Chelsio Precision Time Protocol (PTP)

Introduction

Network Time Protocol (NTP) is the most widely used protocol for time synchronization today, but it provides accuracy only in milliseconds, making it an unfavorable protocol when high precision time synchronization is required.

Precision Time Protocol (PTP) standard defines a protocol for precise synchronization of clock between master and slave devices in a local area network. It can provide timing accuracies in nanosecond units. The protocol is based on time stamping and measuring the send and receive times. Most of the implementation relies on time stamping of the packets in the software which reduces the accuracy of the time measured. One possible solution to this problem is time stamping the packet in the NIC hardware itself.

The T5 hardware provides many features to support PTP implementations:

- High precision timers which can be read through PIO registers.
- Wall clock time based on the time of the day.
- Time stamping of selected PTP packets on both ingress and egress direction.

In the figure below, *M* denotes master and *S* denotes slave: the two different states of PTP protocol.

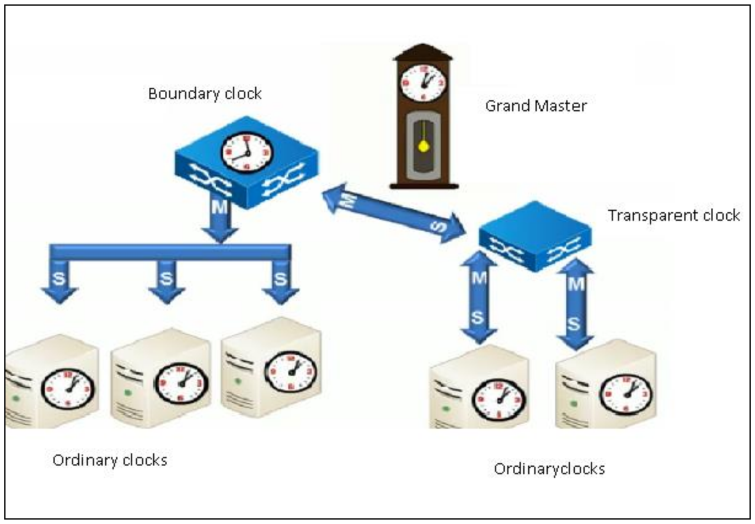

Figure 1 - PTP Protocol States

Grandmaster Clock

As the figure above shows, the Grandmaster Clock is the ultimate time source for the entire system. This clock implements the master PTP states. It determines the timescale (PTP or ARB) and related timescale properties of the domain.

Ordinary Clock

An ordinary clock has the following properties:

1. 1 PTP port, 1 copy of PTP protocol and 1 PTP state at any time.
2. The PTP port can be configured to be the Grandmaster or a slave clock in the master-slave hierarchy.
3. Each PTP port has an event interface for exchanging PTP event messages, and a general interface for exchanging PTP management messages. Only PTP event messages are time-stamped.
4. A local clock, either free running (Grandmaster state) or synchronized with the Grandmaster (slave state).
5. If the ordinary clock port is configured to be in the slave state, the PTP protocol engine on the ordinary clock adjusts the local clock to agree with the time of its master.
6. If the ordinary clock port is configured to be in the master state, the PTP protocol engine on the ordinary clock serves timing information to the system via PTP event messages.
7. Can implement either peer-delay mechanism or delay request-response mechanism.

Boundary Clock

A boundary clock helps to synchronize PTP clocks across different network LANs/subnets, which are bridged / routed by a bridge / router, on which boundary clock is resident. A boundary clock is the only way to interconnect peer-to-peer transparent clock in one region and end-to-end transparent clock in another region. A boundary clock that is also a bridge or router forwards all unicast PTP messages according to the forwarding rules of the network.

It has the following properties:

1. Typically has more than one communication port.
2. Each port is almost like the port of an ordinary clock.
3. One port will act as a client which will receive the time from the Grandmaster. The other ports can act as the master, providing precision time to clients in their respective LAN/subnet. PTP protocol engine determines which PTP port on the node acts as slave and provides the time signal to synchronize local clock.
4. Local physical clock used in boundary clock device is common to all its physical ports.
5. Messages related to synchronization (establishing master-slave hierarchy and signaling) terminate in the protocol engine of the boundary clock and are not forwarded. Management messages may be forwarded, subject to restrictions, to limit their propagation within the system.
6. Can implement either peer-delay mechanism or delay request-response mechanism

Transparent Clock

Transparent clocks forward all PTP event messages similar to boundary clocks. In addition, they make corrections in the timing information received from Grandmaster before relaying it. Since it is modifying the packet with a new value, the transparent clocks need to recalculate the checksum of the packet. There are 2 types:

1. **End-to-End Transparent Clock:** As the name implies, these clocks record information in PTP event messages that is relayed from Grandmaster all the way to an ordinary clock.

