Challenges of time sync in Datacenters

Maciek Machnikowski (NVIDIA)
Oleg Obleukhov (Meta)
Vadim Fedorenko (Meta)
Wojciech Waśko (NVIDIA)

Agenda

- PTP in datacenters
- Multi-NIC setups
- Client-server IEEE 1588
- Window of uncertainty
- PTP clocks and the userland

Time is the fourth dimension of the data center

- **uniformizes** distributed environments
    - profiling
    - high-frequency telemetry
- **accelerates** workloads
    - distributed databases (Google Spanner)
    - k-v stores (Cassandra)
- **connects** the compute to the physical world
    - Telco - 5G/6G virtual base station
    - Far edge - streaming data to/from sensors
    - Automotive - MIMO radars
    - ProViz - massive video walls

sphere uses timesync to display coherent imaging.
Behind the curtains, it's a small datacenter.

Photo: Michael Bittle via Sphere Facebook.

the better the accuracy,
the more usecases are unlocked.

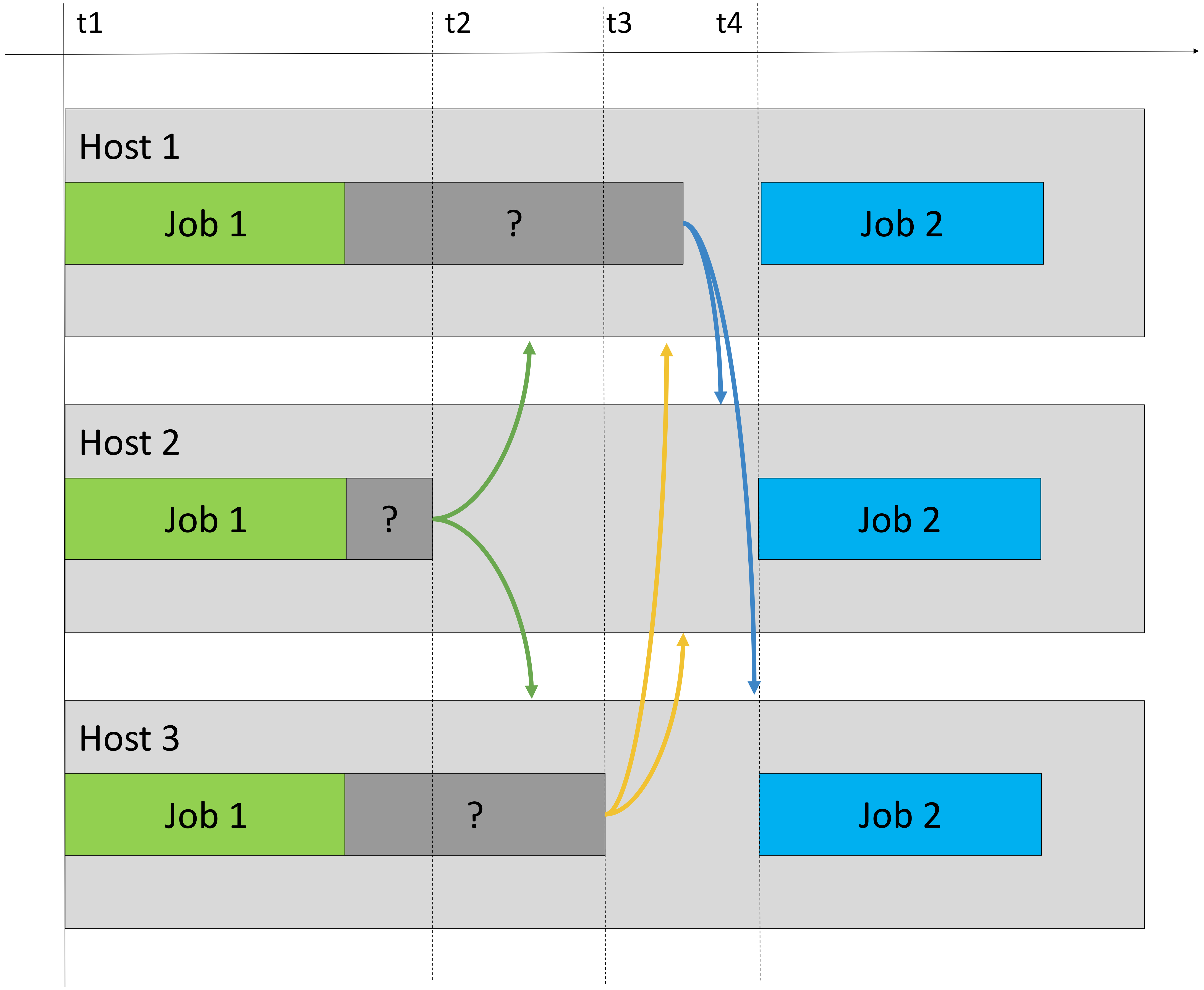
PTP in datacenters

- Significantly larger scale
  - tens to hundreds of thousands of nodes
- More unpredictable failures
- Custom software stacks and distributions
- Heterogeneous environments
- Synchronized servers coexist with unsynchronized

