Ethernet timestamping techniques for real-time performance assessment

Thomas Grudzien, ESO

Nicolas Benes, ESO

Topics:

ERSPANv2
Hardware timestamping with PCI-E NICs
Napatech
Timestamp processing

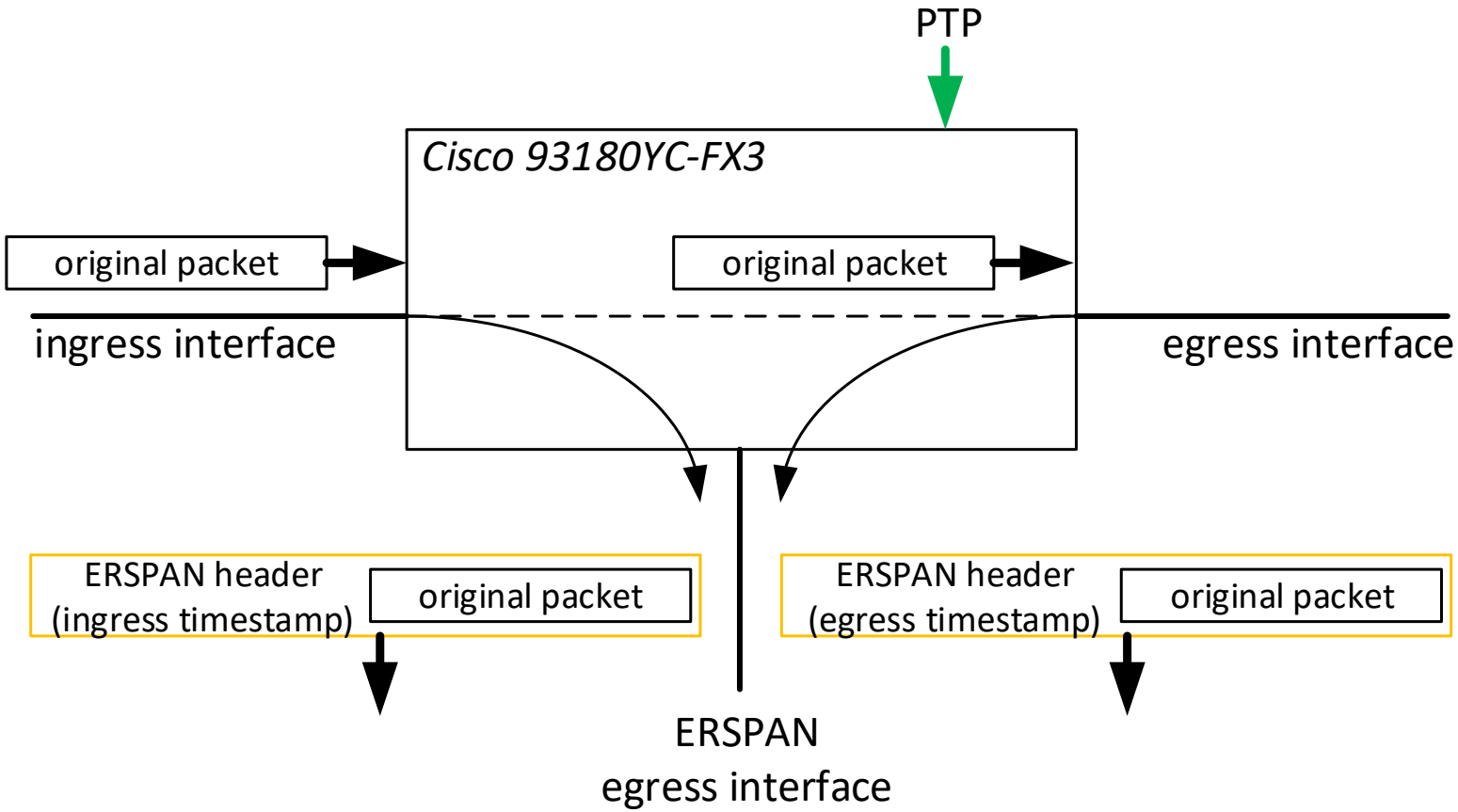
ERSPANv2

ERSPANv2

- Traffic copy feature of some Cisco Nexus switches, evolution of ERSPANv1
  - Behaviour of the 93180YC-FX3 presented herein,
  - Likely model (hardware) dependent.
- Copy of an Ethernet frame or IP datagram
  - Original frame / datagram encapsulated in an ERSPAN type 3 header,
  - ERSPANv2 packet sent to a configurable destination IP address,
  - The ERSPAN type 3 header contains a timestamp:

