Towards Self-Driving Networks: Automated What-if Analysis and Synthesis for Dependable Networks

Stefan Schmid (University of Vienna)

Communication Networks

- **Critical infrastructure** of digital society
  - Popularity of **datacentric applications**: health, business, entertainment, social networking, AI/ML, etc.
  - Evident during ongoing pandemic: online learning, online conferences, etc.
- Traffic is currently growing explosively
  - Especially in, to and, from **datacenters**

Facebook datacenter

Increasingly stringent dependability requirements!

Requirements vs Reality

Entire 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.

… 1000s 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 ▼

BA flights around the world were grounded as a result of the Amadeus outage

… even 911 services 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.

Outages simply due to human error! (No attacks…)

Even Tech-Savvy Companies Struggle

We discovered a misconfiguration on this pair of switches that caused what's called a "bridge loop" in the network.

A network change was […] executed incorrectly […] more "stuck" volumes and added more requests to the re-mirroring storm.

Service outage was due to a series of internal network events that corrupted router data tables.

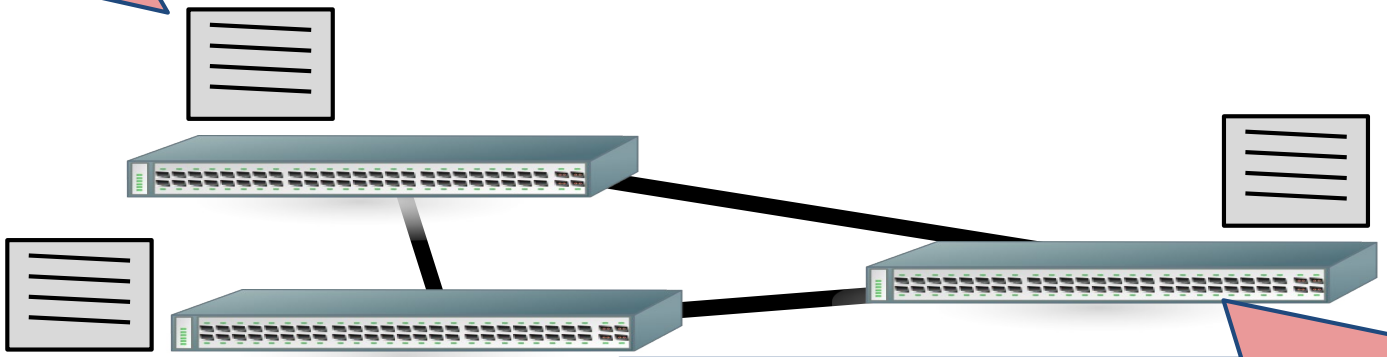
Experienced a network connectivity issue […] interrupted the airline's flight departures, airport processing and reservations systems

Also here: due to human errors.

No Surprise: Networks Are Complex

Manual, device-centric network configurations
(CLI, LANmanager)

Un-evolved best practices
(tcpdump, traceroute - from the 1990s)

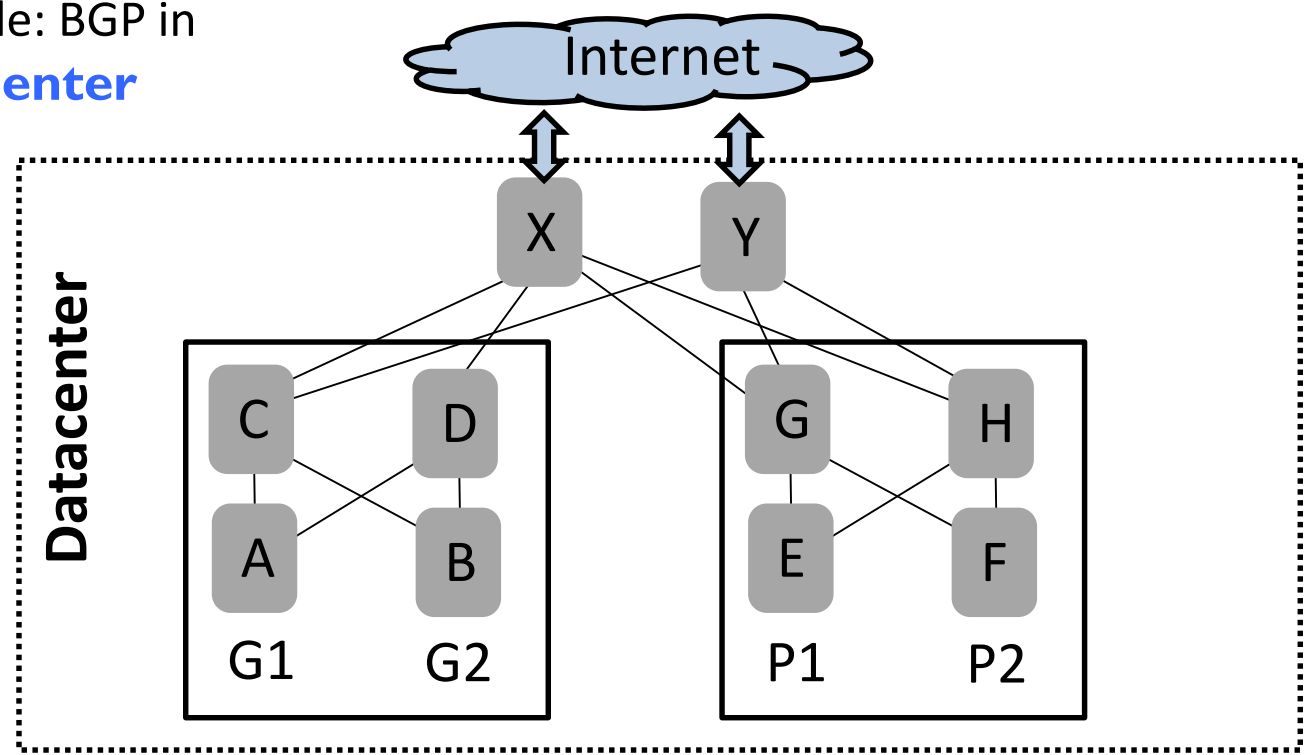


Complex, leaky, low-level interfaces
(VLANs, Spanning Tree, Routing)

500-router network: typically
>1 million lines of configuration

Particularly Challenging for Humans: Reasoning about Policy-Compliance under Failures

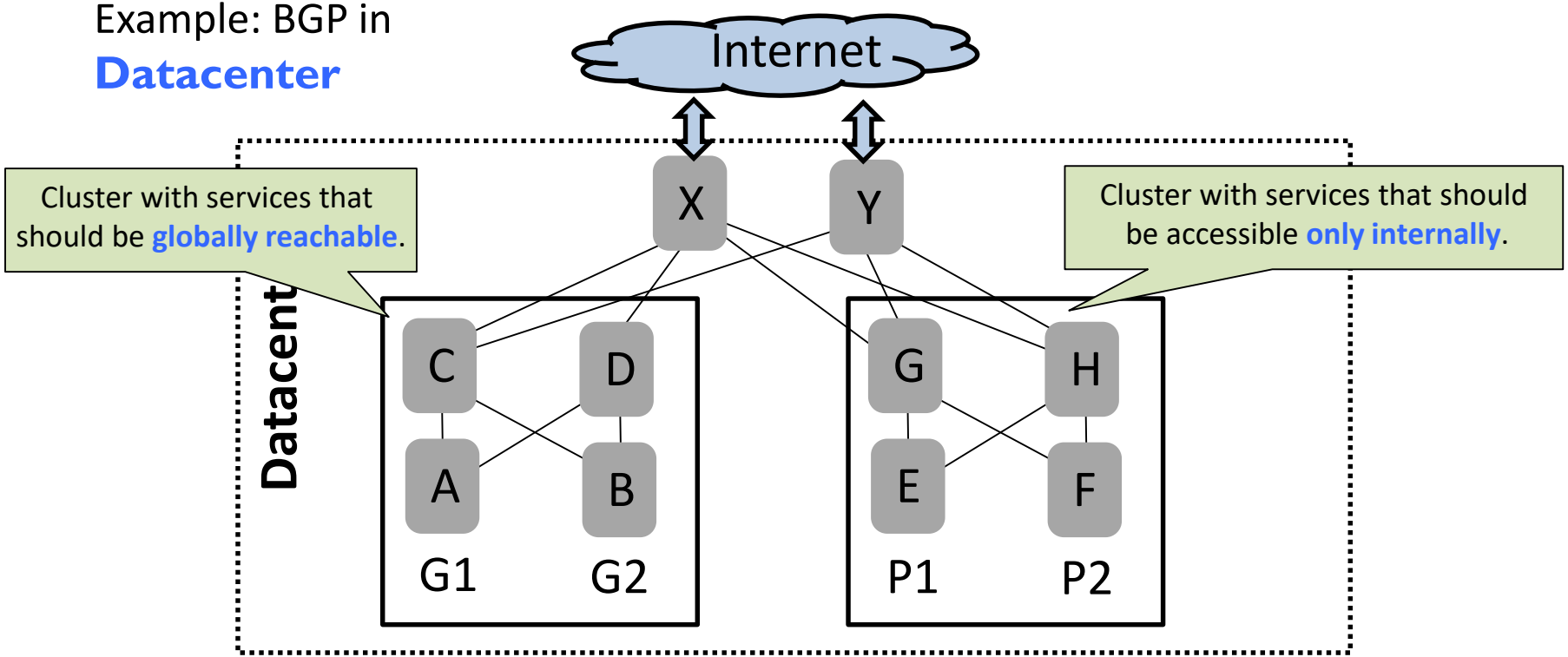
Example: BGP in **Datacenter**



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

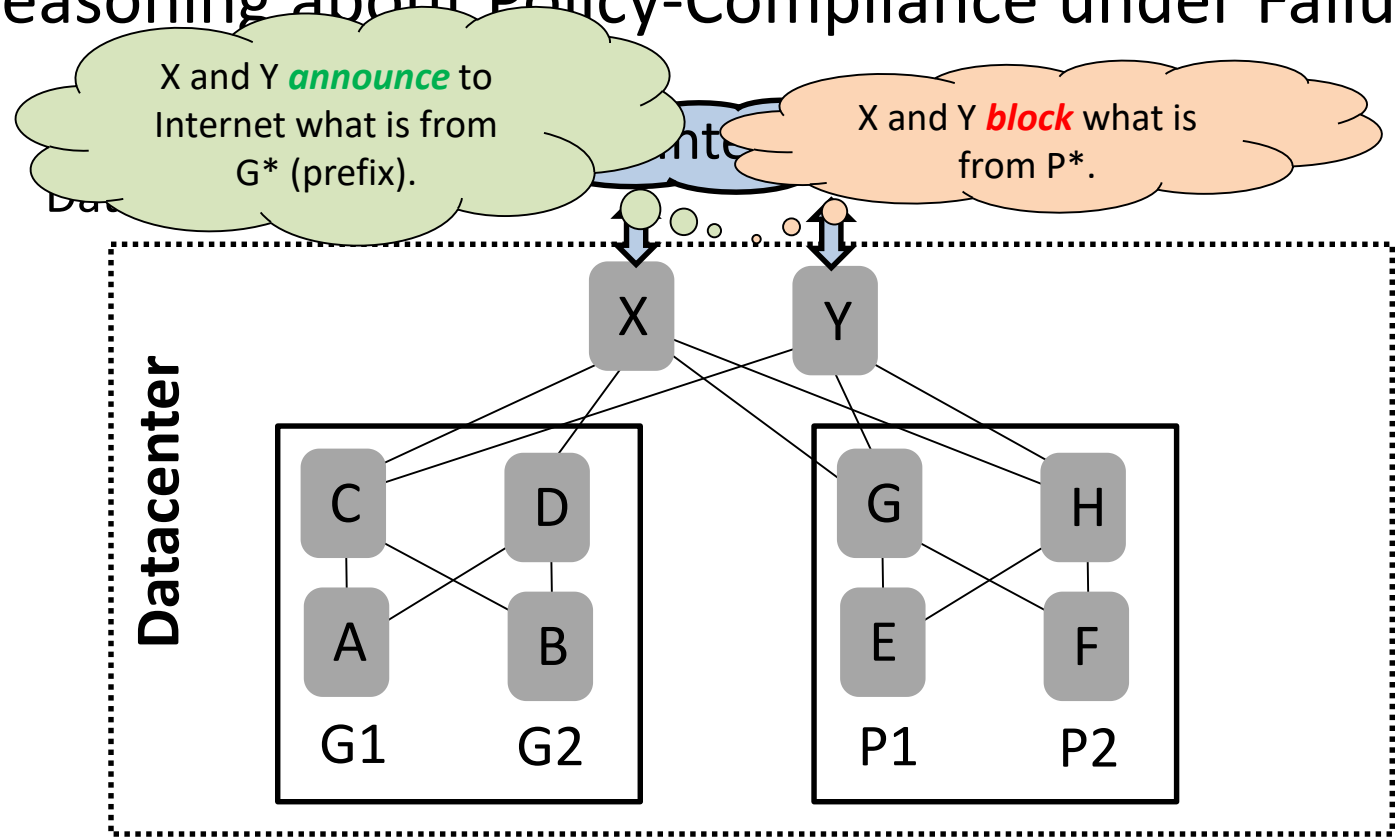
Particularly Challenging for Humans: Reasoning about Policy-Compliance under Failures

Example: BGP in **Datacenter**



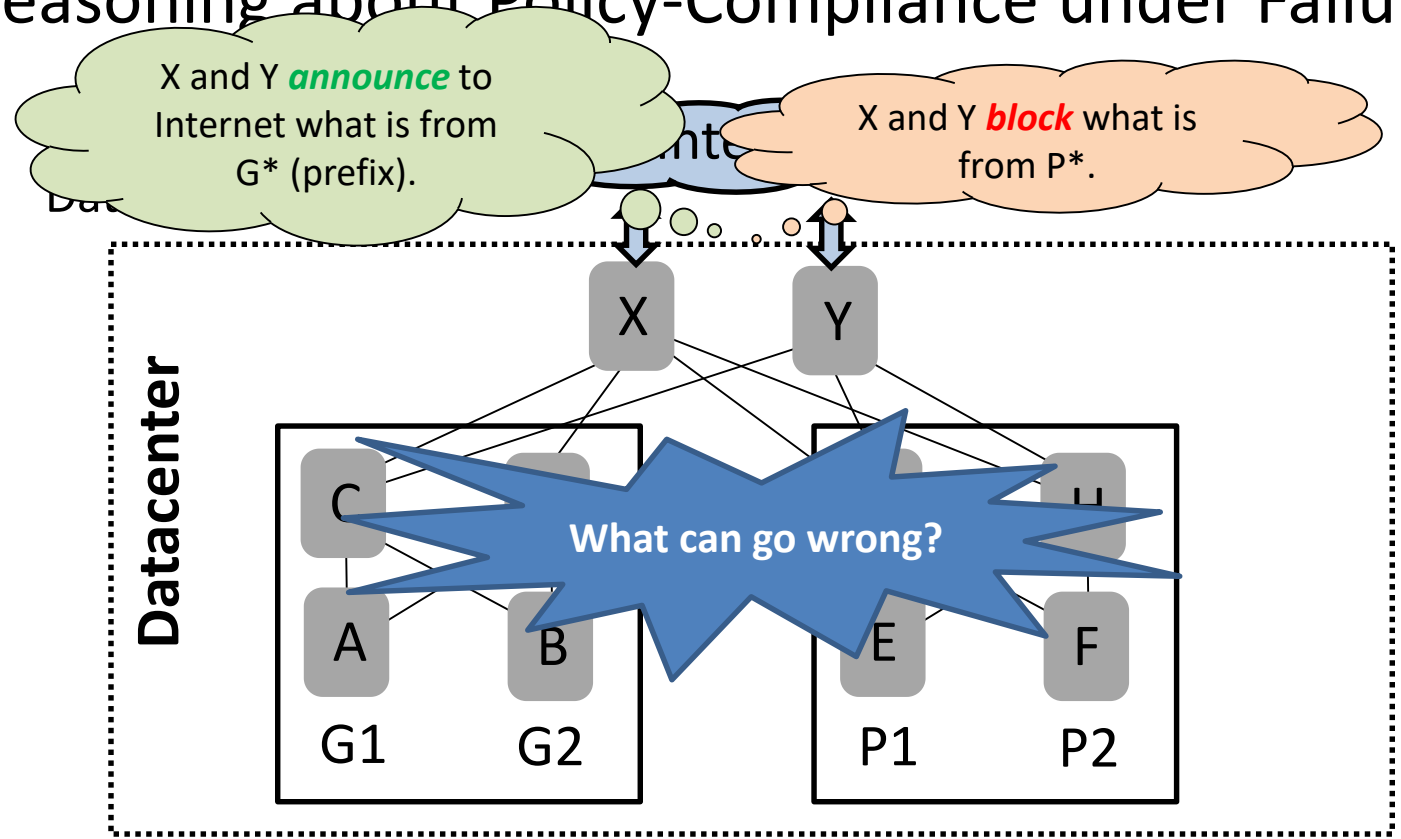
Credits: Beckett et al. (SIGCOMM 2016): Bridging Network-wide Objectives and Device-level Configurations.

Particularly Challenging for Humans: Reasoning about Policy-Compliance under Failures



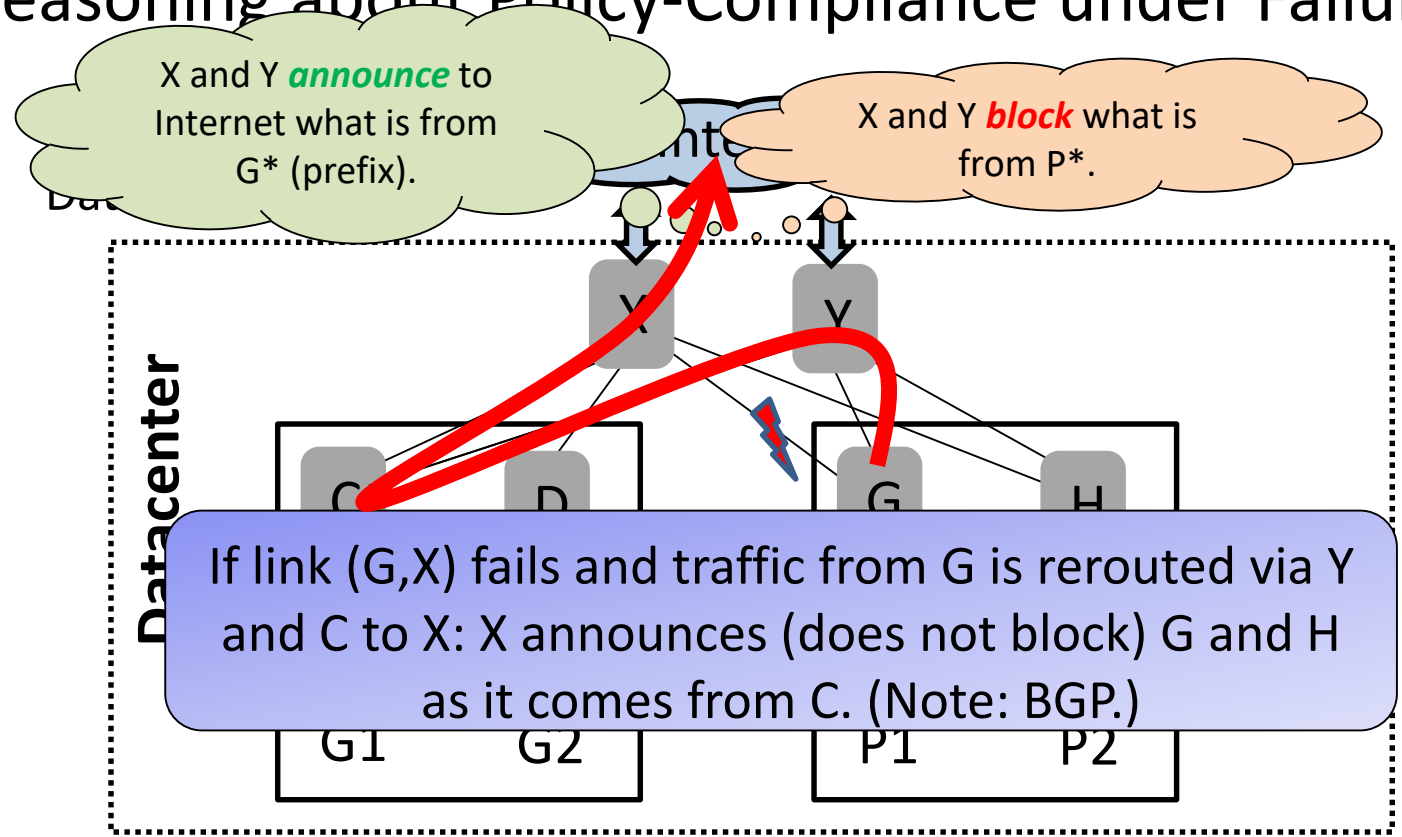
Credits: Beckett et al. (SIGCOMM 2016): Bridging Network-wide Objectives and Device-level Configurations.

