fling: A Flexible Ping for Middlebox Measurements

Ahmed Elmokashfi, Runa Barik, Michael Welzel,
Thomas Dreibholz, Stein Gjessing

AIMs 2018

March 14th 2018

simula

Metropolitan Center for Digital Engineering

Motivation

- Lack of a generic tool that can assess whether an arbitrary communication pattern between end points would succeed

  Will my new protocol/protocol-extension be blocked or modified by middleboxes?

simula

fling (flexible-Ping) is an end-to-end active measurement tool

- Allows testing whether an arbitrary sequence of packets can be exchanged between a fling client and a fling server
- Uses raw sockets and supports both IPv4 and IPv6
- Tests needs to be only specified at the server side
- Can narrow down the location of packet modification or drop

simula

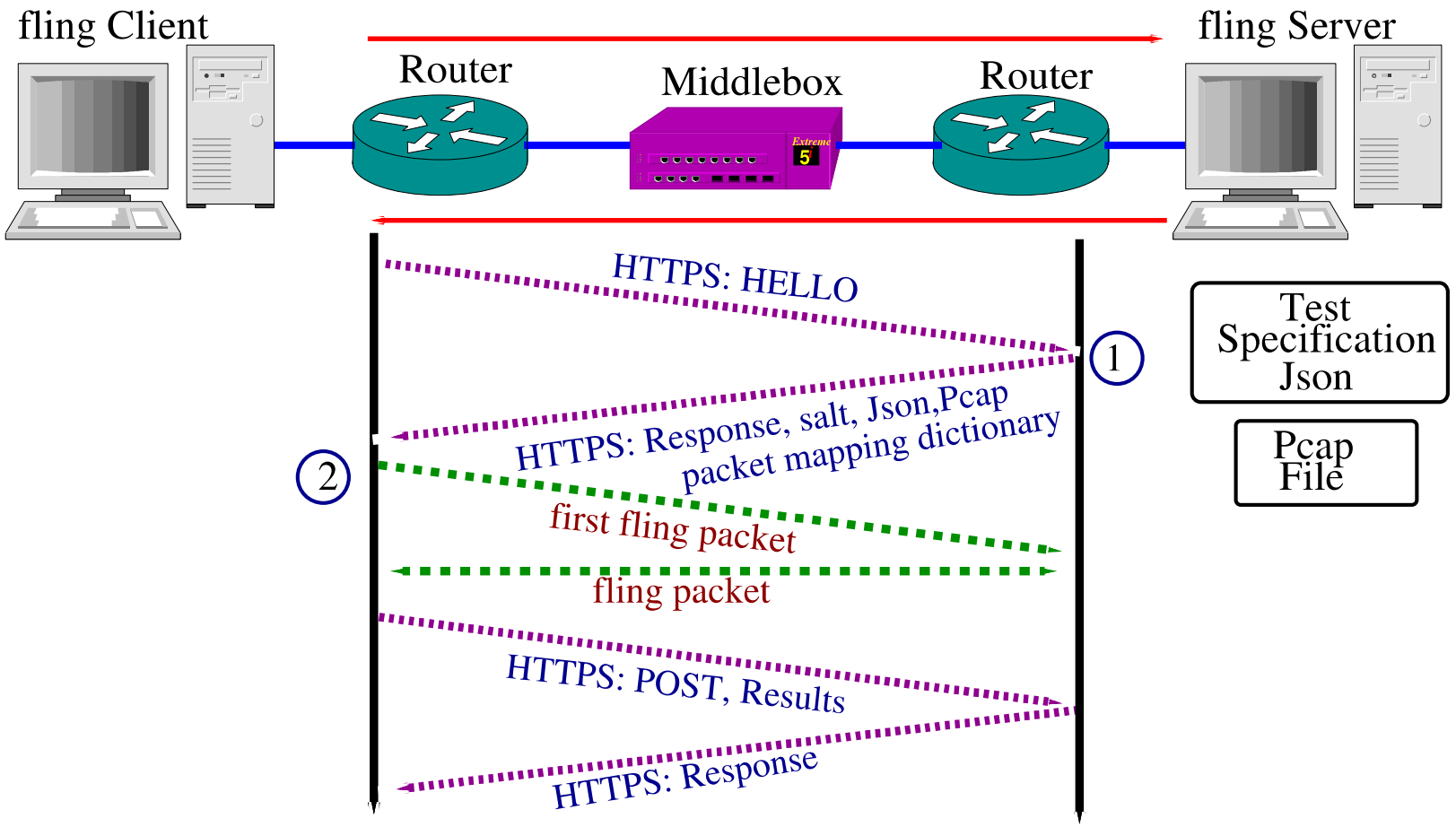
Middleboxes measurement tools

Tool	Raw sock-ets	Test protocols other than TCP	Test update: need to change	Fully con-trolled client-server dialogue	Detecting location of modification and drop
fling	✓	✓	Server	✓	✓
Netalyzr	✗	✓*	Server	✓	✓‡
TCPExposure	✓	✗	Both	✓	✗
HICCUPS	✓	✗	Both	✓	✗
Tracebox	✓	✓	Client	✗	✓‡
PATHspider	✓	✓	Client	✗	✓‡
TBit	✓	✗	Client	✗	✗

Table: Comparison of related tools. *ICMP,UDP; ‡One-sided only

simula

How does it work?

