Solarflare Enhanced PTP User Guide

SF-109110-CD

Issue 5

Table of Contents

1 What's New

1.1 Overview

This document is the *User Guide* for **Solarflare Enhanced PTP** (`sfptpd`) which is an enhanced PTP daemon for use with Solarflare hardware timestamping network adapters.

This issue of the user guide supports `sfptpd` from version 3.2.1.

1.2 New features

Improvements to Statistical Data Format and Reporting

The improvements and enhancements in the 3.2.1 release provide for improved collating, formatting and reporting of PTP clock status, clock alarms and statistical data.

These new sfptpd features export real-time data in JSON format or PTP standards based format to a remote monitoring station. At the monitoring station both self-reported measurements and timestamp data can be compared to the PTP timesource.

Enhancements in the assembly, formatting and reporting of clock data to further the ability of Solarflare sfptpd to facilitate the business to meet the *business clock* regulatory requirements of (EU) MiFID II and (US) FINRA.

Machine Readable Real-Time Stats

Real-time statistical data can now be generated in JSON-lines format providing machine-readable stats logs. The `jsonl` formatted stats_log file can be parsed by a standard host-based or web browser JSON viewer application. An example JSON viewer HTML application is included with the sfptpd distribution.

For details of configuration and use, refer to Machine-Readable Real-Time Stats on page 97.

Machine Readable Long-Term Stats

Long-term statistical data files are now automatically generated in JSON format along with the standard text format files in the `/var/lib/sfptpd` directory. A JSON format file is created for every sync instance or local clock.

See Machine-Readable Long-Term Stats on page 99 for further information.

Standards Based, Application-Independent Clock Data

The draft revision of the IEEE-1588 specification (2017) proposes a mechanism for application-independent monitoring of the timing information from PTP slave clocks.

Solarflare sfptpd supports the draft standard and the sfptpd PTP slave will export timing information in PTP Signaling messages to a remote monitoring station.

Further information on this feature is available: Remote Reporting of Real-Time Stats on page 100

In addition to the specific TLV requirements of the draft IEEE-1588 specification, Solarflare sfptpd will generate a further organization extension TLV to provide additional reporting of alarms, state changes and other events.

Refer to Remote Reporting of Slave State on page 103 for further detail.

Meinberg NetSync Support

Using PTP timing messages with proprietary extensions, the Meinberg NetSync Monitor allows a remote Monitoring System to probe downstream PTP slave devices.

The NetSync monitor will collect timing synchronization data from multiple downstream slave devices and compare these to the Meinberg master clock timesource so users no longer have to rely on 'self-reported' clock accuracy from slave devices.

See Meinberg NetSync Monitor on page 104 for details.

Hot-Plug Support

With the adoption of the Linux Netlink socket-based IPC, sfptpd has improved robustness and availability over link state change events and adapter/interface changes when sfptpd is used over Linux bonded/teamed interfaces.

For more detail refer to Link States, Interface Hotswap on page 53.

Non-Solarflare Adapter Support

The sfptpd daemon is now able to recover hardware timestamps for PTP packets delivered by third party (non-Solarflare) adapters that support the standard PTP Hardware Clock (PHC) API.

Refer to Non-Solarflare Adapters on page 6 for more details.

Enhanced the Automatic Selection policy

The automatic selection criteria which determines the active sync instance can be configured to suit specific network or system requirements.

Refer to Change the Automatic Selection Order on page 21 for details.

Documentation changes

This issue of the user guide includes a TroubleShooting guide to assist users with common PTP issues.

2 Introduction

This document describes Solarflare Enhanced PTP (`sfptpd`) support for Solarflare's 10GbE SFP+ and QSFP+ 40GbE Time Synchronization Server Adapters. These adapters support hardware time stamps of PTP packets, and can be deployed in networks where there is a requirement to support the IEEE 1588 Precision Time Protocol.

The Solarflare sfptpd daemon is an implementation of the IEEE 1588-2008 Precision Time Protocol version 2.

2.1 Definitions, acronyms and abbreviations

1PPS	1 Pulse Per Second
LACP	Link Aggregation Control Protocol, part of IEEE-802.3ad
LRC	Local Reference Clock. The active clock to which all other clocks, on a PTP enabled server, are synchronized
MTIE	Maximum Time Interval Error
NSM	(Meinberg) NetSync Monitor
NTP	Network Time Protocol
PHC	PTP Hardware Clock
PID	Proportional, Integral, Differential filter
PPB	Parts Per Billion
PPM	Parts Per Million
PTP	Precision Time Protocol
ptpd2	An implementation of IEEE-1588-2008 (PTP version 2)
sfptpd	Solarflare Enhanced PTP daemon An implementation of IEEE-1588-2008 (PTP version 2)
TLV	Type, Length, Value - an element of a protocol or signaling message
UUID	Universally Unique Identifier
VLAN	Virtual Local Area Network

2.2 Features

Solarflare Enhanced PTP (`sfptpd`) supports the following features:

- Hardware timestamps for received and transmitted PTP packets.
- The ability to synchronize the high precision clock on multiple adapters.

  One of the adapter clocks is selected as the Local Reference Clock which can then synchronize the server system clock and clocks on other adapters.
- PTP clock roles:
  - PTP slave server
  - PTP master server
  - PTP boundary clock.
- PTP hybrid mode.

  Combines UDP multicast transmission for Announce and Sync messages with unicast transmission for Delay_Request and Delay_Response messages to optimize use of network resources.
- PTP Synchronization mode.
- PPS Synchronization mode.
- NTP fallback mode.
- PTP multiple masters support
- Filtering of outlier values received from master clock sources.
- PTP over VLAN interfaces.
- PTP over active/backup and LACP bonded interfaces.Remote monitoring of PTP real-time and long-term statistical data.
- Standards based, application-independent reporting of clock data.
- Remote monitoring/reporting of PTP alarm states and events.
- PTP using third party adapters.

2.3 Transport

Solarflare sfptpd operates PTP over UDP/IPv4 using the default end-to-end or peer-to-peer profiles.

PTP over layer 2 (Ethernet) is not supported.

2.4 Supported adapters

Solarflare Adapters

- Any Solarflare XtremeScale™ SFN8000 series adapter with a Plus AppFlex™ license installed:
  - this license is pre-installed on all SFN8*nnn*-Plus adapters
  - a PTP license can be added to other SFN8000 series adapters.
- Any Solarflare Flareon™ SFN7000 series adapter with a PTP/timestamping AppFlex™ license installed:
  - this license is pre-installed on the Solarflare Flareon Ultra SFN7322F adapter
  - a PTP license can be added to other SFN7000 series adapters.
- HP 570FLB FlexibleLOM and HP 570M Mezzanine adapters with a PTP/ timestamping AppFlex license installed.
- Solarflare SFN6322F dual-port 10GbE SFP+ server adapter - no additional license required.

Non-Solarflare Adapters

From version 3.2.1 sfptpd can recover hardware timestamps for PTP packets from third party adapters. sfptpd will also discipline the clocks on third party adapters that support the PHC API.

In earlier sfptpd versions, only software timestamping was possible from third party adapters.

This is an experimental feature where testing has shown that synchronization accuracy/performance in mixed adapter systems is best on Solarflare adapters.

Solarflare have conducted limited testing with a range of third party adapters - the list of tested models can be found in the release notes.

NOTE: Implementation of the PHC API is not uniform across all non-Solarflare adapters, therefore this is a best effort feature.

Timestamping ports

Solarflare PTP network adapters support hardware timestamping:

- Solarflare XtremeScale™ SFN8000:
  - Hardware timestamping of all transmitted and received packets on all adapter ports.
- Solarflare Flareon™ SFN7000:
  - Hardware timestamping of all received packets on all adapter ports.
  - The SFN7000 series adapter, when used with OpenOnload® or EnterpriseOnload®, can support hardware timestamping of transmitted packets. For details refer to the Onload User Guide (SF-104474-CD).
- HP branded adapters:
  - Hardware timestamping of all received packets on all adapter ports.
- On SFN6322F: Hardware time stamping is limited to PTP packets and only on a single port of the network adapter which is the port closest to the PCIe connector.

The adapter timestamp function is compliant with the IEEE 1588-2008 (PTP version 2) specifications.

3 Hardware and Software Support

3.1 Supported Linux kernel versions

Solarflare Enhanced PTP is supported on the following OS/kernels versions:

Table 1: OS/Kernel Support

OS Version	Notes
Red Hat Enterprise Linux 6.4 - 6.8, 7.1 - 7.4	RHEL6 built-in Solarflare drivers may not support SFN7000 and SFN8000 series adapters.
Red Hat Messaging Realtime and Grid 2.4, 2.5	
SuSE Linux Enterprise Server 11 sp2, sp3, sp4	Built-in Solarflare drivers may not support SFN7000 and SFN8000 series adapters.
SuSE Linux Enterprise Realtime Extension 11	
SuSE Linux Enterprise Server 12 sp1, sp2	
Linux kernels 2.6.32 - 4.11	
Canonical Ubuntu Server LTS 14.04, 16.04	
Canonical Ubuntu Server 16.10, 17.04	
Debian 7 “Wheezy”	
Debian 8 “Jessie”	
Debian 9 “Stretch”	

3.2 Solarflare PTP network adapters

Solarflare time synchronization adapters generate hardware timestamps for PTP packets in support of a network precision time protocol deployment, and in accordance with the IEEE 1588-2008 specifications. With hardware precision and performance, the PTP adapters facilitate PTP slave servers to accurately synchronize internal clocks to multiple network master clocks, or to serve as either a master clock or a boundary clock.

These adapters contain a dedicated time stamping unit which is driven from a high precision oscillator. Receipt or transmission of a PTP formatted packet triggers the generation of an accurate hardware timestamp which is passed by the adapter to the network device driver. The adapter also enables a PTP stack running on the host server to discipline the adapter's precision oscillator (both absolute time and clock rate).

The PTP stack running on the host server will synchronize the adapter clock and host real time clocks to a remote time source.

XtremeScale™ SFN8000 series dual-port 10GbE SFP+ and 40GbE QSFP+ adapters

The SFN8000 series adapters combine ultra low latency with precision time synchronization and hardware timestamping of all received network packets on either physical port of the adapter.

Ports

PTP packets can be received on any adapter port and ports can be configured in an active/backup failover configuration. LACP bonding is also supported. Hardware timestamping is done on all ports.

License

SFN8*nnn*-Plus adapters, such as the SFN8522-Plus or the SFN8542-Plus, are supplied as a factory-ready PTP adapter. No additional license is required.

Other SFN8000 series adapters can be upgraded with a Solarflare AppFlex™ Technology license to support PTP and hardware timestamping of all received packets.

For more details of the AppFlex Technology licensing refer to the *Solarflare Server Adapter User Guide* (SF-103837-CD).

Flareon™ SFN7000 series dual-port 10GbE SFP+ and 40GbE QSFP+ adapters

The SFN7000 series adapters combine ultra low latency with precision time synchronization and hardware timestamping of all received network packets on either physical port of the adapter.

Ports

PTP packets can be received on any adapter port and ports can be configured in an active/backup failover configuration. LACP bonding is supported. Hardware timestamping is done on all ports.

License

The SFN7322F adapter is supplied as a factory-ready PTP adapter. No additional license is required.

Other SFN7000 series adapters can be upgraded with a Solarflare AppFlex™ Technology license to support PTP and hardware timestamping of all received packets.

For more details of the AppFlex Technology licensing refer to the *Solarflare Server Adapter User Guide* (SF-103837-CD).

PPS Support

A 1PPS bracket kit and cable assembly providing PPS input/output connections can be fitted to the SFN8000 or SFN7000 series adapters:

- 10GbE SFP+ adapters require the SOLR-PPS-DP10G bracket kit
- 40GbE QSFP+ adapters require the SOLR-PPS-DP40G bracket kit.

Customers interested in the optional PPS kit should contact their Solarflare sales channel.

HP-branded server adapters

The HP570FLB and HP570M dual-port 10G Ethernet adapters are server adapters for the HP c-Class ProLiant Gen8 BladeSystem. These adapters are functionally equivalent to the Solarflare SFN7122F adapter and can support hardware timestamping on either port when the PTP/HW timestamping license is installed.

SFN6322F dual-port 10GbE SFP+ adapter

The SFN6322F supports hardware timestamping of PTP packets. Hardware timestamping of non-PTP network traffic is not supported. The SFN6322F features a 1PPS input that can be used to calibrate the PTP offset and an extremely accurate 1PPS output timing signal aligned to the adapter's Stratum 3 clock. The SFN6322F combines precision time synchronization with ultra-low latency 10G Ethernet.

Connect the SFN6322F timestamping port

Hardware time stamping is only functional on a single port of the adapter which is the port with the lowest MAC address, that is closest to the PCIe connector.

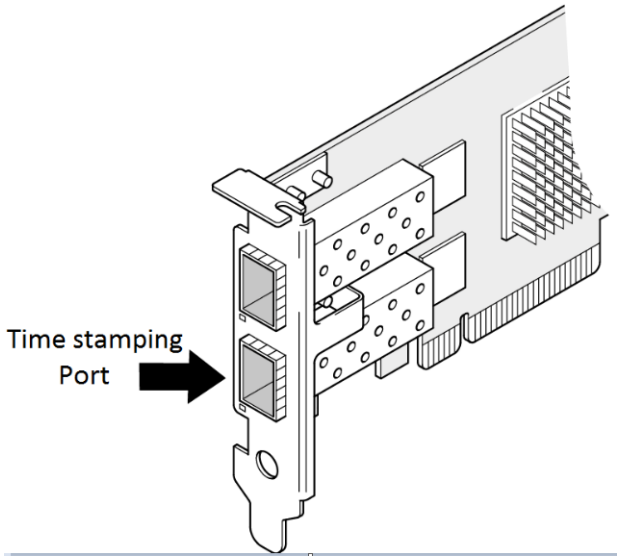


Figure 1: Identify SFN6322F timestamping port

Use ethtool to identify the hardware timestamping port:

```
ethtool -i eth<N>
driver: sfc
version: <driver version>
firmware-version: <firmware version>
bus-info: 0000:07:00.0
```

From the PCI `bus-info` a zero function value (last digit) identifies the timestamping port on the SFN6322F network adapter.

It can be useful to use ethtool to identify the timestamping port on the back panel by 'blinking' the port LED for a specified number of seconds:

```
ethtool -p eth<N> 10
```

3.3 Time synchronization features

Solarflare PTP network adapters have the following time synchronization features:

- Ability to maintain synchronization of the system clock typically within 200ns offset from a network master clock. The accuracy obtained is dependent on the correct operation of the upstream timesource, however, the PTP slave adapter clock can be within 50ns offset from the PTP master.
- Stratum 3 compliant oscillator; Oscillator drift 0.37 PPM per day (c. 32ms/day); oscillator accuracy < 4.6PPM over 20 years.
- Ability to capture a hardware timestamp as selected frames enter/leave the Ethernet MAC. Time stamping for packets formatted according to IEEE 1588-2008 (PTP version 2).
- Hardware timestamps exposed to Linux via the standard SO_TIMESTAMPING socket API on kernels 2.6.30 later.
- Ability to discipline the network adapter's high precision oscillator in response to PTP timing information.
- Support PTP packets over bonded interfaces in an active/standby configuration or LACP bonds.
- Support PTP packets over 802.1Q VLAN interfaces.

3.4 Adapter Driver/Firmware Support

This section identifies the minimum software components required to support time synchronization server adapters.

To identify Solarflare adapters in a server, run the following command:

```
# lspci -vvv -d 1924:
```

The following command typically extracts the adapter name from the above output:

```
# lspci -vvv -d 1924: | egrep 'Product|Subsystem'
```

To identify the Solarflare net driver and firmware versions used by the Solarflare adapter run the following command where N is the Solarflare interface:

```
# ethtool -i eth<N> .
```

Adapter	Linux net driver	Controller firmware
SFN8542	Any	Any
SFN8522	Any	Any
SFN7142Q	v4.1.0.6734	v4.1.1.1022

Adapter	Linux net driver	Controller firmware
SFN7322F	v4.0.2.6628	v4.0.6.6689
SFN7122F		
SFN7002F		
HP 570FLB	v4.0.2.6628	v4.0.7.6711
HP 570M		
SFN6322F	v3.3.0.6246	v3.3.0.6247

OS ‘in-tree’ Driver

The ‘in-tree’ driver included with many Linux distributions does not support PTP features required by sfptpd. To identify if the adapter is using an ‘in-tree’ driver run the following ethtool command:

```
# ethtool -i <interface>
version: 4.0 *
firmware-version: 3.3.0.6298
```

* The ‘in-tree’ driver is version 4.0 or 4.1.

CAUTION: The `sfptpd` daemon must be terminated before upgrading the adapter firmware.

Network driver

The adapter driver is distributed as a standalone RPM (source and DKMS) or with the OpenOnload/EnterpriseOnload distributions (listed below).

- OpenOnload from version 201310-u1.
- EnterpriseOnload from version 3.0.0.2.

The network driver exposes a number of features of the adapter including:

- Hardware time stamping of PTP formatted packets.

  Starting in Linux kernels 2.6.30, support for hardware time stamping of network packets on TX and RX is formalized and integrated via the socket option SO_TIMESTAMPING. Details of this interface can be found at http://lxr.linux.no/linux/Documentation/networking/timestamping.txt. Before an application can receive timestamps on a socket it must first issue the IOCTL SIOCSHWTSTAMP to register both the types of packets it wants to receive timestamps on and the format of timestamps it wishes to receive. SIOCSHWTSTAMP is not required by applications using Onload.

  - SFN8000 series adapters can hardware time stamp all transmitted and received packets.
  - SFN7000 series adapters can hardware time stamp all received packets.

  - HP-branded adapters can hardware time stamp all received packets.
  - The SFN6322F hardware time stamping is limited to PTP formatted packets.

- Access to the control of the precision oscillator.

  The adapters contain a precision clock with drift rated to < 1 PPM per year. The driver allows the absolute time and frequency of this clock to be controlled via the Linux PHC subsystem or the proprietary IOCTL interface. The proprietary interface is used by the Solarflare supplied `sfptpd` stack to run the PTP protocol using the precision oscillator on the adapter.

3.5 Synchronization model

The `sfptpd` Solarflare Enhanced PTP daemon synchronizes local clocks, including the system clock, to a single time source:

- `sfptpd` can process multiple time sources including:
  - one or more remote PTP master clocks
  - a 1PPS signal, with an external NTP server providing time of day
  - an external NTP server
  - a local Solarflare adapter clock.

  Each time source is provided by an instance of a *sync module*. For more information, see Sync modules on page 15.

- `sfptpd` selects a time source to use.

  By default, `sfptpd` automatically selects the best performing time source. For more details, see Sync module selection on page 20.

- `sfptpd` synchronizes a Local Reference Clock (LRC) to the selected time source.

  `sfptpd` designates one local clock as the Local Reference Clock (LRC), and synchronizes this to the selected external time source.

  The LRC used will depend on the sync module selected as the time source, but the LRC is typically the precision clock on a Solarflare PTP adapter. See the descriptions of sync modules in Sync modules on page 15.

- `sfptpd` synchronizes each clock, including the system clock, to the LRC.

  `sfptpd` runs a clock servo for each additional clock in a server i.e. for each additional timestamping adapter and for the system clock. `sfptpd` will measure the difference between each clock and the LRC, and synchronize all clocks to the LRC.

Figure 2 illustrates the synchronization options supported by `sfptpd`.

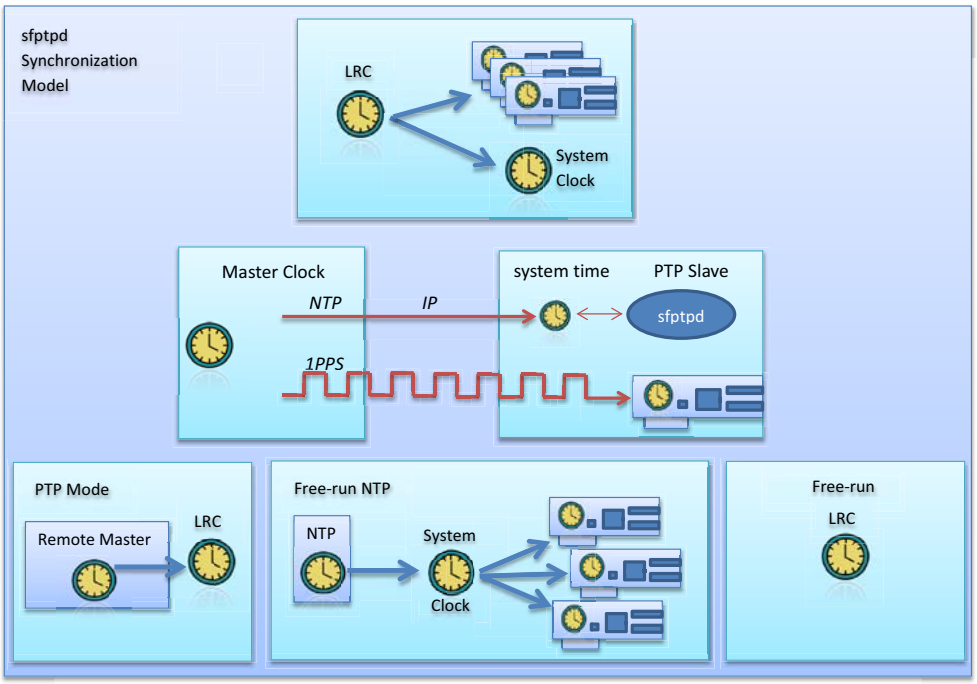


Figure 2: sfptpd synchronization model

3.6 Sync modules

The following sections provide an overview of the available sync modules:

- PTP sync module on page 16
- PPS sync module on page 17
- NTP sync module on page 17
- Freerun sync module on page 19.

Each instance of a sync module represents a candidate time source to `sfptpd`. From the group of candidates, one instance is selected as the active time source. This process is described in Sync module selection on page 20.

PTP sync module

Using a PTP sync module as the time source, sfptpd will synchronize the adapter clock and system clock to a remote PTP master clock:

- When the PTP sync module uses an interface on a Solarflare PTP-enabled adapter, the adapter clock is selected as the LRC.

  `sfptpd` synchronizes the LRC to the PTP master, and keeps all other clocks (including the system clock) synchronized with the LRC.

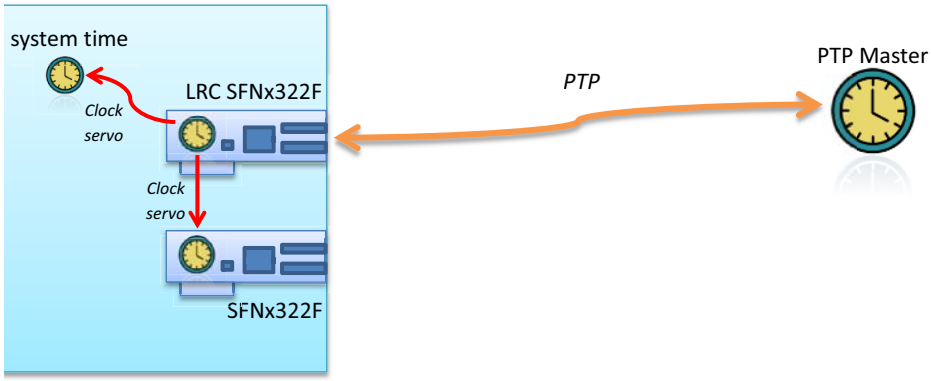


Figure 3: PTP sync module

- When the PTP sync module uses an interface on a non-PTP enabled adapter, sfptpd will discipline the host system clock time and only software timestamping is available.
- Using sfptpd-3.2.1, a third party adapter supporting the PHC API can also be selected as the LRC. Refer to release notes for a list of viable adapters.

Refer to PTP Sync Module on page 51 for more details.

PPS sync module

When `sfptpd` selects a PPS sync module as the time source, it periodically receives time-of-day from an external NTP server, and then maintains clock synchronization using a received 1PPS signal:

- If the PPS sync module is using an interface on a Solarflare PTP-enabled adapter, the adapter clock is selected as the LRC.

  `sfptpd` periodically polls an NTP client for time-of-day. Thereafter `sfptpd` synchronizes the LRC to the 1PPS pulse, and keeps all other clocks (including the system clock) synchronized with the LRC.