Particularly Challenging for Humans: Reasoning about Policy-Compliance under Failures



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

Particularly Challenging for Humans: Reasoning about Policy-Compliance under Failures



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

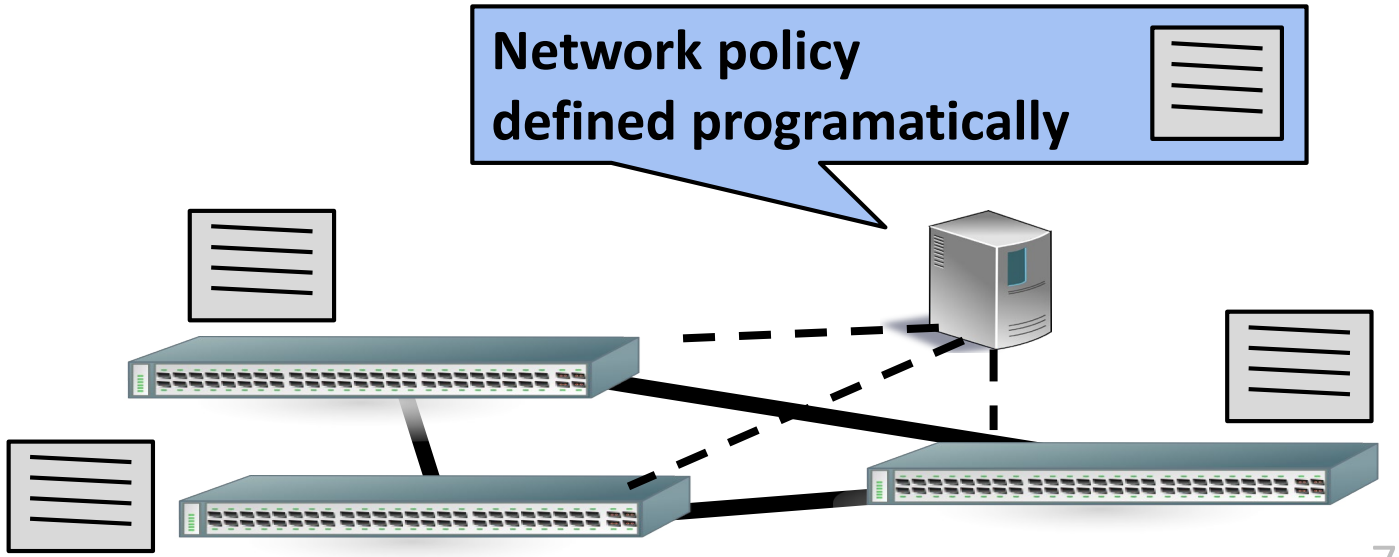
We're Falling Behind the Curve: Increasing Complexity, Software from the 90s

- Anecdote **Wall Street bank**: outage of a datacenter
  - Lost revenue measured in **1 mio$/min**
- Quickly, an emergency team was assembled with experts in compute, storage and networking:
  - **The compute team:** ***reams of logs***, written experiments to reproduce and ***isolate the error***
  - **The storage team:** ***system logs*** were affected, ***workaround programs***.
  - "All the **networking team** had were ***two tools invented over twenty years ago*** to merely test end-to-end connectivity. Neither tool could reveal ***problems with the switches***, the ***congestion*** experienced."

Source: «The world's fastest and most programmable networks» White Paper Barefoot Networks

There is Hope: Software-Defined Networks

- **Automation** and abstraction
- Directly program routing behavior (i.e., push forwarding tables)
- Open interfaces: „the **Linux** of networking“

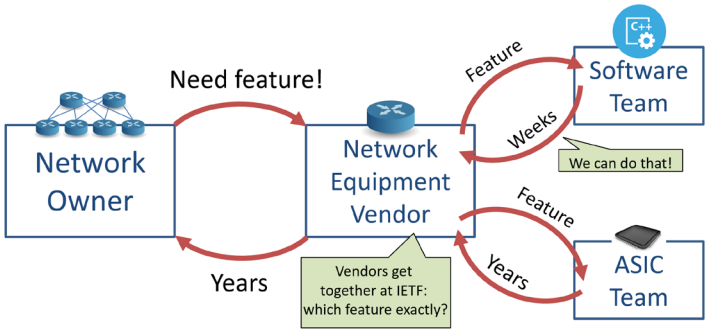


Remark (for the Network Experts…)

- Networks currently become *programmable* in the control plane and the data plane
  - **Control plane**: network-wide algorithms (e.g., routing)
  - **Data plane**: router/switch level algorithms (e.g., forwarding, filtering)

- Motivation in both cases: software usually trumps hardware in terms of *innovation speed*

- **Software** can be *fast*:
  - Our Tofino switch: operates at *6.5 Tb/s*
  - Order of magnitude faster than our faculty's Internet connection: can switch entire *Netflix catalogue* in 20sec
  - While *running a 4000 line program* on any packet...
  - .. and not being more costly or consume more power



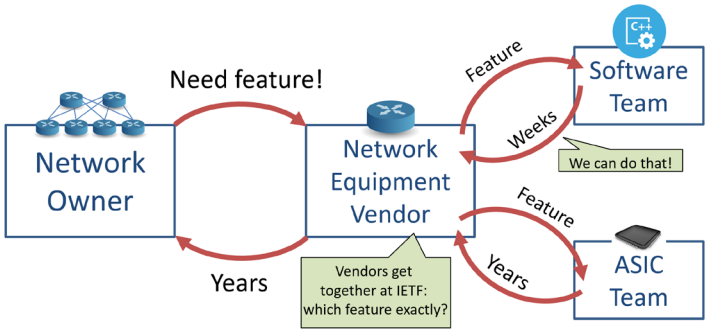
Example: VxLAN

Remark (for the Network Experts…)

- Networks currently become *programmable* in the control plane and the data plane
  - **Control plane**: network-wide algorithms (e.g., routing)
  - **Data plane**: router/switch level algorithms (e.g., forwarding, filtering)

- Motivation in both cases: software usually trumps hardware in terms of *innovation speed*

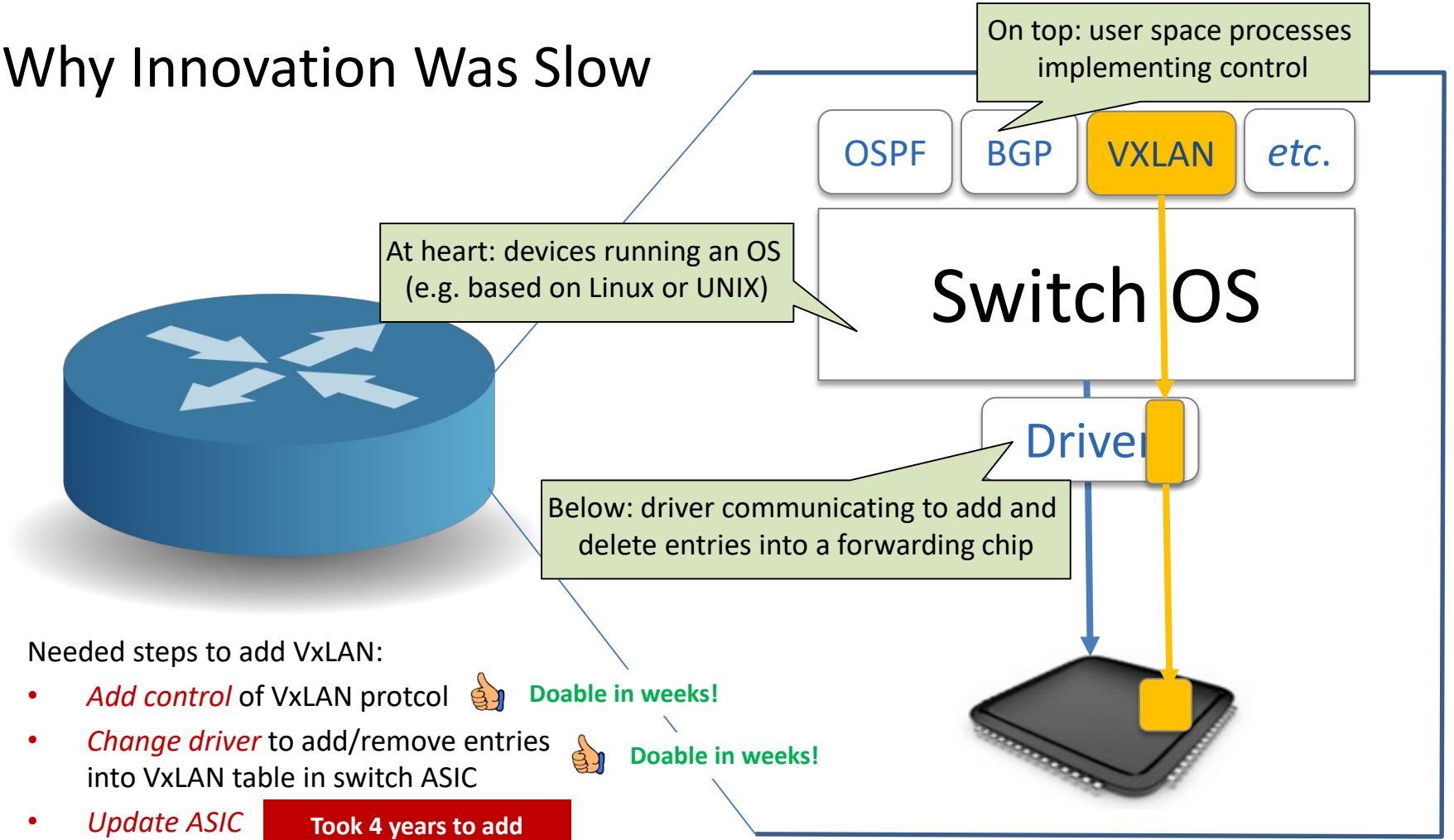
- **Software** can be *fast*:
  - Our Tofino switch: operates at *6.5 Tb/s*
  - Order of magnitude faster than our faculty's Internet connection: can switch entire *Netflix catalogue* in 20sec
  - While *running a 4000 line program* on any packet...
  - .. and not being more costly or consume more power



Example: VxLAN

… and: the automation trend is not limited to SDN.

Why Innovation Was Slow

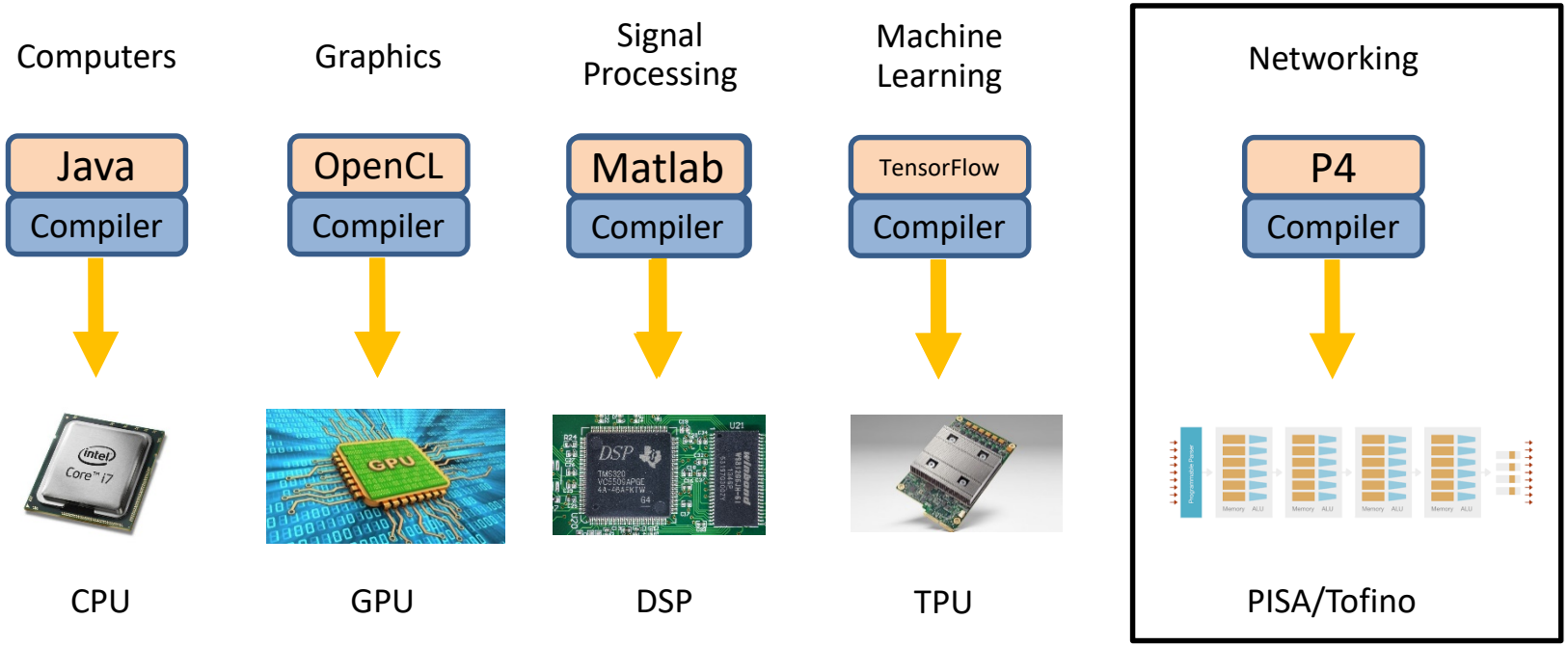


Needed steps to add VxLAN:

- *Add control* of VxLAN protcol — **Doable in weeks!**
- *Change driver* to add/remove entries into VxLAN table in switch ASIC — **Doable in weeks!**
- *Update ASIC* — **Took 4 years to add feature to ASIC! ☹**

Now Networking is Catching Up

Similar to other IT trends: can now *write high-level program* and compile it to *domain specific processor*.



Roadmap

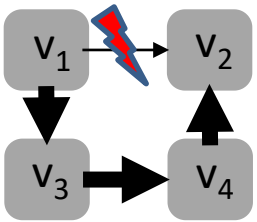
- **A Static Problem:** Policy Compliance *Under Failures*
  - AalWiNes: Fast Automated **What-if Analysis** for Networks (INFOCOM 2018, ACM CoNEXT 2018, ACM CoNEXT 2019, TACAS 2021)

- **A Dynamic Problem: Scheduling** Consistent Network *Updates*
  - Latte and quantitative extensions (PODC 2015, ICALP 2018, PERFORMANCE 2021)

Background: Rerouting Under Failures

Two approaches to react to link failures

- In the **control plane**: just re-invoke (shortest path) routing protocol
  - Always re-establishes connectivity but slow
- In the **data plane**: pre-defined local failover rules
  - Orders of magnitude ***faster*** **Our focus!**

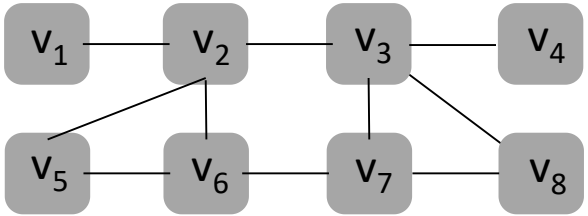


Restoration in control plane takes time -> packet drops!

