Version 1.0 – August 2019

Abstract

After running the first simple tests using the syn1588® PTP Stack as described in the syn1588® PCIe NIC Quick Start Guide one starts setting up a real-life PTP system in a more complex scenario. This application note describes how to analyze the PTP link status using very simple means to identify frequently configuration issues like IP address mis-assignments, network issues or firewall blocking the traffic.

Scenario

You completed the first simple tests using the syn1588® PTP Stack most likely together with the syn1588® PCIe NIC following the Application Note “syn1588® PCIe NIC Quick Start Guide”. Everything runs smoothly so far. But now you connect to your real-life network using your PTP Grandmaster.

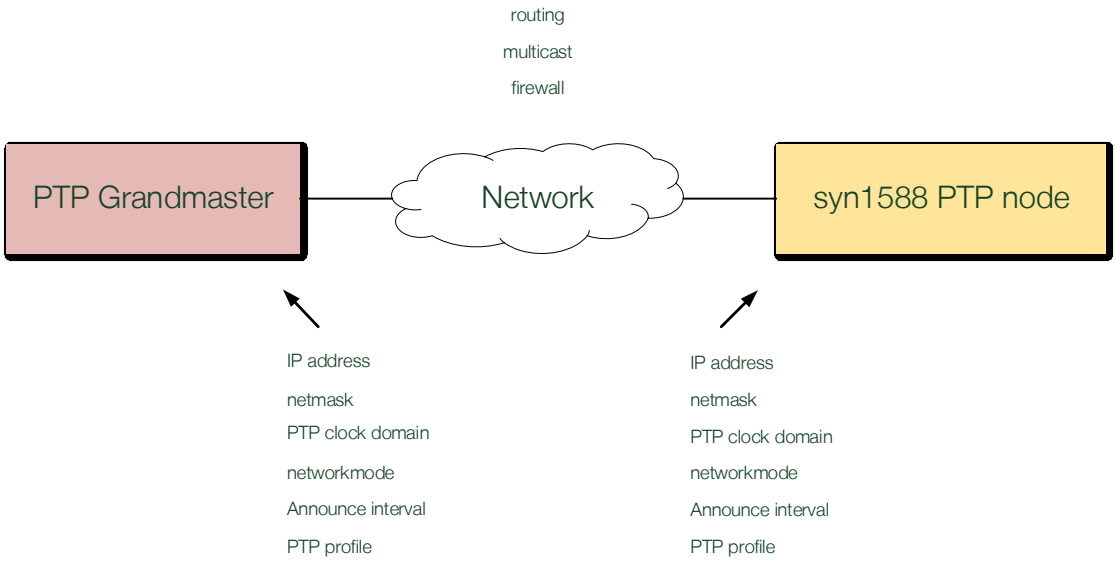


figure 1 Application scenario

Frequently you do not have direct access to the PTP Grandmaster in the network as well as the network infrastructure (switches etc.) since they are managed by your IT department. This application note provides you simple guidelines on how to identify connection or configuration issues using very simple means.

Your Tools

There are two very simple tools one can use to analyze the proper PTP communication on your node.

syn1588® PTP Stack Log Output

This "tool" comes free of charge with the syn1588® PTP Stack. The syn1588® PTP Stack provides a detailed log output with selectable verbosity level "-v <verbosity level>". While for normal operation a log level of 1 or 2 is sufficient one is interested in all details (log level 4) during debugging or setup procedure.

You can easily figure out network or PTP configuration issues by analyzing the output. The following listing shows such a typical syn1588® PTP Stack log output.

```
# ./ptp -ieth2 -CS -v3
syn1588(R) PTP Stack - IEEE1588-2008 Engine
Build date: Aug 20 2019 - V 1.9-13 Rev g7b9bf159
Copyright (c) Oregano Systems - Design & Consulting Gesmbh 2008-2019
Confidential unpublished data - All rights reserved

syn1588(R) PTP Stack started: 2019-08-20 07:30:35.756200 (UTC)
Port 1: adding config "i" = "eth2"
Port 1: adding config "C" = "S"
Port 1: adding config "v" = "3"
Command line: ./ptp -ieth2 -CS -v3
(1) Found Configuration for 1 ports
(1) Syn1588Ifc requires at least:
- linux driver version 1.4-15-g05b7283
- windows driver version 10/05/2017, 10.9.16.182
(1) Syn1588Impl: Device /dev/syncD0 found
(1) syn1588(R) Hardware Clock M 2.3.4 f=125000000 Hz
(1) Found stop clock support
(1) Using MAC TS Version 3148
(1) Using programmable 1-step TS
(1) syn1588(R) PCIe NIC Revision 2, Build 834
(1) Using syn1588 mode
(1) Spike M2S: Init with ival 0, buffer size 16
(1) Spike Path: Init with ival 0, buffer size 16
(1) Init shared mem
(1) read to NULL pointer ignored!
(1) syn1588HwClk: clearing leap second jump
(1) Settings: ClockId 00:1E:C0:FF:FE:85:DE:0B
(1) Settings: Prio1 128 ClkClass 255 clkAccuracy 39 clkVariance 65535
(1) Settings: Prio2 128 Domain 0
(1) SIOCSHWTSTAMP: tx_type 1 requested, got 1; rx_filter 0 requested, got 12
(1) Activated SO_TIMESTAMPING hardware
(1) SIOCSHWTSTAMP: tx_type 1 requested, got 1; rx_filter 0 requested, got 12
(1) Activated SO_TIMESTAMPING hardware
(1) Clk: Using Oregano Systems; syn1588(R) PCIe NIC Revision 2;
00:1E:C0:85:DE:0B
(1)   with ClockId 00:1E:C0:FF:FE:85:DE:0B
(1) Clk: Resetting servos
(1) Clk: Resetting filters
```

software build information

syn1588® hardware found

syn1588® hardware build information

figure 2 syn1588® PTP Stack: example log output verbosity level 3

Wireshark

Wireshark is a network traffic and protocol recorder and analyzer. You can download this free software here: https://www.wireshark.org/