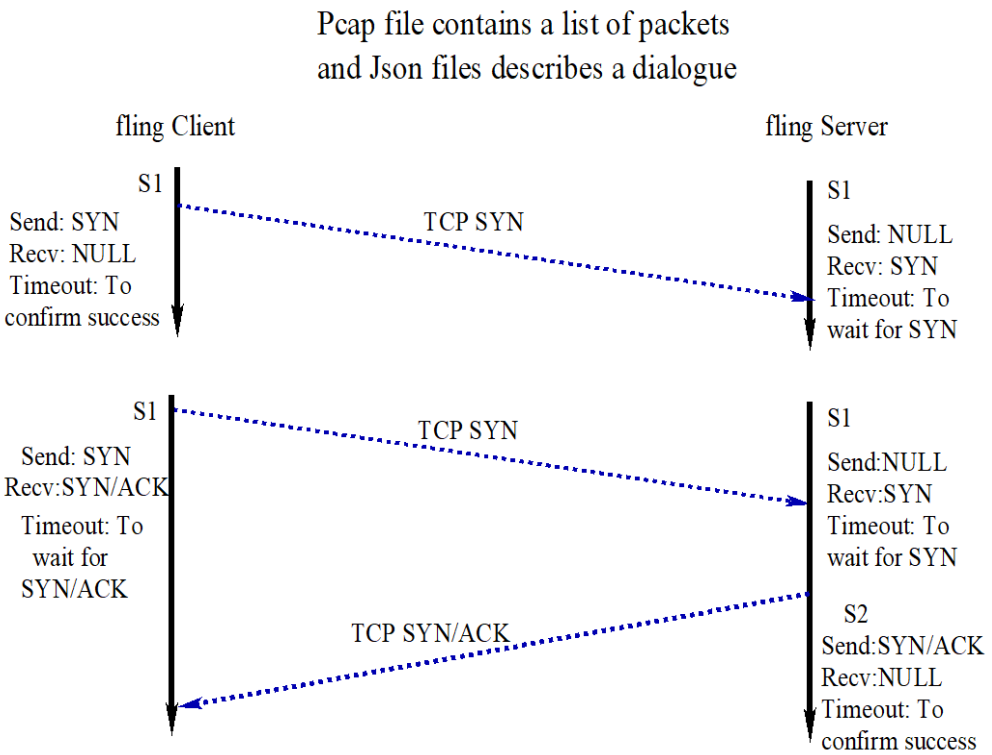


simula

How does it work?


```
{"name":"TCP SYN/ACK test",
"__index":{"0":"TCP SYN",
           "1":"TCP SYN/ACK"},
"packet_Info":[{
           "name":"TCP SYN",
           "swap":[0,2,2],
           "ChksumType":"adler-32",
           "ChksumPos":[16,0],
           "ChksumLen":[2,0],
           "ChksumPseudoHDR":true
           },{
           "name":"TCP SYN/ACK",
           "swap":[0,2,2],
           "ChksumType":"adler-32",
           "ChksumPos":[16,0],
           "ChksumLen":[2,0],
           "ChksumPseudoHDR":true}],
"client":{
        "state_sequence":["S1"],
        "states":[{
           "state":"S1",
           "send":["TCP SYN"],
           "recv":["TCP SYN/ACK"],
