Tracking Transport-Layer Evolution with *PATHspider*

Brian Trammell, Mirja Kühlewind, and **Piet De Vaere,** ETH Zürich, Switzerland
Iain Learmonth and Gorry Fairhurst, University of Aberdeen, Scotland

Applied Networking Research Workshop — Prague — 15 July 2017

mami
measurement and architecture for a middleboxed internet

measurement | architecture | experimentation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 688421.The opinions expressed and arguments employed reflect only the authors' view. The European Commission is not responsible for any use that may be made of that information.

Supported by the Swiss State Secretariat for Education, Research and Innovation under contract number 15.0268. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the Swiss Government.

The problem, a solution

Middleboxes interfere with connections

→ Transport protocols must **react**

& Reaction must be **data-driven**

Data can be provided by **active internet measurement**

We build a toolchain for:

Controlled experiments

Diverse protocol features

Diverse targets

That observes **conditions** associated with **paths**

Overview

PATHspider architecture

Scaling up cloud-based measurements

Long term evolution & reducing noise floor

Measurement methodology for

DSCP

ECN

TFO

Increasing deployment

+ Results and insights

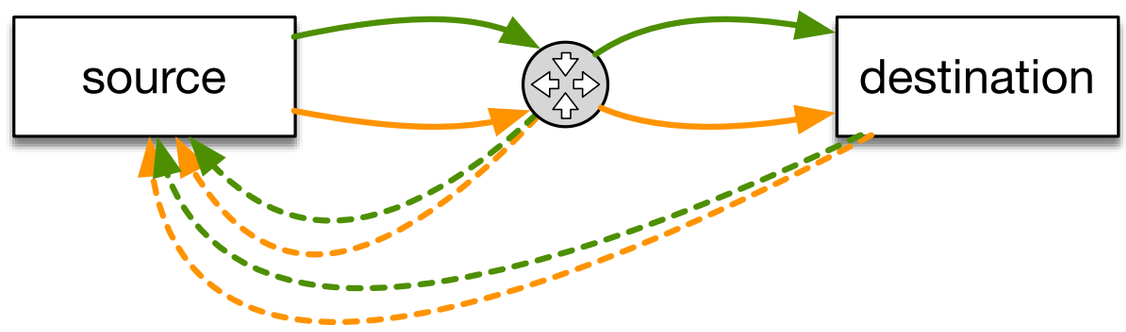
A/B testing of path transparency

Controlled measurement setup

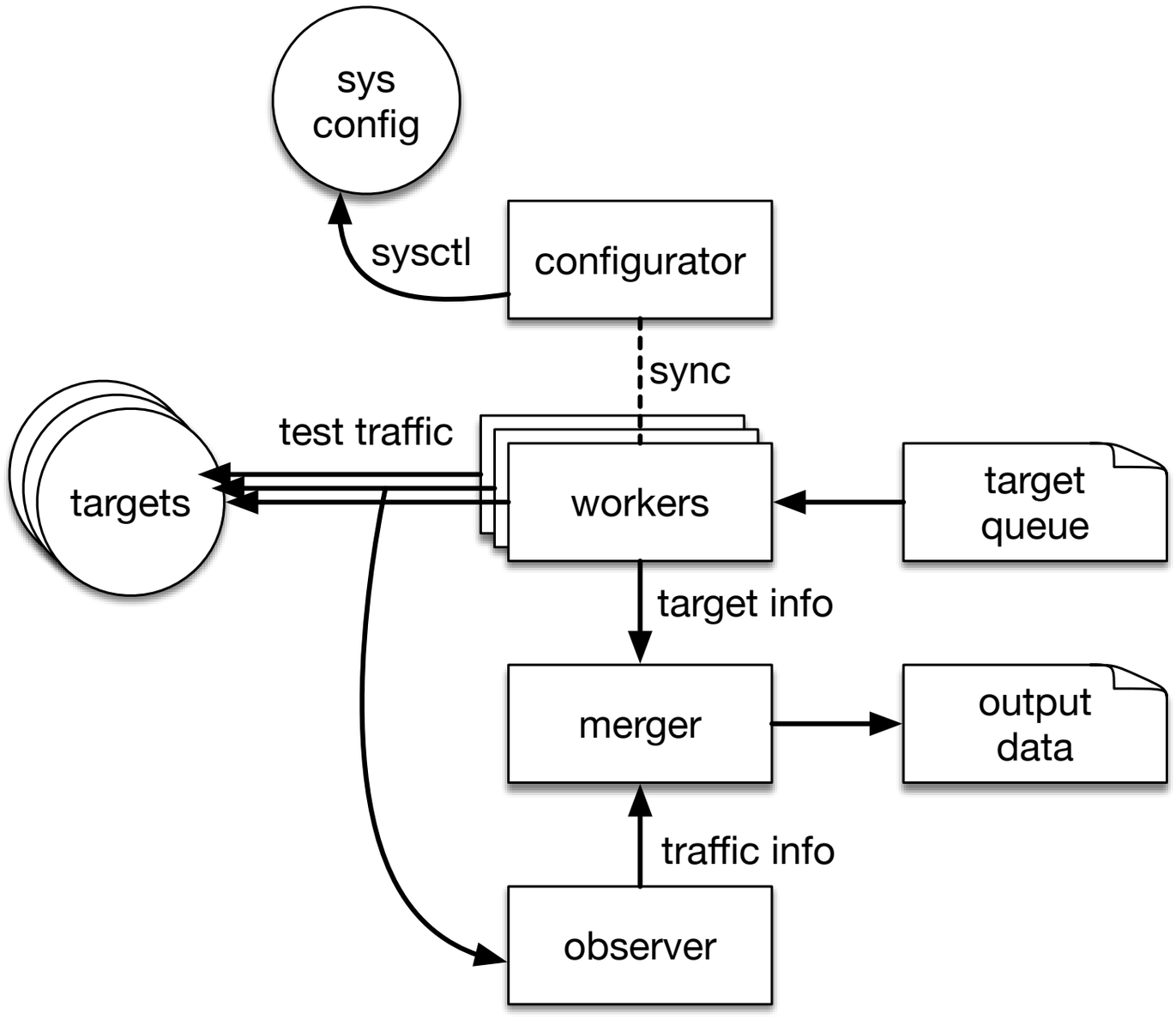
Baseline: vanilla TCP, {no|default} option

Experimental: protocol feature under test

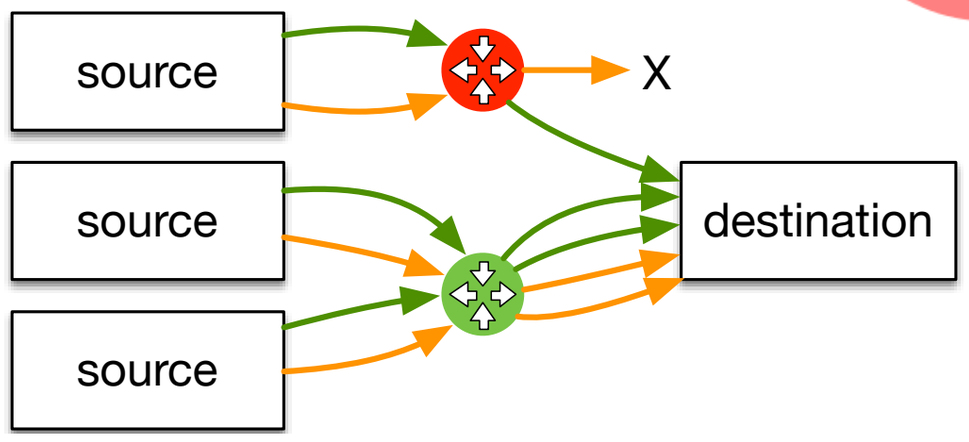
Find feature-dependent issues



Design of *PATHspider*



Site versus path dependency



Impairments exist

- Close to target
- Close to internet core ← More troubling!

