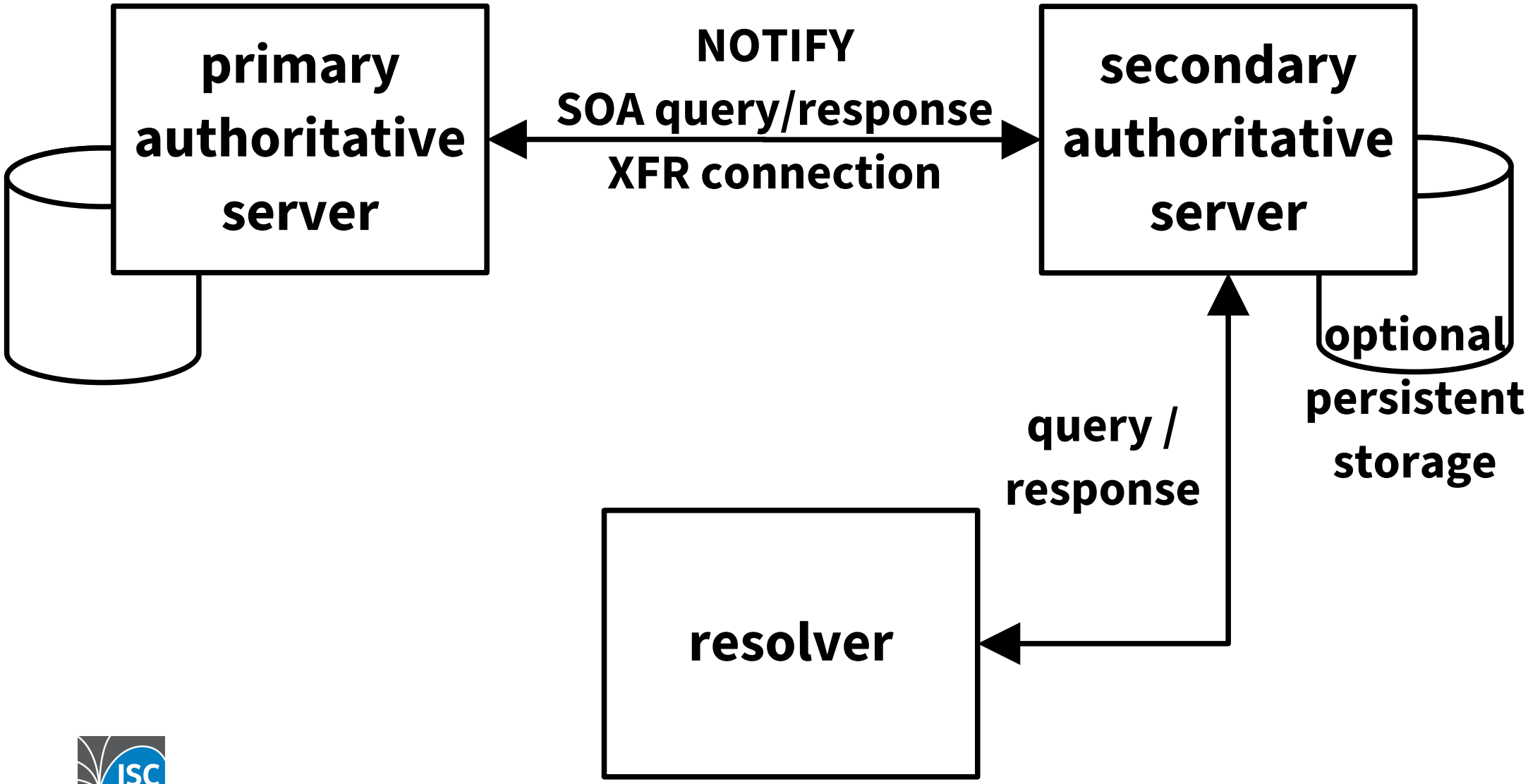
Zone transfer benchmarks: Lessons learned

Petr Špaček

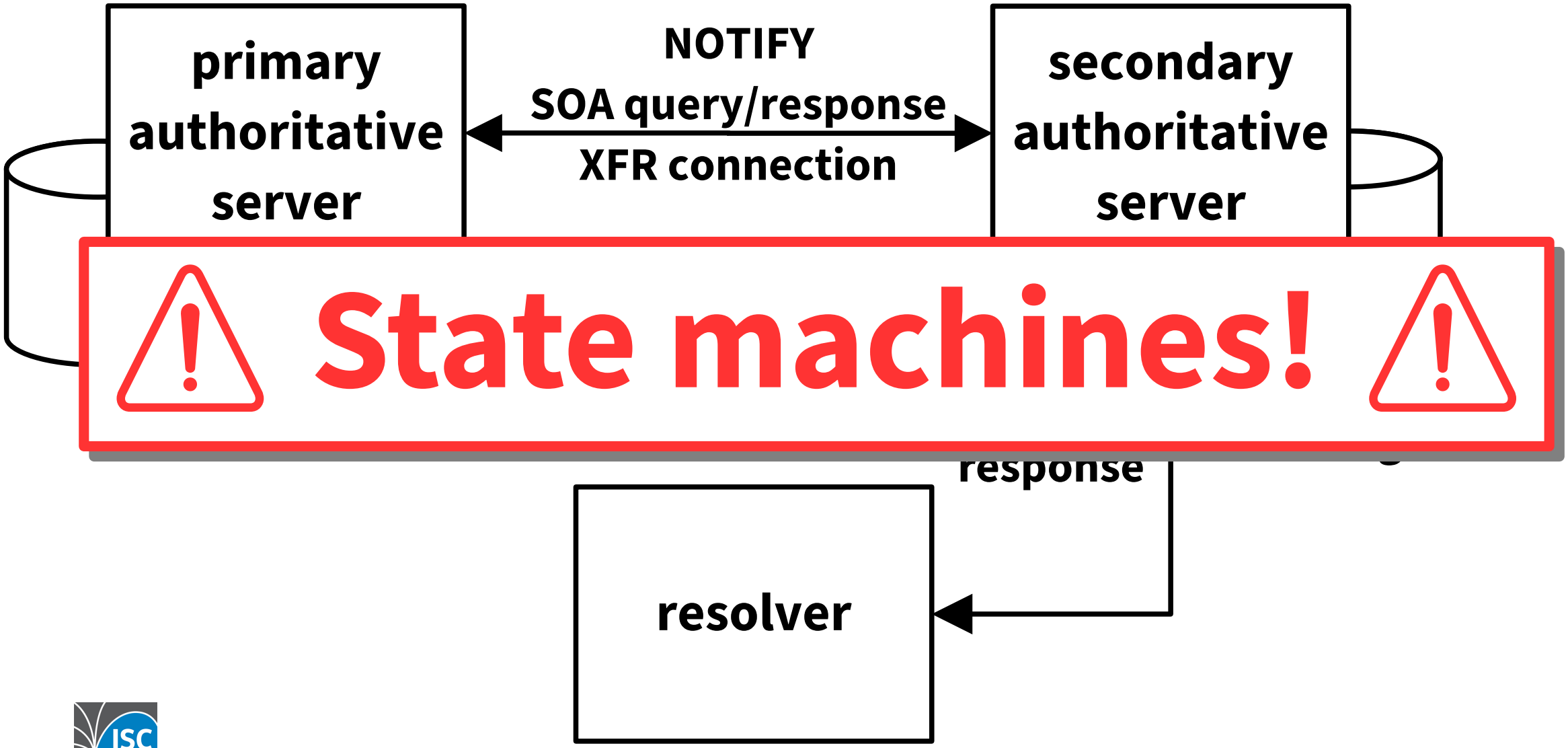
2024-10-26

pspacek@isc.org

XFR vs. DNS query



XFR vs. DNS query



Metrics

- QPS capacity / memory usage
  - steady / transient states
- primary : secondary ratio
- answer latency
- change propagation delay (SLA?)
- server startup time
- …

