

Rate-limiting of IPv6 traceroutes is widespread:
measurements and mitigations.



Pablo Alvarez

Florin Oprea

John Rula



Introduction

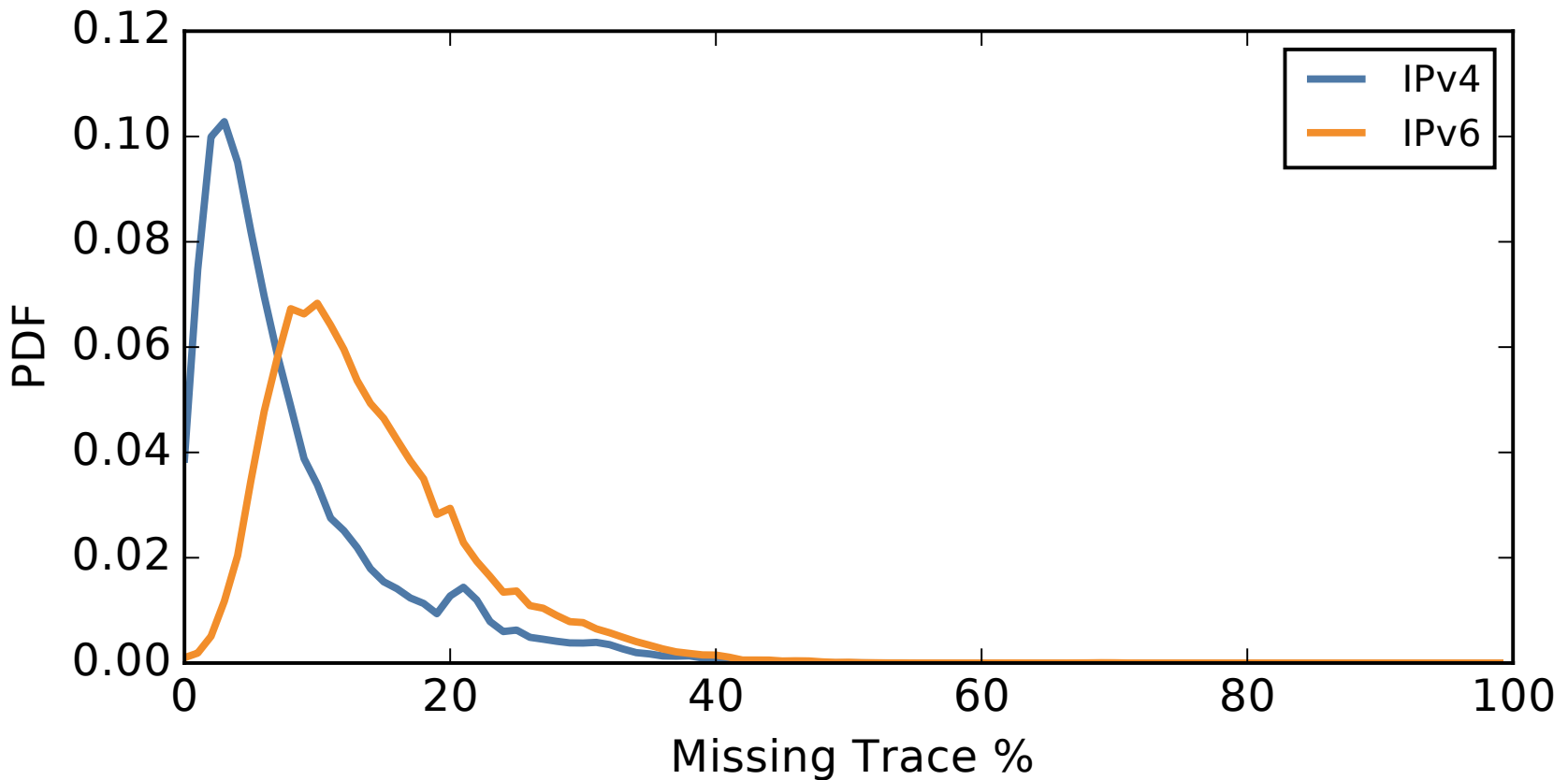
- Traceroute is an important tool in understanding overall Internet topology
- We have observed worse performance for IPv6 traceroutes compared to IPv4
- What are the underlying reasons for data loss, and the characteristics of this loss?
- Is there something we can do to improve performance within the current state of affairs?