           "delaySend":[0],
           "timeout":[2000]
           }]},
"server":{
        "state_sequence":["S1","S2"],
        "states":[{
           "state":"S1",
           "recv":["TCP SYN"],
           "timeout":[2000]
           },{
           "state":"S2",
           "send":["TCP SYN/ACK"],
           "delaySend":[0],
           "timeout":[2000]
}]}}
```


(a) json file for a simple TCP SYN-SYN/ACK dialogue test



(b) Protocol instance

simula

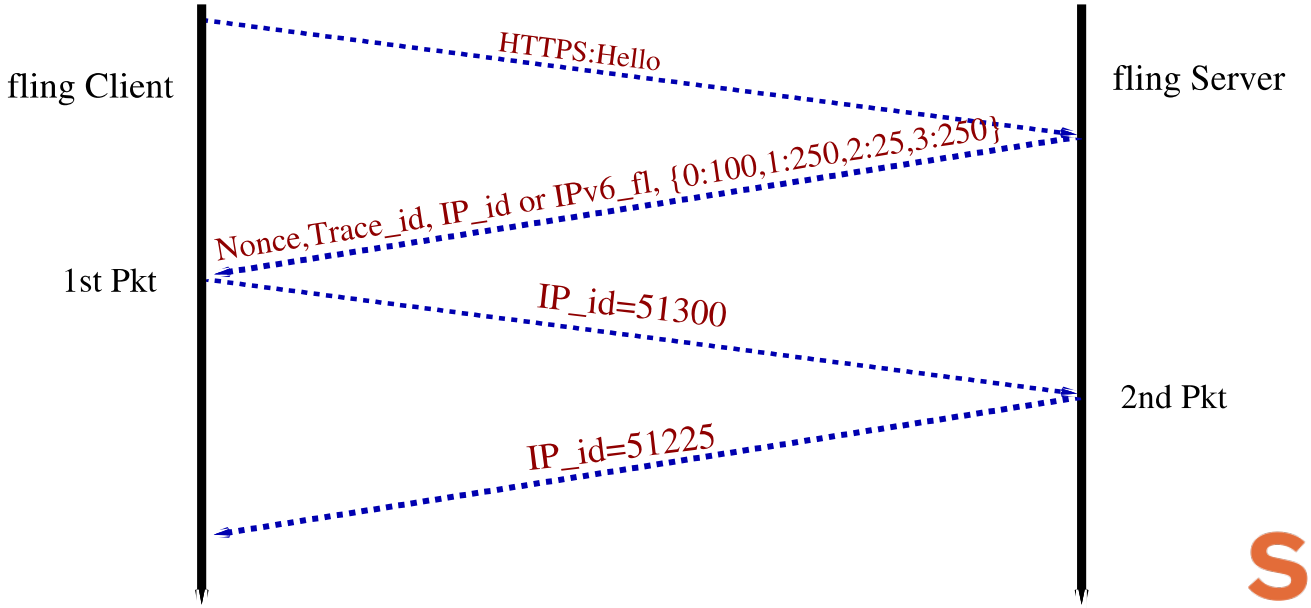
Challenges

- Mapping packets into corresponding test sequence
- Detect whether packets are really dropped
- Infer the location of packet modification or drop

simula

How to identify packets that belong to a particular test?