Event tracing in AI clusters

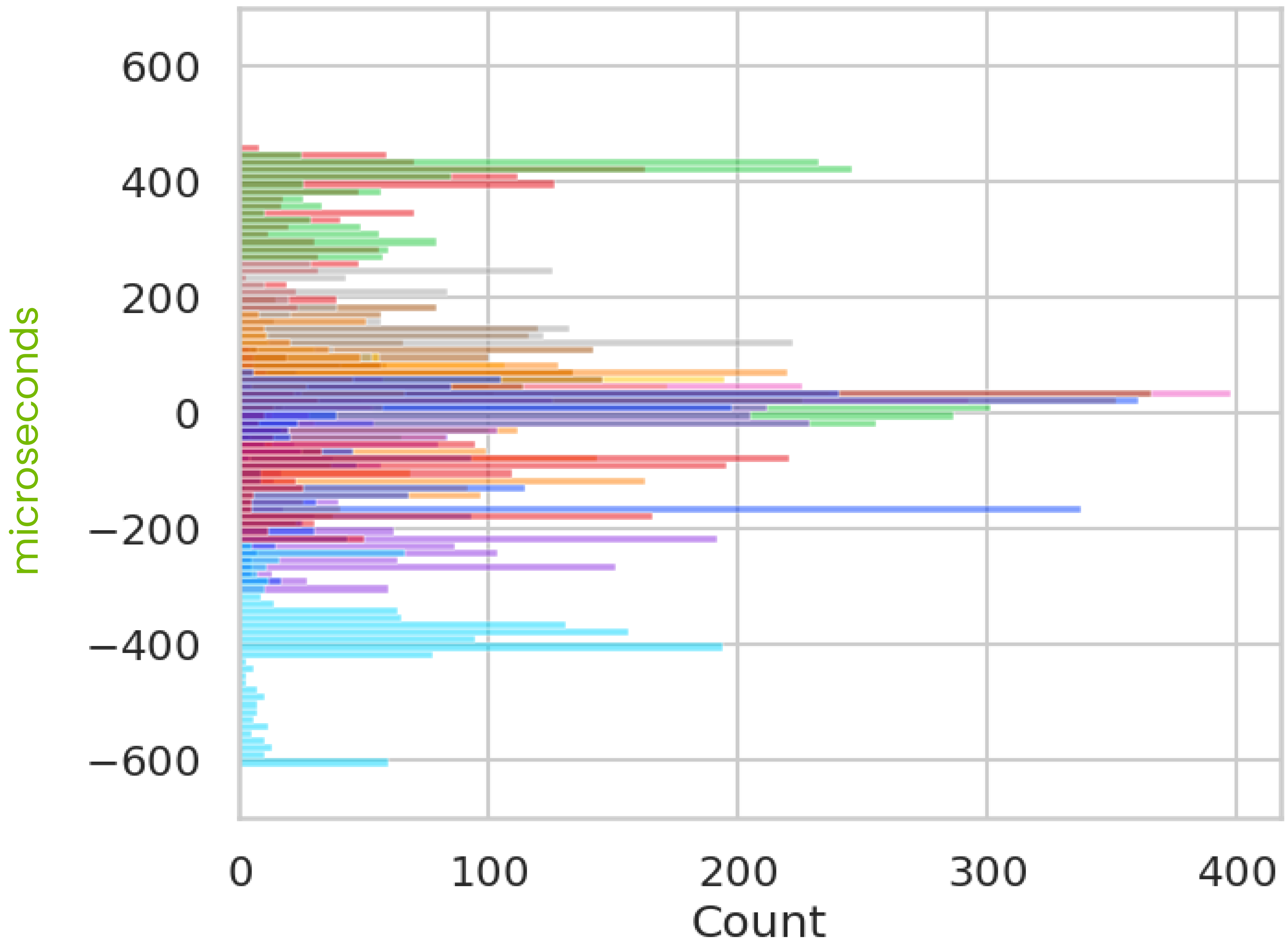
Tracing job start

- Unknown sources of job duration discrepancies
  - Network?
  - GPU?
  - Clocks?
- Wasting the compute power of the "fastest" hosts.
- NTP lacks accuracy

