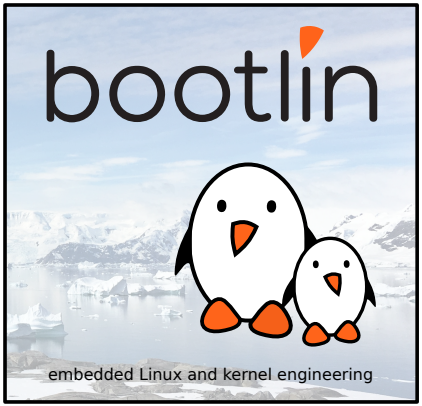
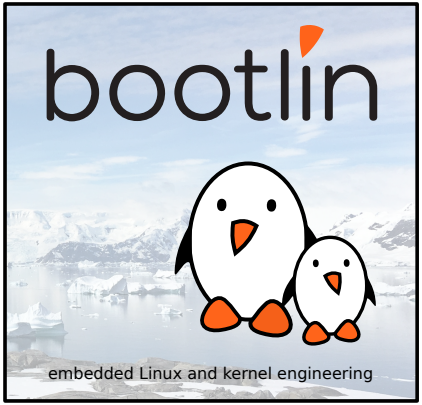
Precision time protocol (PTP) and packet timestamping in Linux

Antoine Ténart
atenart+elce2020@kernel.org



Introduction

Antoine Ténart
atenart+elce2020@kernel.org



Preamble - goals

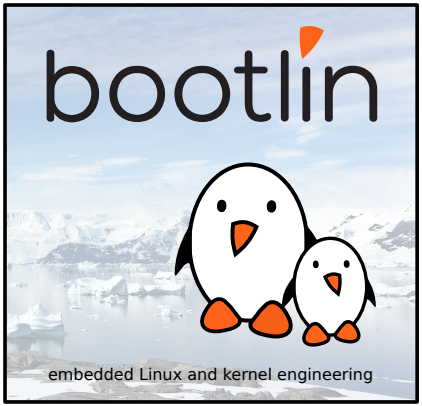
- Understand what is the precision time protocol (PTP) and its modes of operation.
- Have a first glance at what is a packet timestamping and how the kernel supports it.
  - **Disclaimer:** packet timestamping can be used in various applications, we'll only cover it in regard to PTP.
- Understand why hardware timestamping of packets is beneficial.
- See how PTP offloading support (hardware timestamping and PTP hardware clock) can be provided by device drivers.

Agenda

1. Background
2. Overview of the precision time protocol (PTP)
3. Packet timestamping
4. PTP offloading support in Linux
5. User-space PTP implementation

Background

Antoine Ténart
atenart+elce2020@kernel.org



Event ordering

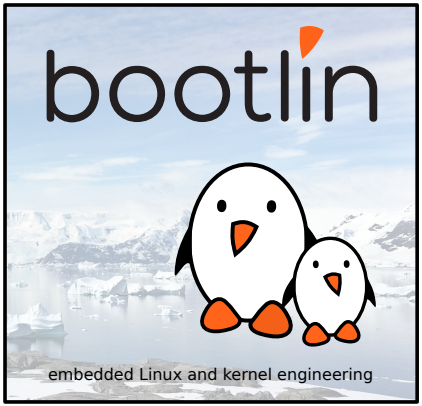
- Event ordering is essential to some applications:
  - Transactions
  - Logs
  - Debugging and performance analysis
- Ordering is based on timestamps...
- ...collected from large ranges of machines.
- Need for clocks synchronization on (local) networks (frequency, phase and time).

NTP

- Network time protocol (NTP) is a network protocol for clock synchronization.
- Provides accuracy within a few milliseconds (best case scenario).
- Not precise enough for some applications: events can occur within the same millisecond.
- Need for a higher accuracy.