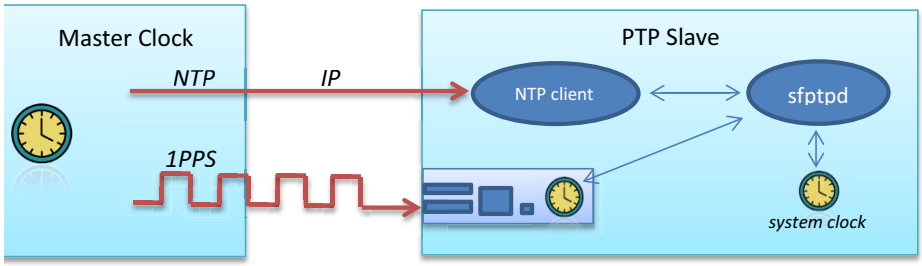


Figure 4: PPS sync module

The NTP client is used to provide the time-of-day to the slave system clock.

NOTE: This sync module does not use PTP, and so does not send or process any PTP messages.

For more details see PPS Sync Module on page 70.

NTP sync module

When `sfptpd` selects an NTP sync module as the time source, it synchronizes the adapter and system clocks to an external NTP server.

The system clock is selected as the LRC.

An local NTP client is used to synchronize the system clock with an external NTP server. `sfptpd` keeps all other clocks synchronized with the system clock, so hardware timestamps (of non-PTP packets) received from the adapter can be compared with system time.

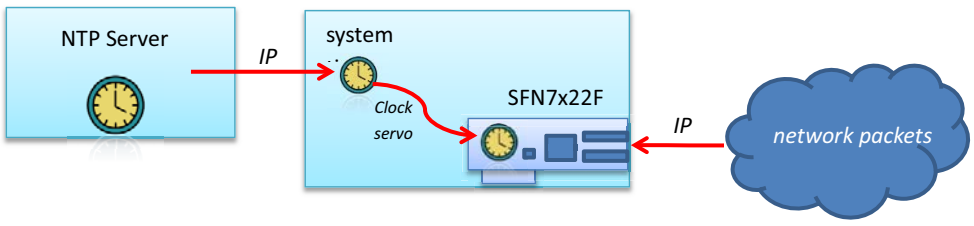


Figure 5: NTP sync module

An accurate and stable NTP server should be selected when using this mode.

NOTE: This sync module does not use PTP, and so does not send or process any PTP messages.

For more details see NTP Sync Module on page 73.

Freerun sync module

When `sfptpd` selects a Freerun sync module as the time source, it synchronizes the adapter and system clocks to a local Solarflare adapter clock.

The precision clock on a Solarflare PTP adapter is selected as the LRC.

At startup, `sfptpd` sets the time of the LRC from the system clock. Thereafter the LRC runs free, and `sfptpd` keeps all other clocks (including the system clock) synchronized with the LRC.

The system is **not** being synchronized to a remote time source.

This sync module can be useful as a backup or failover, for use when other more reliable time sources become unavailable.

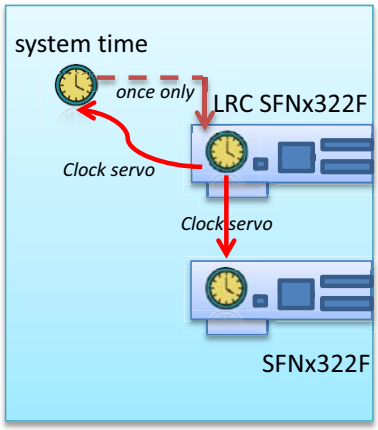


Figure 6: sync_mode freerun with freerun_mode nic

NOTE: This sync module does not use PTP, and so does not send or process any PTP messages.

For more details of the Freerun sync module, see Freerun Sync Module on page 77.

3.7 Sync module selection

This section describes how a single sync module instance is selected as the time source. By default selection is automatic, Manual selection is also an option.

Automatic selection

By default, sfptpd automatically selects the time source to use.

The set of all configured sync instances represents a list of candidates available for selection.

A sync instance in the SLAVE state is considered before another instance in the LISTENING state.

A candidate is automatically selected using the following criteria, in descending order of priority:

1 **State**

A sync instance in the SLAVE state is considered before another in the LISTENING state.

2 **No Alarms**

A sync instance having no PTP alarms present is considered before others where alarms are active.

3 **Instance priority**

The value of the `priority` configuration option for the instance. Smaller values have higher priority. By default, all instances have priority 128, and so this criterion is ignored

NOTE: This is not related to the IEEE-1588 specification priority 1 and priority 2 properties of a PTP master clock.

4 **Clock class**

Locked clocks are considered ahead those which are in holdover or are free-running.

5 **Accuracy**

This is the sum of:

a) **Clock accuracy**

Estimate of error between clock and primary reference source.

b) **Sync module accuracy**

Estimate of error implied by the synchronization mechanism that would be used to sync to the remote clock, such as NTP with software timestamping vs. PTP with hardware timestamping.

6 **Allan variance**

Estimate of stability of clock.

7 **Identity of clock**

If all other comparisons fail, the identity of the clock is used as a tie-breaker.

The source of the clock class and accuracy varies:

- For the PTP sync module, the clock class and clock accuracy are values conveyed in PTP messages from the upstream PTP master clock.
- For the PPS sync module, the clock class and clock accuracy have default values but can also be set by the user.
- For the NTP sync module, the clock class and clock accuracy come from the remote clock.
- For the Freerun sync module, the clock class and clock accuracy use default values.

Change the Automatic Selection Order

The order of selection criteria can be changed to suit a specific network or system requirement.

To change the default ordering of selection criteria, the user should change the order of criteria as they appear with the following configuration file option:

```
# Specify an alternative ordering of rules for the selection policy.
selection_policy_rules manual state no-alarms user-priority clock-class
total-accuracy allan-variance steps-removed
```

See the supplied `/config/default.cfg` file for details.

This feature allows a hybrid manual-automatic selection priority when the ‘manual’ option is moved to a different position.

Manual selection

To use manual selection, set the `selection_policy` configuration option to `manual`, with a second parameter giving the name of the initial sync module instance to use. See Configuration options on page 33.

Use the `sfptpdctl` utility program to change the instance selected as the master. This allows the user to switch timesources simultaneously across an entire network, thus ensuring that all servers use a common timesource. See Daemon control mechanism on page 83.

3.8 Sync Module Accuracy

The following table identifies the notional accuracy value for each sync_module type.

Sync_module	Accuracy
Freerun	0
NTP	10 us
PPS	50 ns
PTP_HW	50 ns
PTP_SW	50 ms

4 Getting Started

Procedure to install components and run sfptpd.

4.1 Installation

Follow steps 1-6 to install the Solarflare Adapter, check driver and firmware versions and check that required licenses are loaded.

Requirements

- A Solarflare PTP network adapter, one of the following:
  - SFN8000 series network server adapter
  - SFN7000 series network server adapter
  - HP570FLB FlexibleLOM or HP570M Mezzanine adapter
  - SFN6322F adapter.
- Solarflare Enhanced PTP daemon.

Step 1: Verify Solarflare adapter driver and firmware versions

To check the Solarflare adapter driver and firmware versions use the Linux `ethtool` command e.g.

```
# ethtool -i eth<N>
```

NOTE: The in-tree Solarflare drivers that are supplied with Linux distributions do not support PTP. The in-tree driver will be version 4.0 or 4.1, for example, this is an in-tree driver that does not support PTP:

```
# ethtool -i eth<N>
sfc 4.0
```

Refer to Chapter 3 on page 8 for driver and firmware information.

Step 2: Install the network adapter

Complete instructions for the deployment and installation of the network adapter can be found in the *Solarflare Server Adapter User Guide* (SF-103837-CD).

Step 3: Install the network driver

The network adapter driver is available from https://support.solarflare.com/ in the following formats:

- DKMS (part number SF-104979-LS)
- source (part number SF-103848-LS).
- The OpenOnload® and EnterpriseOnload® distributions also include a version of the adapter driver.

Complete instructions for installing and loading the driver can be found in the *Solarflare Server Adapter User Guide* (SF-103837-CD).

Step 4: Update the adapter firmware

The adapter firmware only needs to be updated if it does not meet the minimum firmware version required to support PTP and HW timestamping.

Refer to Chapter 3 on page 8 for driver and firmware information.

CAUTION: The `sfptpd` daemon must be terminated before upgrading the adapter firmware. Following firmware upgrade the adapter driver should be reloaded before starting sfptpd.

NOTE: The Solarflare Utilities RPM for Linux contains the Solarflare Boot Manager (sfboot), a flash firmware update utility (sfupdate) and an AppFlex license upgrade utility (sfkey).

The RPM package is available as a 32bit binary and 64bit binary:
- SF-105095-LS is a 32bit binary
- SF-107601-LS is a 64bit binary

1. Download the sfutilities package from https://support.solarflare.com/.
2. Unzip the file to reveal the binary RPM
3. Install the RPM e.g.

   ```
   # rpm -Uvh sfutils-<version>.rpm
   ```

4. Identify the current firmware version on the adapter.

   ```
   # sfupdate
   ```

5. Replace the adapter firmware with the version in this sfupdate.

   ```
   # sfupdate --write
   ```

Full instructions on using `sfupdate`, `sfboot` and `sfkey` can be found in the Solarflare Network Adapter User Guide (SF-103837-CD).

Step 5: Ensure licenses are installed

An adapter must have an AppFlex PTP license installed before it can be used for PTP. The license is pre-installed on certain adapters. Use the sfkey utility to list licenses, and to install a license if necessary.

1 Display the licensing ID and installed license keys for all adapters:

```
# sfkey --all --report
eth2,eth3: 712200201082133128200003 (Flareon)
  Product name          Solarflare SFN7122F SFP+ Server Adapter
  Installed keys        Onload, SolarCapture Pro
```

2 If the `Installed keys` do not include a `PTP` or a `Plus` license key, install a PTP license:

Copy the license key data to a `.txt` file on the target server. All keys can be in the same key file and the file applied on multiple servers. The following example uses a license key file called `keys.txt` created on the local server.

```
# sfkey --adapter=eth2 --install keys.txt
Reading keys...

Writing all keys to eth2...
eth2: 712200201082133128200003 (Flareon)
  Product name          Solarflare SFN7122F SFP+ Server Adapter
  Installed keys        Onload, PTP, SolarCapture Pro
```

Step 6: Identify the timestamping port

Connect the SFN8000 series timestamping port

The Solarflare 8000 series adapters support hardware timestamping of all packets on either port of the adapter.

Ports on the same adapter, or from different adapters in the same server, can be bonded in an LACP or active/standby failover configuration.

Connect the SFN7000 series timestamping port

The Solarflare 7000 series adapters support hardware timestamping of all packets on either port of the adapter.

Ports on the same adapter, or from different adapters in the same server, can be bonded in an LACP or active/standby failover configuration.

Connect the SFN6322F timestamping port

Refer to SFN6322F dual-port 10GbE SFP+ adapter on page 11 for details.

4.2 Download and install sfptpd

The Solarflare Enhanced PTP, `sfptpd`, is a PTP daemon adapted by Solarflare to work with Solarflare time synchronization server adapters. The `sfptpd` daemon is an implementation of IEEE-1588-2008 (PTP version 2).

NOTE: PTP Version 2 is not compatible with PTP Version 1. They use different message sizes and format. Version 1 messages received by sfptpd will be ignored.

The `sfptpd` package is available packaged as a tarball or as an RPM, in 32bit or 64bit binary formats:

- SF-108909-LS is a 32bit binary tarball
- SF-108910-LS is a 64bit binary tarball
- SF-113121-LS is a 32bit binary RPM
- SF-113122-LS is a 64bit binary RPM

Download the required `sfptpd` package from https://support.solarflare.com/.

Install tarball

To install using a tarball package:

1 Unpack the compressed file:

```
# tar -zxvf SF-108910-LS-16.tgz
```

```
sfptpd-3.2.1.1004.x86_64/
sfptpd-3.2.1.1004.x86_64/sfptpd
sfptpd-3.2.1.1004.x86_64/examples/
sfptpd-3.2.1.1004.x86_64/examples/README.sfptpdctl
sfptpd-3.2.1.1004.x86_64/examples/monitoring_console.py
sfptpd-3.2.1.1004.x86_64/examples/sfptpdctl.c
sfptpd-3.2.1.1004.x86_64/examples/sfptpdctl.py
sfptpd-3.2.1.1004.x86_64/examples/sfptpd_json_parse.html
sfptpd-3.2.1.1004.x86_64/examples/Makefile.sfptpdctl
sfptpd-3.2.1.1004.x86_64/examples/sfptpd_stats_collectd.py
sfptpd-3.2.1.1004.x86_64/PTPD2_COPYRIGHT
sfptpd-3.2.1.1004.x86_64/NTP_COPYRIGHT.html
sfptpd-3.2.1.1004.x86_64/init.d/
sfptpd-3.2.1.1004.x86_64/init.d/sfptpd
sfptpd-3.2.1.1004.x86_64/sfptpdctl
sfptpd-3.2.1.1004.x86_64/LICENSE
sfptpd-3.2.1.1004.x86_64/config/
sfptpd-3.2.1.1004.x86_64/config/ptp_master_ntp.cfg
sfptpd-3.2.1.1004.x86_64/config/ptp_slave_multiple.cfg
sfptpd-3.2.1.1004.x86_64/config/freerun.cfg
sfptpd-3.2.1.1004.x86_64/config/ptp_slave_ntp_fallback.cfg
sfptpd-3.2.1.1004.x86_64/config/ptp_slave.cfg
sfptpd-3.2.1.1004.x86_64/config/ntp.cfg
sfptpd-3.2.1.1004.x86_64/config/ptp_domain_bridge.cfg
sfptpd-3.2.1.1004.x86_64/config/many_instances.cfg
sfptpd-3.2.1.1004.x86_64/config/pps_slave.cfg
sfptpd-3.2.1.1004.x86_64/config/ptp_master_freerun.cfg
```

```
sfptpd-3.2.1.1004.x86_64/config/ptp_boundary.cfg
sfptpd-3.2.1.1004.x86_64/config/default.cfg
```

2 Startup components:

- An `init.d` script is supplied to allow `sfptpd` to be stared using standard Linux service: start, stop, restart, status commands.
- An example systemd UNIT file can be found in Appendix B on page 110.
- Default configuration files for all modes are located in the `config` subdirectory of the install directory.

Install RPM

To install using an RPM package:

1 Unpack the RPM package:

```
tar -zxvf SF-113122-LS-
<version>_Solarflare_Enhanced_PTP_Daemon_sfptpd_-
_64_bit_binary_RPM.tgz
```

2 Install the binary RPM:

```
rpm -ivh sfptpd-<version>.x86_64.rpm
Preparing...
########################################### [100%]
   1:sfptpd
########################################### [100%]
```

3 Startup Components:

- The `sfptpd` executable will be installed into root's path.
- Default configuration files for all modes are located in the `/usr/share/doc/packages/sfptpd/config` directory.
- Example software is located in the `/usr/share/doc/packages/sfptpd/examples` directory.

4.3 Pre-Start Checks

NTP

The NTP service should only be running when using an sfptpd mode that requires NTP as a timesource or fallback timesource e.g. NTP fallback mode or NTP/PPS mode.

If NTP is not required by the sfptpd mode, the service should be terminated before starting the sfptpd daemon. This is to prevent NTP from changing the system clock. Sfptpd will synchronize the system clock.

```
# service ntpd stop
```

```
# systemctl status stop
```


On startup sfptpd stats_log, message_log output will warn the user if the NTP service is running.

IPTables

Users must ensure that no rule exists in iptables that will prevent PTP packets from reaching the slave `sfptpd` process. The following command is a useful starting point to examine iptables rules.

```
# iptables -L
```

The iptables service can be temporarily disabled to identify if it is responsible for blocking PTP traffic.

```
# service iptables stop
```

PTP Domain

A PTP slave must be in the same PTP Domain as the upstream master clock.

PTP messages received by sfptpd that are not from the configured domain will be ignored. The default domain value in sfptpd config files is 0 (zero).

The domain value should be configured to match the PTP domain of the upstream master clock. Configure the domain with the `ptp_domain` parameter in the sfptpd config file.

4.4 Run sfptpd

To run `sfptpd`:

1. Ensure that the timestamping port of the adapter is configured with an IP address suitable for the network configuration.
2. Perform Pre-Start Checks on page 27 - especially the PTP domain.
3. Start `sfptpd` using one of the supplied configuration files.
   - Run sfptpd as a PTP slave on page 29
   - Run sfptpd in freerun mode on page 29
   - Run sfptpd in NTP/PPS mode on page 29.

NOTE: `sfptpd` can take 15-30 minutes to initially stabilize the times on the slave machines.

Run sfptpd as a PTP slave

As a PTP slave, sfptpd is receiving and sending PTP messages to an upstream PTP master clock.

For this mode, use the default `ptp_slave.cfg` file:

- from tarball package:

```
cd <install_dir>
./sfptpd -i <interface> -f config/ptp_slave.cfg
```

- from RPM package:

```
cd /usr/share/doc/packages/sfptpd/
sfptpd -i <interface> -f config/ptp_slave.cfg
```

where `<interface>` is the identifier of the timestamping port on the adapter.

Run sfptpd in freerun mode

In Freerun mode, sfptpd will synchronize all clocks in the server, including the system clock, to the high precision oscillator on the selected PTP adapter.

For this mode, use the default `freerun.cfg` file:

- from the tarball package:

```
cd <install_dir>
./sfptpd -i <interface> -f config/freerun.cfg
```

- from the binary RPM package:

```
cd /usr/share/doc/packages/sfptpd/
sfptpd -i <interface> -f config/freerun.cfg
```

where `<interface>` is the identifier of the timestamping port on the adapter.

Run sfptpd in NTP/PPS mode

In NTP/PPS mode, sfptpd will get time-of-day from NTP, the use the 1PPS signal to closely synchronize the adapter clock. All other clocks, including the system clock, are synchronized to the adapter clock.

NOTE: It is a requirement to setup NTP symmetric authentication to allow sfptpd to control the local NTP client.

NOTE: Only a single NTP sync_module is required. Do not create multiple NTP sync modules.

For this mode, use the default `pps_slave.cfg` file:

- from the tarball package:

```
cd <install_dir>
./sfptpd -i <interface> -f config/pps_slave.cfg
```

- from the binary RPM package:

```
cd /usr/share/doc/packages/sfptpd/
sfptpd -i <interface> -f config/pps_slave.cfg
```

where `<interface>` is the identifier of the timestamping port on the adapter.

5 User Interface

5.1 Command line options

The configuration settings for `sfptpd` are enabled, set or disabled in a configuration file.

The path to the configuration file is specified on the `sfptpd` command line. For example, to use the default `ptp_slave.cfg` file in the `config` sub-directory:

```
./sfptpd -i eth<N> -f config/ptp_slave.cfg
```

`sfptpd` also supports a limited number of command line options which override the equivalent config file options. Use the `sfptpd -h` command to identify supported command line options.

5.2 Configuration files

Comments

Blank lines and comment lines, beginning with a # symbol, are ignored.

Structure of configuration files

sfptpd configuration files use a simple INI file format. Options are grouped under labelled sections enclosed in square brackets. For example:

```
[general]
```

Sync modules are created in the [general] section.

Add a single `sync_module` option per type of sync module that is required. The first parameter identifies the type of sync module, and following parameters provide the name of each instance of the module. The following example creates two PTP sync module instances, named `ptp1` and `ptp2`:

```
sync_module ptp   ptp1   ptp2
             |     |      |
           type    |      |
                 name     |
                        name
```

A sync module can have generic options that apply to all instances of the sync_module type, and further instance-specific options that apply only to a specific instance of the sync_module:

- Generic options are set in a section of the configuration file labeled with the type of sync module. For example:

  `[ptp]`

- Instance-specific options are set in a section of the configuration file labeled with the instance name. For example:

  `[ptp1]`

Example of structure

Below is an example of the structure in a configuration file:

```
[general]

# specify instances of the PTP sync module named ptp1 and ptp2
sync_module ptp ptp1 ptp2


# other generic sfptpd options

[ptp]
# generic PTP options, applied to all instances

[ptp1]
# instance-specific PTP options, applied to the instance named ptp1

[ptp2]
# instance-specific PTP options, applied to the instance named ptp2
```

Supplied configuration files

The `sfptpd` distribution provides default configuration files for all supported modes in the `config` sub-directory:

- Default options are set within each config file. For example:

  `ptp_domain 0`

  Some options will require changing the values to match the specific PTP network.

- Additional options are commented out. For example:

  `#ptp_announce_interval 1`

  These can be selected by un-commenting the option line and, if required, entering a different value for the option:

  `ptp_announce_interval 3`

Some of these files contain generic options, and are described in Example configuration files on page 36. The remaining files in the `config` directory give examples of how to use the different sync modules. For descriptions of these, see the chapters describing the sync modules.

Creating additional configuration files

The user is free to create additional configuration files and store these anywhere on the local server. Configuration files can have any name, and by convention have a `.cfg` file extension. The path to the file is given on the `sfptpd` command line.

5.3 Configuration options

Table 2 lists configuration options for `sfptpd`. These are set in a section of the configuration file labeled `[general]`

Table 2: Configuration options for sfptpd

Option	Parameters	Description
sync_module	freerun|ptp|pps|ntp [<instance-name(s)>]	Create one or more instances of the specified sync module type.
		**CAUTION:** Do not create multiple instances of the NTP sync module.
selection_policy	automatic|manual [initial-manual-instance]	Use automatic or manual sync instance switching. Default: automatic.
selection_holdoff_ interval	<number>	Specifies how many seconds to wait after detecting a better instance before selecting it. The holdoff interval prevents rapid switching between instances of very similar quality. Default: 10 seconds.
selection_policy_rules	<manual | state | no-alarms | user-priority | clock-class | total-accuracy | allan-variance | steps-removed>*	Define the list of rules for the automatic selection policy
message_log	syslog|stderr| <filename>	Specifies where to send messages generated by the application. Default: stderr.
stats_log	off|stdout| <filename>	Specifies if and where to log statistics generated by the application. Default: disabled.

Table 2: Configuration options for sfptpd (continued)

Option	Parameters	Description
json_remote_monitor	<filename>	Output realtime information collected by the PTP remote monitor in JSON-lines format to this file (http://jsonlines.org). Default: disabled.
json_stats	<filename>	Enable output of machine-readable stats in JSON-lines format. Same data as produced by the stats_log.
daemon	—	Run `sfptpd` as a daemon. Default: disabled.
lock	off|on	Specifies whether to use a lock file to stop multiple simultaneous instances of the daemon. Default: enabled.
sync_interval	<number>	Specifies the interval in $2^{<number>}$ seconds at which the clocks are synchronized to the local reference clock.
local_sync_threshold	<number>	Specifies the threshold in nanoseconds of the offset between the system clock and a NIC clock over a 60s period to be considered in sync (converged). Default: 1us (hardware timestamping) Default: 100us (software timestamping) See also the instance-specific `sync_threshold` option for each of the sync modules.
clock_control	slew-and-step| step-at-startup| no-step|no-adjust| step-forward	Specifies how the clocks are controlled by sfptpd. Possible values are: • `slew-and-step`: allow clock stepping as necessary • `step-at-startup`: only allow the clock to be stepped at startup • `no-step`: never step the clock • `no-adjust`: do not make any adjustment to the clocks • `step-forward`: only step the clock forward. Default: slew-and-step.

