P-Rex: Fast Verification of MPLS Networks with Multiple Link Failures

Jesper Stenbjerg Jensen, Troels Beck Krøgh, Jonas Sand Madsen, Stefan Schmid, Jiri Srba and Marc Tom Thorgersen

Aalborg University, Denmark
and
University of Vienna, Austria

CoNEXT'18, Heraklion/Crete, December 6th, 2018

Motivation

- Operation of traditional computer networks is an error-prone task.
- Manually ensuring correct operation is particularly challenging in case of failures (combinatorial complexity).
- Possible traffic leaks, bandwidth overload, latency issues.

Motivation

- Operation of traditional computer networks is an error-prone task.
- Manually ensuring correct operation is particularly challenging in case of failures (combinatorial complexity).
- Possible traffic leaks, bandwidth overload, latency issues.

Specific Requirements from NORDUnet Operator

- Preservation of connectivity under multiple link failures.
- A link failure may increase the number of hops by at most three.
- The primary traffic should never be rerouted through Iceland, even under two links failing at the same time.
- Service labels that enable tunneling through the network should never be popped or modified.

Automatic Methods for Network Analysis

- The complexity of manual network operation creates an opportunity for automation.
- Problems of interest are often computationally intractable, in particular with fail-over protection.
- Current approaches restricted to finite header-spaces and often limited to brute-force enumeration of failed links.

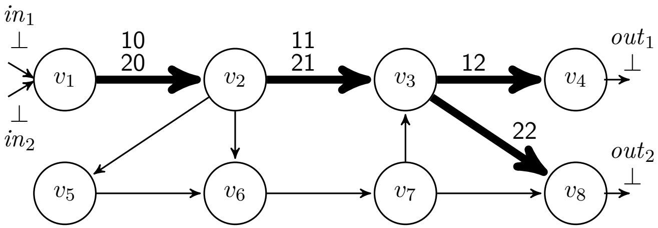
Automatic Methods for Network Analysis

- The complexity of manual network operation creates an opportunity for automation.
- Problems of interest are often computationally intractable, in particular with fail-over protection.
- Current approaches restricted to finite header-spaces and often limited to brute-force enumeration of failed links.

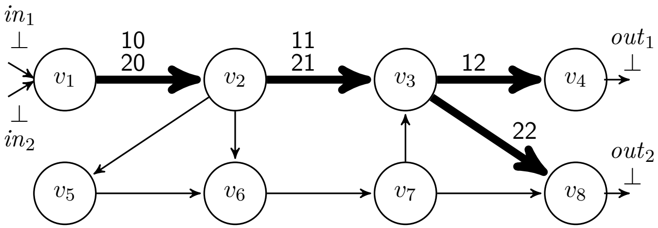
Our Contributions

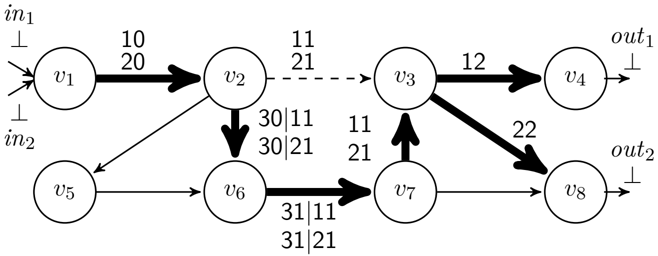
- A method for polynomial time verification of widely-used MPLS (Multi-Protocol Label Switching) networks.
- Unbounded nesting of labels in the packet headers (infinite header-space).
- What-if analysis under multiple link failures.
- Prototype tool implementation (P-REX) and experiments in cooperation with a real network operator.