Distinguish by

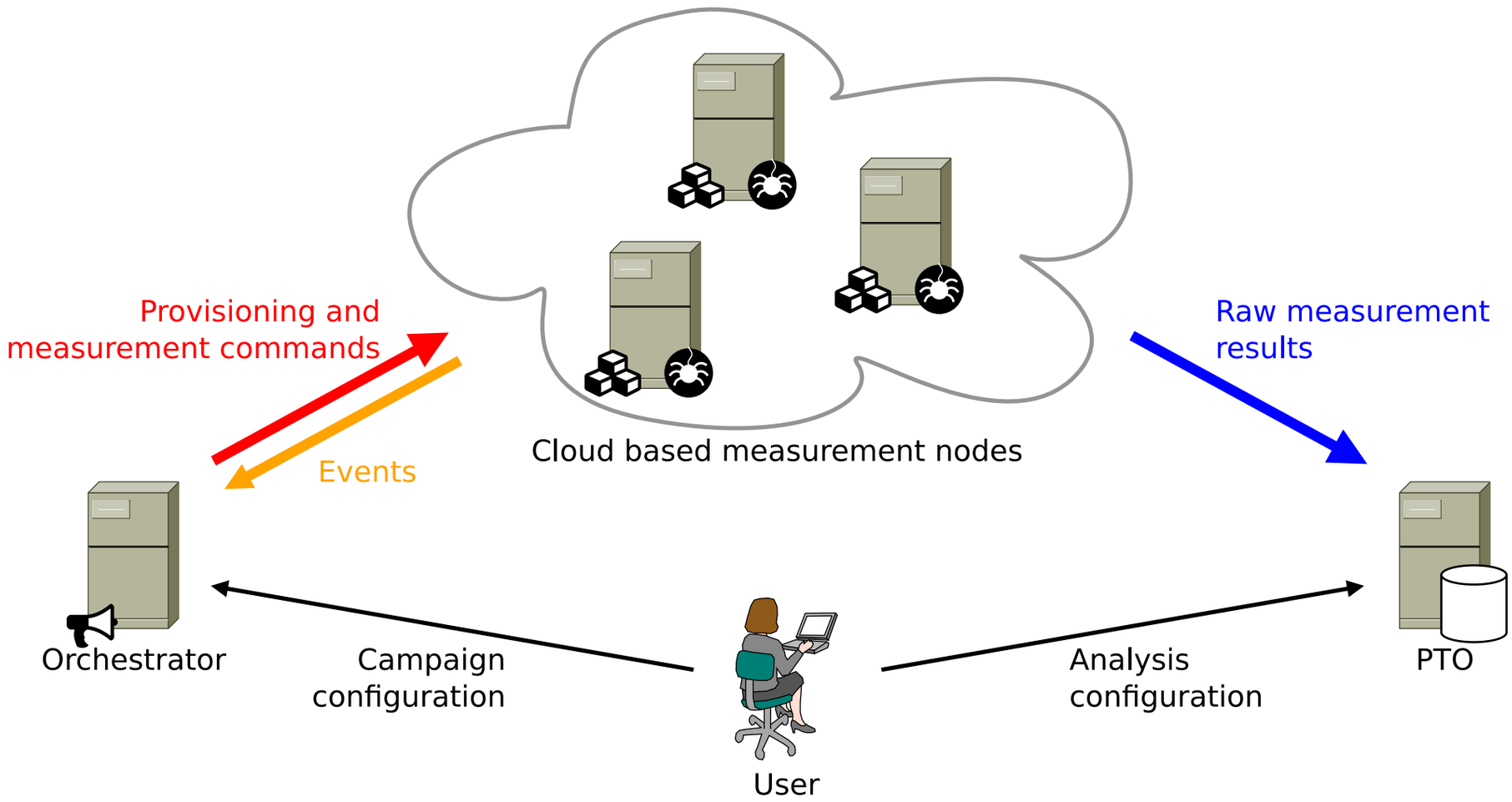
- Measuring from diverse vantage points
- Repeat measurements to detect transients

Scaling up

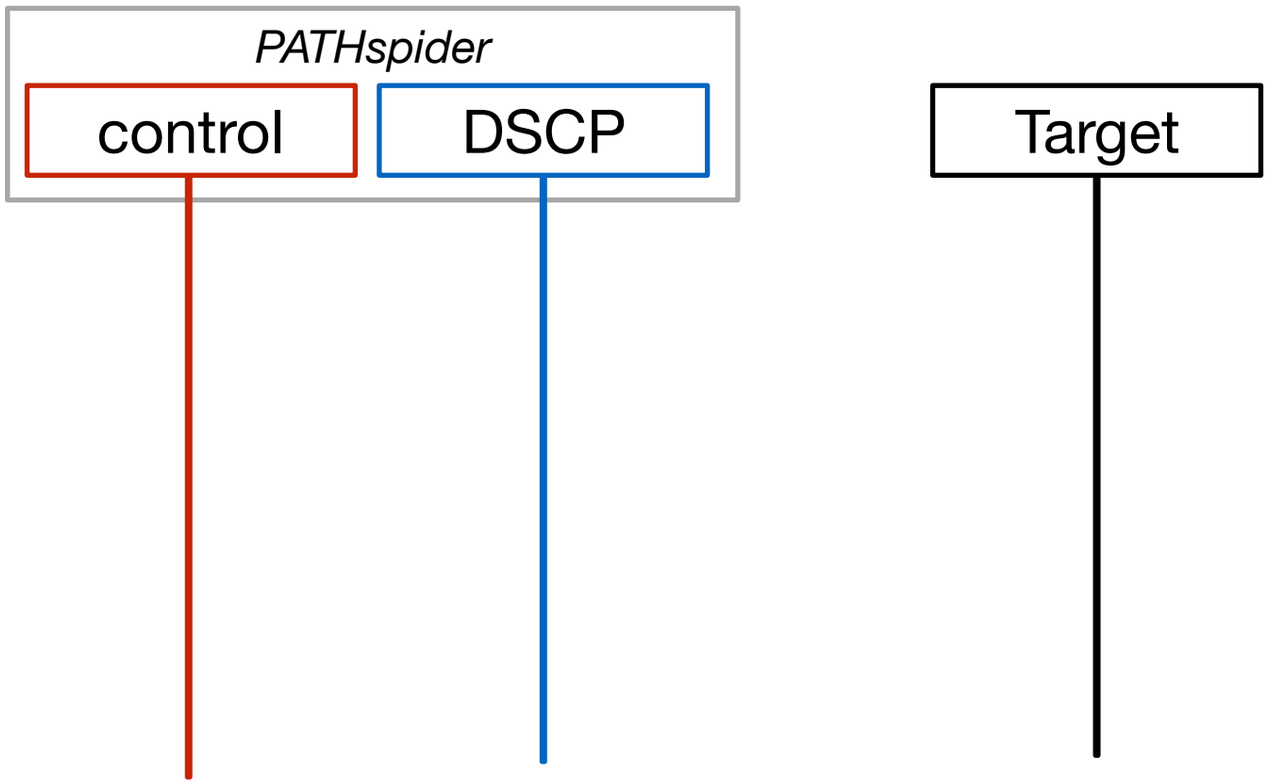
Cloud based measurements

Orchestrated using SaltStack

Results analysed on *Path Transparency Observatory*



Methodology
Differentiated Services (DSCP)