Table 2: Configuration options for sfptpd (continued)

Option	Parameters	Description
clock_list	[<name>| <mac-address>| <clock-id>]	Specifies the set of clocks that sfptpd should discipline. Specify by clock name or MAC address or clock ID: • the clock name can be shown using `ethtool -T <interface>` • the clock id is derived from the MAC address with 0xFFFE in the middle, and is not so easy to use. Default: all clocks are disciplined.
persistent_clock_ correction	off|on	Specifies whether to used saved clock frequency corrections when disciplining clocks. Default: enabled.
timestamping_interfaces	[<name>| <mac-address>| *]	Specifies the set of interfaces on which general receive packet timestamping should be enabled. Specify by interface name or MAC address, or use * to enable receive timestamping on all interfaces that support it. Default: receive packet timestamping is disabled for all interfaces.
timestamping_disable_ on_exit	off|on	Specifies whether to disable timestamping on exit. This affects all interfaces specified with `timestamping_interfaces` as well as the interface selected for PTP. Default: on.
non_solarflare_nics	off|on	Enable PTP functionality and hardware timestamps of PTP packets from non-Solarflare adapter which must support PHC API. Default: off

Table 2: Configuration options for sfptpd (continued)

Option	Parameters	Description
pid_filter_p	<number>	Secondary servo PID filter proportional term coefficient. Default: 0.4.
pid_filter_i	<number>	Secondary servo PID filter integral term coefficient. Default: 0.03.
trace_level	<number>	Specifies the trace level if the application has been built with trace enabled. Default: 0, meaning no trace.

5.4 Example configuration files

This section describes some generic example configuration files:

- In the tarball distribution, configuration files are located in the `config` subdirectory of the install directory.

  To run `sfptpd` using one of these configuration files:

  ```
  cd <install_dir>
  ./sfptpd -i <interface> -f config/<filename>.cfg
  ```

- In the RPM distribution, configuration files are located in the `/usr/share/doc/packages/sfptpd/config` directory.

  To run `sfptpd` using one of these configuration files:

  ```
  cd /usr/share/doc/packages/sfptpd/
  sfptpd -i <interface> -f config/<filename>.cfg
  ```

default.cfg

A general default configuration:

- a PTP sync module is instantiated using the `sync_module` option
- `sfptpd` is run as a daemon by setting the `daemon` option
- the PTP instance is configured to act as a PTP slave using the `ptp_mode` option
- any PTP traffic is transmitted using hybrid mode by using the `ptp_network_mode` option (see Hybrid mode on page 64).

many_instances.cfg

Many instances configurations are supported on SFN7000 and later series adapters.

Example using multiple types of sync module, plus multiple instances of the same type of sync module:

- three PTP sync modules are instantiated using the `sync_module` option:
  - each instance is allocated to a different PTP domain using the `ptp_domain` option
  - all instances are configured to act as a PTP slave by using the `ptp_mode` option in the generic `[ptp]` section
  - any PTP traffic is transmitted using hybrid mode by using the `ptp_network_mode` option in the generic `[ptp]` section (see Hybrid mode on page 64)
- a PPS sync module is instantiated using the `sync_module` option
- an NTP sync module is instantiated using the `sync_module` option:
  - the key-id and key-value are set for authenticating with NTP using the `ntp_key` option
- a Freerun sync module is instantiated using the `sync_module` option.

5.5 Understanding the sfptpd startup sequence

When the `sfptpd` daemon is started it will generate several lines of output. A typical startup sequence is shown below. The PTP slave that graduates from startup to a **listening** state and finally to a **slave** state. Line numbers have been added.

```
[slave-server]# sfptpd -i enp1s0f0 -f config/sfptpd.conf
1   2016-12-05 10:26:16.158651: info: Solarflare Enhanced PTP Daemon,
    version 3.0.0.1003
2   2016-12-05 10:26:16.232403: info: no clock frequency correction file
    /var/lib/sfptpd/freq-correction-000f:53ff:fe01:7ba4
3   2016-12-05 10:26:16.232669: info: ptp ptp1: creating sync-instanc
4   2016-12-05 10:26:16.235757: info: ptp: clock is phc0(enp1s0f0/
    enp1s0f1)
5   2016-12-05 10:26:16.236050: info: interface enp1s0f0: SO_TIMESTAMPING
    enabled
6   2016-12-05 10:26:16.236281: info: using SO_TIMESTAMPING hardware
    timestamps
7   2016-12-05 10:26:16.336741: notice: ptp ptp1: Now in state:
PTP_LISTENING
8   2016-12-05 10:26:16.337149: info: selected sync instance ptp1
9   2016-12-05 10:26:18.837132: info: new best master selected:
    00a0:69ff:fe0c:2eb5(unknown)/1
```



```
10  2016-12-05 10:26:18.837146: notice: now in state: PTP_SLAVE,
    best master: 00a0:69ff:fe0c:2eb5(unknown)/1
11  2016-12-05 10:26:19.775070: info: received first Sync from Master
12  2016-12-05 10:26:20.774850: notice: slewing the clock with the maximum
    frequency adjustment
13  2016-12-05 10:26:20.774874: info: received first DelayResp from Master
```

Each line is described in the following table.

Table 3: sfptpd startup output

Line #	Description
1	Version of `sfptpd` running.
2	A frequency correction file holds the frequency correction value (PPB) that is currently being used to discipline a clock: • On initial startup frequency correction files do not exist. • Once created by `sfptpd` for each clock, they are updated every 60 seconds. • Files are preserved over server reboot and `sfptpd` restart.
3	Create an instance of a PTP sync module named ptp1.
4	The Local Reference Clock: • On an 8000 series or 7000 series adapter this identifies the PTP hardware clock in the form: phc0(ethX/ethY) where ethX is the active clock interface and ethY the second adapter clock interface on this adapter. Both interfaces on an 8000 series or 7000 series adapter share the same clock. • For the SFN6322F adapter this will identify the interface of the adapter clock.
5-6	If hardware timestamps cannot be initialized `sfptpd` will revert to software timestamping using the system clock.
7	The slave remains in a **LISTENING** state listening for Announce messages from any master clock on the network. The Best Master Clock algorithm is used to determine the most accurate master clock to which all slaves will synchronize. Any other master clock on the same network will go into passive mode and not send PTP messages once the best master has been selected. The Best Master Clock algorithm is implemented in such a way that all slaves will arrive at the same conclusion and select the same master clock. Before the adapter clock can begin synchronization with an upstream PTP master clock, the slave must receive an Announce message followed by Sync and Follow_up messages.

Table 3: sfptpd startup output (continued)

Line #	Description
8	The slave selects an initial sync module instance to use, as described in Sync module selection on page 20.
9-10	The slave moves to a **SLAVE** state once it has received an Announce message, and has selected the best master clock. The best master is identified by the following: • the clock identity (a UUID derived from the MAC address) • the clock host • a forward slash '/' • the port number (in hex) of the upstream master (usually 1, because most Grandmasters have only one PTP port).
11-13	The slave accepts PTP Sync messages from the master and starts to synchronize the adapter clock(s) with the master clock. As the slave begins to synchronize the adapter clock with the external master clock, the offset (master→slave), one-way-delay (slave→master), and frequency correction values are output. See Understanding sfptpd output below.

5.6 Understanding sfptpd output

Once it has reached a **SLAVE** state, `sfptpd` generates output describing the current state of the synchronization.

By default `sfptpd` will output the offset data to `stdout`. The user can redirect stats_log output to file using the configuration file `stats_log` parameter.

When logging the `stats_log` to file, it is possible to override this using the `-v` option on the `sfptpd` command line to cause stats to display on stdout.

The following examples are from a PTP sync module on a slave server. The timestamp at the start of the line has been omitted. Other sync modules produce similar output.

```
[ptp1:gm->phc0(enp1s0f0)], offset: 0.500, freq-adj: -663.353, in-sync: 1,
one-way-delay: 7534.500, grandmaster-id: 00a0:69ff:fe0c:2eb5
[phc0(enp1s0f0/enp1s0f1)->system], offset: 2.812, freq-adj: 46378.715,
in-sync: 1
```

Table 4: sfptpd offset output

Parameter	Description
`[ptp1:gm-> phc0(enp1s0f0)]`	A line of output generated for measurements between the external master clock and the Local Reference Clock: • the `->` separator indicates that the right-hand clock is being synchronized to the left-hand one • a `--` separator indicates no synchronization between clocks
`offset`	The current offset (nanoseconds) between the master clock and the slave clock identified at the start of the line. Immediately following startup this is expected to be a large value, but will gradually decrease until it settles to its lowest value. Synchronization can typically take between 15-30 minutes.
`freq-adj`	The current rate (PPB) at which the clock is being disciplined by `sfptpd`. This value is stored in the freq-correction file for this clock every 60 seconds.

Table 4: sfptpd offset output (continued)

Parameter	Description
`in-sync`	The in-sync flag will be 1 when the offset between master and slave clocks is below the value of the local_sync_threshold option (default (hardware timestamping 1µs) for a period of 1 minute. The local_sync_threshold when using software timestamping is 100µs. The in-sync flag will be 0 before the above condition is true. The in-sync flag will be 0 if the offset becomes greater than the local_sync_threshold. The in-sync flag will change to 0 if an alarm condition exists on the server to indicate problems in the PTP network e.g. PTP messages not being sent or received by the slave server. Check the topology file for current alarms status.
`one-way-delay`	The current one-way-delay (nanoseconds) between master and slave servers. This value should not be zero, but, once the server is synchronized, it should remain fairly stable: • If the value is zero, check that Delay_Req and Delay_Resp message are being sent and received. • If the value does not change at all over an extended period, check the Delay_Req interval. Check the topology file for current alarms status.
`parent-id`	UUID of the upstream master clock. This may be a boundary clock or the GM Master clock.
`gm-id`	Master clock UUID derived from its MAC address.
`[phc0(enp1s0f0/ enp1s0f1)->system]`	A line of output generated for measurements between the Local Reference Clock and the server system clock: • the `->` separator indicates that the right-hand clock is being synchronized to the left-hand one • a `--` separator indicates no synchronization between clocks

5.7 Viewing state and statistics files

On a server using the Solarflare `sfptpd` package, PTP alarms, status and performance data is accumulated in files created in the following directory:

`/var/lib/sfptpd`

From these files the user is able to monitor the performance and status of `sfptpd` and the PTP server.

Table 5 lists statistics files created by `sfptpd`.

Table 5: Statistics files created by sfptpd

File name	Persistent	Description	See
freq-correction-system	YES	Contains the frequency correction value used to discipline the server system clock.	page 45
freq-correction-<UUID>	YES	Contains the frequency correction value used to discipline the clock identified by the UUID identifier.	page 45
interfaces	NO	Identifies all adapters, and their timestamping capabilities.	page 45
ptp-nodes	NO	Identifies all PTP clocks present in the network.	page 45
state-system	NO	The contents of this file depend on the current PTP mode. In slave mode this file contains historical offset and status data for the slave server.	page 46
state-<sync_module> or state-<clock_UUID>	NO	Contains status information relevant to the identified sync module or clock.	page 46
stats-system	NO	Contains accumulated PTP performance data for the server including counts of the PTP message types sent and received.	page 47

Table 5: Statistics files created by sfptpd (continued)

File name	Persistent	Description	See
stats-<sync_module> or stats-<clock_UUID>	NO	Contains aggregated/ summarized synchronization data for the identified sync module or clock. From version 3.2.1 stats files are automatically generated and in txt and JSON formats.	page 47
topology	NO	PTP clock hierarchy diagram showing all clocks local to the slave server and PTP network elements between the slave and master clock. Identifies the timestamping mode (HW or SW). Displays active PTP alarms.	page 44
version	NO	Contains the `sfptpd` version.	page 48

File Persistence:

Of all the files created, the `freq-correction-*` files are persistent and will be preserved over `sfptpd` restart and over server reboot. All other files are non-persistent and are created when `sfptpd` is started.

File Update Rate:

The topology file and ptp-nodes file will update immediately to reflect changes in the ptp network topology or alarm conditions.

Alarms will also be updated immediately in clock state files.

Stats files are updated every 60 seconds.

Topology file

When viewed on a `sfptpd` slave server, the topology file presents a PTP clock hierarchy diagram showing all clocks local to the slave server and PTP network elements between the slave and master clock.

- A PTP Boundary Clock would be visible in the file as a parent to the slave.
- A PTP Transparent Clock will also be visible in the topology file.

The topology file identifies the current state of the selected slave clock, the interface being used to receive PTP messages and the timestamping mode being used. Each clock in the topology is identified by its UUID which is derived from its MAC address. Nanosecond values between clocks are the offset values recorded during the last file update.

The following output is a example of a topology file currently showing alarm states.

```
state: ptp-slave-alarm
alarms: no-sync-pkts no-delay-resps
interface: eth2 (eth2)
timestamping: hw
====================

          grandmaster
      000f:53ff:fe16:0474/1
                |
                |
           -170.000 ns
                |
                |
                v
        phc0(eth2/eth3)
       000f:53ff:fe21:9bb0
                |
                |
            -52.688 ns
                |
                |
                v
             system
             system
```

In the above example, the slave is in an alarm state indicating that Sync messages and Delay_Response messages are not currently being received from the master clock.

The topology file can be periodically monitored, for example, using a script to extract key fields, to monitor the current connection state and synchronization status of the PTP slave.

NOTE: During normal operation the topology file is updated every second. However, alarm or state changes are reflected in the file immediately.

PTP nodes file

The `ptp-nodes` file identifies all PTP clocks present in the network.

```
# cat ptp-nodes
|  state |            clock-id | port-number | domain |
-----------------------------------------------------------
| Master | 000f:53ff:fe16:0474 |           1 |      0 |
-----------------------------------------------------------
```

Interfaces file

The `interfaces` file identifies all adapters in the PTP slave server and identifies the timestamping capabilities. In the following example, Solarflare adapter interfaces are identified as being hardware timestamping capable.

```
# cat interfaces
| interface | ptp-caps | pkt-timestamping-caps |                                        product-name |
-------------------------------------------------------------------------------------------------------------
|      eth0 |        - |                     - |             Broadcom NetXtreme II Ethernet Controller |
|      eth1 |        - |                     - |             Broadcom NetXtreme II Ethernet Controller |
|      eth2 |       hw |                     - |  Solarflare SFN5322F SFP+ Time Stamping Server Adapter |
|      eth3 |        - |                     - |  Solarflare SFN5322F SFP+ Time Stamping Server Adapter |
|      eth4 |       hw |                    hw |               Solarflare SFN7122F SFP+ Server Adapter |
|      eth5 |       hw |                    hw |               Solarflare SFN7122F SFP+ Server Adapter |
-------------------------------------------------------------------------------------------------------------
```

Frequency correction files

Each `freq-correction` file contains the frequency correction values used to discipline a clock. The following table describes the values in the file:

Table 6: File: freq-correction-<UUID>

Name	Description
freq-correction-<UUID>	This identifier is constructed from the hardware address of the clock port.
value	This is the frequency correction value used to discipline the clock. The value is updated once per minute when the clock is in sync with its synchronization time source. This file persists over server reboot and `sfptpd` restart. Following either event, the frequency correction value is used to recommence disciplining of the clock ensuring faster re-convergence.

State files

Each `state` file contains status information.

NOTE: The contents of this file depend on the current PTP mode and position of the clock in the PTP hierarchy. A state file exists for each sync module and for the server system clock.

The following table describes the values in a typical `state` file:

Table 7: File: state-<sync_module> or state-<clock_UUID>

Name	Description
clock-name	identify the clock using this clock-id.
clock-id	the clock UUID.
state	ptp_slave | ptp_master also identifies if the clock is subject to any current alarms.
interface	identifies the PTP clock interface.
timestamping	current timestamping mode.
offset-from -master	Offset (nanoseconds) of LRC from the master clock.
one-way-delay	One-way-delay (nanoseconds) between LRC and master clock.
freq-adjustment-ppb	Current frequency correction value used to discipline this clock.
observed-drift	Drift in nanoseconds of slave LRC to master clock.
in-sync	0: observed offset > local_sync_threshold option 1: observed offset < local_sync_threshold option (The default for the local_sync_threshold is 1µs. when using hardware timestamping. 100µs for software timestamping)
steps-removed	steps removed from the master clock.
parent-clock-id	UUID of the PTP parent clock. If there is a boundary clock between LRC and the master clock this will identify the boundary clock.
parent-port-num	Port number relayed to the server by the master clock in the SourcePortID parameter.
grandmaster-id	The UUID grandmaster clock constructed from the grandmaster hardware address.

Table 7: File: state-<sync_module> or state-<clock_UUID> (continued)

Name	Description
grandmaster-clock-class	The current Grandmaster class value.
grandmaster-clock-accuracy	The current Grandmaster accuracy value.
grandmaster-priority1	The current Grandmaster priority 1 value.
grandmaster-priority2	The current Grandmaster priority 2 value.
current-utc-offset	The current UTC offset in seconds from TAI value specified in the config file with the **ptp-utc-offset** value.
leap-59	1 indicates a leap second is scheduled.
leap-61	1 indicates a leap second is scheduled.

Stats files

Each `stats` file contains accumulated data and statistics.

NOTE: The contents of this file depend on the current PTP mode and position of the clock in the PTP hierarchy. A stats file exists for each sync module and also for the server system clock.

The following table describes the values in a typical `stats` file:

Table 8: File: stats-<sync_module> or stats-<clock_UUID>

Name	Description
offset-from-master (ns)	Mean, min and max values are accumulated for these statistics. The number of samples taken during each period is recorded as is the start/end time of each sample period.
one-way-delay (ns)	
freq-adjustment-ppb (ppb)	

Table 8: File: stats-<sync_module> or stats-<clock_UUID> (continued)

Name	Description
synchronized	The stats file retains accumulated counts of each PTP message type sent or received from the server. The statistics file also records the number of PTP packets sent of received without being hardware timestamped. The number of samples taken during each period is recorded, as is the start/end time of each sample period.
announce-pkts-txed	
announce-pkts-rxed	
announce-timeouts	
sync-pkts-txed	
sync-pkts-rxed	
sync-pkt-timeouts	
follow-up-pkts-txed	
follow-up-pkts-rxed	
follow-up-timeouts	
delay-req-pkts-txed	
delay-req-pkts-rxed	
delay-resp-pkts-txed	
delay-resp-pkts-rxed	
delay-resp-timeouts	
delay-mode-mismatch-errors	
clock-steps	
tx-pkt-no-timestamp	
rx-pkt-no-timestamp	
crx-pkt-no-timestamp	Mean, minimum, maximum and standard deviation values are accumulated for these packets. The number of samples taken during each period is recorded, as is the start/end time of each sample period.

Version file

The `version` file contains the `sfptpd` version.

5.8 Logging options

The section explains and demonstrates the various data logging options available with `sfptpd`.

Event logging

PTP events including startup events can be directed to the syslog or stderr by enabling the following option in the configuration file:

```
message_log [syslog | stderr]
```

Stats logging

The following option is used to enable stats logging and display output on stdout or redirect output to a file:

```
stats_log [off | stdout | filename]
```

For more information, see Understanding sfptpd output on page 39.

PTP packet capture

Enabling the following option in the configuration file will display the contents of PTP packets sent and received by the `sfptpd` process:

```
ptp_pkt_dump
```

CAUTION: This option produces extensive output for each received PTP packet, as the following example of a PTP SYNC message demonstrates, and **should only be used for debugging purposes**.

```
2012-12-20 18:45:29.496035 msgDebugHeader: messageType 0
2012-12-20 18:45:29.496035 msgDebugHeader: versionPTP 2
2012-12-20 18:45:29.496035 msgDebugHeader: messageLength 44
2012-12-20 18:45:29.496035 msgDebugHeader: domainNumber 0
2012-12-20 18:45:29.496035 msgDebugHeader: flags 02 00
2012-12-20 18:45:29.496035 msgDebugHeader: correctionfield 0
2012-12-20 18:45:29.496035 msgDebugHeader: sourcePortIdentity.clockIdentity 000f:53ff:fe16:0474
2012-12-20 18:45:29.496035 msgDebugHeader: sourcePortIdentity.portNumber 1
2012-12-20 18:45:29.496035 msgDebugHeader: sequenceId 94
2012-12-20 18:45:29.496035 msgDebugHeader: controlField 0
2012-12-20 18:45:29.496035 msgDebugHeader: logMessageInterval 0
2012-12-20 18:45:29.496035 msgDebugSync: originTimestamp.seconds 1356029129
2012-12-20 18:45:29.496035 msgDebugSync: originTimestamp.nanoseconds 856792000
```

NOTE: This is different to using tcpdump which will capture packets received/sent at an interface.

Trace level

The following option is used to specify the trace level if the application has been built with trace enabled:

```
trace_level <number>
```

By default this is 0, meaning no trace. Setting a higher value enables trace.

PTP trace level

The following option is used to specify the PTP trace level:

```
ptp_trace <number>
```

By default this is 0, corresponding to off. Setting a higher value makes trace more verbose, up to a maximum value of 3.

5.9 Remote Monitoring

sfptpd from version 3.2 supports extensive local and remote monitoring capabilities, producing human-readable and machine-readable statistical/state/alarm data.

For details of the monitoring options refer to Chapter 14 on page 95.

6 PTP Sync Module

6.1 Description

Using the PTP sync module, sfptpd is receiving and sending PTP messages to an upstream PTP master clock.

6.2 Sync characteristics

This sync module can be configured in a wide variety of ways:

- PTP slave, with optional fallback
- PTP master, with optional fallback
- PTP boundary clock
- PTP multiple masters.

Examples of each are provided (see Example configuration files on page 61).

Loss of PTP link network connection

If the network connection to the external master clock is lost at any point, an alarm is triggered in the corresponding PTP sync module. It can no longer be selected as the time source, unless all other sync modules are in the alarm state (e.g. at startup):

- If other sync modules are instantiated and functioning correctly, sfptpd will select one of them as the time source.
- If no other sync module is available, Solarflare's `sfptpd` continues to discipline all clocks in the system including the system clock.

  `sfptpd` maintains a clock frequency correction file for each clock in a server:

  - If the LRC is a Solarflare PTP adapter, `sfptpd` will continue to discipline the LRC using the LRC frequency correction file. `sfptpd` will ensure that other clocks, hardware or system, will continue to synchronize to the LRC.
  - When there is no Solarflare PTP adapter in the server, or if the LRC is the system clock, `sfptpd` will continue to discipline the system clock using the system clock frequency correction file.

Following restoration of the network link, the PTP sync modules receives PTP packets and the alarm is no longer triggered. It is eligible for selection as a time source, and `sfptpd` will resume normal discipline procedure.

6.3 PTP over VLAN

Solarflare Enhanced PTP supports PTP packets over tagged 802.1Q Virtual Local Area Network (VLAN) interfaces. Users should consult the relevant OS documentation for VLAN configuration instructions. Assuming interface `eth2.120` is a network interface configured with VLAN tag 120, the following example identifies `sfptpd` VLAN configuration.

```
./sfptpd -i eth2.120 -f config/ptp_slave.cfg
```

6.4 PTP over bonded interfaces

Solarflare adapters and `sfptpd` support PTP packets over bonded interfaces in an **active/standby mode**. In addition, sfptpd also supports bonding over LACP (802.3ad) bonding.

Bonding of Solarflare interfaces employs the Linux bonding driver. Multiple ports can be included into a single bond where one port is selected as the active interface and all others are standby.

- `sfptpd` will detect which port is active and which ports are passive in the bond.
- `sfptpd` will discipline the high precision clock on the active port's network adapter.
- `sfptpd` will discipline the clocks of passive ports from the active adapter's clock.
- Via the bonding driver the user can select the active port (and therefore clock).
- A bond can include non-PTP capable Solarflare ports.

  `sfptpd` will switch to software time-stamping when a non-hardware time-stamping port becomes active.
- A bond can include non Solarflare ports.

  `sfptpd` will switch to software time-stamping when a non-Solarflare port becomes active.
- A bond can include any number of ports.

NOTE: There are limitations to combining PTP and non-PTP ports in the same bond. See Bonding on page 106 for details.

Bonding configuration

Bonding of Solarflare interfaces is handled by the standard Linux bonding driver. Users should refer to http://www.kernel.org/doc/Documentation/networking/bonding.txt for details of alternative methods for bonding configuration. The following example is a manually bonding configuration using ifenslave:

```
# modprobe bonding miimon=100 mode=active-backup primary=eth5
# ifconfig bond0 172.16.136.27/21
# ifenslave bond0 eth0 eth1
```

To run `sfptpd` over the bonded interfaces:

```
./sfptpd -i bond<N> -f config/ptp_slave.cfg
```

Action on active port failover

The active port in the bonding interface identified on the command line with the -i option is the active clock. In the event of failure of the active port:

- If the standby port is a PTP capable port, synchronization will continue. The standby port clock is synchronized to the active port clock.
- If the standby port is a port that does not support hardware timestamps, the system clock becomes the LRC and `sfptpd` uses software timestamping.