Overview of the precision time protocol (PTP)

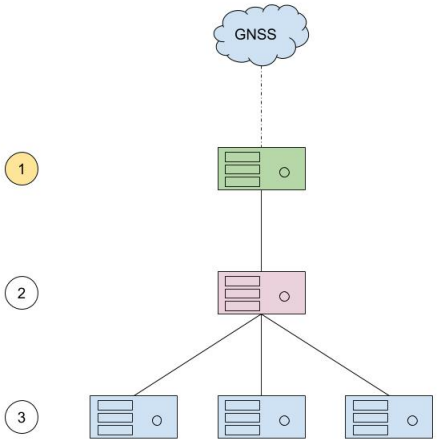
Antoine Ténart
atenart+elce2020@kernel.org



Overview

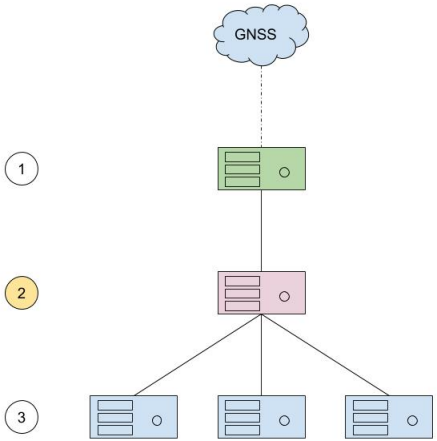
- Precision time protocol (PTP) is a network protocol for clock synchronization.
- Down to sub-microsecond accuracy on local networks.
- Standardized by `IEEE 1588-2002`, `IEEE 1588-2008` and `IEEE 1588-2019`.
- Hierarchical leader/follower architecture for clock distribution.
  - Leader ("grandmaster"), boundary and follower ("slave") clocks.
- PTP packets may be transmitted over Ethernet or UDP over IPv4/IPv6, using multicast or unicast addresses.
  - Depending on the publication used as reference, not all modes are available.

PTP hierarchy 1/3



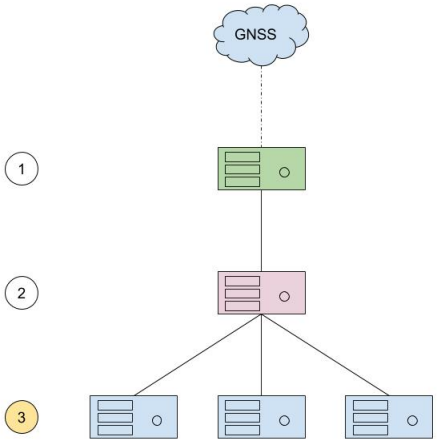
- Leader clock:
  - Time source for the PTP network.
  - Usually synchronize its clock to an external source (GNSS, etc. . .).
  - Is an "ordinary clock" (has a single PTP network connection).

PTP hierarchy 2/3



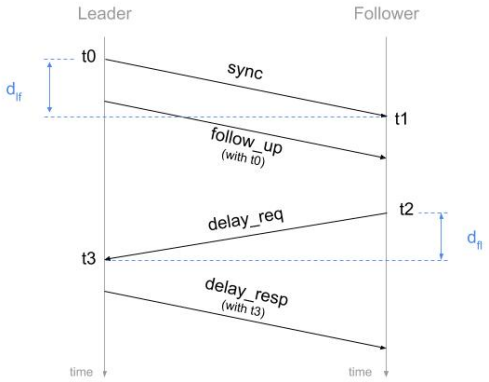
- Boundary clock:
  - Has multiple PTP network connections and relay accurate time:
    - Synchronizes its clock against the leader.
    - Acts as a clock source for the followers.
  - May become the leader if the current leader disappear.
  - Having a boundary clock is optional.

PTP hierarchy 3/3



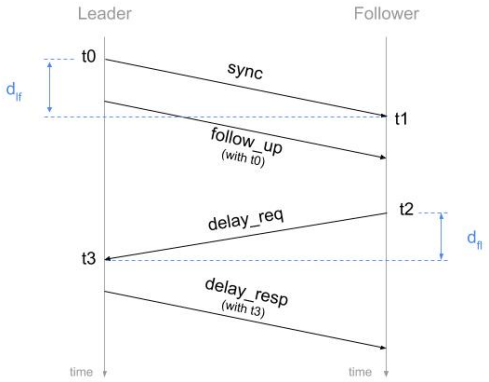
- Follower clock:
  - Synchronize its clock to a leader (here, the boundary clock).
  - May become the leader if the leader disappear.
  - Is an "ordinary clock" (has a single PTP network connection).