MPLS Example — Local Protection

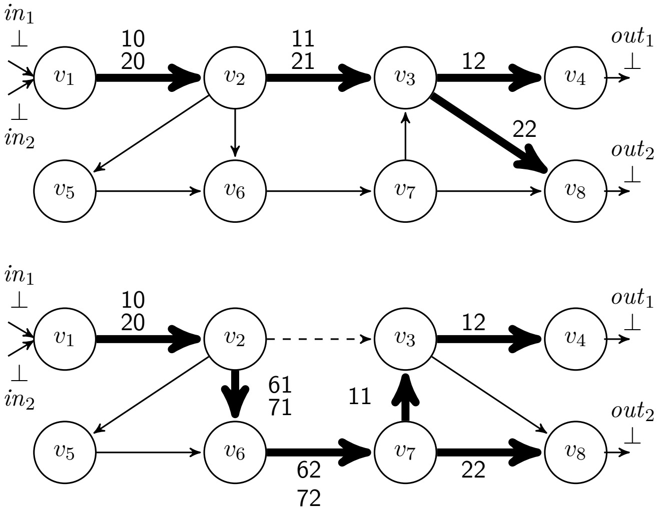


MPLS Example — Local Protection

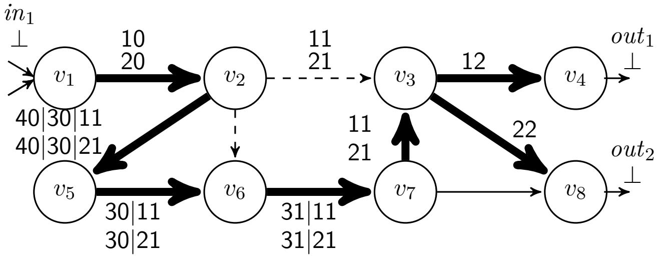


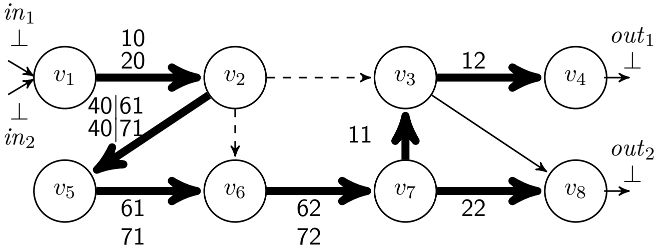


MPLS Example — Global Protection



MPLS Example — Multiple Link Failure





Observations about MPLS Networks

- Packet forwarding is determined by the top-most stack label.
- Labels can be pushed, swapped or popped.
- A packet header behaves like a pushdown stack.

We exploit a strong connection with the automata-theoretic approach for analysing pushdown automata in order to reason, in polynomial time, about MPLS networks.

Formal Model of MPLS Network

An MPLS network is a tuple $N = (V, I, L, E, \tau_v)$ where

- V is a finite set of routers,
- $I = \bigcup_{v \in V} I_v$ is a finite set of interfaces,
- $E \subseteq I \times I$ is the set of links between interfaces,
- $L = M \uplus M^{\perp} \uplus L^{IP}$ is the set of the label-stack symbols
- $\tau_v : I_v \times L \to (2^{I_v \times Op^*})^*$ is the routing table for each $v \in V$, where

$$Op = \{swap(\ell) \mid \ell \in L\} \cup \{push(\ell) \mid \ell \in L\} \cup \{pop\} \ .$$

Formal Model of MPLS Network

An MPLS network is a tuple $N = (V, I, L, E, \tau_v)$ where

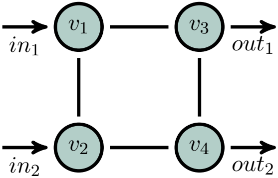
- V is a finite set of routers,
- $I = \bigcup_{v \in V} I_v$ is a finite set of interfaces,
- $E \subseteq I \times I$ is the set of links between interfaces,
- $L = M \uplus M^{\perp} \uplus L^{IP}$ is the set of the label-stack symbols
- $\tau_v : I_v \times L \to (2^{I_v \times Op^*})^*$ is the routing table for each $v \in V$, where

$$Op = \{swap(\ell) \mid \ell \in L\} \cup \{push(\ell) \mid \ell \in L\} \cup \{pop\} \ .$$

We allow

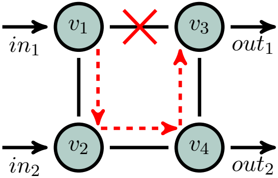
- nondeterminism in routing tables to account for traffic engineering,
- priorities to model fail-over protection, and
- operation sequencing to capture e.g. Segment Routing (SR).