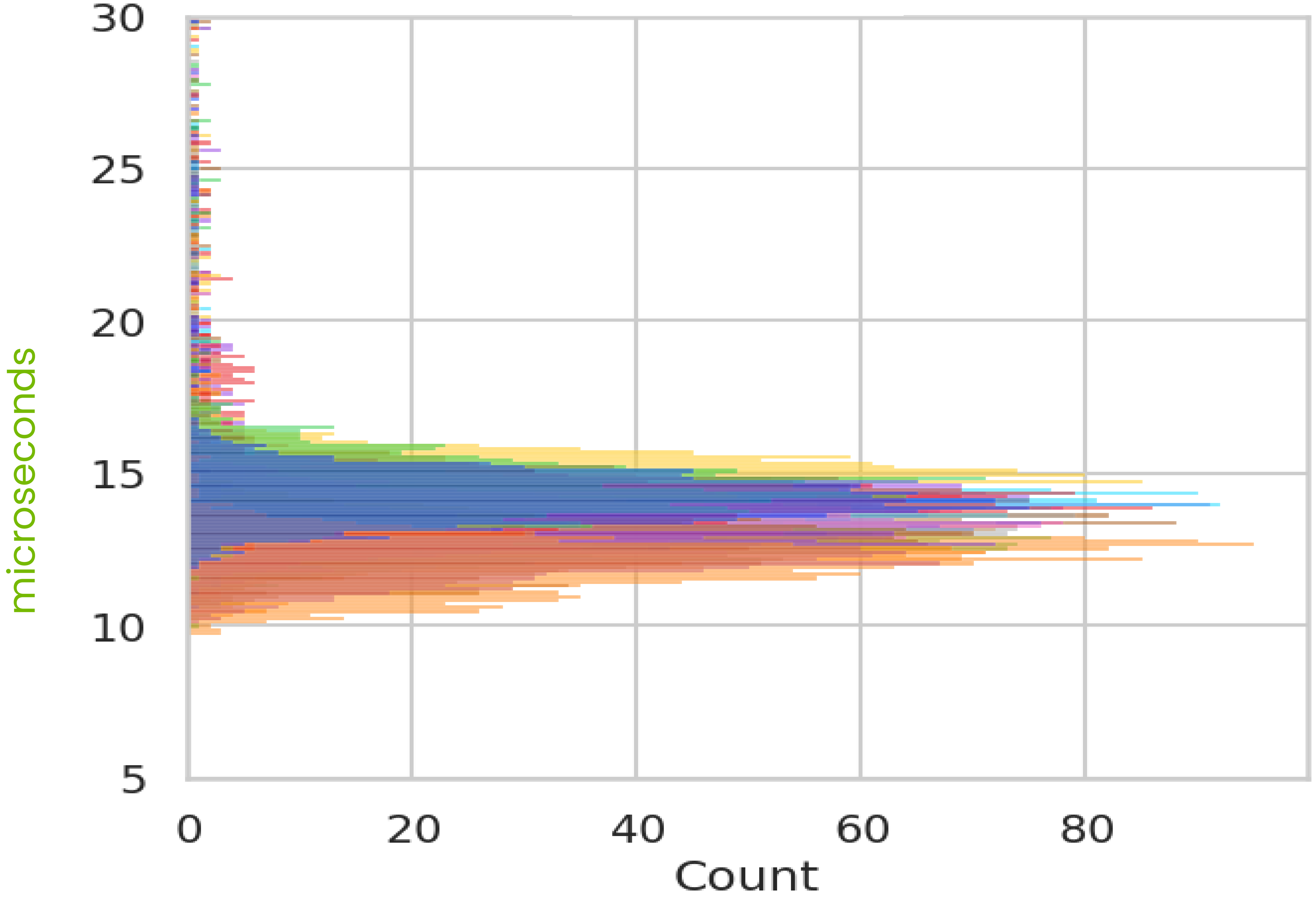


Distribution of job start time of NTP –synchronized tasks

Distribution of job start time of PTP –synchronized tasks



Distribution of latency measurements using NTP time



Distribution of latency measurements using PTP time

Multiple NIC challenges

NIC Challenges

- Datacenter endpoints might feature several NICs.
- Alternatively, a single NIC could accommodate multiple hosts or clients.

Multi-NIC setups

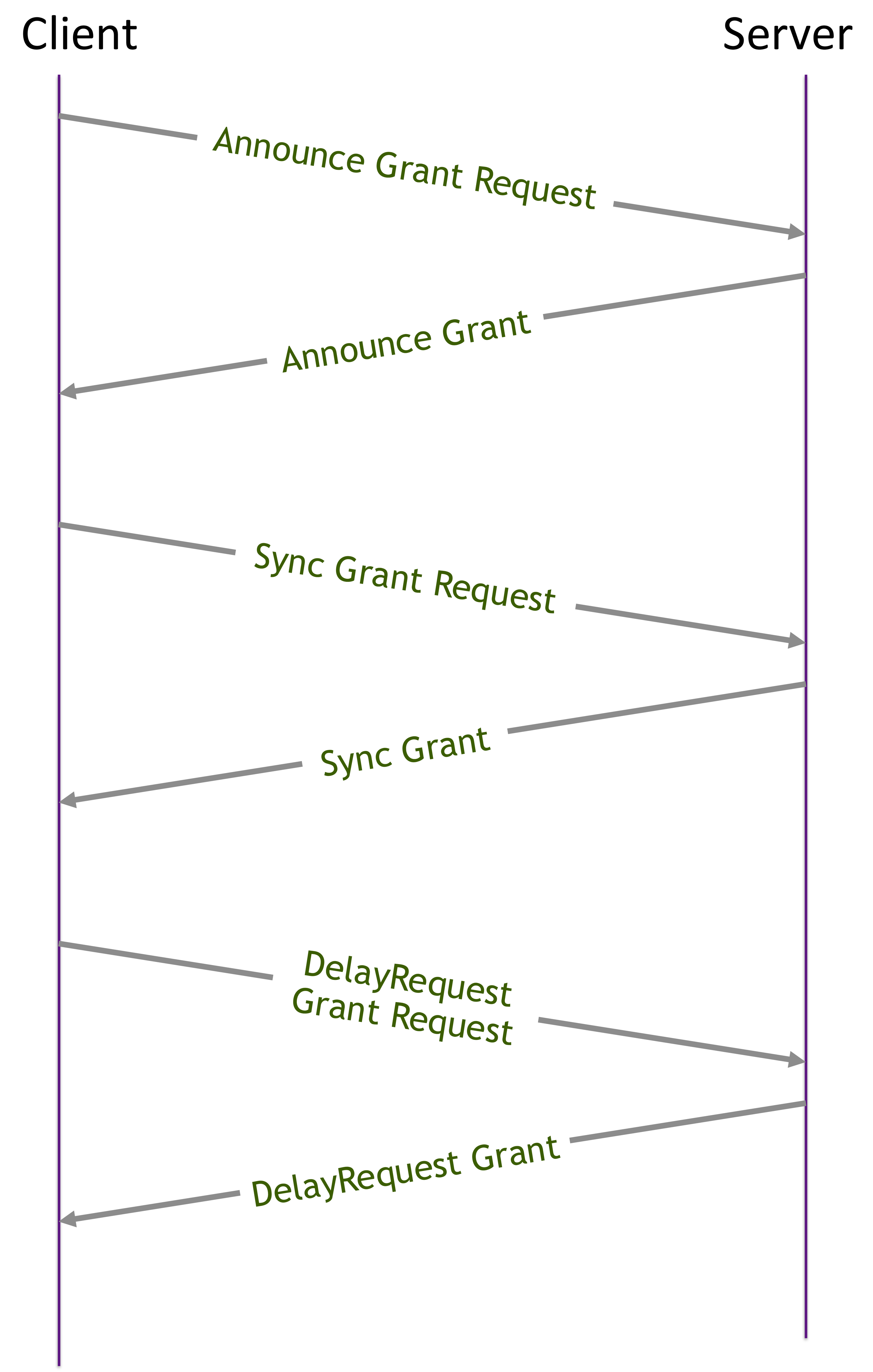
- A host must agree on the best time source.
- Current solutions do not fulfill the requirements
- Chrony can read and monitor multiple PHCs,
- but it does not connect with the PTP stack to determine its state and quality