Caveats

- **Validate test environment!**
- See DNS Benchmarking 101: Essentials and Common Pitfalls @ OARC 42
- No 'echo server' for XFR
- Server-specific config, logs, data formats …
- State machine …
- TSIG, TCP/TLS/…

Minimal zone

Should be immediate, *right*?

Minimal zone

- SOA RR + 1x NS RR
- transfer time =
  (zone load timestamp) – (server start timestamp)
- transfer time = **-1 day, 23:59:59.354000**

Minimal zone

- SOA RR + 1x NS RR
- transfer time =

  - loaded 2024-10-14T12:40:32Z
  - startup 2024-10-14T12:40:32**.646**Z

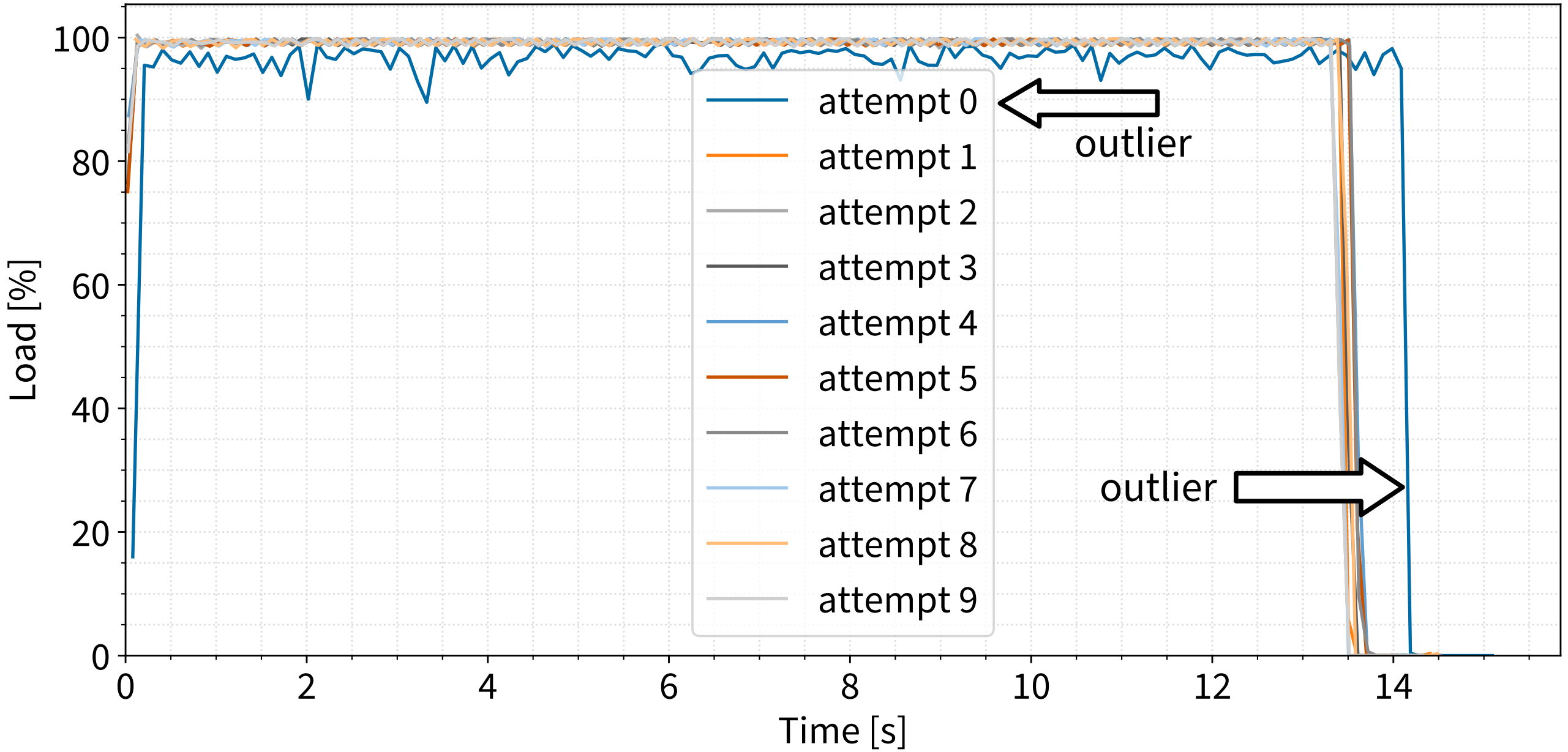
TLD

Two dozen seconds and no trouble,
right?