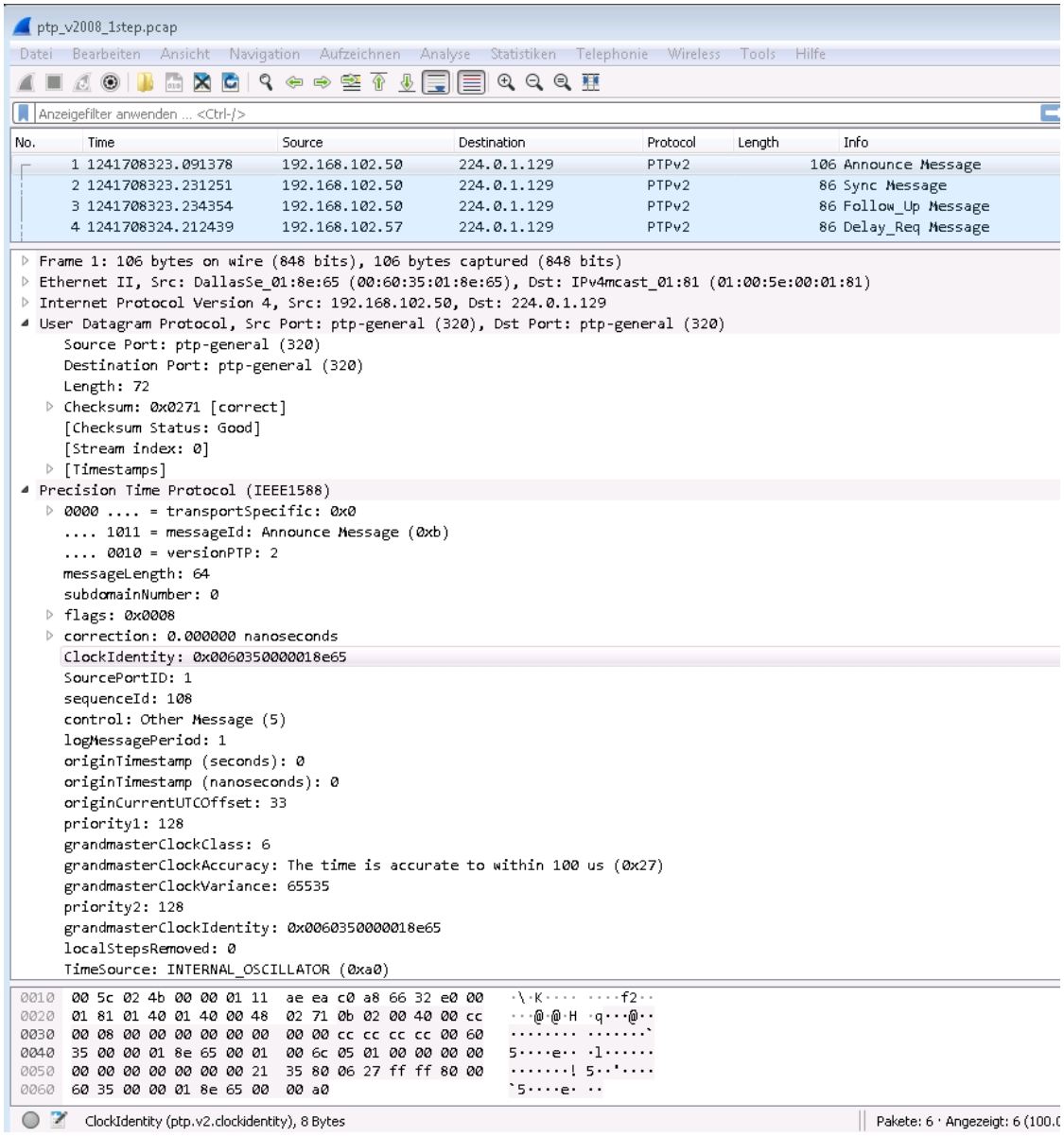


figure 3 Wireshark trace window

Analysis Procedure

The PTP communication (assuming one is using Layer 3 communication, thus UDP via either IPv4 or IPv6) requires a proper setup of the IP communication your system. This is nothing specific to PTP. You have to ensure a proper IP routing from your PTP node, using the selected network interface (typically a syn1588® PCIe NIC) to the Grandmaster. This basic IP setup shall be done using the standard operating system commands or procedures.

Check basic IP communication

One can use the PING command from a shell (DOS command window on Windows machines or any terminal on Linux machines) for this purpose. Simply ping the IP address of your PTP Grandmaster on the network.

```
[user@hugo a]$ ping 192.168.1.20
PING 192.168.1.20 (192.168.1.20) 56(84) bytes of data.
64 bytes from 192.168.1.20: icmp_seq=1 ttl=128 time=0.766 ms
64 bytes from 192.168.1.20: icmp_seq=2 ttl=128 time=0.490 ms
64 bytes from 192.168.1.20: icmp_seq=3 ttl=128 time=0.557 ms
64 bytes from 192.168.1.20: icmp_seq=4 ttl=128 time=0.509 ms
^C
--- 192.168.1.20 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3375ms
rtt min/avg/max/mdev = 0.490/0.580/0.766/0.112 ms
```

