SO_TIMESTAMPING

Netdev 0x17, 2023
Willem de Bruijn
MAINTAINERS: SOCKET TIMESTAMPING

Why Timestamping

- Network *Stack* Latency Debugging

Fathom: Understanding Datacenter Application Network Performance

Amin Vahdat, David Wetherall, Junhua Yan, Mubashir Adnan Qureshi, Neal Cardwell, Soheil Hassas Yeganeh, Van Jacobson, Willem de Bruijn, Yousuk Seung, Yuchung Cheng

Proceedings of ACM SIGCOMM 2023

- Fleetwide Continuous Monitoring

Dapper, a Large-Scale Distributed Systems Tracing Infrastructure

Benjamin H. Sigelman, Luiz André Barroso, Mike Burrows, Pat Stephenson, Manoj Plakal, Donald Beaver, Saul Jaspan, Chandan Shanbhag

Google, Inc. (2010)

Why Linux Kernel Timestamping

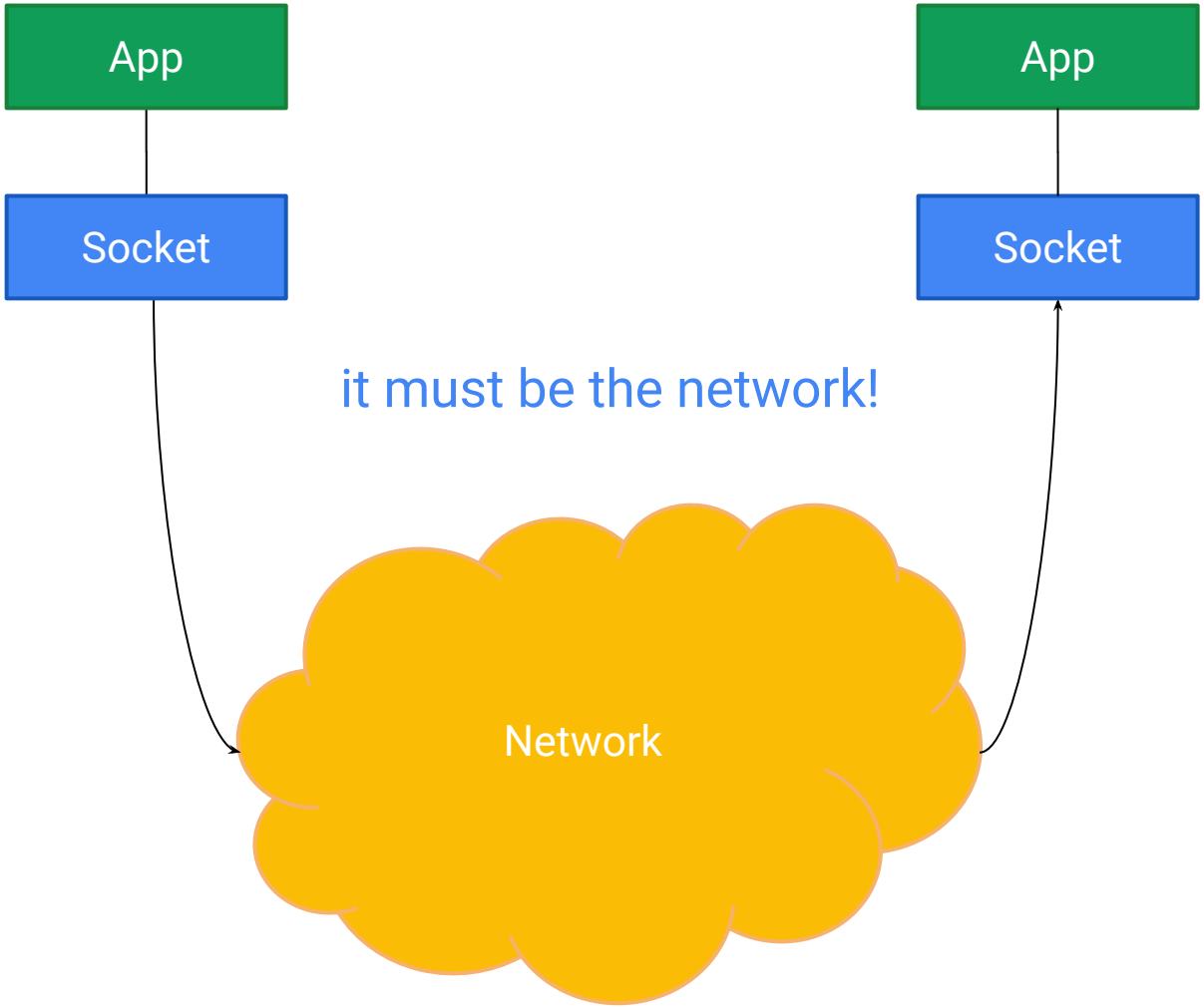
- Network Latency Debugging
- Fleetwide Continuous Monitoring
- Delay-based Congestion Control
- PTP Clock Synchronization
- Applications
  - Consistent Distributed Databases: CockroachDB
  - Financial Markets

