Spinning Applications

If the Onload application is spinning, a received packet is timestamped when the stack is polled at which point the packet is placed on the socket receive queue. Spinning will generally produce more accurate timestamps so long as the receiving application is able to keep pace with the packet arrival rate.

Software Timestamp Values

The value of the software timestamp is the start time for the poll that fetches the packet from the hardware.

Rarely, a packet might arrive during a poll, and can then be given the same timestamp as an earlier packet fetched by the same poll. If you are also using hardware timestamping for such a packet, its software timestamp might be earlier than its hardware timestamp.

Software Timestamp Formats

The format of timestamps is defined by `struct_timeval`.

Applications preferring timestamps with nanosecond resolution can use `SO_TIMESTAMPNS` in place of the normal (microsecond resolution) `SO_TIMESTAMP` value.

Hardware Timestamps

Setting the `SO_TIMESTAMPING` option using `setsockopt()` enables hardware timestamping on TCP or UDP sockets. Timestamps are generated by the adapter for each received packet. A timestamp is generated when the first byte enters the adapter. Timestamp resolution is <8ns on an X2 series adapter.

Functions such as `cmesg()`, `recvmsg()` and `recvmmsg()` can then recover hardware timestamps for packets recovered from a socket.

Requirements

- Supported only on Solarflare Flareon XtremeScale™ SFN8000 and XtremeScale™ X2 series adapters.
- An activation Key for hardware timestamps must be installed on the adapter:
  - The PTP/timestamping activation key is installed during manufacture on the PLUS variants of SFN8000 and X2 series adapters.

- An appropriate activation key can be installed on other SFN8000 and X2 series adapters by the user.

Hardware Timestamp Format

The format of timestamps is defined by `struct_timespec`. Interested users should read the kernel `SO_TIMESTAMPING` documentation for more details of how to use this socket API – kernel documentation can be found, for example, at:

https://www.kernel.org/doc/Documentation/networking/timestamping/

Received Packets

- The Onload stack for the socket must have the environment variable EF_RX_TIMESTAMPING set - see Appendix A: Parameter Reference for details.
- Received packets are timestamped when they enter the MAC on the SFN8000 or X2 series adapter.

Transmitted Packets

Onload supports hardware timestamping of UDP and TCP packets transmitted over a supported interface. A timestamp is generated when the first byte enters the adapter.

Recent Linux kernels support hardware timestamps for TCP, and Onload 8.1 adds similar capability. To recover hardware timestamps for transmitted TCP packets that are similar to the Linux kernel, set the following socket options:

```
SOF_TIMESTAMPING_TX_HARDWARE | SOF_TIMESTAMPING_SYS_HARDWARE |
SOF_TIMESTAMPING_RAW_HARDWARE
```

Because older Linux kernels do not support hardware timestamps for TCP, Onload provides an extension to the standard SO_TIMESTAMPING API with the `ONLOAD_SOF_TIMESTAMPING_STREAM` socket option to support this. To recover hardware timestamps for transmitted TCP packets that use an Onload proprietary format, set the following socket options:

```
SOF_TIMESTAMPING_TX_HARDWARE | SOF_TIMESTAMPING_SYS_HARDWARE |
SOF_TIMESTAMPING_RAW_HARDWARE | ONLOAD_SOF_TIMESTAMPING_STREAM
```

To recover hardware timestamps for transmitted UDP packets, set the following socket options:

```
SOF_TIMESTAMPING_TX_HARDWARE | SOF_TIMESTAMPING_SYS_HARDWARE |
SOF_TIMESTAMPING_RAW_HARDWARE
```

From Onload 8.1 onwards,`SOF_TIMESTAMPING_OPT_ID` and `SOF_TIMESTAMPING_OPT_TSONLY` are also supported with Linux-style timestamps.

Other socket flag combinations, not listed above, will be silently ignored.

To receive hardware transmit timestamps:

- The adapter must support hardware transmit timestamps. The following adapters currently do so:
  - XtremeScale™ SFN8000 series
  - XtremeScale™ X2 series
  - AMD Alveo™ X3 series.
- The adapter must have a PTP/HW timestamping activation key.

  ***Note*:** Alveo X3 series are an exception to this requirement, because they do not use activation keys.
- The adapter must have a SolarCapture Pro activation key or Performance Monitoring activation key.

  ***Note*:** Alveo X3 series are an exception to this requirement, because they do not use activation keys.
- You must set EF_TX_TIMESTAMPING on stacks where transmit timestamping is required.
- You must set EF_TIMESTAMPING_REPORTING to control the type of timestamp returned to the application. This is optional, by default Onload will report translated timestamps if the adapter clock has been fully synchronized to correct time by the Solarflare PTP daemon. In all cases Onload will always report raw timestamps. Refer to Appendix A: Parameter Reference for full details of the EF_TIMESTAMPING_REPORTING variable.
- Solarflare PTP (sfptpd) must be running if timestamps are to be synchronized with an external PTP master clock.