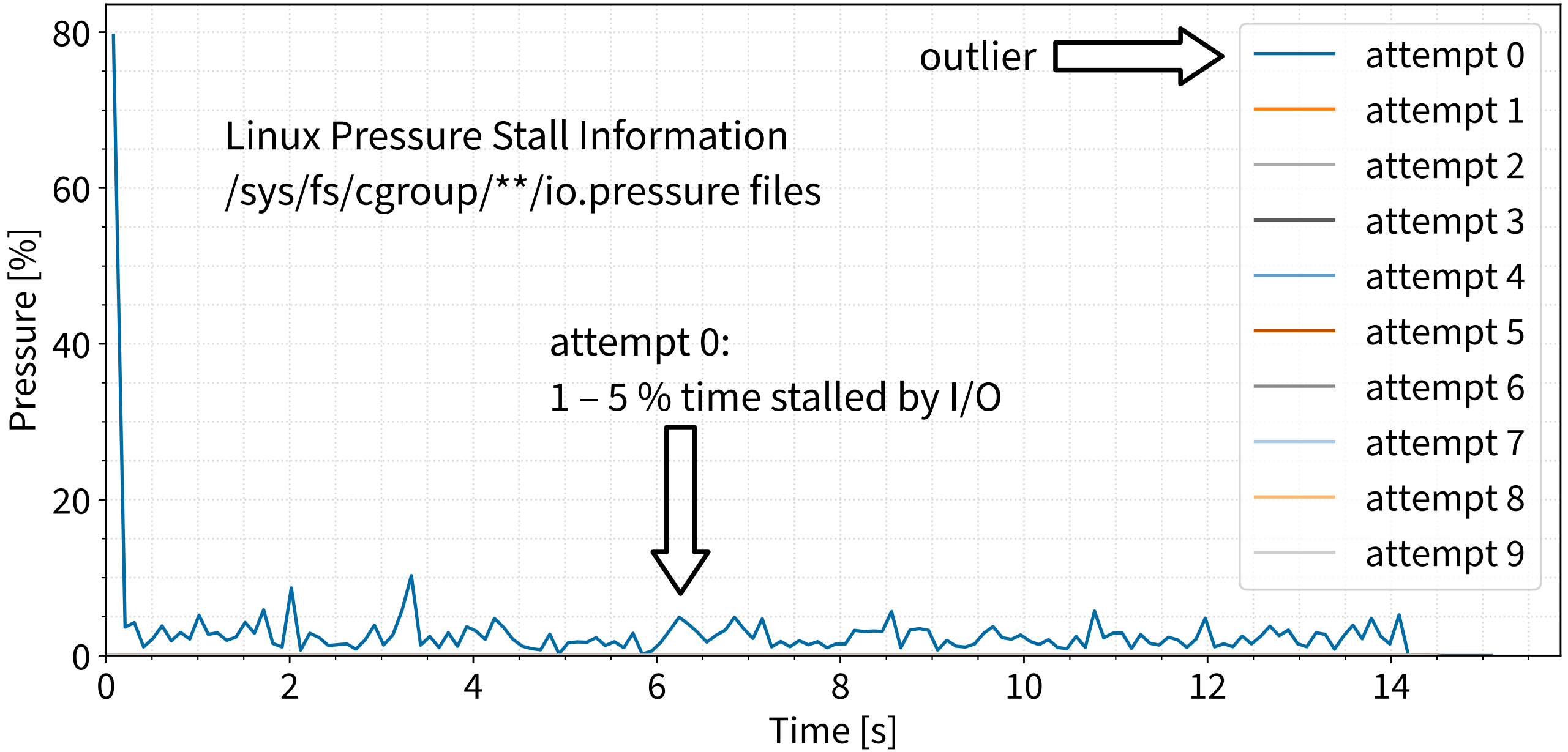
TLD AXFR

- ch. TLD
- 14 M records
- 691 M bytes
- 47 700 messages
- ~ 26.5 secs over UDP+TCP
  - ± 0.6 % across 10 test runs