MPLS Network Example



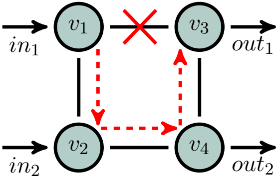
Router	In	Label	Priority	Out	Operation
v_1	in_1	ip_1	1	v_3	push(10)
v_2	in_2	ip_2	1	v_4	push(40)
v_3	v_1	10	1	out_1	pop
v_4	v_2	40	1	out_2	pop

MPLS Network Example



Router	In	Label	Priority	Out	Operation
v_1	in_1	ip_1	1	v_3	push(10)
	in_1	ip_1	2	v_2	push(10) $\circ$ push(101)
v_2	in_2	ip_2	1	v_4	push(40)
	v_1	101	1	v_4	swap(102)
v_3	v_1	10	1	out_1	pop
	v_4	10	1	out_1	pop
v_4	v_2	40	1	out_2	pop
	v_2	102	1	v_3	pop

MPLS Network Example



Router	In	Label	Priority	Out	Operation
v_1	in_1	ip_1	1	v_3	push(10)
	in_1	ip_1	2	v_2	push(10) $\circ$ push(101)
v_2	in_2	ip_2	1	v_4	push(40)
	v_1	101	1	v_4	swap(102)
v_3	v_1	10	1	out_1	pop
	v_4	10	1	out_1	pop
v_4	v_2	40	1	out_2	pop
	v_2	102	1	v_3	pop

Network Trace Assuming a Failed Link from v_1 to v_3

$$(v_1, ip_1) \to (v_2, 101 \circ 10 \circ ip_1) \to (v_4, 102 \circ 10 \circ ip_1) \to (v_3, 10 \circ ip_1) \ldots$$

Query Language for MPLS Networks

A query on an MPLS network:

$$< a > \quad b \quad < c > \quad k$$

where

- a is a regular expression for allowed initial label-stack headers,
- b is a regular expression over routers, describing a path in the network,
- c is a regular expression for allowed final label-stack headers, and
- k is a maximum number of considered failed links.

Query Language for MPLS Networks

A query on an MPLS network:

$$< a > \quad b \quad < c > \quad k$$

where

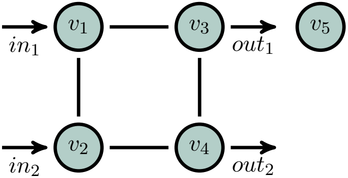
- a is a regular expression for allowed initial label-stack headers,
- b is a regular expression over routers, describing a path in the network,
- c is a regular expression for allowed final label-stack headers, and
- k is a maximum number of considered failed links.

Definition of query satisfaction

A trace $(v_1, h_1), (v_2, h_2), \ldots, (v_n, h_n)$ assuming a set F of failed links *satisfies* a query $< a > \quad b \quad < c > \quad k$ if

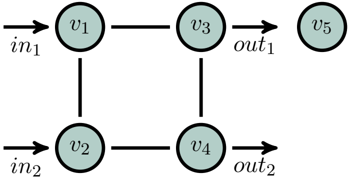
- $|F| \leq k$, and
- $h_1 \in Lang(a)$, $h_n \in Lang(c)$ and $v_1 v_2 \ldots v_n \in Lang(b)$.

