'18b
DISC'16

INFOCOM'19c
INFOCOM'19d
IFIP Networking'19b
SIGCOMM CCR'18
DISC'17

Technology: SDN

Technology: Virtualization

Technology: Optics



Rewinding the clock of the Internet…

Shortest path routing only

Indirect control: via weights only

Proprietary, blackbox implementations

Difficult and slow innovation

Kudos to: Pedro Casas

Roadmap

- Opportunities of self-* networks
  - Example 1: Demand-aware, self-adjusting networks
  - Example 2: Self-repairing networks
- Challenges of desinging self-* networks

Roadmap

- Opportunities of self-* networks
  - **Example 1: Demand-aware, self-adjusting networks**
  - Example 2: Self-repairing networks
- Challenges of desinging self-* networks

Why Demand-Aware…?

Why Demand-Aware…?

1. Easier to collect **data** (SDN, telemetry)

2. **Flexibilities**: possible to adapt

Why Demand-Aware…?

1. Easier to collect **data** (SDN, telemetry)

2. **Flexibilities**: possible to adapt

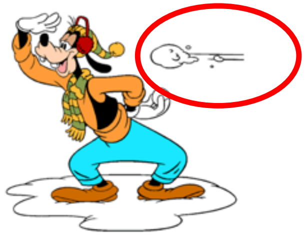
3. Demand has **structure**!

Why Demand-Aware…?

Alternative: demand-oblivious networks

Why Demand-Aware…?

Alternative: demand-oblivious networks



„Demand has structure“

“less than 1% of the rack pairs account for 80% of the total traffic”

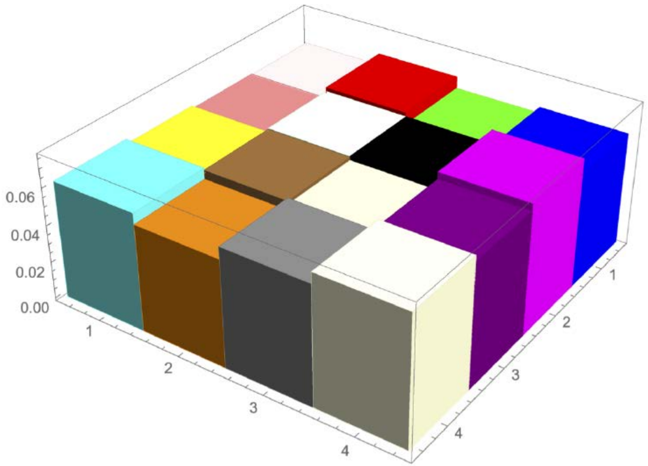
“only a few ToR switches are hot and most of their traffic goes to a few other ToRs”

“over 90% bytes in elephant flows”

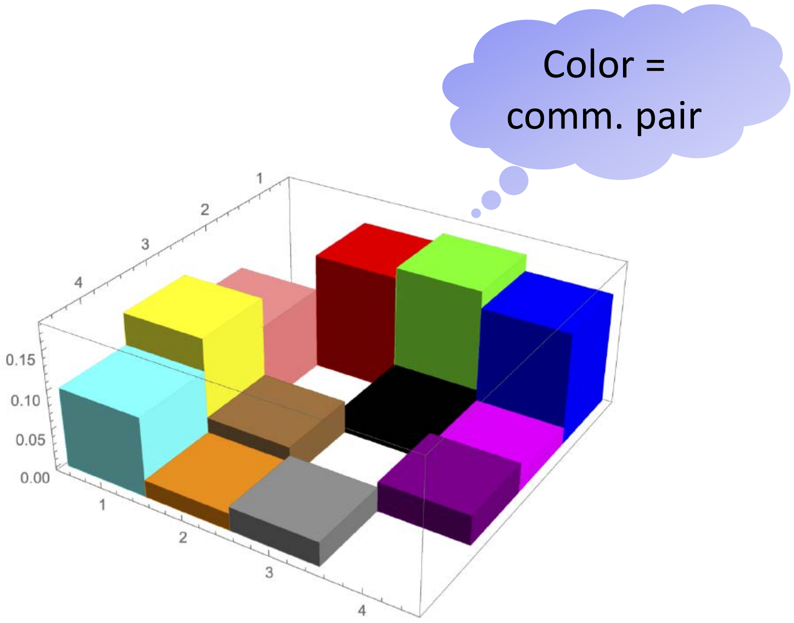
ProjecToR @ SIGCOMM 2016
Understanding Data Center Traffic Characteristics @ WREN 2009

Intuitive but: how to measure and quantify structure? We lack metrics!

Dimension 1: Non-Temporal Structure



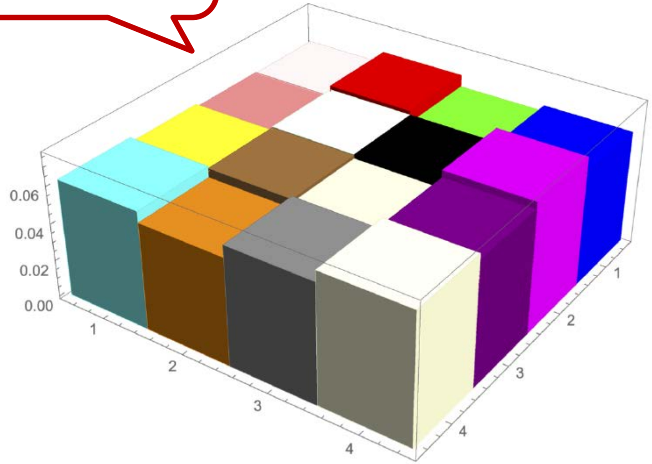
VS



Traffic matrix of two different **distributed ML** applications (GPU-to-GPU):

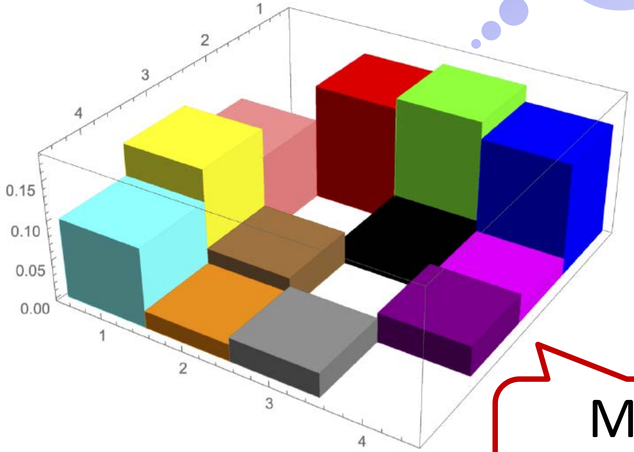
Dimension 1: Non-Temporal Structure

More random



VS

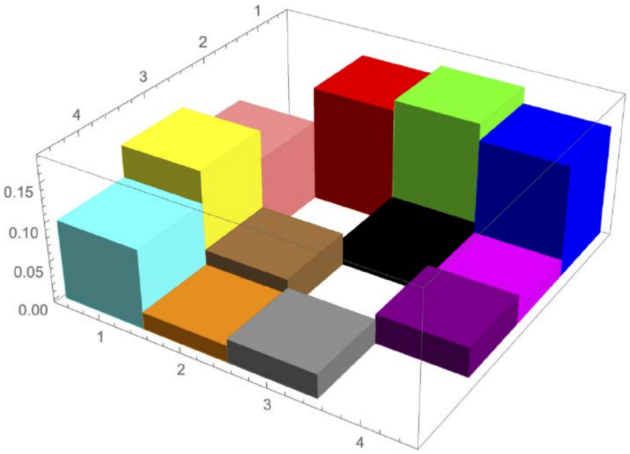
Color = comm. pair

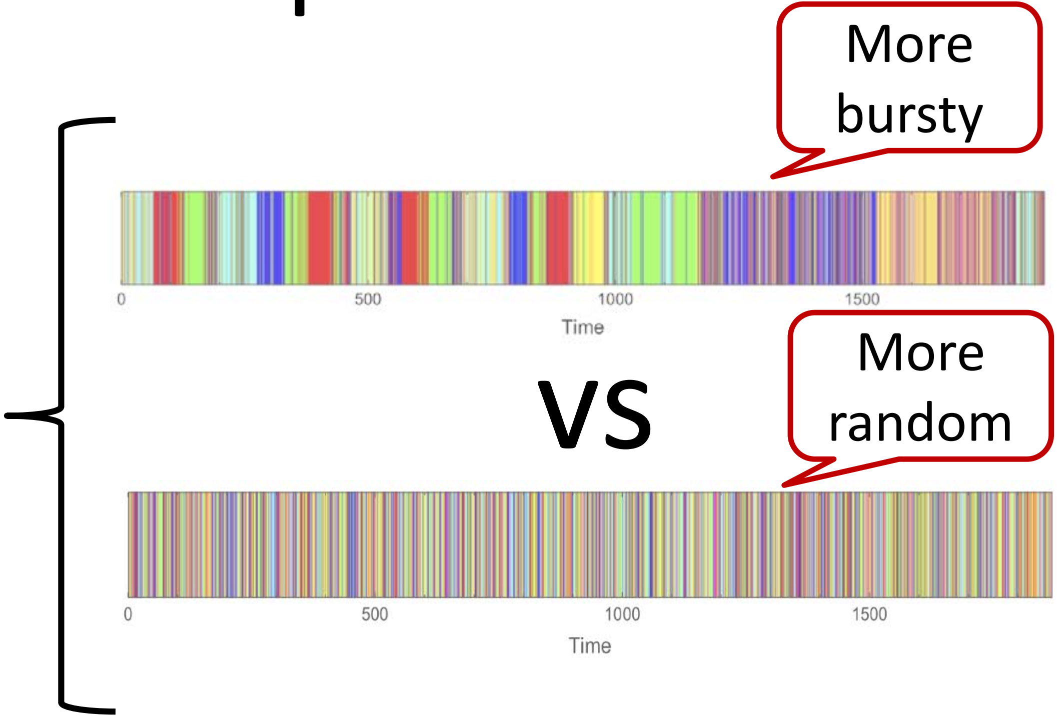


More structure

Traffic matrix of two different **distributed ML** applications (GPU-to-GPU):

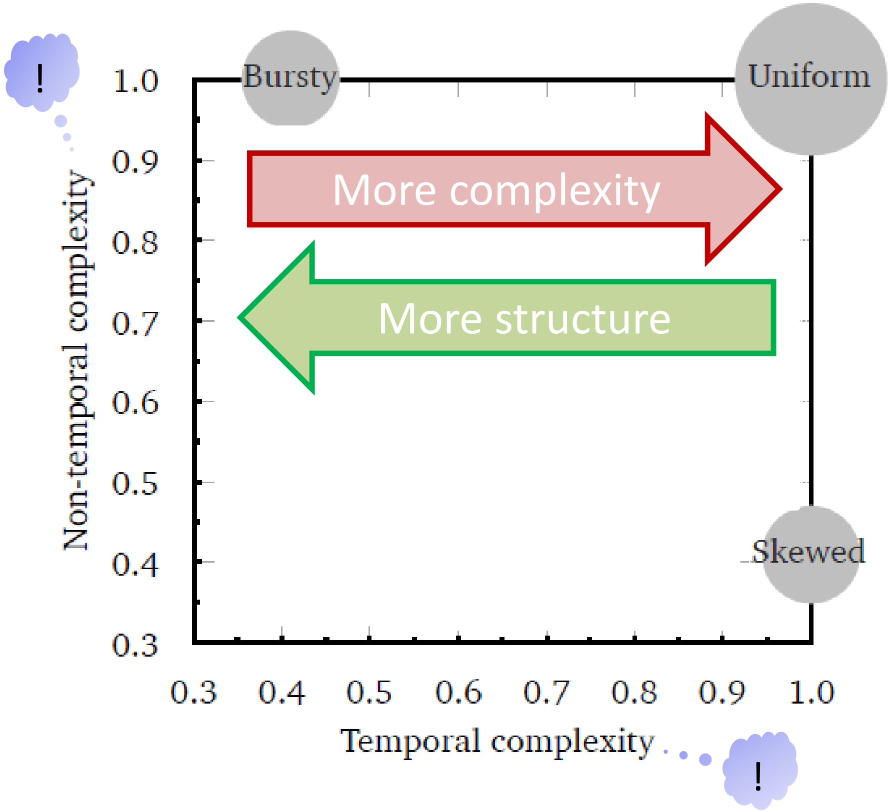
Dimension 2: Temporal Structure





Two different ways to generate *same traffic matrix* (same non-temporal structure)

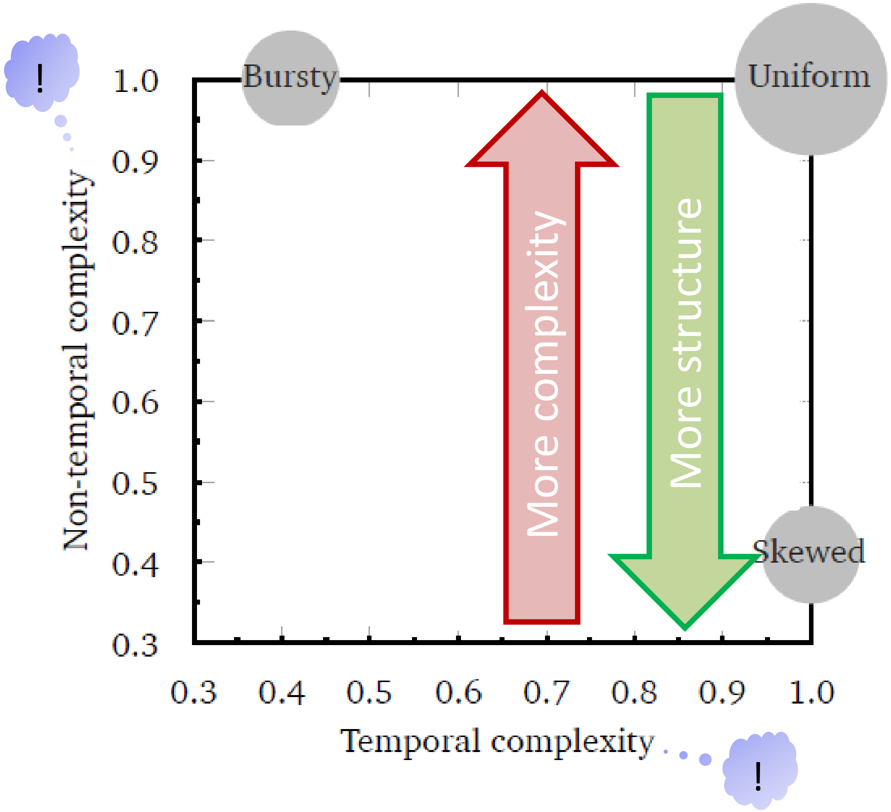
An Entropy Approach: The Complexity Map



Complexity Map: Entropy („complexity“) of traffic traces.