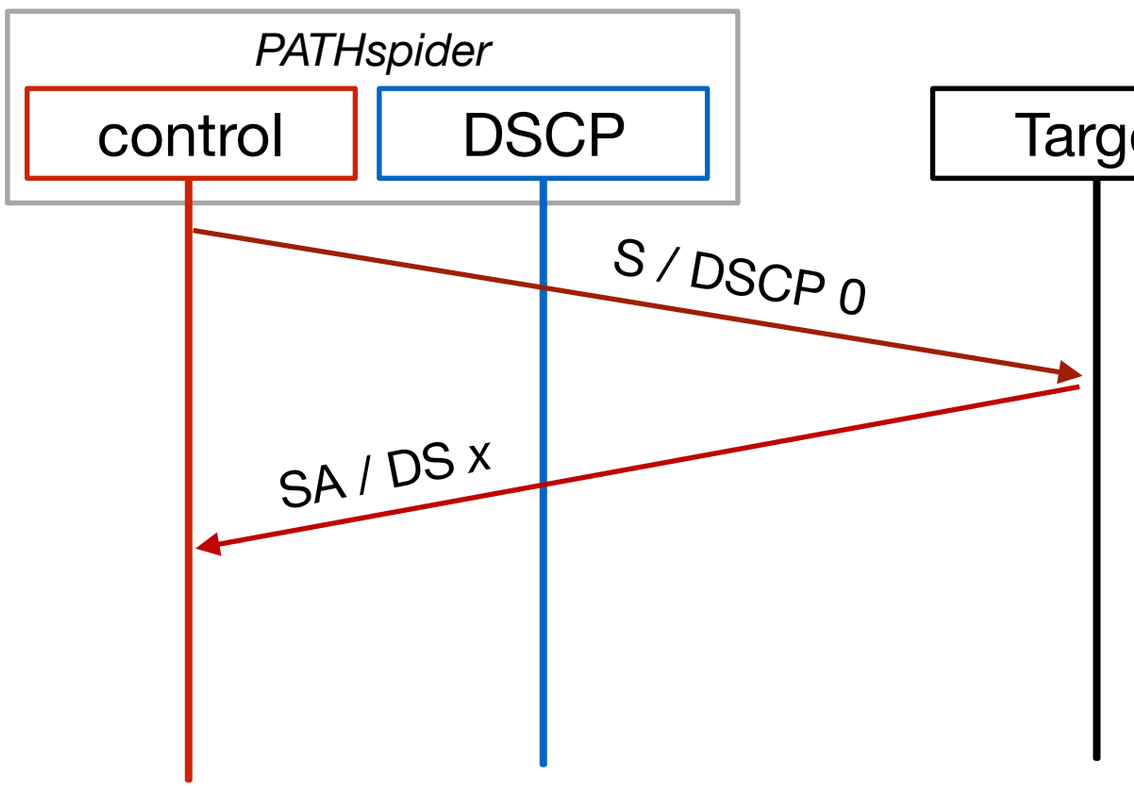


Send SYNs with default (0), and non-default DSCP

Interpret connection failure on non-default as dropped SYN.

Observe DSCP codepoint on SYN+ACK.

Methodology
Differentiated Services (DSCP)

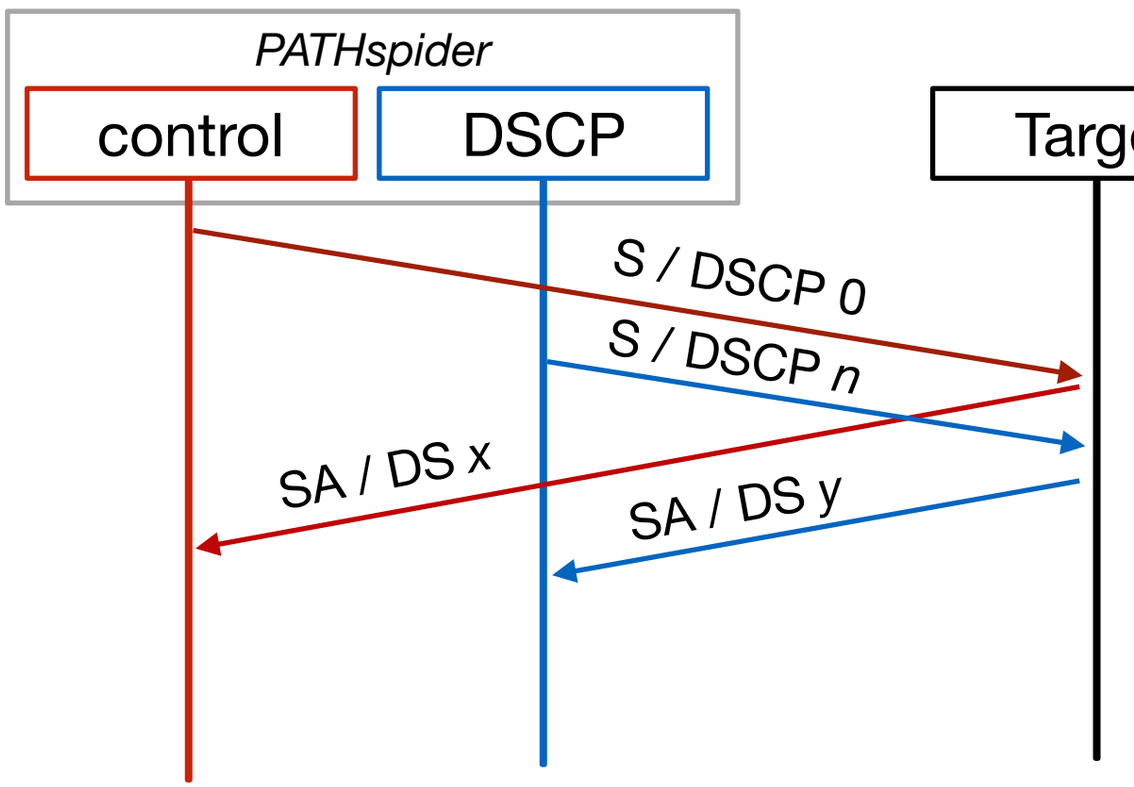


Send SYNs with default (0), and non-default DSCP

Interpret connection failure on non-default as dropped SYN.

Observe DSCP codepoint on SYN+ACK.

Methodology
Differentiated Services (DSCP)



Send SYNs with default (0), and non-default DSCP

Interpret connection failure on non-default as dropped SYN.

Observe DSCP codepoint on SYN+ACK.

Results: DSCP blocking

Measurements from Digital Ocean DCs

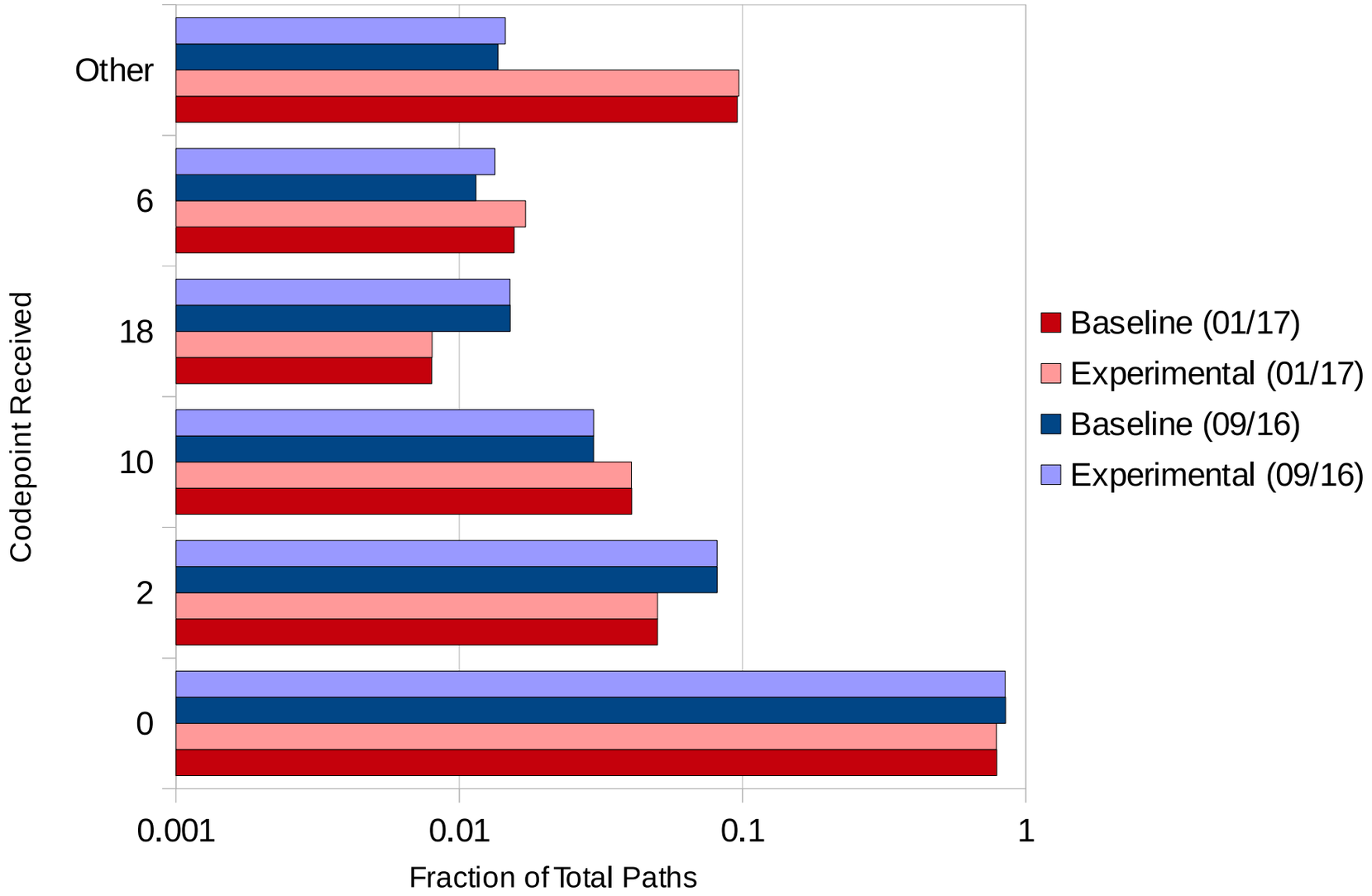
Negligible blocking of codepoint 46 (EF)

Most blocking seems path-dependent.