How much worse are IPv6 traces than IPv4?

- Collect
 - ~20M traceroutes
 - to ~200K IPv4 and ~100K IPv6 targets
 - From ~8K sources
- Count the number of null hops before the last non-null hop on each trace
- Aggregate % missing hops for each target and for each source separately.

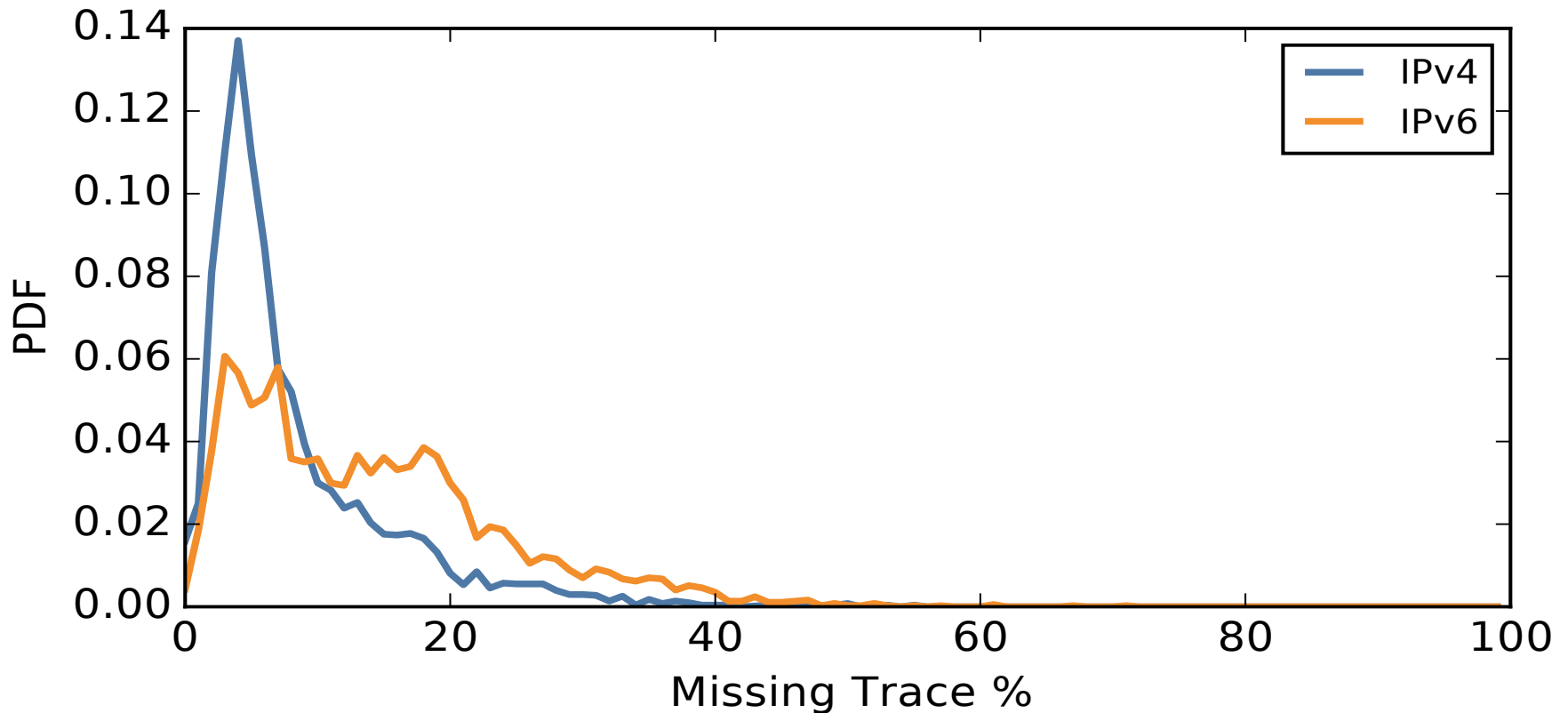
How much worse are IPv6 traces than IPv4?