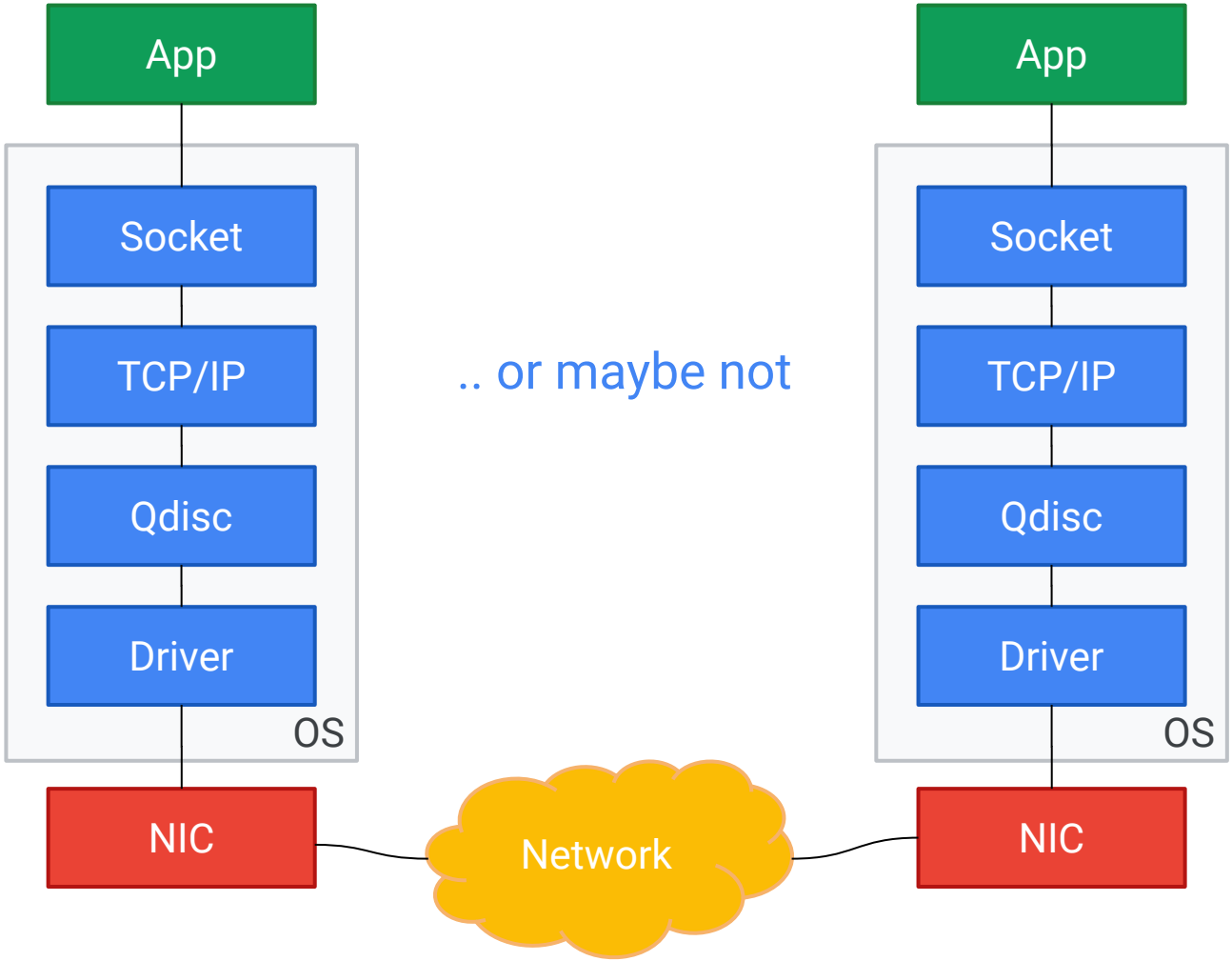
Network Stack Latency Debugging

In the beginning

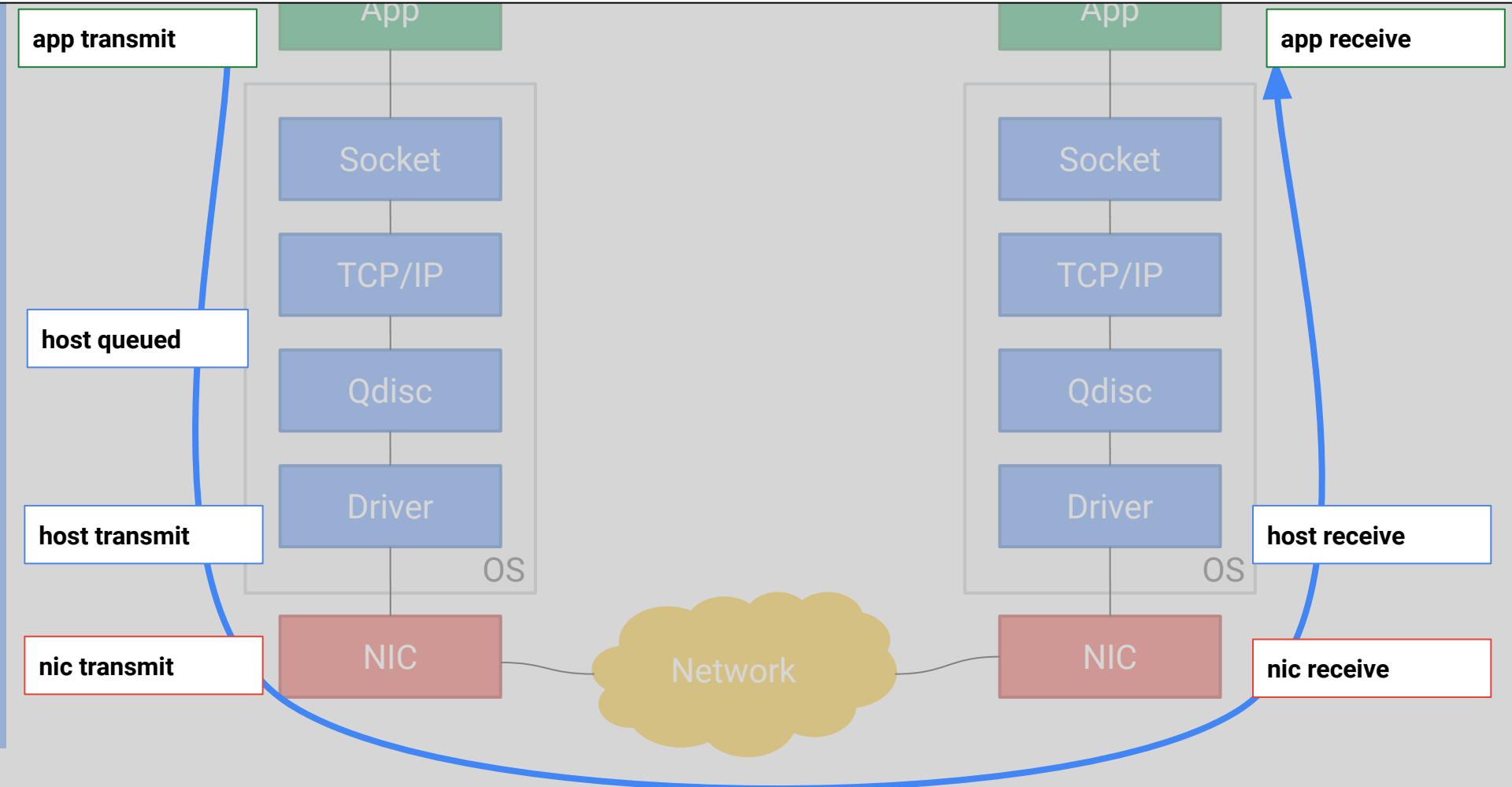
My RPC is slow..

```
clock_gettime(.., &ts_pre);
send(fd, &request, sizeof(request), 0);
recv(fd, &reply, sizef(reply), 0);
clock_gettime(.., &ts_post);
```


App
App
Socket
Socket
it must be the network!
Network


App
Socket
TCP/IP
Qdisc
Driver
OS
NIC
.. or maybe not
Network
App
Socket
TCP/IP
Qdisc
Driver
OS
NIC