Examples of MPLS Queries



$< ip_1 > \quad v_1 \; (.)^* \; v_5 \quad < ip_1 > \quad 0$ TRUE

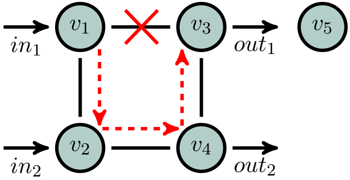
Examples of MPLS Queries



$< ip_1 > \quad v_1 \ (.)^* \ v_5 \quad < ip_1 > \quad 0$ TRUE

$< ip_1 > \quad v_1 \ (.)^* v_4 \ (.)^* \ v_5 \quad < ip_1 > \quad 0$ FALSE

Examples of MPLS Queries

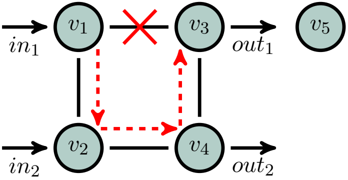


$$<ip_1> \quad v_1 \ (.)^* \ v_5 \quad <ip_1> \quad 0$$ TRUE

$$<ip_1> \quad v_1 \ (.)^* v_4 \ (.)^* \ v_5 \quad <ip_1> \quad 0$$ FALSE

$$<ip_1> \quad v_1 \ (.)^* v_4 \ (.)^* \ v_5 \quad <ip_1> \quad 1$$ TRUE

Examples of MPLS Queries



$< ip_1 > \quad v_1\ (.)^*\ v_5 \quad < ip_1 > \quad 0$ TRUE

$< ip_1 > \quad v_1\ (.)^* v_4\ (.)^*\ v_5 \quad < ip_1 > \quad 0$ FALSE

$< ip_1 > \quad v_1\ (.)^* v_4\ (.)^*\ v_5 \quad < ip_1 > \quad 1$ TRUE

$< ip_1 + ip_2 > \quad v_1\ [\hat{}\, v_2]^*\ v_5 \quad < (.)^* > \quad 2$ TRUE

Automata Theoretic Approach

Fact

Queries allow to specify reachability properties over infinite (but regular) sets of sequences of labels and routers.

Automata Theoretic Approach

Fact

Queries allow to specify reachability properties over infinite (but regular) sets of sequences of labels and routers.

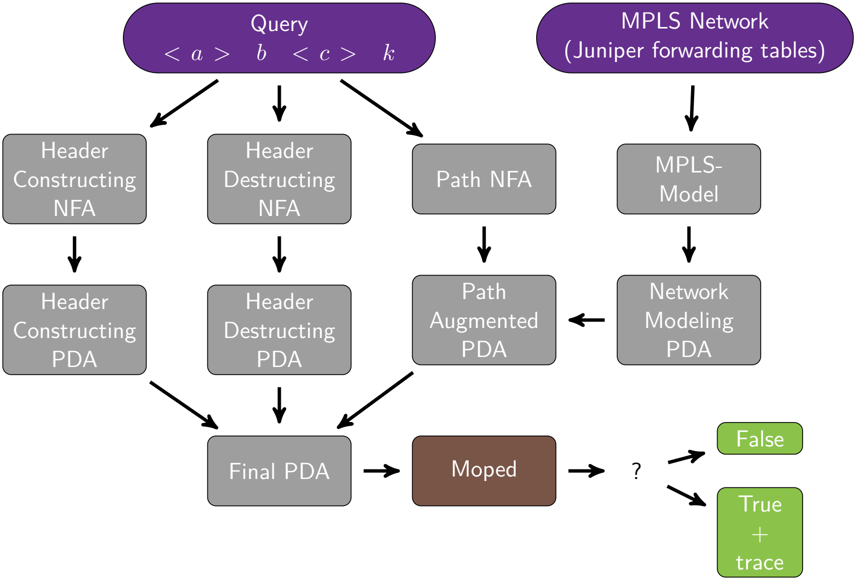
Theorem [Büchi'64, Esparza etc.'00]

Given a pushdown automaton and two of its configurations (q_0, w_0) and (q_f, w_f), it is in polynomial time decidable whether $(q_0, w_0) \rightarrow^* (q_f, w_f)$.

