Genio 700/1200 TSN Evaluation Guide

Version: 1.6
Release date: 2023-07-30

Use of this document and any information contained therein is subject to the terms and conditions set forth in Exhibit 1. This document is subject to change without notice.

Version History

Version	Date	Author	Description
1.0	2021-12-31		Initial release
1.1	2022-03-31		For IoT based on Kernel-5.10
1.2	2022-06-15		For IoT based on Kernel-5.15
1.3	2022-7-29		Add OPC UA and Remote Configuration Samples
1.4	2022-12-20		Add Genio 700 Platform Support
1.5	2023-02-01		Add Chapter-2 to Summary Test Result
1.6	2023-07-30		Update template

Table of Contents

List of Figures

1 Test Environment

The network controller eth0 in GENIO 700 and Genio 1200 have same TSN features, so we use Genio 1200 platform for demo in this document and Genio 1200 platform can be replaced with GENIO 700 platform if you want to verify TSN features on GENIO 700 platform.

1.1 Hardware Environment

The network controller eth0 in Genio 1200 can only serve as end station in TSN network. Connect Genio 1200 platform-1 with platform-2 with cable during our TSN test.
Use the following command to check the TSN ethernet device name:

```
# ls /sys/devices/platform/soc/11021000.ethernet/net
eth0
```

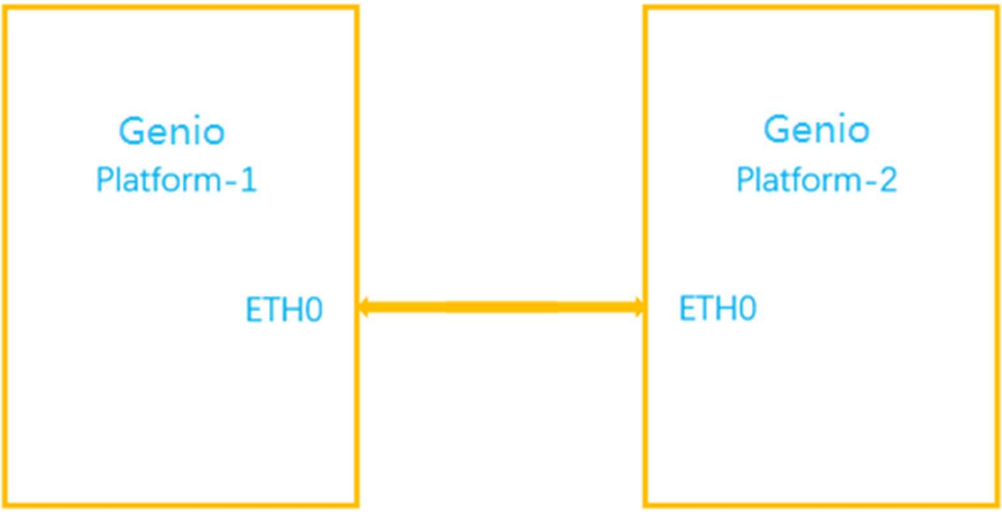


Figure 1-1. TSN Test Hardware Environment Setup

1.2 Software Environment

SW System:

Kirkstone/Kernel-5.15

The test scripts are collected in */etc/tsn-scripts* folder.

config-etf-offload.sh	*config-taprio-offload.sh*
run-udp-tai-tc-etf.sh	*txtime_offset_stats.py*
config-fpe.sh	*filter*
run-udp-tai-tc-taprio.sh	*base-time.sh*

Binaries used in TSN test are installed to */usr/bin*.

udp_tai	*adjust_clock_tai_offset*
check_clocks	*dump-classifier*
tsn_talker	*tsn_listener*

2 Test Result Summary

Test Item	Test Result
802.1AS	Pass
802.1Qav	Pass
ETF	Pass
802.1Qbv	Pass
802.1Qbu	Pass
OPC UA	Pass
NETCONF/YANG	Pass

3 Detailed Test and Analysis for TSN Schedulers

This chapter describes and evaluates the behaviors of different TSN schedulers, TSN schedulers are based on TX multi-queues of NIC Hardware as shown in Figure 2.

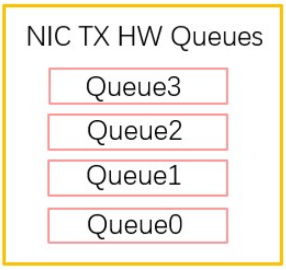


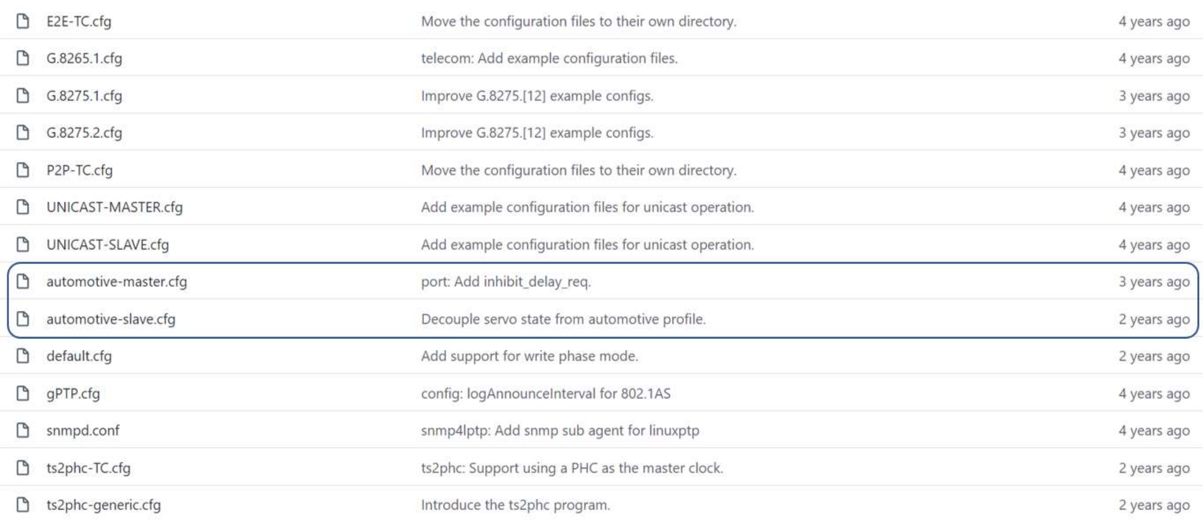
Figure 3-1. Multi-Queues in NIC TX Hardware

Queue0 is reserved for best effort traffic and Queue1-3 are reserved for TSN traffic. The TX queues have different behaviors with different TSN schedulers.