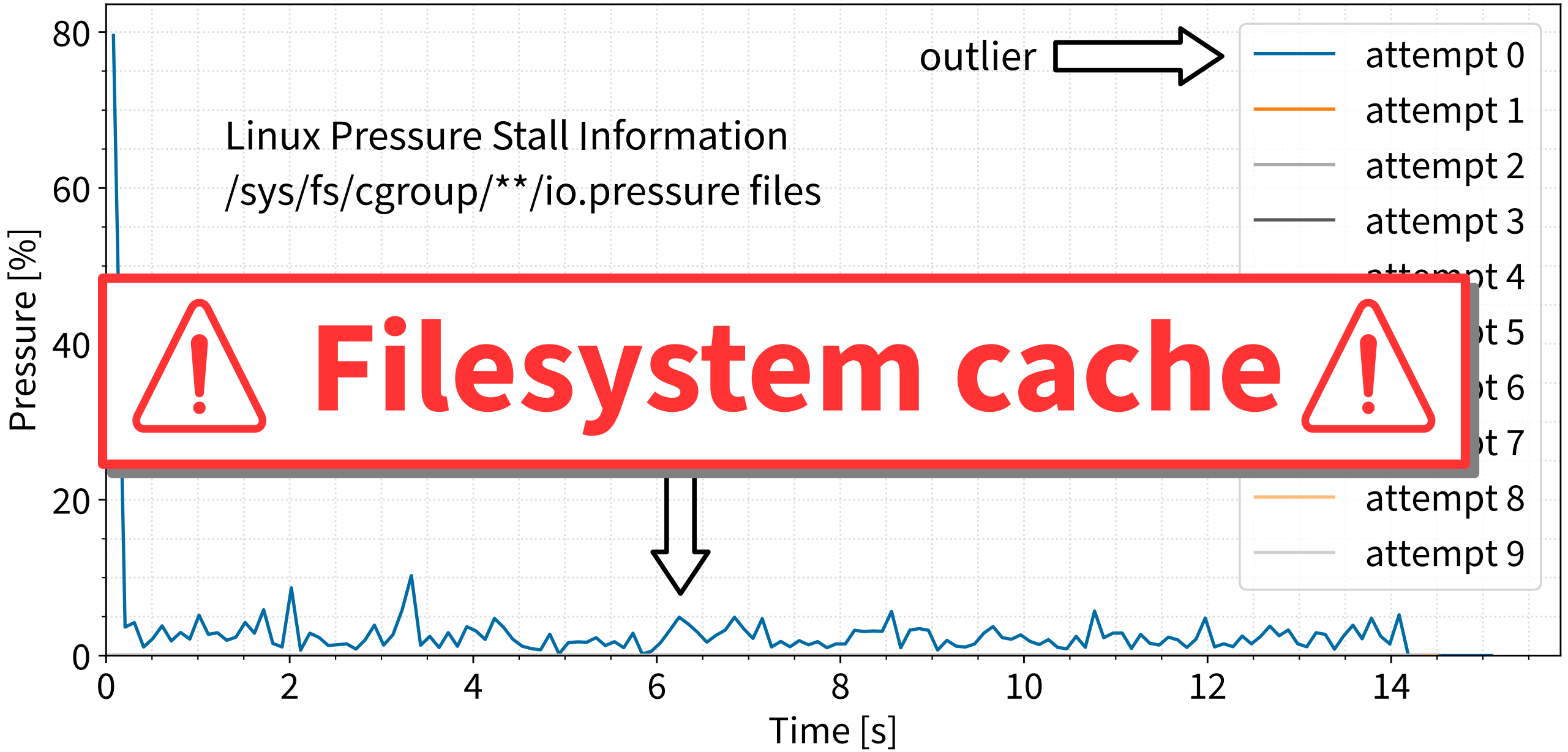
TLD, primary zone load, no TSIG, CPU



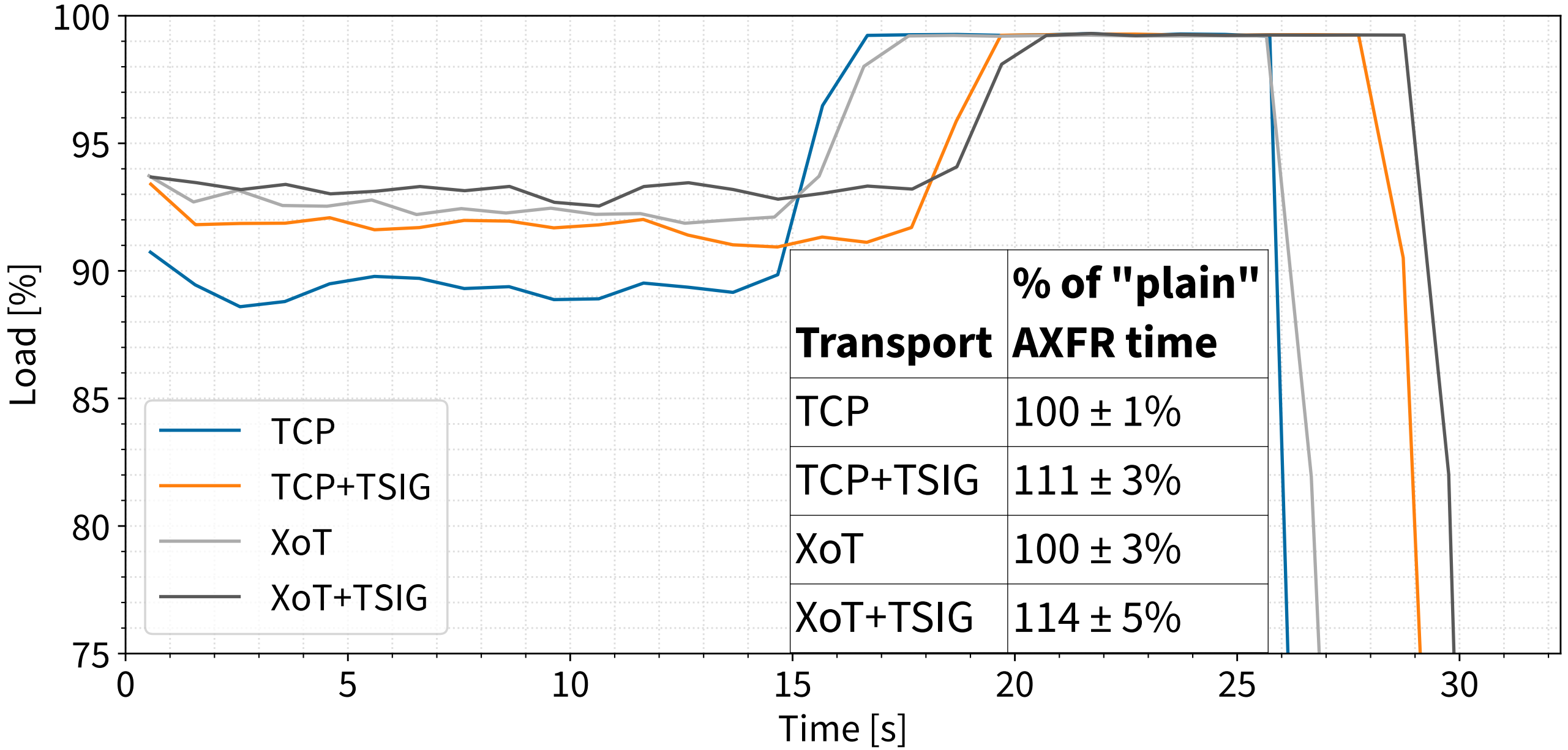
Primary zone load, no TSIG, I/O wait



Primary zone load, no TSIG, I/O wait