Extend measurement to kernel and NIC



SO_TIMESTAMPING Basics

Linux Timestamp APIs

www.kernel.org/doc/Documentation/networking/timestamping.rst

SO_TIMESTAMP: software receive timestamp, usec

SO_TIMESTAMPNS: like SO_TIMESTAMP, but nsec

SO_TIMESTAMPING: configurable timestamps, nsec

Request timestamps with SO_TIMESTAMP

```
const int one = 1;

setsockopt(fd, SOL_SOCKET, SO_TIMESTAMP, &one, sizeof(one));
```

Request timestamps with SO_TIMESTAMPING

```
const int val = SOF_TIMESTAMPING_RX_SOFTWARE |
                SOF_TIMESTAMPING_SOFTWARE;

setsockopt(fd, SOL_SOCKET, SO_TIMESTAMPING, &val, sizeof(val));
```

Receive timestamps

```
recvmsg(fd, &msg, 0);

for (cm = CMSG_FIRSTHDR(msg); cm;
     cm = CMSG_NXTHDR(msg, cmsg))

        if (cm->cmsg_level == SOL_SOCKET &&
            cm->cmsg_type == SCM_TIMESTAMPING)

                struct scm_timestamping *t = (void *) CMSG_DATA(cm);
                printf("sw: %ld.%ld\n", t->ts[0].tv_sec, t->ts[0].tv_nsec);
```

`tools/testing/selftests/net/{[rt]xtimestamp,timestamping}.c`

Receive timestamps: demultiplex fields

```
struct scm_timestamping {
        struct timespec ts[3];
};
```

```
ts[0]: SOF_TIMESTAMPING_SOFTWARE
ts[1]: deprecated
ts[2]: SOF_TIMESTAMPING_RAW_HARDWARE
```

Receive transmit timestamps

```
send(fd, ..);
poll(..);
recvmsg(fd, &msg, MSG_ERRQUEUE);

for (cm = CMSG_FIRSTHDR(msg); cm;
     cm = CMSG_NXTHDR(msg, cmsg))

        if (cm->cmsg_level == SOL_SOCKET &&
            cm->cmsg_type == SCM_TIMESTAMPING)

                struct scm_timestamping *t = (void *) CMSG_DATA(cm);
                printf("sw: %ld.%ld\n", t->ts[0].tv_sec, t->ts[0].tv_nsec);
```

Demultiplex multiple tx timestamp

```
if (cm->cmsg_level == SOL_IP && cm->cmsg_type == IP_RECVERR) {
    struct sock_extended_err *serr = (void *) CMSG_DATA(cm);

    assert(serr->ee_origin == SO_EE_ORIGIN_TIMESTAMPING);
    switch(serr->ee_info) {
        case SCM_TSTAMP_SCHED: type = "qdisc"; break;
        case SCM_TSTAMP_SND: type = "sent"; break;
        case SCM_TSTAMP_ACK: type = "acked"; break;
    };
    printf("tx timestamp: type=%s offset=%u\n", type, serr->ee_data);
}
```