Link States, Interface Hotswap

The sfptpd 3.2.1 release uses the Linux Netlink interface to improve robustness and availability of sfptpd when used over Linux bonded interfaces.

- As long as there is a link available, sfptpd will remain active in the PTP_SLAVE state.
- If all bonded links are down sfptpd will remain active in the PTP_LISTENING state and will activate the no-interface alarm.
- When a failed bond link is restored, sfptpd will resume, changing to a PTP_SLAVE state if the previous state was PTP_LISTENING.
- If an adapter/interface is swapped out/in, sfptpd will identify the new interface(s) and continue.

6.5 Configuration options

The PTP sync module has generic options that apply to all instances of the module, and further instance-specific options that apply only to a single instance of the module.

Generic configuration options

Table 9 shows the generic configuration options for the PTP sync module. These must be in a section labeled `[ptp]`.

Table 9: Generic configuration options for the PTP sync module

Option	Parameters	Description
ptp_mgmt_msgs	disabled|read-only	Configures PTP Management Message support. Possible values are: • `disabled`: management messages disabled • `read-only`: only requests to read information (GET) will be accepted. See PTP management messages on page 66. By default this is disabled.
ptp_max_foreign_records	<number>	The maximum number of PTP foreign master records a node is able to store simultaneously. The default value is 16.
ptp_uuid_filtering	off|on	Specifies whether to use hardware packet filtering against parent UUID for the SFN6322F adapter. By default this is enabled.
ptp_domain_filtering	off|on	Specifies whether to use hardware packet filtering against PTP domain for the SFN6322F adapter. The default value is on.

Instance-specific configuration options

Table 10 shows the instance-specific configuration options for the PTP sync module. These must be in a section labeled with the instance name.

Table 10: Instance-specific configuration options for the PTP sync module

Option	Parameters	Description
ptp_mode	slave|master	Specifies the PTP mode of operation. The default mode is slave.
interface	<interface-name>	Specifies the name of the interface that PTP should use.
priority	<number>	Relative priority of sync module instance.[1] Smaller values have higher priority. The default is 128.
sync_threshold	<number>	Threshold in nanoseconds of the offset from the clock source over a 60s period to be considered in sync (converged). The default is 1000ns with hardware timestamping and 100000ns with software timestamping. See also the `local_sync_threshold` option in Table 2 on page 33.
ptp_pkt_dump	—	Dump each received PTP packet in detail.
ptp_pps_log	—	Enable logging of PPS measurements.
ptp_tx_latency	<number>	Specifies the outbound latency in nanoseconds. Use this option and the ptp_rx_latency option to correct for any network asymmetry. See Chapter 13 on page 89.
ptp_rx_latency	<number>	Specifies the inbound latency in nanoseconds. Use this option and the ptp_tx_latency option to correct for any network asymmetry. See Chapter 13 on page 89.
ptp_delay_mechanism	end-to-end| peer-to-peer	Peer delay mode. The default mode is end-to-end.
ptp_network_mode	multicast|hybrid	Network mode. See Network transmission modes on page 64. The default mode is hybrid.

Table 10: Instance-specific configuration options for the PTP sync module (continued)

Option	Parameters	Description
ptp_ttl	<number>	The TTL value to use in transmitted PTP packets. The default value is 64.
ptp_utc_valid_handling	default|ignore|prefer|require	Controls how the UTC offset valid flag is used. The specification is ambiguous in its description of the meaning of the UTC offset valid flag, and this has resulted in varying different implementations. In most implementations, if the UTC offset valid flag is not set then the UTC offset is not used, but in others the UTC offset valid is an indication that the master is completely confident that the UTC offset is correct. Various options are supported: • `default`: if UTCV is set use the UTC offset, otherwise do not use it • `ignore`: do not used the UTCV flag, always apply the indicated UTC offset • `prefer`: prefer grand masters that have the UTCV flag set above those that don't • `require`: do not accept GMs that do not set UTCV. See UTC offset on page 67.
ptp_domain	<number>	Specifies the PTP domain. The default value is 0.

Table 10: Instance-specific configuration options for the PTP sync module (continued)

Option	Parameters	Description
ptp_timing_acl_allow	<ip-address-list>	Access control permit list for timing packets: The format is a space- or comma-separated list of network prefixes in a.b.c.d/x notation, where a.b.c.d is the subnet and x is the mask. For single IP addresses, 32 should be specified for the mask. See Access control lists on page 67.
ptp_timing_acl_deny	<ip-address-list>	Access control deny list for timing packets: The format is a space- or comma-separated list of network prefixes in a.b.c.d/x notation, where a.b.c.d is the subnet and x is the mask. For single IP addresses, 32 should be specified for the mask. See Access control lists on page 67.
ptp_timing_acl_order	permit-deny| deny-permit	Access control list evaluation order for timing packets. See Access control lists on page 67.
ptp_mgmt_acl_allow	<ip-address-list>	Access control permit list for management packets: The format is a space- or comma-separated list of network prefixes in a.b.c.d/x notation, where a.b.c.d is the subnet and x is the mask. For single IP addresses, 32 should be specified for the mask. See Access control lists on page 67.
ptp_mgmt_acl_deny	<ip-address-list>	Access control deny list for management packets: The format is a space- or comma-separated list of network prefixes in a.b.c.d/x notation, where a.b.c.d is the subnet and x is the mask. For single IP addresses, 32 should be specified for the mask. See Access control lists on page 67.
ptp_mgmt_acl_order	permit-deny| deny-permit	Access control list evaluation order for management packets. See Access control lists on page 67.

Table 10: Instance-specific configuration options for the PTP sync module (continued)

Option	Parameters	Description
ptp_announce_interval	<number>	The PTP Announce packet interval in $2^{<number>}$ seconds. The default value is 1.
ptp_announce_timeout	<number>	The PTP Announce packet receipt timeout as a number of Announce packet intervals. The default value is 6.
ptp_sync_pkt_interval	<number>	The PTP Sync packet interval in 2^<number> seconds. The default value is 0 (1 second).
ptp_sync_pkt_timeout	<number>	The PTP Sync packet receipt timeout as a number of Sync packet intervals. The default value is 6.
ptp_delayreq_interval	<number>	The PTP Delay Request / Peer Delay Request packet interval in $2^{<number>}$ seconds. If specified for a PTP slave, this overrides the value communicated to the slave from the master.
ptp_delayresp_timeout	<number>	The PTP Delay Response receipt timeout in $2^{<number>}$ seconds. The default value is -2 (250ms).
ptp_bmc_priority1	<number>	Best Master Clock algorithm, priority1 parameter. Used when in PTP master mode.
ptp_bmc_priority2	<number>	Best Master Clock algorithm, priority2 parameter. Used when in PTP master mode.
ptp_trace	<number>	PTP trace level: • 0 corresponds to off • 3 corresponds to maximum verbosity. The default value is 0.

Table 10: Instance-specific configuration options for the PTP sync module (continued)

Option	Parameters	Description
pid_filter_p	<number>	Secondary servo PID filter proportional term coefficient. The default value is 0.2.
pid_filter_i	<number>	Secondary servo PID filter integral term coefficient. The default value is 0.003.
remote_monitor		Enable the remote monitor. Collects Slave Event Monitoring messages
json_remote_monitor	<filename>	Output realtime information collected by the PTP remote monitor in JSON-lines format. Default: disabled.
mon_monitor_address	<address>	Address of monitoring station to which to send signaling messages with slave event monitoring data for the mon_rx_sync_timing_data, mon_rx_sync_computed_data and mon_tx_event_timestamps commands. Default is multicast.
mon_rx_sync_timing_data	none	sfptpd will generate/output the timestamps for received Sync messages.
mon_rx_sync_computed_data	none	sfptpd will generate/output the computed offset from the master clock.
mon_tx_event_timestamps	none	sfptpd will generate/output the timestamp for transmitted Delay_Request messages.
outlier_filter_size	<number>	Number of data samples stored in the offset from master filter. The valid range is [5, 60]. The default value is 60.
outlier_filter_adaption	<number>	Controls how outliers are fed into the offset from master filter. The valid range is [0, 1]: • a value of 0 means that outliers are not fed into the filter (not recommended) • a value of 1 means that each outlier is fed into the filter unchanged • values between result in a portion of the value being fed in. The default value is 1.

Table 10: Instance-specific configuration options for the PTP sync module (continued)

Option	Parameters	Description
mpd_filter_size	<number>	Number of data samples stored in the mean path delay filter. The valid range is [1, 25]: • a value of 1 means that the filter is off • higher values will reduce the adaptability of PTP but increase its stability. The default value is 8.
mpd_filter_ageing	<number>	Controls the ageing of samples in the mean path delay filter. The ageing is expressed in units of nanoseconds per second. The default value is 2.0ns/s,
fir_filter_size	<number>	Number of data samples stored in the FIR filter. The valid range is [1, 128]. • a value of 1 means that the filter is off • higher values will reduce the adaptability of PTP but increase its stability. The default value is 1, and so the filter is off.

1. This is the user priority for this sync instance within this daemon and is unrelated to the PTP 'priority1' and 'priority2' values.

6.6 Example configuration files

This section describes the example configuration files that use PTP as the only sync module, or as the principal one. See Example configuration files on page 36.

NOTE: For descriptions of the remaining files in the `config` directory, see the chapters describing the other sync modules.

ptp_slave.cfg

This file shows how to configure `sfptpd` to act as a PTP slave:

- a PTP sync module is instantiated using the `sync_module` option
- the PTP sync module is configured to act as a PTP slave by using the `ptp_mode` option
- the slave also has `ptp_tx_latency` and `ptp_rx_latency` options that can be used to compensate for latency through the interface (see Chapter 13 on page 89)
- any PTP traffic is transmitted using hybrid mode by using the `ptp_network_mode` option (see Hybrid mode on page 64).

ptp_slave_ntp_fallback.cfg

This file shows how to configure `sfptpd` to act as a PTP slave, with fallback to NTP:

- an NTP sync module is instantiated using the `sync_module` option
  - the key-id and key-value are set for authenticating with NTP using the `ntp_key` option
- a PTP sync module is instantiated using the `sync_module` option
- the PTP sync module is configured to act as a PTP slave by using the `ptp_mode` option
- the slave also has `ptp_tx_latency` and `ptp_rx_latency` options that can be used to compensate for latency through the interface (see Chapter 13 on page 89)
- any PTP traffic is transmitted using hybrid mode by using the `ptp_network_mode` option (see Hybrid mode on page 64).

Failover to NTP

sfptpd will failover from PTP to NTP if there are active PTP alarms:

```
pps-no-signal
pps-seq-num-error
no-time-of-day
pps-bad-signal
no-sync-pkts
no-follow-ups
```



```
no-delay-resps
no-pdelay-resps
no-pdelay-resp-follow-ups
no-tx-timestamps
no-rx-timestamps
no-interface
clock-ctrl-failure
```

See PTP Alarms on page 115 for a description of alarms.

If alarm conditions clear, sfptpd will resume synchronization using PTP.

The selection_holdoff_interval (seconds) can be set to prevent immediate failover caused when only one or two PTP packets are not received.

sfptpd will NOT failover to NTP because of synchronization accuracy.

ptp_slave_multiple.cfg

Many instances configurations are supported on SFN7000 and later series adapters.

This file shows how to configure `sfptpd` to act as a PTP slave listening to multiple PTP domains:

- three PTP sync modules are instantiated using the `sync_module` option:
- each instance is allocated to a different PTP domain using the `ptp_domain` option
- all instances are configured to act as a PTP slave by using the `ptp_mode` option in the generic `[ptp]` section
- all instances use `ptp_tx_latency` and `ptp_rx_latency` options in the generic `[ptp]` section that can be used to compensate for latency through the interface (see Chapter 13 on page 89)
- any PTP traffic is transmitted using hybrid mode by using the `ptp_network_mode` option in the generic `[ptp]` section (see Hybrid mode on page 64).

ptp_master_ntp.cfg

This file shows how to configure `sfptpd` to act as a PTP master, using the NTP daemon as the reference:

- an NTP sync module is instantiated using the `sync_module` option
  - the key-id and key-value are set for authenticating with NTP using the `ntp_key` option
- a PTP sync module is instantiated using the `sync_module` option
- the PTP sync module is configured to act as a PTP master by using the `ptp_mode` option

- the master also has `ptp_tx_latency` and `ptp_rx_latency` options that can be used to compensate for latency between the NTP daemon and interface (see Chapter 13 on page 89)
- any PTP traffic is transmitted using hybrid mode by using the `ptp_network_mode` option in the generic `[ptp]` section (see Hybrid mode on page 64).

NOTE: NTP uses UTC time - not atomic (TAI) time. When sfptpd is configured as a master clock, it uses UTC time. Therefore ensure that the ptp_utc_offset parameter in the sfptpd master config file is set to a value of 0.

ptp_master_freerun.cfg

This file shows how to configure `sfptpd` to act as a PTP master, using the NIC clock as the reference:

- a Freerun sync module is instantiated using the `sync_module` option
- a PTP sync module is instantiated using the `sync_module` option
- the PTP sync module is configured to act as a PTP master by using the `ptp_mode` option
- the master also has `ptp_tx_latency` and `ptp_rx_latency` options that can be used to compensate for latency between the clock and interface (see Chapter 13 on page 89)

ptp_boundary.cfg

This file shows how to configure `sfptpd` to act as a PTP boundary clock:

- two PTP sync modules are instantiated using the `sync_module` option
- one instance is associated with a particular `interface`, and is configured to act as a PTP master by using the `ptp_mode` option
- the other instance is associated with a different `interface`, and is configured to act as a PTP slave by using the `ptp_mode` option
- the slave also has `ptp_tx_latency` and `ptp_rx_latency` options that can be used to compensate for latency between the two interfaces (see Chapter 13 on page 89)
- both instances are allocated to the same PTP domain by using the `ptp_domain` option in the generic `[ptp]` section
- any PTP traffic is transmitted using hybrid mode by using the `ptp_network_mode` option in the generic `[ptp]` section (see Hybrid mode on page 64)

NOTE: Using the ptp_boundary.cfg, sfptpd will operate as both master and slave simultaneously within the same or different PTP domains. This boundary clock mode is not intended to meet all the requirements of a boundary clock as specified in the IEEE-1588 specification.

NOTE: When acting as a master clock, sfptpd uses UTC time and not TAI time.

ptp_domain_bridge.cfg

This file shows how to configure `sfptpd` to act as a PTP domain bridge:

- two PTP sync modules are instantiated using the `sync_module` option
- one instance is configured to act as a PTP master by using the `ptp_mode` option
- the other instance is configured to act as a PTP slave by using the `ptp_mode` option
- each instance is allocated to a different PTP domain using the `ptp_domain` option
- the slave also has `ptp_tx_latency` and `ptp_rx_latency` options that can be used to compensate for latency between the two interfaces (see Chapter 13 on page 89)
- any PTP traffic is transmitted using hybrid mode by using the `ptp_network_mode` option in the generic `[ptp]` section (see Hybrid mode on page 64).

6.7 Configuration options in detail

Network transmission modes

The network transmission mode for an instance of the PTP sync module is set by the `ptp_network_mode` option in the configuration file. It can have the following values:

- `multicast`
- `hybrid`.

Multicast mode

The is the standard PTP mode whereby all PTP packets between master and slave are sent as multicast. The implication is that all slaves receive all Delay_Req, Delay_Resp message pairs between all other slaves and the PTP master clock thereby increasing the amount of traffic on the network. The traffic level generated by multicast transmission is usually not a problem on smaller networks employing only a few PTP slaves.

Multicast mode can be selected for `sfptpd` using the configuration file option `ptp_network_mode multicast`.

Hybrid mode

PTP hybrid mode allows a PTP slave clock to use unicast transmission to send Delay_Req messages to the master clock which, in turn, will respond with a unicast Delay_Resp packet direct to the relevant slave. This reduces the level of PTP traffic

on the network which can be a factor when scaling to larger networks employing many PTP slaves. Hybrid mode only requires the network to support multicast transmission from master to slave.

A `sfptpd` slave, using hybrid mode, will make three attempts to contact a master clock when sending a Delay_Req message using unicast transmission. If the master clock fails to respond to unicast transmissions the `sfptpd` slave will revert to multicast transmission. If hybrid mode communication is possible with the master clock, the slave will remain in hybrid mode until/if a new master clock is selected.

`sfptpd` will generate an error message if the PTP master fails to respond to the unicast Delay_Req message. Error messages will go to `stderr` or to `syslog`, depending on the logging configuration.

Hybrid mode is the default mode for `sfptpd` and can be enabled/disabled using the configuration file option `ptp_network_mode`.

Hardware timestamps

NOTE: Features described in this section are not available with the SFN6322F adapter. Customers using the SFN6322F, ignore this section.

`sfptpd` can be used to enable hardware timestamping of all packets (to the Linux kernel) on specified interfaces. Interfaces are identified as a list using the following configuration file option:

```
timestamping_interfaces  [<name | mac-address | *>]
```

- To timestamp all received packets on all interfaces:

  ```
  timestamping_interfaces *
  ```

- To timestamp all received packets on eth2 and eth3:

  ```
  timestamping_interfaces eth2 eth3
  ```

If hardware timestamping is required only for PTP packets, there is no need to enable this parameter.

Hardware timestamps enable/disable

To enable/disable hardware timestamping of all received network packets after `sfptpd` exits, use the following configuration file option:

```
timestamping_disable_on_exit [<off | on>]
```

Hardware timestamps (kernel/Onload)

Applications can recover hardware timestamps for all received packets using the SO_TIMESTAMPING socket option. For more details of hardware packet timestamps when using the kernel driver see the *Solarflare Server Adapter User Guide* (SF-103837-CD). For more details of using hardware packet timestamps when using OpenOnload see the *Onload User Guide* (SF-104474-CD).

PTP management messages

PTP Management 'GET' messages are supported from `sfptpd` v2.2.1. These messages allow a network management node to retrieve PTP clock data from other PTP nodes in the network. Table 11 includes all supported managementId types.

Table 11: Management messages

managementId	Value
MGMT_ID_NULL	0x0000
MGMT_ID_CLOCK_DESCRIPTION	0x0001
MGMT_ID_USER_DESCRIPTION	0x0002
MGMT_ID_DEFAULT_DATA_SET	0x2000
MGMT_ID_CURRENT_DATA_SET	0x2001
MGMT_ID_PARENT_DATA_SET	0x2002
MGMT_ID_TIME_PROPERTIES_DATA_SET	0x2003
MGMT_ID_PORT_DATA_SET	0x2004
MGMT_ID_PRIORITY1	0x2005
MGMT_ID_PRIORITY2	0x2006
MGMT_ID_DOMAIN	0x2007
MGMT_ID_SLAVE_ONLY	0x2008
MGMT_ID_LOG_ANNOUNCE_INTERVAL	0x2009
MGMT_ID_ANNOUNCE_RECEIPT_TIMEOUT	0x200a
MGMT_ID_LOG_SYNC_INTERVAL	0x200b
MGMT_ID_VERSION_NUMBER	0x200c
MGMT_ID_TIME	0x200f
MGMT_ID_CLOCK_ACCURACY	0x2010
MGMT_ID_UTC_PROPERTIES	0x2011
MGMT_ID_TRACEABILITY_PROPERTIES	0x2012
MGMT_ID_DELAY_MECHANISM	0x6000
MGMT_ID_LOG_MIN_PDELAY_REQ_INTERVAL	0x6001

Management messages are disabled by default. To enable `sfptpd` support for GET messages, set the following option in the PTP configuration file.

```
ptp_mgmt_msgs read-only
```

For a detailed description of Management Messages and the managementId values, refer to the Std 1588™-2008, *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*.

UTC offset

The UTC offset (TAI - UTC) is conveyed to PTP slave clocks from a PTP master clock within the Sync or FollowUp messages. The UTC valid flag (UTCV) indicates whether the UTC offset time is considered valid by a master device and therefore should be used by a PTP slave device.

In the `sfptpd` configuration file, the `ptp_utc_valid_handling` option identifies the action taken with the UTC offset value by `sfptpd`.

Table 12: UTC handling

Value	Description
default	If UTCV is set, use UTC, otherwise do not.
ignore	Ignore the UTC valid flag. Always apply the UTC offset.
prefer	Prefer master clocks that set the UTCV flag above those that do not.
require	Do not accept PTP messages from master clocks that do not set the UTCV flag.

Access control lists

Access control lists (ACLs) restrict the set of network addresses from which sfptpd will accept certain types of message or request. There are ACLs to control access to the following features:

- PTP timing messages
- PTP management messages
- PTP monitoring requests

From the PTP configuration file, access is controlled through separate allow and deny lists of IP addresses. A further configuration option dictates the order in which allow and deny lists are evaluated.

The form of the configuration options is:

- ptp_<acl-type>_acl_allow defines the ‘allow’ list.
- ptp_<acl-type>_acl_deny defines the ‘deny’ list.
- ptp_<acl-type>_acl_order defines the priorities of the two lists.

Default behaviour

With no ACL options specified, all messages or requests are accepted.

Allow-Deny order

If ‘allow’ or ‘deny’ lists are defined but no order is specified then ‘allow-deny’ order is assumed. This policy can be specified explicitly:

```
ptp_timing_acl_order allow-deny
```

The default policy for the allow-deny order is that all messages or requests are denied.

The ‘allow’ list defines networks and hosts that are permitted to communicate and the ‘deny’ list defines networks and hosts that will nevertheless be excluded. It makes sense for the deny list to be more specific than the allow list.

```
ptp_timing_acl_allow 172.16.128.0/21
```

```
ptp_timing_acl_deny 172.16.128.48/32 172.16.128.47/32
```

In the above example, the first line identifies a subnet from which, exclusively, PTP timing messages will be accepted. The second, optional, line identifies that access from the two specified hosts will be denied, despite being in the ‘allow’ list. These could perhaps be VPN end-points, DMZ hosts, or hosts running customer applications.

NOTE: Note: prior to sfptpd v3.2.1, this ordering was specified by the deprecated deny-permit form. The new behaviour is consistent with other Internet software.

Deny-Allow order

This policy is introduced with:

```
ptp_timing_acl_order deny-allow
```

The default policy for the deny-allow order is that all messages or requests are accepted.

The ‘deny’ list defines networks and hosts that will be disallowed and the ‘allow’ list defines networks and hosts that will nevertheless be permitted to communicate. It makes sense for the allow list to be more specific than the allow list.

```
ptp_timing_acl_deny 172.16.128.0/21
```

```
ptp_timing_acl_allow 172.16.128.48/32 172.16.128.47/32
```

In the above example, the first line identifies a subnet from which PTP timing messages will be denied. The second, optional, line identifies two specific hosts that will be allowed to send timing messages, despite being in the ‘deny’ list. These hosts could perhaps be company infrastructure that is on a subnet used as a DMZ for visiting vendors.

NOTE: Note: prior to sfptpd v3.2.1, this ordering was specified by the deprecated permt-deny form. The new behaviour is consistent with other Internet software.

Outlier removal

The sync module scans incoming values for outliers, where the offset from the master differs significantly from other previously received values. These outliers can be partially or completely removed, so they do not feedback into the time synchronization:

- Outliers are detected using methods such as Peirce's criterion.
- The number of values stored by using the `outlier_filter_size` option:
  - the default value gives the best accuracy of outlier detection
  - reducing this value lowers overheads, but also reduces accuracy.
- The amount by which outliers feedback into the time synchronization is controlled using the `outlier_filter_adaption` option:
  - the default value of 1 means that outliers are used unchanged
  - reducing this value makes outliers have less effect
  - reducing this value to 0 so outliers have no effect is not recommended.

7 PPS Sync Module

7.1 Description

Using the PPS sync module, the `sfptpd` slave receives time-of-day from NTP, and also receives a 1PPS signal. The slave runs an NTP client which `sfptpd` can both read and control.

PPS pulses are received from a Solarflare PTP adapter with a PPS interface, `sfptpd` selects the precision clock on the adapter as the LRC.

`sfptpd` periodically polls the host server NTP client to provide the time of day to the LRC. Thereafter `sfptpd` synchronizes the LRC to the 1PPS pulse, and keeps all other clocks (including the system clock) synchronized with the LRC.