3.1 802.1AS Test

3.1.1 802.1AS Introduction

Time synchronization is one of the core functionalities of TSN, and it is specified by IEEE 802.1AS, also known as Generalized Precision Time Protocol (gPTP). gPTP is a profile from IEEE 1588, also known as Precision Time Protocol (PTP). gPTP consists of simplifications and constraints to PTP to optimize it to time-sensitive applications. In the Linux* ecosystem, Linux PTP[1] is the most popular implementation of PTP. It supports several profiles including gPTP and AVNU automotive profile as shown in Figure 3 and the 802.1AS test is based on Linux PTP.



E2E-TC.cfg	Move the configuration files to their own directory.	4 years ago
G.8265.1.cfg	telecom: Add example configuration files.	4 years ago
G.8275.1.cfg	Improve G.8275.[12] example configs.	3 years ago
G.8275.2.cfg	Improve G.8275.[12] example configs.	3 years ago
P2P-TC.cfg	Move the configuration files to their own directory.	4 years ago
UNICAST-MASTER.cfg	Add example configuration files for unicast operation.	4 years ago
UNICAST-SLAVE.cfg	Add example configuration files for unicast operation.	4 years ago
automotive-master.cfg	port: Add inhibit_delay_req.	3 years ago
automotive-slave.cfg	Decouple servo state from automotive profile.	2 years ago
default.cfg	Add support for write phase mode.	2 years ago
gPTP.cfg	config: logAnnounceInterval for 802.1AS	4 years ago
snmpd.conf	snmp4lptp: Add snmp sub agent for linuxptp	4 years ago
ts2phc-TC.cfg	ts2phc: Support using a PHC as the master clock.	2 years ago
ts2phc-generic.cfg	Introduce the ts2phc program.	2 years ago

Figure 3-2. Automotive Profile Used in TSN Test

Here we present the steps taken for setting up a test that slave end station(platform-1) synchronizes time with master end station(platform-2).

3.1.2 Test Steps

(1) Enter super user mode on platform-1&2

```
# root
```

(2) Kill NetworkManager process on platform-1&2 to avoid losing IP address and qdisc

```
# ps -aux | grep NetworkManager
# kill xxx
```

where xxx stands for PID of NetworkManager process

(3) Disable EEE on platform-1&2 to reduce path delay caused by PHY enter and exit EEE mode

```
# ethtool --set-eee eth0 eee off advertise 0
# ethtool --show-eee eth0
EEE Settings for eth0:
          EEE status: disabled
          Tx LPI: disabled
          Supported EEE link modes:  100baseT/Full
                                     1000baseT/Full
          Advertised EEE link modes:  Not reported
          Link partner advertised EEE link modes:  Not reported
```

(4) Set platform-2 MAC and IP address

```
# ifconfig eth0 down
# ifconfig eth0 hw ether 00:11:22:33:44:55 up
# ifconfig eth0 192.168.0.10
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500  metric 1
        inet 192.168.0.10  netmask 255.255.255.0  broadcast 192.168.0.255
        ether 00:11:22:33:44:55  txqueuelen 1000  (Ethernet)
        RX packets 35  bytes 8336 (8.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 77  bytes 13750 (13.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 39
```

(5) Set platform-1 IP address

```
# ifconfig eth0 192.168.0.1
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500  metric 1
        inet 192.168.0.1  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::110e:8ccc:3ff3:19ff  prefixlen 64  scopeid 0x20<link>
        ether 00:55:7b:b5:7d:f7  txqueuelen 1000  (Ethernet)
        RX packets 41  bytes 10188 (9.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 87  bytes 14992 (14.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 39
```

(6) Ping platform-1 from platform-2 to confirm the path link is ready

```
# ping 192.168.0.1 -c 3
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=0.596 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=64 time=0.231 ms
64 bytes from 192.168.0.1: icmp_seq=3 ttl=64 time=0.212 ms
```

(7) Check PTP clock and timestamping capability with Hardware timestamps for TX & RX

```
# ethtool -T eth0
Time stamping parameters for eth0:
Capabilities:
           hardware-transmit     (SOF_TIMESTAMPING_TX_HARDWARE)
           software-transmit     (SOF_TIMESTAMPING_TX_SOFTWARE)
           hardware-receive      (SOF_TIMESTAMPING_RX_HARDWARE)
           software-receive      (SOF_TIMESTAMPING_RX_SOFTWARE)
           software-system-clock (SOF_TIMESTAMPING_SOFTWARE)
           hardware-raw-clock    (SOF_TIMESTAMPING_RAW_HARDWARE)
PTP Hardware Clock: 0
Hardware Transmit Timestamp Modes:
           off                   (HWTSTAMP_TX_OFF)
           on                    (HWTSTAMP_TX_ON)
Hardware Receive Filter Modes:
           none                  (HWTSTAMP_FILTER_NONE)
           all                   (HWTSTAMP_FILTER_ALL)
           ptpv1-l4-event        (HWTSTAMP_FILTER_PTP_V1_L4_EVENT)
           ptpv1-l4-sync         (HWTSTAMP_FILTER_PTP_V1_L4_SYNC)
           ptpv1-l4-delay-req    (HWTSTAMP_FILTER_PTP_V1_L4_DELAY_REQ)
           ptpv2-l4-event        (HWTSTAMP_FILTER_PTP_V2_L4_EVENT)
           ptpv2-l4-sync         (HWTSTAMP_FILTER_PTP_V2_L4_SYNC)
           ptpv2-l4-delay-req    (HWTSTAMP_FILTER_PTP_V2_L4_DELAY_REQ)
           ptpv2-event           (HWTSTAMP_FILTER_PTP_V2_EVENT)
           ptpv2-sync            (HWTSTAMP_FILTER_PTP_V2_SYNC)
           ptpv2-delay-req       (HWTSTAMP_FILTER_PTP_V2_DELAY_REQ)
```