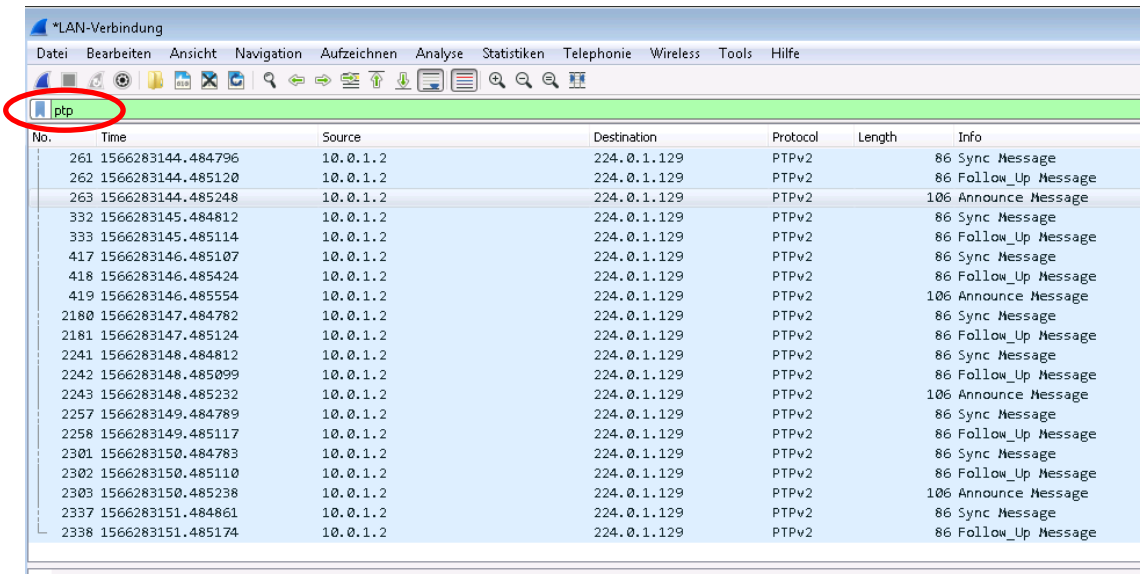
figure 4 Using PING to test basic IP communication

Caution

If your node owns several network interfaces you have to ensure proper IP routing. The PING test just gives you the information that there is a proper route from your node to the Grandmaster. It does not give you the information whether the network interface you plan to use for the PTP communication offers the required IP settings for proper IP communication.

Check PTP traffic on network interface

Now one can use Wireshark to check the incoming network traffic on the network interface one plans to use for PTP communication. One can simply filter the traffic for PTP messages only by entering the string “ptp” into the filtering window. Now, Wireshark displays all PTP messages seen on the selected network interface.



No.	Time	Source	Destination	Protocol	Length	Info
261	1566283144.484796	10.0.1.2	224.0.1.129	PTPv2	86	Sync Message
262	1566283144.485120	10.0.1.2	224.0.1.129	PTPv2	86	Follow_Up Message
263	1566283144.485248	10.0.1.2	224.0.1.129	PTPv2	106	Announce Message
332	1566283145.484812	10.0.1.2	224.0.1.129	PTPv2	86	Sync Message
333	1566283145.485114	10.0.1.2	224.0.1.129	PTPv2	86	Follow_Up Message
417	1566283146.485107	10.0.1.2	224.0.1.129	PTPv2	86	Sync Message
418	1566283146.485424	10.0.1.2	224.0.1.129	PTPv2	86	Follow_Up Message
419	1566283146.485554	10.0.1.2	224.0.1.129	PTPv2	106	Announce Message
2180	1566283147.484782	10.0.1.2	224.0.1.129	PTPv2	86	Sync Message
2181	1566283147.485124	10.0.1.2	224.0.1.129	PTPv2	86	Follow_Up Message
2241	1566283148.484812	10.0.1.2	224.0.1.129	PTPv2	86	Sync Message
2242	1566283148.485099	10.0.1.2	224.0.1.129	PTPv2	86	Follow_Up Message
2243	1566283148.485232	10.0.1.2	224.0.1.129	PTPv2	106	Announce Message
2257	1566283149.484789	10.0.1.2	224.0.1.129	PTPv2	86	Sync Message
2258	1566283149.485117	10.0.1.2	224.0.1.129	PTPv2	86	Follow_Up Message
2301	1566283150.484783	10.0.1.2	224.0.1.129	PTPv2	86	Sync Message
2302	1566283150.485110	10.0.1.2	224.0.1.129	PTPv2	86	Follow_Up Message
2303	1566283150.485238	10.0.1.2	224.0.1.129	PTPv2	106	Announce Message
2337	1566283151.484861	10.0.1.2	224.0.1.129	PTPv2	86	Sync Message
2338	1566283151.485174	10.0.1.2	224.0.1.129	PTPv2	86	Follow_Up Message

figure 5 Wireshark: check for PTP traffic

Caution

Wireshark is a network debugging tool. It does not care about proper IP settings nor on active firewalls or multicast routings on your node. You will see any message physically visible on the network interface although higher levels of your operating systems are instructed to block or re-route this traffic.

Check for syn1588® hardware

If one runs the syn1588® PTP Stack on a syn1588® PCIe NIC one can check whether the correct network interface had been chosen when invoking the syn1588® PTP Stack. The syn1588® PCIe NIC has to be reported in the log file as shown in the following example:

```
syn1588(R) PTP Stack - IEEE1588-2008 Engine
Build date: Aug 20 2019 - V 1.9-13 Rev g7b9bf159
Copyright (c) Oregano Systems - Design & Consulting GesmbH 2005-2018
Confidential unpublished data - All rights reserved

syn1588(R) PTP Stack started: 2019-08-20 07:30:35.756200 (UTC)
Port 1: adding config "i" = "eth2"
Port 1: adding config "C" = "S"
Port 1: adding config "v" = "3"
Command line: ./ptp -ieth2 -CS -v3
(1) Found Configuration for 1 ports
(1) Syn1588Ifc requires at least:
- linux driver version 1.4-15-g05b7283
- windows driver version 10/05/2017, 10.9.16.182
(1) Syn1588Impl: Device /dev/syncD0 found
(1) syn1588(R) Hardware Clock M 2.3.4 f=125000000 Hz
(1) Found stop clock support
(1) Using MAC TS Version 3148
(1) Using programmable 1-step TS
(1) syn1588(R) PCIe NIC Revision 2, Build 834
(1) Using syn1588 mode
(1) Spike M2S: Init with ival 0, buffer size 16
(1) Spike Path: Init with ival 0, buffer size 16
(1) Init shared mem
(1) read to NULL pointer ignored!
(1) syn1588HwClk: clearing leap second jump
(1) Settings: ClockId 00:1E:C0:FF:FE:85:DE:0B
(1) Settings: Prio1 128 ClkClass 255 clkAccuracy 39 clkVariance 65535
(1) Settings: Prio2 128 Domain 0
(1) SIOCSHWTSTAMP: tx_type 1 requested, got 1; rx_filter 0 requested, got 12
(1) Activated SO_TIMESTAMPING hardware
(1) SIOCSHWTSTAMP: tx_type 1 requested, got 1; rx_filter 0 requested, got 12
(1) Activated SO_TIMESTAMPING hardware
(1) Clk: Using Oregano Systems; syn1588(R) PCIe NIC Revision 2;
00:1E:C0:85:DE:0B
(1)   with ClockId 00:1E:C0:FF:FE:85:DE:0B
(1) Clk: Resetting servos
(1) Clk: Resetting filters
(1) Spike M2S: Init with ival 0, buffer size 16
(1) Spike Path: Init with ival 0, buffer size 16
(1) Init shared mem
(1) syn1588HwClk: clearing leap second jump
(1) 711.147750089 State Listening
```