7.2 Configuration options

The PPS sync module has instance-specific options that apply only to a single instance of the module.

Instance-specific configuration options

Table 13 shows the instance-specific configuration options for the PPS sync module. These must be in a section labeled with the instance name.

Table 13: Instance-specific configuration options for the PPS sync module

Option	Parameters	Description
interface	<interface-name>	Specifies the name of the interface that PPS should use.
priority	<number>	Relative priority of sync module instance. Smaller values have higher priority. Default: 128
sync_threshold	<number>	Threshold in nanoseconds of the offset from the clock source over a 60s period to be considered in sync (converged). Default: 1us (hardware timestamping) Default: 100us (software timestamping) See also the `local_sync_threshold` option in Table 2 on page 33.

Table 13: Instance-specific configuration options for the PPS sync module (continued)

Option	Parameters	Description
master_clock_class	locked|holdover|freerunning	Master clock class. Default: locked
master_time_source	atomic|gps|ptp|ntp oscillator	Master time source. Default: GPS
master_accuracy	<number>|unknown	Master clock accuracy in ns, or unknown. Default: unknown
pps_delay	<number>	PPS propagation delay in nanoseconds.
pid_filter_p	<number>	Secondary servo PID filter proportional term coefficient. Default: 0.05
pid_filter_i	<number>	Secondary servo PID filter integral term coefficient. Default: 0.001
outlier_filter_type	disabled|std-dev	Specifies the filter type to use to reject outliers: • disabled specifies no filtering • std-dev applies filtering based on a sample's distance from the mean, expressed as a number of standard deviations. Default: std-dev
outlier_filter_size	<number>	Number of data samples stored in the filter. For std-dev type the valid range is [5,60]. Default: for std-dev is 30.

Table 13: Instance-specific configuration options for the PPS sync module (continued)

Option	Parameters	Description
outlier_filter_ adaption	<number>	Controls how outliers are fed into the filter: • a value of 0 means that outliers are not fed into the filter (not recommended) • a value of 1 means that each outlier is fed into the filter unchanged • values between result in a portion of the value being fed in. Default: 1.0
fir_filter_size	<number>	Number of data samples stored in the FIR filter. The valid range is [1, 128]. • a value of 1 means that the filter is off • higher values will reduce the adaptability of PTP but increase its stability. Default: 4

7.3 Default configuration file

This section describes the default configuration file that use PPS as the only sync module, or as the principal module. See Example configuration files on page 36.

pps_slave.cfg

This file shows how to configure `sfptpd` to use the PPS sync module:

- a PPS sync module is instantiated using the `sync_module` option
- other options are commented out:
  - if no changes are made, the NIC clock is used as the reference
  - configure the NTP options to use the NTP daemon as the reference.
  - configure NTP authentication as perNTP authentication on page 73.
  - uncomment and update the PPS options such as `pps_delay` to refine the PPS performance.

7.4 Constraints

The PPS sync module only works on PTP-enabled Solarflare adapters with a PPS kit fitted. Only one sync_module should be created for each PPS interface.

8 NTP Sync Module

8.1 Description

The NTP sync module allows `sfptpd` to use the NTP daemon as the local reference clock.

NOTE: NTP authentication is necessary to allow sfptpd to control the NTP client.

8.2 Sync characteristics

This sync module provides accuracy that is limited by the quality of your NTP master, and the reliability of NTP traffic over your network.

NTP authentication

Before `sfptpd` can query the local NTP daemon, it is necessary to setup symmetric key authentication parameters in the NTP daemon configuration files and in the PTP configuration file.

1. The following is an extract from a typical `/etc/ntp.conf` file:

```
# Key file containing the keys and key identifiers used when operating
# with symmetric key cryptography.
#keys /etc/ntp/keys

# Specify the key identifiers which are trusted.
#trustedkey 4 8 42

# Specify the key identifier to use with the ntpdc utility.
#requestkey 8

# Specify the key identifier to use with the ntpq utility.
#controlkey 8
```

2. Uncomment the `keys`, `trustedkey` and `requestkey` lines and replace/add an integer value:

```
keys /etc/ntp/keys
trustedkey 15
requestkey 15
```

3. Append the following line to the file `/etc/ntp/keys`

```
15 M MyAuthenticaTionString
```

4. Restart the ntpd service

```
# service ntpd restart
```

5. Edit the `config/pps_slave.cfg` file to match the values specified for NTP:

```
ntp_key 15 MyAuthenTicationString
```

6 Start the `sfptpd` daemon.

NOTE: There should be no spaces in the authentication string.

8.3 Configuration options

The NTP sync module has instance-specific options that apply only to a single instance of the module.

Instance-specific configuration options

Table 14 shows the instance-specific configuration options for the NTP sync module. These must be in a section labeled with the instance name.

Table 14: Instance-specific configuration options for the NTP sync module

Option	Parameters	Description
priority	<number>	Relative priority of sync module instance. Smaller values have higher priority. Default: 128.
sync_threshold	<number>	Threshold in nanoseconds of the offset from the clock source over a 60s period to be considered in sync (converged). Default: 10000000ns. See also the `local_sync_threshold` option in Table 2 on page 33.

Table 14: Instance-specific configuration options for the NTP sync module (continued)

Option	Parameters	Description
ntp_poll_interval	<number>	Specifies how often the NTP daemon will be polled, in seconds. Default: 1
ntp_key	<key-id> <key-value>	NTP authentication key ID and value. This is used to authenticate control requests with the NTP daemon: • The key ID is an integer. It must be identified in the NTP configuration file, in the `trustedkey` and `requestkey` lists. • The key value is a plain text ASCII string up to 31 characters long (i.e. an “m” key as defined by ntpd). It must appear in the `/etc/ntp/keys` file, identified by the key ID. There should be NO spaces in the authentication string. See NTP authentication on page 73.

8.4 Default configuration file

This section describes the default configuration file that use NTP as the only sync module, or as the principal one. See Example configuration files on page 36.

Some other configuration files use NTP as a fallback if the principal module fails. For example, see:

- ptp_master_ntp.cfg on page 62
- ptp_slave_ntp_fallback.cfg on page 61
- pps_slave.cfg on page 72.

NOTE: For descriptions of the remaining files in the `config` directory, see the chapters describing the other sync modules.

ntp.cfg

This file shows how to configure `sfptpd` to use the NTP daemon as the local reference clock:

- an single NTP sync module is instantiated using the `sync_module` option
- the key-id and key-value are set for authenticating with NTP using the `ntp_key` option.

8.5 Constraints

NTP has various performance limitations:

- The `ntpd` daemon disciplines only the system clock, and `sfptpd` disciplines the remainder of the clocks from the system clock.
- The `ntpd` daemon tends to slew the system clock rapidly for a short period of time every few minutes (e.g. 20).

  The result can be periodic errors between the system and NIC clocks, while the adapter clocks catch up with this rapid change in the system clock. The behavior will get better over time. For example, once the host has been up for 24 hours the NTP adjustments will be much smaller.
- The accuracy of NTP can vary massively. Solarflare recommend only using this mode with a GPS-connected NTP server.

9 Freerun Sync Module

9.1 Description

The Freerun sync module allows `sfptpd` to use a NIC clock as the local reference clock. This Solarflare adapter clock is a Stratum 3 compliant oscillator, accurate to 0.37 PPM per day.

9.2 Sync characteristics

This sync module provides low drift, but no synchronization to an external clock.

9.3 Configuration options

The Freerun sync module has instance-specific options that apply only to a single instance of the module.

Instance-specific configuration options

Table 15 shows the instance-specific configuration options for the Freerun sync module. These must be in a section labeled with the instance name.

Table 15: Instance-specific configuration options for the Freerun sync module

Option	Parameters	Description
interface	<interface-name>	Specifies the name of the interface hosting the local reference clock.
priority	<number>	Relative priority of sync module instance. Smaller values have higher priority. The default is 128.

9.4 Example configuration files

This section describes the example configuration files that use Freerun as the only sync module, or as the principal one. See Example configuration files on page 36.

NOTE: For descriptions of the remaining files in the `config` directory, see the chapters describing the other sync modules.

freerun.cfg

This file shows how to configure `sfptpd` to use the NIC clock as the local reference clock:

- a Freerun sync module is instantiated using the `sync_module` option.

9.5 Constraints

The Freerun sync module is not compatible with VLANs, because there is no incoming data.

10 Sfptpd Operation

10.1 Saved clock frequency correction data

For each Solarflare adapter clock and for the server system clock, `sfptpd` will save the current frequency correction value every 60 seconds. If the server is rebooted or if `sfptpd` is restarted, the last saved value is used to continue clock frequency adjustment rather than revert to a zero value which would delay re-synchronization to an external master clock (or re-synchronization to the LRC if the adapter clock is not the active clock receiving from the remote master clock).

Frequency correction files are saved in `/var/lib/sfptpd`. Refer to Table 6 on page 45 for details.

10.2 sfptpd in operation

The following charts show `sfptpd` performance when the Solarflare adapter is configured in PTP slave mode to a third party Grandmaster clock via a network switch. In this example Ethernet interface 4 (eth4) is the interface receiving PTP packets and `sfptpd` is disciplining the system clock.

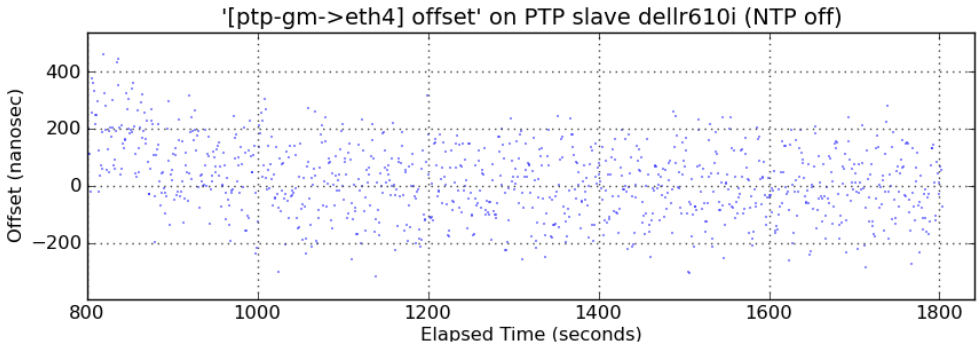


Figure 7: Offset of adapter's clock to the PTP master

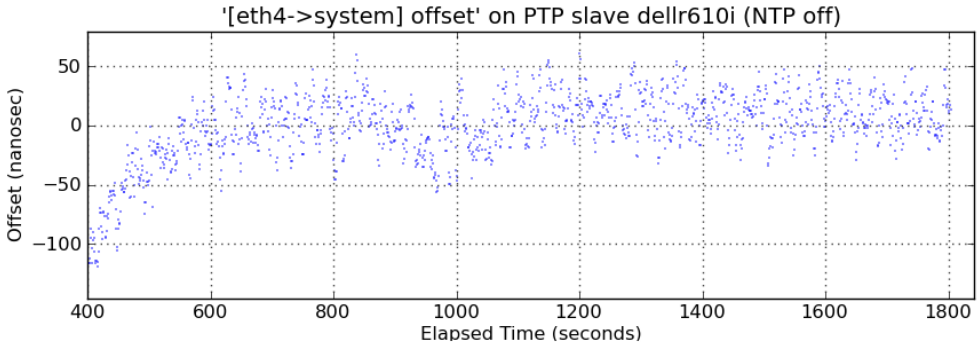


Figure 8: Offset of the system clock to the adapter's clock

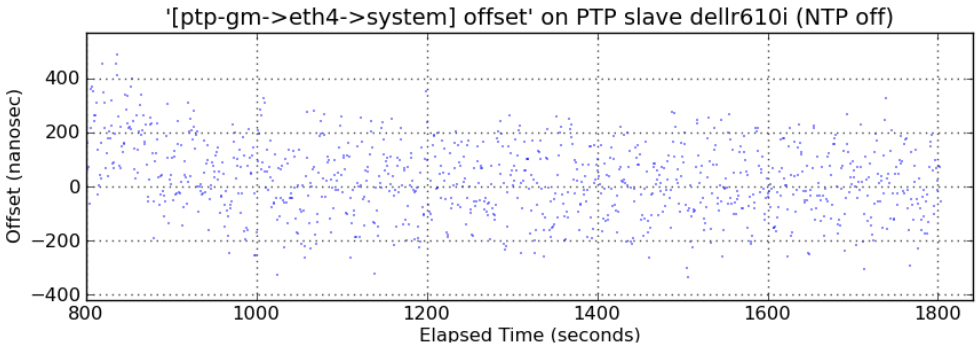


Figure 9: Offset of the server's system clock to the PTP master

The following charts show `sfptpd` performance when two Solarflare PTP adapters are present and configured in a bonded interface in a PTP slave server. Arrows on the chart identify what happens during bond failover and failback events. The upper chart shows the consistent offset of the server's system clock from the clock on the active port during repeated bond failover and failback events. For the same period, the middle chart shows the offset of the active adapter clock from the external grandmaster during the failover and failback of the bonded ports. For the same period the lower chart shows the offset of the server system clock from the external PTP master clock.

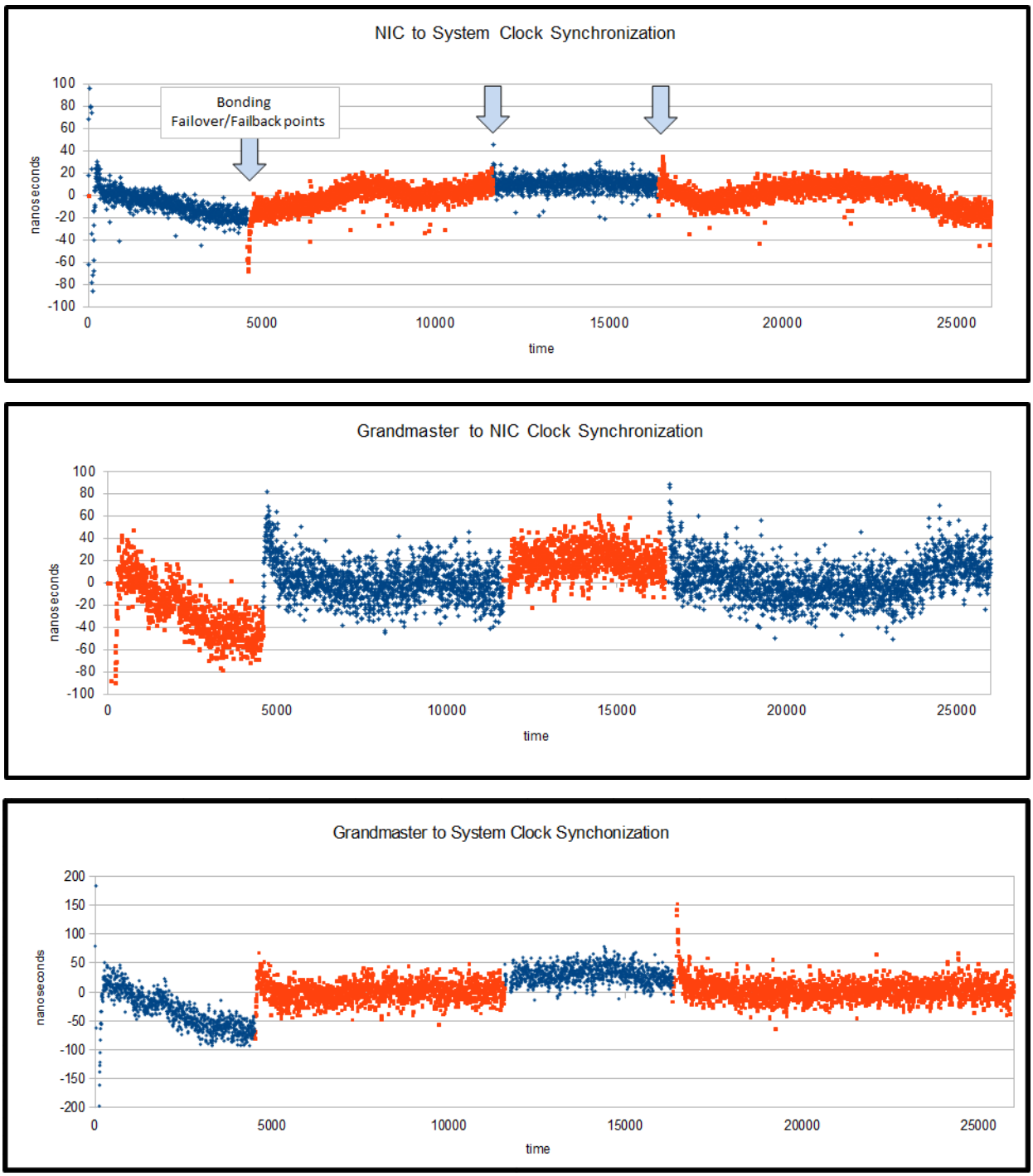


Figure 10: Bonding failover performance

10.3 Handling of leap seconds

The timestamps in PTP messages received from a master clock are in TAI (atomic) timescale. When a PTP message is received, `sfptpd` will first convert the atomic timestamp to a UTC timestamp for comparison with the internal UTC clock time. When a positive leap second is inserted in the UTC timescale, an additional second is inserted which effectively causes the UTC time to step backwards by one second. Following the leap second the UTC offset, currently 35 seconds, becomes 36 seconds.

```
UTC time =(TAI time - UTC offset)
```

The action applied when a leap second occurs will depend on the `sfptpd clock_control` configuration option enabled in the `sfptpd` configuration file.

The clock_control option determines the step/slew action applied to the Local Reference Clock (LRC) at all times, including when a leap second occurs. (The LRC is the clock that is currently being disciplined by `sfptpd`.)

Solarflare `sfptpd` is able to slew the clock at a maximum rate of 1 million ppb (1 part in a 1000) so slewing will take approximately 20 minutes to recover 1 second.

Corrective action is applied to the LRC. Other clocks in the server will be synchronized with the LRC.

Table 16: Leap second action

Clock_control Option	Leap second Action
slew-and-step	**Default**. Suspend clock synchronization, step the clock currently being disciplined backwards by one second and then recommence normal synchronization when the next announce message is received.
step-at-startup	Slew the clock to correct for leap second insertion.
no-step	Slew the clock to correct for leap second insertion.
no-adjust	Do not adjust clock.
step-forward	Will not step the clock when a positive leap second occurs, but will slew the clock to correct for leap second insertion.

10.4 Daemon control mechanism

`sfptpdctl` as a utility that can be used by to trigger housekeeping operations in the daemon.

Control commands

Table 17 shows the control commands:

Table 17: Control commands

Command	String for ctrl proto	equiv. signal	Description
Rotate logs	logrotate	SIGHUP	Closes and reopens all log files
Step clocks	stepclocks	SIGUSR1	Causes all clocks to be advanced or retarded to the time of the parent clock
Exit daemon	exit	SIGINT SIGTERM	Causes the application to exit without error
Select instance	selectinstance=<instance-name>	—	Selects the sync module instance when in manual selection mode. See Manual selection on page 21.

Control protocol

The client connects to the control server address. The version of the protocol is included in the address. The current version is 1 and the address is:

```
/var/run/sfptpd-control-v1.sock
```

The client will only succeed in connecting to the server if it has write permissions to the server socket and the server socket is created using the umask the server process inherited, therefore normally it will be necessary for the client to be run as root.

A command is sent as plain text in a Unix Domain datagram, for example:

```
<command string>
```

There is no return channel and it does not matter to what address, if any, the client's socket is bound.

Two example implementations of the client are included, both of which are suitable to invoke from the command line or to incorporate in custom scripts or operations code.

Standalone client

A standalone `sfptpdctl` client is supplied:

- In the tarball distribution, this client is located in the install directory.
- In the RPM distribution, this client is located in `/usr/sbin`.

To run, invoke with the desired control command:

```
sfptpdctl <command>
```

To select a sync module instance named `ptpdomain1`:

```
sfptpdctl selectinstance=ptpdomain1
```

Standalone client source code

C source code for the standalone client is supplied as `sfptpdctl.c`:

- In the tarball distribution, this source code is located in the `examples` subdirectory of the install directory.
- In the RPM distribution, this source code is located in the `/usr/share/doc/packages/sfptpd/examples` directory.

To build, run `make` in that directory.

Python client

A simple Python client is supplied as `sfptpdctl.py`:

- In the tarball distribution, this client is located in the `examples` subdirectory of the install directory.
- In the RPM distribution, this client is located in the `/usr/share/doc/packages/sfptpd/examples` directory.

System log rotation

The sfptpdctl logrotate command causes the sfptpd daemon to close and reopen the log file, creating the file if it does not already exist.

NOTE: This action does not rename the log file.

The sfptpdctl logrotate command could be called from the `postrotate` section of a `standard linux logrotate` configuration file. For example:

```
postrotate
    <path>/sfptpdctl logrotate || true
endscript
```

An alternative action would be to move/rename the current log file before invoking the sfptpdctl logrotate command.

11 Sfptpd Master Clock Use Cases

11.1 Master NTP mode

Using the Solarflare adapter as a PTP master clock in NTP mode (`ptp_master_ntp.cfg`), the local NTP client periodically synchronizes with a remote NTP server (configurable through the local NTP daemon). NTP is used to set the system clock and adapter clock and `sfptpd` keeps the system clock in sync with the adapter precision oscillator.

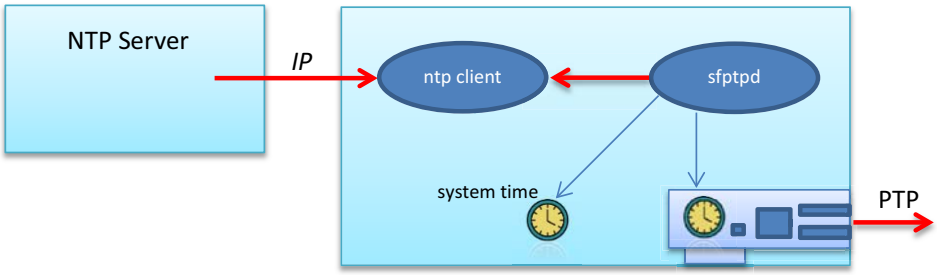


Figure 11: Master NTP mode

The `sfptpd` daemon synchronizes with the local NTP client every 1 second (default) and this is configurable in the `sfptpd` configuration file.

If the master is not the active PTP master, it will by default, revert to a PTP slave clock.

NOTE: When in master clock mode, sfptpd uses the UTC timescale. Ensure the ptp_utc_offset option in the sfptpd master configuration file is set to 0.

11.2 Standby master clock

It is quite common to deploy the Solarflare adapter in master clock mode as a secondary/backup master when the primary master is usually a dedicated PTP Grandmaster clock device having a time source reference i.e. GPS.

The Best Master Clock Algorithm (BMC) uses a hierarchical selection algorithm to select the active master clock based on properties in the periodic Announce messages sent by all master clocks. Clock properties are considered in the following order:

1. BMC priority 1
2. class
3. accuracy

4. variance
5. BMC priority 2
6. unique identifier (tie breaker)

In the `ptp_master_freerun.cfg` file the following clock properties are configurable (default values are shown):

```
# ptp_clock_priority1 128
# ptp_clock_priority2 128
```

Priority1 and priority2 values are in the range 0-255 (0 is the highest priority, 255 the lowest).

12 Performance

12.1 Tickless kernels and the nohz option

A feature of modern operating systems is that they use a “tickless” kernel which aims to reduce power consumption during kernel idle periods, and to increase performance:

- Kernel version 2.6 introduced this feature, stopping the regular timer tick on CPU cores which are idle. However, experiments at Solarflare have proven that this “tickless” mode degrades PTP performance, producing less consistent results than when the kernel always receives periodic timer ticks.

  PTP relies on the ability to accurately change the speed of the system clock by very small and precise amounts. The Linux kernel implements this adjustment to system clock rate with integer arithmetic, minimizing the error term to the target clock rate in every timer tick. However, when the timer tick doesn’t run, the error in tracking to the requested clock rate increases, and the system time diverges from the clock rate requested. When the system wakes from idle, the timer tick runs and the kernel corrects for the error term.

- Kernel version 3.10 introduced a new mode where only the boot CPU requires a tick. Experiments at Solarflare have shown this “full tickless” mode does not degrade PTP performance.