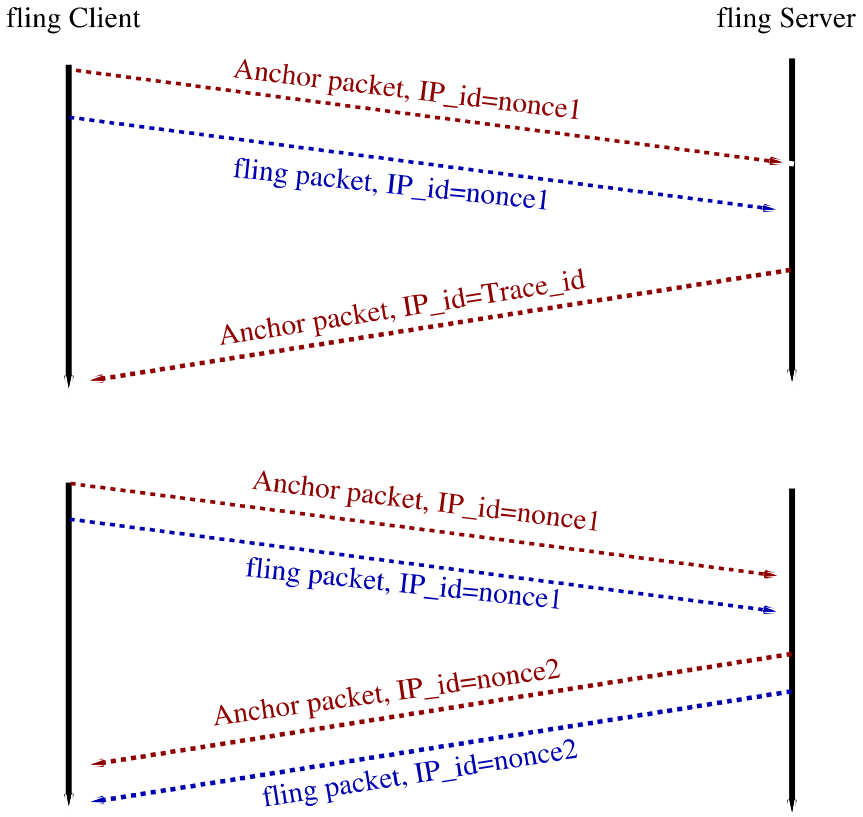
- fling uses nonce and protocol numbers for packet identification
- The packet's nonce is (salt,random_number)
  - Salt is 8-bit number generated by the server for each test
  - The server also generates a random number for each packet
  - The nonce position in the packet is defined in the Json file



simula

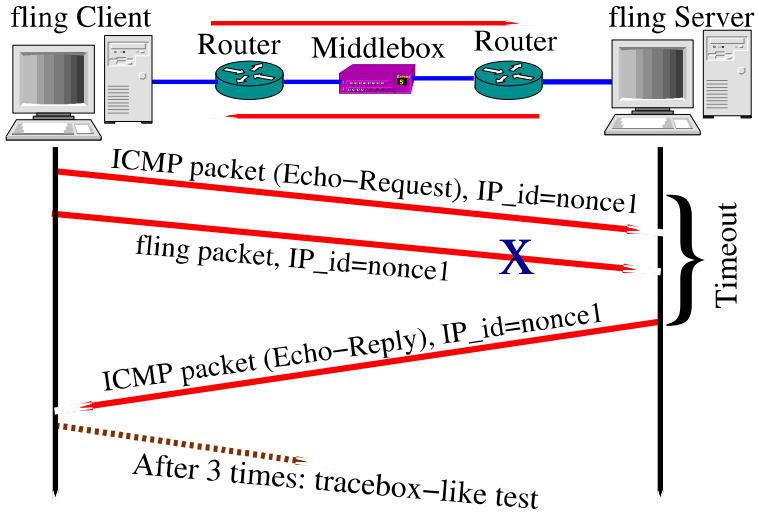
Detect the drop of test packets

- To confirm drops of tests packets fling sends, along with every test packet, an anchor packet (TCP SYN-SYN/ACK, UDP, or ICMP)