figure 6 syn1588® PTP Stack: check for syn1588® hardware

If one runs the syn1588® PTP Stack on any network interface card no syn1588® hardware support will be found and reported as shown in the next example.

```
syn1588(R) PTP Stack - IEEE1588-2008 Engine
Build date: Aug 20 2019 - V 1.9-13 Rev g7b9bf159
Copyright (c) Oregano Systems - Design & Consulting GesmbH 2005-2018
Confidential unpublished data - All rights reserved

syn1588(R) PTP Stack started: 1970-01-01 00:14:36.296065 (UTC)
Port 1: adding config "i" = "eth2"
Port 1: adding config "C" = "S"
Port 1: adding config "v" = "3"
Command line: ./ptp -ieth2 -CS -v3
(1) Found Configuration for 1 ports
(1) Using software mode
(1) Spike M2S: Init with ival 0, buffer size 16
(1) Spike Path: Init with ival 0, buffer size 16
(1) Init shared mem
(1) read to NULL pointer ignored!
(1) Settings: ClockId 00:1E:C0:FF:FE:85:DE:0B
(1) Settings: Prio1 128 ClkClass 255 clkAccuracy 39 clkVariance 65535
(1) Settings: Prio2 128 Domain 0
(1) Running software mode without SO_TIMESTAMPING support
(1) Running software mode without SO_TIMESTAMPING support
(1) Clk: Using Linux System Clock,,
(1)   with ClockId 00:1E:C0:FF:FE:85:DE:0B
(1) Clk: Resetting servos
(1) Clk: Resetting filters
(1) Spike M2S: Init with ival 0, buffer size 16
(1) Spike Path: Init with ival 0, buffer size 16
(1) Init shared mem
(1) 913.303372000 State Listening
```

ClockID =
MAC address

no syn1588® hardware support

figure 7 syn1588® PTP Stack: without syn1588® hardware (software mode)

Check for ANNOUNCE messages

After successfully starting the syn1588® PTP Stack on the intended network interface one shall receive ANNOUNCE messages from the PTP Grandmaster. For this run the syn1588® PTP Stack at least with log level 3. All incoming, received PTP messages are flagged by an text arrow to the right "--->". One can simply grep for these lines in the text output.