PTP synchronization mechanism 1/2



- Time offset is computed based on timestamps of packet sent and received.
- Done by the follower.
- Timestamps made on the leader side are sent to the follower by follow up packets (`follow_up`, `delay_resp`).

PTP synchronization mechanism 2/2



The trip time include the transmit time and the delta between the two clocks:

$$d_{lf} = t_1 - t_0 + \delta t$$

$$d_{fl} = t_3 - t_2 + \delta t$$

We assume the two trip times are equal, hence we have:

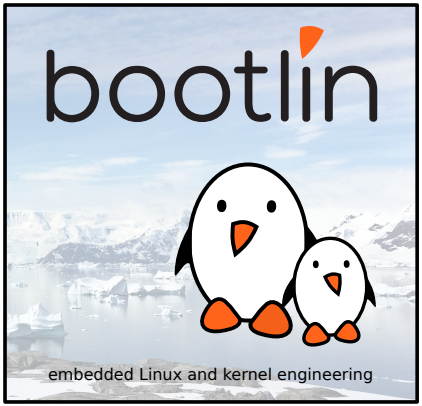
$$\delta t = \frac{1}{2}(t_1 - t_0 + t_2 - t_3)$$

One-step vs two-step

- PTP can work in two operating modes: `1-step` and `2-step`.
- `1-step` includes t_0 in the `sync` packet. There is no `follow_up` packet.
  - The difference lies in the leader side. It needs a hardware enabled device to include t_0 in the `sync` packets.
  - All followers (should) support both modes: there is no hardware requirement for receiving `1-step sync` packets.

Packet timestamping

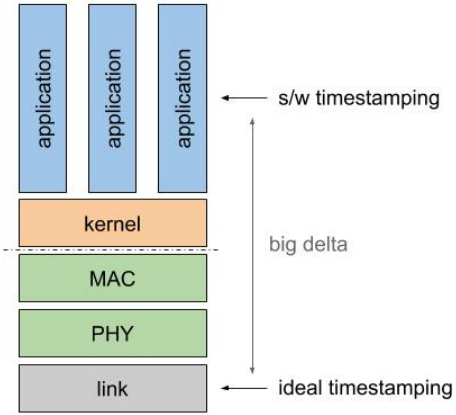
Antoine Ténart
atenart+elce2020@kernel.org



Timestamp accuracy

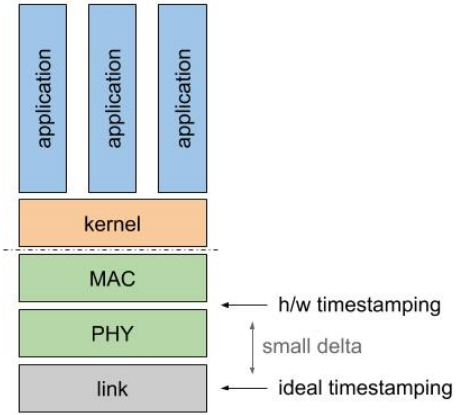
- Packet timestamps, when used for PTP, must be **accurate**: they play a critical role in the time offset computation.
- Ideally we would like a timestamp issued at the exact time of transmission, when the packet leave the device.
- Not possible in the real world, the timestamp has to occur before.
- Two possibilities: in software and in hardware.

Software timestamping



- Timestamp is done in the application, or in the kernel.
- Uses the system clock.
- Error and deltas are big:
  - Timestamp is done far away from the actual transmission.
  - A lot can interfere: scheduling, queuing, interrupts...