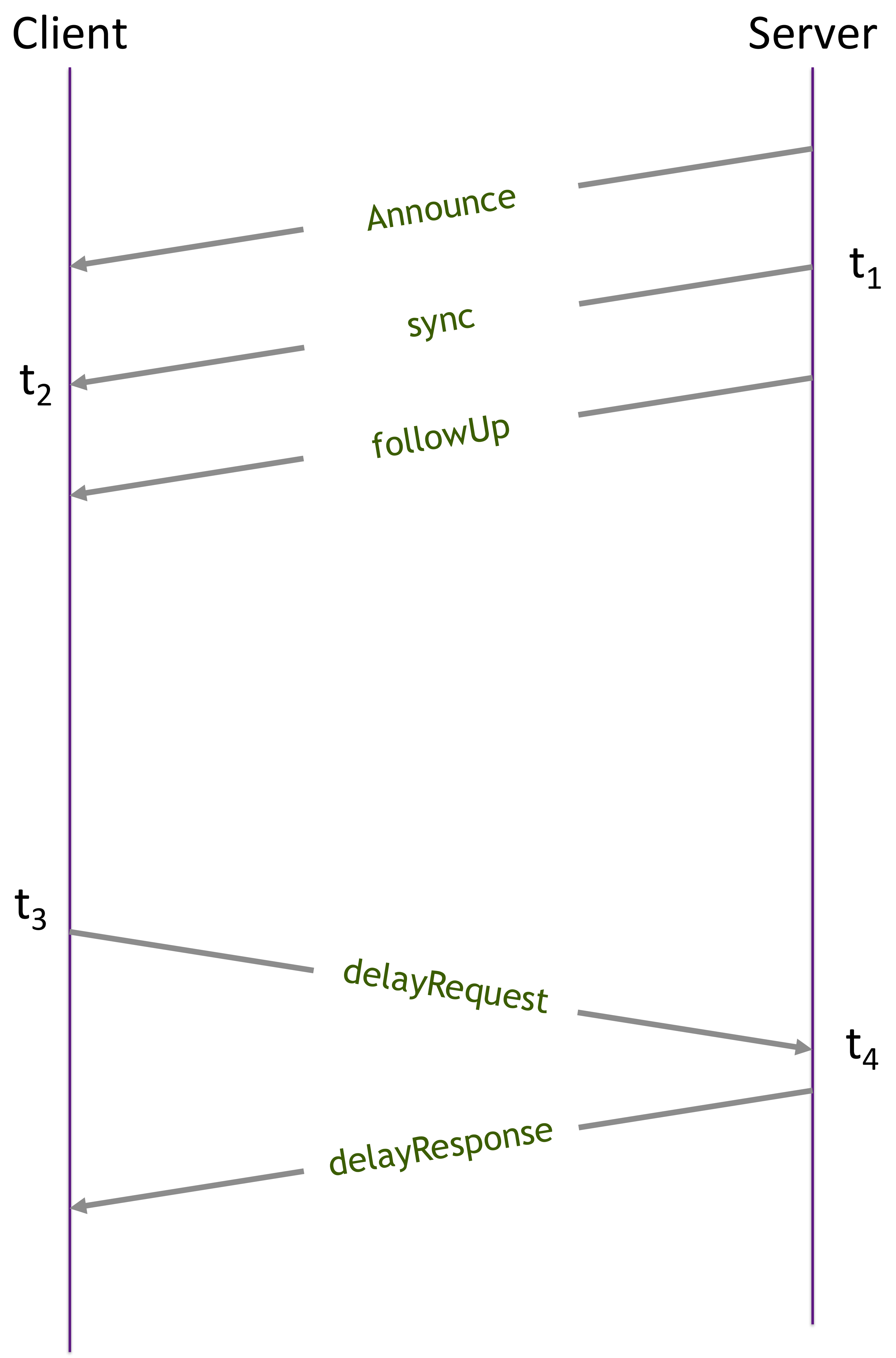
Single NIC may serve multiple hosts

- A single NIC may serve multiple:
  - Containers
  - VMs
  - Or even Hosts
- Only one entity can steer the clock
- But all need information about the sync state

Client-server 1588

PTP Unicast messaging





Client-server IEEE 1588