Histogram of percent missing routers for each target



How much worse are IPv6 traces than IPv4?

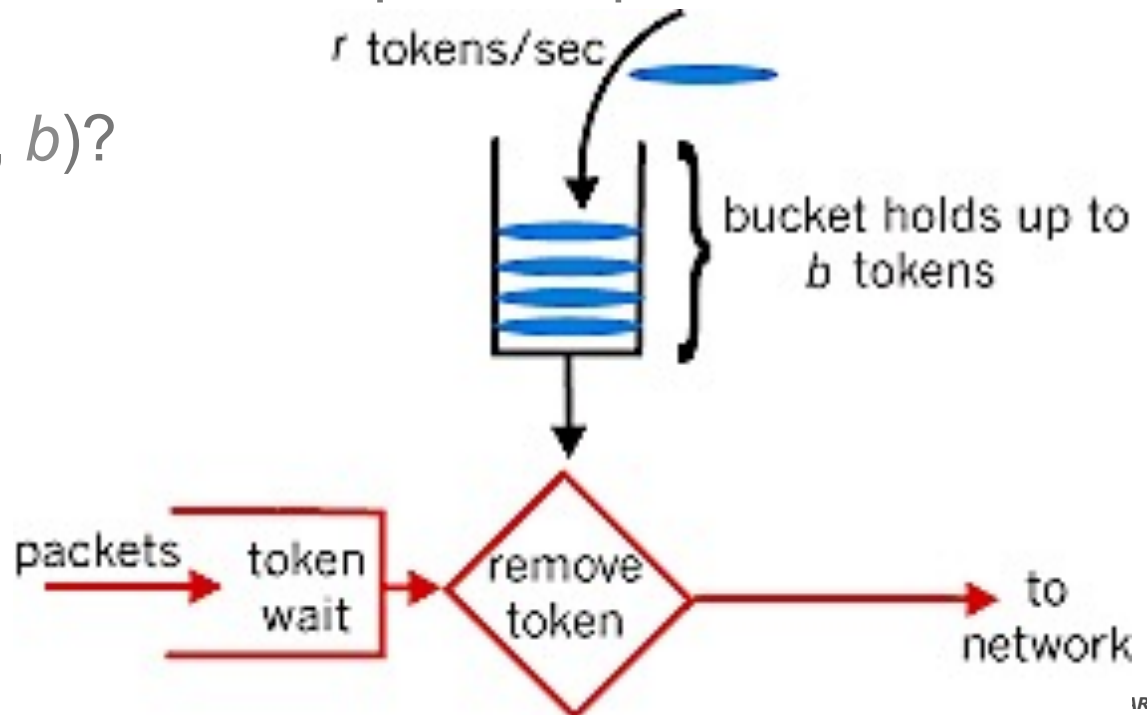
Histogram of percent missing routers for each source



Many more missing hops in IPv6 traces, whether we aggregate over source or over target.

Why routers drop IPv6 error packets

- Analysis shows presence of rate-limiting for v6
- RFC 4443 actually *requires* routers to rate-limit v6 error messages
- RFC recommends using token-bucket for rate-limiting
- Token-bucket(r, b) allows r error packets per second, bursts of b packets
- Can we estimate (r, b) ?