Measuring the Complexity of Packet Traces.
Avin, Ghobadi, Griner, Schmid. ArXiv 2019.

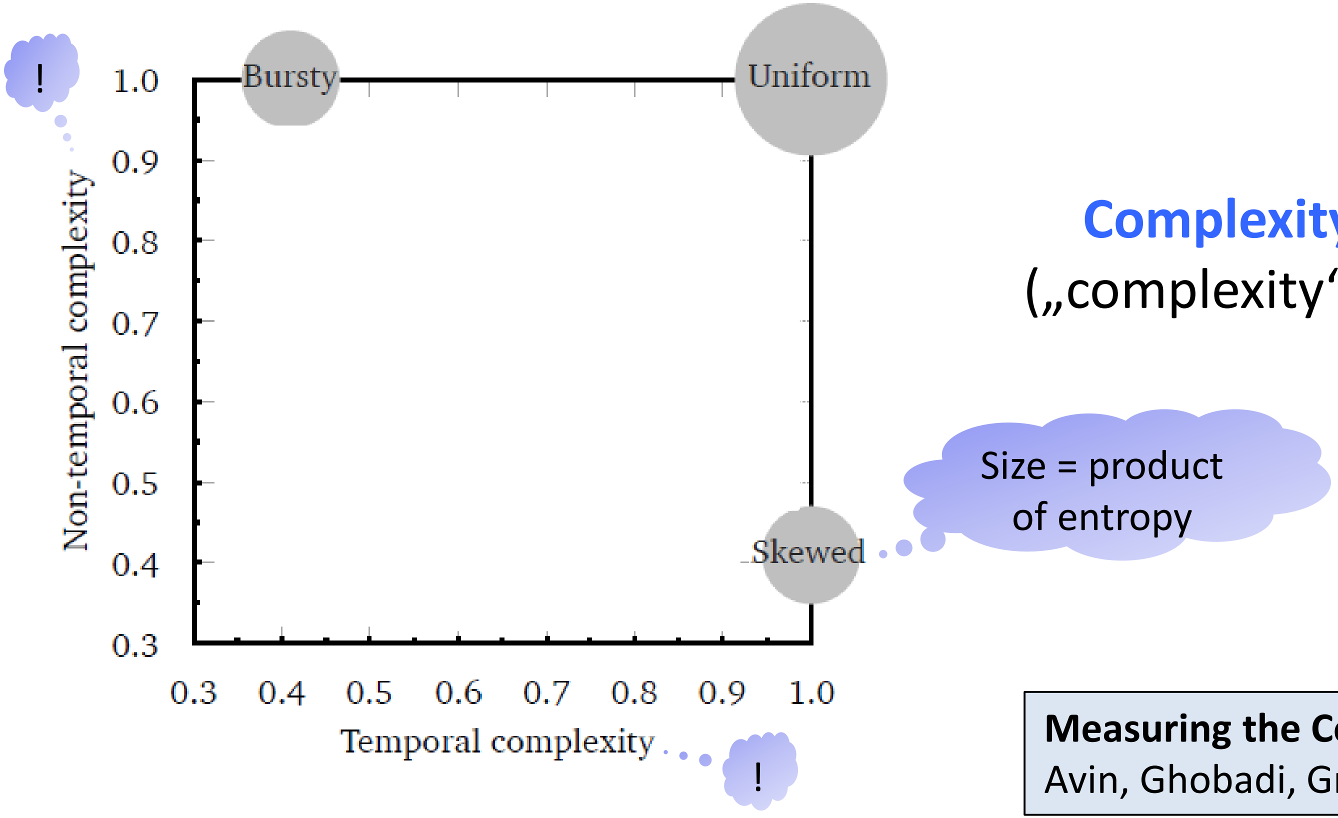
An Entropy Approach: The Complexity Map



Complexity Map: Entropy („complexity“) of traffic traces.

Measuring the Complexity of Packet Traces.
Avin, Ghobadi, Griner, Schmid. ArXiv 2019.

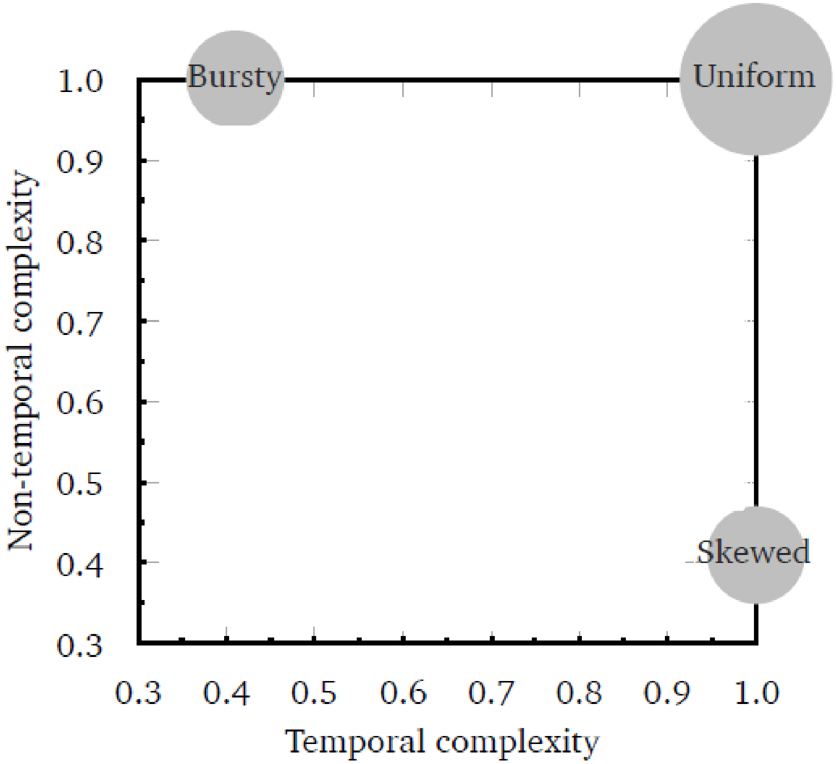
An Entropy Approach: The Complexity Map



Complexity Map: Entropy („complexity“) of traffic traces.

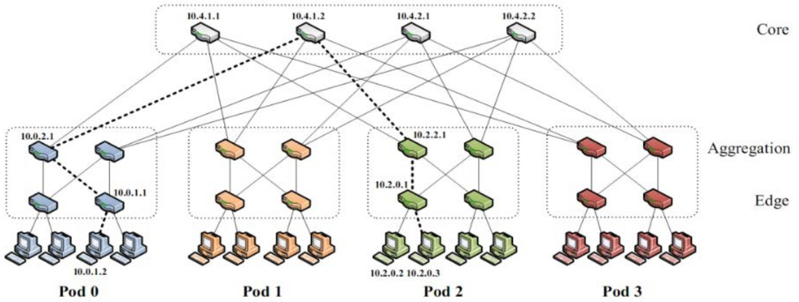
Measuring the Complexity of Packet Traces.
Avin, Ghobadi, Griner, Schmid. ArXiv 2019.

An Entropy Approach: The Complexity Map

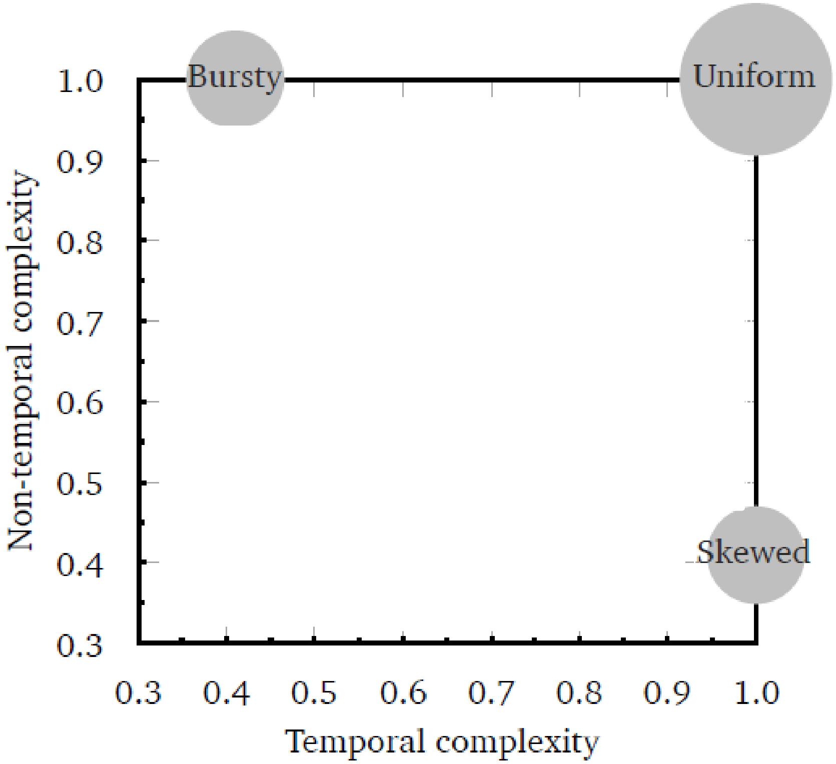


Uniform: Today's datacenters

- Traditional networks are optimized ***for the "worst-case"*** (**all-to-all** communication traffic)
- Example, fat-tree topologies: provide **full bisection bandwidth**

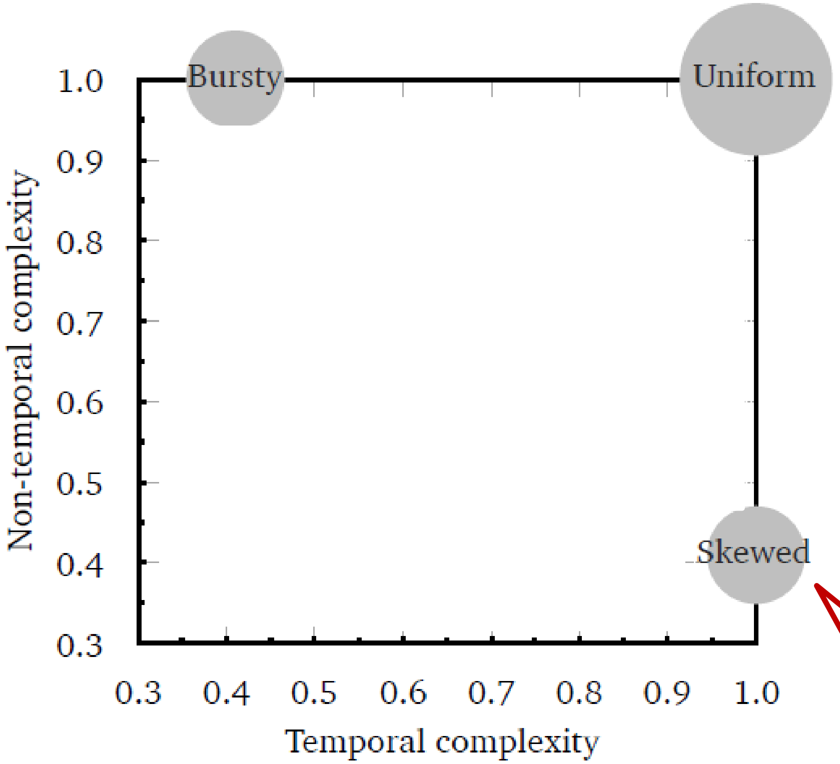


An Entropy Approach: The Complexity Map



Good in the worst case ***but***: cannot leverage different **temporal** and **non-temporal** structures of traffic traces!

An Entropy Approach: The Complexity Map

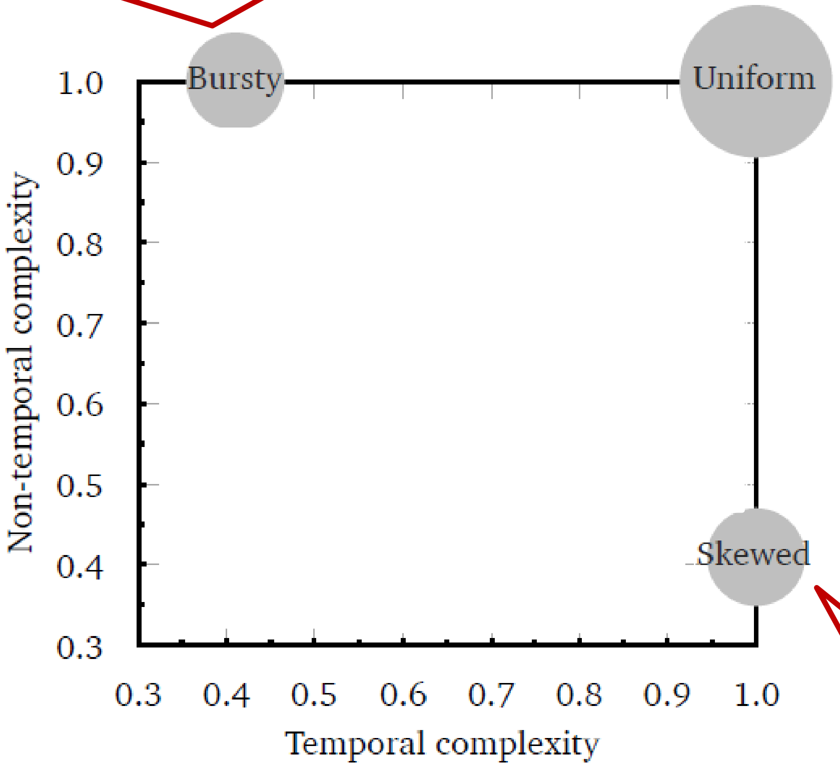


Good in the worst case ***but***: cannot leverage different **temporal** and **non-temporal** structures of traffic traces!

Non-temporal structure could be exploited already with ***static demand-aware networks***!