The result can be generalized to reachability between regular sets of configurations.

There is an efficient tool support for checking reachability in pushdown automata. We use the tool Moped.

Answering Query Satisfability Problem



Details of the Encoding to PDA

Approximation of Failed Links

Instead of brute force exploration of all possible sets of up to k failed links, we keep track of the number of times we use a rule with a lower priority.

- Over-approximation — in every router at any moment we assume up to k failed outgoing links.
- Under-approximation — in every router, once a lower priority rule is used we increase correspondingly a global counter of failed links.

Details of the Encoding to PDA

Approximation of Failed Links

Instead of brute force exploration of all possible sets of up to k failed links, we keep track of the number of times we use a rule with a lower priority.

- Over-approximation — in every router at any moment we assume up to k failed outgoing links.
- Under-approximation — in every router, once a lower priority rule is used we increase correspondingly a global counter of failed links.

Theorem (Over-approximation)

Any network trace implies the existence a corresponding computation in the over-approximating PDA.

Details of the Encoding to PDA

Approximation of Failed Links

Instead of brute force exploration of all possible sets of up to k failed links, we keep track of the number of times we use a rule with a lower priority.

- Over-approximation — in every router at any moment we assume up to k failed outgoing links.
- Under-approximation — in every router, once a lower priority rule is used we increase correspondingly a global counter of failed links.

Theorem (Over-approximation)

Any network trace implies the existence a corresponding computation in the over-approximating PDA.

Theorem (Under-approximation)

Any loop-free computation in the under-approximating PDA implies the existence of the corresponding network trace.

Framework for Checking Query Satisfaction

To check a query satisfaction problem in a given MPLS network:

- First, we construct the over- and under-approximating PDA and ask a reachability question.
- If the answer is positive in the under-approximating PDA then the query holds in the MPLS network.
- If the answer is negative in the over-approximating PDA then the query does not in the MPLS network.
- Otherwise the answer is inconclusive.

Complexity

All steps in the algorithm can be performed in polynomial time.

Implementation Details

- Algorithms are implemented in a prototype command-line tool P-REX (Python 3.6).
- Open source, available at `www.github.com/p-rexmpls`.
- Our case study for a basic reachability query produced a pushdown with 86,256,450 transitions (file size 4.4 GB).

Implementation Details

- Algorithms are implemented in a prototype command-line tool P-REX (Python 3.6).
- Open source, available at `www.github.com/p-rexmpls`.
- Our case study for a basic reachability query produced a pushdown with 86,256,450 transitions (file size 4.4 GB).

Removal of Redundant Rules

We perform a static analysis in order to over-approximate the possible top-most labels in every control state of the pushdown automaton.

Implementation Details

- Algorithms are implemented in a prototype command-line tool P-REX (Python 3.6).
- Open source, available at `www.github.com/p-rexmpls`.
- Our case study for a basic reachability query produced a pushdown with 86,256,450 transitions (file size 4.4 GB).

Removal of Redundant Rules

We perform a static analysis in order to over-approximate the possible top-most labels in every control state of the pushdown automaton.

- This results in a large reduction in the number of pushdown rules.
- In our case we ended up with 17,847,465 rules (file of size 875 MB).