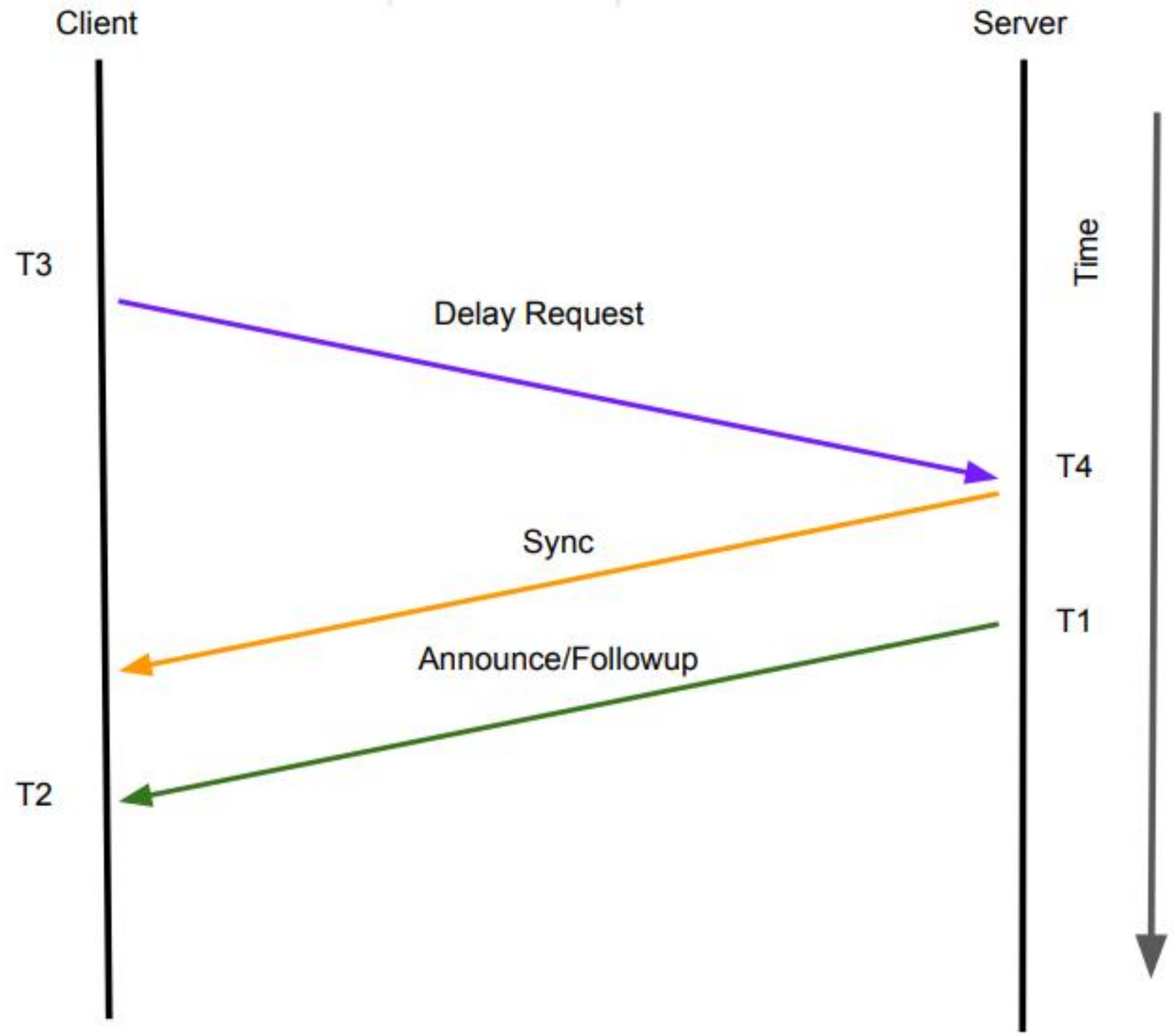
- Eliminates the need for
  - unicast negotiation
  - maintaining the per-client state
- Reduce network bandwidth
- Server responds to each request

Client-server IEEE 1588

- At least three projects implement this idea
  - SPTP
  - FlashPTP
  - NTP-over-PTP (currently in Chrony)