Hardware timestamping

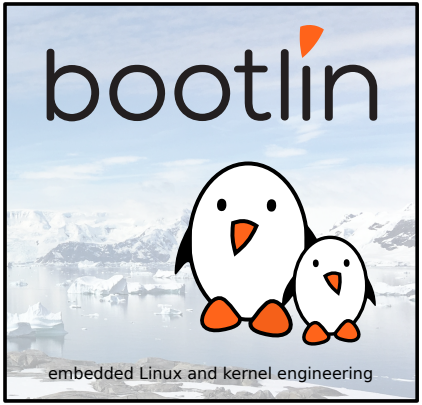


- Timestamp is done in the hardware.
  - Can be done in the `MAC`,
  - in a `PHY`,
  - or using a dedicated controller.
- Uses a PTP hardware clock (PHC).
- Error and deltas are small.
  - Timestamp occurs close to the actual transmission.
  - The packet is already in the hardware.

PTP offloading support in Linux

Antoine Ténart

atenart+elce2020@kernel.org



Overview

- Two mechanisms are combined to provide support for offloading PTP packets timestamping:
  - The `SO_TIMESTAMPING` socket option.
  - The PTP hardware clock (PHC) infrastructure.
- Read the full documentation at `Documentation/networking/timestamping.rst`.

SO_TIMESTAMPING (1/3)

- Configured using `setsockopt`.
- Generates timestamps on reception, transmission or both.
- Works for streams and datagrams.
- Supports multiple timestamp sources:
  - `SOF_TIMESTAMPING_RX_SOFTWARE`: timestamp is generated just after the network device driver hands the packet to the Rx stack.
  - `SOF_TIMESTAMPING_TX_SOFTWARE`: timestamp is generated in the network device driver, as close as possible to passing the packet to the hardware. Requires driver support and may not be available for all devices.
  - `SOF_TIMESTAMPING_RX_HARDWARE`: requires driver support.
  - `SOF_TIMESTAMPING_TX_HARDWARE`: requires driver support.
  - and two other options not used for PTP applications: `SOF_TIMESTAMPING_TX_SCHED` and `SOF_TIMESTAMPING_TX_ACK`.

SO_TIMESTAMPING (2/3)

- Hardware timestamping must be initialized for each device driver expected to be used.
- Configuration passed using the `SIOCSHWTSTAMP` ioctl. Must choose a `tx_type` and an `rx_filter`.
- Possible values for `tx_type`:
  - `HWTSTAMP_TX_OFF`
  - `HWTSTAMP_TX_ON`: report timestamps through the socket error queue.
  - `HWTSTAMP_TX_ONESTEP_SYNC`: insert timestamps directly into `sync` packets.
  - `HWTSTAMP_TX_ONESTEP_P2P`: same as before but also insert timestamps into `delay_resp` packets.
- Possible values for `rx_filter`:
  - `HWTSTAMP_FILTER_NONE`
  - `HWTSTAMP_FILTER_ALL`
  - `HWTSTAMP_FILTER_PTP_V2_L2_EVENT`: PTP v2, Ethernet, all event packets.
  - `HWTSTAMP_FILTER_PTP_V2_L4_SYNC`: PTP v2, UDP sync packets.
  - For the full list, see `include/uapi/linux/net_tstamp.h`

SO_TIMESTAMPING (3/3)

```
# 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: 0
Hardware Transmit Timestamp Modes:
        off                   (HWTSTAMP_TX_OFF)
        on                    (HWTSTAMP_TX_ON)
        one-step-sync         (HWTSTAMP_TX_ONESTEP_SYNC)
Hardware Receive Filter Modes:
        none                  (HWTSTAMP_FILTER_NONE)
        all                   (HWTSTAMP_FILTER_ALL)
```

Supporting SO_TIMESTAMPING in a device driver (1/2)

- In a networking Ethernet driver (MAC), implementing:
  - `get_ts_info` in `struct ethtool_ops`
  - `ndo_do_ioctl` in `struct net_device_ops`, for `SIOCSHWTSTAMP` and `SIOCGHWTSTAMP`.
  - Filling `hwtstamps` in `struct skbuff` with Rx timestamps.
  - Calling `skb_tstamp_tx()` when a Tx timestamp is reported by the hardware.
- In a networking PHY or other dedicated engines driver: implementing the `struct mii_timestamper` callbacks, in `struct phy_device`:
  - `ts_info`
  - `hwtstamp`
  - `rxtstamp`
  - `txtstamp` and calling `skb_complete_tx_timestamp()`

Supporting SO_TIMESTAMPING in a device driver (2/2)

- Both interfaces allow us to:
  1. Report the timestamping capabilities (`ts_info` and `get_ts_info`).
  2. Configure the mode to use (`hwtstamp` and `ndo_do_ioctl`).
  3. Report Rx timestamps (`rxtstamp` and `hwtstamps`).
  4. Report Tx timestamps (`txtstamp`/`skb_complete_tx_timestamp()` and `skb_tstamp_tx`).