    -Taken ingress, when the original data starts to enter the ingress interface,
    -Or taken egress, when the original data starts to exit the egress interface.

ERSPANv2


PTP
Cisco 93180YC-FX3
original packet
original packet
ingress interface
egress interface
ERSPAN header
(ingress timestamp)
original packet
ERSPAN header
(egress timestamp)
original packet
ERSPAN
egress interface

ERSPANv2

- Allows to measure switch (or network) transit time,
  - Timestamp resolution 100 picoseconds,
  - Limited by PTP accuracy, in practice a few 100's nanoseconds.
- Requires the switch to be synchronized by PTP,
- Doesn't require any specific feature on the network adapter capturing ERSPAN traffic,
- Timestamps are relative to an epoch (origin) sent periodically in ERSPAN "marker packets" interleaved with the rest of the ERSPAN traffic,
- Decoding ERSPAN and the timestamps requires* custom live traffic parsers or PCAP parsers,

*Native Linux ERSPAN-type link not tested.

ERSPANv2 (type 3 header)

```
> Frame 8: 139 bytes on wire (1112 bits), 139 bytes captured (1112 bits) on interface eth1, id 0
> Ethernet II, Src: Cisco_5d:74:fb (48:2e:72:5d:74:fb), Dst: Broadcom_6f:b2:70 (5c:6f:69:6f:b2:70)
> Internet Protocol Version 4, Src: 192.168.10.2, Dst: 192.168.10.1
> Generic Routing Encapsulation (ERSPAN III)
v Encapsulated Remote Switch Packet ANalysis Type III
    0010 .... .... .... = Version: Type III (2)
    .... 0000 0000 0000 = Vlan: 0
    000. .... .... .... = COS: 0
    ...0 0... .... .... = Bad/Short/Oversized: Good or unknown integrity (0)
    .... .0.. .... .... = Truncated: Not truncated (0)
    .... ..00 0000 1010 = SpanID: 10
    0101 0101 1001 0000 1111 1010 0100 1001 = Timestamp: 1435564617
    0000 0000 0000 0000 = Security Group Tag: 0
    0... .... .... .... = Has Ethernet PDU: 0
    .000 00.. .... .... = Frame Type: Ethernet (0)
    .... ..00 0001 .... = Hardware ID: 1
    .... .... .... 0... = Direction: Ingress
    .... .... .... .11. = Timestamp granularity: Custom granularity (3)
    .... .... .... ...1 = Optional Sub headers: 1
    0001 11.. .... .... .... .... .... .... = Platform ID: 7
    .... ..00 0000 .... .... .... .... .... = Reserved: 0
    .... .... .... 0000 0000 0001 0010 1100 = Source Index: 300
    0000 0000 0000 0000 0101 0011 0011 0001 = Upper 32-bit Timestamp: 21297
> Ethernet II, Src: Broadcom_6f:ca:a0 (5c:6f:69:6f:ca:a0), Dst: IPv4mcast_01 (01:00:5e:00:00:01)
> Internet Protocol Version 4, Src: 192.168.50.1, Dst: 239.128.0.1
> User Datagram Protocol, Src Port: 60000, Dst Port: 60000
> Data (35 bytes)
```

ERSPANv2 (type 3 header)

- Type 3 header
  - Natively parsed by Wireshark (4.0.10 tested).
- Contains two 32-bit timestamp fields, “upper 32-bit timestamp” and “timestamp”
  - Need to be concatenated in a 64-bit unsigned long int,
  - The “upper 32-bit timestamp” is the most significant 32-bit word,
  - The “timestamp” is the least significant 32-bit word,
  - The 64-bit long int contains the number of 100 picoseconds ticks since epoch,
  - Epoch is contained in the “marker packets” (next slide).