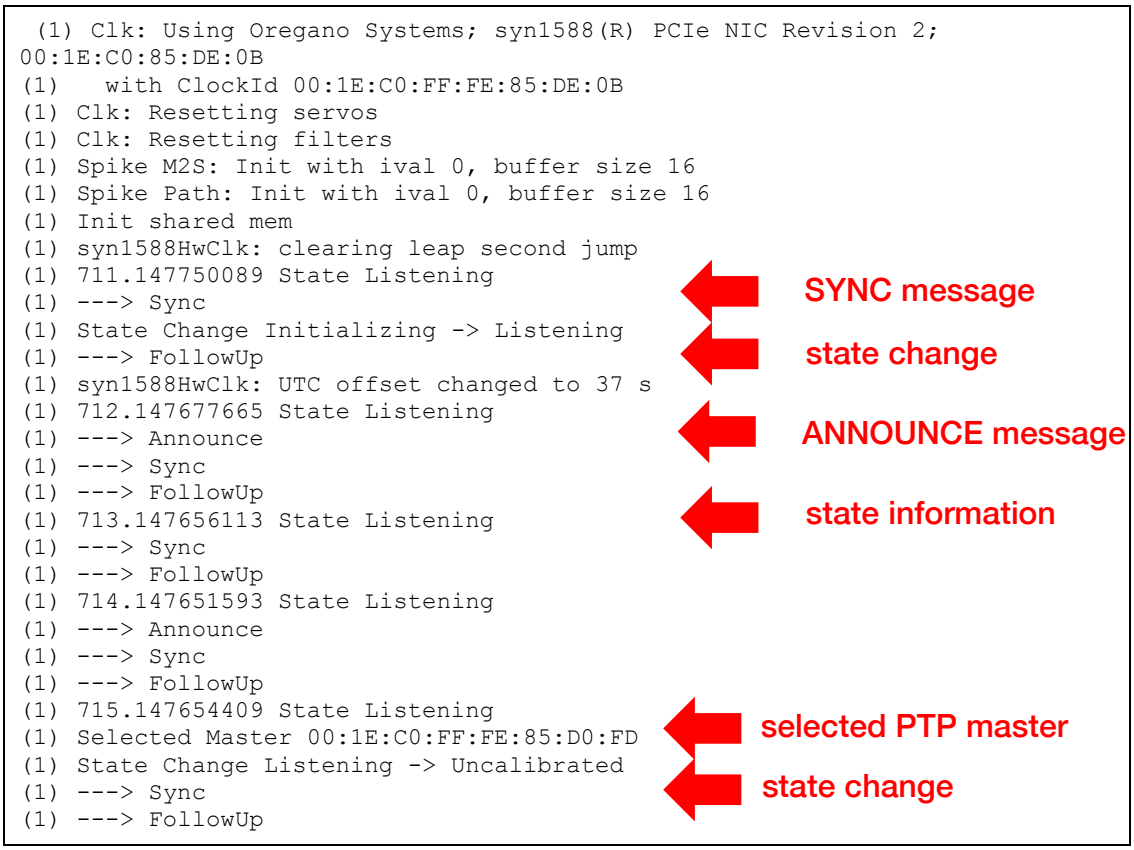


figure 8 syn1588® PTP Stack: check for incoming ANNOUNCE messages

In this example a SYNC messages is received first. The local PTP node now moves from the state "Initializing" to the state "Listening". Without receiving SYNC or ANNOUNCE messages the node will remain in the state "Initializing". If one does not see ANNOUNCE messages here your PTP, IP or node setup is corrupted. Either an improper IP configuration may inhibit the syn1588® PTP Stack to receive the Grandmaster's PTP messages or a local firewall blocks the PTP traffic. PTP utilize the UDP ports 319 and 320. Correct your system settings until you see the expected PTP messages here. Additionally, the Grandmaster may use another PTP clock domain than it is configured for the PTP slave. One can use Wireshark and check the PTP clock domain used by the Grandmaster.

Note, that the announce interval of the PTP slaves must match the announce interval used by the Grandmaster. The PTP slaves derive an announce timeout period based on the configured announce interval. If the Grandmaster sends ANNOUNCE messages not within the announce timeout period the PTP slave will also never change its state to “Slave”.

After receiving twice the correct ANNOUNCE message the PTP slave moves from the state “Listening” to the state “Uncalibrated”; the PTP master had been accepted now.

One can use Wireshark on the same network interface to check whether the expected ANNOUNCE messages are physically available on the network interface.

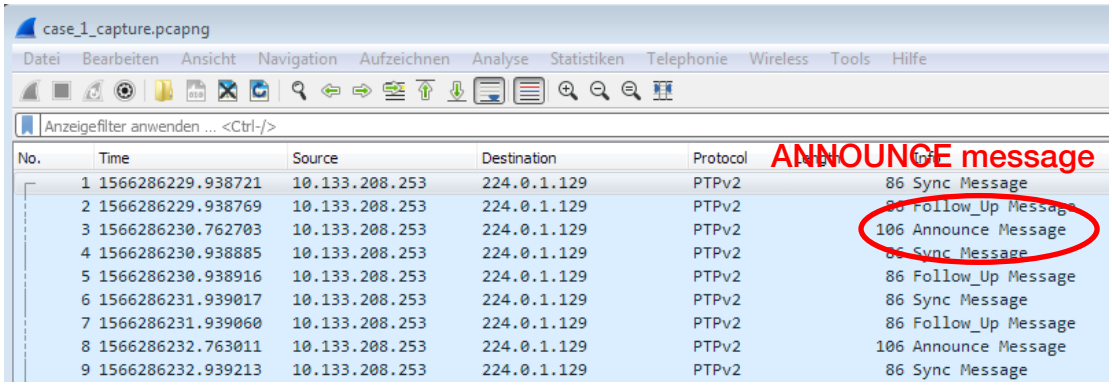


figure 9 Wireshark: check for incoming ANNOUNCE messages

If one would like to check the PTP clock domain used by the Grandmaster, select e.g. an Announce message sent by the Grandmaster and check the PTP message details as shown in the next figure.