XML Format for Network Topology

```
<network>
  <routers>
    <router name="s1">
      <interfaces>
        <interface name="i1"/>
        <interface name="i2"/>
      </interfaces>
    </router>
    ...
  </routers>
  <links>
    <link>
          <shared_interface router="s1" interface="i2"/>
          <shared_interface router="s2" interface="i5" />
     </link>
    ...
  </links>
</network>
```

XML Format for Routing Tables

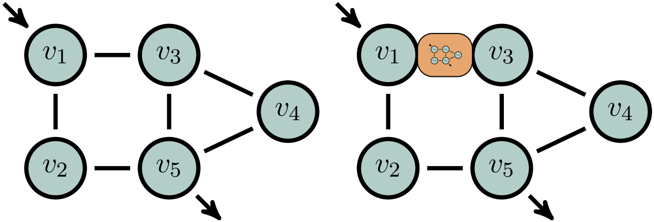
```
<routing for="s3">
    <destination from="s2" label="11">
        <te-group>
          <routes>
            <route to="s4">
               <action type="pop"/>
            </route>
          </routes>
        </te-group>
        <te-group>
          <routes>
            <route to="s5">
              <action arg="12" type="swap"/>
            </route>
          </routes>
        </te-group>
        ...
    </destination>
</routing>
```

Running P-REX

```
compute3:~/P-Rex$ PATH="./bin/:$PATH" PYTHONPATH=. python3 \
prex/main.py xml res/nestable/topo.xml  res/nestable/routing.xml \
adv-query "<> .* <>" 1
compile run
Loading: 0.0273s
Under is False
Compiling: 0.020s
Size: 481
Transitions/s: 23596.023 t/s
Verifying: 0.133s
Transitions/s: 3623.490 t/s
YES

build_0 <Label.BOS> --> C_0 Label.STAR Label.BOS
C_0 <Label.STAR> --> build_1
build_1 <*> --> (<Loc start@14037>,<NFALoc 'N_2' 14032>) ...
```

Scalable Experiments on a Synthetic Model



Query that is not satisfied

$$<.>\ v_1(.)^* v_5\ <..>\ k$$

- Each nesting creates an additional MPLS tunnel.
- We compare our tool P-REX with HSA tool.
- HSA enumerates the possible sets of failed links.
- P-REX uses over-approximation (all answers are conclusive).

Comparison with HSA (time in seconds)

P-REX HSA	$k = 0$	$k = 1$	$k = 2$	$k = 3$
Nesting: 1 Routers: 10	0.6 0.1	0.6 0.1	0.6 0.4	0.6 3.7
Nesting: 2 Routers: 15	0.6 0.1	0.6 0.3	0.6 1.9	0.6 55.9
Nesting: 3 Routers: 20	0.6 0.1	0.6 0.3	0.6 6.8	0.6 335.6
Nesting: 4 Routers: 25	0.6 0.1	0.6 0.6	0.6 16.4	0.6 567.2
Nesting: 5 Routers: 30	0.6 0.1	0.6 1.0	0.6 34.6	0.6 1901.1
Nesting: 6 Routers: 35	0.6 N/A	0.6 N/A	0.6 N/A	0.7 N/A

Comparison with HSA (time in seconds)

P-REX HSA	$k = 0$	$k = 1$	$k = 2$	$k = 3$
Nesting: 1 Routers: 10	0.6 0.1	0.6 0.1	0.6 0.4	0.6 3.7
Nesting: 2 Routers: 15	0.6 0.1	0.6 0.3	0.6 1.9	0.6 55.9
Nesting: 3 Routers: 20	0.6 0.1	0.6 0.3	0.6 6.8	0.6 335.6
Nesting: 4 Routers: 25	0.6 0.1	0.6 0.6	0.6 16.4	0.6 567.2
Nesting: 5 Routers: 30	0.6 0.1	0.6 1.0	0.6 34.6	0.6 1901.1
Nesting: 6 Routers: 35	0.6 N/A	0.6 N/A	0.6 N/A	0.7 N/A

Comparison with HSA (time in seconds)