...ch: The Complexity Map

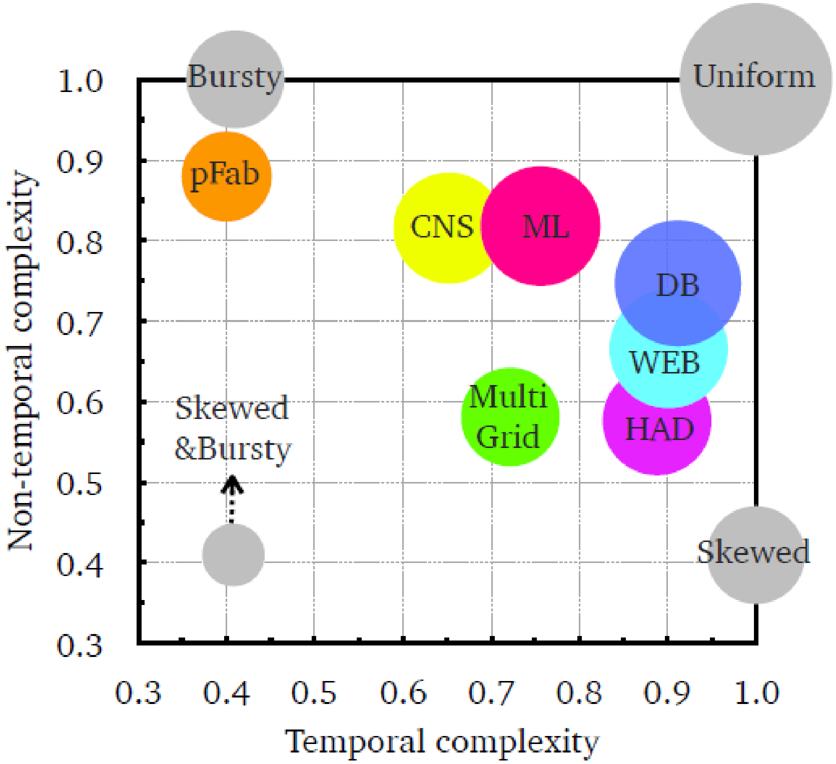
To exploit **temporal** structure, need ***adaptive demand-aware ("self-adjusting") networks***.



Good in the worst case ***but***: cannot leverage different **temporal** and **non-temporal** structures of traffic traces!

Non-temporal structure could be exploited already with ***static demand-aware networks***!

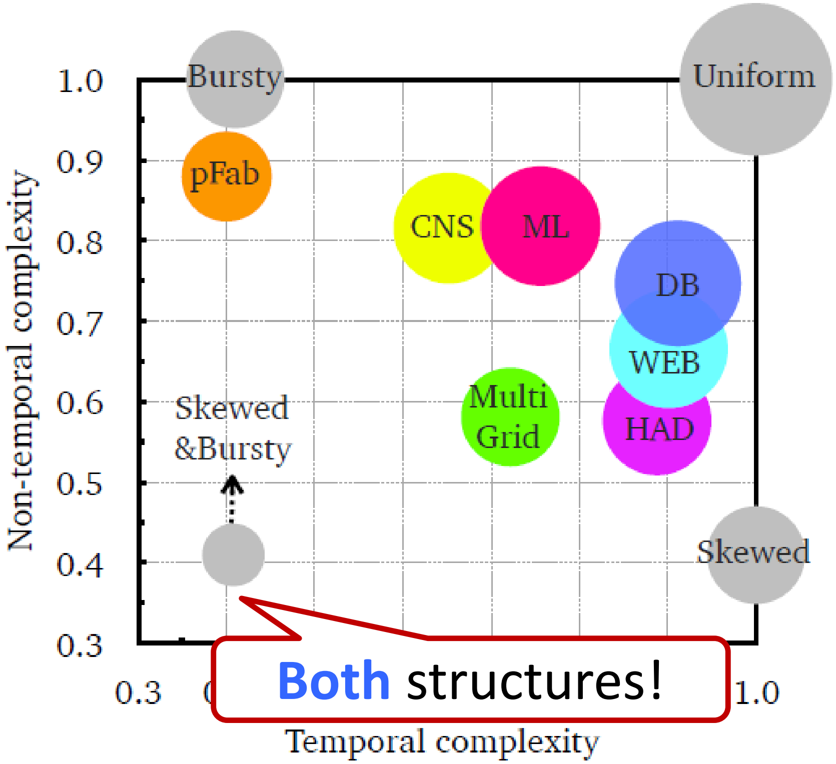
An Entropy Approach: The Complexity Map



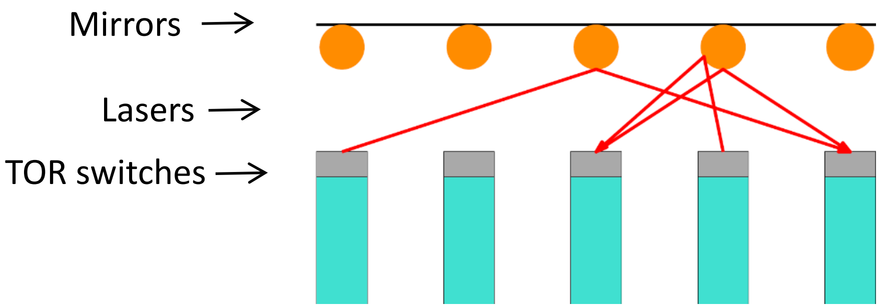
Observation: different applications feature quite significant (and different!) **temporal** and **non-temporal** structures.

- **Facebook** clusters: DB, WEB, HAD
- **HPC** workloads: CNS, Multigrid
- Distributed **Machine Learning** (ML)
- Synthetic traces like **pFabric**

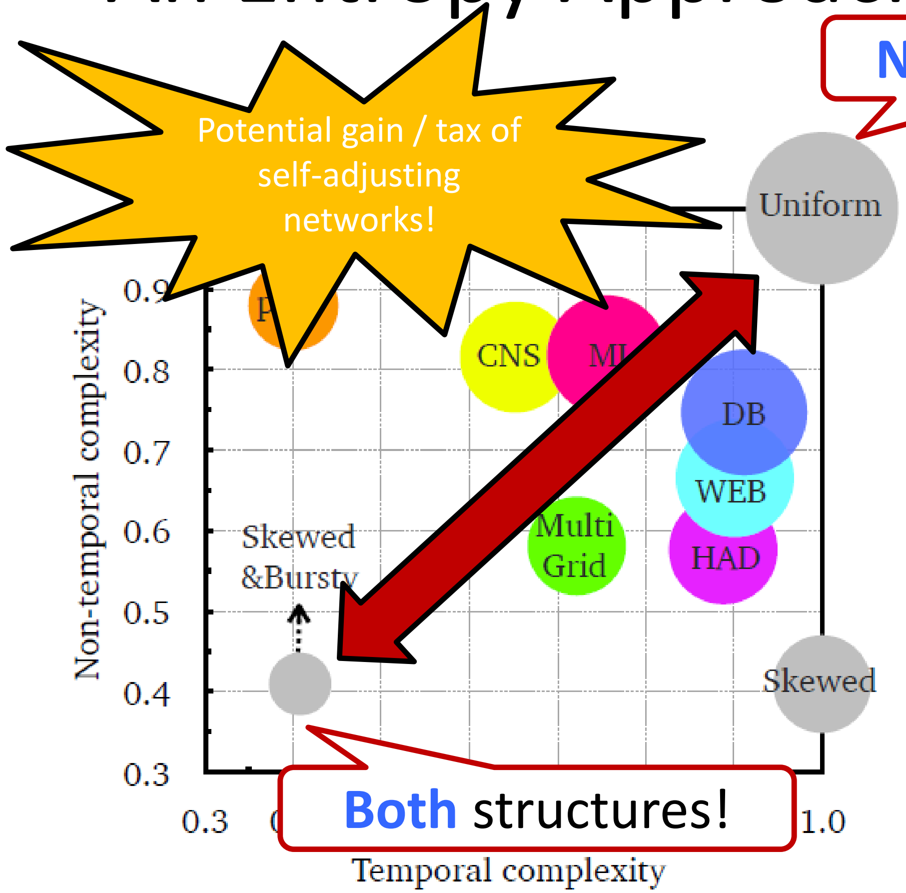
An Entropy Approach: The Complexity Map



Goal: Design **self-adjusting networks** which leverage ***both*** dimensions of structure!

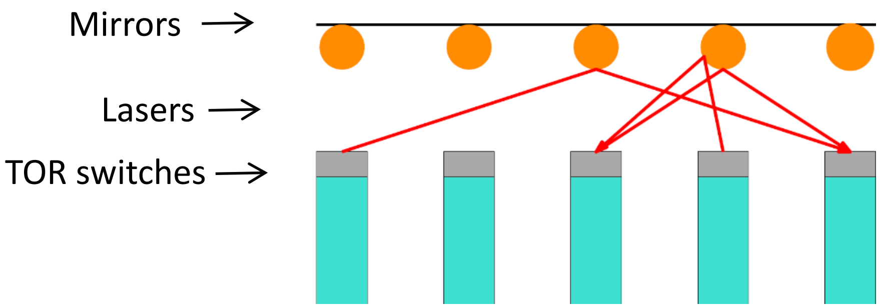


An Entropy Approach: The Complexity Map



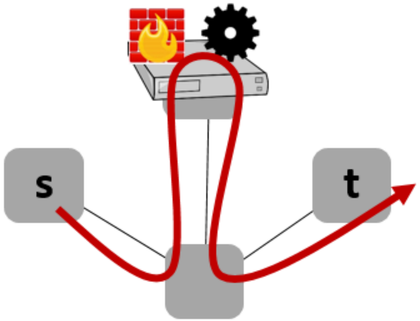
No structure!

Goal: Design **self-adjusting networks** which leverage ***both*** dimensions of structure!

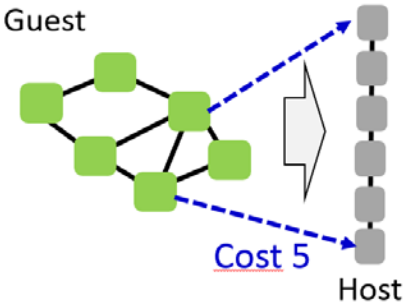


Algorithms to Exploit Structure

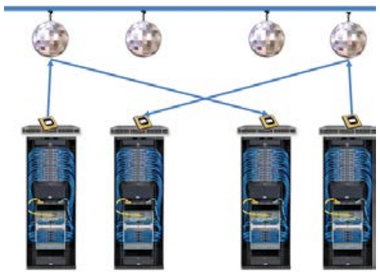
We are mainly interested in **online algorithms** (with *provable guarantees*: competitive ratio)



Online **admission control** and **routing** (***joint optimization***: placement and routing)

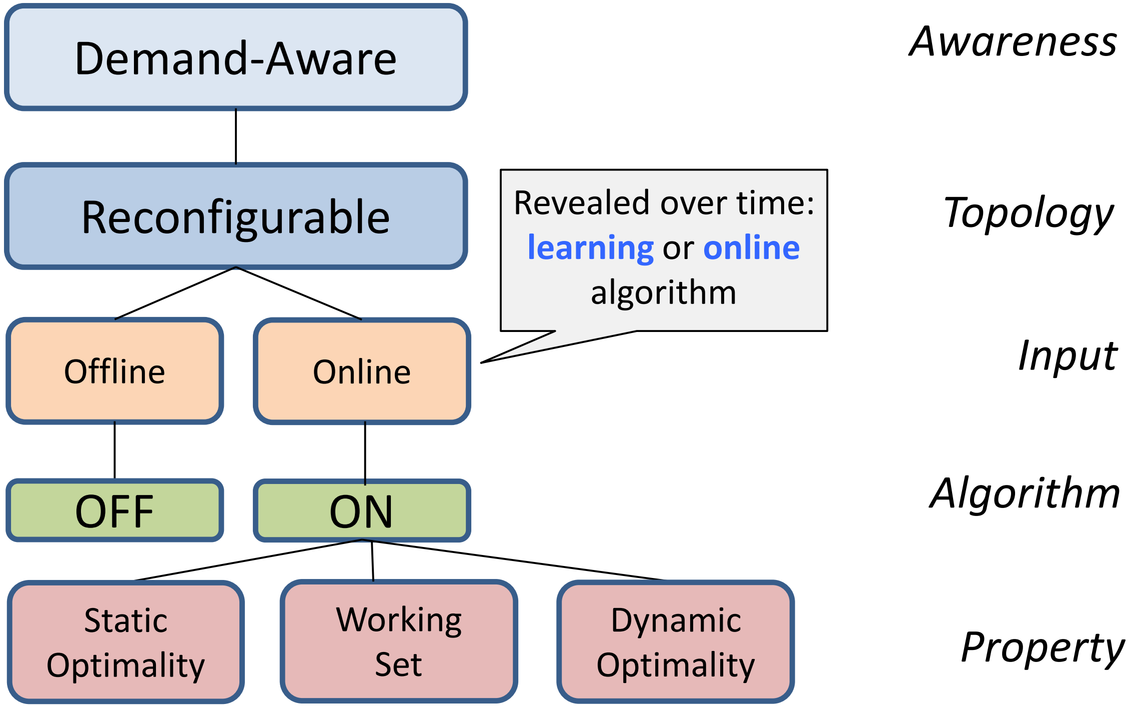


Virtual network embedding (slicing) and demand-aware reconfiguration/***migration***



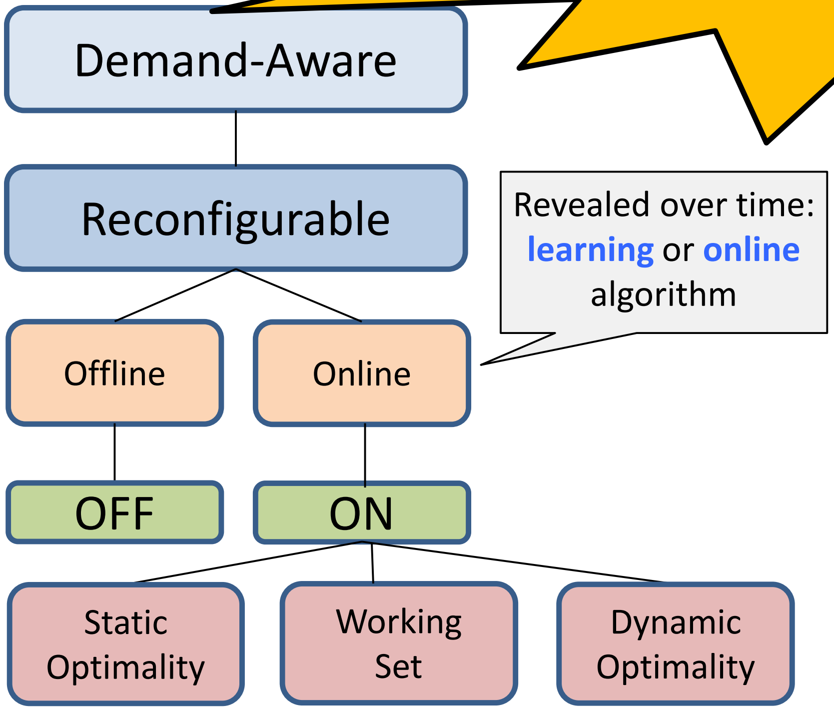
Topology design: **Graph spanners**

A Taxonomy: Reconfigurable Networks



A Taxonomy: Reconfigurable Networks

Structure does not imply predictable!