(8) Execute PTP slave on platform-1 with automotive-slave.cfg configuration file

```
# ptp4l -i eth0 -f /etc/ptp4l_cfg/automotive-slave.cfg --step_threshold=1 -m &
port 1: INITIALIZING to SLAVE on INIT_COMPLETE
ptp4l[2204.752]: port 0: INITIALIZING to LISTENING on INIT_COMPLETE
```

(9) Execute PTP master on platform-2 with automotive-master.cfg configuration file

```
# ptp4l -i eth0 -f /etc/ptp4l_cfg/automotive-master.cfg --step_threshold=1 -m &
ptp4l[2114.520]: port 1: INITIALIZING to MASTER on INIT_COMPLETE
ptp4l[2114.520]: port 0: INITIALIZING to LISTENING on INIT_COMPLETE
```

3.1.3 Test Results

Sync log will show on platform -1, as follows, and master offset value should be smaller than 100ns

```
ptp4l[361.950]: rms 15002 max 22371 freq -20066 +/- 2791 delay   774 +/-   0
ptp4l[362.950]: rms 2665 max 5325 freq -10086 +/- 2564 delay   779 +/-   0
ptp4l[363.950]: rms 2368 max 2613 freq  -3772 +/- 1078 delay   785 +/-   0
ptp4l[364.951]: rms 2004 max 2471 freq  -1853 +/- 139 delay   787 +/-   0
ptp4l[365.951]: rms  850 max 1264 freq  -2035 +/- 171 delay   785 +/-   0
ptp4l[366.951]: rms  127 max  263 freq  -2646 +/- 132 delay   785 +/-   0
ptp4l[367.951]: rms  147 max  180 freq  -2999 +/-  77 delay   787 +/-   0
ptp4l[368.952]: rms  113 max  137 freq  -3099 +/-  21 delay   783 +/-   0
ptp4l[369.952]: rms   48 max   86 freq  -3085 +/-  21 delay   787 +/-   0
ptp4l[370.953]: master offset         -9 s3 freq   -3056 path delay       787
ptp4l[371.953]: master offset         21 s3 freq   -3029 path delay       787
ptp4l[372.953]: master offset         37 s3 freq   -3007 path delay       787
ptp4l[373.953]: master offset         21 s3 freq   -3011 path delay       787
ptp4l[374.953]: master offset        -18 s3 freq   -3044 path delay       787
ptp4l[375.953]: master offset         22 s3 freq   -3010 path delay       787
ptp4l[376.953]: master offset        -17 s3 freq   -3042 path delay       787
ptp4l[377.953]: master offset          7 s3 freq   -3023 path delay       787
ptp4l[378.953]: master offset         -1 s3 freq   -3029 path delay       787
ptp4l[379.953]: master offset         -1 s3 freq   -3029 path delay       787
```

The slave offset from master in startup stage and the offset quickly decreases in 10s to get a stable state as shown in Figure 4.

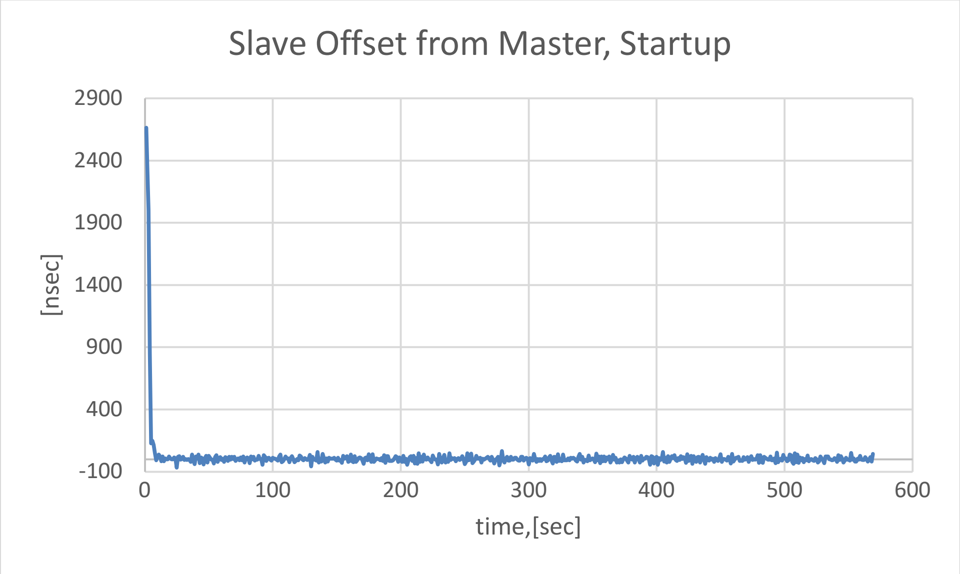


Figure 3-3. Slave Offset from Master, in Startup State

The slave offset from master in stable stage and the fluctuation range of offset is[-68,66] as shown in Figure 5.