Only delay request-response mechanism is supported. It performs the following processing:

- **a. Residence Time Computation for PTP event messages:**
    - i. Ingress and egress time-stamps of the PTP event message are recorded for each PTP event message.
    - ii. Using the ingress and egress time-stamps of a PTP event message, the residence time bridge measures its residence time and accumulates it in the *correctionField* of that PTP event message.
- **b. Correction of drift in local clock, used for time-stamping purpose.**
  It is corrected using one of the following mechanisms:
    - i. Synchronization of local clock to master: make the rate of local clock equal to that of the master (by ratio calculation), before using it for time-stamping purposes. Clock rate adjustment in local node influences the adjustments at downstream nodes, which must account for residence time in upstream nodes for getting correct master time.
    - ii. Passing the difference in clock rates to downstream nodes: uncorrected local clock used to calculate and accumulate residence time in *correctionField*. Ratio of local clock rate to Grandmaster clock rate is also computed. Difference between the residence time corrected by this ratio and the uncorrected residence time must be accumulated and passed to the downstream slave clock.

2. **Peer-to-Peer Transparent Clock:** Very similar to end-to-end transparent clock but differs in the way PTP timing messages are handled:
    - **a. Link Delay Computation for each Rx port:** An additional per port functional block computes the link delay for *Sync* messages between each Rx port and a similarly equipped upstream port on another node sharing the link i.e. link peer. Peer delay mechanism is used for this purpose.
      The correction method for drift in local clock will be identical for both residence time and link delay computation of messages. The methods used for local clock drift correction are same as those in end-to-end transparent clock.
    - **b. Correction and forwarding of PTP event messages:** Performed only on *Sync* and *Follow_Up* messages. *CorrectionField* is updated both for residence time of *Sync* message within peer-to-peer Transparent clock and link delay on the port receiving the *Sync* message.

Chelsio T5 ASIC supports time stamping and classification of PTP packets. The following figure shows the general mode of operation. The PTP packets on the ingress or egress direction are identified by PTP time stamping and classification unit. The time stamped packets will be delivered either to the host or an embedded microprocessor within the T5. The highly accurate clock inside the T5 can be synchronized with an external clock source either from the host or from the microprocessor through clock management unit.

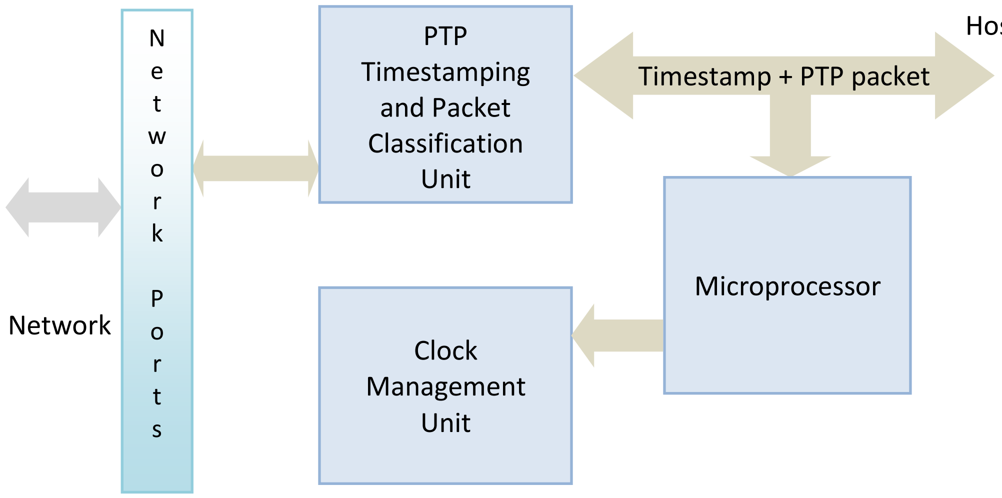

Figure 2 – Chelsio PTP

T5's PTP packet time stamping and clock management functionalities are exposed to the user mode PTP implementations through the NIC (*cxgb4*) driver. The user mode PTP implementations like *ptpd* and *linuxptp* can make use of this capability for hardware assisted time stamping and clock management operations.

In Linux, NIC driver exposes the time stamping capability through socket option *SO_TIMESTAMPING*. The clock management functions (PHC) are supported using the PHC infrastructure of the Linux.

Usage

ptp4l tool is used to synchronize the slave clock. Chelsio T5 can act as both master and slave.

General Installation Instructions

i. Enable the following in the kernel configuration file (*.config*), compile and reboot to the modified kernel:

```
# PTP clock support #
CONFIG_PTP_1588_CLOCK=y
```

ii. Install T5 adapter in the system.
iii. Download the latest Linux Unified Wire driver package from service.chelsio.com
iv. Untar the tar-ball and change your working directory to *ChelsioUwire-x.xx.x.x* directory
v. Install Chelsio Unified Wire drivers.

```
[root@host~]# make
[root@host~]# make install
```

vi. Load the Network driver

```
[root@host~]# modprobe cxgb4
```

The above command will also update the firmware if required. Use *ethtool -i <iface>* to verify the firmware version.

vii. Clean up iptable entries which may lead to dropped PTP frames.
viii. Choose the network port for PTP operations and connect the network cable accordingly.
ix. The ptp4l program tries to use hardware time stamping by default. To use ptp4l with hardware time stamping capable drivers and NICs, you must provide the network interface to use with the *-i* option. Run the following command:

```
[root@host~]# ptp4l -i ethX -m
```

x. To start the ptp4l program in slave mode use the *-s* option

```
[root@host~]# ptp4l -i ethX -m -s
```

To view the complete list of available options, refer the ptp4l help manual.