Aw

Topology

Input

Algorithm

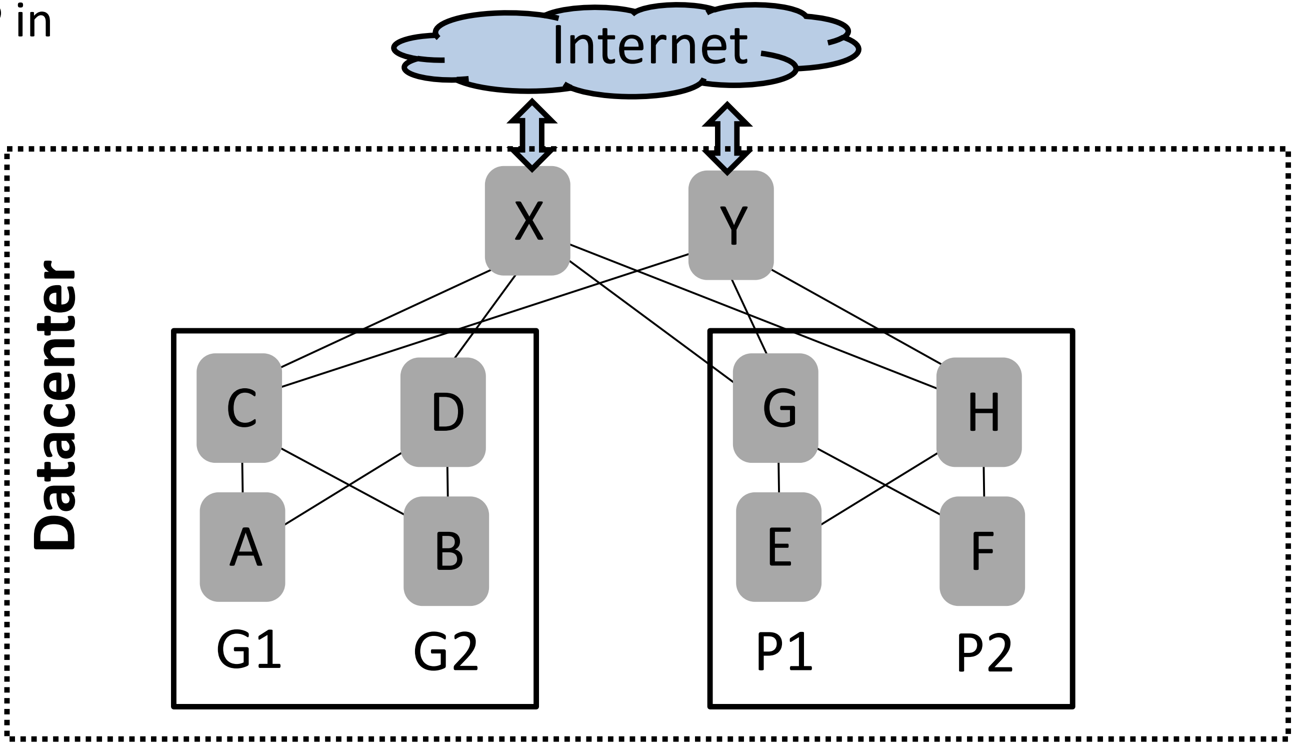
Property

Roadmap

- Opportunities of self-* networks
  - Example 1: Demand-aware, self-adjusting networks ✓
  - **Example 2: Self-repairing networks**
- Challenges of desinging self-* networks

Reasoning About Failures is Hard for Humans

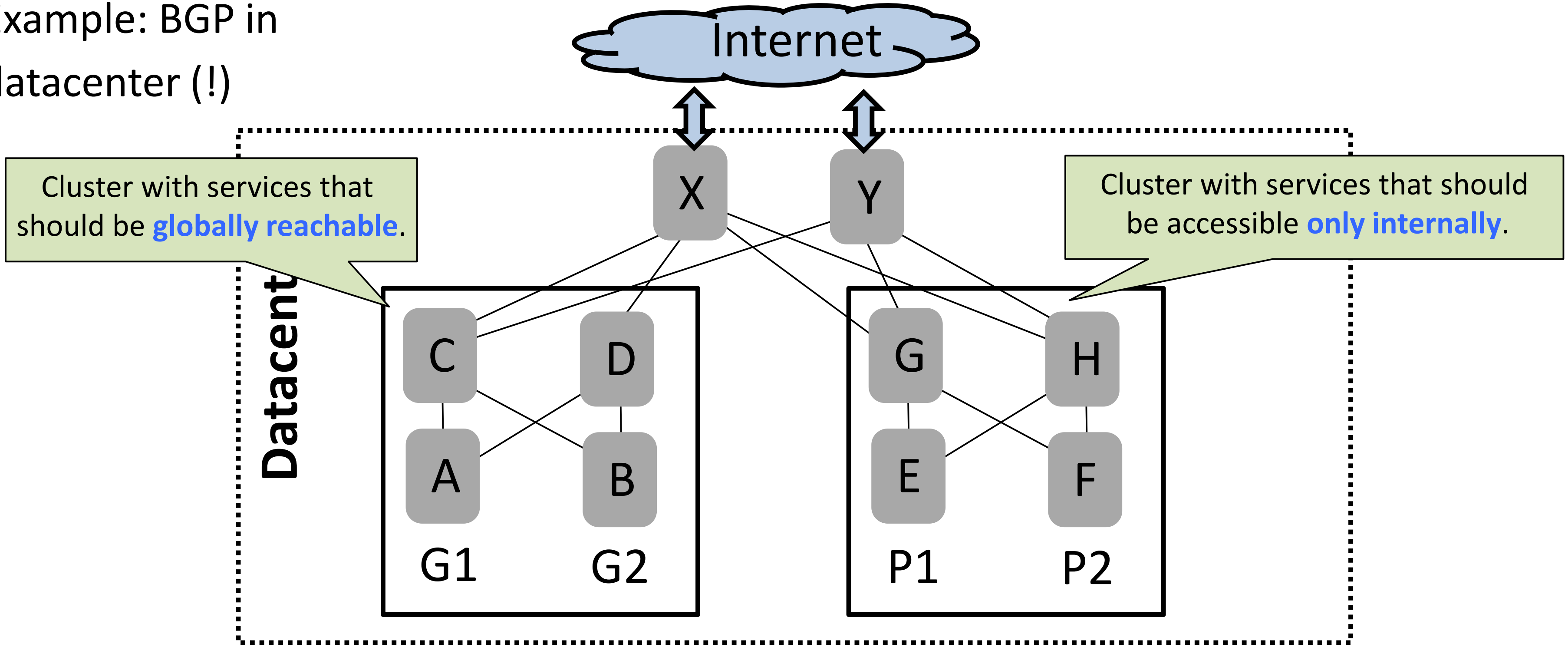
Example: BGP in datacenter (!)



Credits: Beckett et al. (SIGCOMM 2016): Bridging Network-wide Objectives and Device-level Configurations.

Reasoning About Failures is Hard for Humans

Example: BGP in datacenter (!)

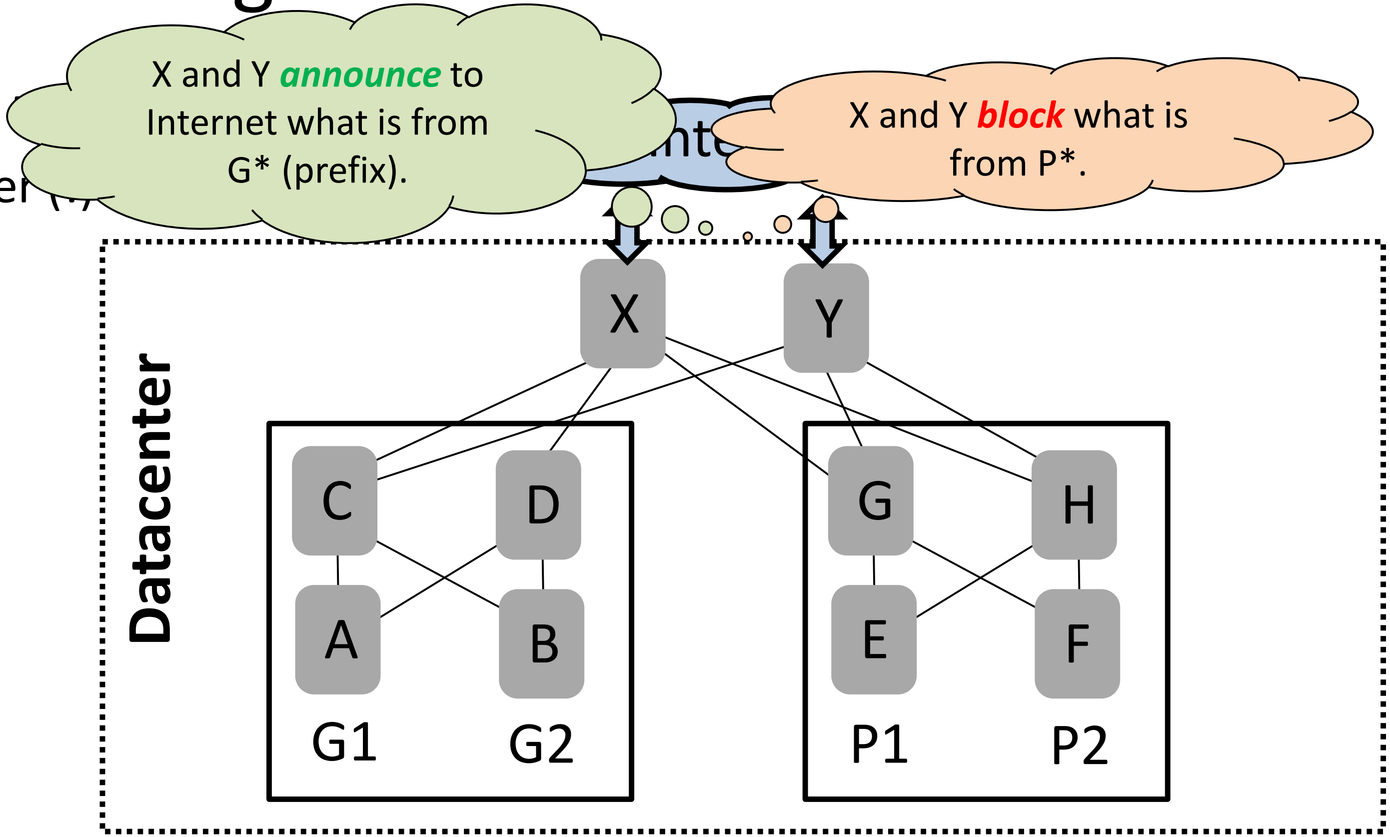


Credits: Beckett et al. (SIGCOMM 2016): Bridging Network-wide Objectives and Device-level Configurations.

Reasoning About Failures is Hard for Humans

Example:

datacenter

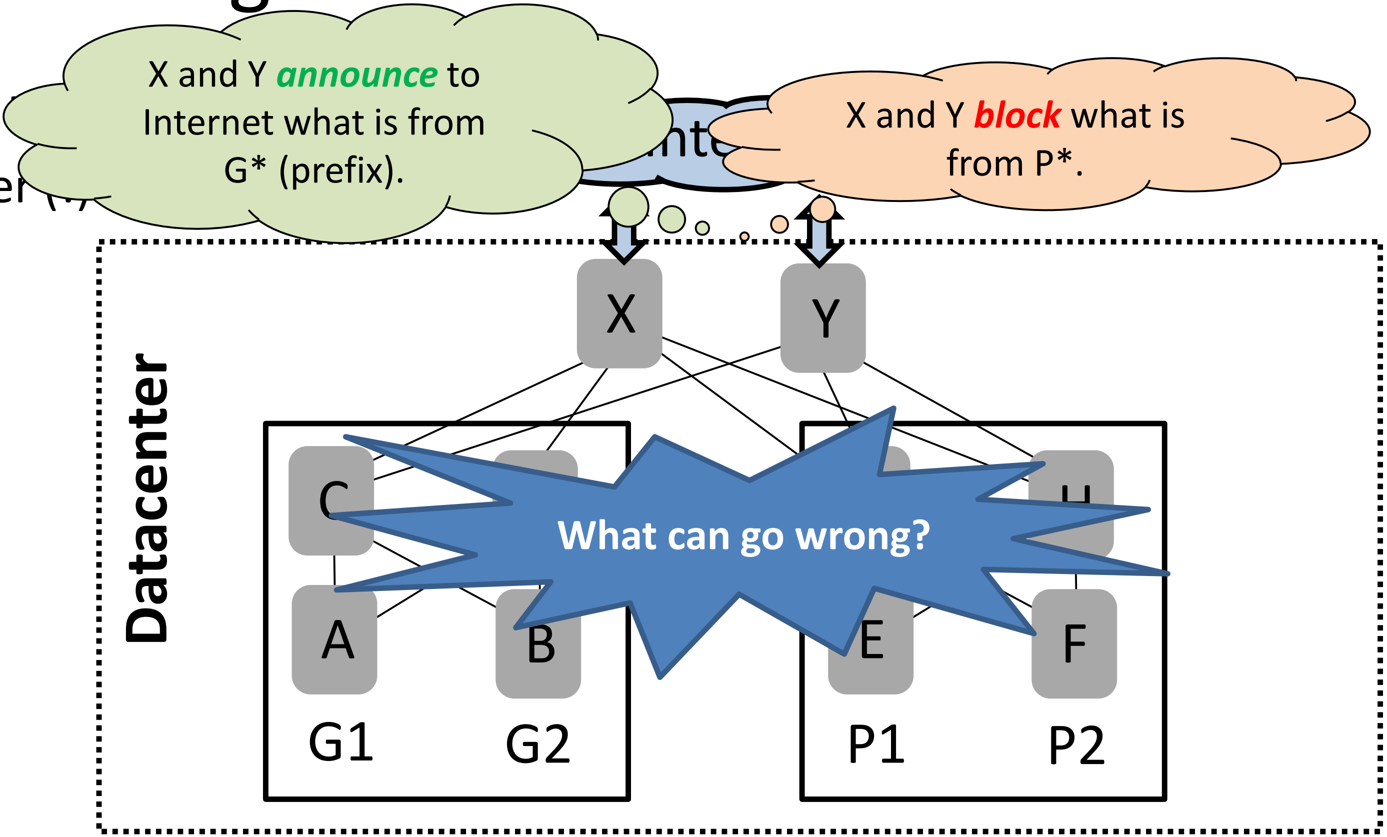


Credits: Beckett et al. (SIGCOMM 2016): Bridging Network-wide Objectives and Device-level Configurations.