simula

Detect the drop of test packets

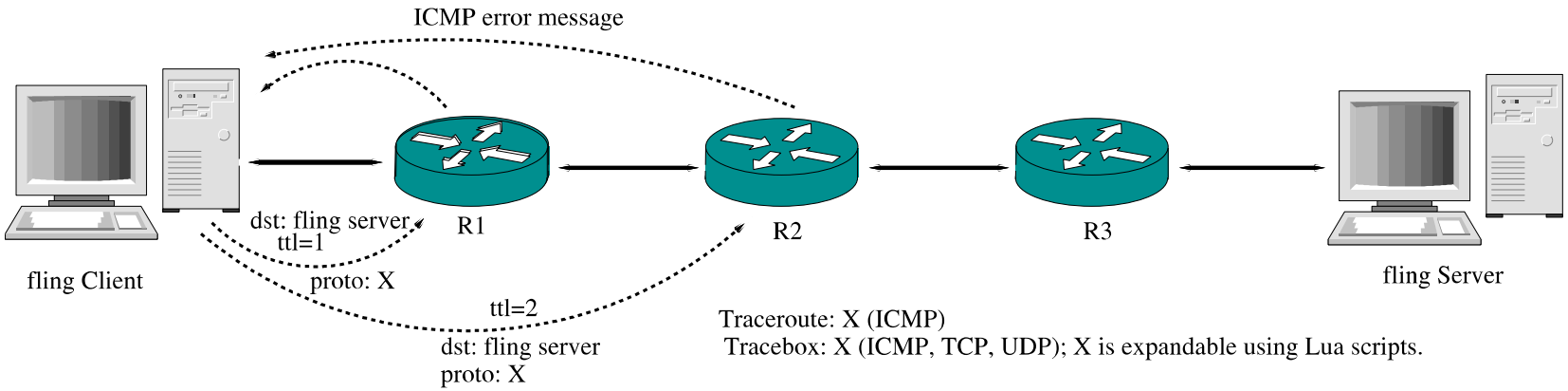


		X DROP; start tracebox-like test
DROPPED	DROPPED	Repeat 3×. Then, assume: CONGESTION; start tracebox-like test

simula

Detecting the location of modification or drop

- In case a test packet is dropped fling re-sends the test packet with an increasing TTL
- RFC 1812- compliant routers enclose the entire packet in the payload of the ICMP error message



simula

Case study: uses fling to check whether DSCP code points survive end-to-end paths

- WebRTC would like use DSCP code-points to signal QoS expectations but does it really work?
- We tested three DSCP values: CS1 (low priority data), AF42 (Multimedia conferencing) and EF (Telephony)