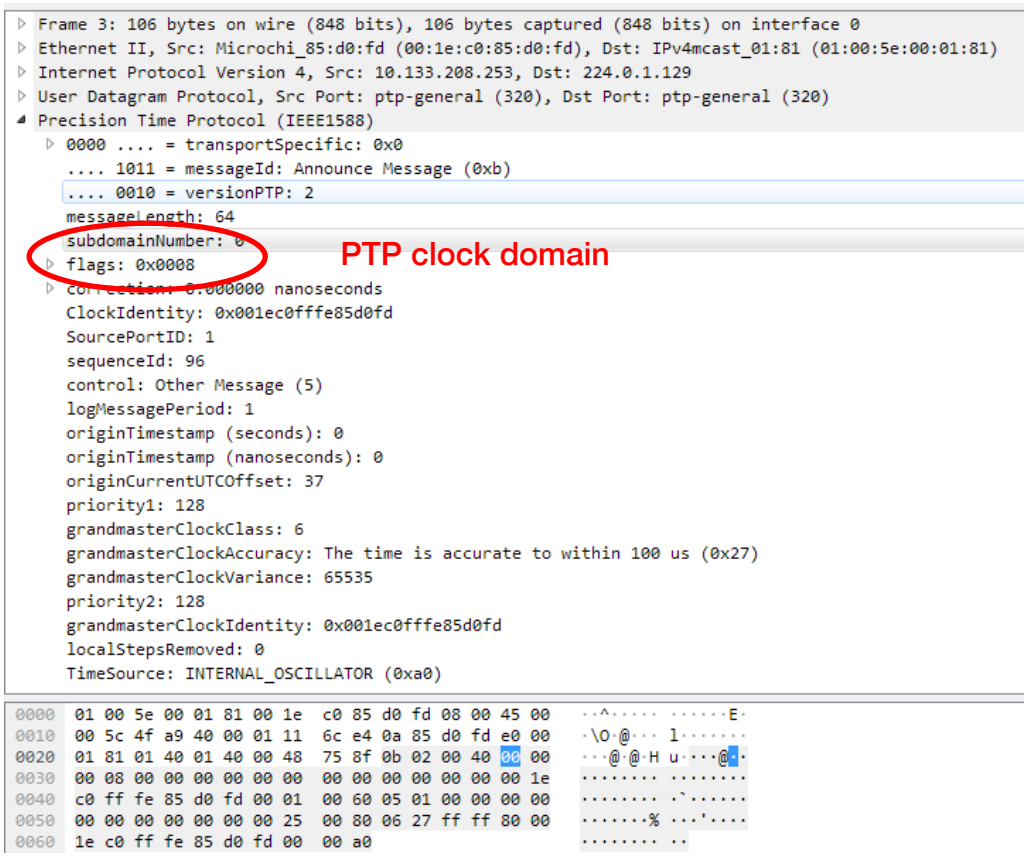


figure 10 Wireshark: check for the PTP clock domain

The following syn1588® PTP Stack log output shows an error condition. This happens if a local firewall blocks the PTP traffic on the PTP slave. No SYNC and ANNOUNCE messages can be received by the syn1588® PTP Stack.

```
syn1588(R) PTP Stack - IEEE1588-2008 Engine
Build date: Aug 20 2019 - V 1.9-13 Rev g7b9bf159
Copyright (c) Oregano Systems - Design & Consulting GesmbH 2005-2018
Confidential unpublished data - All rights reserved

syn1588(R) PTP Stack started: 1970-01-01 01:24:56.070509 (UTC)
Port 1: adding config "i" = "eth2"
Port 1: adding config "C" = "S"
Port 1: adding config "v" = "3"
Command line: ./ptp -ieth2 -CS -v3
(1) Found Configuration for 1 ports
(1) Syn1588Ifc requires at least:
- linux driver version 1.4-15-g05b7283
- windows driver version 10/05/2017, 10.9.16.182
(1) Syn1588Impl: Device /dev/syncD0 found
(1) syn1588(R) Hardware Clock M 2.3.4 f=125000000 Hz
(1) Found stop clock support
(1) Using MAC TS Version 3148                    no incoming messages
(1) Using programmable 1-step TS                 and
(1) syn1588(R) PCIe NIC Revision 2, Build 834    no outgoing messages
(1) Using syn1588 mode
(1) Spike M2S: Init with ival 0, buffer size 16
(1) Spike Path: Init with ival 0, buffer size 16
(1) Init shared mem
(1) read to NULL pointer ignored!
(1) syn1588HwClk: clearing leap second jump
(1) Settings: ClockId 00:1E:C0:FF:FE:85:DE:0B
(1) Settings: Prio1 128 ClkClass 255 clkAccuracy 39 clkVariance 65535
(1) Settings: Prio2 128 Domain 0
(1) SIOCSHWTSTAMP: tx_type 1 requested, got 1; rx_filter 0 requested, got 12
(1) Activated SO_TIMESTAMPING hardware
(1) SIOCSHWTSTAMP: tx_type 1 requested, got 1; rx_filter 0 requested, got 12
(1) Activated SO_TIMESTAMPING hardware
(1) Clk: Using Oregano Systems; syn1588(R) PCIe NIC Revision 2;
00:1E:C0:85:DE:0B
(1)   with ClockId 00:1E:C0:FF:FE:85:DE:0B
(1) Clk: Resetting servos
(1) Clk: Resetting filters
(1) Spike M2S: Init with ival 0, buffer size 16
(1) Spike Path: Init with ival 0, buffer size 16
(1) Init shared mem
(1) syn1588HwClk: clearing leap second jump
(1) 5133.069558229 State Listening
(1) State Change Initializing -> Listening
(1) syn1588HwClk: UTC offset changed to 37 s
(1) 5134.069538148 State Listening
(1) 5135.069543179 State Listening                state Listening
(1) 5136.069617835 State Listening
(1) 5137.069747346 State Listening
(1) 5138.069806770 State Listening
```

figure 11 syn1588® PTP Stack: active firewall blocking PTP traffic

Note, there are no incoming messages flagged by "--->" seen in the log file. The PTP slave remains in the state "Listening". Please further note, that the syn1588® PTP Stack does not send any packet in this state since it instructed

to act as PTP slave; there are no outgoing messages as well “<---”. In Wireshark one can still see these PTP messages on the network interface. Wireshark does not care about firewalls.

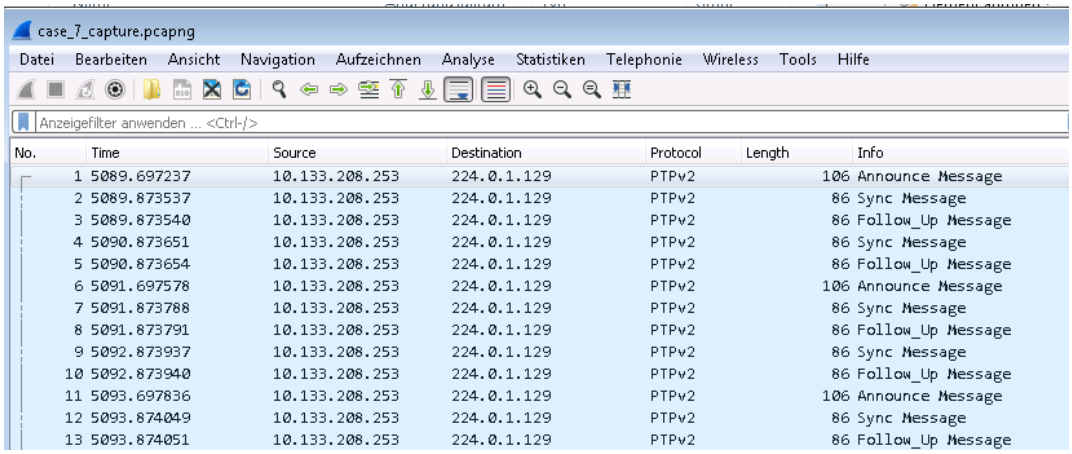


figure 12 Wireshark: PTP messages with active firewall blocking PTP traffic

The following syn1588® PTP Stack log output shows what happens if the PTP slave receives messages from

If the Grandmaster sends PTP messages with a wrong PTP clock domain these messages have to be ignored by the syn1588® PTP Stack. Thus the PTP slave remains in the state “Listening”. If one increases the syn1588® PTP Stack’ log level using the maximum verbosity level of 4 one will see a hint for such a wrong PTP clock domain value.

```
(1) Sem 1: Using tout 0.0 for lock operation
(1) Sem 0: Using tout 0.0 for unlock operation
(1) Event message of size 44
(1) Message with wrong domain received (0)
(1) Invalid message received
(1) General message of size 44
(1) Message with wrong domain received (0)
(1) Invalid message received
(1) Timeout: state Listening
(1) State Change Initializing -> Listening
```

wrong PTP clock domain

figure 13 syn1588® PTP Stack: wrong clock domain with log level 4

Check for SYNC messages

One can use the identical procedure as described in the previous chapter to check for SYNC messages received. The following excerpt of the syn1588® PTP Stack log file shows the received SYNC messages.