Reasoning About Failures is Hard for Humans

Example:

datacenter

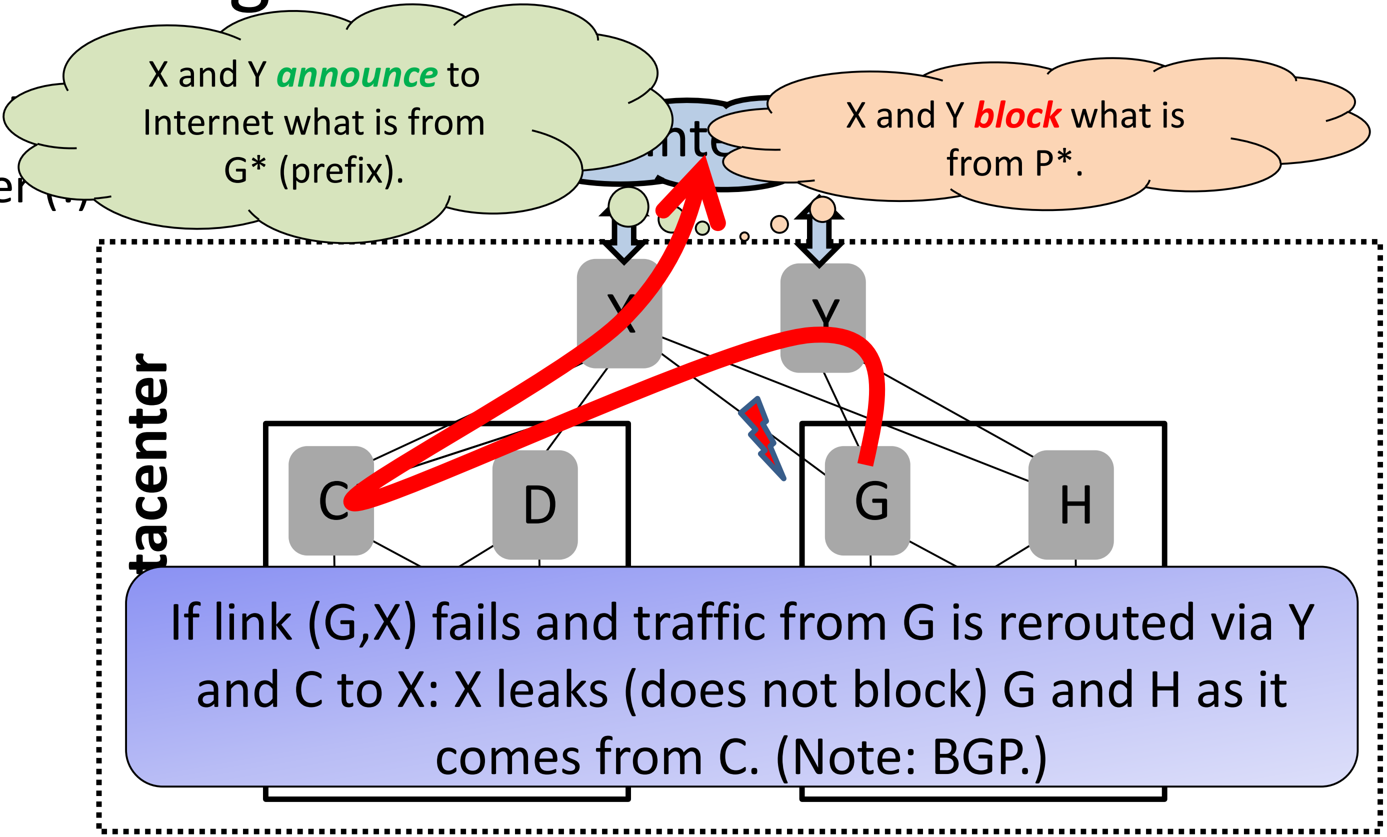


Credits: Beckett et al. (SIGCOMM 2016): Bridging Network-wide Objectives and Device-level Configurations.

Reasoning About Failures is Hard for Humans

Example: [partly hidden]

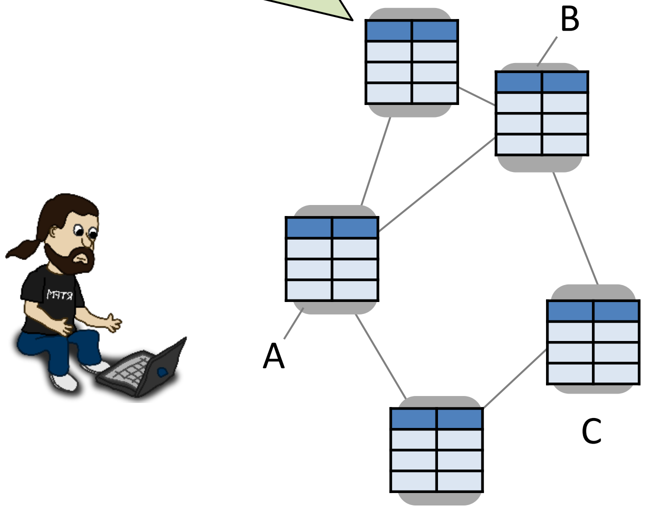
datacenter [partly hidden]



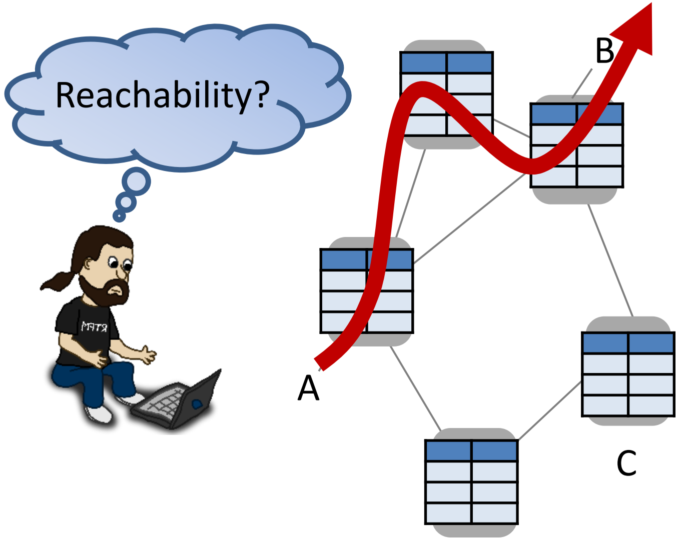
Credits: Beckett et al. (SIGCOMM 2016): Bridging Network-wide Objectives and Device-level Configurations.

Responsibilities of a Sysadmin

Routers and switches store list of forwarding rules, and conditional failover rules.



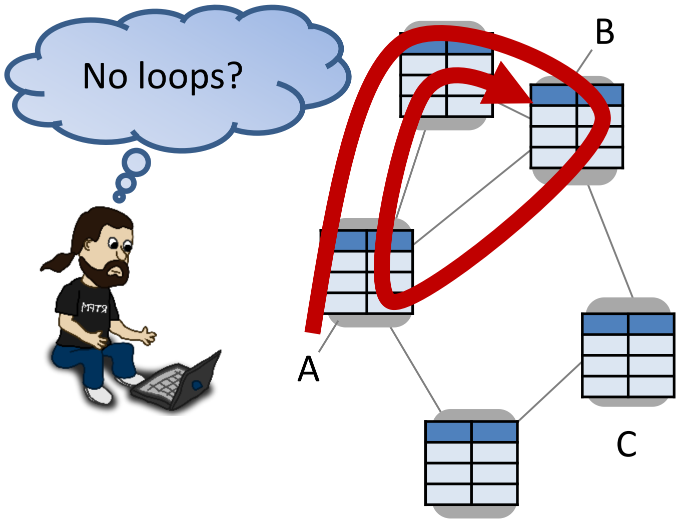
Responsibilities of a Sysadmin



Sysadmin responsible for:

- **Reachability:** Can traffic from ingress port A reach egress port B?

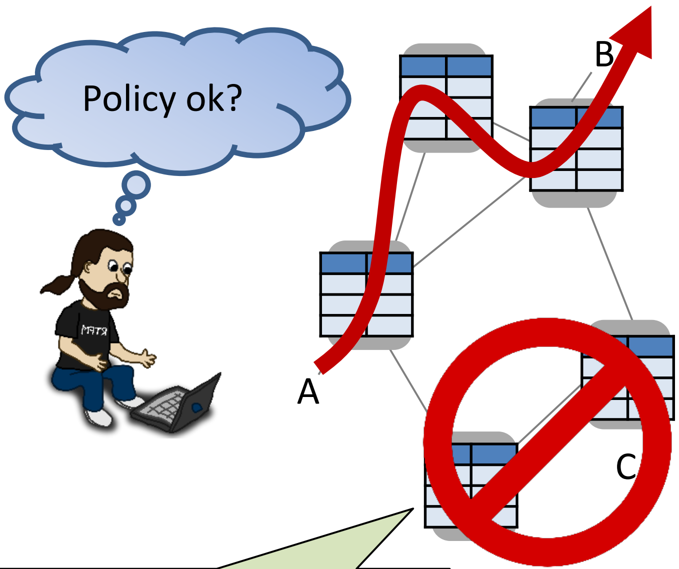
Responsibilities of a Sysadmin



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?

Responsibilities of a Sysadmin

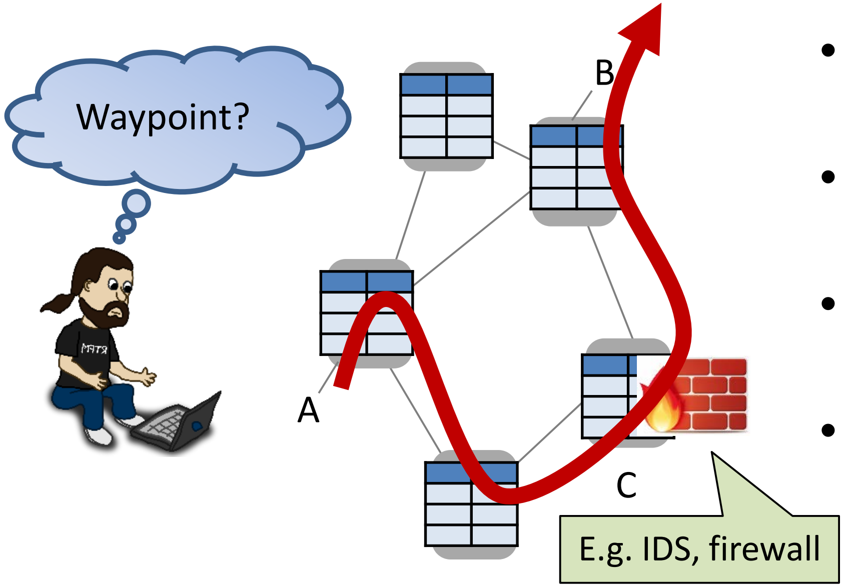


E.g. ***NORDUnet***: no traffic via Iceland (expensive!). Or no traffic through ***route reflectors***.

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?

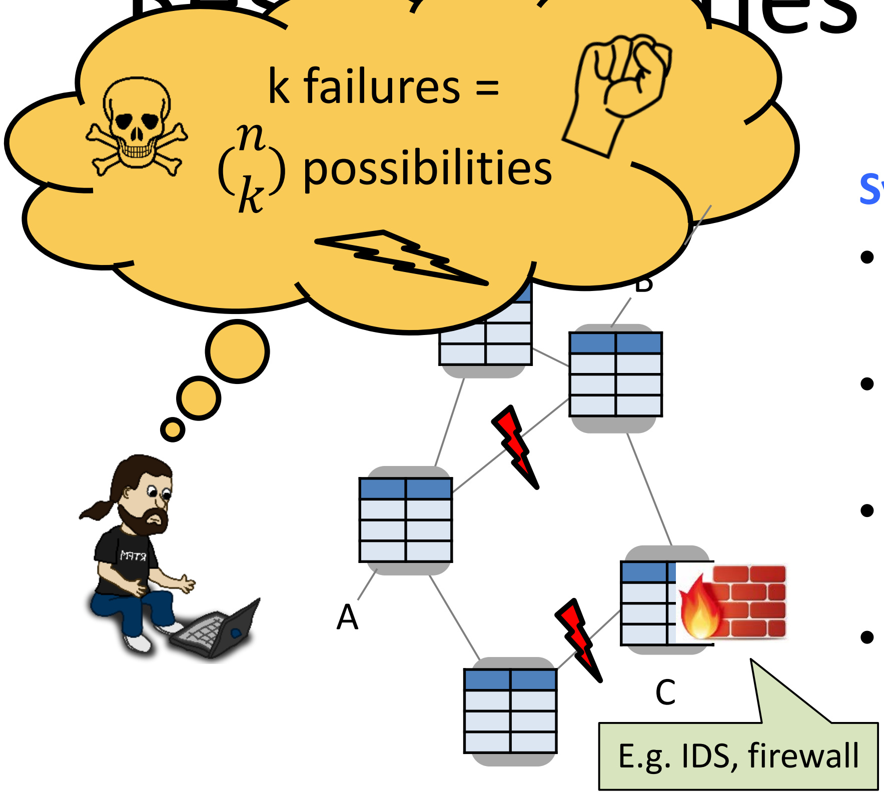
Responsibilities of a Sysadmin



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?

Responsibilities of a Sysadmin



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?

… and everything even under multiple failures?!

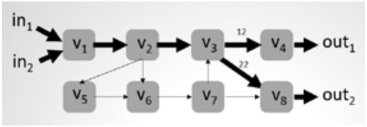
Can we automate such tests
or even self-repair?

Can we automate such tests or even self-repair?

Yes! Automated **What-if Analysis Tool** for MPLS and SR in ***polynomial time***.

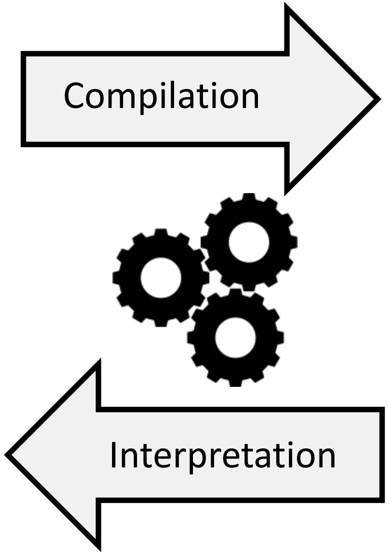
Leveraging Automata-Theoretic Approach

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)	$push(40)$

MPLS **configurations**, Segment Routing etc.