January 2017				
IPv4 n =620 611		IPv6 n =52 766		
hosts	pct	hosts	pct	
63 177	10.18%	28 985	54.93%	Completely failed to connect
557 434	**89.82%**	**23 781**	**45.06%**	Successfully connted with DSCP 0 (default); of which:
1 770	0.32%	124	0.52%	Failed to connect when DSCP 46 (EF) used;
1 334	0.24%	121	0.50%	but succeeded from at least one vantage point
556 998	**99.92%**	**23 778**	**99.98%**	Successfully connected with 46 (EF)

Results: DSCP return codepoint

DSCP on **return not dependent** on DSCP **on request**.



Methodology
TCP Fast Open (TFO)

Check connectivity without TFO, abort on fail.

API issue → long timeout

TFO experiment:

Connect once with TFO:

`tfo.cookie.received`

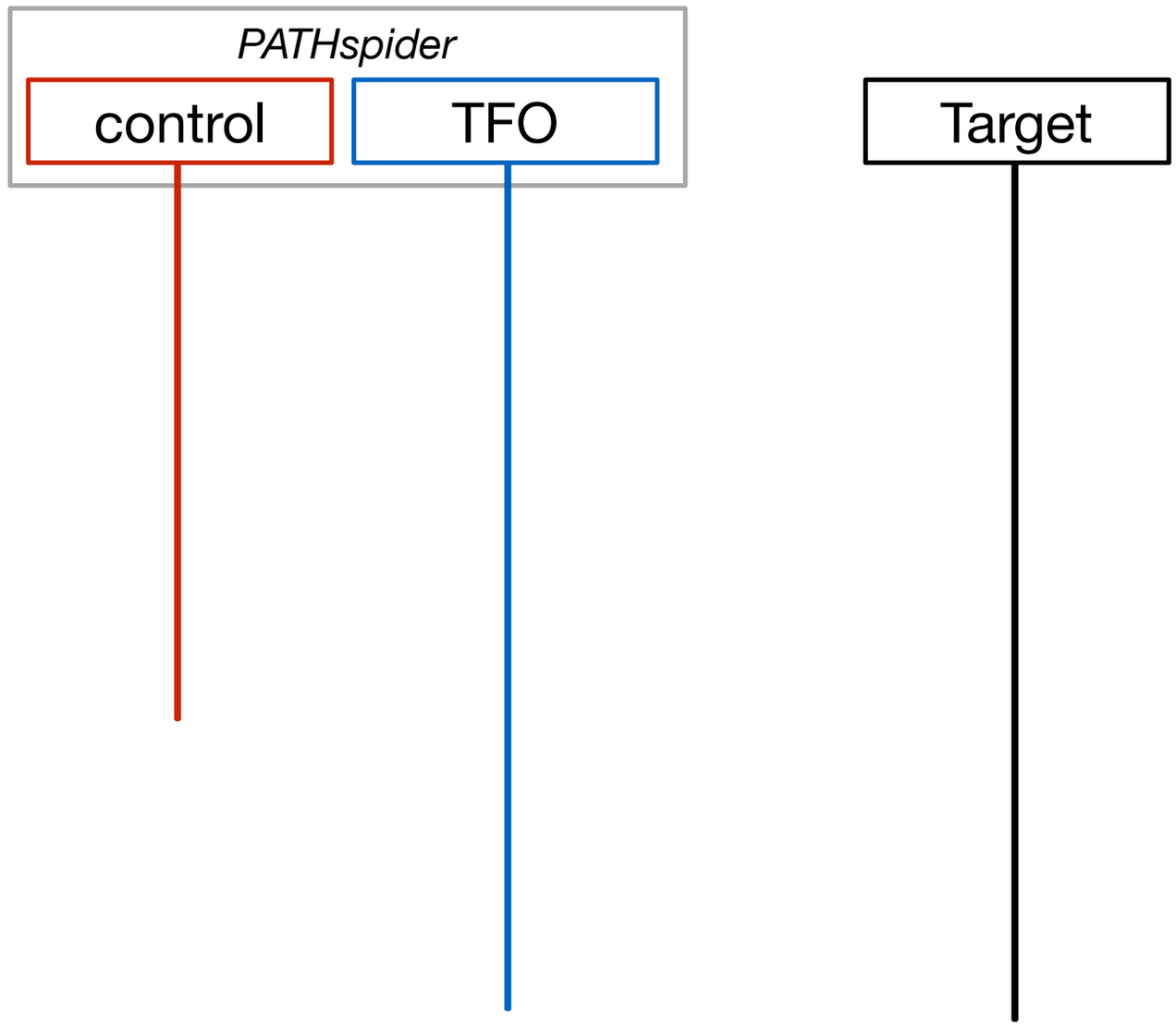
`tfo.cookie.not_received`

Connect again with TFO:

`tfo.syndata.acked`

`tfo.syndata.not_acked`

`tfo.syndata.failed`



Methodology
TCP Fast Open (TFO)

Check connectivity without TFO, abort on fail.

API issue → long timeout

TFO experiment:

Connect once with TFO:

`tfo.cookie.received`

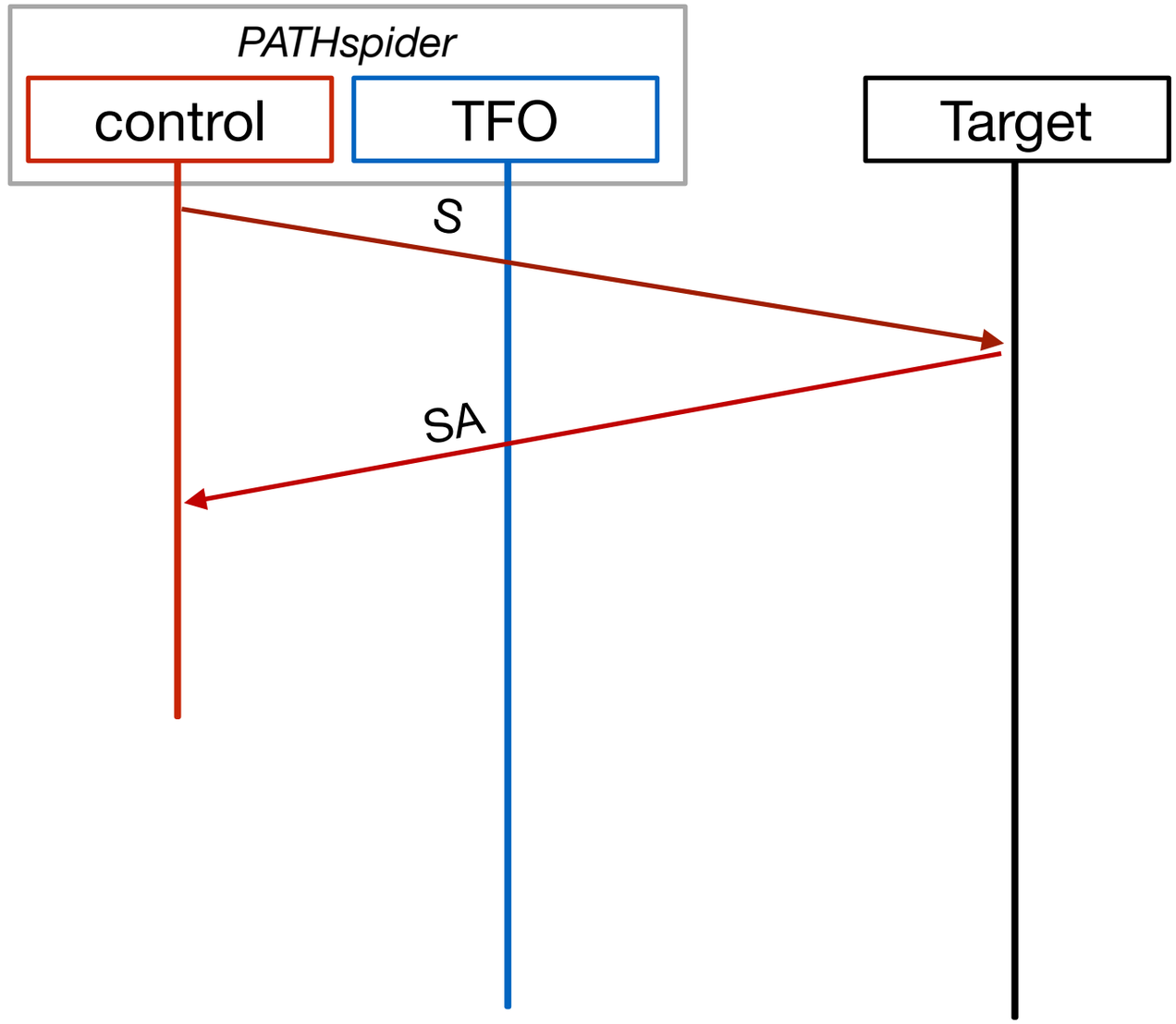
`tfo.cookie.not_received`

Connect again with TFO:

`tfo.syndata.acked`

`tfo.syndata.not_acked`

`tfo.syndata.failed`



Methodology
TCP Fast Open (TFO)

Check connectivity without TFO, abort on fail.