TLD, secondary, CPU load & time



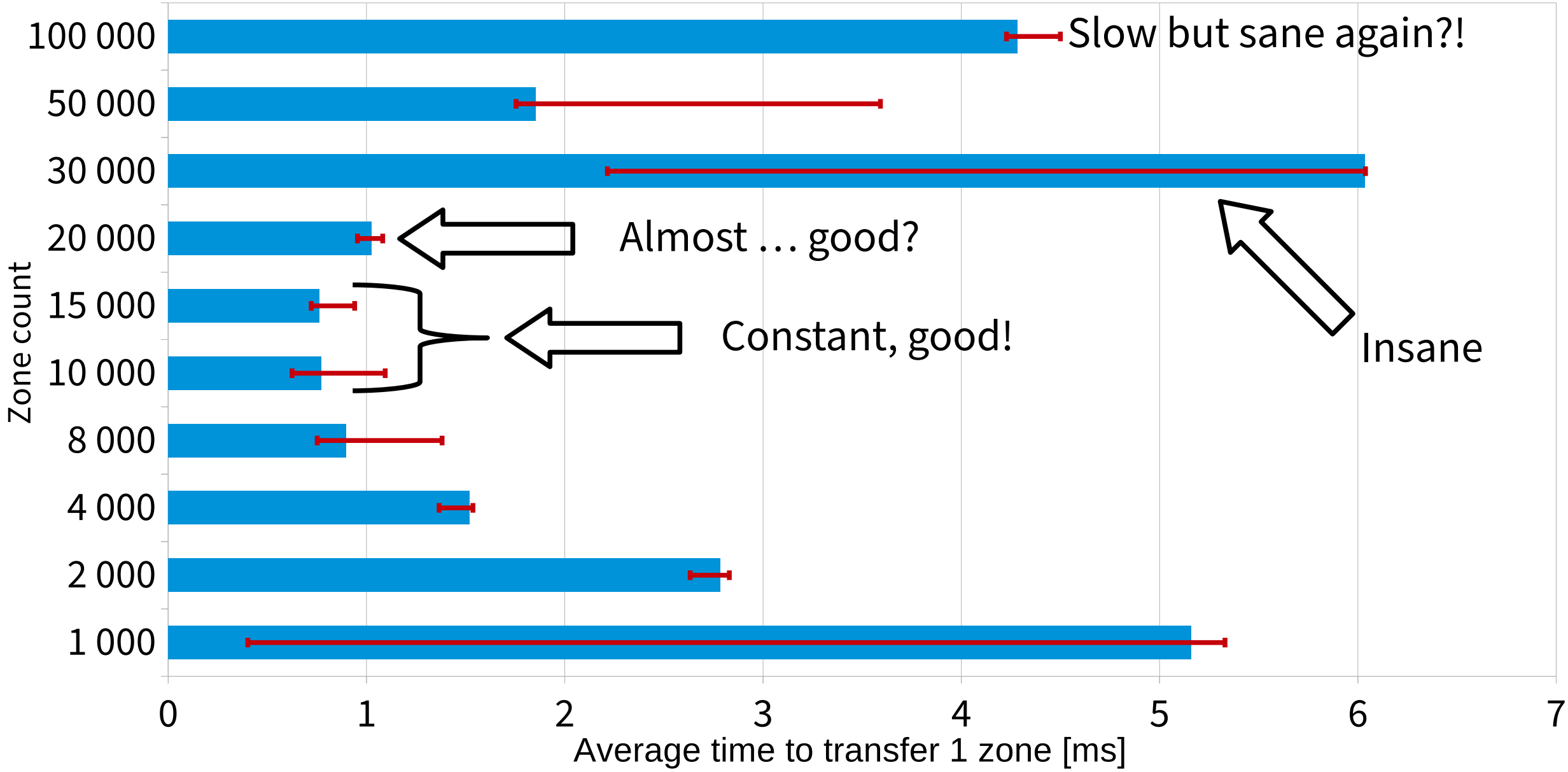
Transport	% of "plain" AXFR time
TCP	100 ± 1%
TCP+TSIG	111 ± 3%
XoT	100 ± 3%
XoT+TSIG	114 ± 5%