(c) Marco Chiesa, video shot taken from “Lemmings” designed and developed by DMA Design

How (MPLS) Networks Work

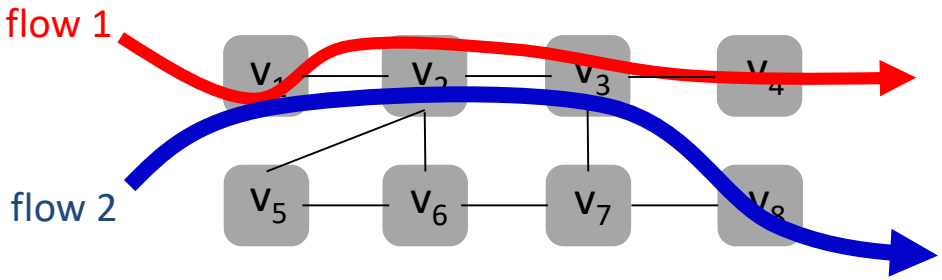
- Forwarding based on **top label** of label **stack**



Default routing of two flows

How (MPLS) Networks Work

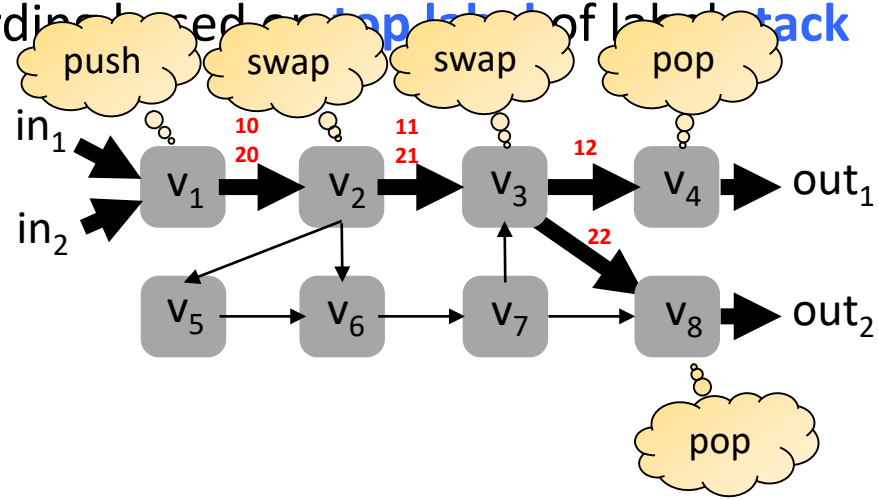
- Forwarding based on **top label** of label **stack**



Default routing of two flows

How (MPLS) Networks Work

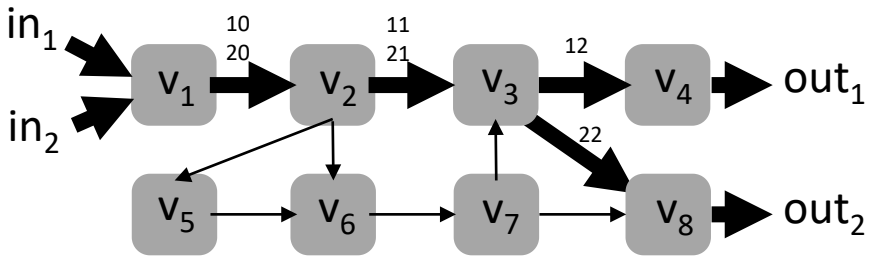
- Forwarding based on top label of label stack



Default routing of two flows

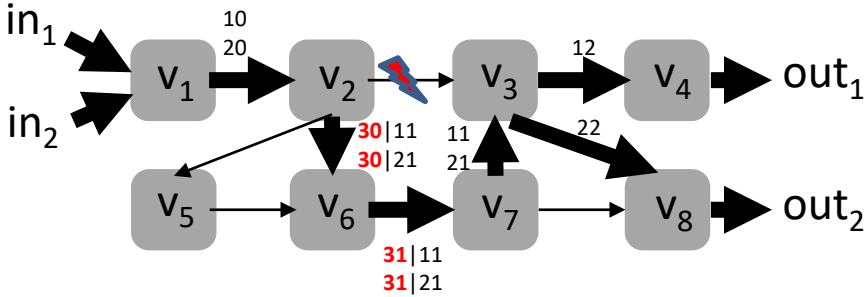
Fast Reroute Around *1 Failure*

- Forwarding based on **top label** of label **stack** (in packet header)



Default routing of two flows

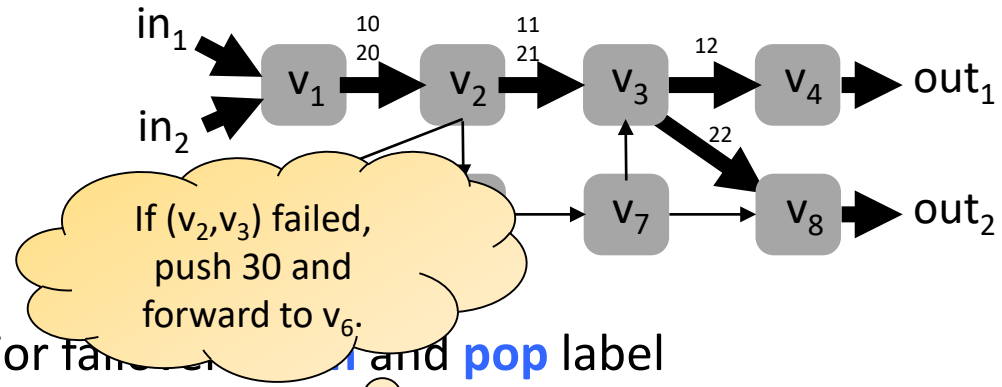
- For failover: **push** and **pop** label



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

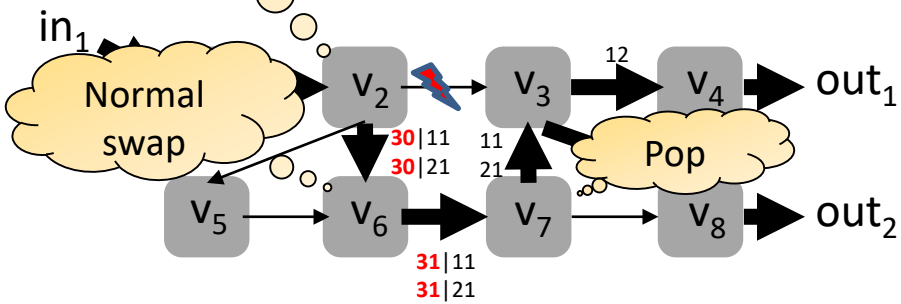
Fast Reroute Around *1 Failure*

- Forwarding based on **top label** of label **stack** (in packet header)



Default routing of two flows

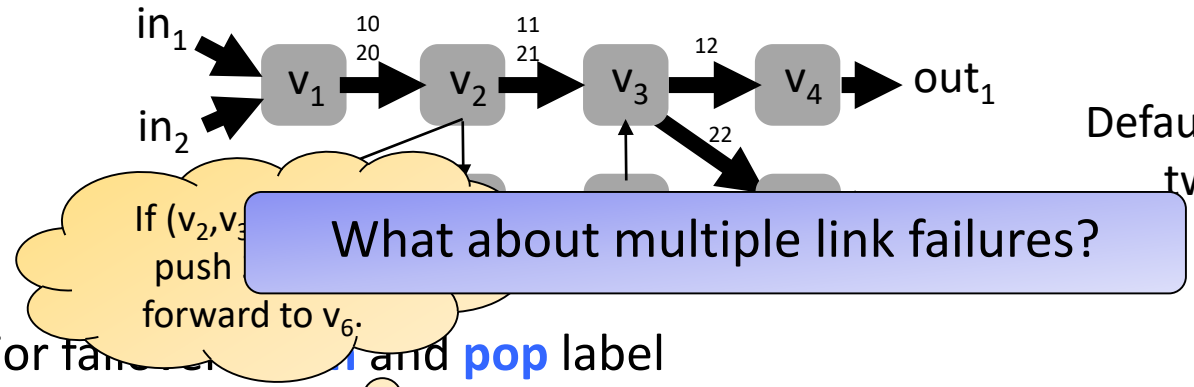
- For fai[illegible] and **pop** label



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

Fast Reroute Around *1 Failure*

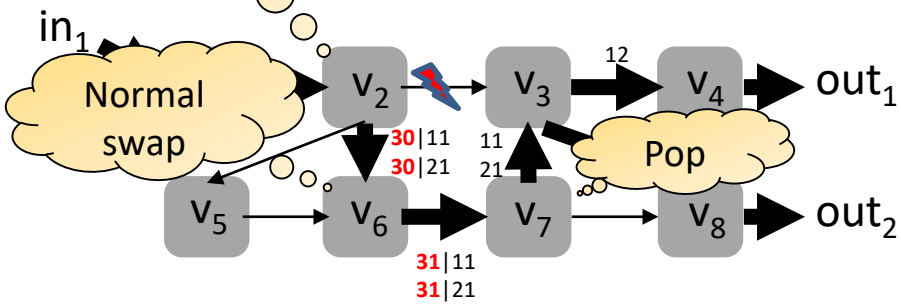
- Forwarding based on **top label** of label **stack** (in packet header)



What about multiple link failures?

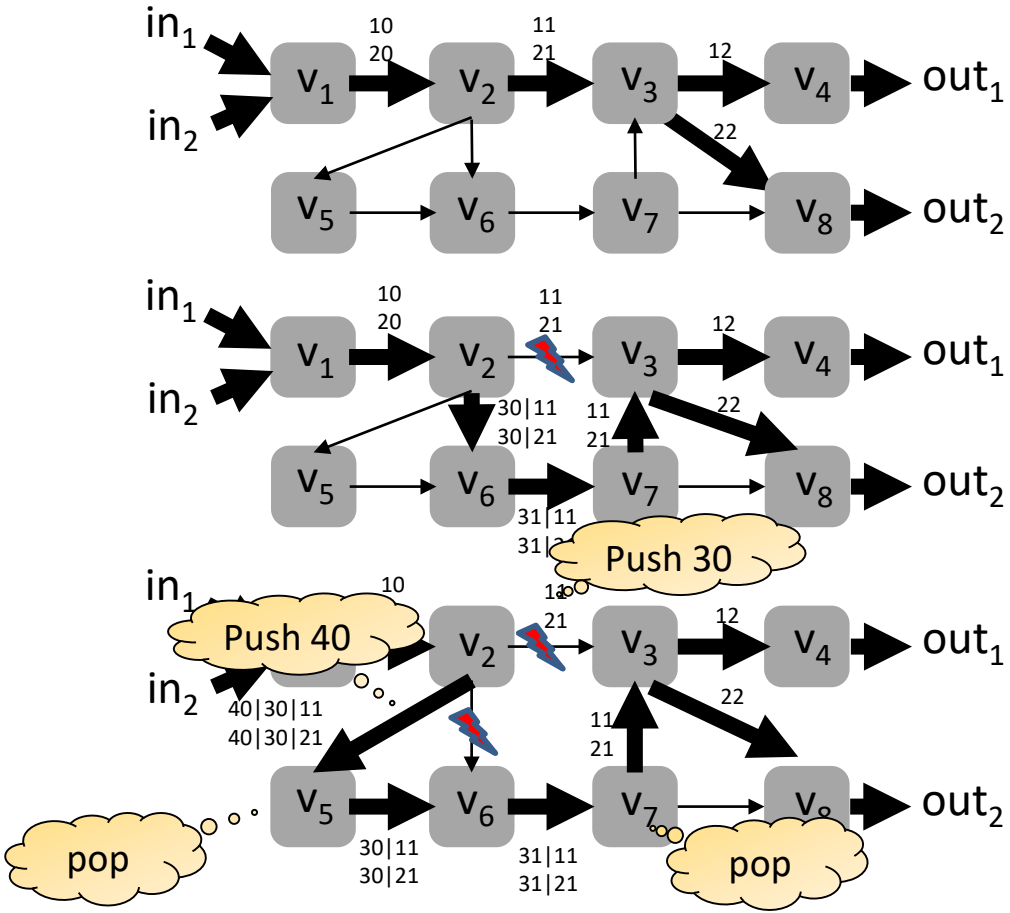
Default routing of two flows

- For fai… …and **pop** label



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

2 Failures: Push *Recursively*



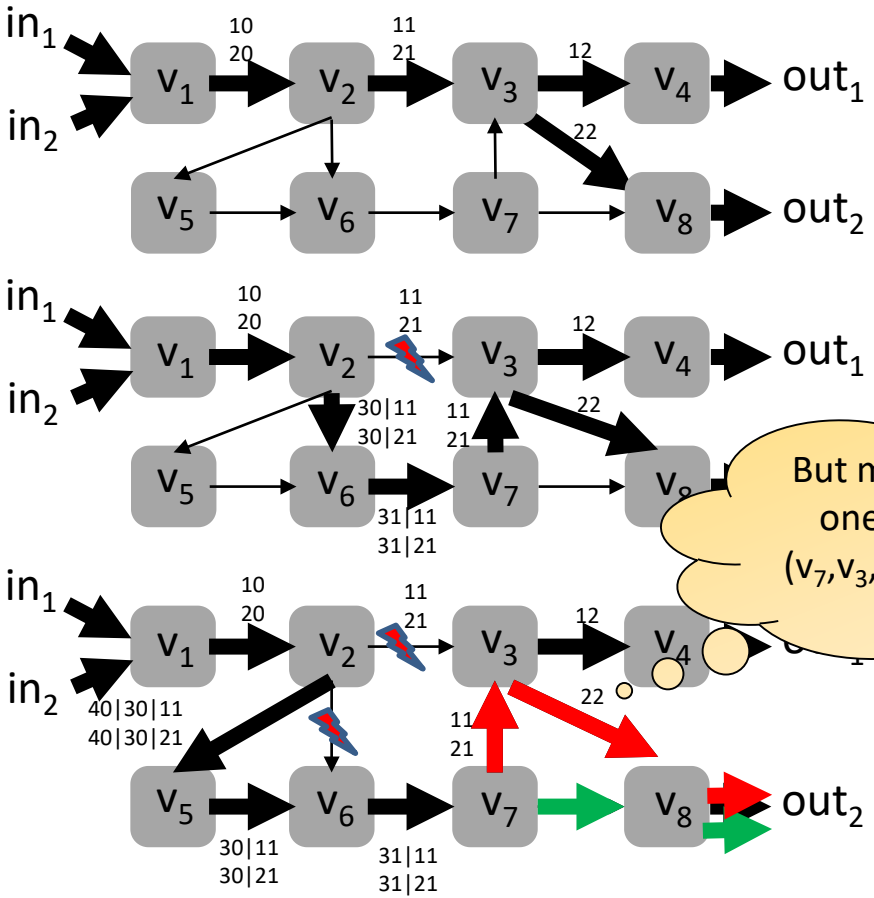
Original Routing

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)

2 Failures: Push *Recursively*



Original Routing

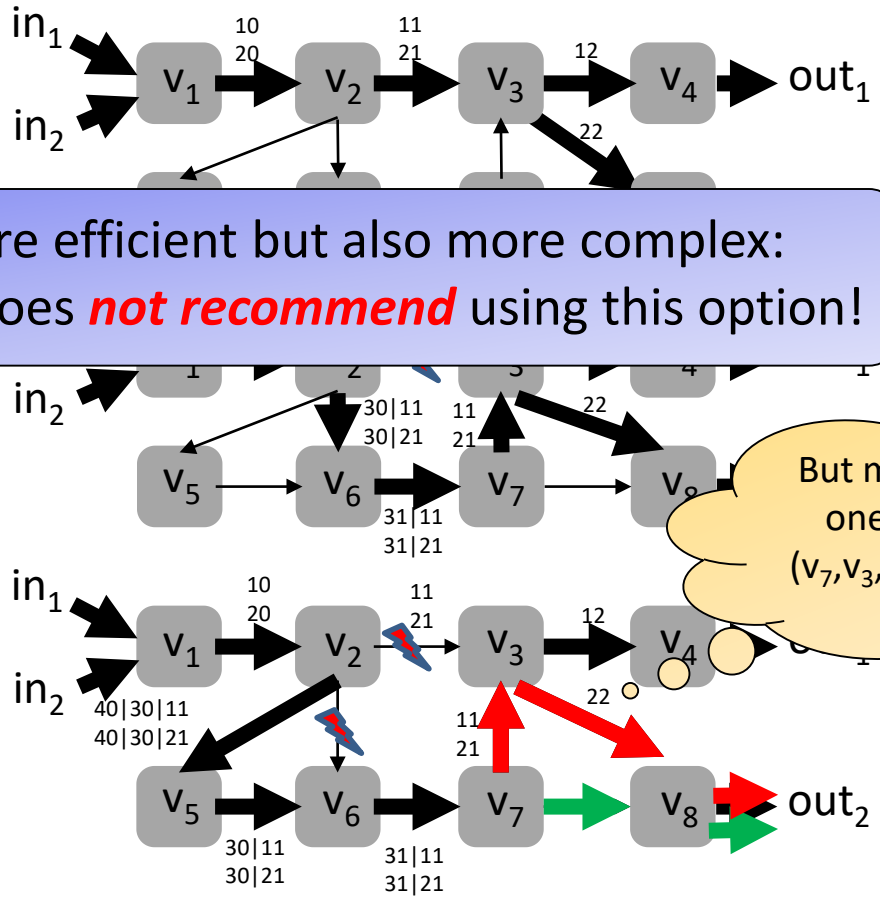
One failure: push 30:

But masking links one-by-one can be inefficient: (v_7,v_3,v_8) could be shortcut to (v_7,v_8).

…sh 30: route around (v_2,v_3)

Push recursively 40: route around (v_2,v_6)

2 Failures: Push *Recursively*



More efficient but also more complex:
Cisco does ***not recommend*** using this option!

Original Routing

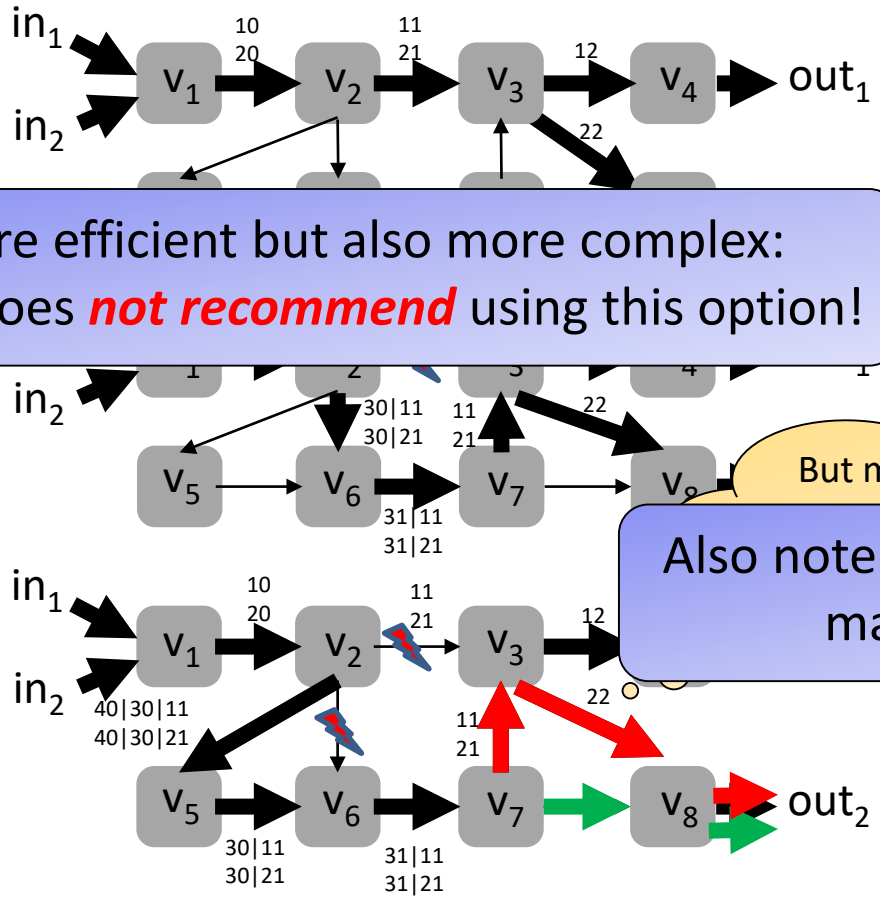
One failure: push 30:

But masking links one-by-one can be inefficient: (v_7,v_3,v_8) could be shortcut to (v_7,v_8).

…sh 30: route around (v_2,v_3)

Push recursively 40: route around (v_2,v_6)

2 Failures: Push *Recursively*



More efficient but also more complex:
Cisco does ***not recommend*** using this option!

Original Routing

One failure: push 30:

But masking links one-by-

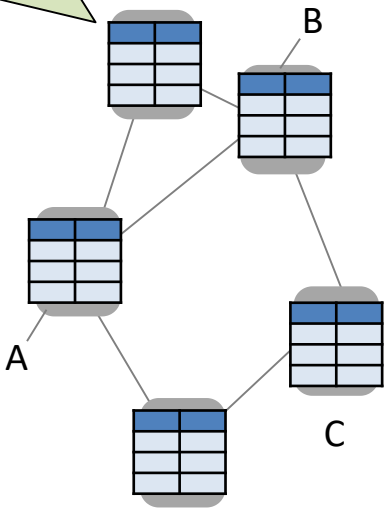
around (v_2,v_3)

Also note: due to push, ***header size*** may grow arbitrarily!