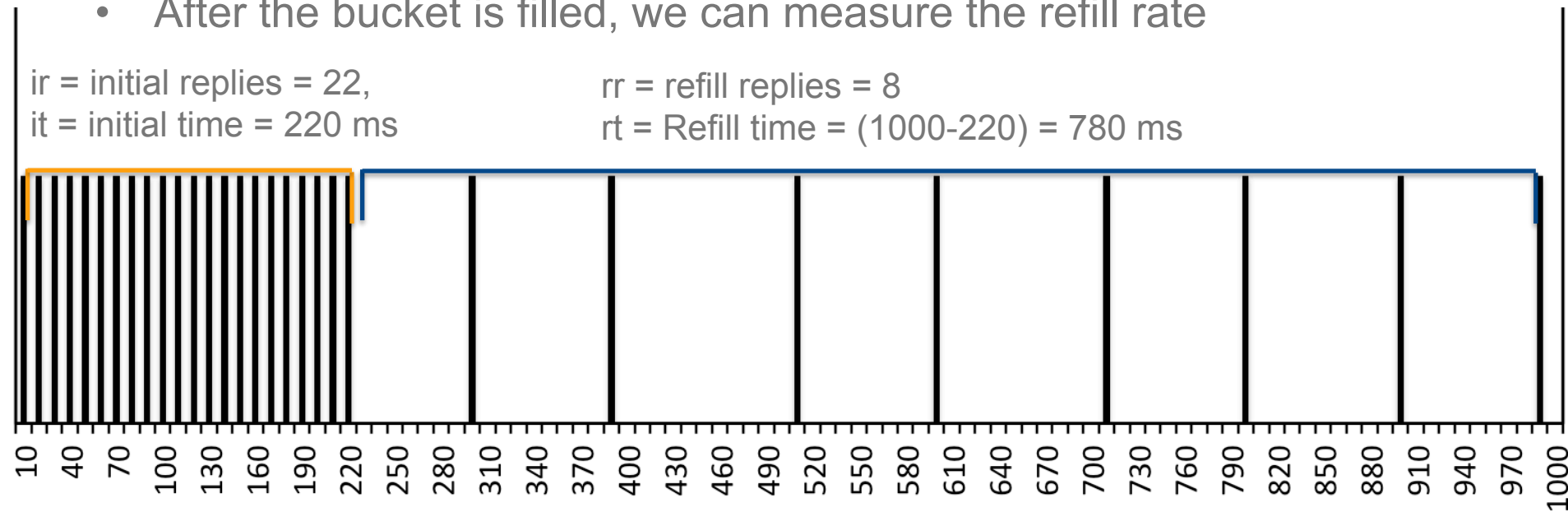


Measuring ICMPv6 error packet rate limiting

- Send a packet with the same TTL to the same target every ~10ms
- See how many come back, and when
- The “clump” at the start is the bucket size
- After the bucket is filled, we can measure the refill rate