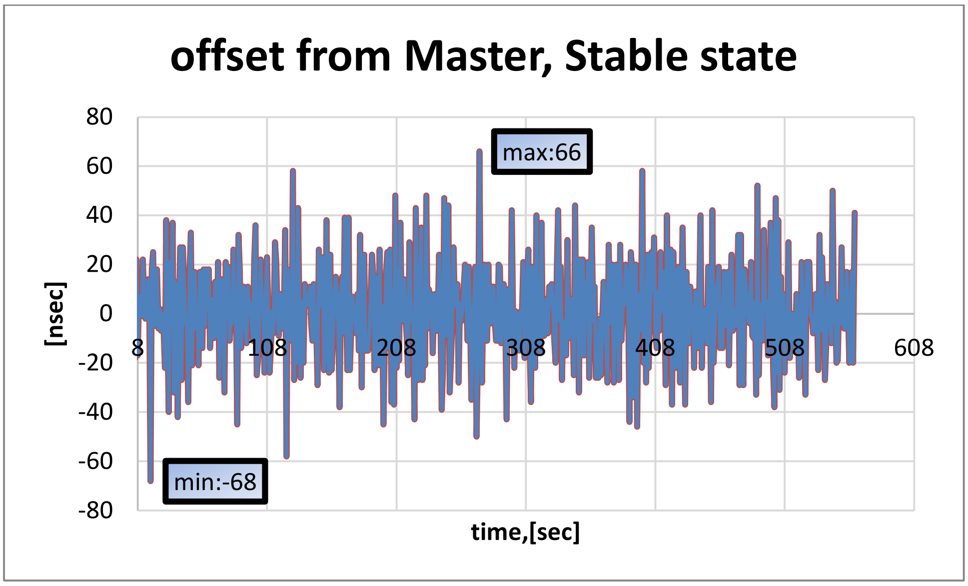


Figure 3-4. Slave Offset from Master, in Stable State

3.2 802.1Qav Test

3.2.1 80.21Qav Introduction

802.1Qav allows to provide guarantees for time-sensitive (i.e. bounded latency and delivery variation), loss-sensitive real-time audio video (AV) data transmission (AV traffic). It specifies per priority ingress metering, priority regeneration, and timing-aware queue draining algorithms. Virtual Local Area Network (VLAN) tag encoded priority values are allocated, in aggregate, to segregate frames among controlled and non-controlled queues, allowing simultaneous support of both AV traffic and other bridged traffic over and between Local Area Networks (LANs). Credit-based shaping is an alternative scheduling algorithm used in network schedulers to achieve fairness when sharing a limited network resource. Here we present the steps taken for setting up a test that the CBS qdisc setting on platform-1, and packets received on platform-2 will indicate that CBS qdisc takes effect.

3.2.2 Test Steps

(1) Execute steps1-6 in 3.1.2 to make sure the path link between platform-1&2 is ready.

(2) Map skb->priority to traffic class on platform-1

3pri -> tc3, 2pri -> tc2, (0,1,4-7)pri -> tc0

Map traffic class to transmit queue on platform-1

tc0 -> txq0, tc2 -> txq2, tc3 -> txq3

```
# tc qdisc replace dev eth0 parent root mqprio num_tc 4 map 0 0 2 3 0 0 0 0 0 0 0 0 0 0
0 0 queues 1@0 1@1 1@2 1@3 hw 0
```

(3) Check classes settings on platform-1 and logs will show as follows on platform-1

```
# tc -g class show dev eth0
+---(8001:ffe3) mqprio
|    +---(8001:4) mqprio
|
+---(8001:ffe2) mqprio
|    +---(8001:3) mqprio
|
+---(8001:ffe1) mqprio
|    +---(8001:2) mqprio
|
+---(8001:ffe0) mqprio
+---(8001:1) mqprio
```

(4) Configure CBS qdisc to reserve 20Mbps bandwidth for traffic class-A (priority 3) of Queue3 on platform-1

```
# tc qdisc replace dev eth0 parent 8001:4 cbs locredit -1470 hicredit 30 sendslope -
980000 idleslope 20000 offload 1
```

(5) Configure CBS qdisc to reserve 20Mbps bandwidth for traffic class-B (priority 2) of Queue2 on platform-1

```
# tc qdisc replace dev eth0 parent 8001:3 cbs locredit -1470 hicredit 61 sendslope -
980000 idleslope 20000 offload 1
# tc qdisc show dev eth0
qdisc mqprio 8001: root tc 4 map 0 0 2 3 0 0 0 0 0 0 0 0 0 0 0 0
      queues:(0:0) (1:1) (2:2) (3:3)
qdisc pfifo 0: parent 8004: limit 1000p
qdisc pfifo 0: parent 8003: limit 1000p
qdisc pfifo_fast 0: parent 8001:2 bands 3 priomap 1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
qdisc pfifo_fast 0: parent 8001:1 bands 3 priomap 1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
qdisc cbs 8003: parent 8001:4 hicredit 30 locredit -1470 sendslope -980000 idleslope 20000 offload
1
qdisc cbs 8004: parent 8001:3 hicredit 61 locredit -1470 sendslope -980000 idleslope 20000 offload
1
```

(6) Create vlan 100 to map sk->priority to vlan QoS on platform-1

```
# ip link add link eth0 name eth0.100 type vlan id 100
```

(7) Map skb->priority to L2 priority, 1 to 1 on platform-1

```
# ip link set eth0.100 type vlan egress 0:0 1:1 2:2 3:3 4:4 5:5 6:6 7:7
```

(8) Up eth0.100 VLAN NIC node on platform-1

```
# ifconfig eth0.100 up
```

(9) Create vlan 100 on platform-2

```
# ip link add link eth0 name eth0.100 type vlan id 100
```

(10) Up eth0.100 VLAN NIC node on platform-2

```
# ifconfig eth0.100 up
```

(11) Execute listener on platform-2 to receive specific class-A and class-B traffic

```
# tsn_listener -d 00:11:22:33:44:55 -i eth0.100 -s 1500 &
```

(12) Execute talker on platform-1 to send class-A(priority 3) traffic to Queue3

```
# tsn_talker -d 00:11:22:33:44:55 -i eth0.100 -p3 -s 1500&
```

(13) Execute talker on platform-1 to send class-B(priority 2) traffic to Queue2

```
# tsn_talker -d 00:11:22:33:44:55 -i eth0.100 -p2 -s 1500&
```