API issue → long timeout

TFO experiment:

Connect once with TFO:

`tfo.cookie.received`

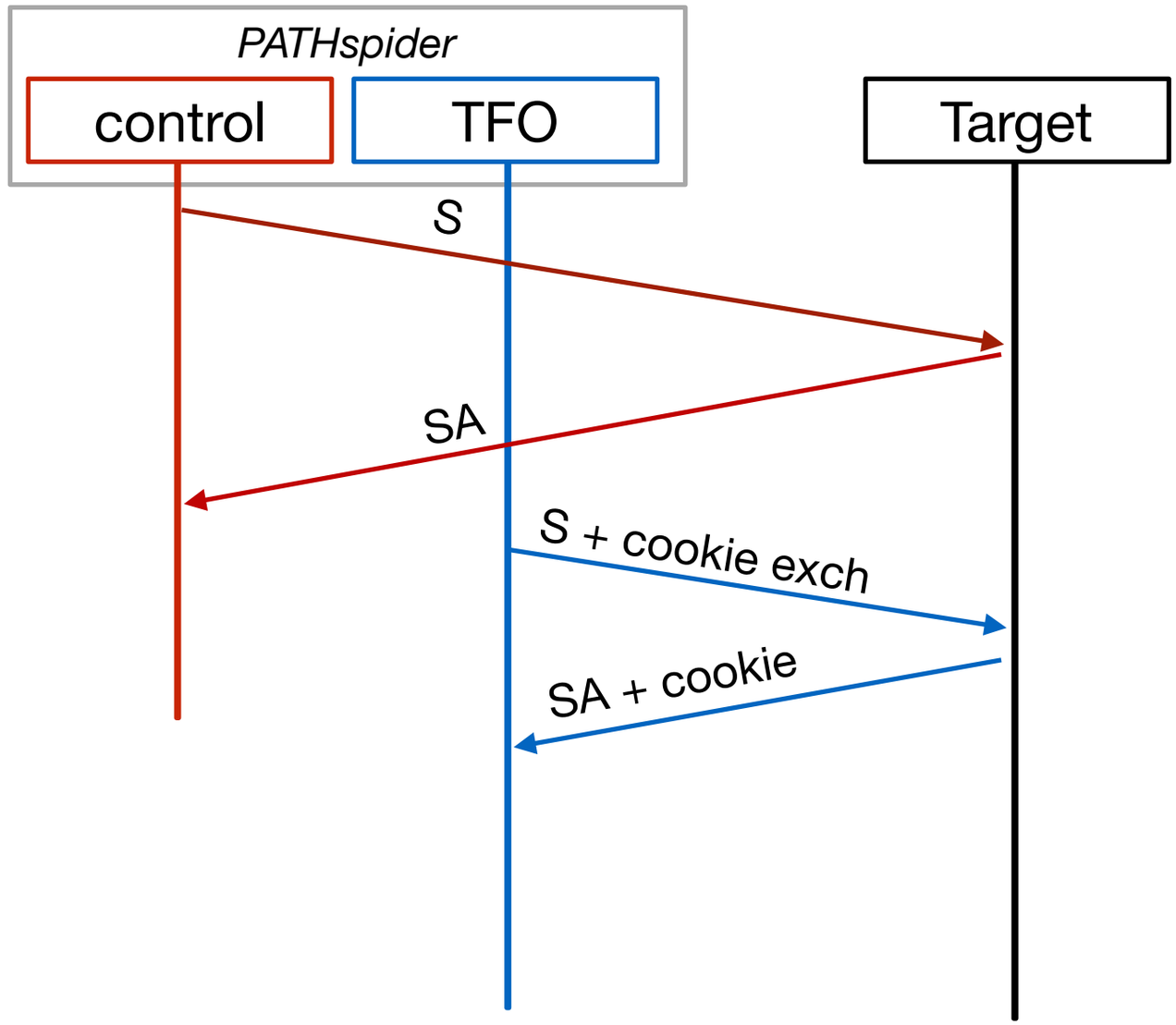
`tfo.cookie.not_received`

Connect again with TFO:

`tfo.syndata.acked`

`tfo.syndata.not_acked`

`tfo.syndata.failed`



Methodology
TCP Fast Open (TFO)

Check connectivity without TFO, abort on fail.

API issue → long timeout

TFO experiment:

Connect once with TFO:

`tfo.cookie.received`

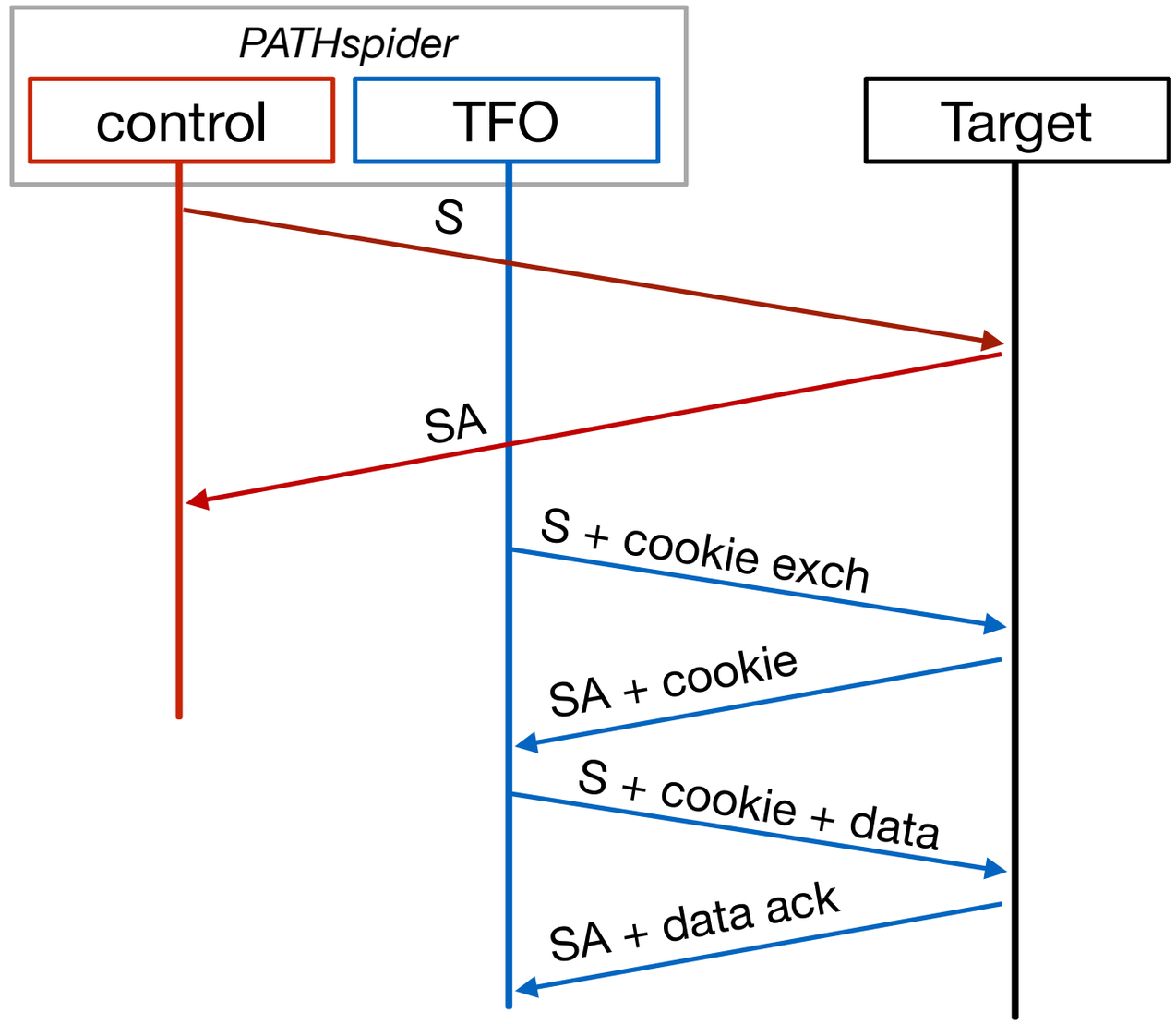
`tfo.cookie.not_received`

Connect again with TFO:

`tfo.syndata.acked`

`tfo.syndata.not_acked`

`tfo.syndata.failed`



Results: TFO

Web, Jan '17 n =939 680 hosts	pct	DNS, Jan '17 n =53 267 hosts	pct	description
29 839	3.18%	4 906	9.21%	Completely failed to connect
177	0.019%	26	0.049%	Failed to connect w/TFO option
908 464	96.7%	48 276	90.6%	Did not negotiate TFO
866	**0.092%**	**56**	**0.105%**	Negotiated TFO (exchanged a cookie); of which:
830	95.8%	54	96.4%	ACKed data on SYN †
0	0%	2	3.57%	Failed connection with data on SYN
33	3.81%	0	0%	Returned a cookie on ACKed data
12	1.39%	2	3.57%	Responded with a 6-byte cookie
31	3.58%	0	0%	Responded with an experimental option †
690	79.7%	53	94.6%	are in AS15169 (Google)

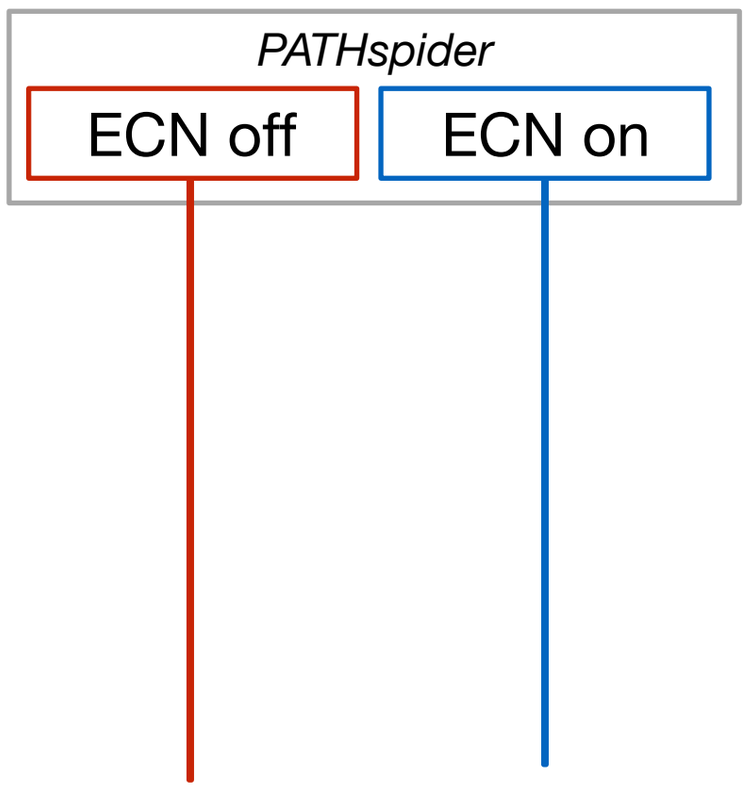
Correct TFO limited mostly to Google.

One anomaly linked to a single firm: unique implementation?

No path dependency seen (but N is small…)

Previous findings (Paasch): TFO impairment is in access networks

Methodology
Explicit Congestion Notification (ECN)



Connect **with and without ECN**

Measure

`ecn.connectivity.`*`status`*

`works`: off + on OK

`broken`: off OK, on fails

`transient`: on OK, off fails

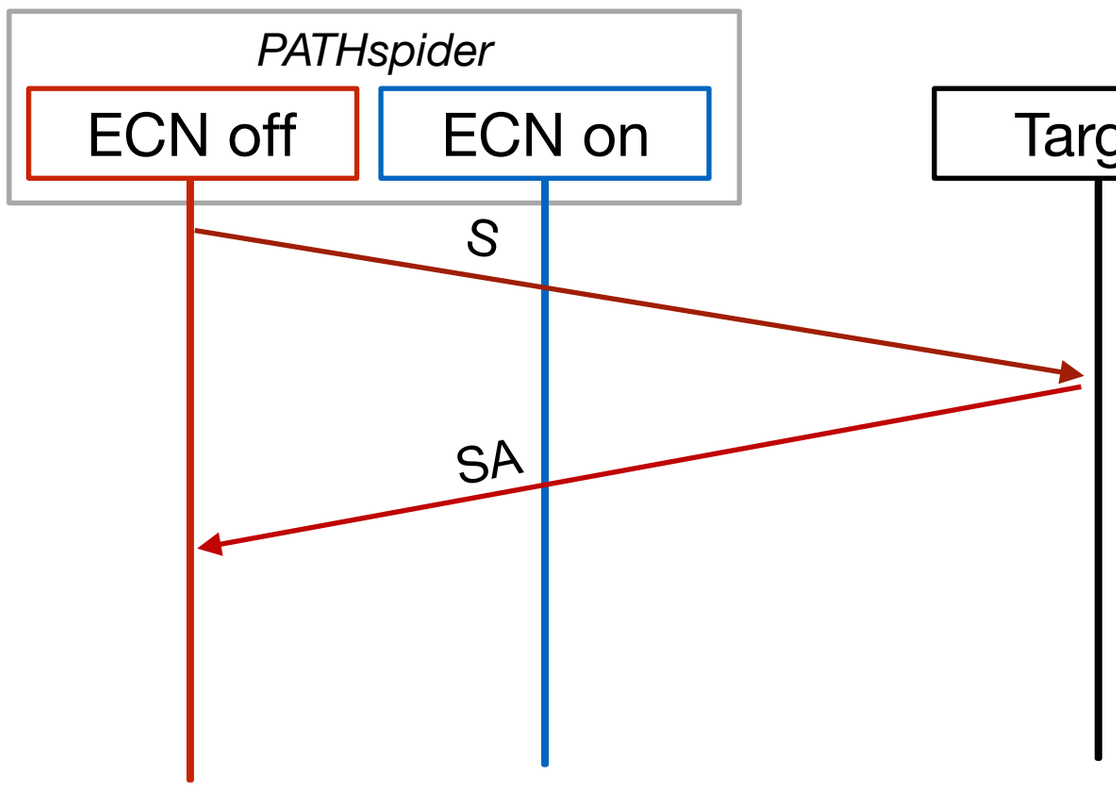
`offline`: no connection

`ecn.`*`codepoint`*`.seen`

generate traffic

negotiation $\overset{?}{\rightarrow}$ ECT marking

Methodology
Explicit Congestion Notification (ECN)