ir = initial replies = 22,
it = initial time = 220 ms

rr = refill replies = 8
rt = Refill time = (1000-220) = 780 ms

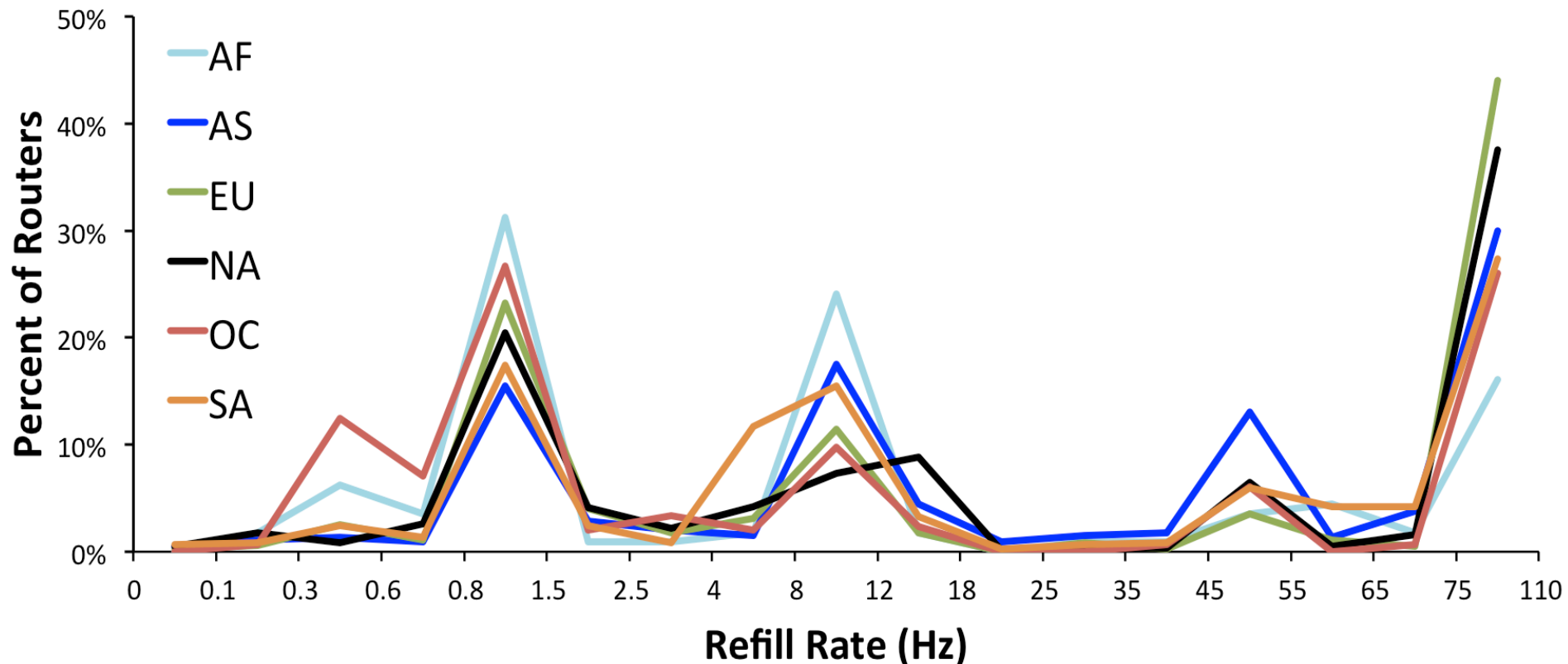


Time at which ICMPv6 error packets were received (ms)

- refill rate estimate = $rr/rt = 11$ Hz
- Bucket size = $ir - (it * \text{refill rate}) = 20$
- Adjusted refill rate = $(rr + (ih - \text{bucket size})) / rt = 10$ Hz
- Note: if refill rate > 66Hz, it's very difficult to find end of clump: ignore bucket size