Whether the kernel operates in a tickless mode is configured by the `nohz` and/or `nohz_full` boot time options, with the majority of Linux distributions defaulting to a tickless kernel. To achieve the highest accuracy with PTP, Solarflare suggest configuring the kernel in one of the following ways:

- Do not run tickless, and so receive timer ticks even when the system is idle.

  This can be achieved by adding "`nohz=off`" to the kernel boot parameters in the `/boot/grub/grub.conf` file.

- Run in the “full tickless” mode (requires kernel version 3.10 or later).

  This can be achieved by adding "`nohz_full=<cpu_list>`" to the kernel boot parameters in the `/boot/grub/grub.conf` file, where <cpu_list> is a list of the CPU cores that are to use this mode (and cannot include the boot CPU).

  Alternatively, you can use the `CONFIG_NO_HZ_FULL_ALL=y` kernel configuration parameter to enable this mode for all CPUs except the boot CPU.

  You must also ensure that `sfptpd` runs on a dedicated full tickless CPU, using a CPU-affinitizing tool such as `taskset`.

12.2 Accuracy under network load

To obtain the highest accuracy the PTP protocol requires a network with constant latency. Standards such as “PTP boundary clock” and “PTP transparent clock” allow network switches to be PTP aware and measure latencies to allow the PTP end points to compensate for any variance in switching times for PTP packets. However, even with standard non-PTP aware switches, the two stage PTP synchronization approach used by the adapter can provide good accuracy under significant network load.

Solarflare has demonstrated slave to master offsets within 100ns on a lightly loaded network. However, even under bursty conditions of up to 80% 10G line rate, the same network demonstrated slave to master offsets of within 500ns. When the bursty condition cleared, the slave to master offsets returned to within 100ns.

Figure 12 shows PTP accuracy when used in an environment with bursty network load of up to 80% line rate.

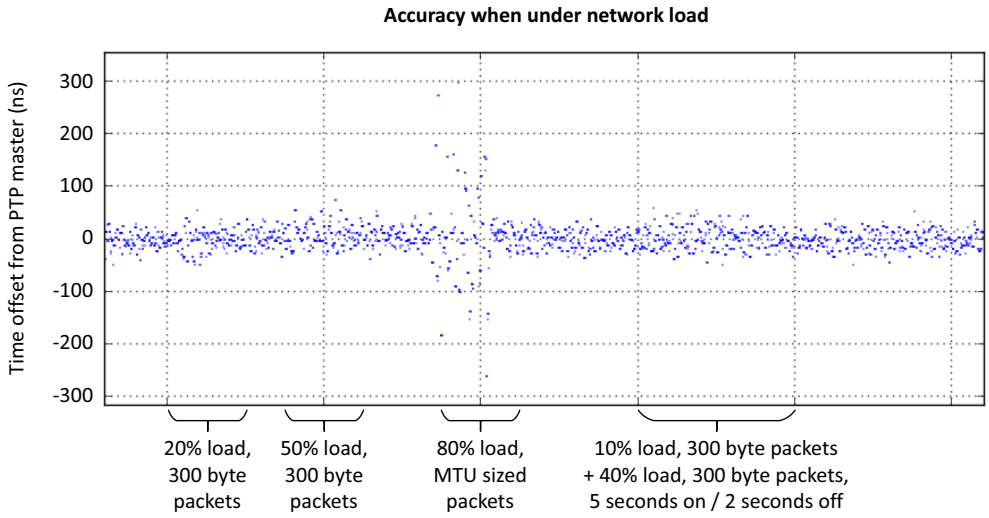


Figure 12: PTPd under load

13 PPS Measurements

13.1 Asymmetric networks

Asymmetric networks present a particular problem when attempting to account for network latency during PTP offset calculations between master and slave servers. PTP assumes symmetry in the network and the PTP protocol is not able to detect asymmetry in the network paths between master and slave.

Asymmetry can be present for a number of reasons including the store and forward delay in switches serving asymmetric networks as illustrated in Figure 13.

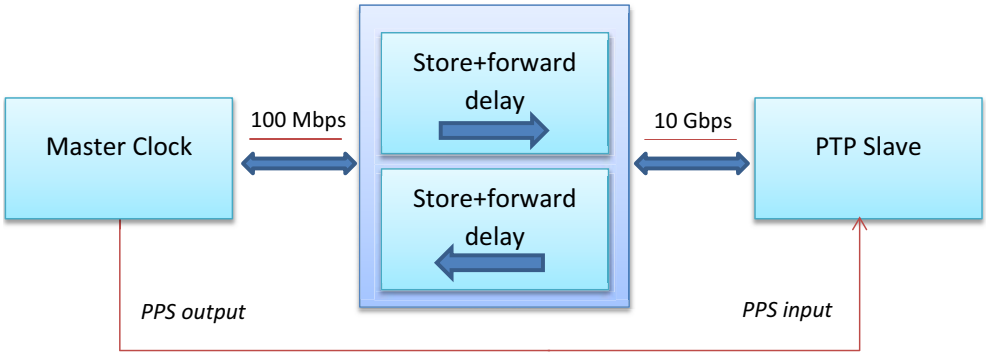


Figure 13: Asymmetric network

The result is that PTP offsets between master and slave will converge, but will be wrong by a constant offset from the master equal to half the asymmetry, for example

```
Transmit time master to slave:   5us
Transmit time slave to master:   1us
               One way delay:   (5+1)/2 = 3us
So 3us is added to the time offsets received from the master clock.
1PPS will display a mean offset value of 2us (5-3)
Actual asymmetry should be double this observed value i.e. 4us.
```

Measuring and adjusting for asymmetric latency

Solarflare timestamping adapters support 1PPS input/output interfaces[1] to allow asymmetry in the network to be measured. On a dedicated wire connection between master 1PPS output and slave 1PPS input, the master emits a single pulse every second. The leading edge of each pulse denotes the exact start of a one

1. The SFN6322F adapter is factory fitted with 1PPS I/O connectors. Other Solarflare adapters require an optional PPS bracket kit and cable assembly (product code SOLR-PPS-DP10G or SOLR-PPS-DP40G) available from Solarflare sales channels.

second period. When the leading edge of a pulse is detected by the slave adapter, firmware on the adapter is able to calculate the offset from its own 'start of second period'.

- If the initial observed mean 1PPS offset value is a negative value, it means the master→slave path is slower than the slave→master path, therefore the `ptp_rx_latency` configuration file option on the slave server is used to compensate the receive latency.
- If the initial observed mean 1PPS offset value is a positive value, it means the slave→master path is slower than the master→slave path, therefore the `ptp_tx_latency` configuration file option on the slave server is used to compensate the transmit latency.

This 1PPS calibration is only required once when configuring the network and need only be performed on one slave server in each network segment which share a common network path to the PTP master. There is no need for a permanent 1PPS connection to the Solarflare adapter. Refer to 1PPS in practice on page 92.

13.2 1PPS measurement procedure

1. `sfptpd` should be running between master and slave servers, and should be synchronized before the 1PPS value is measured and applied.
2. The master 1PPS output should be connected to a single slave 1PPS input.
3. On the slave server, for a short period e.g. 5 minutes, observe the 1PPS mean offset value from the `pps_off_mean` file to identify the mean offset value. Refer to 1PPS statistical data on page 94 for instructions on reading the 1PPS statistical data files.
4. On all slaves on the same network segment, configure `sfptpd` with knowledge of the mean 1PPS offset.
   - **If the initial observed 1PPS offset is a negative value, then all subsequent offsets should be added as positive values to the ptp_rx_latency option. The ptp_tx_latency option in this case should be zero.**
   - **If the initial observed 1PPS offset is a positive value, then all subsequent offsets should be added as positive values to the ptp_tx_latency option. The ptp_rx_latency option in this case should be zero.**
5. Continue to observe the 1PPS compensated mean offsets.
6. Repeat steps 3-5 adding or subtracting the 1PPS mean offset (doubled) value each time to the last applied value until the observed 1PPS mean value is as close to zero as possible.

1PPS asymmetric compensation examples

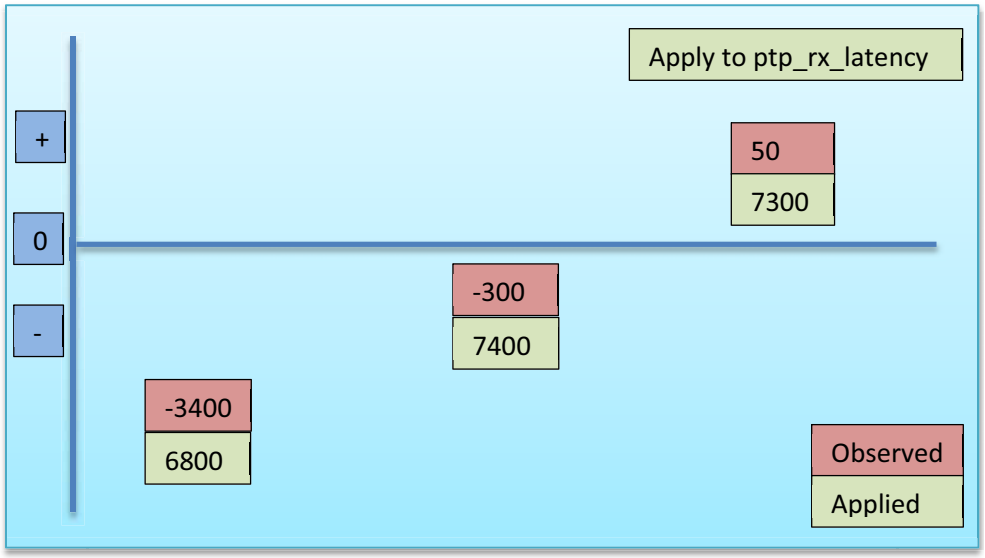


Figure 14: 1PPS example

In the above example the initial observed 1PPS offset is -3400. This value is doubled and applied to `sfptpd` using the ptp_rx_latency parameter as 6800.

`sfptpd` is restarted and the next observed 1PPS offset is -300, this value again is doubled and applied to the original compensation value (6800 + 600 = 7400).

`sfptpd` is restarted and the final observed 1PPS offset is +50 meaning the previous compensation value caused `sfptpd` to over-compensate, so the +50 is doubled and subtracted from the previous compensation value (7400 - 100 = 7300).

13.3 1PPS in practice

The following sections demonstrate the effect of applying the 1PPS mean offset value to `sfptpd`.

1PPS measurements on an asymmetric network

Figure 15 on page 92 shows the 1PPS offsets observed on an asymmetric network consisting of a third party grandmaster clock (100Mbps interface) and Solarflare 10Gbps (slave) adapter connected via a standard network switch.

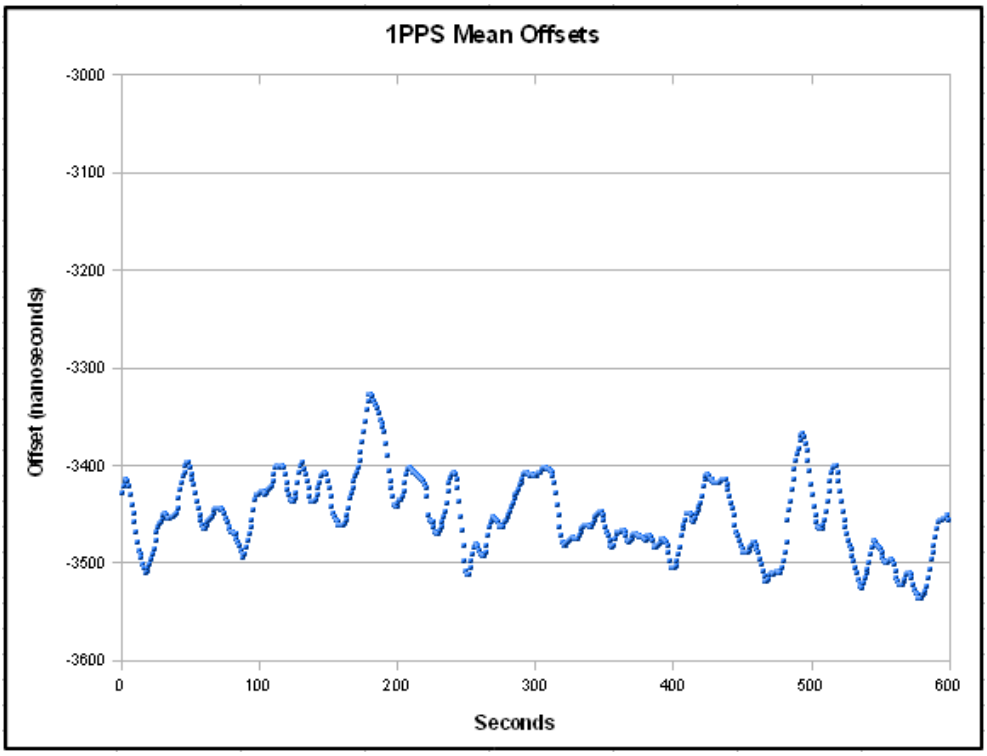


Figure 15: 1PPS measurement on an asymmetric network

Identifying the mean offset

In this particular instance the 1PPS offset is observed for a period of 10 minutes before the mean offset value is identified as `pps_off_mean: -3450`. Refer to 1PPS statistical data on page 94 for instructions on reading the 1PPS statistical data.

Applying the mean offset

The 1PPS mean offset value **should be doubled** and applied to `sfptpd` via the slave server configuration file as follows e.g.

```
ptp_rx_latency 6900 ptp_tx_latency 0
```

Figure 16 demonstrates 1PPS output after the (doubled) mean offset has been applied to `sfptpd`.

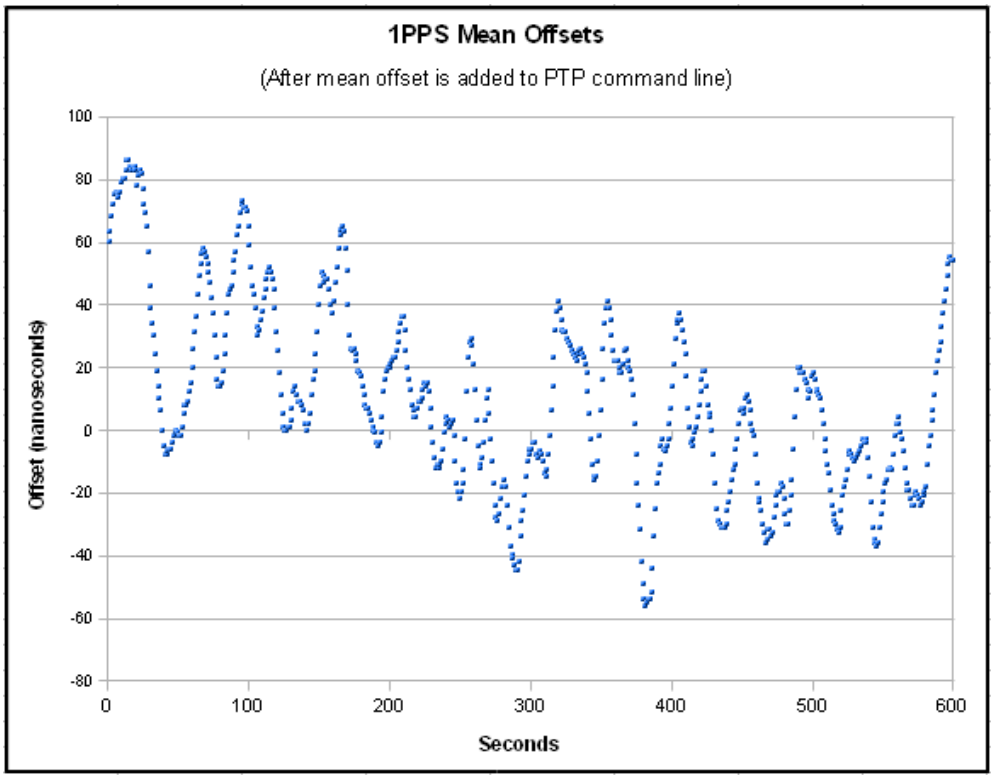


Figure 16: 1PPS measurement with offset

13.4 Solarflare sfptpd 1PPS I/O specification

- 1PPS-input: SMA
  - Rising edge active, TTL into 50Ω
- 1PPS-output: SMA
  - Rising edge on-time, TTL into 50Ω
- Pulse Width
  - 200ms high, 800ms low

13.5 1PPS statistical data

1PPS statistical counters and error data is available from the following files:

`/sys/class/net/eth<N>/device/pps_stats/<filename>`

where *<filename>* is one of the files listed in Table 18.

Two sets of data are provided in the form of 1PPS offsets (min, max, mean and last) and 1PPS periods (min, max, mean and last). **All measurements are in nanoseconds**.

Table 18: PPS statistics

File	Description
`pps_off_min`	Minimum offset value.
`pps_off_max`	Maximum offset value.
`pps_off_mean`	Average offset value observed over the last 8 values.
`pps_off_last`	Most recent offset value.
`pps_per_min`	Minimum 1PPS period.
`pps_per_max`	Maximum 1PPS period.
`pps_per_mean`	Average 1PPS period observed over the last 8 periods.
`pps_per_last`	Most recent 1PPS period.
`pps_oflow`	Too many 1PPS values received. Operation is suspended until the next `sfptpd` enable. This can occur when a cable is connected or as the result of a bad signal or noise on the 1PPS input. Re-start the `sfptpd` processes.
`pps_bad`	Very bad 1PPS period seen. The 1PPS period measured is too long to be a pulse per second i.e. period > 1 second. Check the 1PPS input is connected to a genuine 1PPS output.

Reset statistics counters

It is possible to reset 1PPS counters in the stats files by writing a '1' to the `ptp_stats` file relevant for the Solarflare interface.

```
echo 1 > /sys/class/net/eth<N>/device/ptp_stats
```

14 Monitoring

This section identifies the mechanisms sfptpd uses to generate message logs, state logs and statistical data logs.

14.1 Files

State/State Files

sfptpd generates state and statistical data in files in the /var/lib/sfptpd directory. These files are automatically generated whenever sfptpd is running - no additional configuration is required.

```
# ls /var/lib/sfptpd
freq-correction-000f:53ff:fe11:2233
freq-correction-000f:53ff:fe44:5566
freq-correction-system
interfaces
ptp-nodes
state-000f:53ff:fe44:5566
state-ntp
state-ptp1
state-system
stats-000f:53ff:fe44:5566
stats-000f:53ff:fe44:5566.json
stats-ntp
stats-ntp.json
stats-ptp1
stats-ptp1.json
stats-system
stats-system.json
topology
version
```

NOTE: Note: stats files in plain text and JSON formats are automatically generated for each sync module instance and local clock.

Message Log

By default sfptpd generates a message event log to stderr. The user can redirect the message log to file using the following option in the sfptpd config file.

```
message_log <path/filename>
```

Statistics Log

By default sfptpd generates a plain text stats log to stdout. The user can redirect the stats log to file using the following option in the sfptpd config file.

```
stats_log <path/filename>
```

The stats_log can also be generated in JSON-lines format - see Machine-Readable Real-Time Stats below.

Statistics Log - Journalctl

When running sfptpd as a service on Linux system supporting systemd, real-time stats_log output can be viewed in the journal:

```
# journalctl -fu sfptpd
```

PTP Packet Dump

The sfptpd configuration file ptp_pkt_dump option captures/displays all PTP packets sent/received by the sfptpd process.

CAUTION: It is strongly recommended NOT to run with the ptp_pkt_dump option enabled for extensive periods as this produces a lot of additional data. **This option should be used for specific debug purposes only.**

TCPDUMP Packet Capture

Packet capture applications such as tcpdump can be useful to identify if messages reported missing by sfptpd are actually being received at the PTP interface.

The following example identifies how to use tcpdump to capture only PTP traffic.

```
# tcpdump -i <interface> dst port 319 or dst port 320 [-w <filename.pcap>]
```

The interface is the Solarflare adapter interface being used by sfptpd.

- Keep pcap files as small as possible, but large enough to capture an instance of the issue sfptpd is experiencing.

NOTE: tcpdump captures packets sent/received on an interface - but these packets may be not be delivered to the sfptpd daemon when they are prevented by other factors such as iptables rules, firewalls or reverse-path filters.

14.2 Machine-Readable Real-Time Stats

Real-time statistical data are generated in JSON-lines format that is both human-readable and machine-readable. The jsonl format files are the same data as generated by the sfptpd stats_log plain text file.

NOTE: With some standard JSON viewer applications it may be necessary to convert the JSON-lines format .jsonl to .json formatted files.

Enable

Add or enable the following option in the [general] section of the sfptpd config file:

```
json_stats <path><filename>.jsonl
```

The stats_log is a JSON-lines formatted file (jsonl) with one stats log record per line and where each record is a JSON object. The following extract is an example:

```
{
  "instance":"ptp1",
  "time":"2017-09-05 15:34:03.702499",
  "clock-master":{"name":"gm"},
  "clock-slave":
  {
    "name":"phc0(enp4s0f0/enp4s0f1)",
    "time":"2017-09-05 15:34:03.702669635",
    "primary-interface":"enp4s0f0"
  },
  "is-disciplining":true,
  "in-sync":true,
  "alarms":[],
  "stats":
  {
    "offset":7.500000,
    "freq-adj":-1150.850663,
    "one-way-delay":37.500000,
    "parent-id":"000f:53ff:fe43:2740",
    "gm-id":"000f:53ff:fe43:2740",
    "active-interface":"enp4s0f0",
    "p-term":1.500000,
    "i-term":1149.373163
  }
}
```

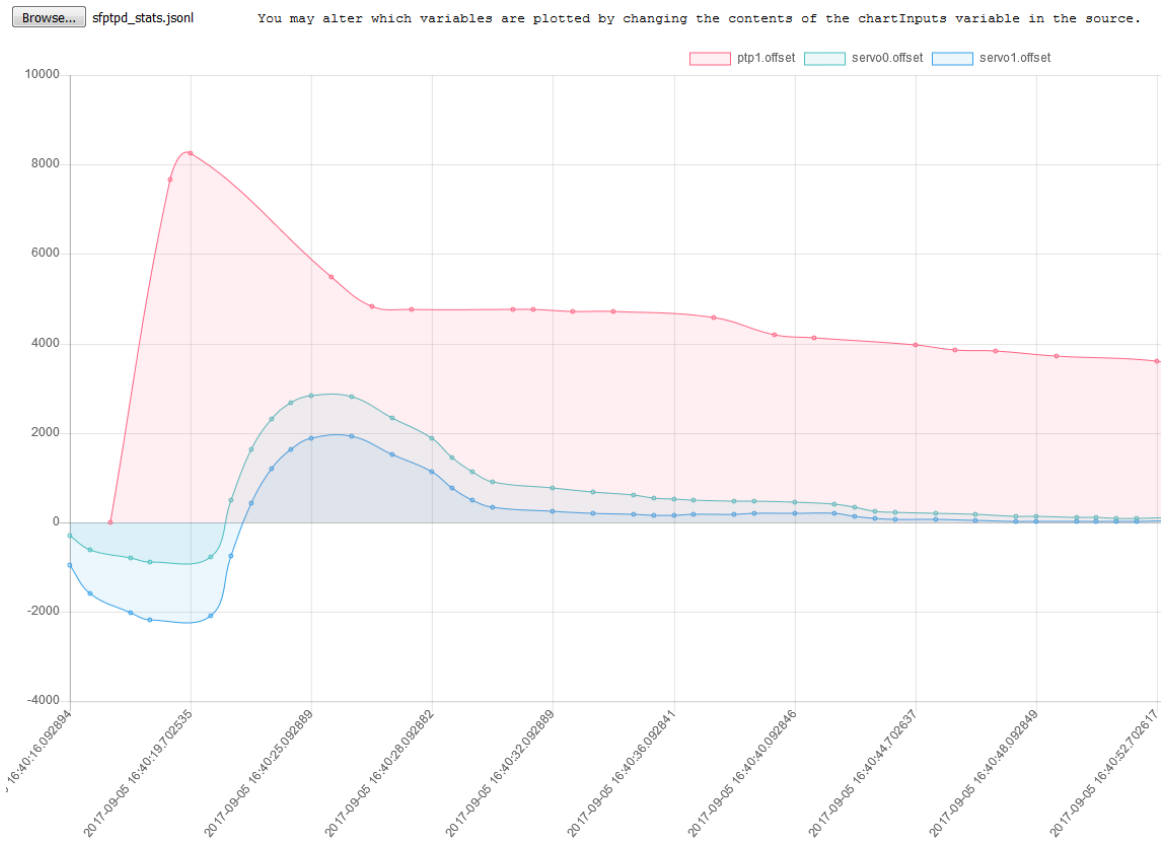
Viewing JSON Format Stats

The JSON-lines formatted stats_log can be read by any standard web browser JSON viewer application. An example JSON viewer HTML application is included with the sfptpd distribution:

`/sfptpd-3.2.1.1004/examples/sfptpd_json_parse.html`

The JSON viewer is able to parse the stats_log.jsonl file in real-time, i.e. while sfptpd is writing to the file, however it is recommended that the stats_log file is rotated periodically to keep files to a manageable size.