(14) Excute iperf3 serve on platform-2

```
# iperf3 -s -i 5 &
```

(15) Excute iperf3 client on platform-1 to send best effort traffic to Queue0

```
# iperf3 -c 192.168.0.10 -t 300 -i 5 &
```

3.2.3 Test Results

(1) Reserved bandwidth for class-A traffic of Queue3 is 20Mbps, and result bandwidth shown in log is 19.8Mbps. The listener only gets the payload of the packet to statistics and the head of ethernet packet is ignored.

```
Receiving data rate: 19884 kbps
Receiving data rate: 19872 kbps
Receiving data rate: 19884 kbps
Receiving data rate: 19872 kbps
Receiving data rate: 19884 kbps
Receiving data rate: 19884 kbps
Receiving data rate: 19872 kbps
```

(2) Reserved bandwidth for class-B traffic of Queue2 is 20Mbps, and result bandwidth is 19.8Mbps. The listener only gets the payload of the packet to statistics and the head of ethernet packet is ignored.

```
Receiving data rate: 19884 kbps
Receiving data rate: 19872 kbps
Receiving data rate: 19884 kbps
Receiving data rate: 19872 kbps
Receiving data rate: 19884 kbps
Receiving data rate: 19884 kbps
Receiving data rate: 19872 kbps
```

(3) Total reserved bandwidth for class-A traffic of Queue3 and class-B traffic of Queue2 is 40Mbps, and result bandwidth is 39.7Mbps. The listener only gets the payload of the packet to statistics and the head of ethernet packet is ignored.

```
Receiving data rate: 39756 kbps
Receiving data rate: 39756 kbps
Receiving data rate: 39756 kbps
Receiving data rate: 39756 kbps
Receiving data rate: 39768 kbps
Receiving data rate: 39756 kbps
Receiving data rate: 39756 kbps
```

(4) High priority traffic reserved bandwidth can be seized by lower priority traffic

1) First stage: Send traffic class-A stream to Queues3 and traffic class-B stream to Queue2. Send best effort stream to Queue0. The total bandwidth of NIC is about 942Mbps. Queue3 and Queue2 have high propriety than Queue0, so the actual bandwidth is aligned with bandwidth reserved for Queue3 and Queue2 (total 40Mbps) and Queue0 takes rest 904Mbps bandwidth.

```
[  5]  94.00-96.00  sec   215 MBytes   904 Mbits/sec
Receiving data rate: 39000 kbps
Receiving data rate: 39024 kbps
[  5]  96.00-98.00  sec   215 MBytes   904 Mbits/sec
Receiving data rate: 39000 kbps
Receiving data rate: 39000 kbps
[  5]  98.00-100.00 sec   215 MBytes   904 Mbits/sec
Receiving data rate: 39000 kbps
Receiving data rate: 39024 kbps
[  5] 100.00-102.00 sec   215 MBytes   904 Mbits/sec
Receiving data rate: 39000 kbps
Receiving data rate: 36516 kbps
[  5] 102.00-104.00 sec   217 MBytes   910 Mbits/sec
```

2) Second stage: Send traffic class-A stream to Queues3. Send best effort stream to Queue0. The total bandwidth of NIC is about 942Mbps. Queue3 and Queue2 have high propriety than Queue0, even the reserved bandwidth of Queue3 and queue2 is total 40Mbps, but there is no traffic in Queue2. The reserved bandwidth of Queue2 is seized by Queue0, so Queue0 takes 923Mbps bandwidth.

```
Receiving data rate: 19500 kbps
Receiving data rate: 19500 kbps
[  5] 104.00-106.00 sec   220 MBytes   923 Mbits/sec
Receiving data rate: 19512 kbps
Receiving data rate: 19500 kbps
[  5] 106.00-108.00 sec   220 MBytes   923 Mbits/sec
Receiving data rate: 19512 kbps
Receiving data rate: 19500 kbps
[  5] 108.00-110.00 sec   220 MBytes   923 Mbits/sec
Receiving data rate: 19500 kbps
Receiving data rate: 17544 kbps
[  5] 110.00-112.00 sec   221 MBytes   929 Mbits/sec
```

3) Third stage: There is no traffic in Queue3 and Queue2 and the bandwidth reserved for Queue3 and Queue2 is seized by Queue0, so Queue0 takes the whole 942Mbps bandwidth.

```
Receiving data rate: 0 kbps
Receiving data rate: 0 kbps
[  5] 112.00-114.00 sec   224 MBytes   941 Mbits/sec
Receiving data rate: 0 kbps
Receiving data rate: 0 kbps
[  5] 114.00-116.00 sec   225 MBytes   942 Mbits/sec
Receiving data rate: 0 kbps
Receiving data rate: 0 kbps
[  5] 116.00-118.00 sec   224 MBytes   941 Mbits/sec
```

3.3 Earliest TxTime First(ETF) Test

3.3.1 ETF Introduction

The ETF (Earliest TxTime First) qdisc allows applications to control the instant when a packet should be dequeued from the traffic control layer into the network device. If offload is configured and supported by the network interface card, then it will also control when packets leave the network controller. The ETF provides per-queue TxTime-based scheduling also known as Time-Based Scheduling[5]. TxTime is the launch time for NIC TX Hardware. Here we present the steps taken for setting up a test that uses *only* the ETF qdisc. That means that only Time-based transmission is exercised.[2]
The 'talker' side of the example will transmit a packet every 1ms. The packet's txtime is set through the SO_TXTIME API, and is copied into the packet's payload. At the 'listener' side, we capture traffic and then post-process it to compute the delta between each packet's arrival time and their txtime.
ptp4l is used for synchronizing the PHC clocks over the network and phc2sys is used on the 'talker' and 'listener' side for synchronizing the system clock to the PHC. CLOCK_TAI is the reference clockid used throughout the example for the qdiscs and the applications. In Linux，ETF hardware character is enabled by SO_TXTIME(socket option) and ETF qdisc.