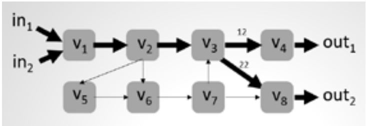
$$pX \Rightarrow qXX$$
$$pX \Rightarrow qYX$$
$$qY \Rightarrow rYY$$
$$rY \Rightarrow r$$
$$rX \Rightarrow pX$$

Pushdown Automaton (PDA) and **Prefix Rewriting Systems** Theory

Leveraging Autom… …oach

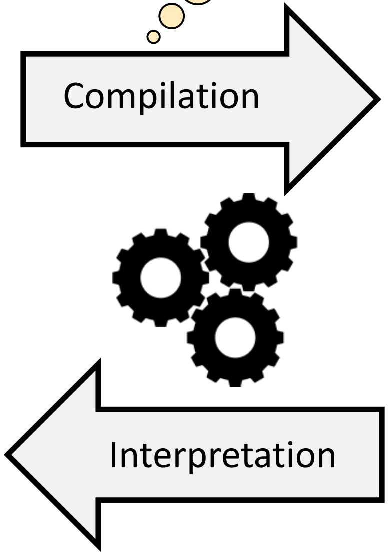
Use cases: Sysadmin ***issues queries*** to test certain properties, or do it on a ***regular basis*** automatically!

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)	$push(40)$

MPLS **configurations**, Segment Routing etc.



$$pX \Rightarrow qXX$$
$$pX \Rightarrow qYX$$
$$qY \Rightarrow rYY$$
$$rY \Rightarrow r$$
$$rX \Rightarrow pX$$

Pushdown Automaton (PDA) and **Prefix Rewriting Systems** Theory

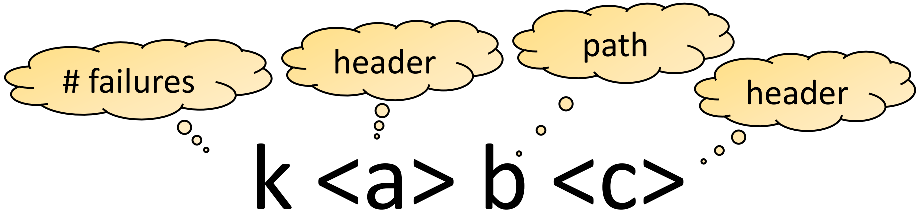
Tool and Query Language

<u>Part 1:</u> Parses query and constructs Push-Down System (PDS)

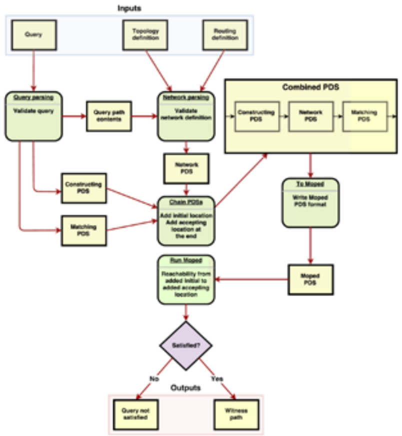
- In Python 3

<u>Part 2:</u> Reachability analysis of constructed PDS

- Using ***Moped*** tool



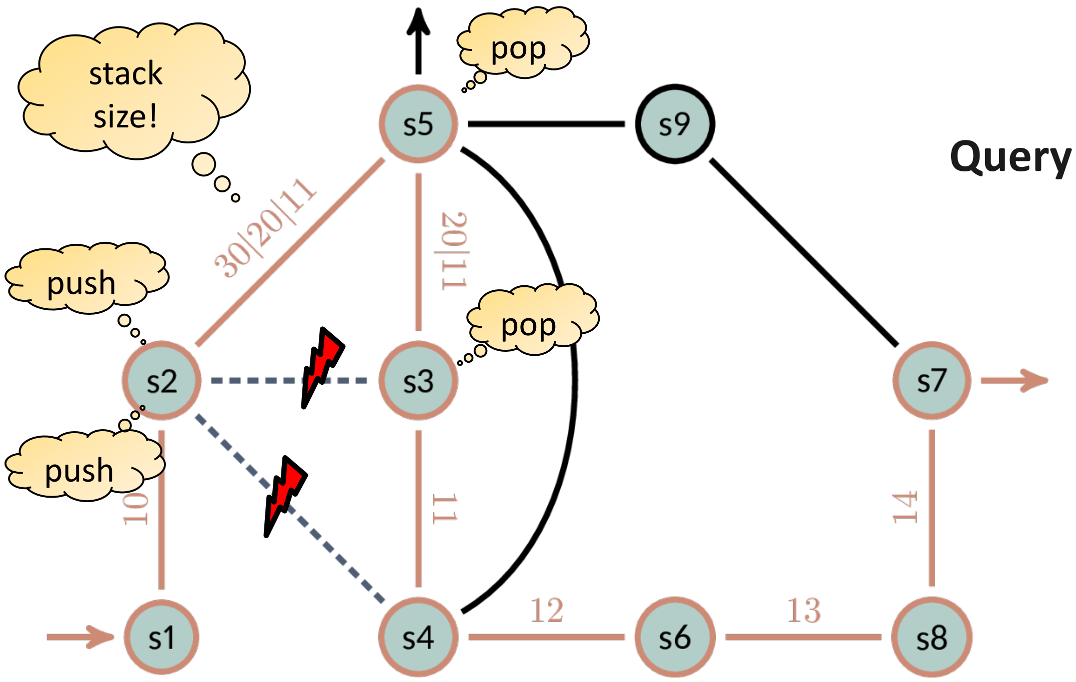
Regular query language



query processing flow

Example: Traversal Testing With 2 Failures

Traversal test with k=2: Can traffic starting with [] go through s5, under up to k=2 failures?



2 failures

Query: k=2 [] s1 >> s5 >> s7 []

YES!

(Gives witness!)

A Complex and Big Formal Language!
Why Polynomial Time?

- Arbitrary number k of failures: How can I avoid checking all $\binom{n}{k}$ many options?
- Even if we reduce to **push-down automaton**: simple operations such as emptiness testing or intersection on PDA is computationally non-trivial and sometimes even **undecidable**!

Time for Automata Theory (from Switzerland!)

- Classic result by **Büchi** 1964: the set of all reachable configurations of a pushdown automaton a is regular set
- Hence, we can operate only on Nondeterministic Finite Automata (NFAs) when reasoning about the pushdown automata
- The resulting **regular operations** are all polynomial time
  - Important result of **model checking**

Julius Richard Büchi

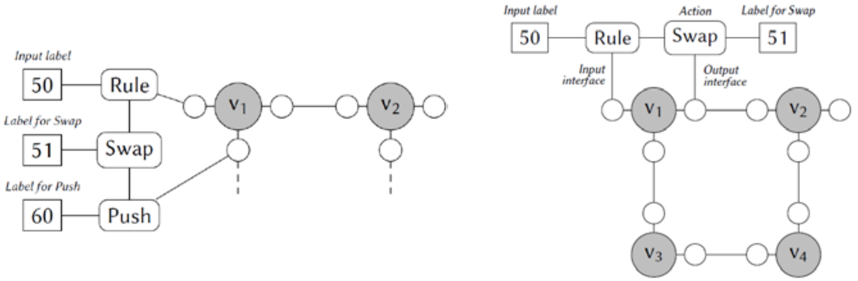
1924-1984

Swiss logician

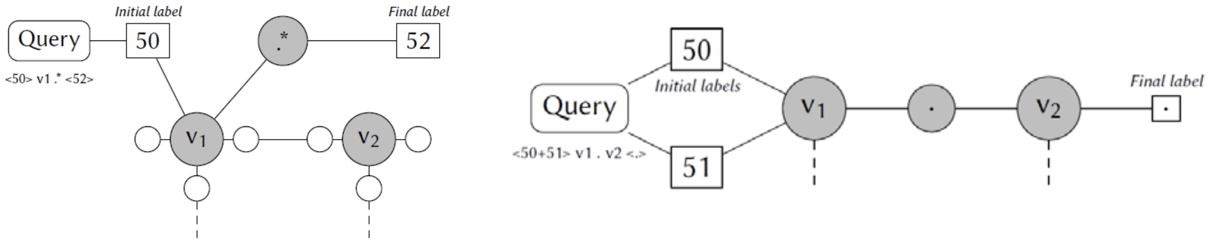
Speedup with Machine Learning

Speed Up Further and Synthesize: Deep Learning

- Yes sometimes **without losing guarantees**
- Extend **graph-based neural networks**
- **Predict** counter-examples and **fixes**



Network topologies and MPLS rules



Network topologies and query

Roadmap

- Opportunities of self-* networks
  - Example 1: Demand-aware, self-adjusting networks
  - Example 2: Self-repairing networks
- Challenges of desinging self-* networks

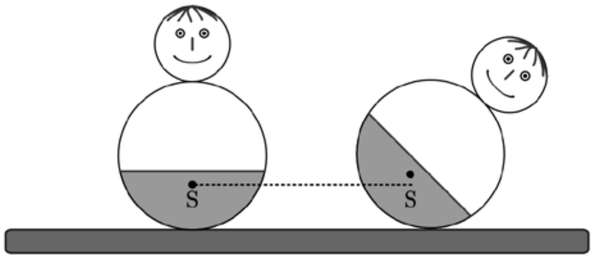
Challenge 1: Realizing Limits?

- Can a self-* network realize its **limits**?
- E.g., when quality of **input data** is not good enough?
- When to hand over to human? Or **fall back** to „safe/oblivious mode“?
- Can we learn from self-driving **cars**?

Challenge 2: Self-Stabilization

- Could be an attractive property of self-* network!

A **self-stabilizing** system guarantees that it *reconverges to a desirable configuration* or state, *from any initial state*.



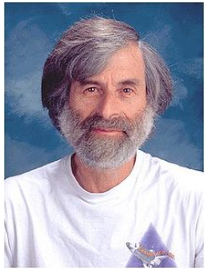
„Stehaufmännchen“

Self-Stabilization

Self-stabilizing algorithms pioneered by **Dijkstra** (1973): for example self-stabilizing mutual exclusion.

“I regard this as Dijkstra’s most brilliant work. Self-stabilization is a very important concept in fault tolerance.”

Leslie **Lamport** (PODC 1983)

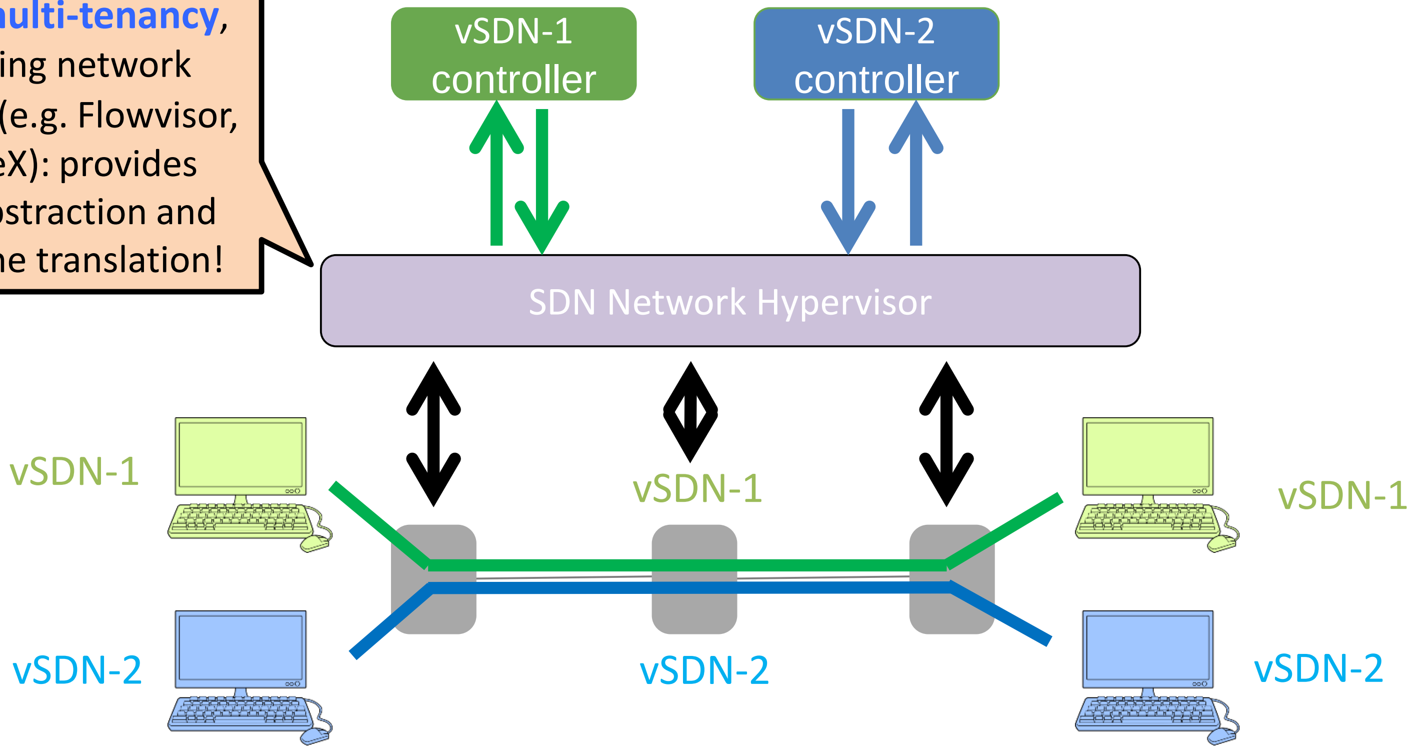


Some notable works by **Perlman** toward self-stabilizing Internet, e.g., self-stabilizing spanning trees.

Yet, many protocols in the Internet are *not* self-stabilizing. Much need for future work.

Challenge 3: Modelling

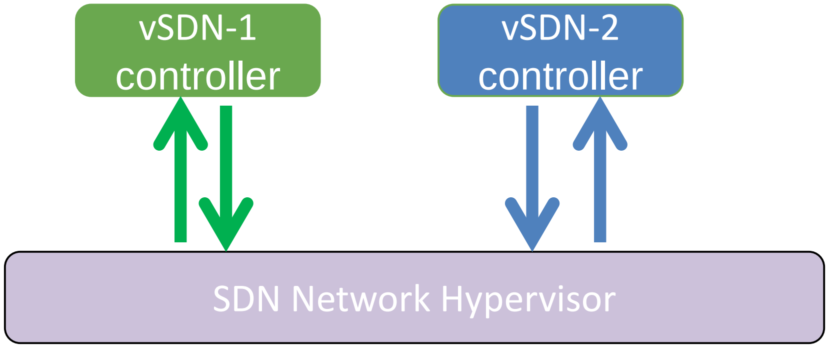
To enable **multi-tenancy**, take existing network **hypervisor** (e.g. Flowvisor, OpenVirteX): provides network abstraction and control plane translation!