The following is a screen-shot taken from the supplied sfptpd_json_parse.html viewer.



The user is able to select measurements to be displayed by the sfptpd_json_parse viewer by editing fields in the html file.

14.3 Machine-Readable Long-Term Stats

Long-term statistical data is available in the /var/lib/sfptpd directory where there is a stats file for every sync_module or local clock e.g.

```
stats-000f:53ff:fe44:5566
stats-000f:53ff:fe44:5566.json
stats-ntp
stats-ntp.json
stats-ptp1
stats-ptp1.json
stats-system
stats-system.json
```

Plain text and JSON formatted files are automatically generated by sfptpd and can be read by any standard JSON viewer application. Text files and JSON files contain exactly the same data.

Long-term stats files are updated every 60 seconds and retain stats data for up to 3 weeks. Stats files include counters for all ALL PTP messages sent/received by sfptpd.

Enable

Automatically generated by sfptpd - no configuration is required. The following is an extract from the JSON formatted stats file:

```
"name": "announce-pkts-rxed",
"type": "count",
"values": [
        {
                "period": "minute",
                "period-secs": 60,
                "seq-num": 15,
                "samples": 1169,
                "total": 30,
                "start-time": "2017-09-05 17:30:40",
                "end-time": "2017-09-05 17:31:40" },
        {
                "period": "minute",
                "period-secs": 60,
                "seq-num": 14,
                "samples": 1170,
                "total": 30,
                "start-time": "2017-09-05 17:29:40",
                "end-time": "2017-09-05 17:30:40" },
        {
                "period": "minute",
                "period-secs": 60,
                "seq-num": 13,
                "samples": 1171,
                "total": 30,
                "start-time": "2017-09-05 17:28:40",
                "end-time": "2017-09-05 17:29:40"
```

The sequence number identifies successive measurement periods.

14.4 Remote Reporting of Real-Time Stats

The IEEE-1588 draft specification proposes a PTP message-based mechanism for application-independent monitoring of the timing information from PTP nodes.

Solarflare sfptpd supports the draft standard and the sfptpd PTP slave will include the optional three data sets TLVs in PTP Signaling messages (message type 0xc) sent from the sfptpd slave server to a remote monitoring station.

- SLAVE_RX_SYNC_TIMING_DATA TLV

The slave will report the timestamp for received Sync messages.

- SLAVE_RX_SYNC_COMPUTED_DATA TLV

The slave will report the calculated offset and one-way-delay values.

- SLAVE_TX_EVENT_TIMESTAMPS TLV

The slave will report the timestamp for sent Delay_Request messages.

For further details of the IEEE draft specification users should refer to the specification documentation:

NOTE: The draft IEEE-1588 specification document is not yet published. The sfptpd implementation follows the draft proposal, but may be revised to meet any changes in the released standard.

Enable - Monitored Node

On the PTP slave server to be monitored, the following options should be enabled in the PTP generic section of the sfptpd config file:

```
# PTP Generic Configuration
#
[ptp]
mon_monitor_address <address of remote monitoring station interface>
mon_rx_sync_timing_data
mon_rx_sync_computed_data
mon_tx_event_timestamps
```

If address is omitted, PTP Signaling messages from the monitored slave will be multicast.

Enable - Monitoring Station

A server equipped with a Solarflare PTP enabled adapter can be configured in remote monitor mode to received data from all monitored slave servers.

1. Run sfptpd-v3.2.1.
2. Create the required sync instances in the general header section:

```
[general]
sync_module ptp1  ptp_all_monitor
json_remote_monitor /tmp/remote.jsonl
```

In the above example two instances are created.

- [ptp1] is an optional normal PTP slave instance.
- [ptp_all_monitor] is an optional monitor only instance.
- The file in which monitored data is to collected is also specified.

3 In the PTP generic section:

```
[ptp]
remote_monitor
```

With the remote_monitor option enabled, all configured PTP instances are able to monitor received PTP signaling messages.

4 In the PTP instance (ptp1) section:

```
[ptp1]
ptp_mode slave
ptp_domain 10
interface <interface>
```

The [ptp1] instance is a normal PTP slave, but it will also monitor PTP signaling messages from domain 10 received on the specified interface.

5 In the PTP instance (ptp_all_monitor) section:

```
[ptp_all_monitor]
ptp_mode monitor
ptp_domain 127
interface <interface>
```

The parameters in the [ptp_all_monitor] section identify this as a ptp instance in ‘monitor’ mode. The instance is assigned a **unique** ptp_domain value and it will monitor PTP signaling messages from all domains on the specified interface.

ptp_mode monitor is a passive mode. The instance is not a ptp slave and not a ptp master.

Output

The following is an extract from the real-time stats log emitted from the slave reporting the TLVs for a received Sync message.

```
{ "rx-event": {"monitor-seq-id": 55,
"monitor-timestamp": "2017-09-06 13:58:29.911882",
"node": "000f:53ff:fe21:9bb0.2",
"parent-port": "000f:53ff:fe43:2740.1",
"sync-seq": 9817,
"offset-from-master": 2920.500000,
"mean-path-delay": 7.500000,
"sync-ingress-timestamp": 1504702709.911463074 } }
```

The table provides a description of the reported fields: from the **rx-event**.

Field	Description
monitor-seq-id	Sequence number
monitor-timestamp	System clock timestamp
node	Identifies the slave interface clock UUID
parent-port	Identifies the master clock UUID
sync-seq	Sequence number of the received Sync message
offset-from-master	Current offset from master clock
mean-path-delay	Current one-way-delay value
sync-ingress-timestamp	Ingress timestamp generated when the Sync message was received by the adapter.

The following is an extract from the real-time stats log emitted from the slave reporting the TLVs for a transmitted Delay_Request message.

```
{ "tx-event": {"monitor-seq-id": 0,
"monitor-timestamp": "2017-09-06 13:57:43.332403",
"node": "000f:53ff:fe21:9bb0.2",
"source-port": "000f:53ff:fe21:9bb0.2",
"message-type": "Delay_Req",
"event-seq-id": 11,
"egress-timestamp": 1504702403.475124702 } }
```

The table provides a description of the reported fields: from the **tx-event**.

Field	Description
monitor-seq-id	Sequence number
monitor-timestamp	System clock timestamp
node	Identifies the slave interface clock UUID
source-port	Identifies the slave clock port that sent this message
message-type	Delay_Request message
event-seq-id	Sequence ID of the sent Delay_Request
egress-timestamp	Timestamp generated by the adapter when the Delay_Request was sent.

14.5 Remote Reporting of Slave State

The standards based slave event monitoring described in Remote Reporting of Real-Time Stats above makes no provision for reporting clock alarm and clock status information.

Solarflare sfptpd supports an additional organization extension TLV to provide for the reporting of alarms, state and state-change events to also be exported to a remote monitoring node.

When this feature is enabled, the PTP slave server alarms and state changes are sent to the remote monitoring node.

Enable

On the PTP slave server to be monitored, the following options should be enabled in the PTP generic section of the sfptpd config file:

```
# PTP Generic Configuration
#
[ptp]
mon_monitor_address <address of remote monitoring node interface>
mon_slave_status
```

When an address is specified, sfptpd will send unicast data. If the address is omitted, the sfptpd slave will multicast the data.

Output

```
{ "slave-status": {"monitor-seq-id": 0,
"monitor-timestamp": "2017-09-06 13:53:18.663545",
"node": "000f:53ff:fe41:c700.1",
"gm-id": "0000:0000:0000:0000.0",
"state": "DISABLED",
"bond-changed": false,
"selected": false,
"in-sync": false,
"msg-alarms": [],
"alarms": []} }

{ "slave-status": {"monitor-seq-id": 1,
"monitor-timestamp": "2017-09-06 13:53:18.663563",
"node": "000f:53ff:fe21:9bb0.2",
"gm-id": "0000:0000:0000:0000.0",
"state": "LISTENING",
"bond-changed": false,
"selected": true,
"in-sync": false,
"msg-alarms": [],
"alarms": []} }

{ "slave-status": {"monitor-seq-id": 2,
"monitor-timestamp": "2017-09-06 13:53:21.538047",
"node": "000f:53ff:fe21:9bb0.2",
```

```
"gm-id": "0000:0000:0000:0000.0",
"state": "SLAVE",
"bond-changed": false,
"selected": true,
"in-sync": false,
"msg-alarms": [],
"alarms": []} }
```

14.6 Remote Monitor

A simple example remote monitor (python) script is included in the

`sfptpd-<version>/examples/monitoring_console.py`

14.7 Meinberg NetSync Monitor

The Meinberg NetSync Monitor capability, supported on Meinberg LANTIME and IMS models, allows a remote Monitoring System (MS) to probe downstream PTP slave devices using standard PTP messages. NetSync will collect timing data from multiple downstream slave devices and compare these to the Meinberg master clock timesource so users no longer have to rely on 'self-reported sync accuracy' from slave devices. An additional MTIE TLV enables the verification of compliance with specified MTIE time deviation limits.

NetSync Monitor uses a mechanism of 'Reverse PTP' sending standard PTP messages with TLVs attached between the MS and slave devices. Multiple MS can be deployed on a network.

Solarflare sfptpd supports the NetSync feature, recognizes the probe TLVs sent by the MS device and will operate the reverse PTP protocol to cooperate with the Monitoring Station.

Enable

Add the following option to the [general] section of the sfptpd config file:

```
mon_meinberg_netsync
```

Output

For further details of NetSync Monitor reports and output formats refer to the following documentation:
https://www.meinbergglobal.com/download/docs/sw/english/netsyncmonitor.pdf

14.8 Rotate Log Files

The sfptpdctl logrotate command causes the sfptpd daemon to close and reopen the log file,creating the file if it does not already exist.

NOTE: This action does not rename the log file.

The sfptpdctl logrotate command could be called from the `postrotate` section of a standard Linux `logrotate` configuration file. For example:

```
postrotate
    <path>/sfptpdctl logrotate || true
endscript
```

An alternative action would be to move/rename the current log file before invoking the sfptpdctl logrotate command.

15 Known Issues and Limitations

15.1 Firmware upgrade

CAUTION: The `sfptpd` daemon must be terminated before upgrading the adapter firmware. Following firmware upgrade the adapter driver should be reloaded.

- If onload is installed:

  ```
  # onload_tool reload
  ```

- If onload is not installed:

  ```
  # modprobe -r sfc
  # modprobe sfc
  ```

CAUTION: The `ethtool -t` command is disruptive, and should not be used while `sfptpd` is running. It takes the interface offline, and so interrupts service to it. It also resets the adapter clock back to the UTC epoch.

15.2 PTP and SolarCapture

When using Solarflare SolarCapture to capture packets from an interface also being used to send/receive PTP messages, PTP hybrid mode will not function correctly when SolarCapture consumes the ARP response messages. This prevents the unicast Delay_Request messages being sent from the PTP slave. `sfptpd` in multicast mode is not affected and users are advised to select multicast mode in the `sfptpd` configuration file.

```
ptp_network_mode multicast
```

Refer to the *SolarCapture User Guide* configuration options when using SolarCapture and `sfptpd` on the same server.

15.3 Bonding

Using the Linux bonding driver, `sfptpd` supports LACP and active/passive bonding. There are limitations to combining PTP and non-PTP ports in the same bond:

- for LACP bonding, a combination of PTP and non-PTP ports does not work
- for active/passive bonding, the quality of convergence depends on whether the bond supports timestamping.

A How PTP Works

This section provides a basic description of how the PTP protocol operates between a master and a slave server. For a complete description of the PTP protocol refer to the IEEE 1588-2008 Standard for a Precision Clock.

A.1 Message sequence

The following diagram describes the PTP protocol message sequence which must occur for master and slave servers to synchronize.

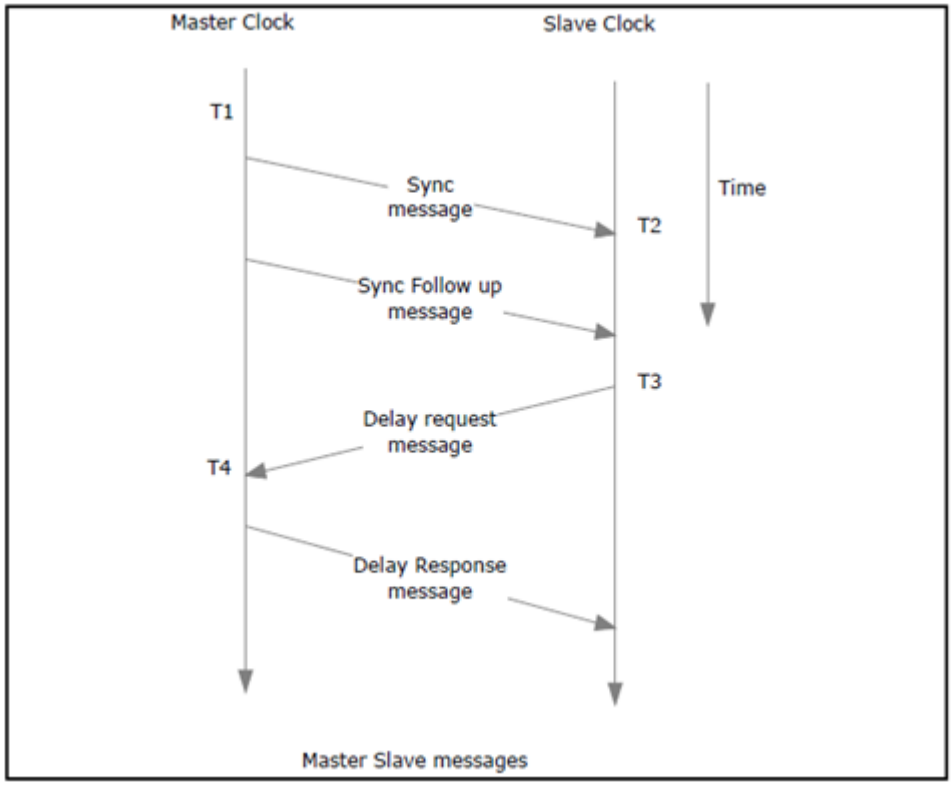


Figure 17: PTP message sequence

The **Sync message** is multicast to all slaves at a fixed interval of between 1 and 64 messages per second, configurable by the master clock. On most PTP networks a sync interval of between 1-4 sync messages per second is sufficient to ensure accurate synchronization and increasing the sync interval does not always result in greater accuracy of synchronization. The Sync message contains the time the message was transmitted (T1). The slave generates a hardware timestamp (T2) when the message is received.

The **Follow_up message** is sent immediately following every Sync by master clocks using 2-step synchronization. The Follow_up message contains the actual time the preceding Sync message was sent. A master clock using 1-step synchronization does not transmit the Follow_up message.

When the slave has received the Follow_up message (or just Sync message in the case of 1-step synchronization) it will generate a **Delay_Request message**. When this message is sent the slave generates and retains a hardware timestamp (T3).

The master will record the time the Delay_Request is received (T4) and this timestamp is then relayed back to the slave in the **Delay_Response message**.

Using the timestamp information derived from the message sequence, the slave is able to calculate the one-way-delay between slave and master clocks and the time offset from the master clock.

```
one_way_delay=((T2-T1) + (T4-T3)) / 2
offset=((T2-T1) - (T4-T3)) / 2
```

A.2 One-way-delay interval

The interval between Delay_Request messages is determined by the master clock. A parameter of the Delay_Response message from the master is the logMessagePeriod. This is a power of 2 value that defines a send window period designed to ensure (1) that multiple slaves send at random intervals during the period, (2) that the master clock is able to respond to Delay_Requests from multiple slaves without queuing these messages.

```
window = (2^logMessagePeriod)
```

So a logMessagePeriod of 3:

```
window = (2^3) = 0-8 second window
```

Solarflare `sfptpd` will allow the slave to override the logMessagePeriod using the config file option `ptp_delayreq_interval` causing the `sfptpd` slave to send Delay_Request messages at a fixed interval.

A.3 Announce message

Another message periodically generated by the master clock is the **Announce message** which contains data advertising the master clock type, accuracy and priority levels. The Announce message is used by the Best Master Clock algorithm to determine the most accurate master clock on a PTP network.

If a PTP sync module expects to receive an Announce messages, but does not do so, it outputs a warning message:

```
2016-12-21 11:47:19.652778: warning: failed to receive Announce within 12.000 seconds
```

Refer to Missing Delay_Response Messages on page 119

A.4 Solarflare sfptpd 2-stage synchronization

The PTP messages are used by `sfptpd` to synchronize the adapter clock with the master clock. `sfptpd` runs a second clock servo to synchronize the system clock to the adapter clock as illustrated by Figure 18 on page 109. This unique two stage synchronization has a number of benefits including:

- Improved accuracy with the ability to more frequently discipline the system clock than is supported by standard PTP masters.
- Ability to discipline the system clock to a high precision clock during periods, for whatever reason, whereby the upstream PTP master is inaccessible or offline.

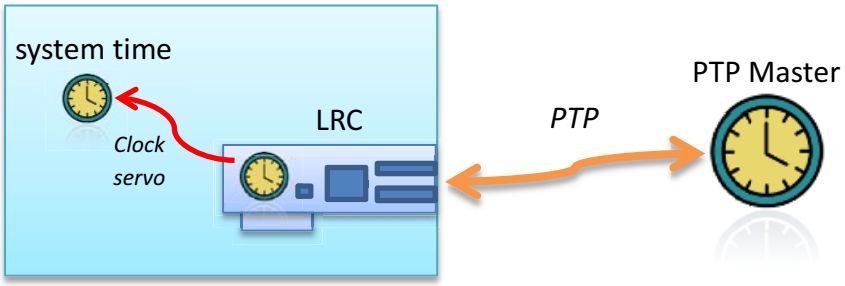


Figure 18: sfptpd 2-stage synchronization

B Automatic Startup

B.1 Using sfptpd under systemd control

The following procedure is an example of how to run sfptpd under systemd control on RHEL7, CENTOS7 or Ubuntu (15.04 and later).

Running sfptpd as a service requires a UNIT file so the sfptpd service can be managed by systemctl and automatically restarted following server reboot.

Using the Tarball Package: SF-108910-LS

Create the UNIT file

Create the sfptpd.service unit file as /usr/lib/systemd/system/sfptpd.service

```
[Unit]
Description=sfptpd
DefaultDependencies=true
Requisite=NetworkManager.service
After=NetworkManager.service
Wants=network-online.target
After=network-online.target

[Service]
ExecStart=/usr/sbin/sfptpd -f/<path>/ptp_slave.cfg

[Install]
WantedBy=multi-user.target
```

The above example ensures that sfptpd is not started until after the NetworkManager service. Waiting also for the network-online.target should ensure that all network interfaces are configured and in an ‘up’ state before sfptpd is started. The sfptpd interface can be specified in the ptp_slave.cfg file.

NOTE: The full path is required to the sfptpd config file.

NOTE: The sfptpd interface must be configured with an IP address and must be in an ‘up’ state before sfptpd can be run.

NOTE: Run ‘systemctl daemon-reload’ after changes are made to the unit file.

Locate the sfptpd binary

Copy the sfptpd binary into the directory where it will run from:

```
/usr/sbin/sfptpd
```

Enable required services

```
# systemctl enable NetworkManager
# systemctl enable sfptpd
```

```
Created symlink from /etc/systemd/system/multi-user.target.wants/
sfptpd.service to /usr/lib/systemd/system/sfptpd.service.
```

- The enable command will create the symbolic links from the systems copy of the unit file (usr/lib/systemd/system/sfptpd.service) to the location where systemd looks for autostart files (/etc/systemd/system.<target>.target.wants).
- The enable command also means that the NetworkManager service and sfptpd service will be automatically started following server reboot.

To prevent automatic startup following reboot, disable the service:

```
# systemctl disable sfptpd
```

This Failed message, when seen after trying the enable command, usually indicates a syntax error in the unit file:

```
Failed to execute operation: Bad message
```

Configure the sfptpd interface

Users should make sure the sfptpd interface is configured to meet network requirements. One method is to create the interface config file which is picked up by the NetworkManager service from the network-scripts directory:

```
# cat /etc/sysconfig/network-scripts/ifcfg-enp4s0f0
DEVICE="enp4s0f0"
TYPE="Ethernet"
BOOTPROTO="none"
ONBOOT="yes"
HWADDR="00:0f:53:1a:3b:4c"
IPADDR="172.11.123.123"
PREFIX=24
DEFROUTE="yes"
UUID="04426332-5e84-45fd-ad70-17c8790c99a9"
```

Check service status

Following server reboot - check sfptpd status:

```
# systemctl status sfptpd
â sfptpd.service - sfptpd
   Loaded: loaded (/usr/lib/systemd/system/sfptpd.service; enabled; vendor
preset: disabled)
   Active: active (running) since Mon 2017-08-07 12:32:25 BST; 14min ago
  Process: 941 ExecStartPre=/bin/sleep 5 (code=exited, status=0/SUCCESS)
 Main PID: 1101 (sfptpd)
   CGroup: /system.slice/sfptpd.service
           ââ1101 /usr/sbin/sfptpd -f/tmp/sfptpd-3.0.1.1004.x86_64/config/
ptp_slave.cfg
```

Identify startup sequence problems

If sfptpd fails to start or indicates that the PTP interface is not in an 'up' state:

- check systemctl status for information:

```
# systemctl status sfptpd
```

- examine the system log for startup sequence/issues:

```
# cat /var/log/messages | grep -E 'enp4s0f0|NetworkManager|sfptpd'
```

Replace *enp4s0f0* with the sfptpd interface.

Realtime sfptpd stats log

When sfptpd is run as a service with the default stats_log setting of stdout:

```
stats_log stdout
```

The stats log output can be viewed in realtime from journalctl:

```
# journalctl -fu sfptpd
```

Using the init.d script

- Installing the binary RPM package (SF-113122-LS) will place the sfptpd init.d script into the following directory:

```
/etc/rc.d/init.d
```

- The default sfptpd.conf file is in directory:

```
/etc
```

- All installed files can be located from the RPM list:

```
# rpm -qlp sfptpd-3.0.1.1004-1.x86_64.rpm
warning: sfptpd-3.0.1.1004-1.x86_64.rpm: Header V4 RSA/SHA256 Signature,
key ID ca969f38: NOKEY
/etc/init.d/sfptpd
/etc/sfptpd.conf
/usr/sbin/sfptpd
/usr/sbin/sfptpdctl
/usr/share/doc/packages/sfptpd
/usr/share/doc/packages/sfptpd/LICENSE
/usr/share/doc/packages/sfptpd/PTPD2_COPYRIGHT
/usr/share/doc/packages/sfptpd/config
/usr/share/doc/packages/sfptpd/config/convert_config_to_v3.py
/usr/share/doc/packages/sfptpd/config/default.cfg
/usr/share/doc/packages/sfptpd/config/freerun.cfg
/usr/share/doc/packages/sfptpd/config/many_instances.cfg
/usr/share/doc/packages/sfptpd/config/ntp.cfg
/usr/share/doc/packages/sfptpd/config/pps_slave.cfg
/usr/share/doc/packages/sfptpd/config/ptp_boundary.cfg
/usr/share/doc/packages/sfptpd/config/ptp_domain_bridge.cfg
/usr/share/doc/packages/sfptpd/config/ptp_master_freerun.cfg
/usr/share/doc/packages/sfptpd/config/ptp_master_ntp.cfg
/usr/share/doc/packages/sfptpd/config/ptp_slave.cfg
/usr/share/doc/packages/sfptpd/config/ptp_slave_multiple.cfg
```

```
/usr/share/doc/packages/sfptpd/config/ptp_slave_ntp_fallback.cfg
/usr/share/doc/packages/sfptpd/examples
/usr/share/doc/packages/sfptpd/examples/Makefile.sfptpdctl
/usr/share/doc/packages/sfptpd/examples/README.sfptpdctl
/usr/share/doc/packages/sfptpd/examples/sfptpdctl.c
/usr/share/doc/packages/sfptpd/examples/sfptpdctl.py
```

Configure the sfptpd interface

Edit the default configuration file - or copy one of the example configuration files to the /etc directory and add the sfptpd interface.