Select timestamps to generate

SOF_TIMESTAMPING_RX_HARDWARE

SOF_TIMESTAMPING_TX_HARDWARE

SOF_TIMESTAMPING_RX_SOFTWARE

SOF_TIMESTAMPING_TX_SOFTWARE

SOF_TIMESTAMPING_TX_SCHED

SOF_TIMESTAMPING_TX_ACK

Select timestamps to report

SOF_TIMESTAMPING_SOFTWARE

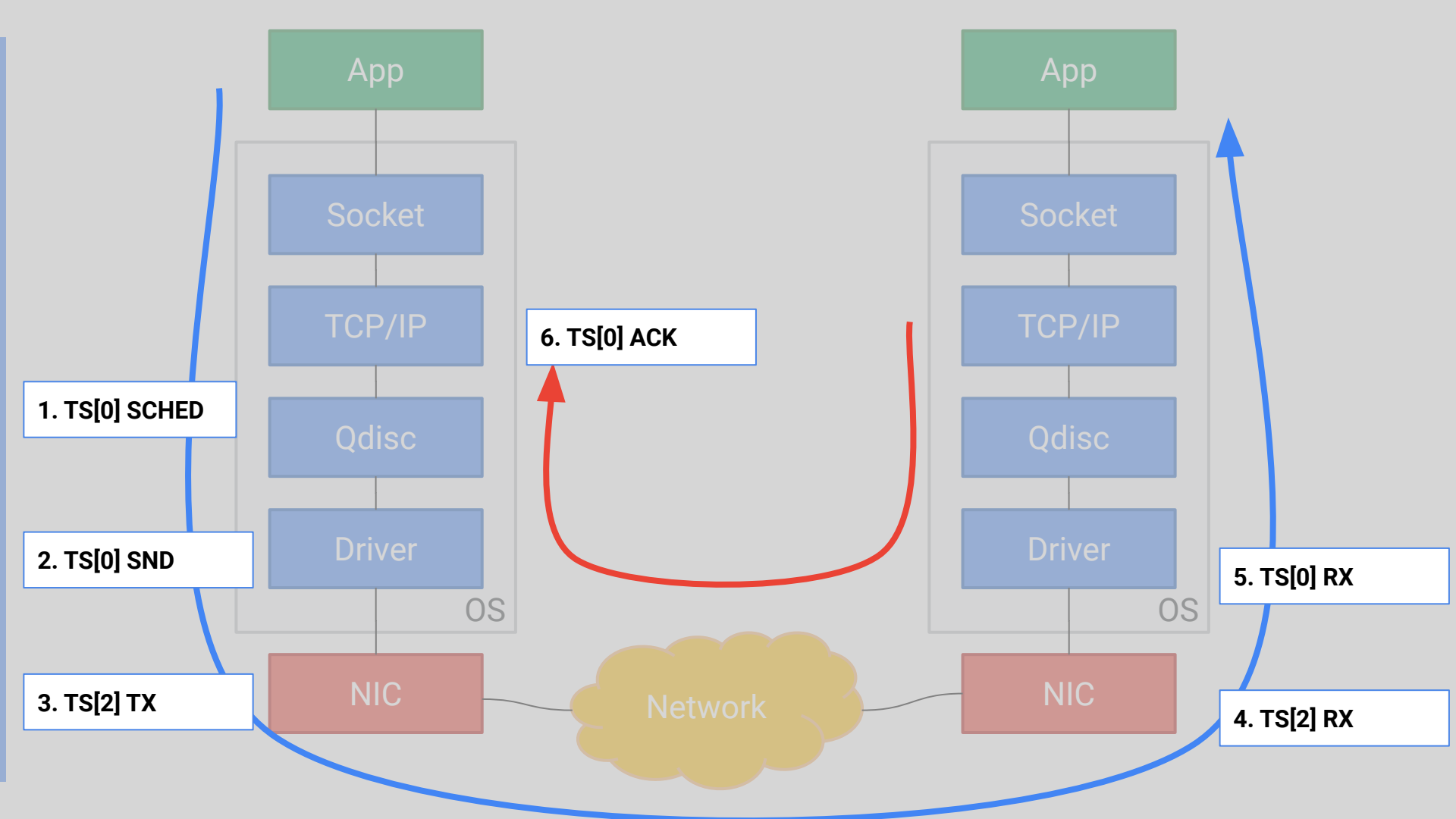
SOF_TIMESTAMPING_SYS_HARDWARE

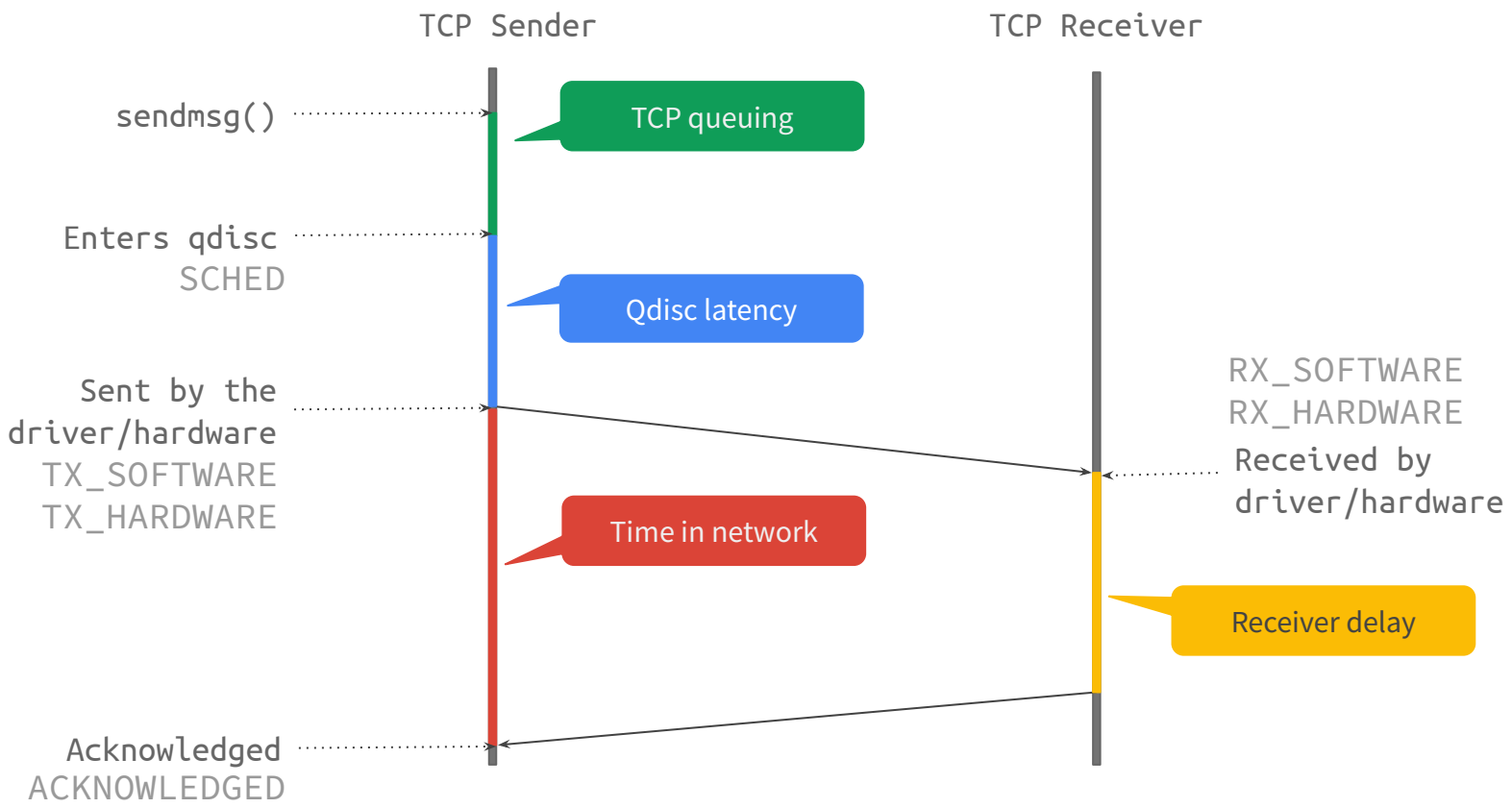
SOF_TIMESTAMPING_RAW_HARDWARE

Select timestamps to generate & report

```
const int val = SOF_TIMESTAMPING_RX_SOFTWARE |
                SOF_TIMESTAMPING_SOFTWARE;

setsockopt(fd, SOL_SOCKET, SO_TIMESTAMPING, &val, sizeof(val));
```


App
Socket
TCP/IP
Qdisc
Driver
OS
NIC
App
Socket
TCP/IP
Qdisc
Driver
OS
NIC
Network
1. TS[0] SCHED
2. TS[0] SND
3. TS[2] TX
6. TS[0] ACK
5. TS[0] RX
4. TS[2] RX


TCP Sender
TCP Receiver
sendmsg()
TCP queuing
Enters qdisc
SCHED
Qdisc latency
Sent by the
driver/hardware
TX_SOFTWARE
TX_HARDWARE
Time in network
RX_SOFTWARE
RX_HARDWARE
Received by
driver/hardware
Receiver delay
Acknowledged
ACKNOWLEDGED