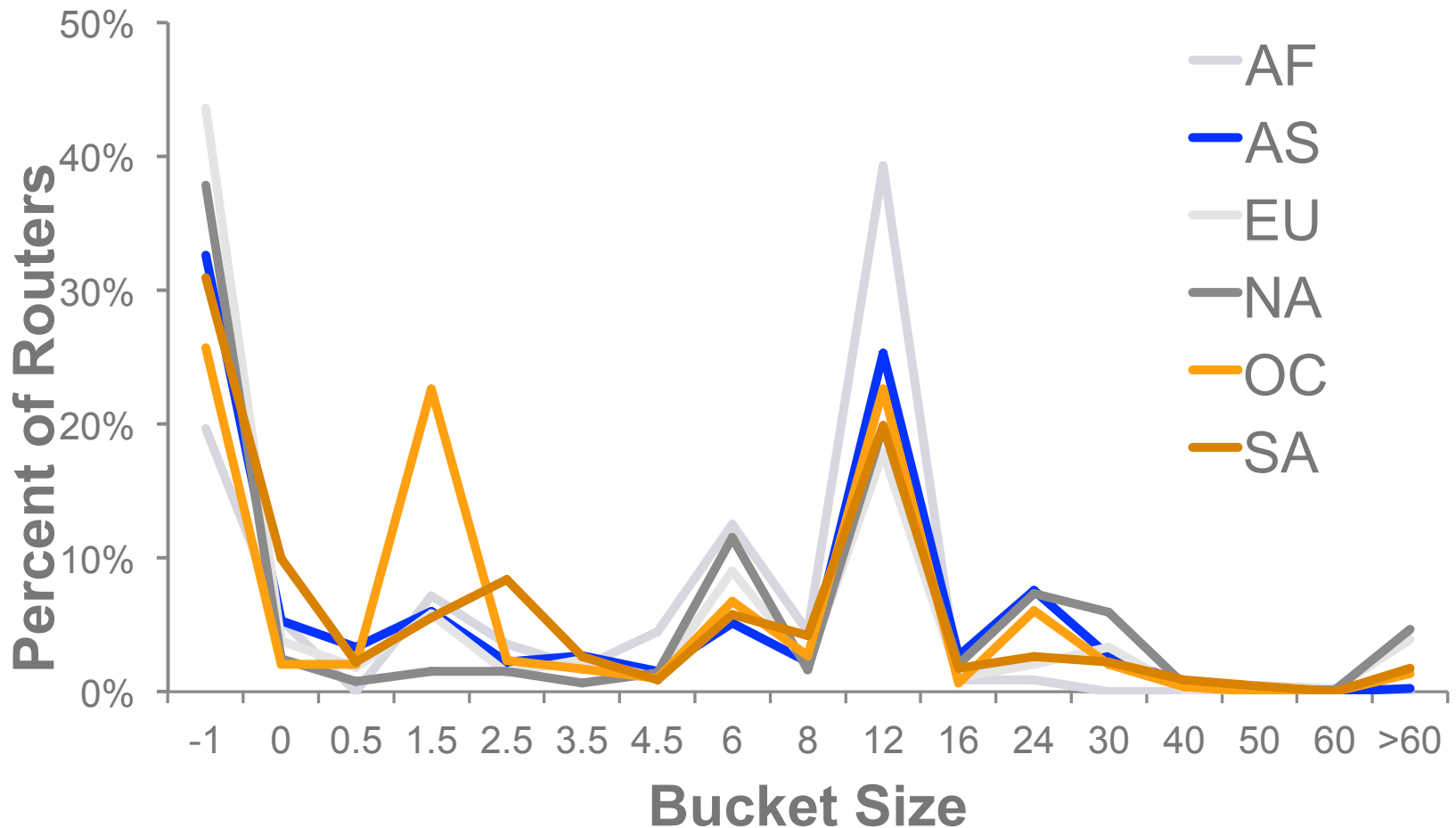
Results: many routers show similar rate limits

- 3078 routers in 6 Akamai continents (112 in AF ... 906 in NA)
- 18% of hops (8700/48000) had more than one router, not used.
- ~1/3 of routers do not show rate-limiting at the ~100Hz frequency we tested.
- There are consistent peaks in refill rate across continents, probably reflecting default factory settings on the routers



