Time-Stamping

Table of contents

List of Figures

Time-Stamping Service

Time-stamping is the process of keeping track of the creation of a packet. A time-stamping service supports assertions of proof that a datum existed before a particular time. Incoming packets are time-stamped before they are distributed on the PCI depending on the congestion in the PCI buffers. Outgoing packets are time-stamped very close to placing them on the wire.

Enabling Time-Stamping

Time-stamping is off by default and should be enabled before use.

To enable

➢ ***time-stamping for a socket:***

Call setsockopt() with SO_TIMESTAMPING and with the following flags:

SOF_TIMESTAMPING_TX_HARDWARE:	try to obtain send time-stamp in hardware
SOF_TIMESTAMPING_TX_SOFTWARE:	if SOF_TIMESTAMPING_TX_HARDWARE is off or fails, then do it in software
SOF_TIMESTAMPING_RX_HARDWARE:	return the original, unmodified time-stamp as generated by the hardware
SOF_TIMESTAMPING_RX_SOFTWARE:	if SOF_TIMESTAMPING_RX_HARDWARE is off or fails, then do it in software
SOF_TIMESTAMPING_RAW_HARDWARE:	return original raw hardware time-stamp
SOF_TIMESTAMPING_SYS_HARDWARE:	return hardware time-stamp transformed into the system time base
SOF_TIMESTAMPING_SOFTWARE:	return system time-stamp generated in software
SOF_TIMESTAMPING_TX/RX	determine how time-stamps are generated
SOF_TIMESTAMPING_RAW/SYS	determine how they are reported

***ping for* a net device:**

Admin privileged user can enable/disable time stamping through calling ioctl (sock, SIOCSH-WTSTAMP, &ifreq) with the following values:

- Send side time sampling, enabled by ifreq.hwtstamp_config.tx_type when:

```
/* possible values for hwtstamp_config->tx_type */
enum hwtstamp_tx_types {
/*
* No outgoing packet will need hardware time stamping;
* should a packet arrive which asks for it, no hardware
* time stamping will be done.
*/
HWTSTAMP_TX_OFF,

/*
* Enables hardware time stamping for outgoing packets;
* the sender of the packet decides which are to be
* time stamped by setting %SOF_TIMESTAMPING_TX_SOFTWARE
* before sending the packet.
*/
HWTSTAMP_TX_ON,
/*
* Enables time stamping for outgoing packets just as
* HWTSTAMP_TX_ON does, but also enables time stamp insertion
* directly into Sync packets. In this case, transmitted Sync
* packets will not received a time stamp via the socket error
* queue.
*/
HWTSTAMP_TX_ONESTEP_SYNC,
};
Note: for send side time stamping currently only HWTSTAMP_TX_OFF and
HWTSTAMP_TX_ON are supported.
```

- Receive side time sampling, enabled by ifreq.hwtstamp_config.rx_filter when:

```
/* possible values for hwtstamp_config->rx_filter */
```

```
enum hwtstamp_rx_filters {
/* time stamp no incoming packet at all */
HWTSTAMP_FILTER_NONE,

/* time stamp any incoming packet */
HWTSTAMP_FILTER_ALL,
/* return value: time stamp all packets requested plus some others */
HWTSTAMP_FILTER_SOME,

/* PTP v1, UDP, any kind of event packet */
HWTSTAMP_FILTER_PTP_V1_L4_EVENT,
/* PTP v1, UDP, Sync packet */
HWTSTAMP_FILTER_PTP_V1_L4_SYNC,
/* PTP v1, UDP, Delay_req packet */
HWTSTAMP_FILTER_PTP_V1_L4_DELAY_REQ,
/* PTP v2, UDP, any kind of event packet */
HWTSTAMP_FILTER_PTP_V2_L4_EVENT,
/* PTP v2, UDP, Sync packet */
HWTSTAMP_FILTER_PTP_V2_L4_SYNC,
/* PTP v2, UDP, Delay_req packet */
HWTSTAMP_FILTER_PTP_V2_L4_DELAY_REQ,
/* 802.AS1, Ethernet, any kind of event packet */
HWTSTAMP_FILTER_PTP_V2_L2_EVENT,
/* 802.AS1, Ethernet, Sync packet */
HWTSTAMP_FILTER_PTP_V2_L2_SYNC,
/* 802.AS1, Ethernet, Delay_req packet */
HWTSTAMP_FILTER_PTP_V2_L2_DELAY_REQ,

/* PTP v2/802.AS1, any layer, any kind of event packet */
HWTSTAMP_FILTER_PTP_V2_EVENT,
/* PTP v2/802.AS1, any layer, Sync packet */
HWTSTAMP_FILTER_PTP_V2_SYNC,
/* PTP v2/802.AS1, any layer, Delay_req packet */
HWTSTAMP_FILTER_PTP_V2_DELAY_REQ,
};
Note: for receive side time stamping currently only HWTSTAMP_FILTER_NONE
and
HWTSTAMP_FILTER_ALL are supported.
```