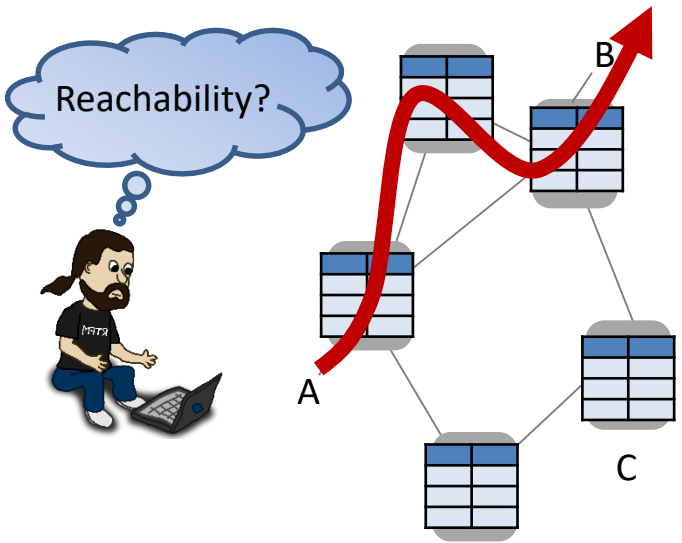
Push recursively 40:
route around (v_2,v_6)

Responsibilities of a Sysadmin

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



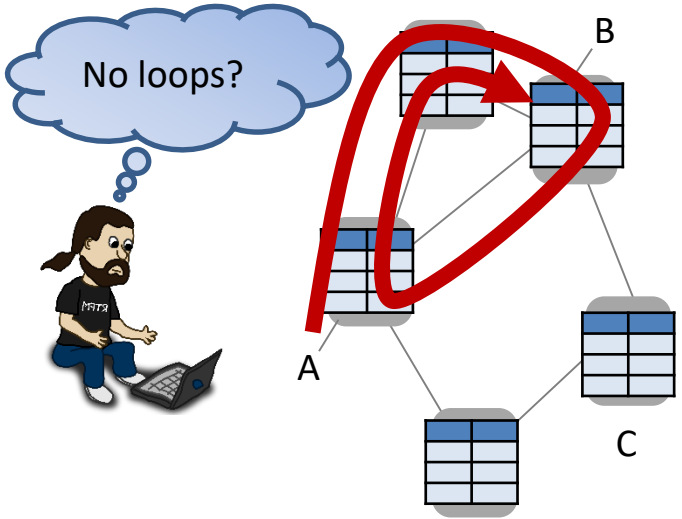
Responsibilities of a Sysadmin



Sysadmin responsible for:

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

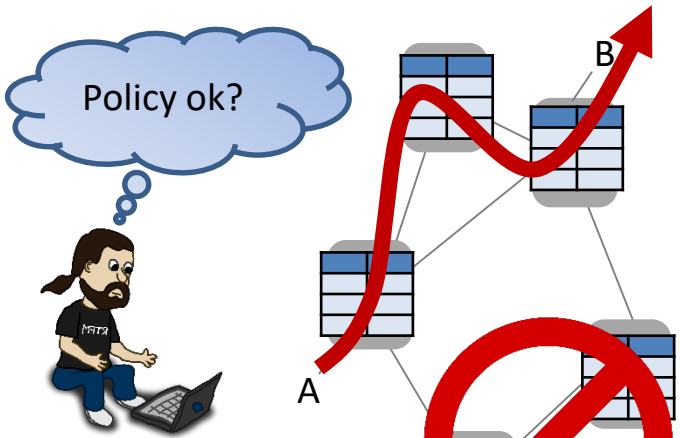
Responsibilities of a Sysadmin



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

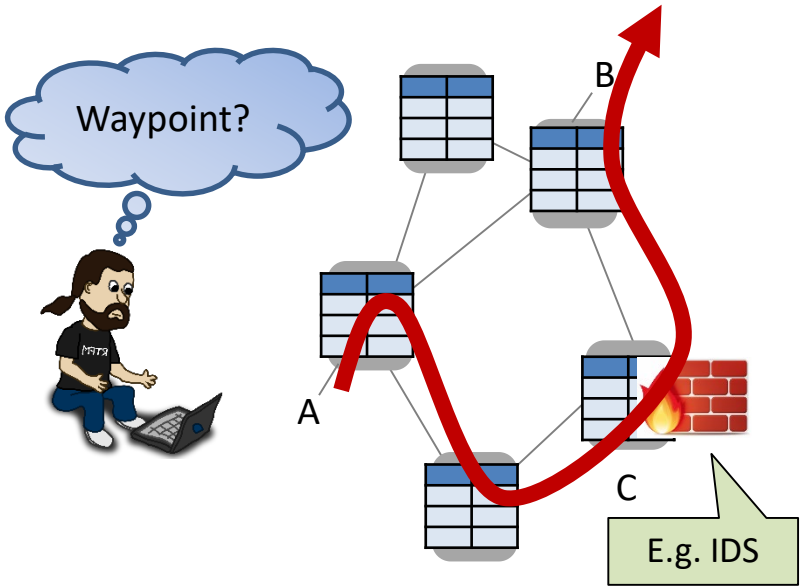


E.g. ***NORDUnet***: no traffic via Iceland (expensive!).

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?

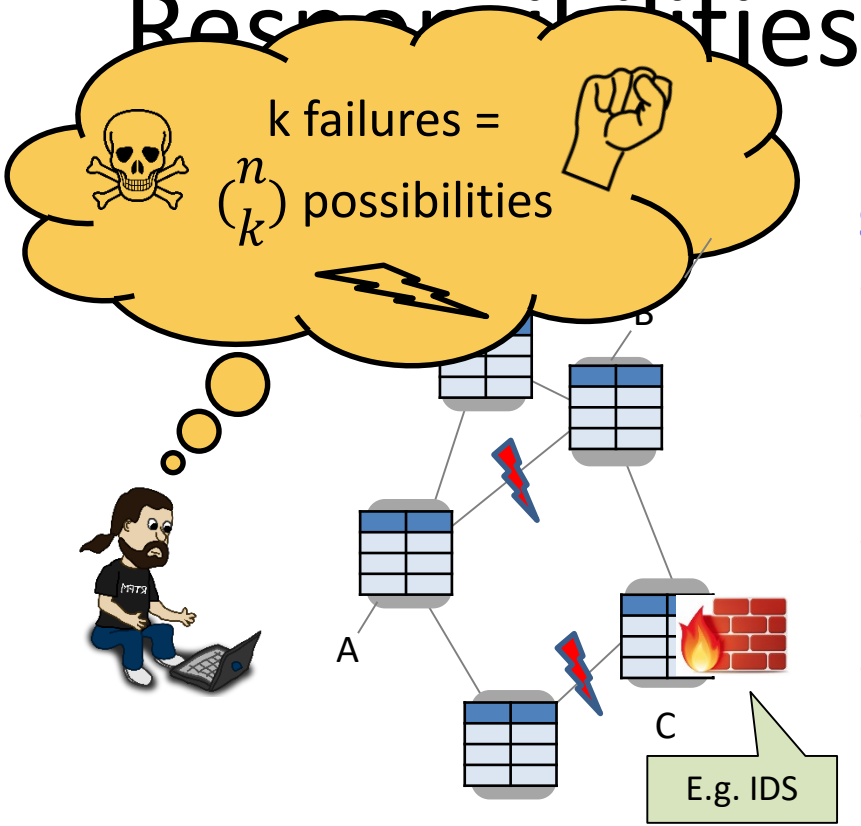
Responsibilities of a Sysadmin



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 (e.g., intrusion detection system or a firewall)?

Responsibilities of a Sysadmin

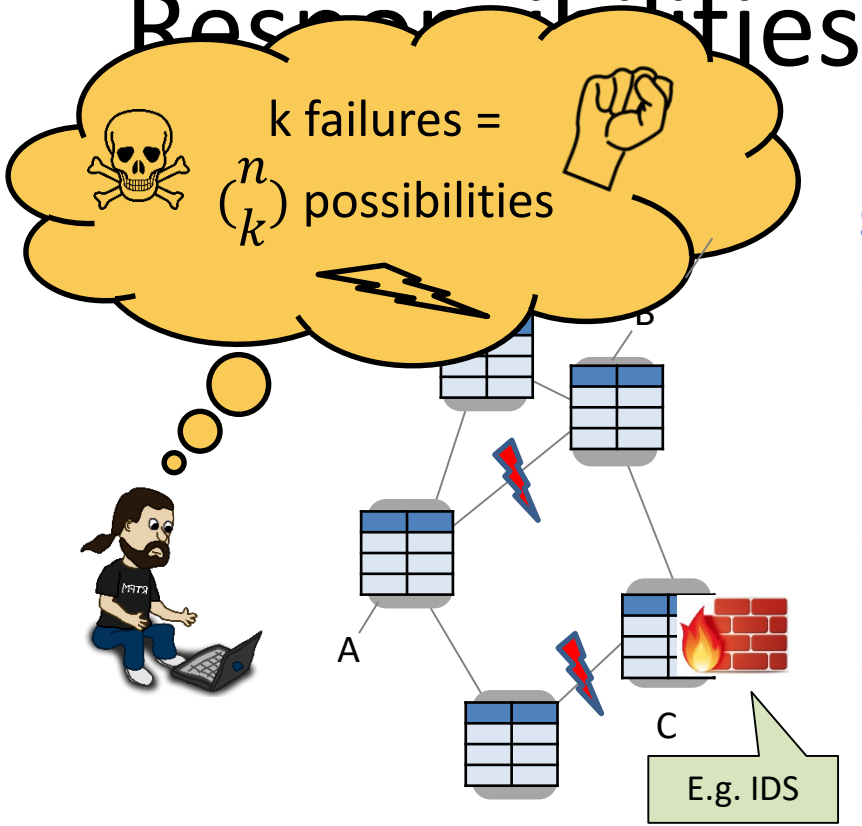


Sysadmin responsible for:

- **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 (e.g., intrusion detection system or a firewall)?

… and everything even under multiple failures?!

Responsibilities of a Sysadmin



Sysadmin responsible for:

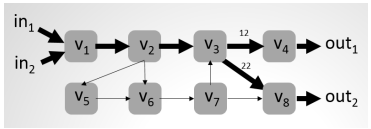
- **Reachability:** Can traffic from ingress port A reach egress port B?
- **Loop-freedom:** Are the routes implied by the forwarding rules loop-free?
- **Policy:** Is it ensured that traffic from A to B never goes via C?
- **Waypoint enforcement:** Is it ensured that traffic from A to B is always routed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

… and everything even under multiple failures?!

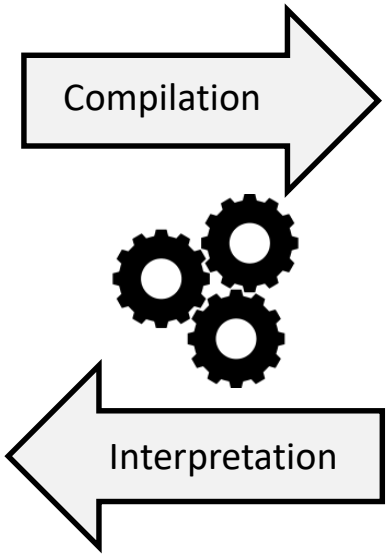
Generalization: service chaining!

Approach: Automation and Formal Methods

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)	$push(40)$



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

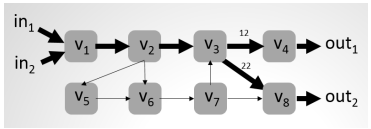
Router **configurations**
(Cisco, Juniper, etc.)

Pushdown Automaton and
Prefix Rewriting Systems

Approach: Automa…ods

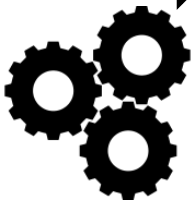
Use cases: Sysadmin ***issues queries*** to test certain properties, or do it on a ***regular basis*** automatically!

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)	$push(40)$

Compilation



Interpretation

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

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

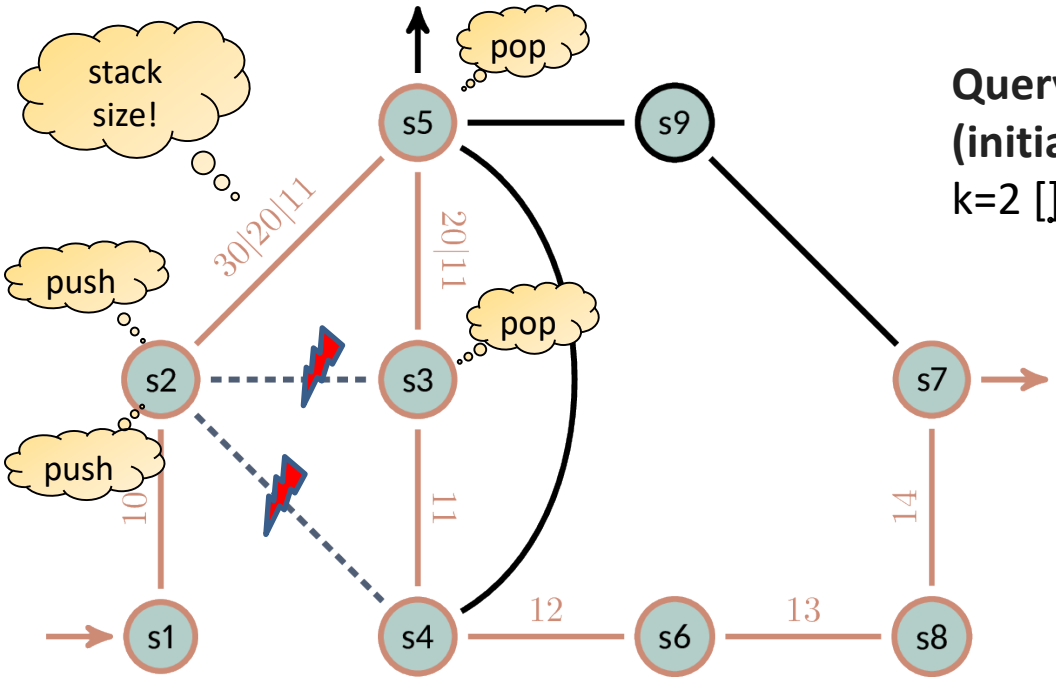
Pushdown Automaton and **Prefix Rewriting Systems**

AalWiNes

Online demo: https://demo.aalwines.cs.aau.dk/
Source code: https://github.com/DEIS-Tools/AalWiNes

Example

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



Query: 3 regular expressions (initial and final header, route)

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

2 failures

YES

(Polynomial time!)

Why AalWiNes is Fast (Polytime): Automata Theory

- For fast verification, we can use the result by **Büchi**: the set of all reachable configurations of a pushdown automaton a is regular set
- We hence simply use Nondeterministic Finite Automata (NFAs) when reasoning about the pushdown automata
- The resulting **regular operations** are all polynomial time

Julius Richard Büchi

1924-1984

Swiss logician

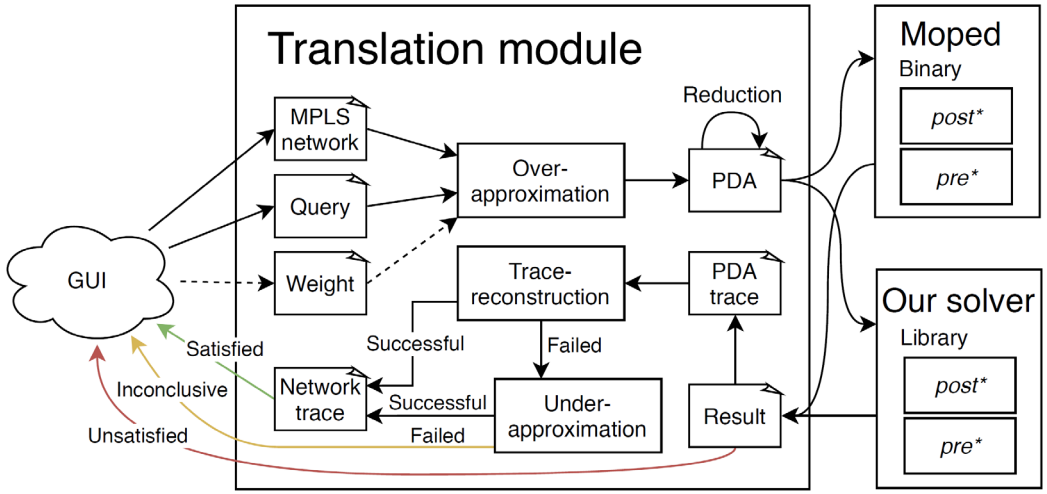
AalWiNes

Part 1: Parses query and constructs Push-Down System (PDS)

- In Python 3

Part 2: Reachability analysis of constructed PDS

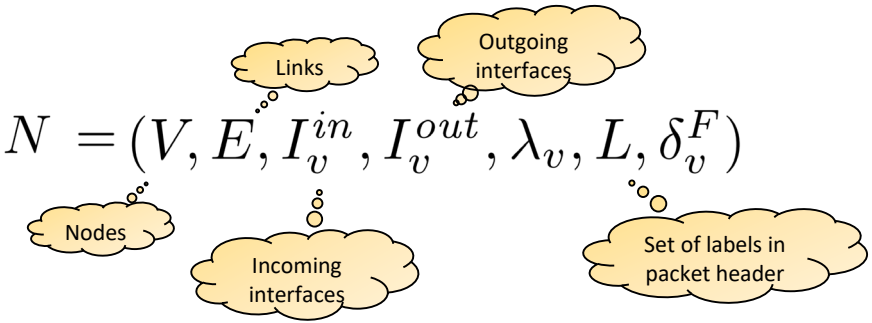
- Using ***Moped*** tool



Resp. our new weighted extension and much faster implementation in C++.

Network Model

- Network: a 7-tuple



Network Model

- Network: a 7-tuple

$$N = (V, E, I_v^{in}, I_v^{out}, \lambda_v, L, \delta_v^F)$$

Interface function

Interface function: maps outgoing interface to next hop node and incoming interface to previous hop node

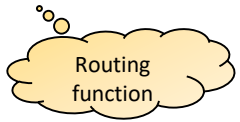
$$\lambda_v : I_v^{in} \cup I_v^{out} \rightarrow V$$

That is: $(\lambda_v(in), v) \in E$ and $(v, \lambda_v(out)) \in E$

Network Model

- Network: a 7-tuple

$$N = (V, E, I_v^{in}, I_v^{out}, \lambda_v, L, \delta_v^F)$$



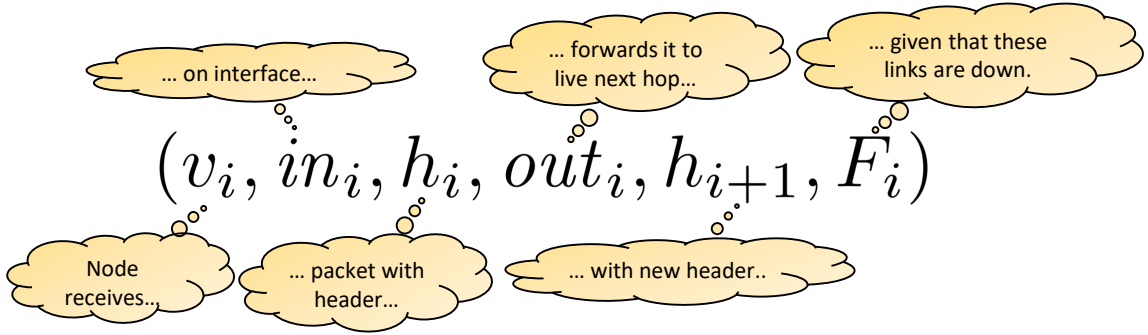
Routing function: for each set of failed links $F \subseteq E$, the routing function

$$\delta_v^F : I_v^{in} \times L^* \rightarrow 2^{(I^{out} \times L^*)}$$

defines, for all incoming interfaces and packet headers, outgoing interfaces together with modified headers.