Routers also show similar bucket sizes

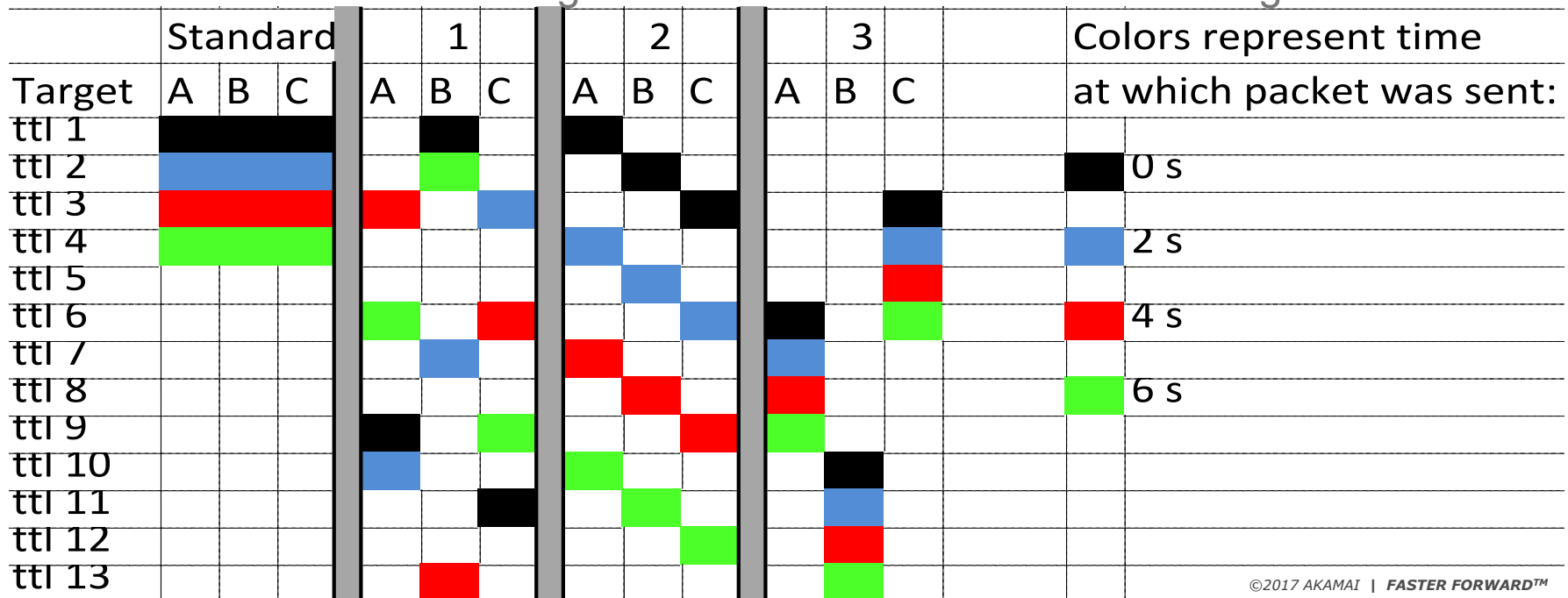
- Again, the data support the idea most routers are set to default configurations
- Size == -1 indicates no bucket size detected (router allows rates close to 100 Hz).