Getting Time-Stamping

Once time stamping is enabled time stamp is placed in the socket Ancillary data. recvmsg() can be used to get this control message for regular incoming packets. For send time stamps the outgoing packet is looped back to the socket's error queue with the send time-stamp(s) attached. It can
be received with recvmsg (flags=MSG_ERRQUEUE). The call returns the original outgoing packet data including all headers prepended down to and including the link layer, the scm_time-stamping control message and a sock_extended_err control message with ee_errno==ENOMSG and ee_origin==SO_EE_ORIGIN_TIMESTAMPING. A socket with such a pending bounced packet is ready for reading as far as select() is concerned. If the outgoing packet has to be fragmented, then only the first fragment is time stamped and returned to the sending socket.

Note

When time-stamping is enabled, VLAN stripping is disabled. For more info please refer to Documentation/networking/timestamping.txt in kernel.org.

Note

On ConnectX-4 and above adapter cards, when time-stamping is enabled, RX CQE compression is disabled (features are mutually exclusive).

Time Stamping Capabilities via ethtool

To display Time Stamping capabilities via ethtool:

Show Time Stamping capabilities:

```
ethtool -T eth<x>
```

Example:

```
ethtool -T eth0
Time stamping parameters for p2p1:
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: 1
Hardware Transmit Timestamp Modes:
off (HWTSTAMP_TX_OFF)
on (HWTSTAMP_TX_ON)

Hardware Receive Filter Modes:
none (HWTSTAMP_FILTER_NONE)
all (HWTSTAMP_FILTER_ALL)
```

For more details on PTP Hardware Clock, please refer to:
https://www.kernel.org/doc/Documentation/ptp/ptp.txt

Steering PTP Traffic to Single RX Ring

As a result of Receive Side Steering (RSS) PTP traffic coming to UDP ports 319 and 320, it may reach the user space application in an out of order manner. In order to prevent this, PTP traffic needs to be steered to single RX ring using ethtool.

Example:

```
# ethtool -u ens7
8 RX rings available
Total 0 rules
# ethtool -U ens7 flow-type udp4 dst-port 319 action 0 loc 1
# ethtool -U ens7 flow-type udp4 dst-port 320 action 0 loc 0
# ethtool -u ens7
8 RX rings available
Total 2 rules
Filter: 0
Rule Type: UDP over IPv4
Src IP addr: 0.0.0.0 mask: 255.255.255.255
Dest IP addr: 0.0.0.0 mask: 255.255.255.255
TOS: 0x0 mask: 0xff
Src port: 0 mask: 0xffff
Dest port: 320 mask: 0x0
Action: Direct to queue 0
Filter: 1
Rule Type: UDP over IPv4
Src IP addr: 0.0.0.0 mask: 255.255.255.255
Dest IP addr: 0.0.0.0 mask: 255.255.255.255
TOS: 0x0 mask: 0xff
Src port: 0 mask: 0xffff
Dest port: 319 mask: 0x0
Action: Direct to queue 0
```

Tx Port Time-Stamping

Transmitted packet time-stamping accuracy can be improved when using a timestamp generated at the port level instead of a timestamp generated upon CQE creation. Tx port time-stamping better reflects the actual time of a packet's transmission.

Normal Send queues (SQs) are open with CQE time-stamp support. When this feature is enabled, the driver is expected to open extra Tx port time-stamped SQ per traffic class

(TC).

The stream must meet the following conditions in order to be transmitted through a Tx port time-stamped SQ.