SPTP

- Client sends a Delay Request
- Server responds with a Sync
- Server sends FollowUp and Announce



Window of uncertainty

Window of uncertainty

- Applications must know not only the time
- But also the associated uncertainty of it
- Earliest-latest
- Kernel_timex:
  - Maxerror
  - Esterror

Different error sources

- Class of a GM
  - And its error
- Local oscillator drift
  - And holdover
- clock read delays
- Clock resolution
- Accumulated path elements delays
- Link asymmetry

Window of uncertainty

Synchronous workload task start relative time offsets
ntpd-synchronized cluster

PTP Hardware Clocks and ~~friends~~ userspace

Access permissions

- no permissions checks for POSIX dynamic clocks ioctls

  -> only root granted any access

- even read-only apps had to have write permissions

- fixed in net-next

How userland tells the time

and how long it can take, under system/PCIe load – very roughly

- TSC (x86), CNTPCT_EL0 (arm), … — < 10 nsec
  - arch-dependent CPU cycle counter
  - free-running; starts at 0 on boot
- system clock (vDSO) — < 30 nsec
  - kernel exports page with GTOD (Generic Time Of Day) data
  - userspace reconstructs clock (_REALTIME, _MONOTONIC, …) with GTOD data and CPU counter
- system clock (syscall) — < 200 nsec
  - userspace calls kernel
  - kernel calculates and returns value (similar math to vDSO)
- device clock (syscall) — < 30’000 nsec
  - userspace calls kernel
  - device driver retrieves time from device

Let’s improve that

vDSO for dynamic clocks
(eliminating syscall)

- can save ~100-200 nsec
- much work
- smol reward

read clock without PCIe access
(??)

- can save 10s of microseconds
- ?? work
- much reward

Proof of Concept

Hermóðr

/dev/ptpX (syscall + PCIe) Diff between consecutive clock readings: Min: 1098 p01: 1388 p00.1 1380 Avg: 1612 p99: 3016 p99.9 28986 Max: 2119606 Avg call cost: 1612

How much faster?

```
/dev/ptpX (syscall + PCIe)

Diff between consecutive clock readings (ns):

  p0.01          1432
  p01:           1466
  Avg:           5334
  p99:           4252
  p99.9        403882
```

```
approx. /dev/ptpX

Diff between consecutive clock readings (ns):

  p0.01            36
  p01:             36
  Avg:             59
  p99:             61
  p99.9           174
```

kernel 6.13

load:

- 5x iperf3 bidir (external loopback between ports)
- stress-ng cpu (32) iomix (32) pci (32) vm (32) fork (32)

HW: HPE DL380 Gen11 + Intel Xeon Gold 6426Y + NVIDIA ConnectX-7 2x200G

How do we verify the quality?