ERSPANv2 (marker packet)

```
> Frame 7: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface eth1, id 0
> Ethernet II, Src: Cisco_5d:74:fb (48:2e:72:5d:74:fb), Dst: Broadcom_6f:b2:70 (5c:6f:69:6f:b2:70)
> Internet Protocol Version 4, Src: 192.168.10.2, Dst: 192.168.10.1
> User Datagram Protocol, Src Port: 59564, Dst Port: 8880
v CISCO ERSPAN3 Marker Packet
    Proprietary CISCO Header: 004022eb000234dd82fc40000000000000000410
    Header: True
    .... 0001 .... .... = Version: 1
    0000 .... .... .... = Type: 0
    .... .... 0000 0001 = SSID: 1
    0000 0100 .... .... = Granularity: 4
    .... .... 0010 0101 = UTC Offset: 37
    0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 = ASIC 48-bit Timestamp: 0
    0110 0011 1010 0000 0101 1111 1111 0010 = UTC Seconds: 1671454706
    0010 1001 1110 1010 0011 1111 0011 0011 = UTC Microseconds: 703217459
    0000 0000 0000 0010 0011 0100 1101 1101 = Sequence Number: 144605
    0000 0000 0000 0000 1111 0110 1110 0111 = Reserved: 63207
    .... .... .... .... .... .... .... .... 1010 0101 1010 0101 1010 0101 1010 0101 = TAIL: 0xa5a5a5a5
```

ERSPANv2 (marker packet)

- Not decoded natively by Wireshark (4.0.10 tested),
  - Right click packet, decode as, select “CISCO ERSPAN3 MARKER” from the “current” field,
  - Still, the two relevant fields are not decoded properly (switch dependent?).
- “UTC seconds” should be interpreted as “TAI seconds” (if PTP is in TAI),
- “UTC microseconds” should be interpreted as “TAI nanoseconds” (if PTP is in TAI)
- To be added to the ERSPAN header timestamps to get the timestamps in TAI
- Capturing and decoding one marker packet should be enough for the whole ERSPANv2 traffic captured in a session
  - A change of origin in the marker packet is not expected to happen during a realistic capture session (long rollover time)

ERSPANv2 pitfalls

- Rounding errors when using IEEE 754 double precision (64 bit)
  - For timestamp computations including the origin, one may be tempted to use a double precision float, but timestamps currently have 19 significant digits e.g. 1,671,454,706.703217459,
  - Loss of significance: double precision can only accommodate ~16 significant digits. In this case the timestamp precision quantum is 2^-22 ~ 0.24µs, and 2^-21 from 2038 on,
  - Solution(s): for timestamp computations use int64_t and nanoseconds, or use a closer epoch so <seconds>.<nanoseconds> is a double with less than 15 significant digits,
  - 0.24µs is however consistent with the typical PTP accuracy, and usually doesn't matter for the delays measured in the ELT use-cases.
- The 93180YC-FX{2|3} is "store-and-forward" for unknown unicast and for multicast traffic
  - Ingress serialization time to take into account: packet size divided by media speed

ERSPANv2 take-away

- Convenient source of truth:
  - Feature available on the 93180YC-FX3, which is (at the time of writing) the recommended switch model for AO RTC and ICS local communication infrastructures,
  - High precision hardware timestamping for any traffic, taken when traffic enters the switch (start of frame delimiter) and / or when traffic is ready to leave the switch (first bit sent egress),
  - Capability to measure when a packet is sent out of an RTC node (=ingress ERSPAN timestamp), which could be difficult to measure at the node level,
- Will likely be used in the ELT network infrastructure as a “timestamping network”
  - Indisputable way to measure the network transit time
- One interesting outcome: the measured multicast **routing** latency of a 93180YC-FX3 is ~1.2µs