P-REX HSA	$k = 0$	$k = 1$	$k = 2$	$k = 3$
Nesting: 1 Routers: 10	0.6 0.1	0.6 0.1	0.6 0.4	0.6 3.7
Nesting: 2 Routers: 15	0.6 0.1	0.6 0.3	0.6 1.9	0.6 55.9
Nesting: 3 Routers: 20	0.6 0.1	0.6 0.3	0.6 6.8	0.6 335.6
Nesting: 4 Routers: 25	0.6 0.1	0.6 0.6	0.6 16.4	0.6 567.2
Nesting: 5 Routers: 30	0.6 0.1	0.6 1.0	0.6 34.6	0.6 1901.1
Nesting: 6 Routers: 35	0.6 N/A	0.6 N/A	0.6 N/A	0.7 N/A

Industrial Case Study

- A case study on a MPLS network operated by NORDUnet.
- NORDUnet is a regional service provider and has 24 MPLS routers, primarily Juniper.
- Geographically distributed across several countries.

Extracting tables from routers

```
show route forwarding-table family mpls extensive | display .ml
show isis adjacency detail | display .ml
```

- Complex MPLS routing, more than 30,000 MPLS labels.
- Almost one million forwarding rules in our MPLS model.

Reachability Matrix

Question

Compute connectivity in NORDUnet network with up to 2 link failures.

$$<.> \; Y \; (.)^* \; Z \; <.> 2$$

Can a packet with some IP-label only be forwarded from router Y to router Z, assuming at most 2 *link failures?*

- On average about 1 hour to compute this for any pair of routers.
- We run both over- and under-approximation.

Reachability Matrix: $<.>\ Y\ (.)^*\ Z\ <.>\ 2$

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
A	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
B	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
C	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	✓	.	.	.	.	.
D	.	.	.	.	.	.	.	.	.	.	?	.	.	.	.	.	.	.	✓	.	.	.	.	.
E	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
F	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
G	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
H	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
I	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
J	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
K	.	.	.	✓	.	.	.	.	.	.	.	.	.	.	.	.	?	.	.	.	.	.	.	.
L	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
M	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
N	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
O	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
P	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Q	.	.	.	.	.	.	.	.	.	.	✓	.	.	.	.	.	.	.	.	.	.	.	.	.
R	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
S	.	.	?	✓	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
T	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
U	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
V	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
W	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

Reachability Matrix: $<..>\ Y\ (.)^*\ Z\ <(.)^*>2$

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
A	?	✓	✓	✓	✓	.	.	✓	✓	.	✓	.	.	.	✓	.	✓	.	✓	✓	✓	✓	.	.
B	✓	?	✓	✓	✓	✓	.	?	✓	.	✓	.	.	.	✓	.	✓	.	✓	.	✓	✓	.	.
C	✓	✓	?	✓	✓	✓	.	✓	✓	.	✓	.	.	.	✓	.	✓	.	✓	✓	✓	✓	.	.
D	✓	✓	✓	?	✓	.	.	✓	✓	.	✓	.	.	.	✓	.	✓	.	✓	✓	✓	✓	.	.
E	✓	✓	✓	✓	?	.	.	✓	✓	.	✓	.	.	.	✓	.	✓	.	✓	✓	✓	✓	.	.
F	.	✓	✓	?	?	.	.	.	.	.	?	.	.	.	✓	.	.	.	?	.	.	.	.	.
G	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	✓	.	.	.	.	.
H	✓	✓	✓	✓	✓	.	.	.	✓	.	✓	.	.	.	✓	.	✓	.	✓	✓	✓	✓	.	.
I	✓	✓	✓	✓	✓	.	.	✓	?	.	✓	.	.	.	✓	.	✓	.	✓	✓	✓	✓	.	.
J	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
K	✓	✓	✓	✓	✓	✓	.	✓	✓	.	?	.	.	.	✓	.	✓	.	✓	✓	✓	✓	.	.
L	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
M	.	.	.	.	✓	.	.	.	✓	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
N	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
O	✓	✓	✓	✓	✓	✓	.	✓	✓	.	✓	.	.	.	?	.	✓	✓	✓	✓	✓	✓	.	.
O	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Q	✓	✓	✓	✓	✓	.	.	✓	✓	.	✓	.	✓	.	✓	.	?	✓	✓	✓	✓	✓	.	.
R	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
S	✓	✓	✓	✓	✓	.	.	✓	?	.	✓	.	.	.	✓	.	✓	.	?	✓	✓	✓	.	.
T	✓	✓	✓	✓	✓	.	.	✓	✓	.	✓	.	.	.	✓	.	✓	.	✓	.	✓	✓	✓	.
U	✓	✓	✓	✓	✓	.	.	✓	✓	.	✓	.	.	.	✓	.	✓	.	✓	✓	.	✓	.	.
V	✓	✓	✓	✓	✓	.	.	✓	✓	.	✓	.	.	.	✓	.	.	.	✓	✓	✓	.	.	.
W	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	✓	.	.	.	.
X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