SO_TXTIME allows the application to set the sending time for each packet, ETF qdisc ensures that packets from multiple sockets are sent to the network card in chronological order. Like CBS qdisc, ETF qdisc is based on a single queue, so a configuration similar to mqprio is the premise.
If Q3 in MQ is set to ETF qidsc and offload is enabled to support launch time, the following commands are required:

*tc qdisc replace dev eth0 parent 100:4 etf *
*clockid CLOCK_TAI *
*delta 200000 *
offload

The parameter clockid is used to specify which clock is the benchmark of packet transmission time. Currently only CLOCK_TAI is supported. ETF requires the system clock to be synchronized with the PTP hardware clock. The parameter delta specifies how long before the sending time, and the packet should be sent to the hardware. This value is related to many factors, and 200us is used in the example.
For more information, please refer to tc-etf(8).

3.3.2 Test Steps

3.3.2.1 Time Synchronization

(1) Execute steps1-9 in 3.1.2 to run basic PTP engine

(2) Capture UDP packets on platform-2 for statistical analysis

```
# tcpdump -c 20000 -i eth0 -w tmp.pcap -j adapter_unsynced -tt --time-stamp-
precision=nano udp port 7788&
```

(3) Sync PHC to system time on platform-1&2

```
# phc2sys -s eth0 -c CLOCK_REALTIME --step_threshold=1 --transportSpecific=1 -w&
```

(4) Adjust CLOCK_TAI 37s offset on platform-1&2

```
# adjust_clock_tai_offset
```

(5) Check CLOCK_TAI is synchronized with PHC or not on platform-1 & 2, and log shows as follows. The value of phc-tai delta should be smaller than 10000 and it means that time synchronization is already done.

```
# check_clocks eth0
console:/data/local # ./check_clocks eth0
rt tstamp:      1637576933896599762
tai tstamp:     1637576970896602224
phc tstamp:     1637576970896607395
rt latency:     77
tai latency:    384
phc latency:    837
phc-rt delta:   37000007633
phc-tai delta:  5171
```

3.3.2.2 ETF Qdisc Configuration and Send Packets

(1) Configure ETF Hardware offload qdisc on platform-1

```
# sh /etc/tsn-scripts/config-etf-offload.sh eth0
# tc qdisc show dev eth0
qdisc mqprio 100: root tc 4 map 0 1 2 3 0 0 0 0 0 0 0 0 0 0 0 0
queues:(0:0) (1:1) (2:2) (3:3)
qdisc etf 8001: parent 100:4 clockid TAI delta 200000 offload on deadline_mode off skip_sock_check
off
qdisc pfifo_fast 0: parent 100:3 bands 3 priomap 1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
qdisc pfifo_fast 0: parent 100:2 bands 3 priomap 1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
qdisc pfifo_fast 0: parent 100:1 bands 3 priomap 1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
```

(2) Send UDP packets with preset timestamps on platform-1

```
# sh /etc/tsn-scripts/run-udp-tai-tc-etf.sh eth0
```

(3) UDP packets capture is done on platform-2 after about 2 min later and log shows as follows

```
console:/data/local # 20000 packets captured
20064 packets received by filter
0 packets dropped by kernel
```

3.3.3 Test Results

3.3.3.1 Traffic Analysis and Generate Statistics

(1) Connect Ubuntu machine and platform-2 with Ethernet cable and set Ubuntu IP address

```
# ifconfig ethx 192.168.0.100
```

where ethx stands for the name of NIC on Ubuntu machine

(2) Put captured file tmp.pcap in step-2 of 3.3.2.1 from platform-2 to Ubuntu machine by tftp

```
# ifconfig eth0 192.168.0.10
# tftp 192.168.0.100
# put tmp.pcap
```

(3) Use tshark tool to analyze traffic on Ubuntu machine

```
# tshark -r tmp.pcap --disable-protocol dcp-etsi --disable-protocol dcp-pft -t e -E separator=, -T fields -e frame.number -e frame.time_epoch -e data.data > tmp.out
```

(4) Use txtime_offset_stats.py tool to get statistics result on Ubuntu machine and log shows as follows, average statistics result should be smaller than 10e+4

```
# ./txtime_offset_stats.py -f tmp.out
min:            3.019500e+04
max:            5.795100e+04
jitter(pk-pk):  2.775600e+04
avg:            3.174164e+04
std dev:        3.436093e+02
count:          20000
```

3.3.3.2 Packet TX Flow Timestamps Diagram on platform-1

The TX flow timestamps of the packet from user space to Hardware is shown as Figure 6. txtime is the instant when a packet should be sent by Hardware. A polling timer is executed is user space for detecting CLOCK_TAI time and once CLOCK_TAI time is in [txtime-600, txtime] the user space sending packet process will be woke up. It takes about 12us from waking up to sending out the packet in user space. The packet is passed from user space to kernel space and it takes about 8us when it arrives network stack. Another 11us is spent when the packet is sent from network stack and arrives Qdisc enqueue. The packet is not dequeued until CLOCK_TAI timestamp is in [txtime-200us, txtime] and it takes 181us from packet dequeue to arriving driver layer. Finally, the packet is sent out by Hardware at txtime. All the timestamps in TX flow are got by ftrace tool.