1. SKBTX_HW_TSTAMP flag was set at tx_flag (SO_TIMESTAMPING was set via setsockopt() or similarly)
2. Packet type is:
    1. Non-IP, with EtherType of PTP over IEEE 802.3 (0x88f7)
       or
    2. UDP over IPv4/IPv6

This feature is disabled by default in order to avoid extra SQ memory allocations. The feature can be enabled or disabled using the following command.

```
ethtool --set-priv-flags <ifs-name> tx_port_ts on / off
```

PTP Cyc2time Hardware Translation Offload

Note

This feature is supported on ConnectX-6 Dx and above adapter cards only.

Overview

Device timestamp can be in one of two modes: real time or free running internal time.
In free running internal time mode, the device clock is not editable in any way. Driver and/or user space must adjust it to the real-time nanosecond values.
In real time mode, the hardware clock device can be adjusted and can provide timestamps which are already translated into real-time nanoseconds.

Both modes are global per device. Once a mode is set, all clock-related features (such as PPS, CQE TS, PCIe bar, etc) will work with the chosen clock mode only.

Free running internal time is the default mode configured in the hardware. The driver will modify the hardware real time clock based on PTP daemon clock adjustments.

Only physical functions are allowed to modify the hardware real-time clock, so PTP daemon adjustments from VFs will be treated as NOP. In case more than one physical function tries to modify the hardware real-time clock, the device will select one of the functions as its designated clock provider. All other input will also be treated as a NOP. The designated clock provide can be replaced by the device if no new adjustments have been received from the current provider after some period.

Timestamp Format

CQE hardware timestamp format for ConnectX-6 Dx and ConnectX-6 Lx NICs is 64 bit, as follows.

{32bit sec, 32 bit nsec}

Configuration

In order to enable the feature, set REAL_TIME_CLOCK_ENABLE in NV_CONFIG via mlxconfig and **restart the driver**.

Limitations

- Administrator must restart the driver and perform a FW reset for the configuration to take effect. Otherwise, mismatch between HW and driver timestamp mode might occur.
- Once real time mode is activated on a given device (see configuration section), version 5.3 or newer must run on all device functions. Any older driver running on a device function at this configuration will fail to open any traffic queues (RDMA or ETH), hence becoming dysfunctional.
- In real time mode, all device functions must be PTP-synchronized by a single clock domain—do not use multiple GMs for different functions on the same device.

- Regarding hardware clock ownership, the hardware is configured only from a single elected function; other function settings are ignored by the device. There is no indication as to which function is the hardware-clock’s owner. After an internal timeout without modifying the hardware clock, a function losses the hardware-clock’s ownership and is open to be grasped by any of the functions.
- All PFs/VFs within the same device must sync to the same 1588 master clock. If multiple masters are used, the device will use a single elected function. This might lead to wrong clock representation by device, wrong 1588 TLVs and hiccups on replacement of elected function.
- This feature is supported on ConnectX-6 Dx and above adapter cards only.

RoCE Time-Stamping

RoCE Time-Stamping allows you to stamp packets when they are sent to the wire/received from the wire. The time-stamp is given in raw hardware cycles but could be easily converted into hardware referenced nanoseconds based time. Additionally, it enables you to query the hardware for the hardware time, thus stamp other application's event and compare time.

Query Capabilities

Time-stamping is available if and only the hardware reports it is capable of reporting it. To verify whether RoCE Time-Stamping is available, run ibv_query_device_ex.

For further information, please see ibv_query_device_ex manual page.

Creating a Time-Stamping Completion Queue

To get time stamps, a suitable extended Completion Queue (CQ) must be created via a special call to ibv_create_cq_ex verb.

For further information, please see ibv_create_cq_ex manual page.

Note

Time Stamping in not available when CQE zipping is used.

Querying the Hardware Time

Querying the hardware for time is done via the ibv_query_rt_values_ex verb. For example:

For further information, please see ibv_query_rt_values_ex manual page.

One Pulse Per Second (1PPS)

1PPS is a time synchronization feature that allows the adapter to be able to send or receive 1 pulse per second on a dedicated pin on the adapter card using an SMA connector (SubMiniature version A). Only one pin is supported and could be configured as 1PPS in or 1PPS out.
For further information, refer to HowTo Test 1PPS on NVIDIA Adapters Community post.