ERSPANv2 NX-OS config snippets

Ingress timestamping:

```
monitor session <1-4> type erspan-source
 header-type 3
 marker-packet <interval in ms>
 erspan-id <1-4>
 vrf default
 destination ip <destination IP of ERSPAN traffic>
 source interface <switch interface ID to monitor> rx
 no shutdown

monitor erspan origin ip-address <src IP of E. traffic> global
```

Egress timestamping:

```
monitor session <1-4> type erspan-source
 header-type 3
 marker-packet <interval in ms>
 erspan-id <1-4>
 vrf default
 destination ip <destination IP of ERSPAN traffic>
 source interface <switch interface ID to monitor> tx
 no shutdown

monitor erspan origin ip-address <src IP of E. traffic> global
```

Hardware timestamping with (certain) PCI-E network adapters

Hardware timestamping with PCI-E NICs

- Certain general-purpose Ethernet adapters have the capability to timestamp all traffic
  - How to check? Use ethtool:

```
# ethtool -T <interface name>
Time stamping parameters for <interface name>:
Capabilities:
        hardware-transmit
        hardware-receive
        hardware-raw-clock
PTP Hardware Clock: 1
Hardware Transmit Timestamp Modes:
        off
        on
Hardware Receive Filter Modes:
        none
        all
```

Hardware timestamping with PCI-E NICs

- 100GE adapters:
  - Mellanox (now Nvidia) ConnectX-6 Dx (MCX623106AN-CDAT)
  - Intel E810-CQDA2 (not tested)
- 10/25GE adapters:
  - Mellanox (now Nvidia) ConnectX-6 Lx (MCX631102AN-ADAT),
  - Intel E810-XXVDA{2|4} (not tested)
- 1GE adapters
  - Intel cards based on the 82580 chip, i.e. model i340-4T (not tested)

Hardware timestamping with PCI-E NICs

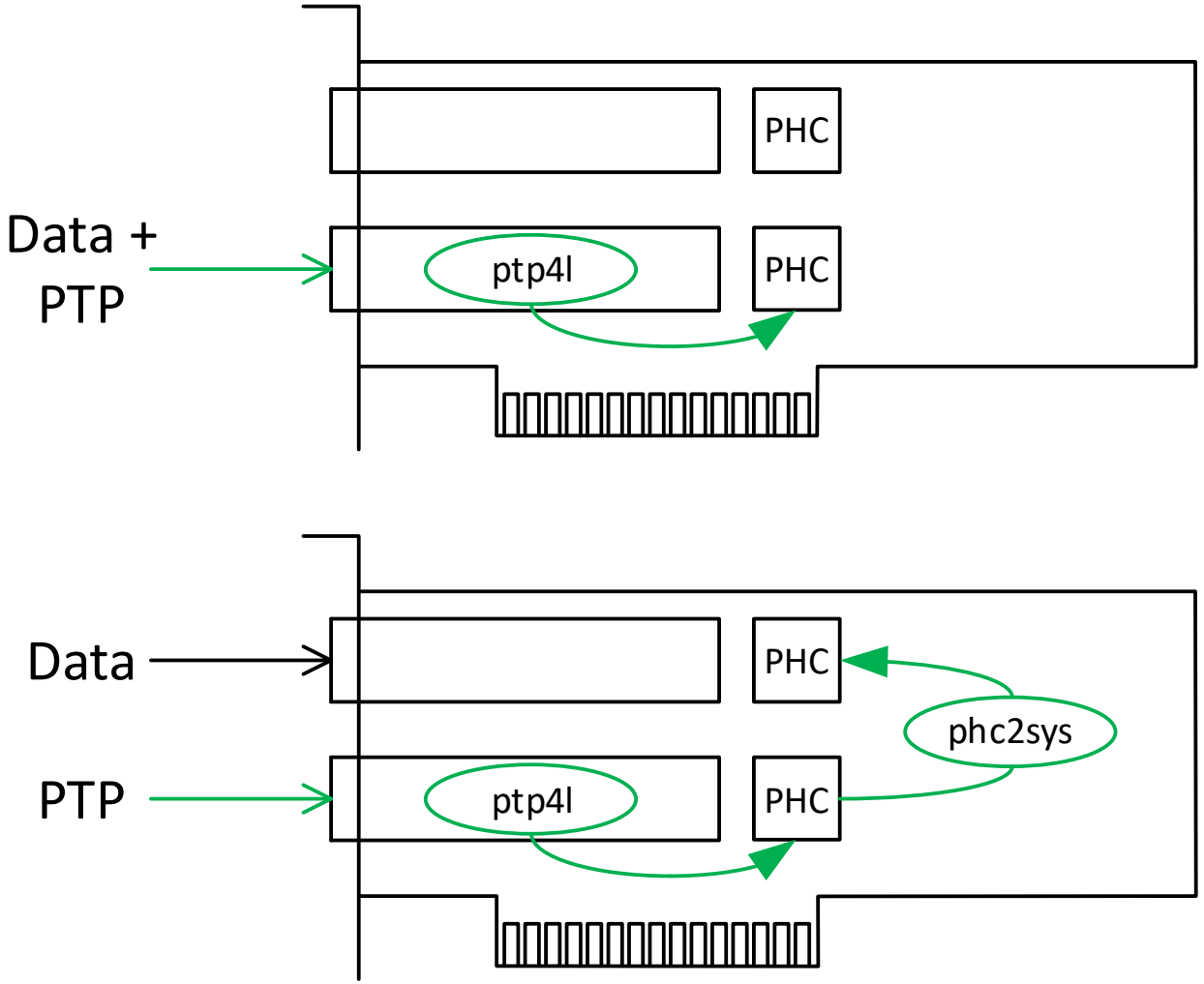
- The HW timestamps are in the PCAP timestamps, but how to enable them?
  - Using tshark (part of wireshark, or wireshark-cli depending on the Linux distribution), or tcpdump:

```
# tshark | tcpdump (…) --time-stamp-type adapter_unsynced (…)
```