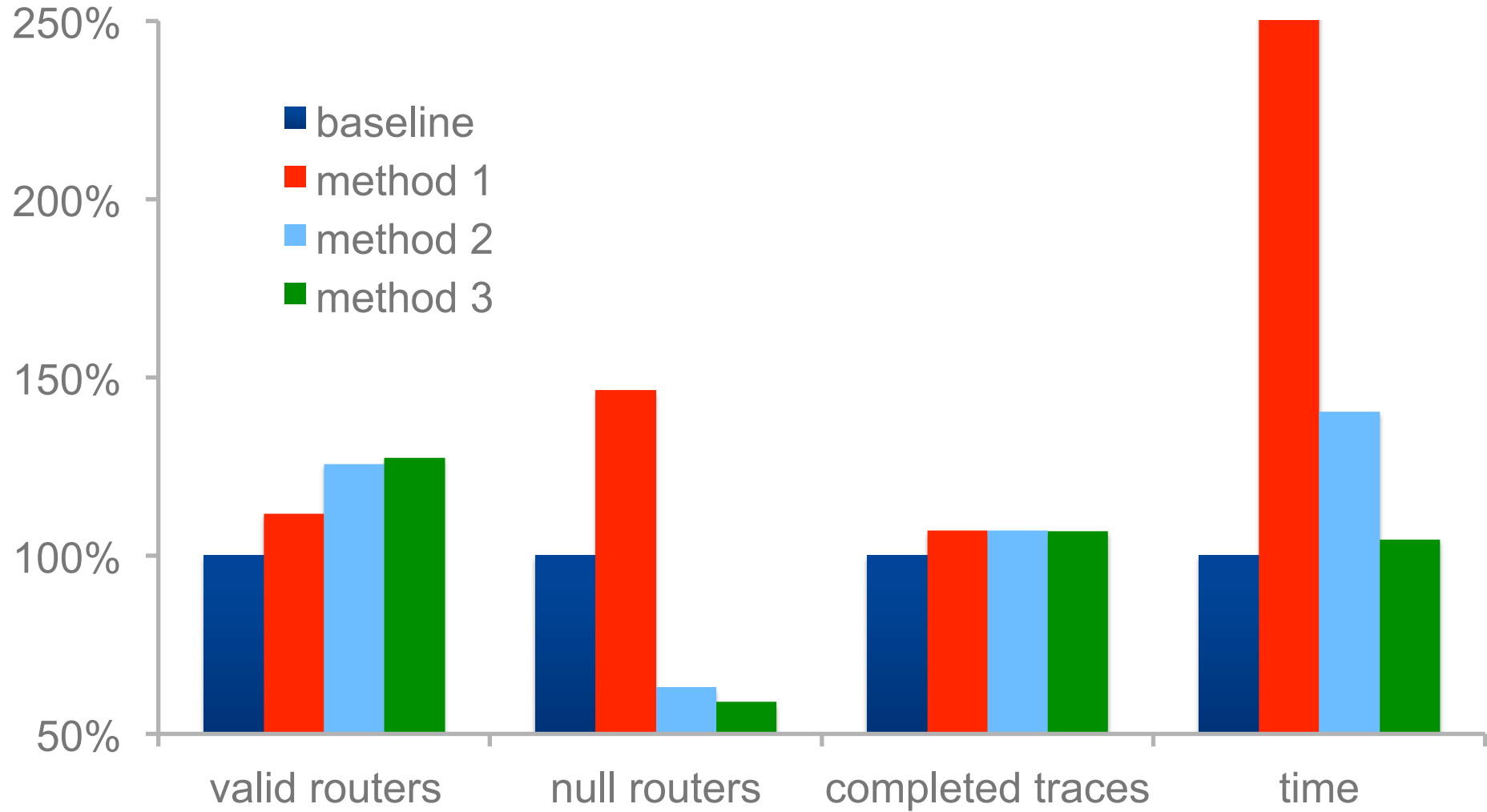
Mitigation 1: Spread out TTLs

- Standard method: send out a packet with TTL 1 for each target, wait for return, send out packet with TTL 2, etc...
- This is the worst possible way to do things for multiple targets: since many traces hit the same routers at the same time, we empty the token bucket as fast as we can and do not allow time for refills.

1. Random TTLs overall
2. TTLs increasing across targets
3. Start trace at random TTL for each target.



How much does spreading out TTLs help?



Method 3 seems the best choice: pretty fast, and clear improvements

Mitigation 2: Fill in traces with known data

- If a router's bucket is empty, that means it has previously sent back at least one error packet
- We are likely to have that information, and can use it to fill in subsequent trace efforts

Example:

Trace 1: A->**B**->C->D->E->F->G->H

Trace 2: A->*->C->**T**->U->V->W

Trace 3: A->*->C->*->U->M->N->O->P

Trace 3 filled in: A->**B**->C->**T**->U->M->N->O->P

Caveats:

- Cannot keep old data around too long
- Ignores load balancing routers
- Sometimes we just don't have the data



Summary

- Starting many IPv6 traces at the same time is problematic: routers will drop many of the return packets.
- Different routers have very different rate-limiting properties
- Rate-limiting properties are little known and appear to remain at factory defaults.

Proposed mitigations:

- Changing the order of TTLs to avoid having multiple traces hit the same router at nearby times.
- Using recently collected traces to complete other traces
- Both solutions help, improvements on the order of 10-20% more routers