Lots of minimal zones

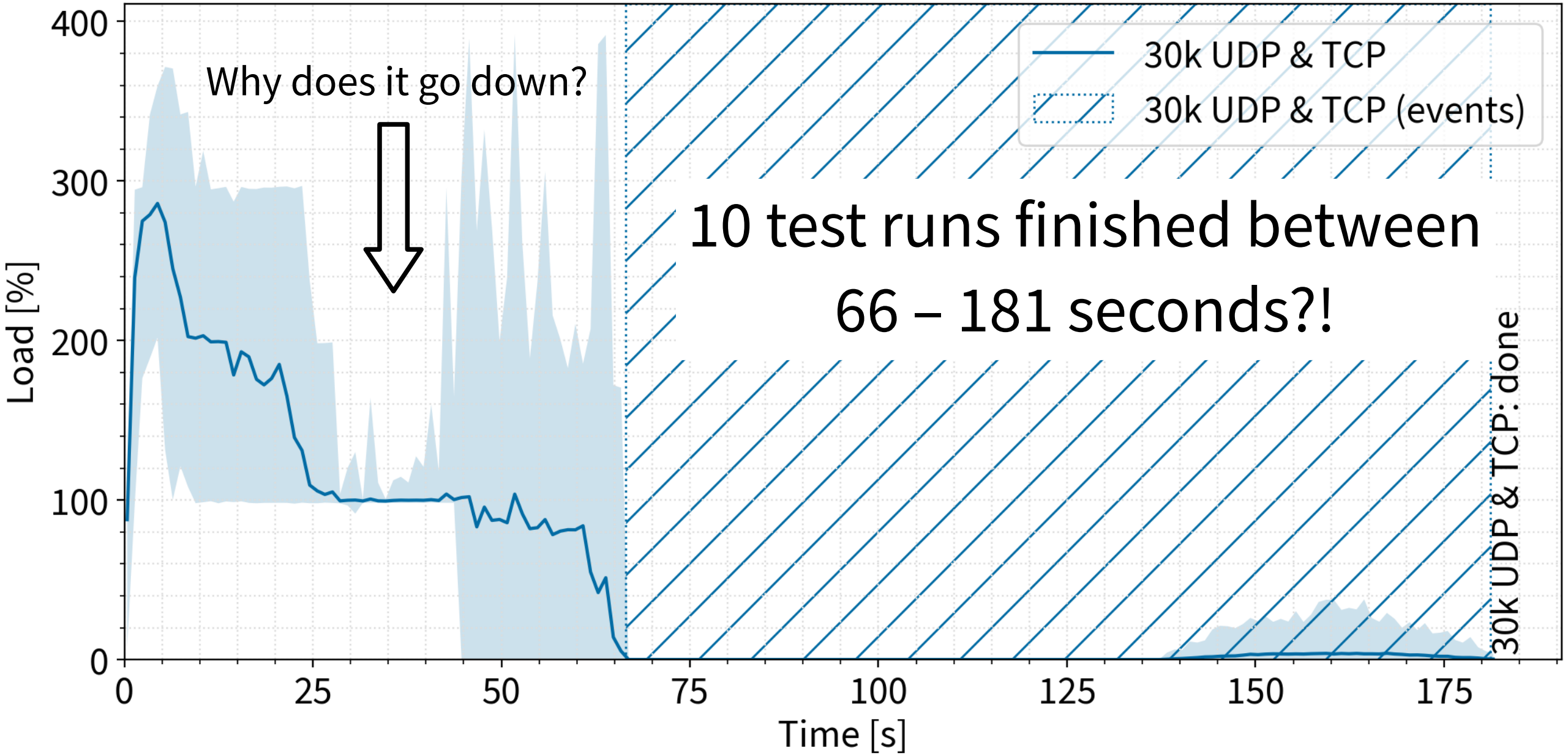
That's a lot of zones

- 1, 2, 4, 8, 10, 15, 20, 30, 50, 100 k zones
- Typical hosting setup
- Secondary cold start time
  - No local data
- **Total time linear with number of zones?**
- **Different transports?**

UDP & TCP, transfer time per 1 zone



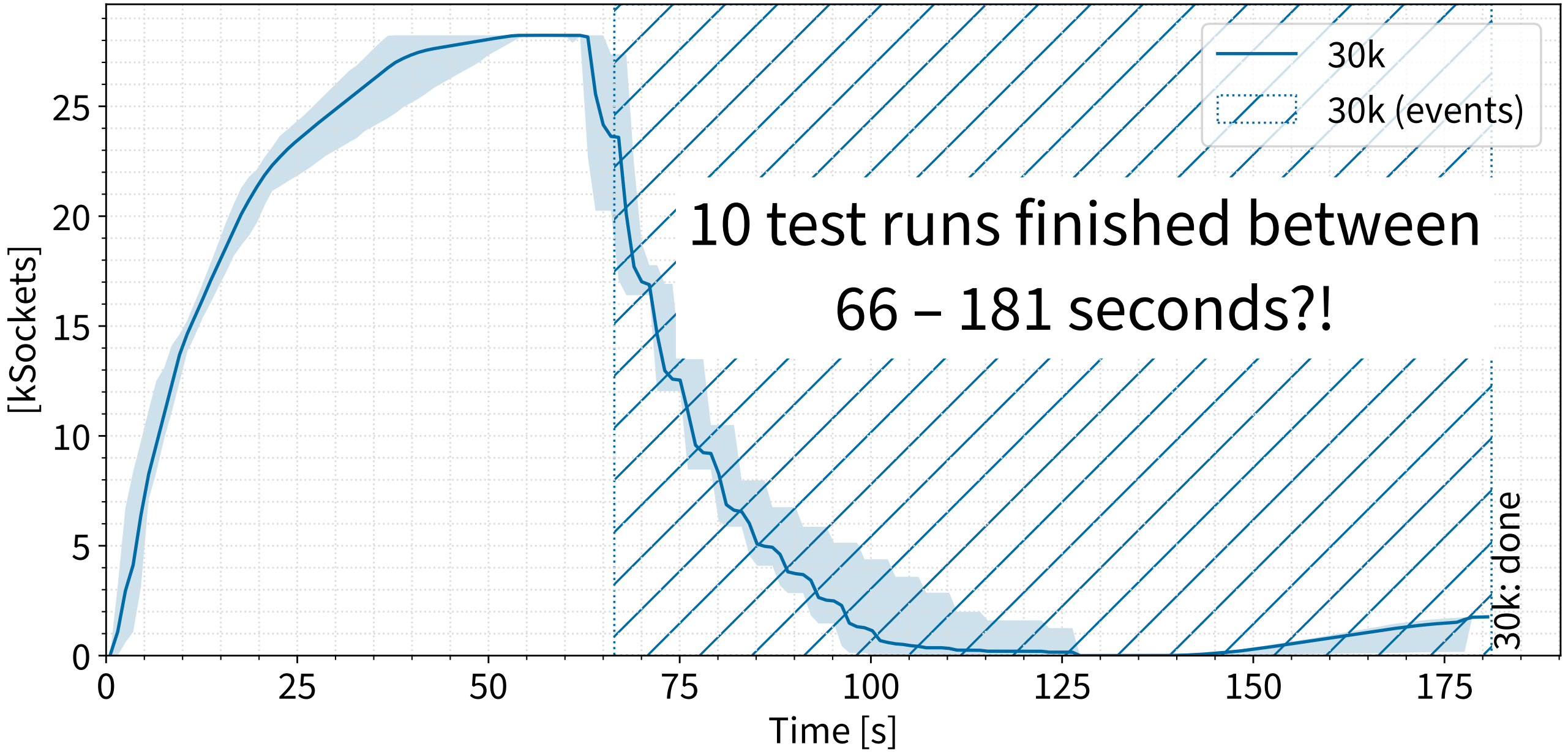
30k zones: CPU load



30k small zones: secondary log

- ~ 38 seconds after start
  - **`failed to connect: address not available`**
    - … for the first time