  - The “adapter_unsynced” syntax can be misleading. From the pcap-tstamp(7) manpage: “*this is a high-precision timestamp, it is not synchronized with the host operating system’s clock*” i.e. the adapter needs to be synchronized by PTP:

    - Either by ptp4l if PTP is received on the traffic capture interface
    - Or by ptp4l + phc2sys if PTP is received on a different interface, in which case ptp4l synchronizes the PHC of the PTP interface and phc2sys synchronizes the PHC of the capture interface from the PHC of the PTP interface (picture next slide)

Hardware timestamping with PCI-E NICs



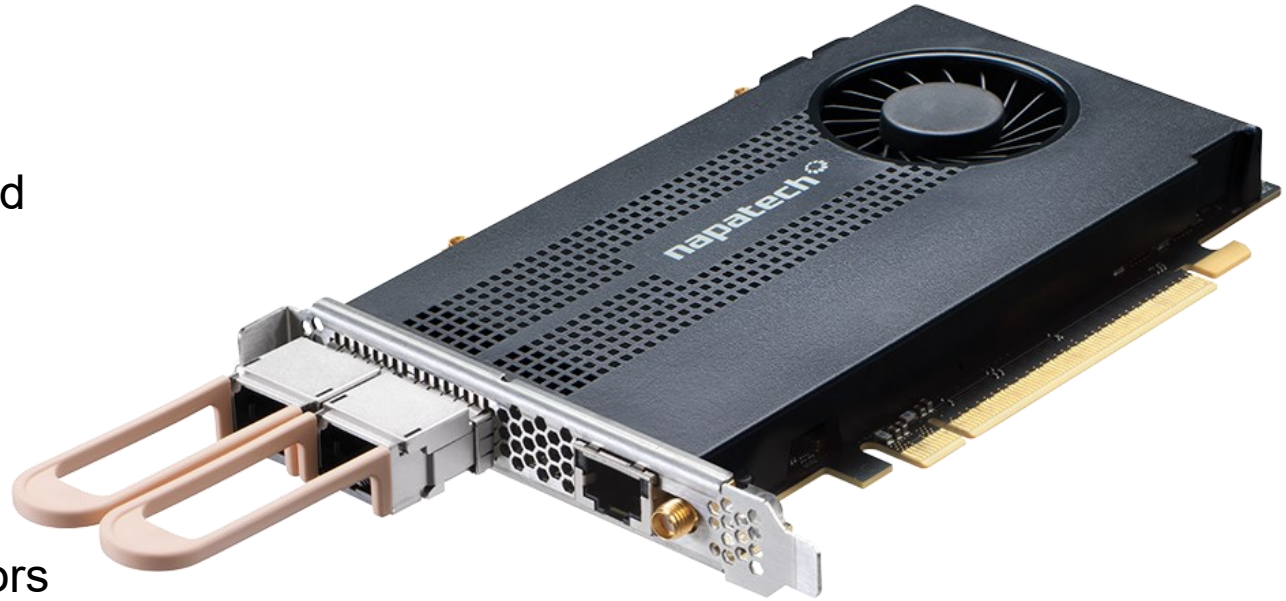
Hardware timestamping with PCI-E NICS take-away