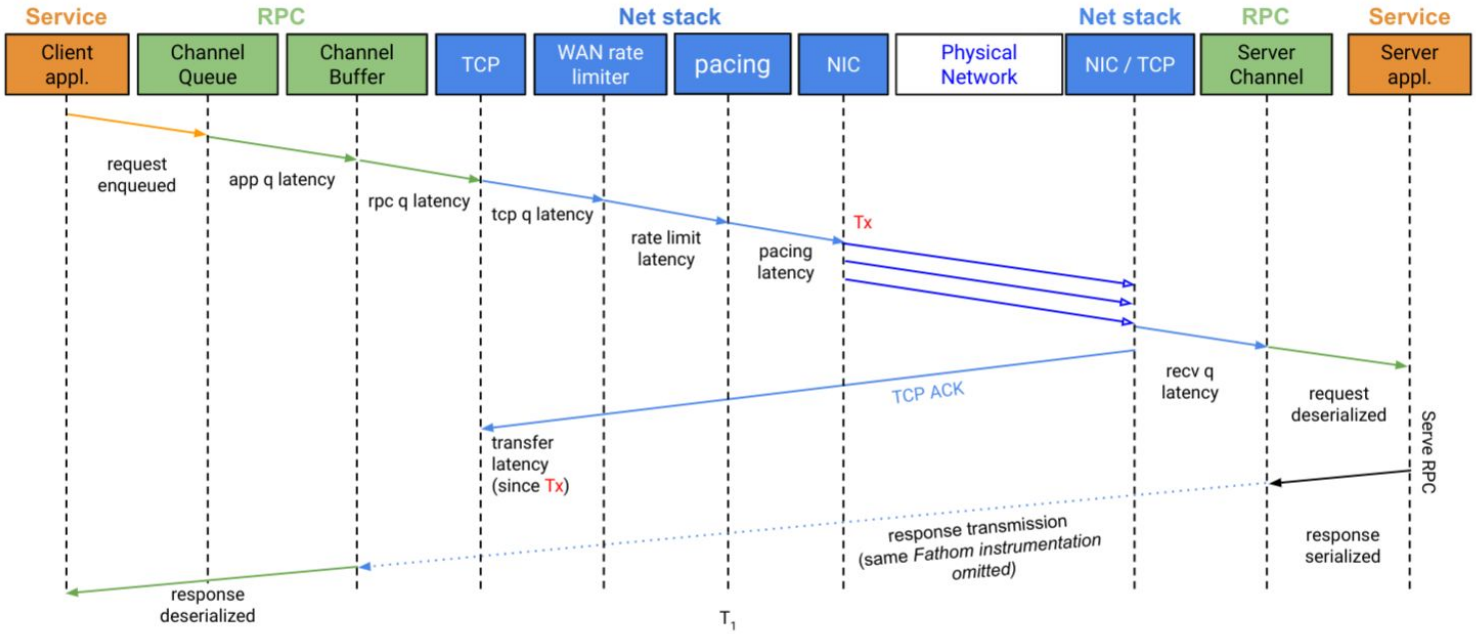


Figure 4: RPC latency breakdown. The latency metrics in the network stack (blue) are instrumented by Fathom.

SO_TIMESTAMPING Extensions

Goal 1: Support TCP

`sendmsg()`: convert from discrete RPC messages to TCP bytestream

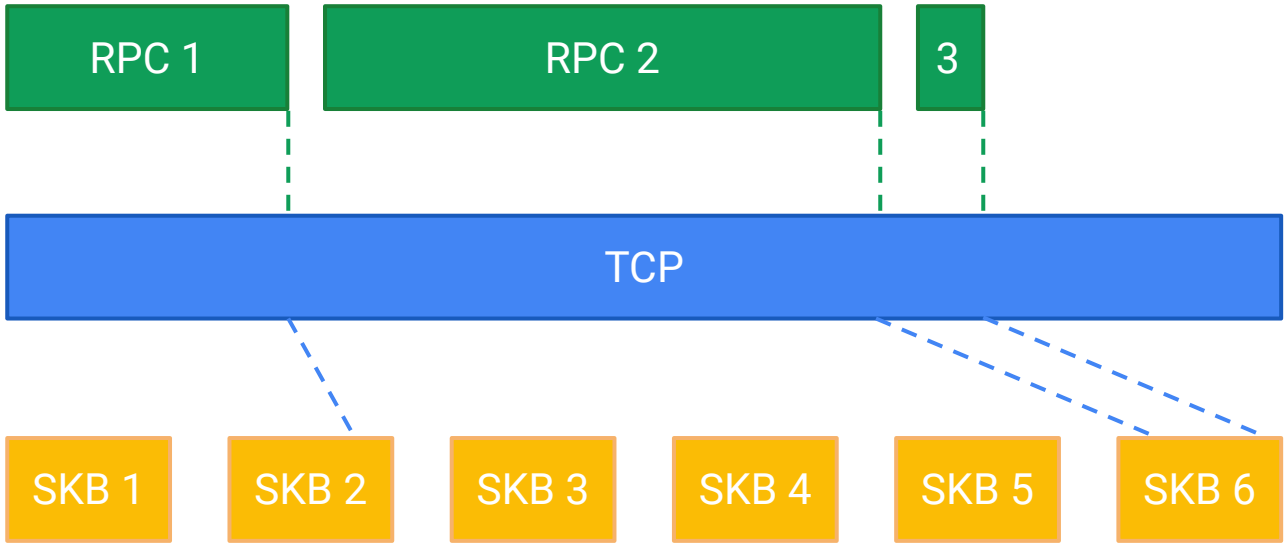
Convert from bytestream to discrete skbs: TSO, retransmits, mss probing, ..