PTP hardware clock

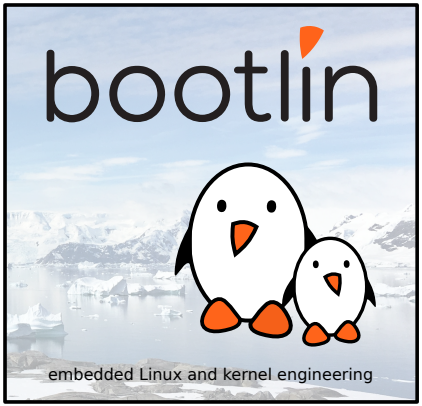
- PHC are used by hardware engines to timestamp packets.
- The PHC must be synchronized.
- Described by `struct ptp_clock_info`, which embeds operation callbacks:
  - `gettimex64`: reports the current time from the hardware clock by filling a `timespec64` structure.
  - `settime64`: sets the time on the hardware clock.
  - `adjfine`: adjusts the frequency of the hardware clock, using an "offset from nominal frequency in parts per million, but with a 16 bit binary fractional field".
  - `adjtime`: shifts the time of the hardware clock by an `s64 delta`.
  - `adjphase`: adjusts the phase of the hardware clock by an `s32 phase`.
- The structure also contains a few parameters, including `max_adj` which defines the maximum frequency adjustment in parts per billion.

Driver examples

- At Bootlin, we had the opportunity to introduce PTP offloading support for some hardware engines.
- For the Microsemi Ocelot network switch:
  - `drivers/net/ethernet/mscc/`
- For the Microsemi VSC85xx PHYs:
  - `drivers/net/phy/mscc/mscc_ptp.c`

User-space PTP implementation

Antoine Ténart
atenart+elce2020@kernel.org



Linux PTP

- The Linux PTP project is an implementation of the Precision Time Protocol according to `IEEE-1588`, for Linux.
- Maintained by Richard Cochran, who also maintains PTP support in Linux.
- Provides a reliable implementation of PTP for Linux, and correctly uses the kernel interfaces for PHC and timestamping.
- Provides a few utilities including `ptp4l` and `phc2sys`.

The ptp4l command

- Implementation of PTP, for ordinary and boundary clocks.
- Can use software or hardware timestamping.
- Can perform PTP operations on top of UDP (IPv4/IPv6) and Ethernet.
- Can optionally use a configuration file.

```
# ptp4l -i eth0 -H -2 -m
selected /dev/ptp4 as PTP clock
port 1: INITIALIZING to LISTENING on INIT_COMPLETE
port 0: INITIALIZING to LISTENING on INIT_COMPLETE
port 1: new foreign master 7e7618.fffe.b52b26-1
selected best master clock 7e7618.fffe.b52b26
port 1: LISTENING to UNCALIBRATED on RS_SLAVE
master offset       1949 s0 freq    -552 path delay      2807
master offset       1953 s2 freq    -548 path delay      2807
port 1: UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
master offset       1974 s2 freq   +1426 path delay      2807
master offset        629 s2 freq    +673 path delay      2807
```

The `phc2sys` command

- Synchronizes two (or more) clocks.
- Typically used to keep the system clock in sync with the PHC.
- When using hardware timestamping, `ptp4l` adjusts the PHC and then `phc2sys` adjusts the system clock.
- When using software timestamping, `phc2sys` isn't used; the system clock is directly adjusted by `ptp4l`.

Thank you!
Questions? Comments?

info@bootlin.com