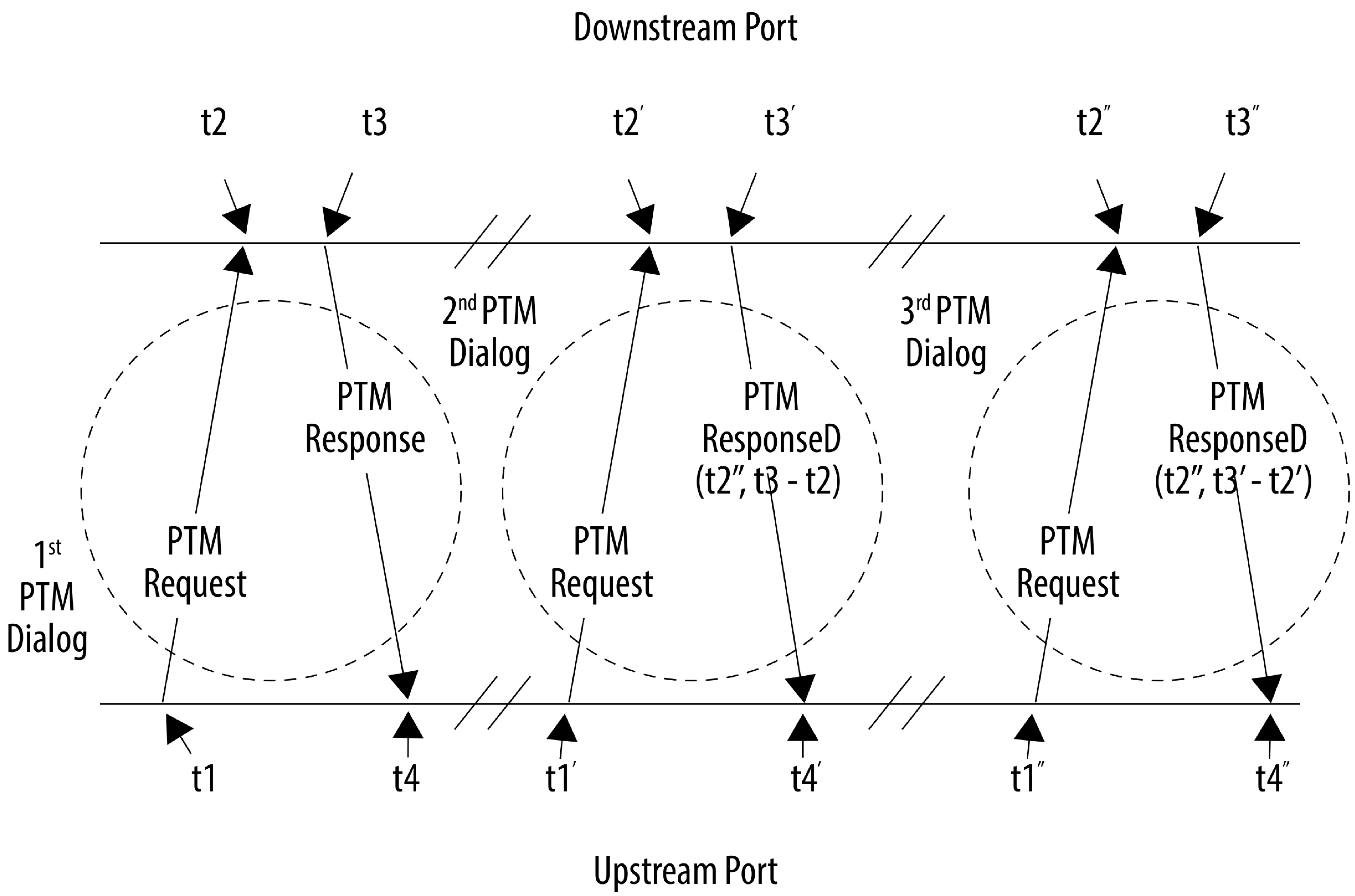
Enter… **PCIe PTM**

- **P**recise
- **T**ime
- **M**easurement

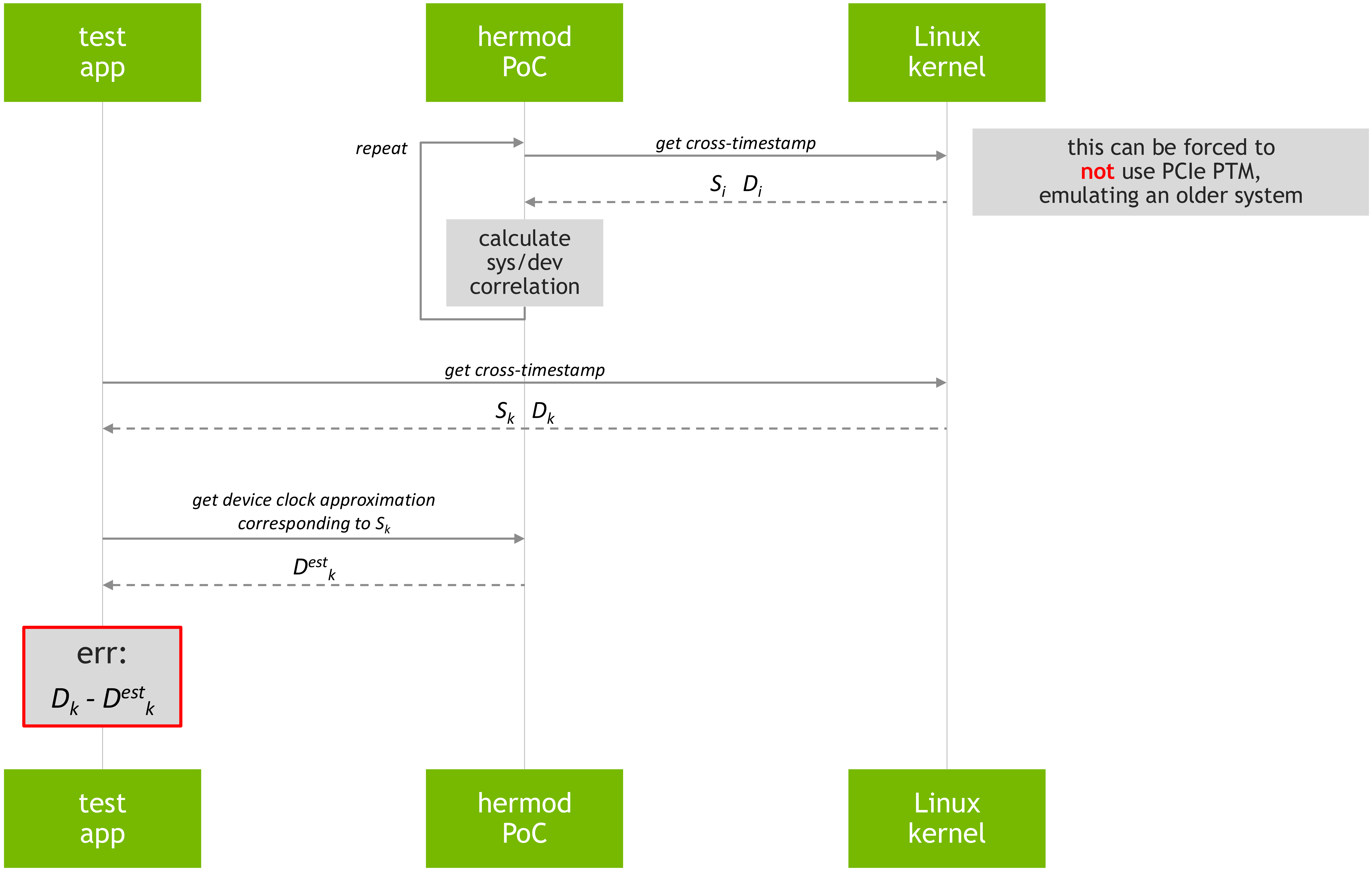
- PCIe link-local Message protocol
- timestamped in HW