simula

Clients

Testbed	IPv4	IPv6
Ark	111	46
NorNet Core	40	19
PlanetLab	14	-

34 IPv4 servers
18 IPv6 servers

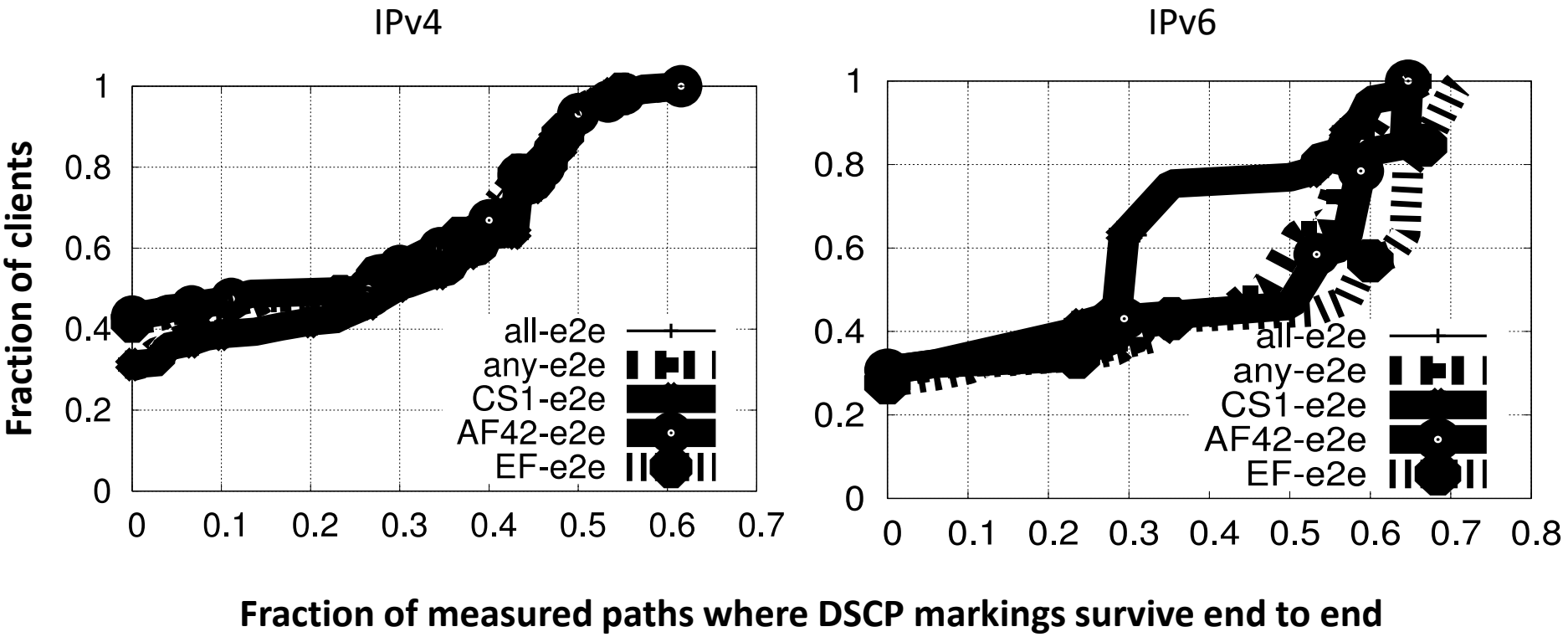
~10K IPv4 paths
~2K IPv6 paths

298 IPv4 Ases and 119 IPv6 ASes
All key large transit providers + many access providers e.g. ComCast, Bharti AirTel and CenturyLink.

simula

Case study: uses fling to check whether DSCP code points survive end-to-end paths

DSCP markings survived e2e in 33% and 50% for IPv4 and IPv6, respectively



simula

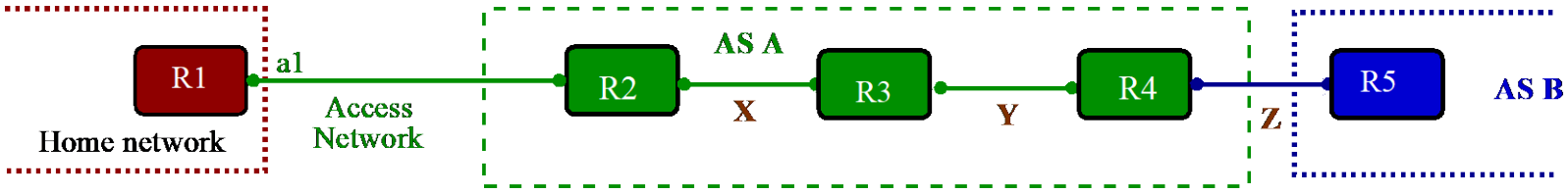
Do packets with DSCP markings risk being blackholed?

Code Point	Direction	Total failures	# clients	#servers
CS1	Forward	18	6	10
CS1	Reverse	74	27	31
AF42	Forward	28	9	16
AF42	Reverse	74	27	28
EF	Forward	28	9	17
EF	Reverse	76	23	32
All	Forward	13	3	6
All	Reverse	27	11	15

None of these failures happened at TTL 1 or 2

simula

Where was DSCP re-marked?