Enable the sfptpd service

```
# systemctl enable sfptpd

sfptpd.service is not a native service, redirecting to /sbin/chkconfig.
Executing /sbin/chkconfig sfptpd on
```

The enable command means that the sfptpd daemon will be automatically started following reboot.

Useful systemd commands

```
# systemctl status sfptpd

# systemctl is-active sfptpd

# systemctl is-enabled sfptpd

# systemctl is-failed sfptpd

# systemctl list-units
```

list-units only displays units that systemd has attempted to parse and load into memory.

To see all – including the units it did not attempt to parse/load:

```
# systemctl list-unit-files
```

C Troubleshooting Guide

C.1 Description

The following sections identify diagnostic procedures and sfptpd features useful when debugging sfptpd problems and when reporting issues to support@solarflare.com.

sfreport

sfreport is a Solarflare diagnostic (perl) script that collects information about the host server and installed Solarflare adapters.

The script will generate a HTML output file which should be returned to Solarflare support when reporting an issue. sfreport is part of the Solarflare Linux Diagnostics package (SF-108317-LS) available from support@solarflare.com.

sfreport is non-intrusive, but can be run out-of-hours on production systems:

```
# perl sfreport.pl
```

PTP Packet Dump

The sfptpd configuration file ptp_pkt_dump option captures all PTP packets sent/received by the sfptpd process. This is different from tcpdump which captures packets sent/received at the adapter interface.

The `ptp_pkt_dump` option is useful for:

- Checking that PTP packets are actually being sent/received by sfptpd. (counters for all messages types sent/received are accumulated in the /var/lib/sfptpd/stats-<clock id> files.
- Debugging badly formatted PTP packets.

CAUTION: It is strongly recommended NOT to run with the ptp_pkt_dump option enabled for extensive periods as this produces a lot of additional data. **This option should be used for debug purposes only.**

TCPDUMP Packet Capture

Packet capture applications such as tcpdump can be useful to:

- Identify if any PTP messages are actually being received on the PTP interface.
- Identify if messages reported missing by sfptpd are actually being received at the PTP interface.
- To inspect individual fields in PTP messages.

The following example shows how to use tcpdump to capture only PTP traffic.

```
# tcpdump -i <interface> dst port 319 or dst port 320 [-w <filename.pcap>]
```

The interface is the Solarflare adapter interface being used by sfptpd.

- Keep pcap files as small as possible, but large enough to capture an instance of the issue sfptpd is experiencing.

NOTE: tcpdump captures packets sent/received on an interface - but these packets may be not be delivered to the sfptpd daemon when they are prevented by other factors such as iptables rules, firewalls or reverse-path filters.

PTP Alarms

The following list identifies all PTP alarms generated by sfptpd. Active alarms will be present in the /var/lib/sfptpd/topology file and in state files within this directory.

no-sync-pkts

PTP Sync packet(s) are not being received. From a tcpdump pcap file, identify if packets are actually being received at the PTP interface.

Sync/FollowUp pairs have sequential sequence ID values. If some or all Sync packets are missing, the user should check these are being generated by the upstream master clock.

no-follow-ups

PTP FollowUp packet(s) are not being received. From a tcpdump pcap file, identify if packets are actually being received at the PTP interface.

When the upstream master is using 2-step synchronization it will send a FollowUp packet for every Sync packet sent. Sync/FollowUp pairs have sequential sequence ID values. If some or all FollowUp packets are missing, the user should check these are being generated by the upstream master clock.

The counters in the /var/lib/sfptpd/stats-<clock id> files will identify the number of FollowUp messages received. There should be the same number of FollowUps as there are Sync messages received.

no-delay-resps

The slave will send periodic Delay_Request messages to the upstream master clock. The master clock should respond to each with a Delay_Response message having the same sequence ID value as the request.

From a tcpdump pcap file, identify if packets reported as missing are actually being received at the PTP interface.

If some or all Delay_Request packets are missing, the user should check the master clock configuration or consult the master clock vendor.

no-pdelay-resps

When using the peer-to-peer delay method this alarm identifies that PDelay_Response messages are not being received for all PDelay_Request messages sent by the slave server.

no-pdelay-resp-follow-ups

When using the peer-to-peer delay method this alarm identifies that PDelay_Response_FollowUp messages are not being received by the sfptpd slave.

no-tx-timestamps

Identifies that sfptpd is not able to timestamp outgoing packets.

no-rx-timestamps

Identifies that sfptpd is not able to recover adapter timestamps for received PTP packets.

no-interface

In a bond, there are no slave interfaces available.

clock-ctrl-failure

An instance or clock servo is unable to control a clock and attempts to control it result in errors.

pps-no-signal

sfptpd is unable to detect any incoming PPS signal. Check that the upstream clock is generating the PPS signal. Check that PPS cable is connected to the PPS INPUT on the Solarflare adapter.

pps-seq-num-error

sfptdp is receiving PPS pulses, but reports missing or unexpected PPS sequence numbers. Ensure the driver and firmware being by the adapter are up to date. If the issue persists, run the sfreport script and return the HTML output along with the sfptpd stats log output to support@solarflare.com

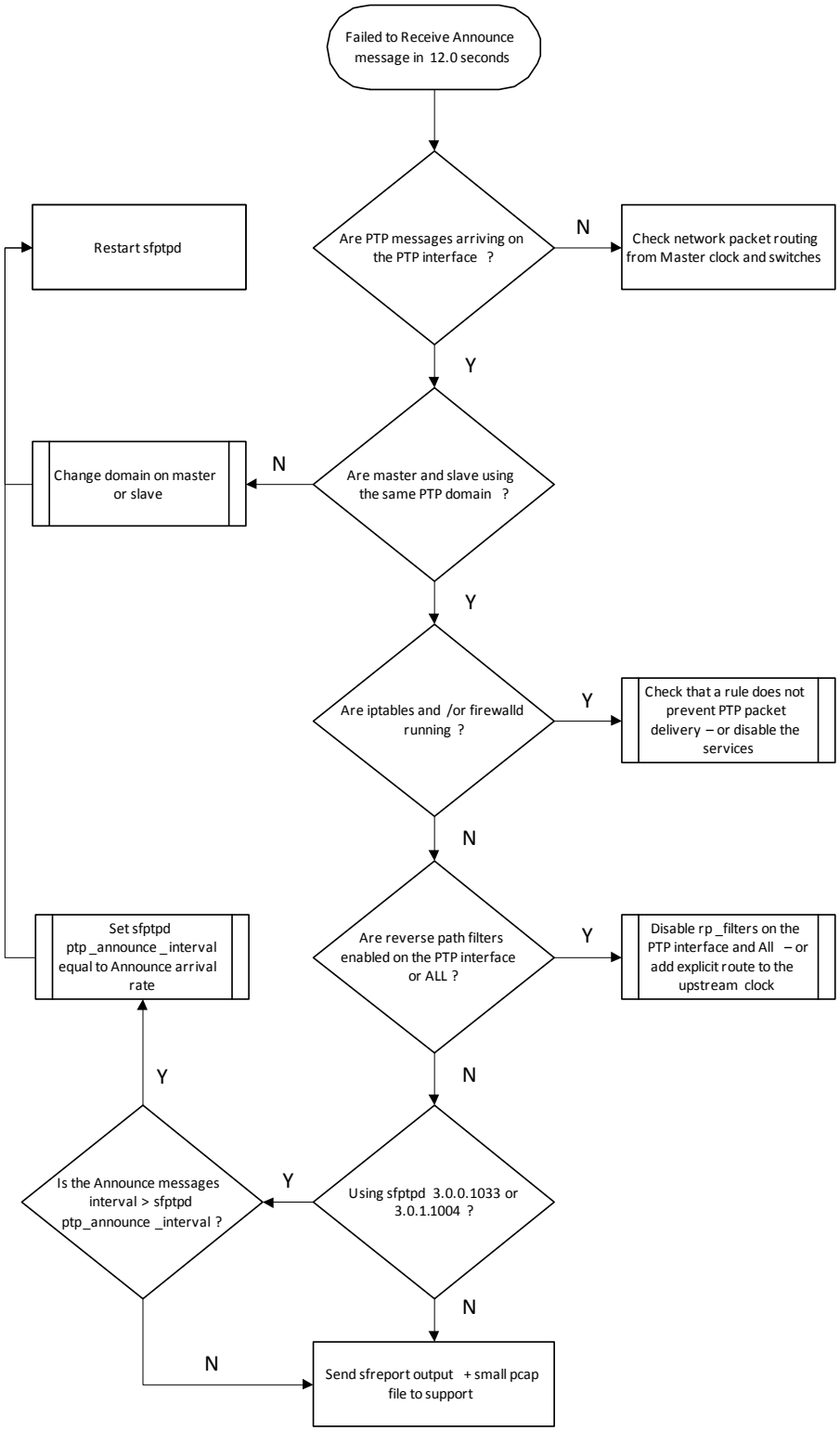
pps-bad-signal

sfptpd has detected an invalid PPS period i.e. the measured PPS period is too long to be a pulse because it exceeds 1 second.

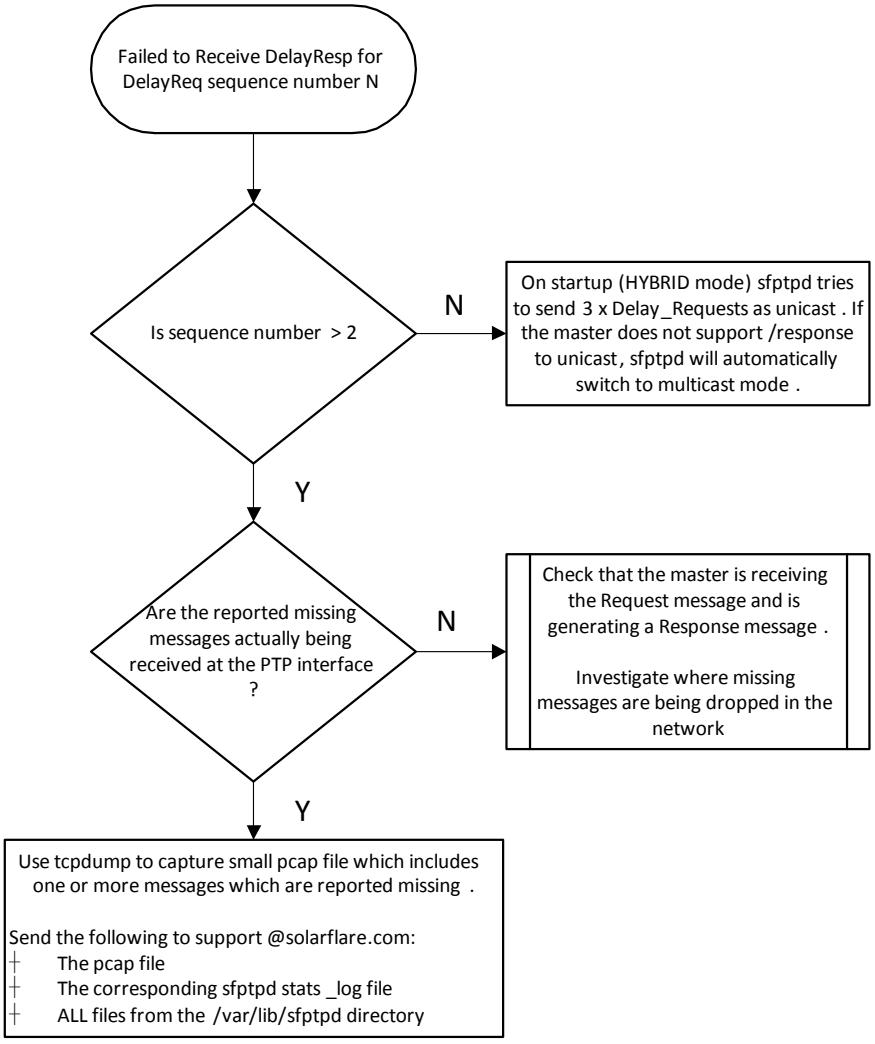
no-time-of-day

sfptpd is not detecting a time-of-day signal. This is normally seen when running sfptpd in one of the supported NTP/PPS modes. Time-of-day is the time supplied by the NTP server.

C.2 Failed to Receive Announce Messages



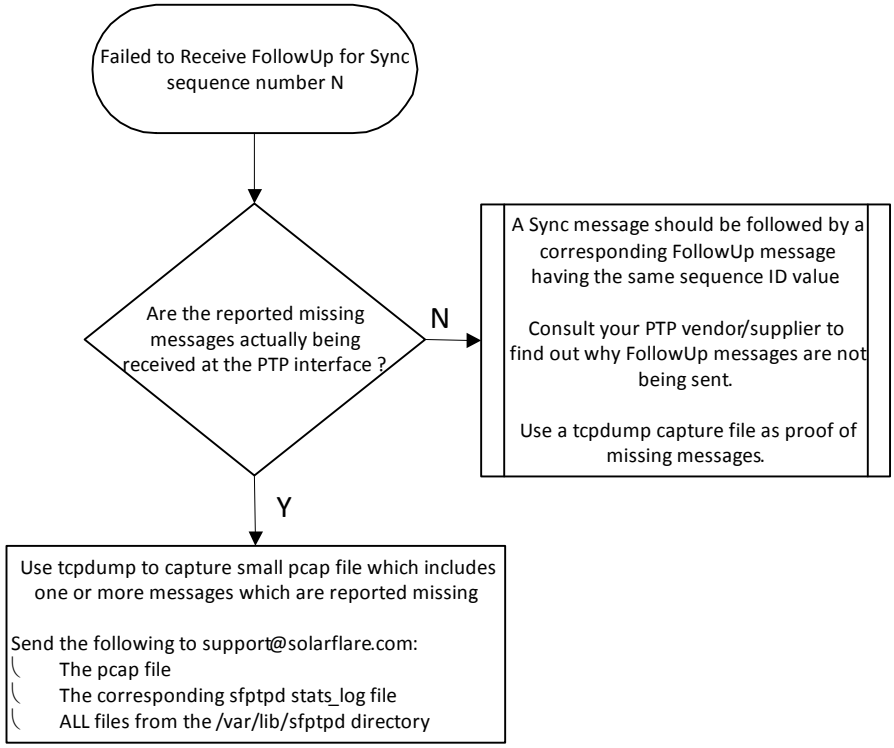
C.3 Missing Delay_Response Messages



The upstream master clock should respond to every Delay_Request message with a corresponding Delay_Response message having the same sequence ID value.

Missing one or two Delay_Response messages should not affect synchronization accuracy or precision, but missing many of these messages or missing consecutive sequence IDs may harm synchronization of the clocks.

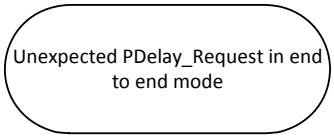
C.4 Missing Followup Messages



When the upstream master uses 2-step synchronization it should send periodic Sync messages. For every Sync there should be FollowUp message having the same sequence ID value.

Missing the occasional FollowUp message should not affect synchronization accuracy, but missing many of these messages or missing consecutive sequence IDs may harm the synchronization of the clocks.

C.5 Unexpected PDelay_Request Message



The peer-to-peer delay method is an alternative to the more widely used end-to-end delay method. Solarflare sfptpd will support both methods with the default being end-to-end. The delay method is configurable in the sfptpd configuration file.

When sfptpd is using the end-to-end delay method, it will generate the “Unexpected...” warning if a PDelay_Request message is received.

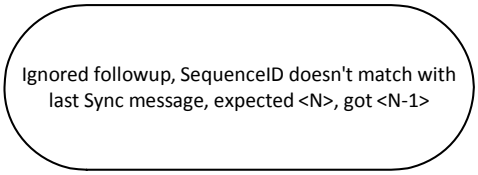
A similar “Unexpected...” message will be generated for any Delay_Request message received when sfptpd is using the peer-to-peer method.

When peer-to-peer working is used in the network, every PTP node must support it.

Mixing end-to-end with peer-to-peer methods is not supported in any PTP domain. All PTP nodes within a PTP domain must use the same method.

When peer-to-peer delay is used every node in the PTP network sends both the PDelay_Request and PDelay_Response messages to the neighbouring PTP node.

C.6 Ignored followup, SequenceID doesn't match with last Sync message



Using 2-step synchronization, an upstream master clock will send periodic Sync messages and corresponding FollowUp messages to the downstream slave. Sync and FollowUp message pairs have the same sequence ID value.

Sync/FollowUp message pairs should be received by the PTP slave node in sequence i.e.

Receive Sync (seq N),
  Receive FollowUp (seq N)
    Receive Sync (seq N+1)
      Receive FollowUp (seq N+1)

Under extreme congestion conditions, FollowUp messages may be delayed, lost or dropped in the Network.

The above message is generated by sfptpd if it has received a FollowUp (seq N) after it has already received a Sync with (seq N+1) and so it expecting the corresponding FollowUp (seq N+1).

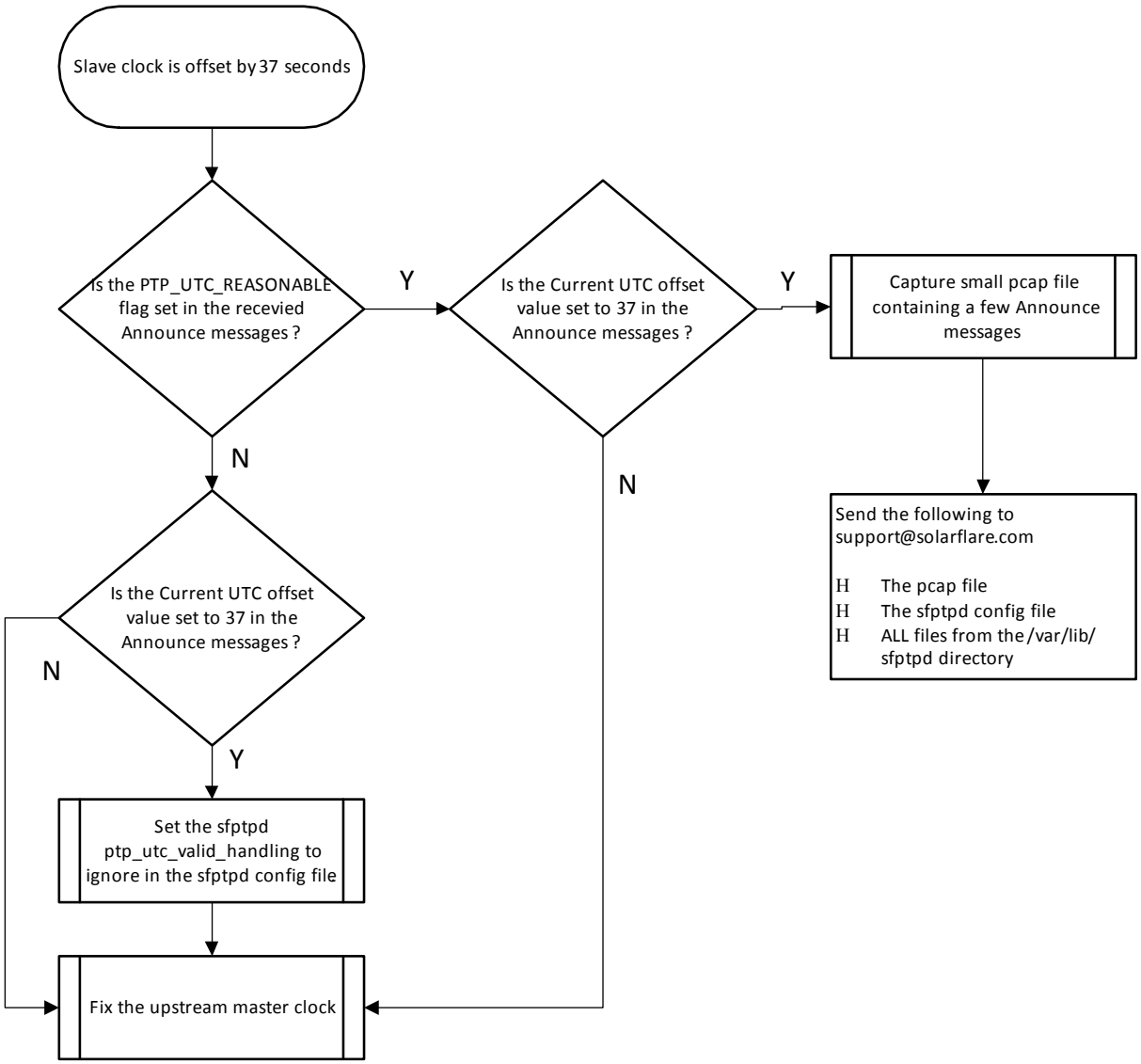
These rare events are indicative of extreme network congestion – which is delaying messages, the effects of which will probably be evident in other network applications.

Use tcpdump to capture small pcap file which includes one or more messages which are reported out of sequence.

Send the following to support@solarflare.com:

- The pcap file
- The corresponding sfptpd stats_log file
- ALL files from the /var/lib/sfptpd directory

C.7 Slave Clock offset by 37 seconds



PTP Master clocks use the atomic timescale (TAI). Servers in business networks use the UTC timescale (UTC).

The difference between the two timescales is the UTC offset - currently 37 seconds. This increments by one second whenever a leap second occurs.

UTC time = (TAI time - UTC offset)

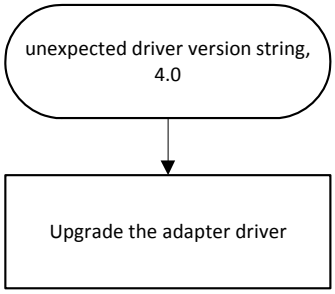
NOTE: When sfptpd is used as a master clock it uses UTC time - ensure the ptp_utc_offset option in the sfptpd master config file is set to 0:

If the sfptpd slave clocks are observed to be ~37 seconds offset from the master clock, the user should collect a small tcpdump pcap file and examine the UTC offset values in the received Announce messages.

A pcap file allows the user to examine the fields of the received PTP Announce messages:

```
flags: 0x023c
        0... .... .... .... = PTP_SECURITY: False
        .0.. .... .... .... = PTP profile Specific 2: False
        ..0. .... .... .... = PTP profile Specific 1: False
        .... .0.. .... .... = PTP_UNICAST: False
        .... ..1. .... .... = PTP_TWO_STEP: True
        .... ...0 .... .... = PTP_ALTERNATE_MASTER: False
        .... .... ..1. .... = FREQUENCY_TRACEABLE: True
        .... .... ...1 .... = TIME_TRACEABLE: True
        .... .... .... 1... = PTP_TIMESCALE: True
        .... .... .... .1.. = PTP_UTC_REASONABLE: True
        .... .... .... ..0. = PTP_LI_59: False
        .... .... .... ...0 = PTP_LI_61: False
      ...
    originCurrentUTCOffset: 37
```

C.8 Unexpected Driver Version String 4.0



This error message may be observed in the sfptpd stats_log/message_log during sfptpd startup.

The Linux ‘in-tree’ driver is the Solarflare adapter driver distributed with the Linux OS. This is normally the 4.0 or 4.1 version driver.

This driver does not support the PTP features required by sfptpd.

The driver should be upgraded to a later version from the Solarflare download portal at support@solarflare.com.

The driver being used by the Solarflare adapter can be identified using the following command:

```
# ethtool -i <interface>
driver: sfc
version: 4.0
firmware-version: 4.7.1.1001 rx1 tx1
```