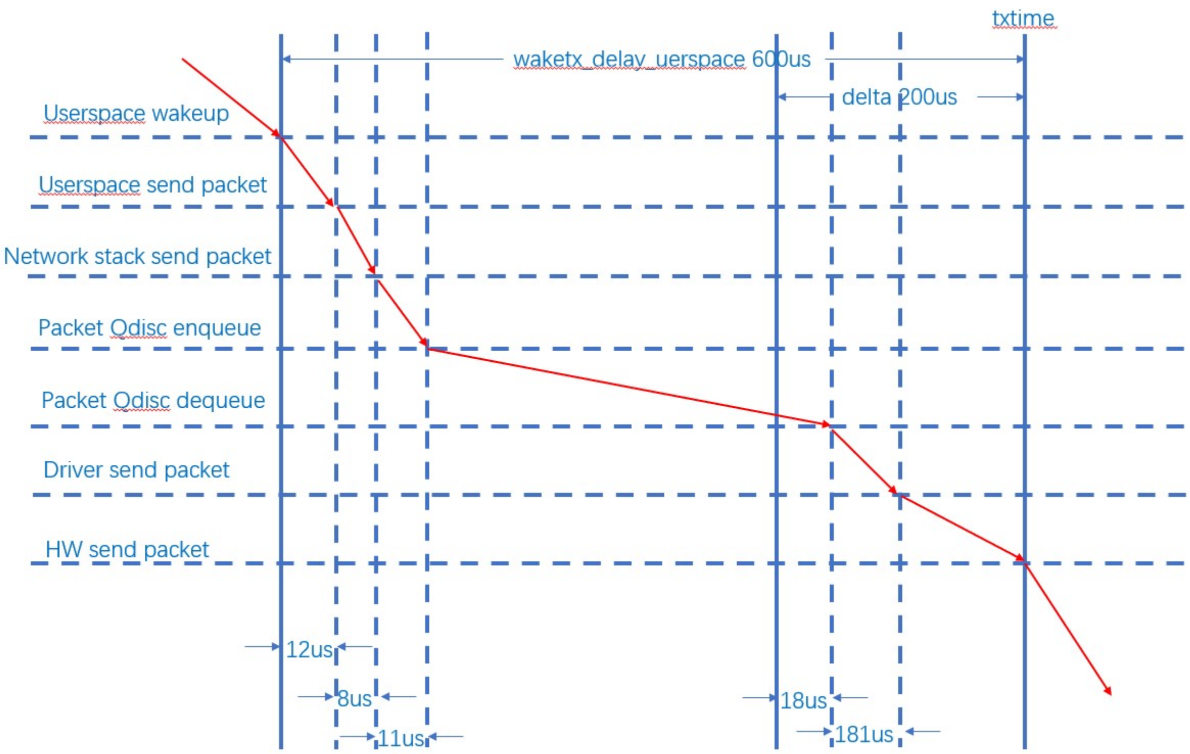


Figure 3-5. ETF Qdisc Hardware Offload TX Flow Timestamps

3.3.3.3 Packet RX Flow Timestamps Diagram on platform-2

The RX flow timestamps of the packet from Hardware to user space is show as Figure 7. It takes about 2.2us from TX Hardware sending out the packet to RX Hardware receiving it. Another 44us is spent for RX driver ISR receiving the packet. Network stack gets the packet about 48us after RX driver ISR. There is a user space process to receive the packet with specific UDP port number. It needs 5.8us from kernel space to user space process receiving the packet. All the timestamps in RX flow are got by ftrace tool.