- Convenient / affordable timestamping:
  - The Nvidia / Mellanox ConnectX-6 Lx is a good all-around card, good for PTP, easy to tune for deterministic applications, good for capture with hardware timestamping,
  - Used to capture the full M1 LCS traffic: ~0.8 Mpps without losses (not tested above).
- Performance impact:
  - Unless the RTC software can parse timestamps directly, there is a performance hit from running a capture process in addition. In that case, the capture should be done out-of-band with a dedicated adapter which will need a copy of the traffic,
  - To create this copy on the switch: use the SPAN feature for unicast or multicast traffic, or configure a static igmp snooping entry for multicast traffic (preferred for multicast).

Napatech

Napatech NT200A02 sniffer card with HW timestamping

- FPGA images for
  2x100 Gbit/s, 2x40 Gbit/s, 4x25/10 Gbit/s,
  2x25/10 Gbit/s, 8x10 Gbit/s, 2x10 Gbit/s
  Adapters and break-out cables may be needed
- 2x QSFP28 data ports
- 1x RJ45 for PTP synchronization
- 1x SMA connector for e.g. PPS sync.
- Multiple internal sync and expansion connectors
  e.g. to other Napatech card
- PCIe Gen3 x16 host interface
- sustained 200 Gbit/s not supported
- On-board 12 GiB RAM buffer



https://docs.napatech.com/r/NT200A02-SCC-Installation/Specifications

Napatech NT200A02 sniffer

- Native API and libpcap interface, does *not* work as NIC on Linux
- High quality documentation but a lot of it
- Extensive filtering and deep packet inspection functionality, most of which we do not use (exception: slicing for truncating packets after the protocol headers)
- Expensive (~10 k€)
- At ESO:
  Timestamping system in few selected lab setups,
  Reference for testing of other timestamping methods
- Sniffer cards will *not* be available at Armazones

https://docs.napatech.com/r/NT200A02-SCC-Installation/Specifications

Timestamp processing

Timestamp processing

- libpcap, see pcap (3pcap):
  - Capture from live device or read from capture file
  - Use filters to select your packet streams (see pcap-filter),
    they have very low overhead and greatly simplify your packet processing
  - Iterate over filtered packets
  - Extract/analyse timestamps e.g. PCAP (i.e. wire) timestamps, protocol timestamps (e.g. MUDPI)
- In-band hardware timestamps:
  - Setup socket for hardware timestamping with rx_filter=HWTSTAMP_FILTER_ALL
  - Use recvmsg with buffer in msg_control and extract the returned raw timestamp (e.g. int64_t nanoseconds, TAI)
  - HW and SW timestamping can be used simultaneously

More information

ESO-519864 *Timestamping with the Cisco ERSPAN3 protocol*

ESO-415414 *Timestamping with the Cisco Nexus 93180YC-FX and the Mellanox ConnectX-6 Dx*

ESO-385223 *Standard Ethernet Cards for the ELT Control System*

Thank you!

Thomas Grudzien
thomas.grudzien@eso.org

Nicolas Benes
nicolas.benes@eso.org

@ESOAstronomy
@esoastronomy
@ESO
european-southern-observatory
@ESOobservatory