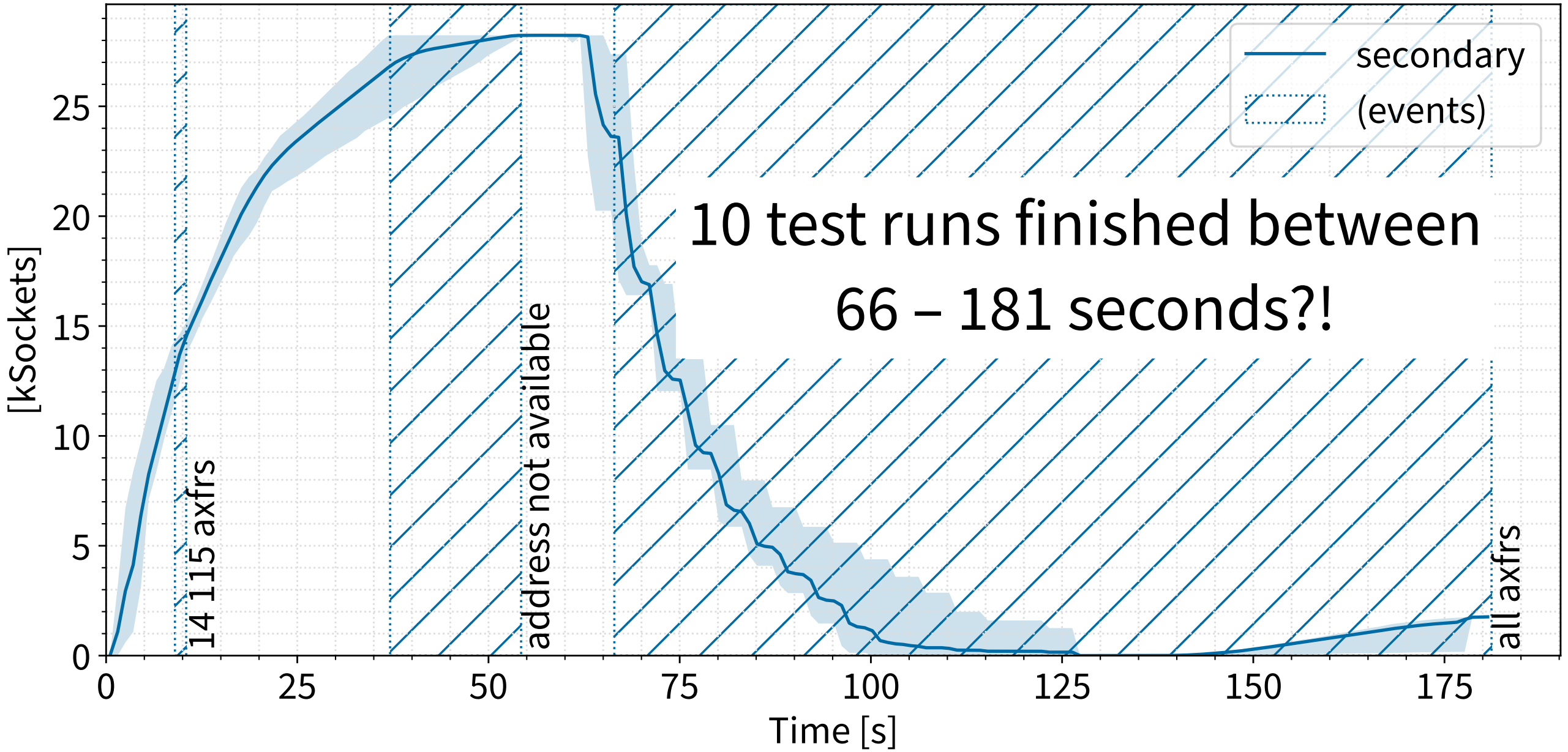
30k zones: TCP TIME_WAIT sockets



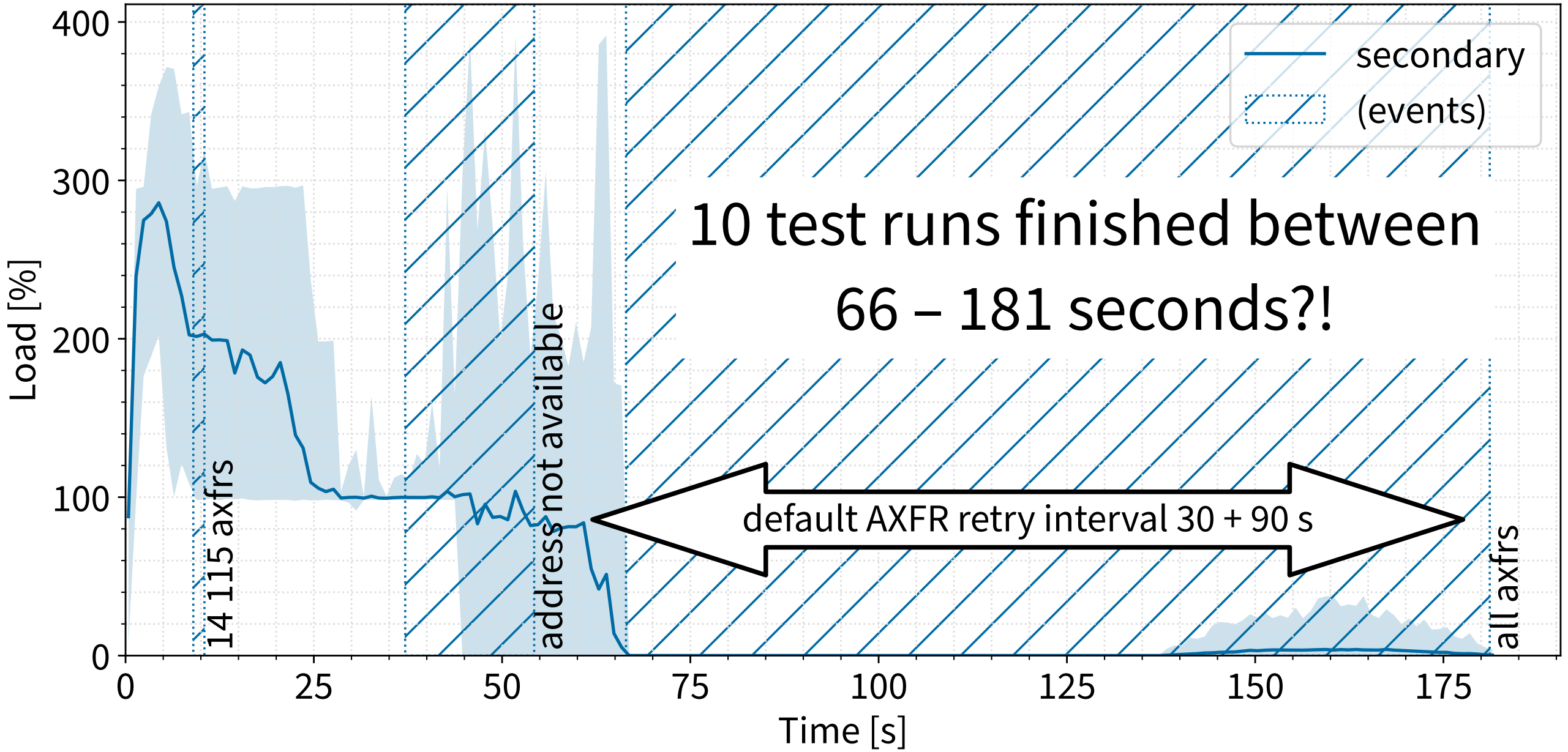
TCP state machine strikes back

- connection: (src IP, **src port**, dest IP, dest port)
- Ephemeral port range
  - `$ sysctl net.ipv4.ip_local_port_range`
  - `net.ipv4.ip_local_port_range = 32768 60999`
    - **28 231 ephemeral ports *by default***
- `$ ss -t -o state time-wait`
  - … `timer:(timewait,60sec,0)`
  - default wait 60 secs

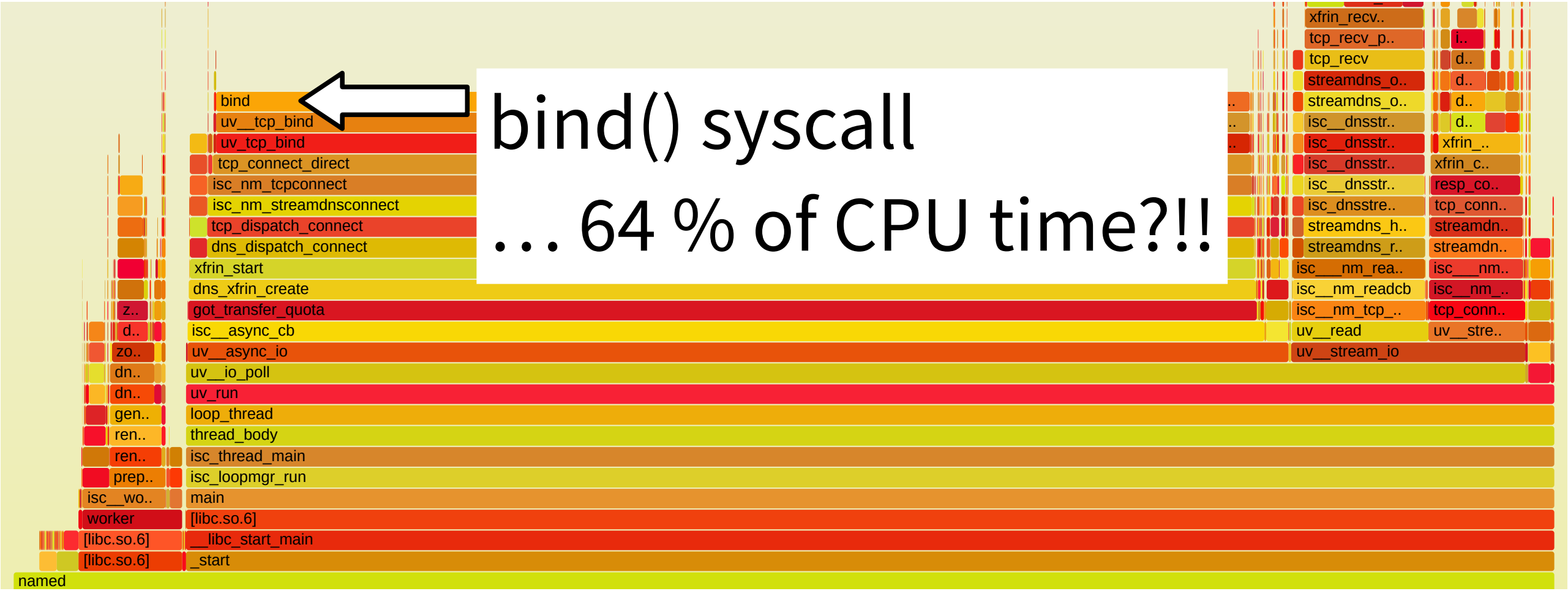
30k zones: TCP TIME_WAIT sockets



30k zones: CPU load



30k small zones: CPU profile



bind() syscall

… 64 % of CPU time?!!