Connect **with and without ECN**

Measure

`ecn.connectivity.`*`status`*

`works`: off + on OK

`broken`: off OK, on fails

`transient`: on OK, off fails

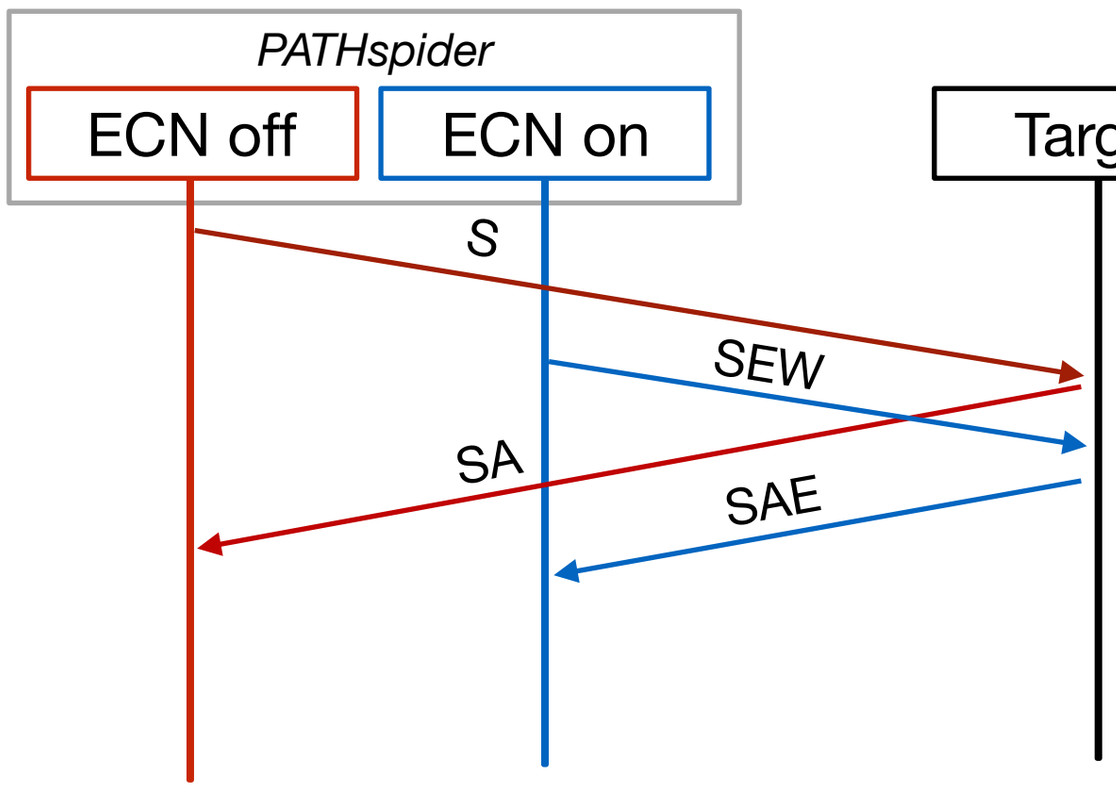
`offline`: no connection

`ecn.`*`codepoint`*`.seen`

generate traffic

negotiation $\xrightarrow{?}$ ECT marking

Methodology
Explicit Congestion Notification (ECN)



Connect **with and without ECN**

Measure

`ecn.connectivity.`*`status`*

`works`: off + on OK

`broken`: off OK, on fails

`transient`: on OK, off fails

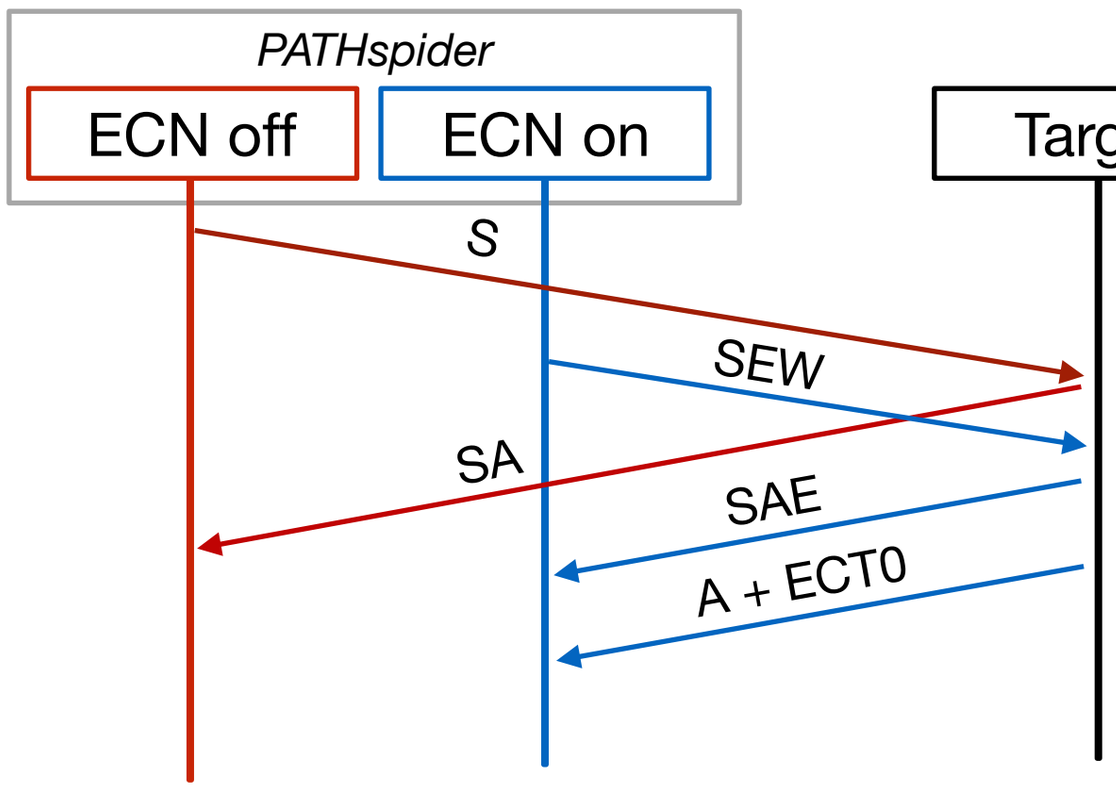
`offline`: no connection

`ecn.`*`codepoint`*`.seen`

generate traffic

negotiation $\xrightarrow{?}$ ECT marking

Methodology
Explicit Congestion Notification (ECN)



Connect **with and without ECN**

Measure

`ecn.connectivity.`*`status`*

`works`: off + on OK

`broken`: off OK, on fails

`transient`: on OK, off fails

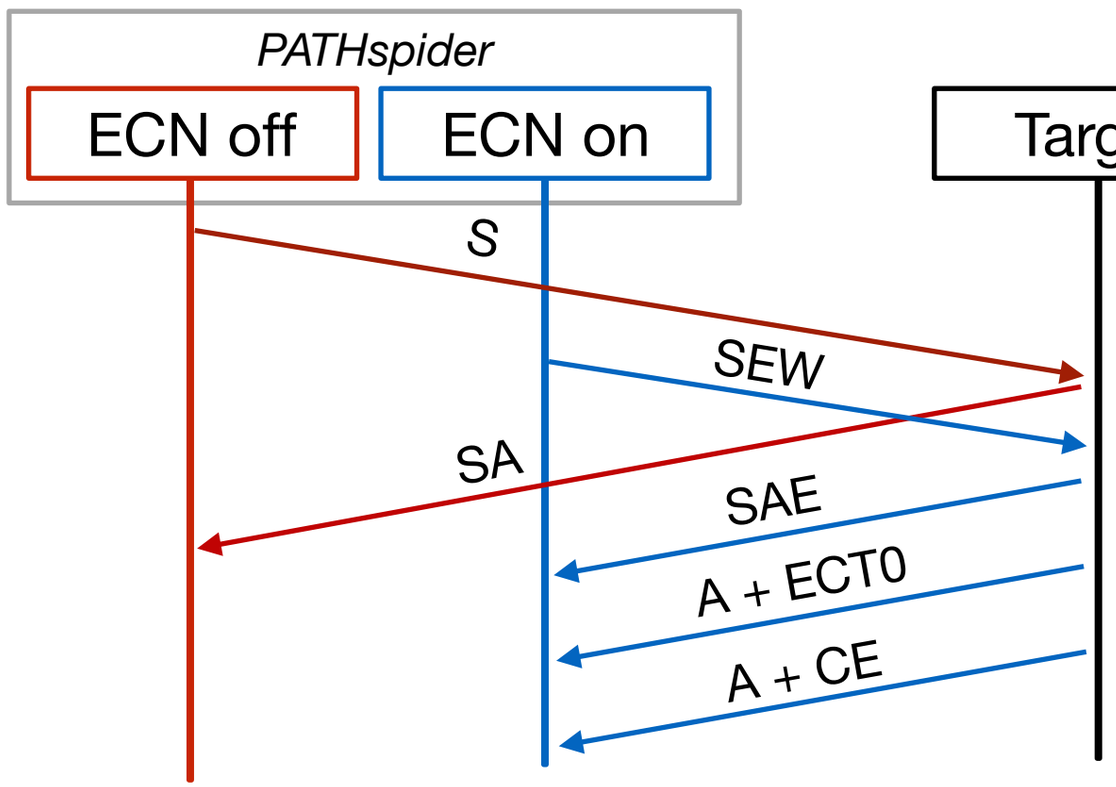
`offline`: no connection

`ecn.`*`codepoint`*`.seen`

generate traffic

negotiation $\xrightarrow{?}$ ECT marking

Methodology
Explicit Congestion Notification (ECN)