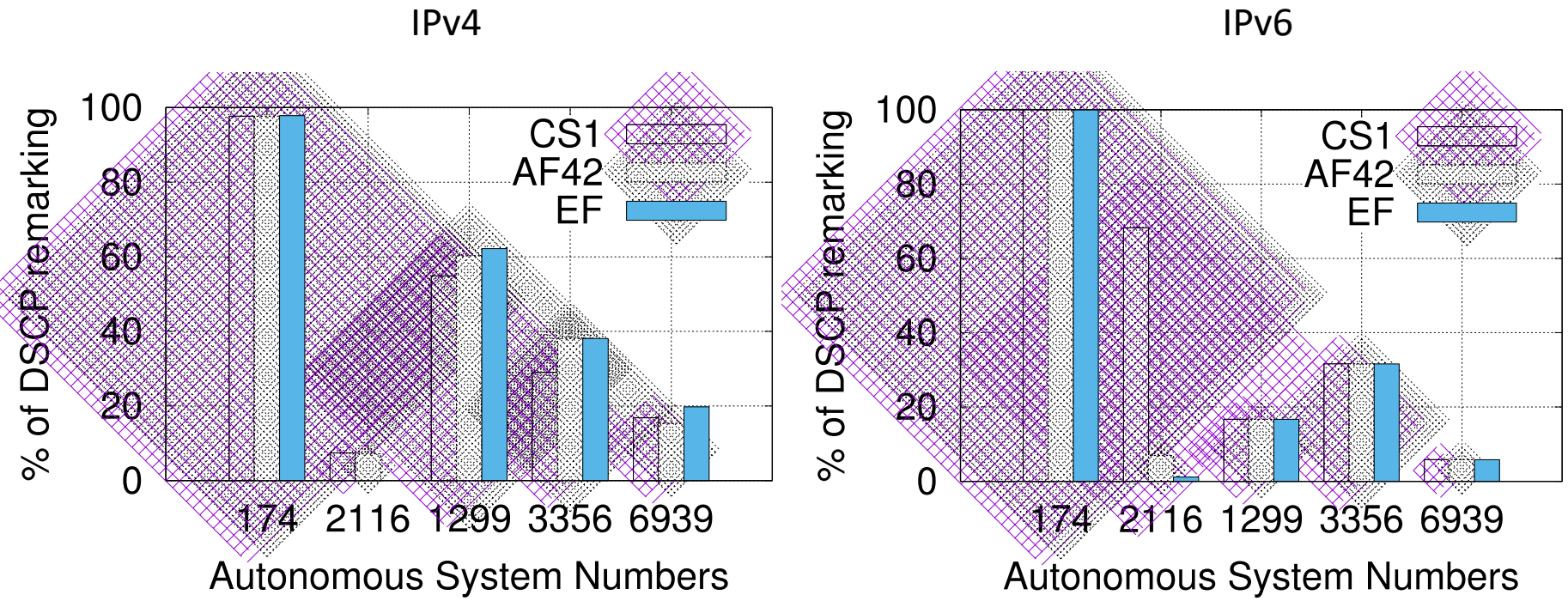


Changed in	IPv4	IPv6
Home network	21%	12%
First-hop AS	43%	31%
Beyond the first-hop AS	36%	57%

- Home gateways treats DSCP in a myriad of ways: zero, re-write to unused value, re-write to a used value
- First hop Ases often zero DSCP

simula

ASes beyond the first-hop AS employ a diverse set of re-marking policies



- Cogent remarks everything to either AF11 or AF21
- Other large ISPs do not seem to modify DSCP markings

simula

Limitations of the DSCP study

- Although we have around 10k paths, the coverage remains sparse
- The fact that DSCP marking survives does not imply that marked traffic will be treated differently
- All probes are in fixed networks

simula

Takeaways

- fling is a flexible tool that allows for a wide range of middlebox tests
- We have used fling to investigate whether DSCP markings survive routers and middleboxes
- Please help us increasing our coverage by running fling (email me ahmed@simula.no)

simula