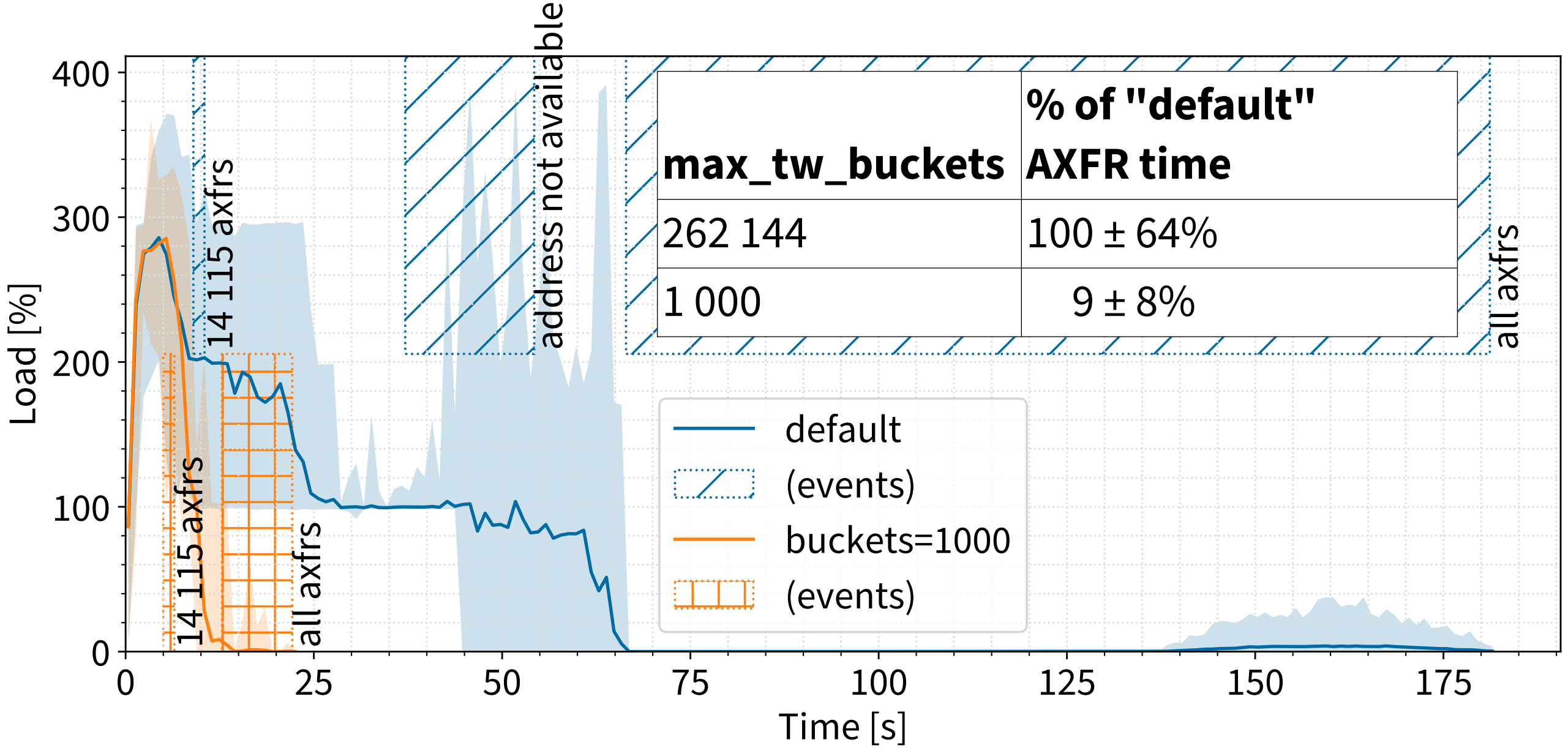
Linux TCP stack strikes back

- Linux Plumbers Conference 2023: **connect() – why you so slow?!**
  - https://lpc.events/event/17/contributions/1593/
  - https://youtu.be/J5Hm6PrJWI4?t=19000

Linux sysctl tcp_max_tw_buckets

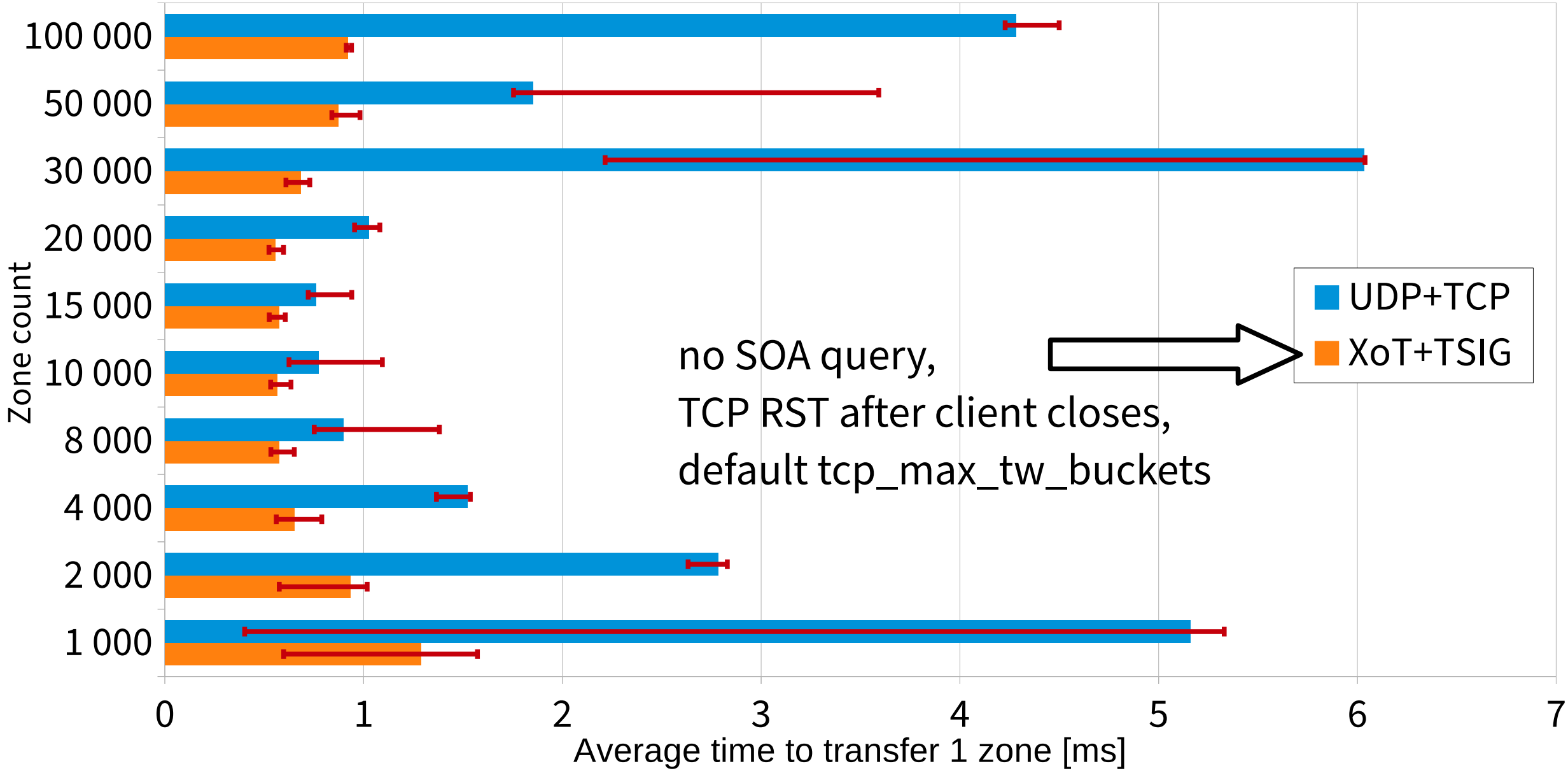
- tcp_max_tw_buckets - INTEGER
- Maximal number of timewait sockets held by system simultaneously.
- **If this number is exceeded time-wait socket is immediately destroyed** and warning is printed. This limit exists only to prevent simple DoS attacks,
- **you _must_ not lower the limit artificially**,
- but rather increase it (probably, after increasing installed memory), if network conditions require more than default value.
- ☠ `$ sysctl -w net.ipv4.tcp_max_tw_buckets=1000` ☠

CPU load, 30 k zones, tcp_max_tw_buckets



max_tw_buckets	% of "default" AXFR time
262 144	100 ± 64%
1 000	9 ± 8%

Time to transfer: TCP vs. XoT



Takeaways

- Test environment validation is a **MUST** 
- Lots of zones => lots of TCP => tweaks
  - Connection reuse?
  - Protocol tweaks?
- XFR over TLS can be *fast*
  - … with couple hacks
- **Non-linearity everywhere**

Thank you!

- Main website: https://www.isc.org
- Presentations: https://www.isc.org/presentations
- Main GitLab: https://gitlab.isc.org