Connect **with and without ECN**

Measure

`ecn.connectivity.`*`status`*

- `works`: off + on OK
- `broken`: off OK, on fails
- `transient`: on OK, off fails
- `offline`: no connection

`ecn.`*`codepoint`*`.seen`

generate traffic

negotiation $\xrightarrow{?}$ ECT marking

Results: ECN

Server-side ECN deployment continues to increase

And hosters are figuring out how to make v6 work…

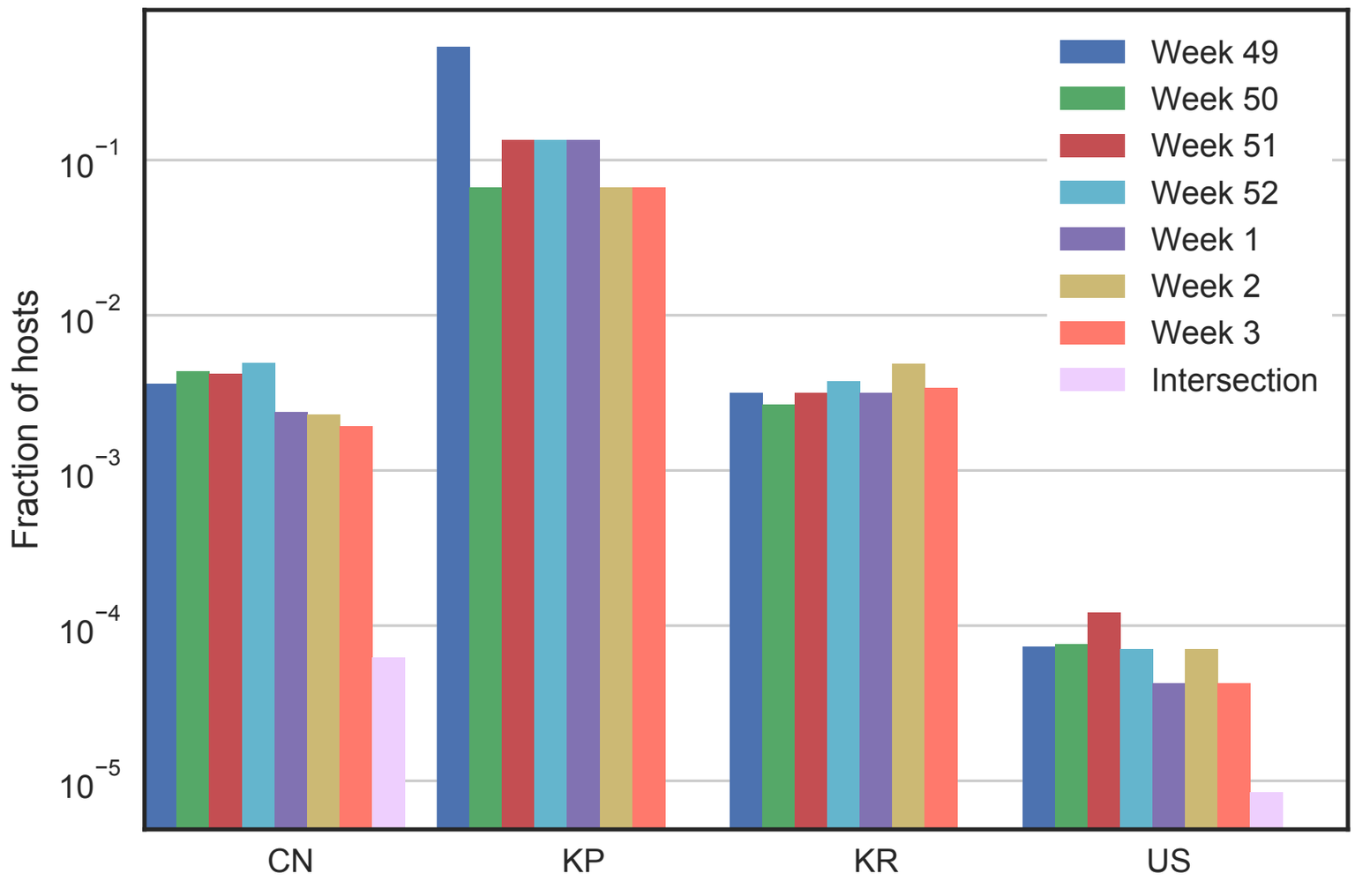
Continued: no change in errors in ECN once negotiated

June 2016				January 2017				
IPv4 n =617 873		IPv6 n =24 472		IPv4 n =675 289		IPv6 n =90 531		Description
hosts	pct	hosts	pct	hosts	pct	hosts	pct	
9221	1.49%	2637	10.78%	12583	1.863%	3621	4.000%	Completely failed to connect
432544	68.78%	20262	76.77%	498866	73.874%	82722	95.232%	Capable of negotiating ECN, of which:
11718	1.86%	2167	8.21%	15000	3.007%	6622	8.005%	Never mark ECT
-	-	-	-	30	0.006%	16	0.019%	Mark ECT1
1112	0.18%	964	3.65%	1851	0.274%	23	0.025%	Failed to connect w/ECN

On censorship and ECN interference

Heterogeneous TCP-layer censorship → more ECN path dependency

Automated measurements reduce path dependency noise floor



Conclusions

ECN brokenness good indicator for path impairments at L3/ L4

- Comprises additional codepoints both in IP and TCP
- Correlates with purposeful interference (censorship)

TCP Fast Open remains effectively a niche extension.

- Impairments are mainly access network linked

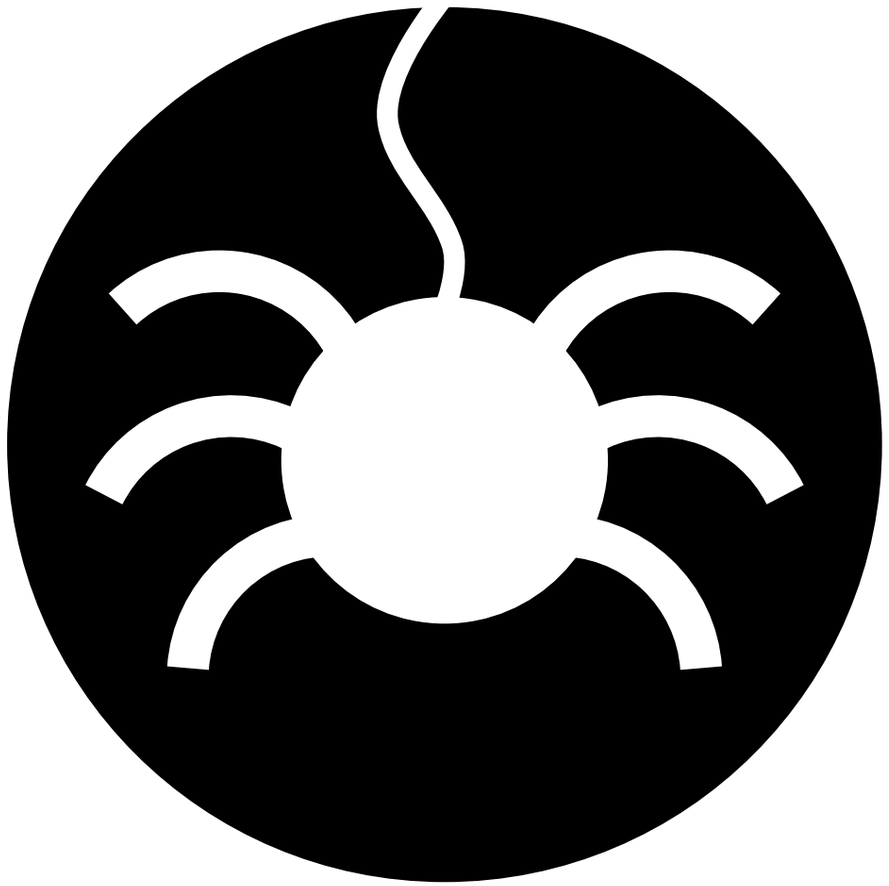
DSCP may be widely bleached

- But using DSCP doesn’t lead to connectivity risk.

Learn More

https://github.com/mami-project/pathspider

https://pathspider.net



```
# apt install pathspider
```

(2.0.0 release pending fixes to curl)