Routing

Packet routing sequence can be represented using sequence of tuples:

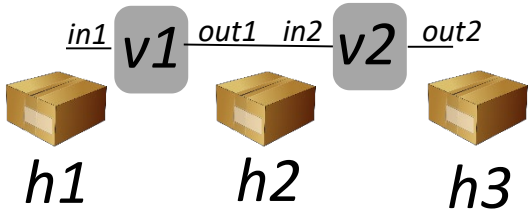


- Example: **routing** (in)finite sequence of tuples

$$(v_1, in_1, h_1, out_1, h_2, F_1),$$

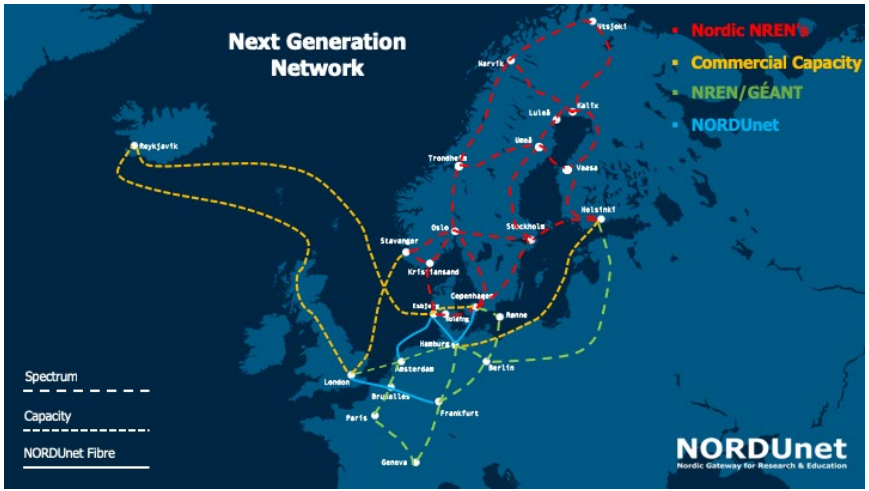
$$(v_2, in_2, h_2, out_2, h_3, F_2),$$

$$\ldots$$



Case Study: NORDUnet

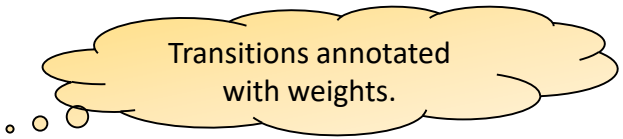
- Regional service provider
- **24 MPLS routers** geographically distributed across several countries
- Running **Juniper** operating system
- More than 30,000 labels
- Ca. **1 million** forwarding rules in our model
- For most queries of operators: answer *within seconds*



Generalizes to Quantitative Properties

- AalWiNes can also be used to test **quantitative properties**

- If query is satisfied, find trace that minimizes:
  - **Hops**
  - Latency (based on a latency value per link)
  - Tunnels



- Approach: **weighted** pushdown automata
  - Fast *poly-time algorithms* exist also for weighted pushdown automata (area of dataflow analysis)
  - Indeed, experiments show: *acceptable overhead* of weighted (quantitative) analysis

Roadmap

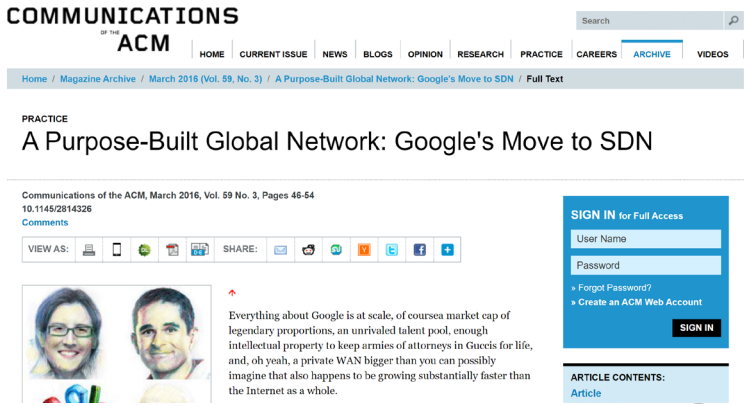
- **A Static Problem:** Policy Compliance *Under Failures*
  - AalWiNes: Fast Automated **What-if Analysis** for Networks (INFOCOM 2018, ACM CoNEXT 2018, ACM CoNEXT 2019, TACAS 2021)

- **A Dynamic Problem: Scheduling** Consistent Network *Updates*
  - Latte and quantitative extensions (PODC 2015, ICALP 2018, PERFORMANCE 2021)

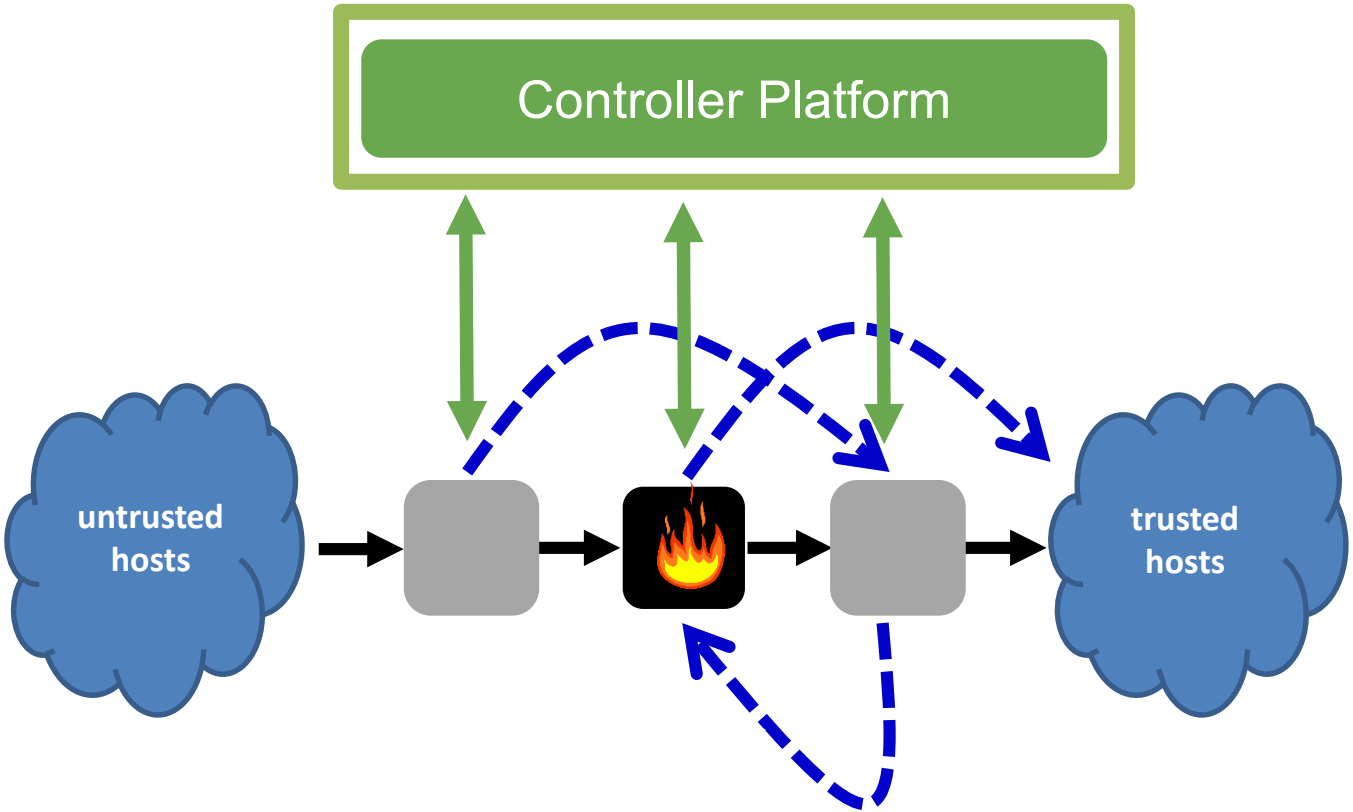
More Adaptable Networks

- Software-defined networking also enables networks to be more **adaptable**

- Attractive for:
  - Fine-grained *traffic engineering* (e.g., at Google)
  - Accounting for changes in the demand (*spatio-temporal structure*)
  - Security policy changes
  - Service relocation
  - *Maintenance* work
  - Link/node failures
  - …

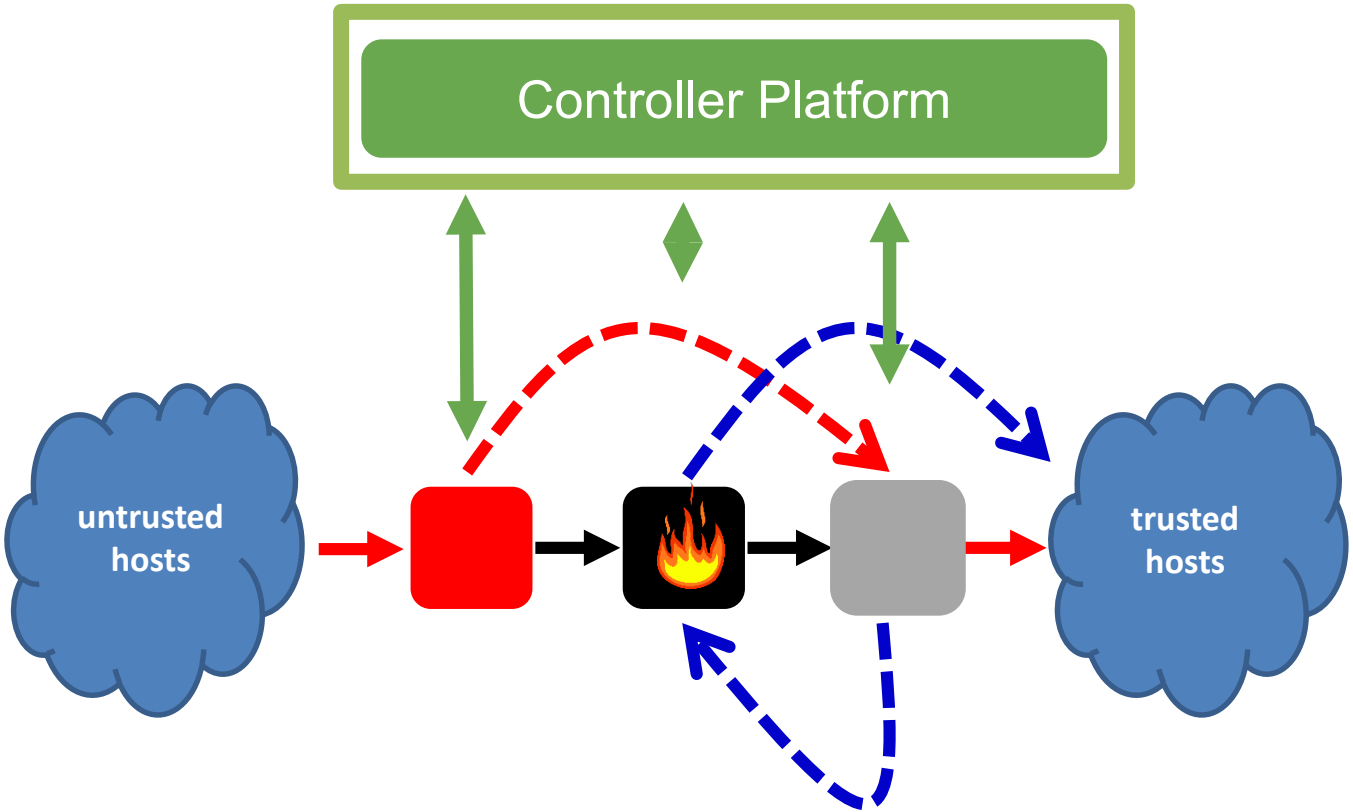


Introduces a New Challenge: Consistent Update



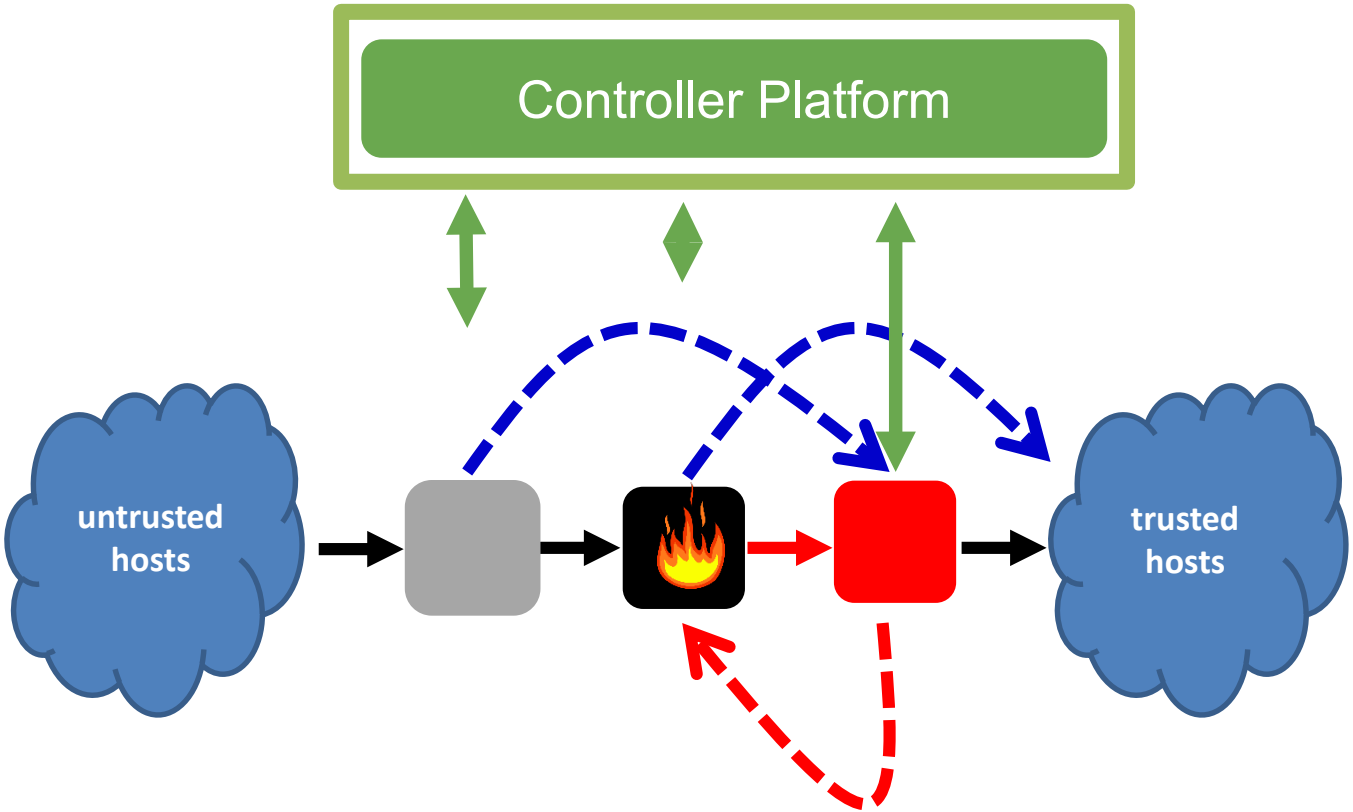
Invariant: Traffic from untrusted hosts to trusted hosts via **firewall**!

Introduces a New Challenge: Consistent Update



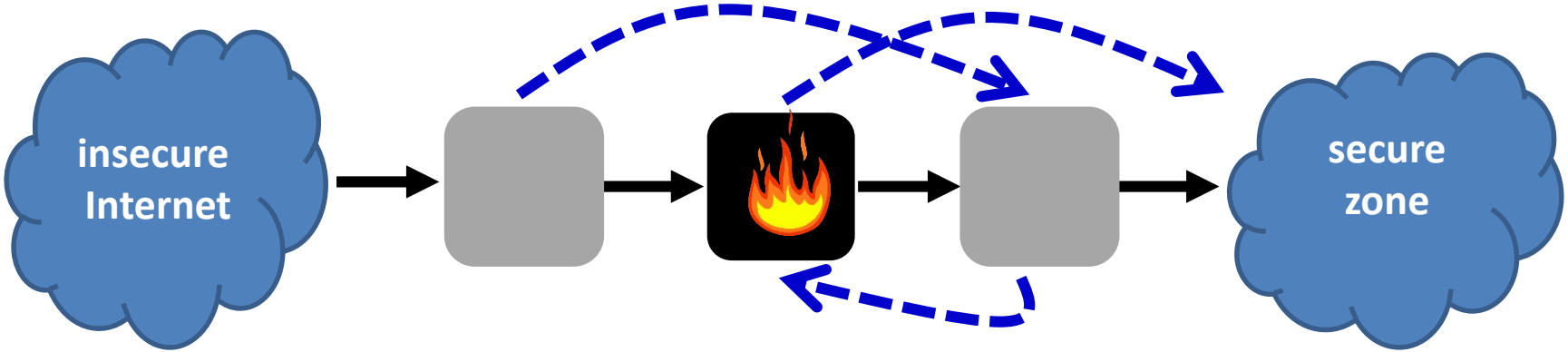
Invariant: Traffic from untrusted hosts to trusted hosts via **firewall**!