Operator Specific Queries

Can a packet arriving to A with the service label 234 on top of some IP-label be routed to B while visiting router F such that the labels are popped?

$$< 234\ .>\ A\ (.)^*\ F\ (.)^*\ B <>\ 0$$

Operator Specific Queries

Can a packet arriving to A with the service label 234 on top of some IP-label be routed to B while visiting router F such that the labels are popped?

$$< 234\ .\ >\ A\ (.)^*\ F\ (.)^*\ B <>\ 0$$

- FALSE for $k = 0$ (over-approximation, 38m 26s, 7.00 GB RAM)
- TRUE for $k = 1$ (under-approximation, 91m 14s, 13.95 GB RAM) and returns a trace

Operator Specific Queries

Can a packet arriving to some router with the service label 234 do 3 or more hops in the network before all labels are popped?

$$< 234\ .> \ldots (.)^* <> \ 0$$

Operator Specific Queries

Can a packet arriving to some router with the service label 234 do 3 or more hops in the network before all labels are popped?

$$< 234 \ . > \ . \ . \ . \ (.)^* <> \ \ 0$$

- FALSE for $k = 0$ (over-approximation, 48m 44s, 6.01 GB RAM)

Operator Specific Queries

Can a packet arriving to some router with the service label 234 do 3 or more hops in the network before all labels are popped?

$$< 234\ . > \ .\ .\ .\ (.)^* <> \ 0$$

- FALSE for $k = 0$ (over-approximation, 48m 44s, 6.01 GB RAM)

$$< 234\ . > \ .\ .\ .\ .\ .\ (.)^* <> \ 1$$

- TRUE for $k = 1$ (under-approximation, 109m 32s, 14.18 GB RAM) and returns a trace

Operator Specific Queries

Can the service label 800 ever be swapped?
(It defines a tunnel through the network.)

$$< 800\ .> (.)^* < [\hat{}800]\ .> \ 1$$

Operator Specific Queries

Can the service label 800 ever be swapped?
(It defines a tunnel through the network.)

$$< 800 \ . > (.)^* < [\hat{\ }800] \ . > \ 1$$

- FALSE for $k = 1$ (over-approximation 28m 9s, 5.04 GB RAM)

Conclusion and Future Work

Summary

- Polynomial time algorithm for reachability on MPLS networks.
- Polynomial time over-/under-approximation for failed links.
- Expressive query language based on regular expressions.
- General enough to include also e.g. (MPLS-based) Segment Routing.
- Industrial case study, promising results.

Conclusion and Future Work

Summary

- Polynomial time algorithm for reachability on MPLS networks.
- Polynomial time over-/under-approximation for failed links.
- Expressive query language based on regular expressions.
- General enough to include also e.g. (MPLS-based) Segment Routing.
- Industrial case study, promising results.

Challenges

- Verification speed (C++ implementation, CEGAR, machine learning).
- Quantitative properties (bandwidth, latency, ...).
- Automatic synthesis of fail-over protection.