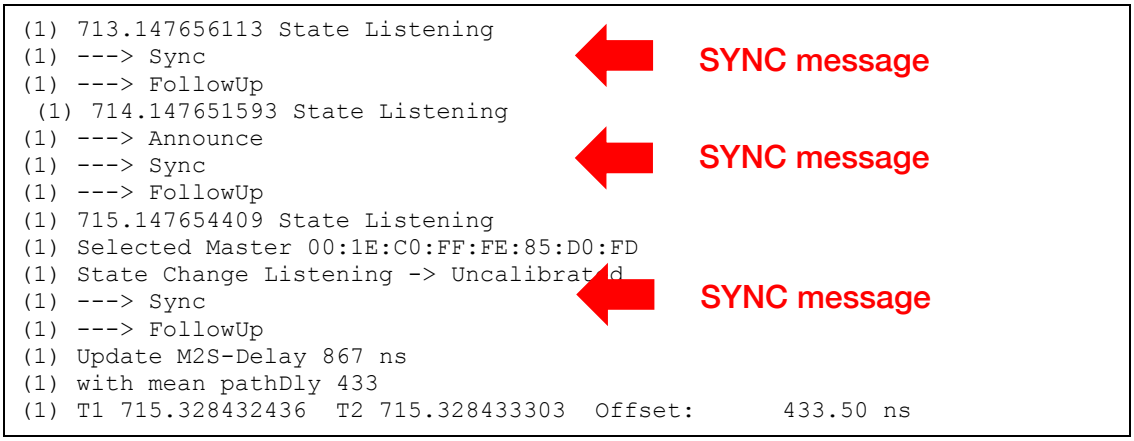


figure 14 syn1588® PTP Stack: check for incoming SYNC messages

PTP defines two basic communication mechanisms.

- 2-step mode
- 1-step mode

In the example shown above 2-step mode is used. Every SYNC is accompanied by a FOLLOW-UP message that sends the timestamp. For 1-step operation the sender inserts the timestamp on-the-fly directly into the SYNC packet.

Check for DELAY REQUEST & DELAY RESPONSE messages

As soon as the PTP slave proceeds in the “Uncalibrated” state it starts sending DELAY REQUEST messages that shall be answered by DELAY RESPONSE message by the Grandmaster. Note, that the DELAY REQUEST messages are sent (outgoing message) by the PTP slave and thus are preceded by a arrow to the left “<---“. One can again grep for this pattern.

```
(1) 714.147651593 State Listening
(1) ---> Announce
(1) ---> Sync
(1) ---> FollowUp
(1) 715.147654409 State Listening
(1) Selected Master 00:1E:C0:FF:FE:85:D0:FD
(1) State Change Listening -> Uncalibrated
(1) ---> Sync
(1) ---> FollowUp
(1) Update M2S-Delay 867 ns
(1) with mean pathDly 433
(1) T1 715.328432436  T2 715.328433303  Offset:      433.50 ns
(1) <--- DlyReq
(1) ---> DlyResp
(1) T3 715.397722898  T4 715.397722803
(1) S2M-Delay -95 ns
(1) Starting slew lock
(1) <--- DlyReq
(1) ---> DlyResp
(1) T3 716.087025233  T4 716.087025131
(1) S2M-Delay -102 ns
(1) 716.147678513 State Uncalibrated
(1) ---> Announce
(1) ---> Sync
(1) ---> FollowUp
(1) Update M2S-Delay 868 ns
(1) with mean pathDly 383
(1) T1 716.328572900  T2 716.328573768  Offset:      485.00 ns
(1) <--- DlyReq
(1) ---> DlyResp
(1) T3 716.888139260  T4 716.888139155
(1) S2M-Delay -105 ns
```

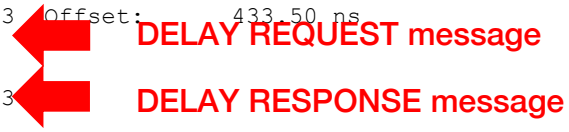


figure 15 syn1588® PTP Stack: check for outgoing DELAY REQUEST and incoming DELAY RESONSE messages

If one does not receive DELAY RESPONSE messages the DELAY REQUEST messages might have been filtered on their way to the Grandmaster.

Transition to Slave State

After successfully receiving several DELAY RESPONSE messages the PTP slave proceeds from the state “Uncalibrated” to the state “Slave”.

```
(1) ---> Sync
(1) ---> FollowUp
(1) Update M2S-Delay -3462 ns
(1) with mean pathDly 382
(1) T1 2751.684201916  T2 2751.684198454  Offset:    -3844.00 ns
(1) 2752.223902573 State Uncalibrated
(1) ---> Announce
(1) <--- DlyReq
(1) ---> DlyResp
(1) T3 2752.533240365  T4 2752.533244579
(1) S2M-Delay 4214 ns
(1) ---> Sync
(1) ---> FollowUp
(1) Update M2S-Delay -3454 ns
(1) with mean pathDly 380
(1) T1 2752.684329812  T2 2752.684326358  Offset:    -3834.00 ns
(1) DriftCalc: drift is 4 ns/s
(1) Adjusting Rate by 1100.19362 ns/s
(1) Clk: Resetting filters
(1) Spike M2S: Init with ival 0, buffer size 16
(1) Spike Path: Init with ival 0, buffer size 16
(1) Drift calc completed
(1) <--- DlyReq
(1) State Change Uncalibrated -> Slave
(1) ---> DlyResp
(1) T3 2753.095305026  T4 2753.095309227
(1) S2M-Delay 4201 ns
(1) 2753.223962588 State Slave
(1) ---> Sync
(1) Missing Sync SeqID 1; actual 1683
(1) ---> FollowUp
(1) Update M2S-Delay -3457 ns
(1) with mean pathDly 372
(1) T1 2753.684459828  T2 2753.684456371  Offset:    -3829.00 ns
(1) <--- DlyReq
(1) ---> DlyResp
(1) T3 2753.734605489  T4 2753.734609707
(1) S2M-Delay 4218 ns
(1) 2754.224041520 State Slave
```

state change

Slave-to-Master delay

Master-to-Slave delay

state Slave

figure 16 syn1588® PTP Stack: transition to state “Slave”

Now the PTP slave is fully synchronized with the master. The syn1588® PTP Stack log output also reports the measured delays Master-to-Slave (M2S) and slave-to-Master (S2M) as well as the timestamps T1 – T4.