Introduces a New Challenge: Consistent Update

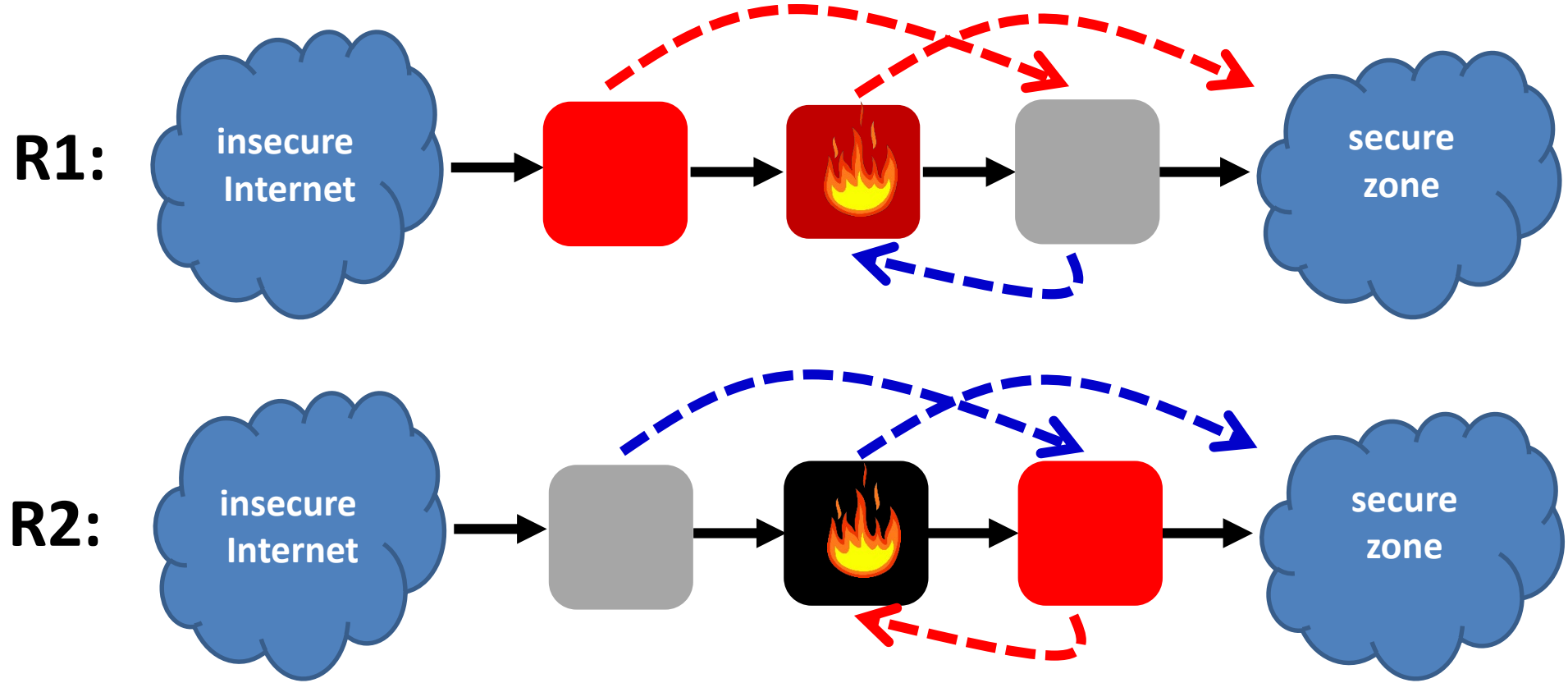


Invariant: Traffic from untrusted hosts to trusted hosts via **firewall**!

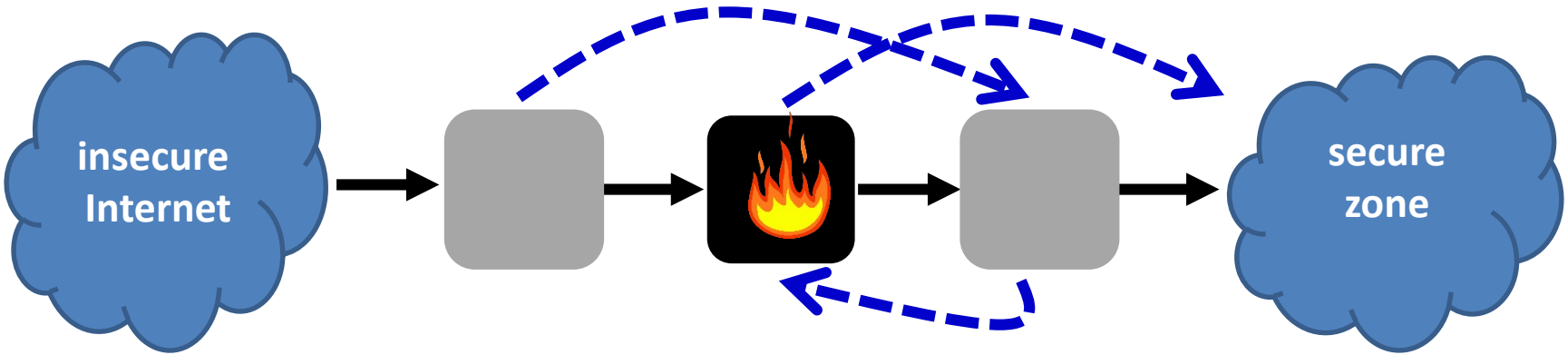
Question: How To Update Loop-Free?



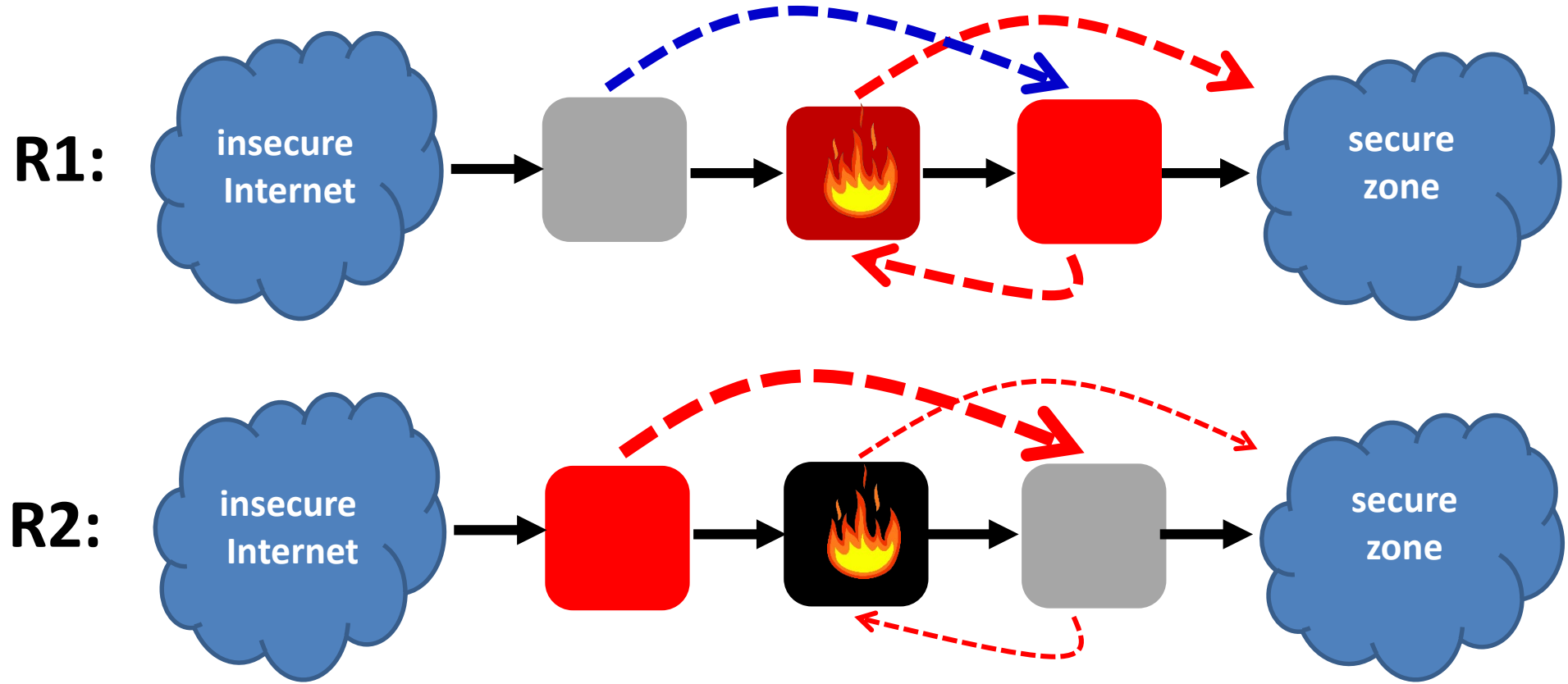
In 2 Rounds!


R1:
insecure Internet
secure zone
R2:
insecure Internet
secure zone

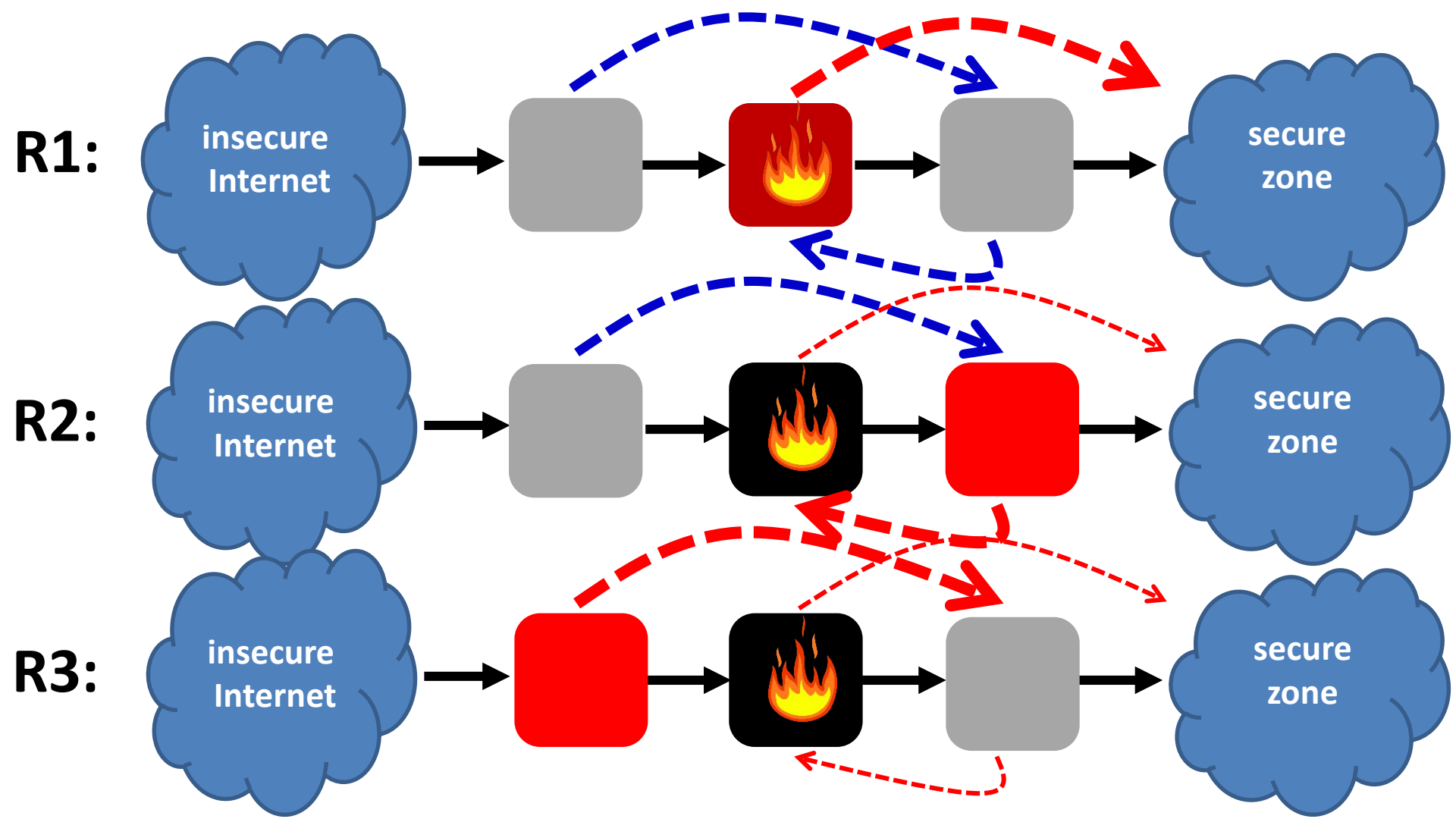
Background: How To Enforce Waypoint?



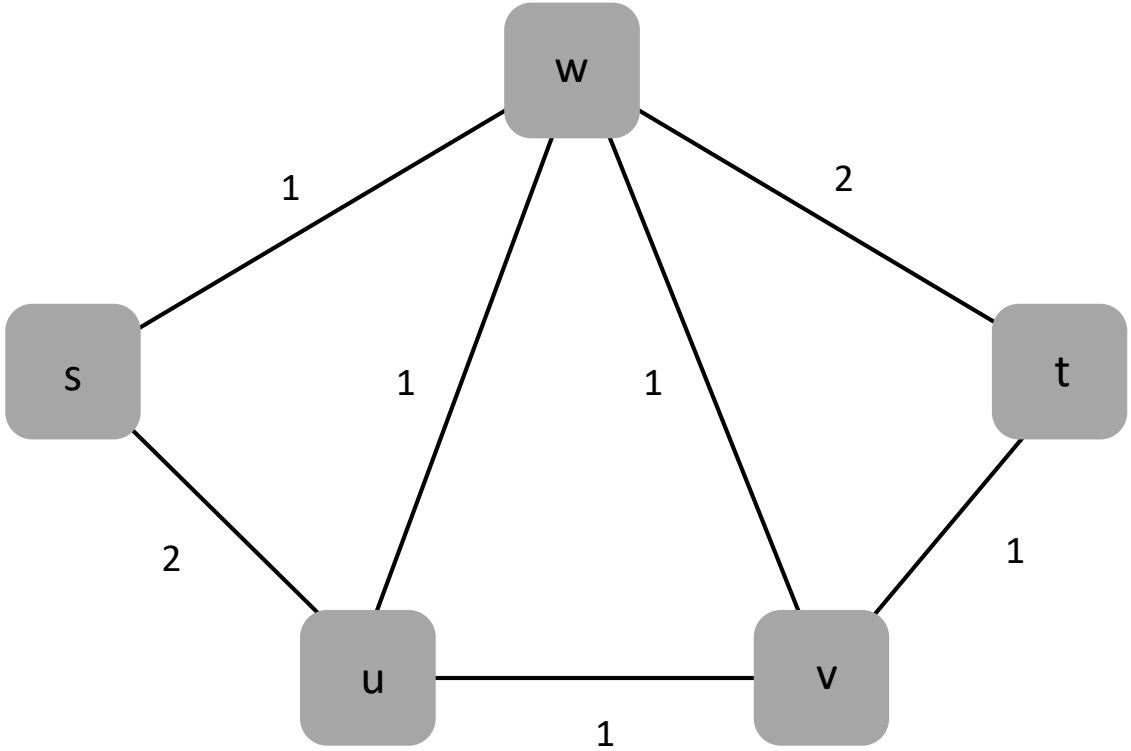
In 2 Rounds!


R1:
insecure Internet
secure zone
R2:
insecure Internet
secure zone

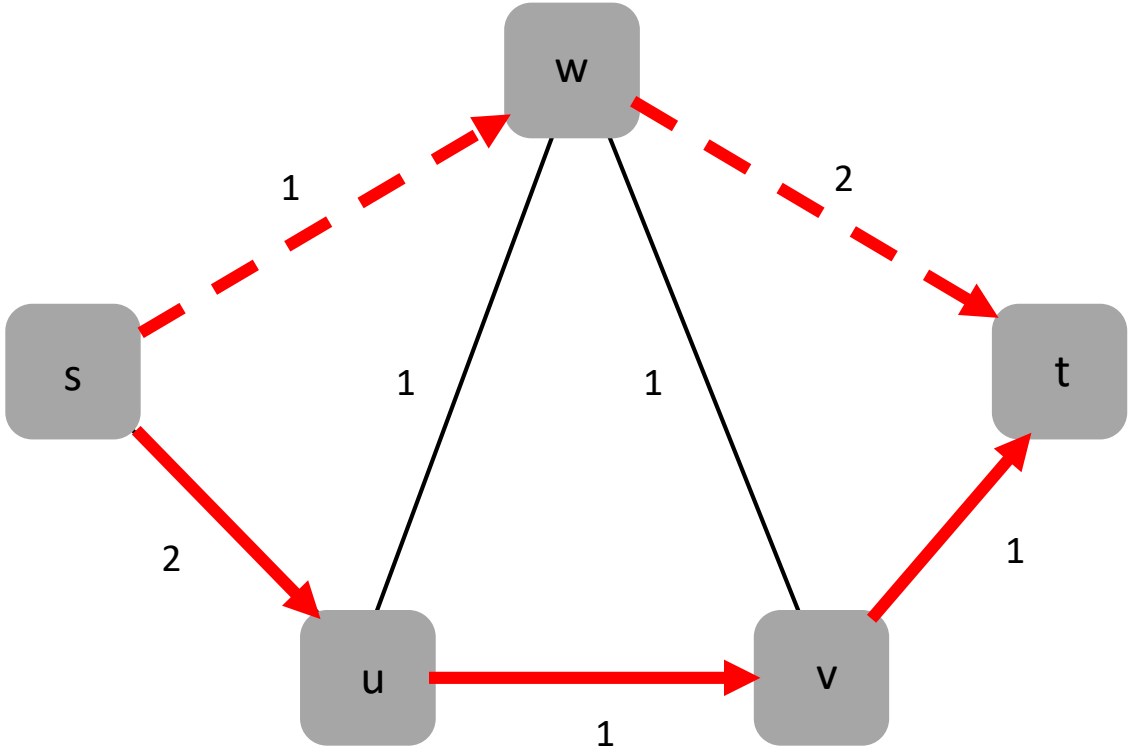
Loop-Free and Waypoint?
3 Rounds!