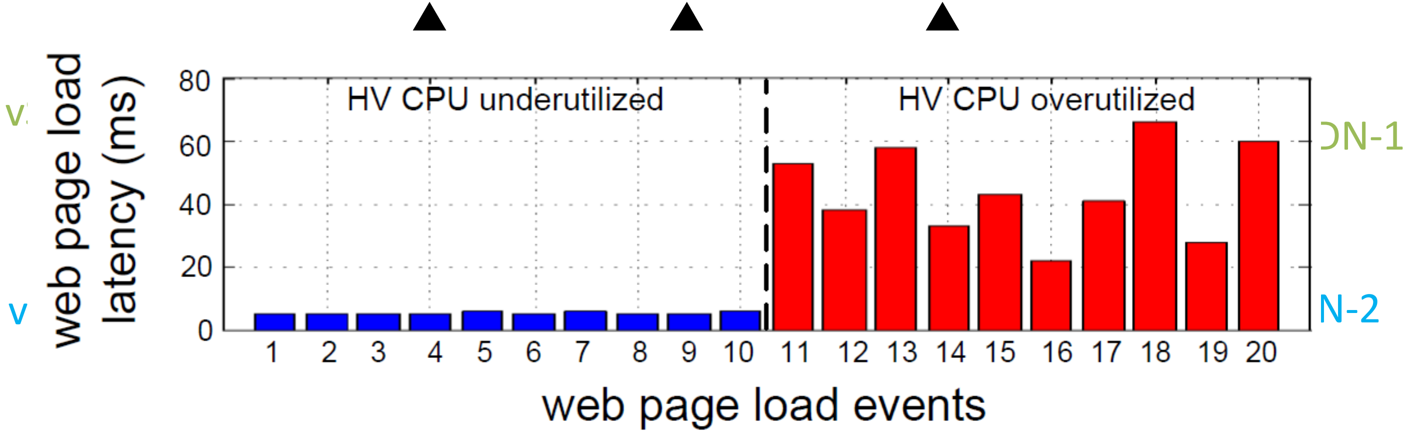


An Experiment: 2 vSDNs with bw guarantee!

Challenge 3: Modelling

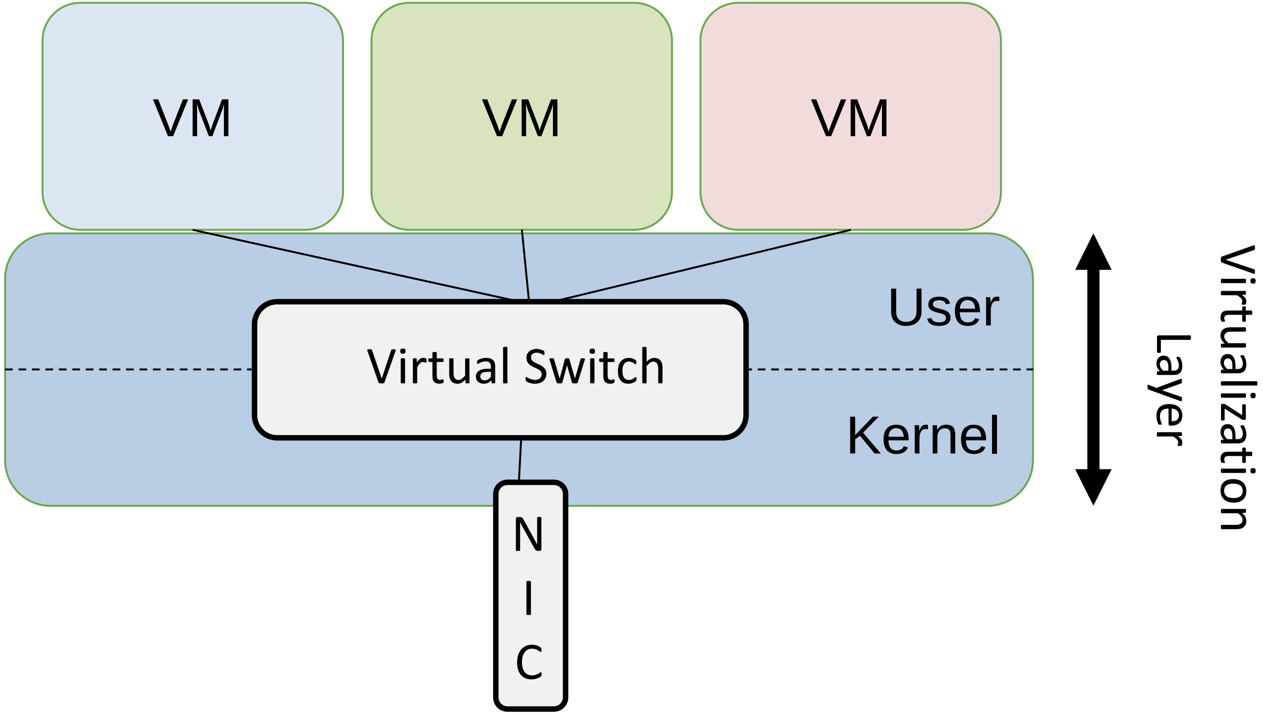
To enable **multi-tenancy**, take existing network **hypervisor** (e.g. Flowvisor, OpenVirteX): provides network abstraction and control plane translation!





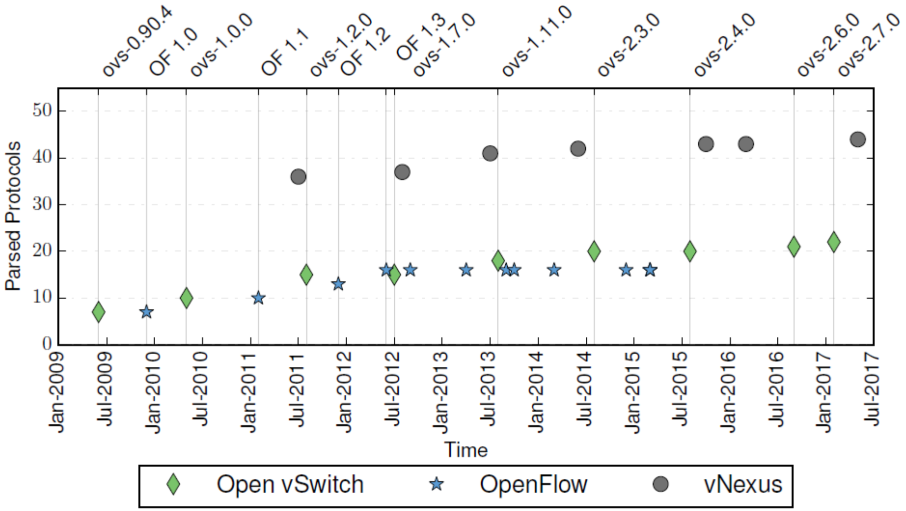
An Experiment: 2 vSDNs with bw guarantee!

Challenge 4: Security



Virtual switches reside in the **server's virtualization layer** (e.g., Xen's Dom0). Goal: provide connectivity and isolation.

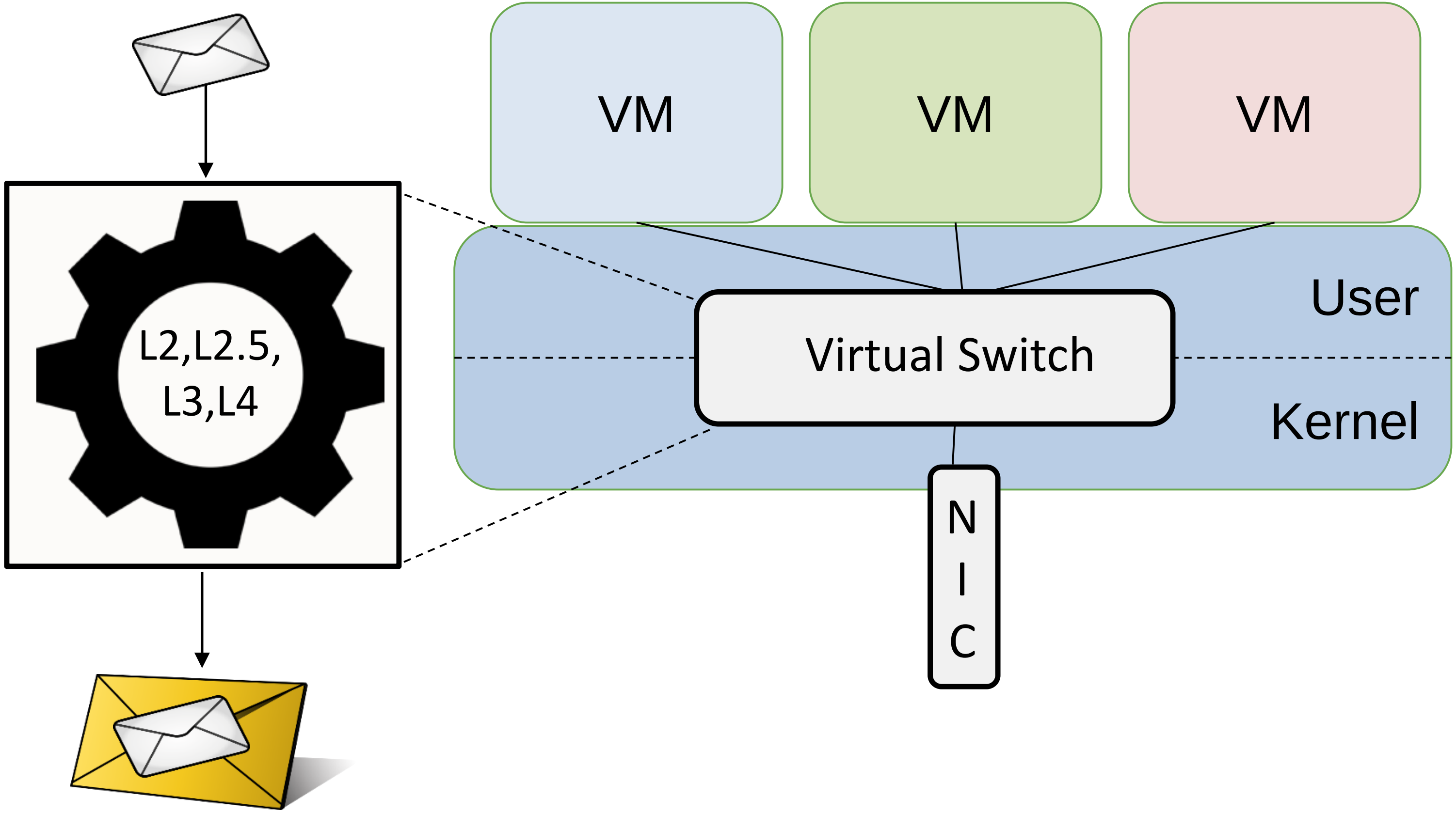
Challenge 4: Security



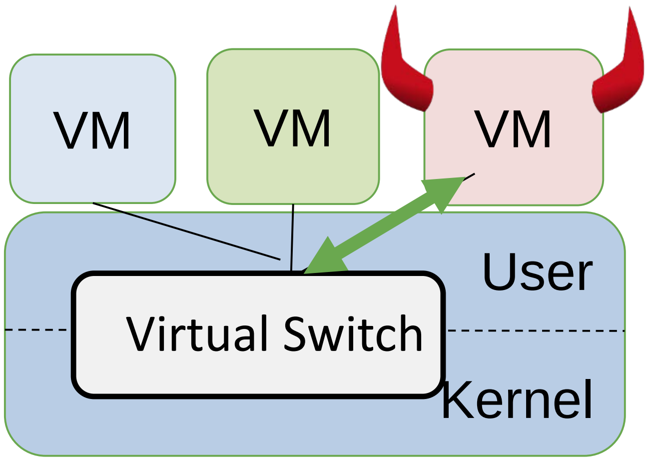
Number of parsed high-level protocols constantly increases…

Challenge 4: Security

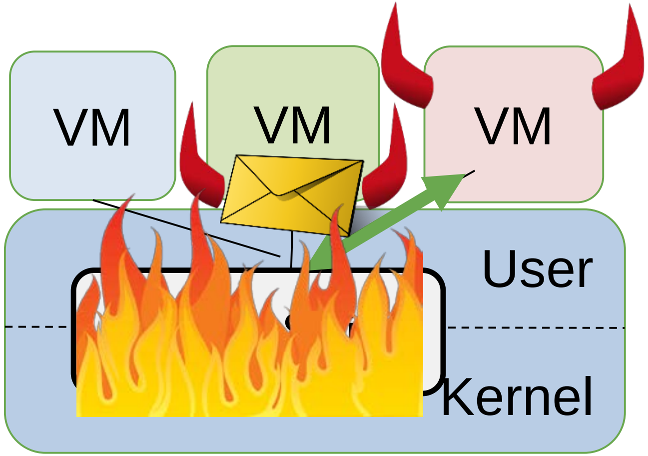
Ethernet
LLC
VLAN
MPLS
IPv4
ICMPv4
TCP
UDP
ARP
SCTP
IPv6
ICMPv6
IPv6 ND
GRE
LISP
VXLAN
PBB
IPv6 EXT HDR
TUNNEL-ID
IPv6 ND
IPv6 EXT HDR
IPv6HOPOPTS
IPv6ROUTING
IPv6Fragment
IPv6DESTOPT
IPv6ESP
IPv6 AH
RARP
IGMP