- an atomic cross-timestamp of CPU and PCIe device clock counters
  - that’s an oversimplification, but good enough for our purposes

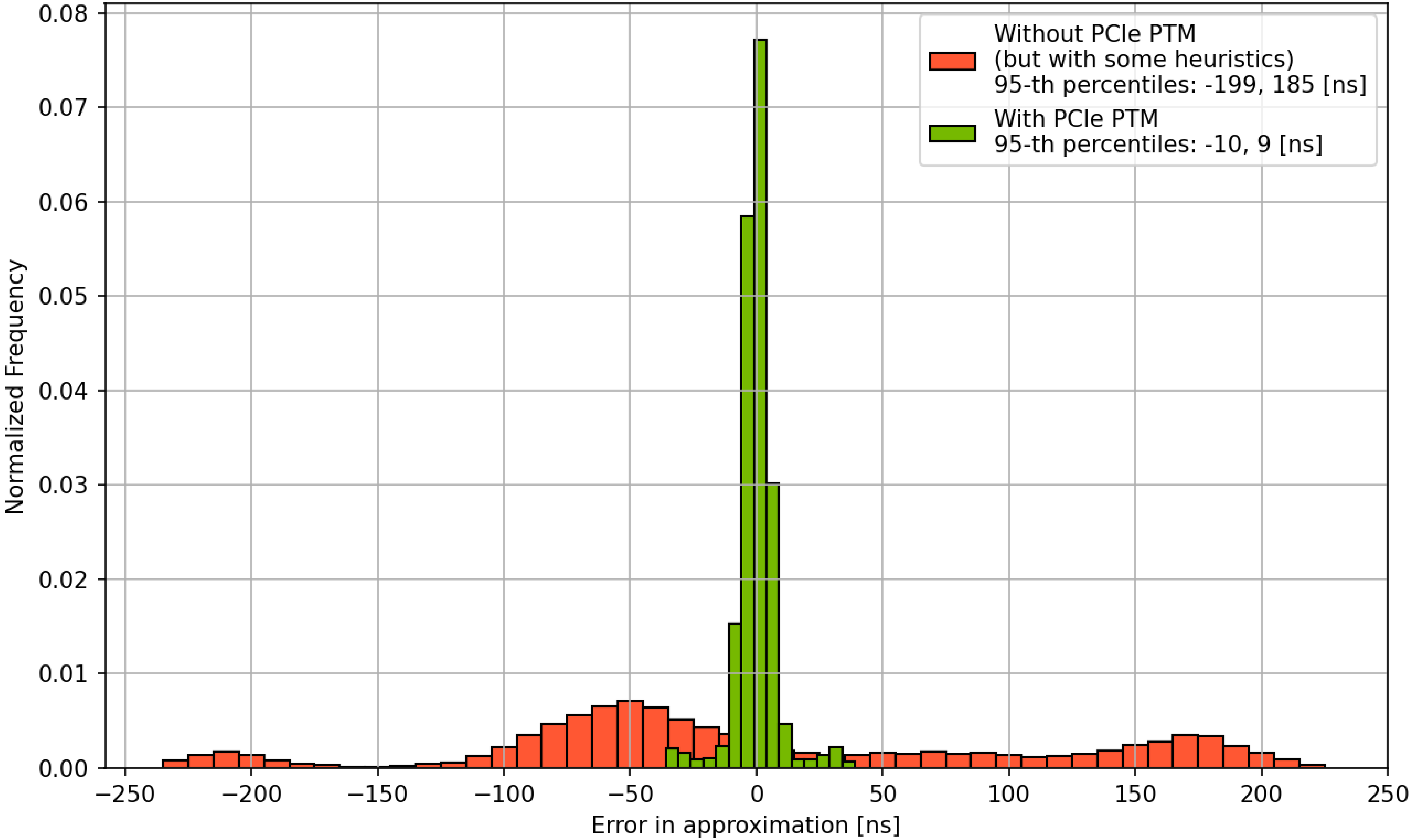
- this gives us **ground truth**



Verification via PCIe PTM



Approximation error histogram



kernel 6.13. Load: 5x iperf3 bidir (external loopback between ports) + stress-ng cpu (32) iomix (32) pci (32) vm (32) fork (32)

HW: HPE DL380 Gen11 + Intel Xeon Gold 6426Y + NVIDIA ConnectX-7 2x200G

Wrapping up…

What's next

- should the PHC approximation functionality be provided by the kernel?
  - an "approximation driver"
  - aliases existing PHC, provides approximated value in clock_gettime()

- now that apps rely on clocks more and more, how do they know what's happening to the clock?
  - not only window of uncertainty
  - did someone change the clock while I wasn't watching?
  - NETLINK messages for clock modification events

- API for time uncertainty
  - Use adjtimex esterror presented on netdev 0x18

NVIDIA®