R1:
insecure Internet
secure zone
R2:
insecure Internet
secure zone
R3:
insecure Internet
secure zone

Accounting for Quantitative Aspects

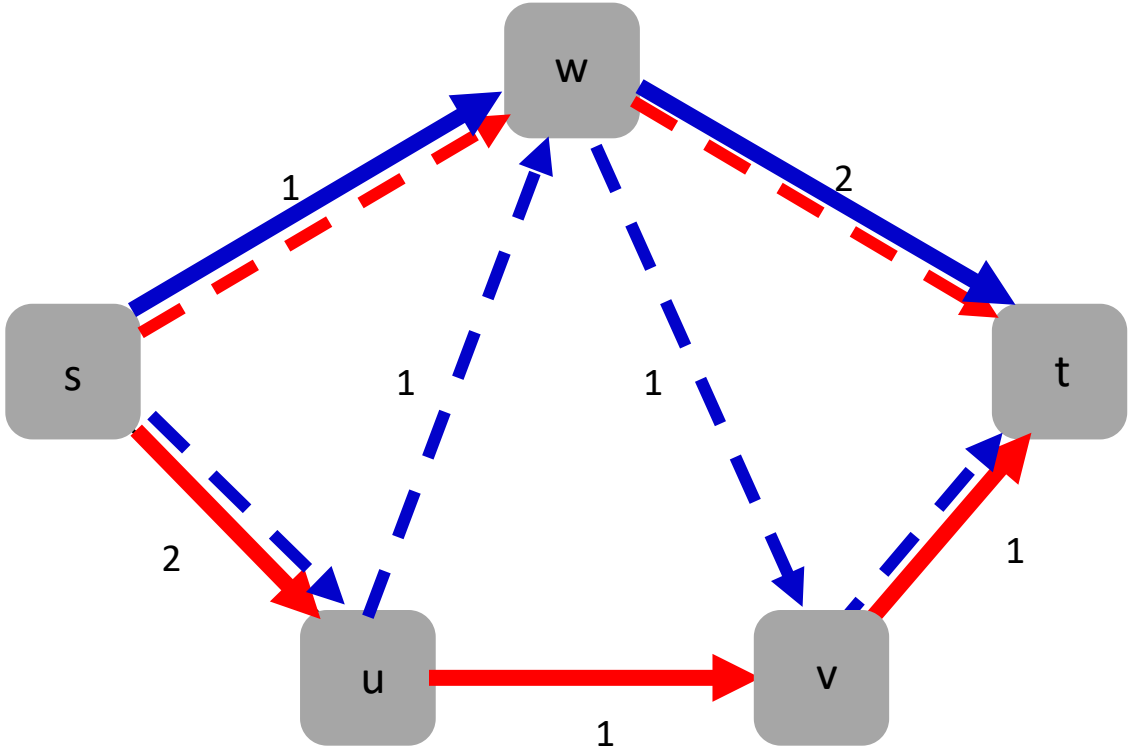


Accounting for Quantitative Aspects



Flow 1

Accounting for Quantitative Aspects

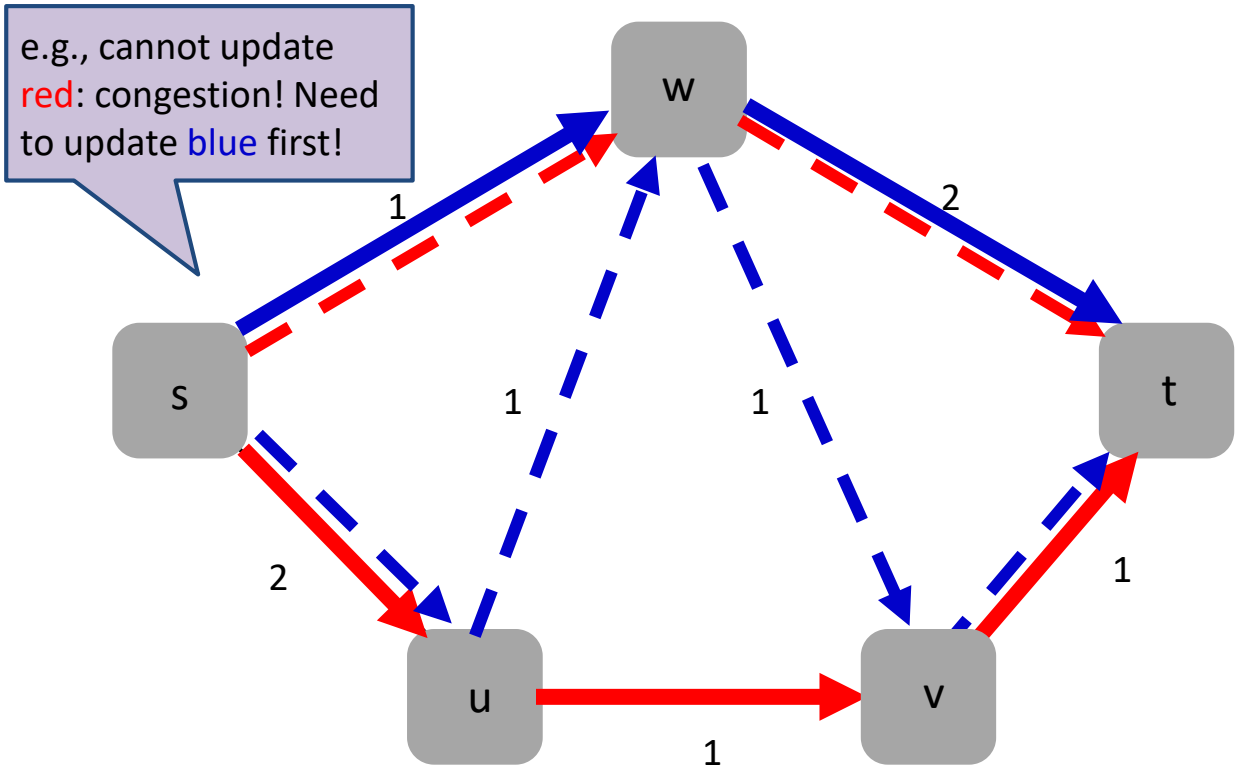


Can you find an update schedule?

Flow 1

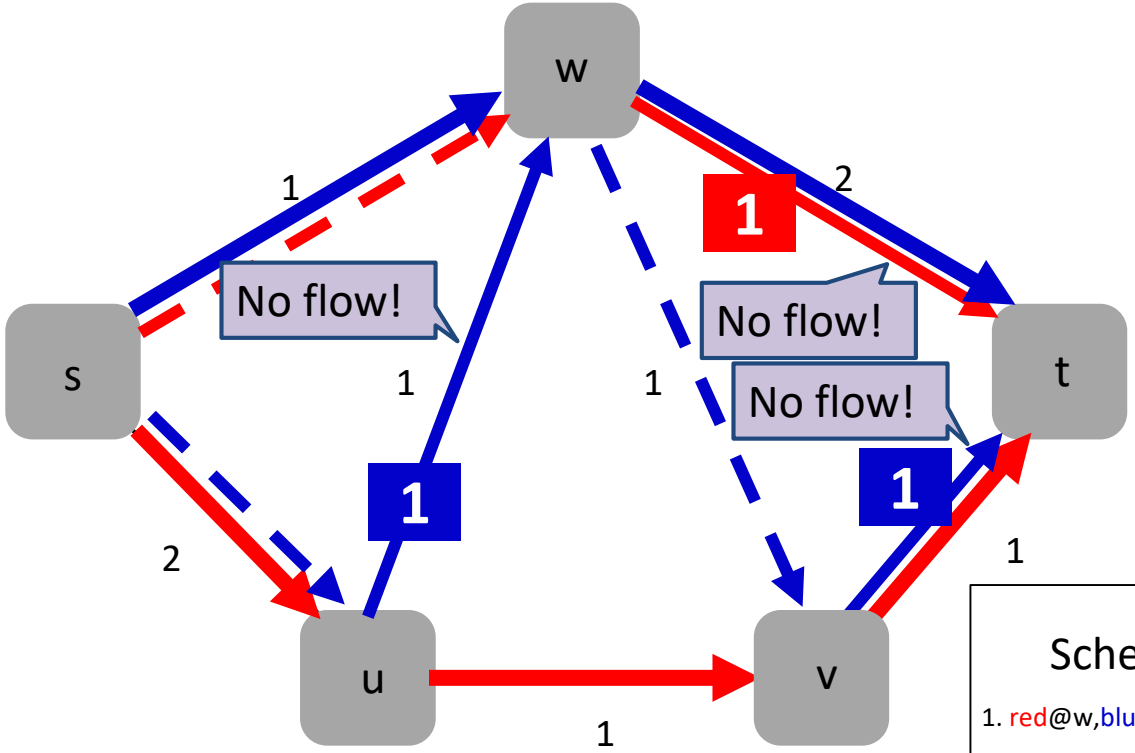
Flow 2

Accounting for Quantitative Aspects



Can you find an update schedule?

Accounting for Quantitative Aspects

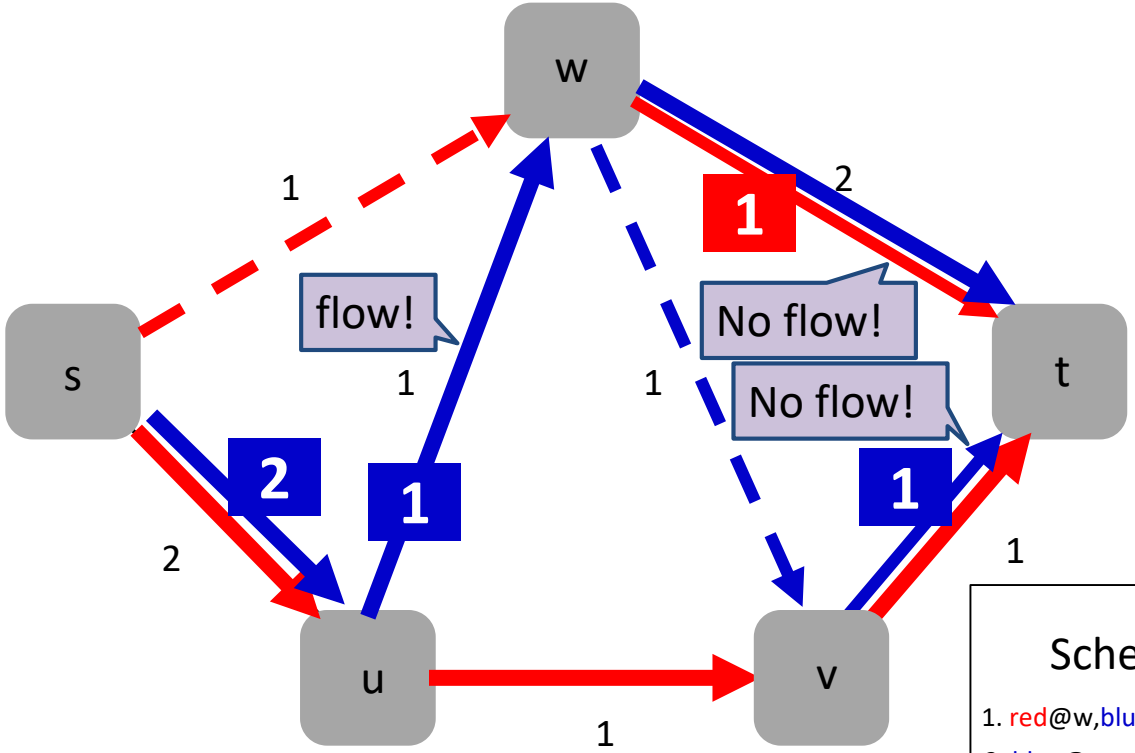


Round 1: prepare

Schedule:

1. red@w,blue@u,blue@v

Accounting for Quantitative Aspects

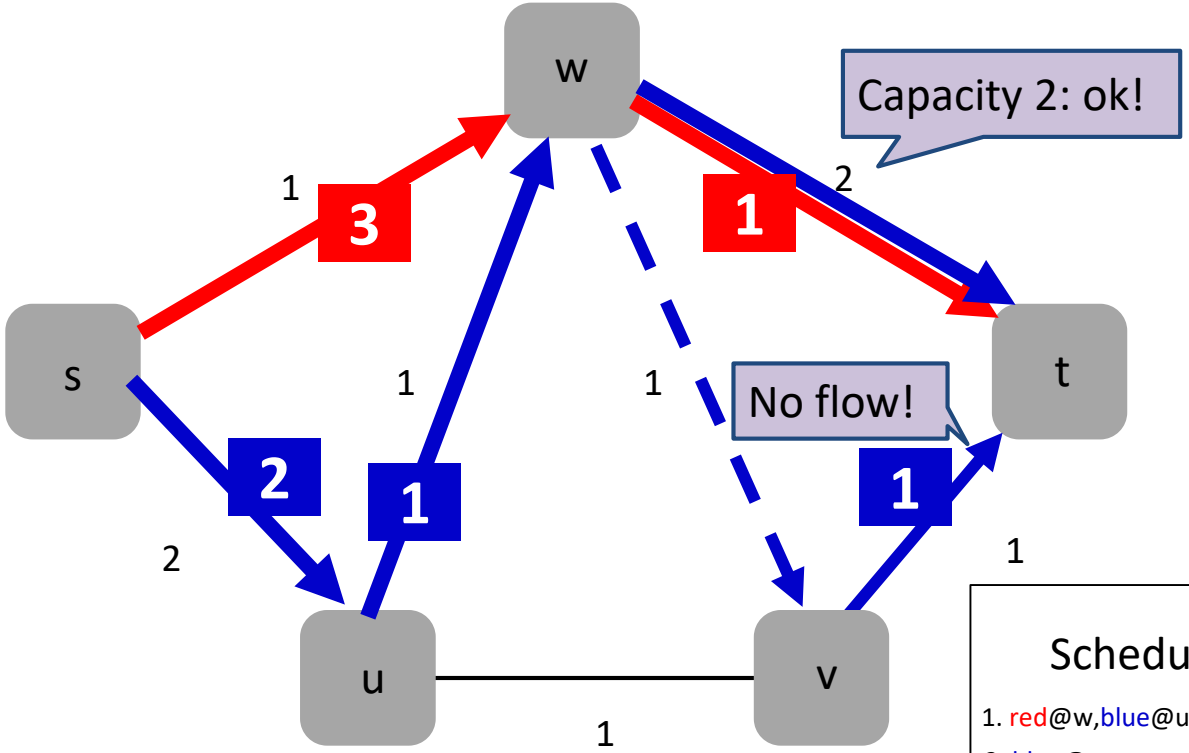


Round 2

Schedule:

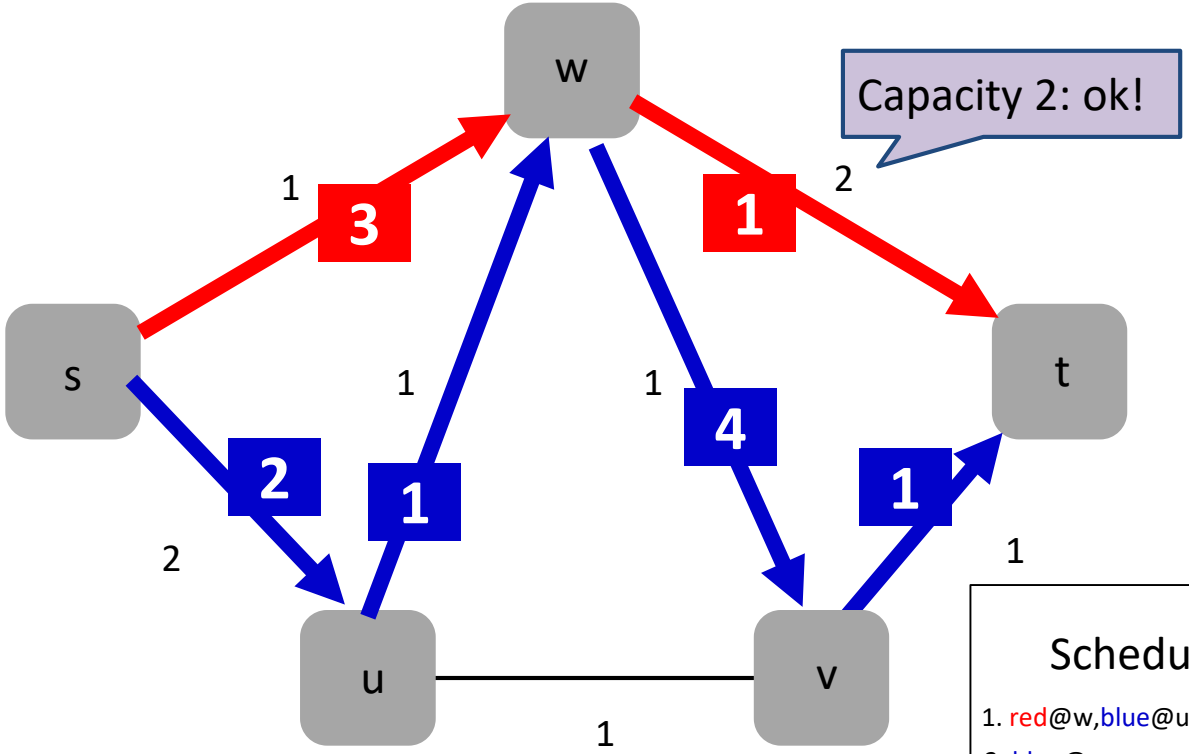
1. red@w,blue@u,blue@v
2. blue@s

Accounting for Quantitative Aspects



Round 3

Accounting for Quantitative Aspects

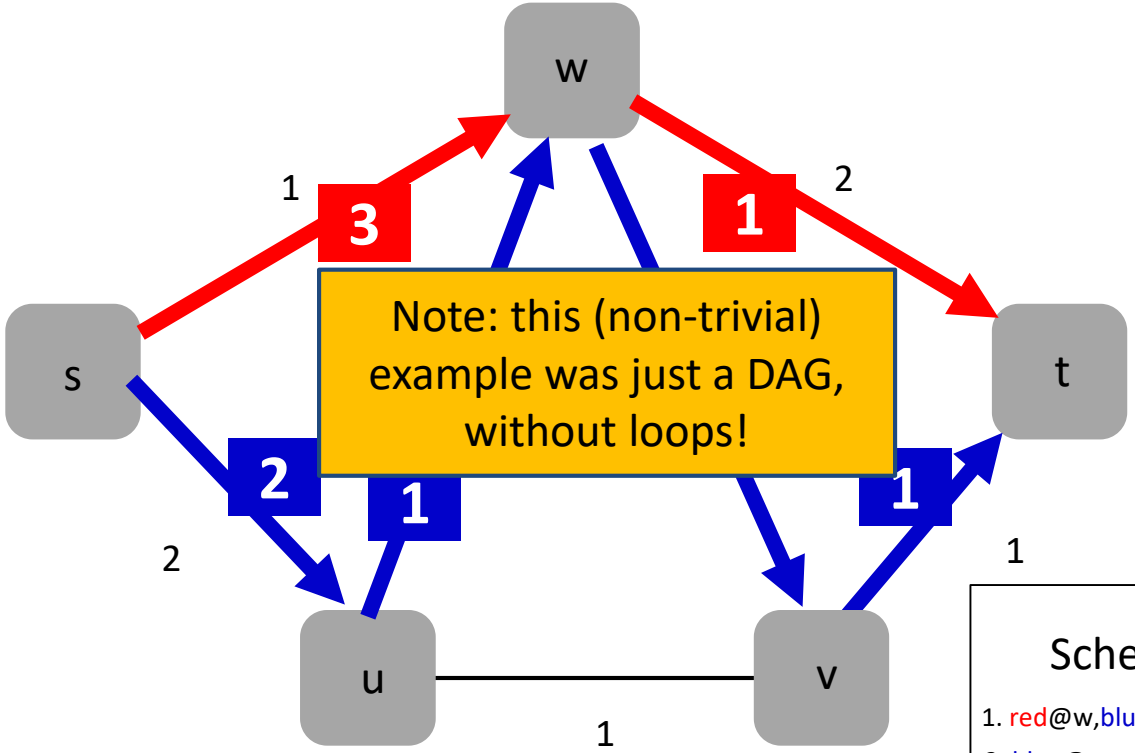


Round 4

Schedule:

1. red@w,blue@u,blue@v
2. blue@s
3. red@s
4. blue@w

Accounting for Quantitative Aspects



Round 4

Schedule:

1. red@w,blue@u,blue@v
2. blue@s
3. red@s
4. blue@w

Latte: Shortest Consistent Update Schedules

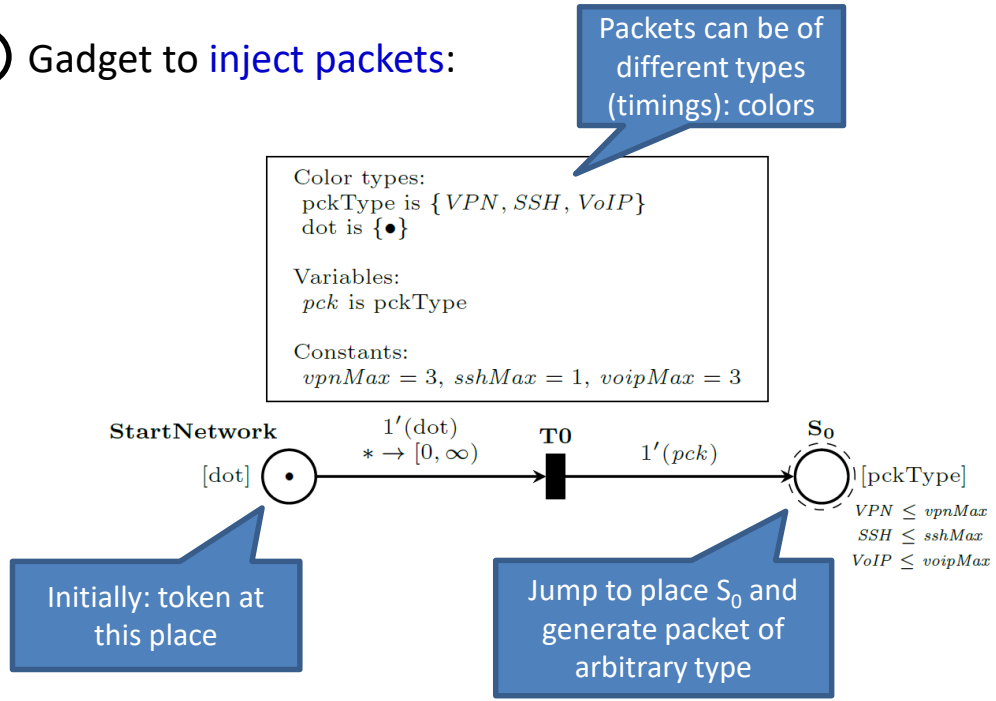
- A first approach: fast updates by accounting for temporal properties
  - E.g., different packet types have different *processing times*
  - Requires a fixed *update order* (e.g., produced by NetSynth)
  - Limited to loop-freedom and waypoint enforcement, and scheduling latency (no congestion)