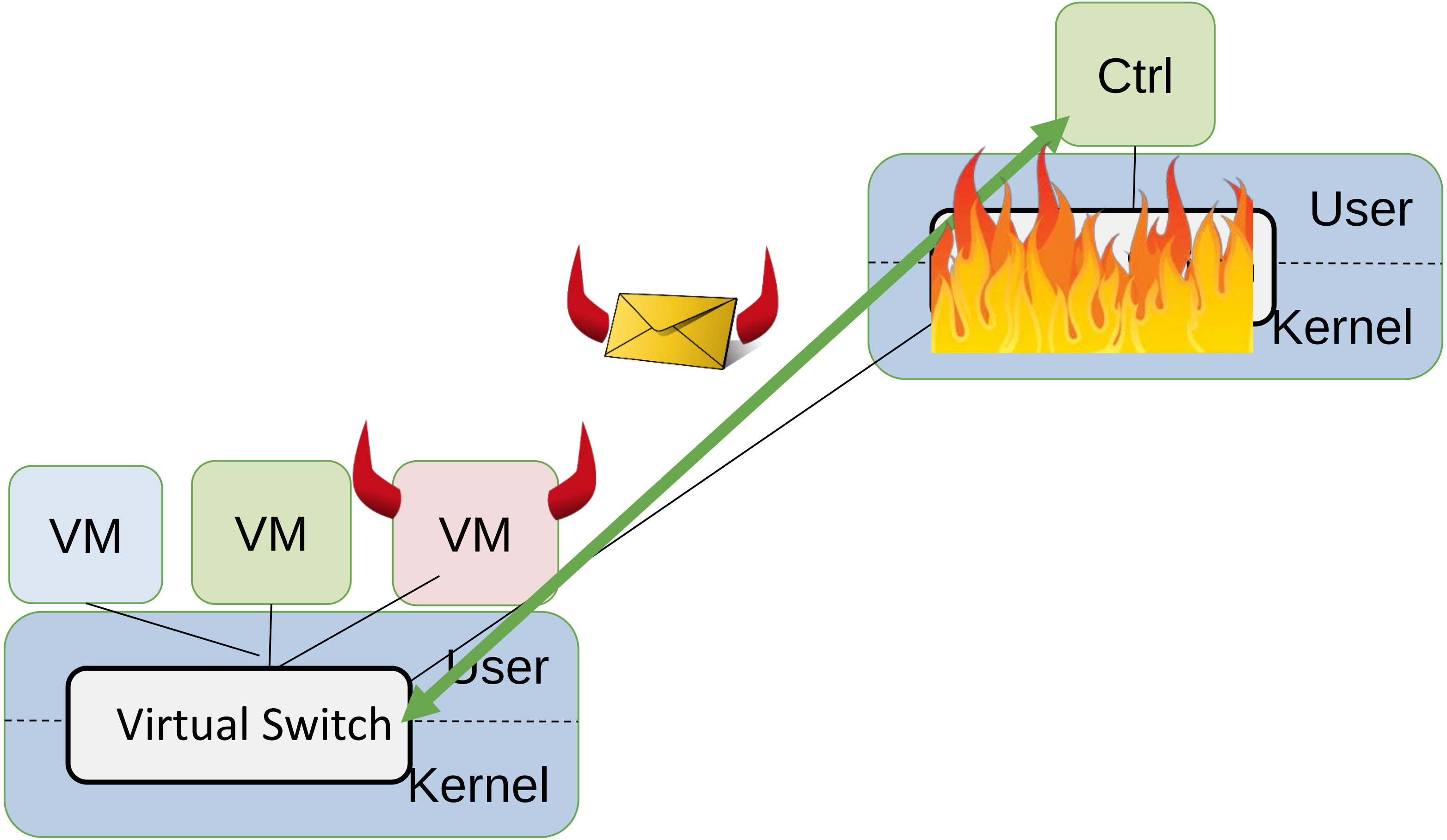
Challenge 4: Security



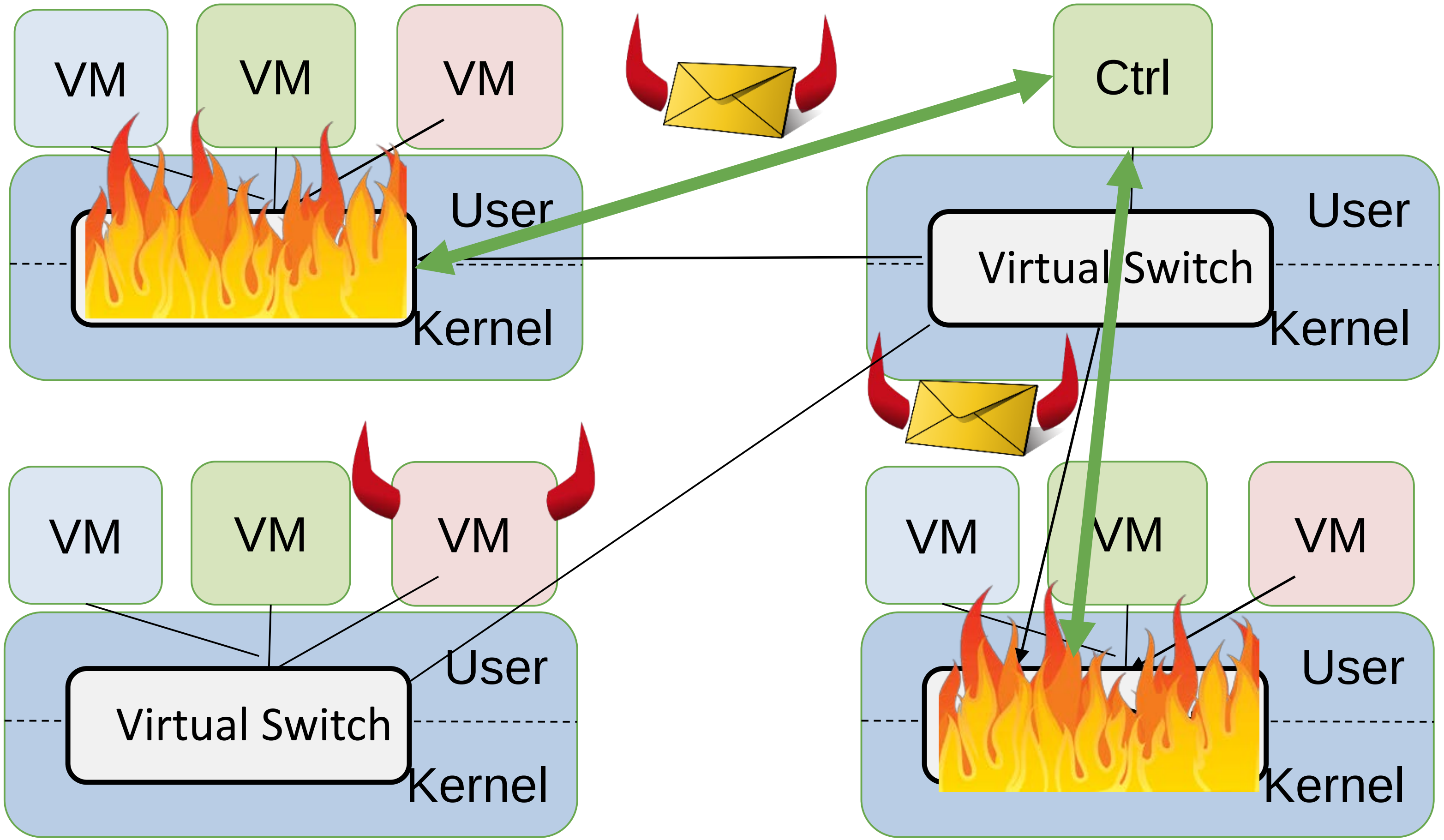
Challenge 4: Security



Challenge 4: Security



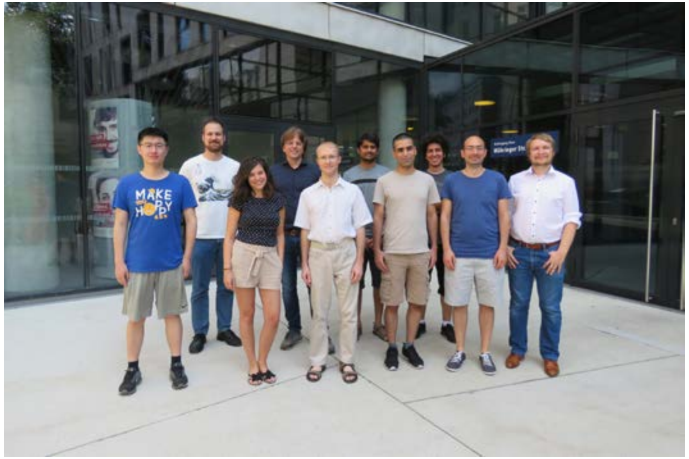
Challenge 4: Security



Conclusions

- Self-* networks: great opportunities (data and flexibilities), great challenges (algorithm design: metrics, formal methods, machine learning)

- We are hiring and looking for interns



universität wien

Fakultät für Informatik

Thanks. Questions?

Univ.-Prof. Dr. Stefan Schmid

stefan_schmid@univie.ac.at

Further Reading

Flexibilities and Complexity

On The Impact of the Network Hypervisor on Virtual Network Performance
Andreas Blenk, Arsany Basta, Wolfgang Kellerer, and Stefan Schmid.
IFIP Networking, Warsaw, Poland, May 2019.
Adaptable and Data-Driven Softwarized Networks: Review, Opportunities, and Challenges (Invited Paper)
Wolfgang Kellerer, Patrick Kalmbach, Andreas Blenk, Arsany Basta, Martin Reisslein, and Stefan Schmid.
Proceedings of the IEEE (**PIEEE**), 2019.
Efficient Distributed Workload (Re-)Embedding
Monika Henzinger, Stefan Neumann, and Stefan Schmid.
ACM/IFIP **SIGMETRICS/PERFORMANCE**, Phoenix, Arizona, USA, June 201
Parametrized Complexity of Virtual Network Embeddings: Dynamic & Linear Programming Approximations
Matthias Rost, Elias Döhne, and Stefan Schmid.
ACM SIGCOMM Computer Communication Review (**CCR**), January 2019.
Charting the Complexity Landscape of Virtual Network Embeddings (Best Paper Award)
Matthias Rost and Stefan Schmid.
IFIP Networking, Zurich, Switzerland, May 2018.
Tomographic Node Placement Strategies and the Impact of the Routing Model
Yvonne Anne Pignolet, Stefan Schmid, and Gilles Tredan.
ACM **SIGMETRICS**, Irvine, California, USA, June 2018. hmid.
ACM/IEEE Symposium on Architectures for Networking and Communications Systems (**ANCS**), Ithaca, New York, USA, July 2018.

Further Reading

Demand-Aware and Self-Adjusting Networks

Survey of Reconfigurable Data Center Networks: Enablers, Algorithms, Complexity
Klaus-Tycho Foerster and Stefan Schmid.
SIGACT News, June 2019.
Toward Demand-Aware Networking: A Theory for Self-Adjusting Networks (Editorial)
Chen Avin and Stefan Schmid.
ACM SIGCOMM Computer Communication Review (**CCR**), October 2018.
Demand-Aware Network Design with Minimal Congestion and Route Lengths
Chen Avin, Kaushik Mondal, and Stefan Schmid.
38th IEEE Conference on Computer Communications (**INFOCOM**), Paris, France, April 2019.
Documents: paper pdf, bibtex bib
Distributed Self-Adjusting Tree Networks
Bruna Peres, Otavio Augusto de Oliveira Souza, Olga Goussevskaia, Chen Avin, and Stefan Schmid.
38th IEEE Conference on Computer Communications (**INFOCOM**), Paris, France, April 2019.
Efficient Non-Segregated Routing for Reconfigurable Demand-Aware Networks
Thomas Fenz, Klaus-Tycho Foerster, Stefan Schmid, and Anaïs Villedieu.
IFIP Networking, Warsaw, Poland, May 2019.
DaRTree: Deadline-Aware Multicast Transfers in Reconfigurable Wide-Area Networks
Long Luo, Klaus-Tycho Foerster, Stefan Schmid, and Hongfang Yu.
IEEE/ACM International Symposium on Quality of Service (**IWQoS**), Phoenix, Arizona, USA, June 2019.
Demand-Aware Network Designs of Bounded Degree
Chen Avin, Kaushik Mondal, and Stefan Schmid.
31st International Symposium on Distributed Computing (**DISC**), Vienna, Austria, October 2017.
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. Early version: IEEE **IPDPS** 2013.
Characterizing the Algorithmic Complexity of Reconfigurable Data Center Architectures
Klaus-Tycho Foerster, Monia Ghobadi, and Stefan Schmid.
ACM/IEEE Symposium on Architectures for Networking and Communications Systems (**ANCS**), Ithaca, New York, USA, July 2018.

Further Reading

Self-Repairing Networks

P-Rex: Fast Verification of MPLS Networks with Multiple Link Failures
Jesper Stenbjerg Jensen, Troels Beck Krogh, Jonas Sand Madsen, Stefan Schmid, Jiri Srba, and Marc Tom Thorgersen.
14th International Conference on emerging Networking EXperiments and Technologies (**CoNEXT**), Heraklion, Greece, December 2018.

Polynomial-Time What-If Analysis for Prefix-Manipulating MPLS Networks
Stefan Schmid and Jiri Srba.
37th IEEE Conference on Computer Communications (**INFOCOM**), Honolulu, Hawaii, USA, April 2018.

Renaissance: A Self-Stabilizing Distributed SDN Control Plane
Marco Canini, Iosif Salem, Liron Schiff, Elad Michael Schiller, and Stefan Schmid.
38th IEEE International Conference on Distributed Computing Systems (**ICDCS**), Vienna, Austria, July 2018.

Empowering Self-Driving Networks
Patrick Kalmbach, Johannes Zerwas, Peter Babarczi, Andreas Blenk, Wolfgang Kellerer, and Stefan Schmid.
ACM SIGCOMM 2018 Workshop on Self-Driving Networks (**SDN**), Budapest, Hungary, August 2018.

DeepMPLS: Fast Analysis of MPLS Configurations using Deep Learning
Fabien Geyer and Stefan Schmid.
IFIP Networking, Warsaw, Poland, May 2019.

Further Reading

Attacks on OVS

MTS: Bringing Multi-Tenancy to Virtual Switches
Kashyap Thimmaraju, Saad Hermak, Gabor Retvari, and Stefan Schmid.
USENIX Annual Technical Conference (**ATC**), Renton, Washington, USA, July 2019.
Taking Control of SDN-based Cloud Systems via the Data Plane (Best Paper Award)
Kashyap Thimmaraju, Bhargava Shastry, Tobias Fiebig, Felicitas Hetzelt, Jean-Pierre Seifert, Anja Feldmann, and Stefan Schmid.
ACM Symposium on SDN Research (**SOSR**), Los Angeles, California, USA, March 2018.
The vAMP Attack: Taking Control of Cloud Systems via the Unified Packet Parser
Kashyap Thimmaraju, Bhargava Shastry, Tobias Fiebig, Felicitas Hetzelt, Jean-Pierre Seifert, Anja Feldmann, and Stefan Schmid.
9th ACM Cloud Computing Security Workshop (**CCSW**), collocated with ACM CCS, Dallas, Texas, USA, November 2017.

Modeling Challenges

NetBOA: Self-Driving Network Benchmarking
Johannes Zerwas, Patrick Kalmbach, Laurenz Henkel, Gabor Retvari, Wolfgang Kellerer, Andreas Blenk, and Stefan Schmid.
ACM SIGCOMM Workshop on Network Meets AI & ML (**NetAI**), Beijing, China, August 2019.
On The Impact of the Network Hypervisor on Virtual Network Performance (Nominated for Best Paper Award)
Andreas Blenk, Arsany Basta, Wolfgang Kellerer, and Stefan Schmid.
IFIP Networking, Warsaw, Poland, May 2019.

Self-Stabilization

Renaissance: A Self-Stabilizing Distributed SDN Control Plane
Marco Canini, Iosif Salem, Liron Schiff, Elad Michael Schiller, and Stefan Schmid.
38th IEEE International Conference on Distributed Computing Systems (**ICDCS**), Vienna, Austria, July 2018.