Return timestamp for the last byte passed to `send()`

MSG_EOR

SO_RCVLOWAT

Goal 1: Support TCP



MSG_EOR: End of Record

SO_RCVLOWAT: SOF_TIMESTAMPING_OPT_POLLIN

Goal 2: Scale to continuous monitoring

SOF_TIMESTAMPING_TSONLY

do not return skb data

SOF_TIMESTAMPING_OPT_ID_TCP

`serr->ee_data` returns offset from setsockopt SO_TIMESTAMPING

`sendmsg()` cmsg SCM_TIMESTAMPING

selective sampling: override tx generation selection

Goal 3: Attribute To Root Causes

SOF_TIMESTAMPING_OPT_STATS: correlate with TCP connection state

```
struct sk_buff *tcp_get_timestamping_opt_stats(const struct sock *sk,
                                               const struct sk_buff *orig_skb,
                                               const struct sk_buff *ack_skb)
{
        const struct tcp_sock *tp = tcp_sk(sk);
        struct sk_buff *stats;
        struct tcp_info info;
        unsigned long rate;
        u64 rate64;

        stats = alloc_skb(tcp_opt_stats_get_size(), GFP_ATOMIC);
        if (!stats)
                return NULL;

        tcp_get_info_chrono_stats(tp, &info);
        nla_put_u64_64bit(stats, TCP_NLA_BUSY,
                          info.tcpi_busy_time, TCP_NLA_PAD);
        nla_put_u64_64bit(stats, TCP_NLA_RWND_LIMITED,
                          info.tcpi_rwnd_limited, TCP_NLA_PAD);
        nla_put_u64_64bit(stats, TCP_NLA_SNDBUF_LIMITED,
```

Analysis

Some Surprising Results

- Source vs Cause
  - source might be sender transmit stack
  - but cause might be receiver rwin
- Scheduling Latency
  - sk_data_ready: thread wake-up
- Incast: Host Congestion

Understanding Host Interconnect Congestion

Amin Vahdat, Behnam Montazeri, David E Culler, Gautam Kumar, Khaled Elmeleegy, Luigi Rizzo, Marc Asher de Kruijf, Masoud Moshref, Rachit Agarwal, Saksham Agarwal, Sylvia Ratnasamy

Association for Computing Machinery, New York, NY, USA (2022), 198–204

Beyond SO_TIMESTAMPING

Alternative: tracepoints + kprobes + uprobes

Open Questions

- how to correlate a measurement with { packet, seqno, rpc }
- how much do measurements perturb the measured system?
- how practical if it requires superuser privileges

1. Debugging and Monitoring
2. SO_TIMESTAMPING API
3. Adding TCP, sampling and correlation with state

Questions

More Info:

- Documentation/networking/timestamping.rst
- SIGCOMM 2023: Fathom
- netdev 0x13: TCP SO_TIMESTAMPING with OPT_STATS ...

Backup

Hardware Timestamps: Query capabilities

```
# ethtool -T eth0
Time stamping parameters for eth0:
Capabilities:
        hardware-transmit     (SOF_TIMESTAMPING_TX_HARDWARE)
        software-transmit     (SOF_TIMESTAMPING_TX_SOFTWARE)
        hardware-receive      (SOF_TIMESTAMPING_RX_HARDWARE)
        software-receive      (SOF_TIMESTAMPING_RX_SOFTWARE)
        software-system-clock (SOF_TIMESTAMPING_SOFTWARE)
        hardware-raw-clock    (SOF_TIMESTAMPING_RAW_HARDWARE)
PTP Hardware Clock: none
Hardware Transmit Timestamp Modes:
        off                   (HWTSTAMP_TX_OFF)
        on                    (HWTSTAMP_TX_ON)
Hardware Receive Filter Modes:
        none                  (HWTSTAMP_FILTER_NONE)
        all                   (HWTSTAMP_FILTER_ALL)
```

Hardware Timestamps: Configure

```
struct hwtstamp_config hwconfig = {
        .tx_type = HWTSTAMP_TX_ON,
        .rx_filter = HWTSTAMP_FILTER_PTP_V2_SYNC,
};

struct ifreq hwtstamp = {
        .ifr_name = "eth0",
        .ifr_data = &hwconfig,
};

ioctl(fd, SIOCSHWTSTAMP, &hwtstamp);
```