- Based on petri nets: powerful modeling language for distributed systems
  - Configurations: tokens located at places

- Our extension: Timed-Arc Colored Petri Nets (TACPN)
  - Tokens also contain: *color* information (e.g., different packet *types*) and time information (e.g., modeling *age*)
  - Places and input arcs have *time constraints* for each color

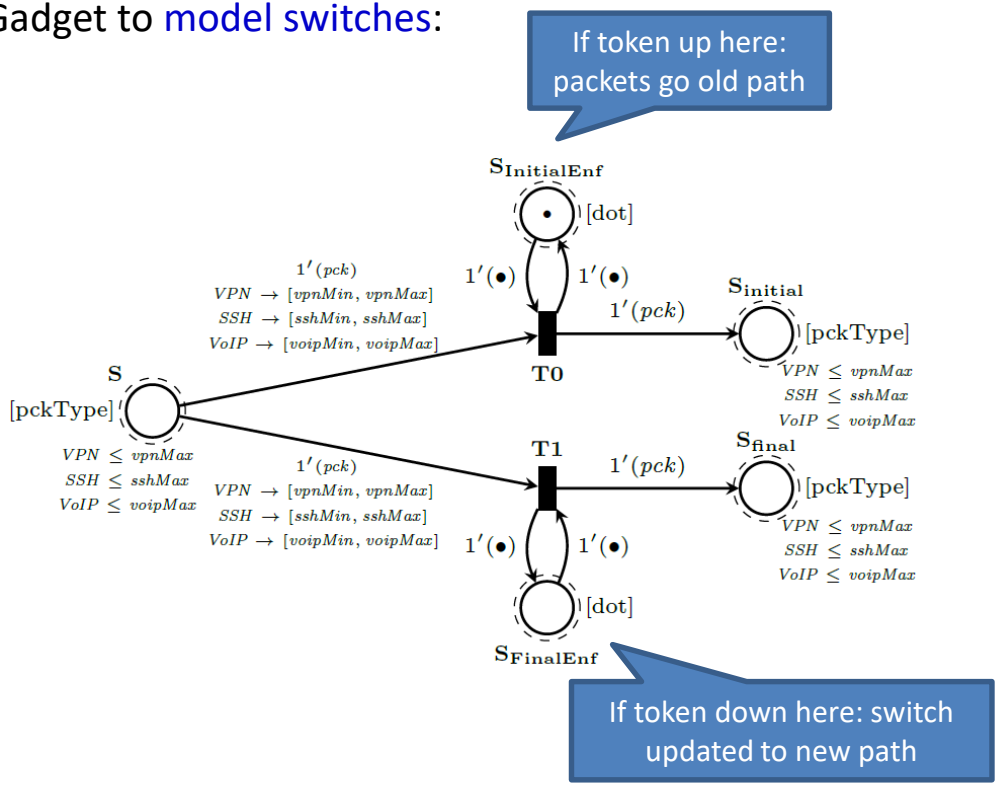
Example: Encoding Network Updates in TACPNs

① Gadget to inject packets:



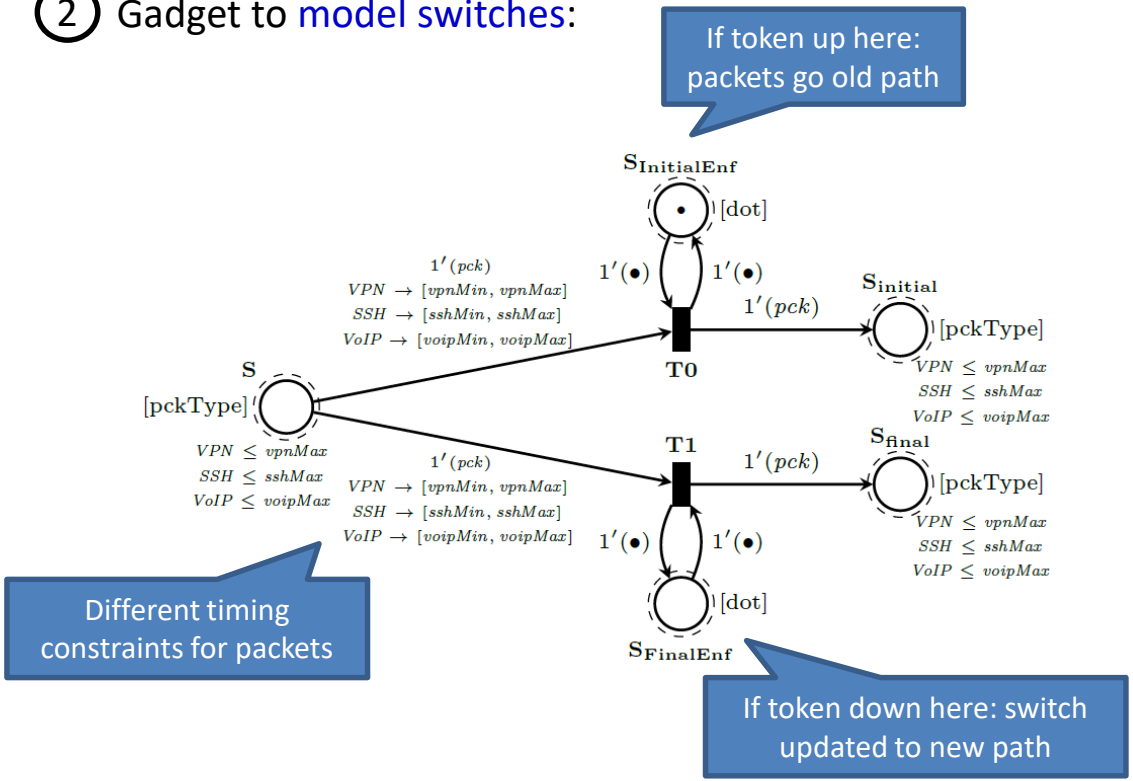
Example: Encoding Network Updates in TACPNs

② Gadget to model switches:



Example: Encoding Network Updates in TACPNs

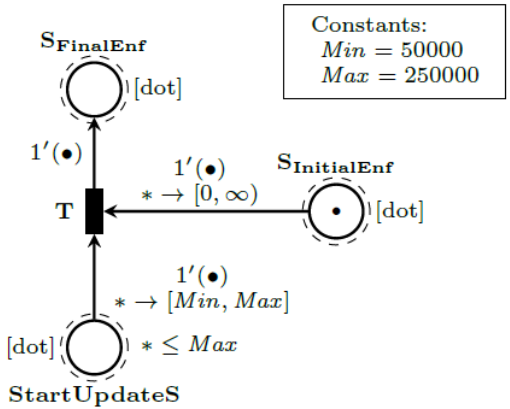
② Gadget to model switches:



Example: Encoding Network Updates in TACPNs

③ Gadget to model switch update:

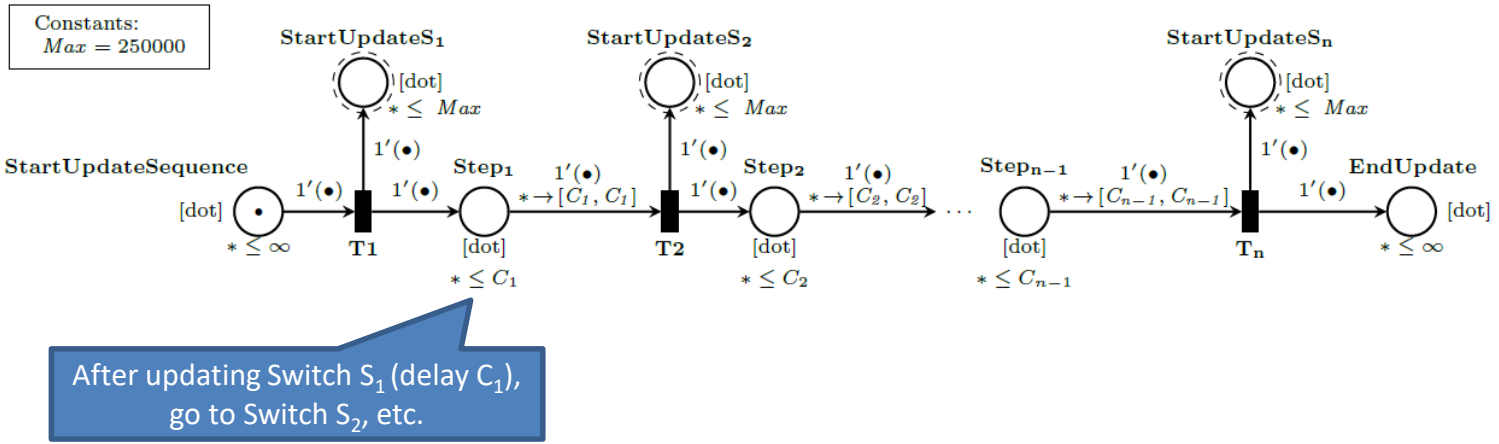
How to change between initial and final switch configuration



Starting here, the update can take time between min and max

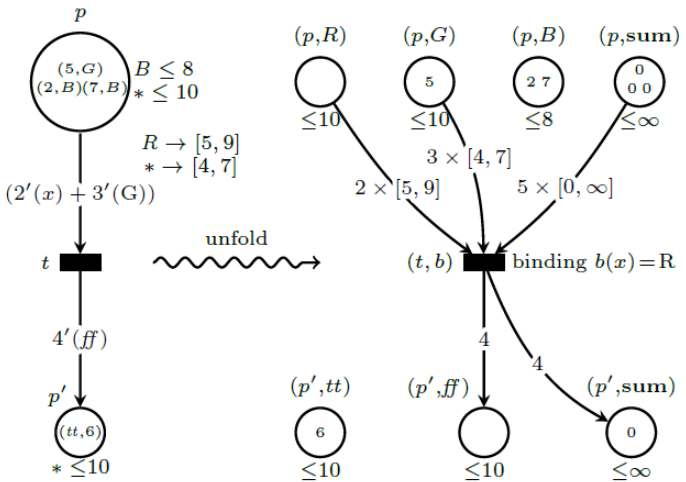
Example: Encoding Network Updates in TACPNs

④ Connecting the pieces: initialization of update sequence for all *n* switches



Analysis

The constructed nets can be analyzed efficiently via their *unfolding* into existing *timed-arc Petri nets*.



Preserves bisimilarity!

Improved Latency of Update Schedules

Network	Route length	Verification time[s]	Default update time [s]	Optimized update time [s]	Improvement [%]
TLex	4	0.74	3.58	0.25	92.30%
HiberniaIreland	5	1.02	6.05	0.28	95.50%
Harnet	6	1.42	9.08	0.28	96.97%
UniC	7	1.49	12.65	0.28	97.83%
Oxford	8	2.02	16.78	0.28	98.36%
Xeex	10	5.86	26.68	0.28	98.97%
Sunet	11	10.23	32.45	0.28	99.15%
SwitchL3	12	18.88	38.78	0.28	99.29%
Psinet	14	89.67	53.01	0.28	99.48%
Uunet	15	211.86	61.05	0.28	99.55%
Renater2010	16	480.52	69.58	0.28	99.60%
Missouri	25	timeout	171.05	67.10	60.77%
Syringa	35	timeout	336.05	295.35	12.11%
VtlWavenet2011	35	timeout	336.06	295.35	12.11%

- Network topologies from the Topology Zoo
- Experiments run on a 64-bit Ubuntu 18.04 laptop

Improved Latency of Update Schedules

Compared to conservative delays as produced by NetSynth: over 90% improvement.

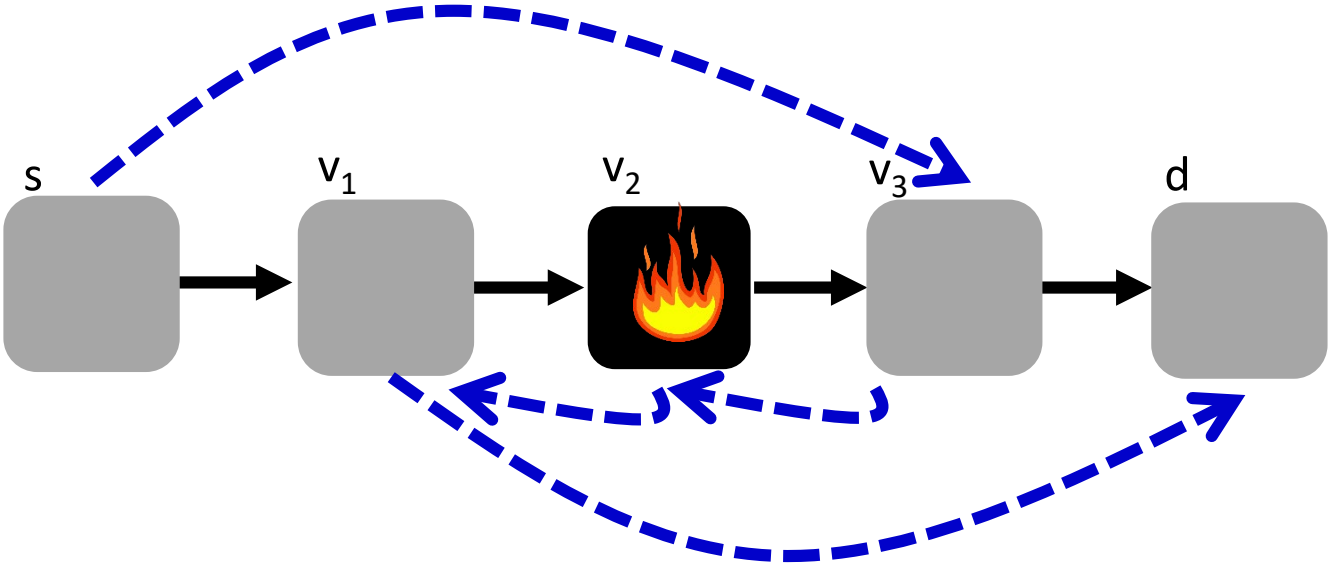
Network	Route length	Verification time[s]	Default update time [s]	Optimized update time [s]	Improvement [%]
TLex	4	0.74	3.58	0.25	92.30%
HiberniaIreland	5	1.02	6.05	0.28	95.50%
Harnet	6	1.42	9.08	0.28	96.97%
UniC	7	1.49	12.65	0.28	97.83%
Oxford	8	2.02	16.78	0.28	98.36%
Xeex	10	5.86	26.68	0.28	98.97%
Sunet	11	10.23	32.45	0.28	99.15%
			18.88	38.78	0.28	99.29%
			89.67	53.01	0.28	99.48%
			211.86	61.05	0.28	99.55%
			480.52	69.58	0.28	99.60%
Missouri	25	timeout	171.05	67.10	60.77%
Syringa	35	timeout	336.05	295.35	12.11%
VtlWavenet2011	35	timeout	[illegible]	[illegible]295.35	12.11%

Up to route length 16, optimal update time can be computed.

Too many updates can be performed concurrently: could be tackled with static analysis (future work).

- Network topologies from the To[illegible]
- Experiments run on a 64-bit Ubuntu 18.04 laptop

Support Beyond „Simple Solutions“



- No loop-free solution with waypoint: cannot update any edge
- But could first update s to v_2, then v_1,v_2,v_3, and finally s again to v_3

Conclusion

- Finally: networks are moving from manual to more automated operations
- Supported by emerging programmable networks and their solid theoretical foundations and languages
- Automata-theoretical approaches can be used to perform fast what-if analysis of the policy compliance (e.g., P-Rex, *AalWiNes*, etc.)
- More adaptive network operations further require tools for consistent network update scheduling (e.g., *Latte, QSynth*)

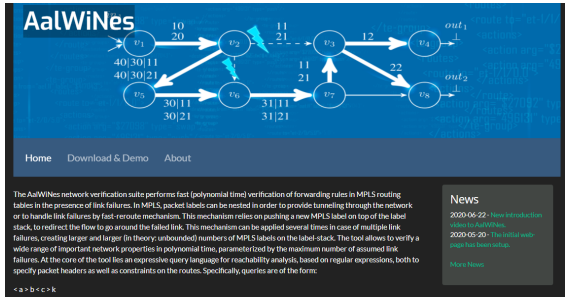
Efficient solutions to automatically verify and improve (synthesize) network configurations perhaps **#1 open research challenge** in networking.

- E.g., control plane verification and hybrid, complex network functions (IDS), quantitative aspects, performance aspects and scalability…

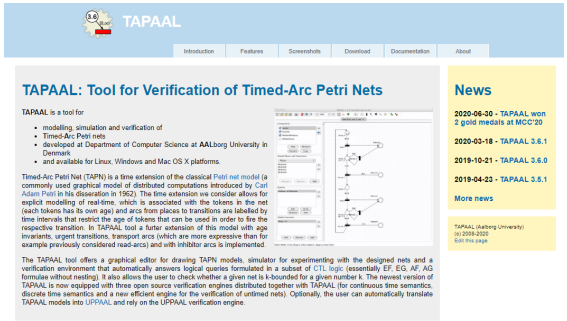
Hence looking for collaborations.

Further Reading

The AalWines project
https://aalwines.cs.aau.dk/



TAPAAL.net



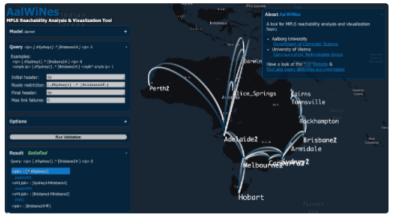
Netverify.fun

RESEARCH, NETWORK, VERIFICATION

Toward Polynomial-Time Verification of Networks with Infinite State Spaces: An Automata-Theoretic Approach


Stefan Schmid View
Jul 20, 2020 · 6 mins read


Jiri Srba View
Jul 20, 2020 · 6 mins read



With the increasing scale of communication networks, failures (e.g. link failures) are becoming the norm rather than the exception. Given the critical role such networks play for our digital society, it is important to

References

Resilient Capacity-Aware Routing
Stefan Schmid, Nicolas Schnepf and Jiri Srba.
27th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (**TACAS**), Virtual Conference, March 2021.

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.

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 ACM International Conference on emerging Networking EXperiments and Technologies (**CoNEXT**), Heraklion/Crete, Greece, December 2018.

Congestion-Free Rerouting of Flows on DAGs
Saeed Akhoondian Amiri, Szymon Dudycz, Stefan Schmid, and Sebastian Wiederrecht.
45th International Colloquium on Automata, Languages, and Programming (**ICALP**), Prague, Czech Republic, July 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.


Questions?