Offset to Master

One shall observe that over the time the offset of the slave to the master is reduced to a minimum. The speed of this adaptation depends on the selected clock servo algorithm and parameters. The syn1588® PTP Stack log output displays two types of offset values:

- unfiltered offset
- filtered offset

Only the latter value is a useful measure for the current state of the synchronization.

```
(1) ---> Sync
(1) ---> FollowUp                                    unfiltered offset
(1) Update M2S-Delay 385 ns
(1) with mean pathDly 378
(1) T1 749.332054100  T2 749.332054485  Offset:        6.36 ns
(1) <--- DlyReq
(1) ---> DlyResp
(1) T3 750.043813054  T4 750.043813427
(1) S2M-Delay 373 ns
(1) 750.149951168 State Slave
(1) Adjusting clock at 6.28 ns offset    filtered offset
(1) Spike M2S: mean: 385, median: 384, variance: 140
(1) Spike Path: mean: 375, median: 374, variance: 75
(1) Adjusting Rate by 1106.55417 ns/s
(1) ---> Announce
(1) ---> Sync
(1) ---> FollowUp
(1) Update M2S-Delay 386 ns
(1) with mean pathDly 379
(1) T1 750.332192180  T2 750.332192566  Offset:        6.93 ns
(1) <--- DlyReq
(1) ---> DlyResp
(1) T3 750.884249238  T4 750.884249619
(1) S2M-Delay 381 ns
(1) 751.150041129 State Slave
(1) Adjusting clock at 4.39 ns offset
(1) Spike M2S: mean: 385, median: 384, variance: 66
(1) Spike Path: mean: 375, median: 374, variance: 287
(1) Adjusting Rate by 1106.43542 ns/s
```

figure 17 syn1588® PTP Stack: offset to the master

Multicast Operation

When using multicast network mode for PTP the whole (!) network infrastructure has to be set up to properly connect also the multicast messages from the Grandmaster to all PTP nodes.

Presence of a Transparent Clock

In case transparent clocks (TCs) are on the path from the Grandmaster to the PTP slave another field will be displayed in the syn1588® PTP Stack log output:

- the correction field

The correction field value summarizes all time the PTP packet resided onto such a transparent clock while packet traverses from the Grandmaster to the PTP slave and vice versa. The syn1588® PTP Stack uses this information in it's offset calculation. If no standard Ethernet switches are used in the network but just TCs, the resulting master-to-slave offset is almost the same as if the Grandmaster and PTP slave would have been connected by a direct network cable.

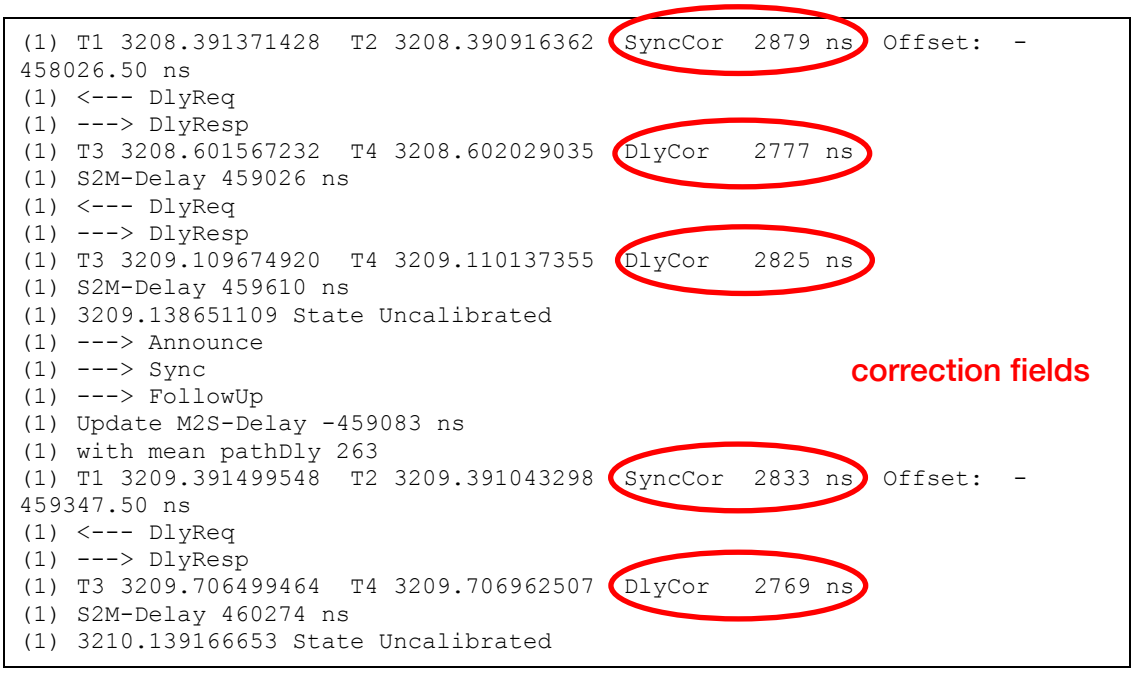


figure 18 syn1588® PTP Stack: presence of TC – correction fields

Version 1.0 – August 2019

Oregano Systems
A Meinberg Company

Franzosengraben 8
A-1030 Vienna
Austria
http://oregano.at
contact@oregano.at