For details of the SO_TIMESTAMPING API refer to the Linux documentation:

https://www.kernel.org/doc/Documentation/networking/timestamping/

Zeroed Timestamps

If timestamps returned from the adapter are zeroed, refer to Setting the Adapter Clock Time.

Synchronizing Time

Solarflare Enhanced PTP can be enabled to synchronize the time across all clocks within a server or between multiple servers.

The sfptpd daemon supports clock synchronization with external NTP and PTP time sources and includes an optional PTP/NTP fallback configuration.

For details of Solarflare PTP refer to the *Enhanced PTP User Guide* (UG1602).

Example Timestamping Applications

The onload distribution includes example applications to demonstrate receive and transmit hardware timestamping. With Onload installed, source code is located in the following subdirectory:

```
/onload-<version>/src/tests/onload/hwtimestamping
```

Building the Examples

Following the onload_install, the example applications: rx_timestamping and tx_timestamping are located in the following directory:

```
onload-<version>/build/gnu_x86_64/tests/onload/hwtimestamping
```

Using earlier versions of Onload the user should run the make command in the following directory to build example timestamping applications:

```
openonload-<version>/src/tests/onload/hwtimestamping
```

Running the Examples

The following conditions are required to run the example applications:

- The server must have a Solarflare SFN8000 or X2 series adapter.
- The adapter must have a PTP/HW timestamping activation key.
- The connection from which packets are to be timestamped must be routed over the timestamping adapter.
- To receive TX timestamps, the adapter must have a SolarCapture Pro activation key or Performance Monitoring activation key
- The Onload environment variable EF_RX_TIMESTAMPING or EF_TX_TIMESTAMPING must be enabled in the Onload environment.

Note: User should also read the specific requirements from the RX/TX timestamping sections above.

Setting the Adapter Clock Time

It might be necessary to ‘seed’ the adapter clock time - otherwise timestamps might be zeroed or reported as 01 Jan 1970. This can be done by briefly running Solarflare PTP (sfptpd) as a slave - the adapter clock is seeded from the system clock.

Running sfptpd in freerun mode will achieve the same result. It is not required to actually receive any PTP packets to seed the adapter clock and sfptpd can be terminated after a few seconds as it is only required to 'seed' the adapter clock.

Users who wish to synchronize the adapter clock with an external time source should refer to the *Enhanced PTP User Guide* (UG1602).

Order of Timestamps in the Example Applications

Timestamps in the example applications are displayed in the following order:

- System: software timestamp from the system clock.
- Transformed: hardware timestamp converted to software timestamp. This can be ignored because the adapter is using UTC time and transformation is not required. Transformed timestamps are identical to Raw timestamps.
- Raw: hardware timestamp generated by the adapter clock.

rx_timestamping Example

This demonstrates the `rx_timestamping` example application.

Server1

Server1 sets the EF_RX_TIMESTAMPING environment variable and starts the `rx_timestamping` application:

```
# EF_RX_TIMESTAMPING=2 onload ./rx_timestamping --proto tcp
oo:rx_timestamping[31250]: Using OpenOnload 201509 Copyright 2006-2015
Solarflare Communications, 2002-2005 Level 5 Networks [0]
Socket created, listening on port 9000
Socket accepted
Selecting hardware timestamping mode.
Packet 1 - 27 bytes     timestamps 1460374944.990960465
1460374944.993421129 1460374944.993421129
Packet 2 - 27 bytes     timestamps 1460374966.478980336
1460374966.481623531 1460374966.481623531
Packet 3 - 0 bytes      no timestamp
recvmsg returned 0 - end of stream
```

Server2

Server2 uses the Linux `netcat` utility to send packets to server1:

```
# nc <server1 ip> 9000
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
```

tx_timestamping Example

This demonstrates the `tx_timestamping` example application.

From Onload 8.1 onwards this example defaults to using Linux format timestamps for TCP. Use the `--stream` option to instead use the Onload proprietary format.

Server1

Server1 sets the EF_TX_TIMESTAMPING environment variable and starts the `tx_timestamping` application:

```
# EF_TX_TIMESTAMPING=3 onload ./tx_timestamping --proto tcp --ioctl eth4
oo:tx_timestamping[16139]: Using OpenOnload 201509 Copyright 2006-2015
Solarflare Communications, 2002-2005 Level 5 Networks [4]
TCP listening on port 9000
 TCP connection accepted
Accepted SIOCHWTSTAMP ioctl.
Selecting hardware timestamping mode.
Packet 1 - 27 bytes
Timestamp for 27 bytes:
First sent timestamp 1453436034.615029223
Last sent timestamp 0.000000000
```

Server2

Server2 uses the Linux `netcat` utility to send a packet to server1 which is then echoed back to the sender:

```
# nc <server1 ip> 9000
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz  (echoed back from server 1)
```

Example UDP Commands

This section gives an example of how the preceding commands can be modified to use UDP.:

Server1

```
# EF_RX_TIMESTAMPING=2 onload ./rx_timestamping --proto udp --port 9000
```

Server2

```
# nc -u <server1_ipaddr> 9000
```

Zeroed Timestamps

If timestamps returned from the example applications are zeroed, refer to Setting the Adapter Clock Time.