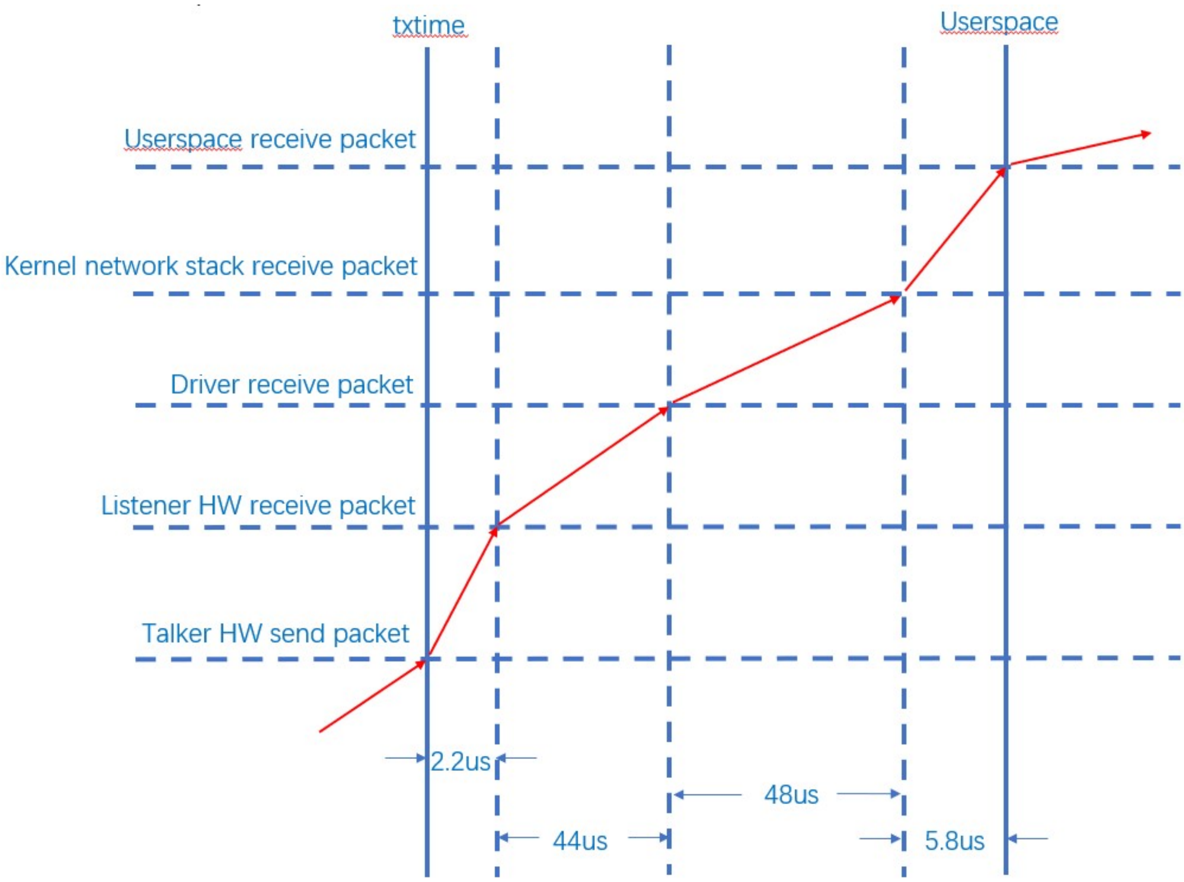


Figure 3-6. ETF Qdisc Hardware Offload RX Flow Timestamps

3.4 802.1Qbv Test

3.4.1 802.1Qbv Introduction

The TAPRIO(Time Aware Priority) qdisc implements a simplified version of the scheduling state machine defined by IEEE 802.1Q-2018 Section 8.6.9, which allows configuration of a sequence of gate states, where each gate state allows outgoing traffic for a subset (potentially empty) of traffic classes. The TAORIO qdisc is the per-port Time-Aware schedule qdisc[6]. Here we present the steps taken for setting up a test that uses both the ETF qdisc and the TAPRIO one[2] as ETF qdisc with a specific txtime can help us to analyze the TAPRIO qdisc more accurately. That means that we'll use a (Qbv-like) port scheduler with a fixed Tx schedule for traffic classes (TC), while using Time-based transmission for controlling the Tx time of packets within each TC.

The port schedule is thus comprised by 4 time-slices, with a total cycle-time of 1 millisecond allocated as:

- Traffic Class 3 (TC 3): duration of 300us, 'hw offload' is used.
- Traffic Class 2 (TC 2): duration of 300us, 'hw offload with deadline txtime' is used.

- Traffic Class 1 (TC 1): duration of 100us, not used.
- Traffic Class 0 (TC 0): duration of 300us, best-effort traffic.

```
|___x_|......D_|  ......  |bbbbbbbbbb|
0      299       599    699               999us
```

Talker side uses udp_tai to send time-sensitive traffic. TC3 / 2 uses different UDP ports so that the listener can identify which TC's traffic is through the port. Tc0 can use PTP packets as best effort traffic.

The current test is based on the implementation of Hardware offload + deadline of ETF. According to their timing relationship, the settings are as follows：

1. TC3 uses strict ETF to send, and packets will be sent between [txtime – Delta, txtime], so set the base time offset to 50us.
2. TC2 uses deadline ETF to send packets, and the packets will be sent in around txtime – waketx_delay, set the base time offset to 450us and waketx_delay is set to 600us.
3. TC0 uses PTP packets as best effort traffic.

On the listener side, capture packets by tcpdump tool and obtain the GCL settings on the talker side for subsequent analysis:

1. Determine which TC the packet belongs to according to the UDP port
2. For each packet, tcpdump will record the arrival time. Because the path delay in talker listener is very small, it can be considered that the time recorded by tcpdump is almost the time when the packet is sent.
3. According to the base time and arrival time of talker's GCL, the entry corresponding to this packet can be calculated to see whether the entry is open or closed
4. If the settings of open / closed and TC of the entry can correspond, it can be regarded as arriving on time.

Like ETF, the above process is based on PTP synchronization, so ptp4l / phc2sys should still be used to establish the synchronization mechanism of talker / listener. CLOCK_TAI is still the baseline clockid for qdiscs / applications.

3.4.2 Test Steps

3.4.2.1 Time Synchronization

(1) Execute steps1-9 in 3.1.2 to run basic PTP engine.

(2) Capture UDP packets on platform-2 for statistics analysis

```
# tcpdump -c 20000 -i eth0 -w tmp.pcap -j adapter_unsynced -tt --time-stamp-
precision=nano "inbound"&
```

(3) Enable Hardware timestamps for RX packets on platform-2

```
# hwstamp_ctl -i eth0 -r 1
```

(4) Sync PHC to system time on platform-1&2

```
# phc2sys -s eth0 -c CLOCK_REALTIME --step_threshold=1 --transportSpecific=1 -w&
```

(5) Adjust CLOCK_TAI 37s offset on platform-1&2

```
# adjust_clock_tai_offset
```

(6) Check CLOCK_TAI is synchronized with PHC or not on platform-1 & 2, and log shows as follows. The value of phc-tai delta should be smaller than 10000 and it means that time synchronization is already done.

```
# check_clocks eth0
console:/data/local # ./check_clocks eth0
rt tstamp:      1637576933896599762
tai tstamp:     1637576970896602224
phc tstamp:     1637576970896607395
rt latency:     77
tai latency:    384
phc latency:    837
phc-rt delta:   37000007633
phc-tai delta:  5171
```

3.4.2.2 TAPRIO Configuration and Send Packets

(1) Configure TAPRIO Hardware offload qdisc on platform-1

```
# sh /etc/tsn-scripts/config-taprio-offload.sh eth0
# tc qdisc show dev eth0
qdisc taprio 100: root tc 4 map 0 1 2 3 0 0 0 0 0 0 0 0 0 0 0 0
queues offset 0 count 1 offset 1 count 1 offset 2 count 1 offset 3 count 1
        clockid invalid flags 0x2       base-time 1600598754000000000 cycle-time 1000000 cy-cle-
time-extension 0
       index 0 cmd S gatemask 0x8 interval 300000
       index 1 cmd S gatemask 0x4 interval 300000
       index 2 cmd S gatemask 0x2 interval 100000
       index 3 cmd S gatemask 0x1 interval 300000
qdisc etf 8001: parent 100:4 clockid TAI delta 200000 offload on deadline_mode off skip_sock_check
off
qdisc pfifo 0: parent 100:2 limit 1000p
qdisc pfifo 0: parent 100:1 limit 1000p
qdisc etf 8002: parent 100:3 clockid TAI delta 200000 offload on deadline_mode on skip_sock_check
off
```

(2) Send UDP packets on platform-1

```
# sh /etc/tsn-scripts/run-udp-tai-tc-taprio.sh eth0
```

(3) Packets capture is done on platform-2 after about 2 min later and log shows as follows

```
console:/data/local # 20000 packets captured
20064 packets received by filter
0 packets dropped by kernel
```

3.4.3 Test Results

3.4.3.1 Traffic Analysis

(1) Connect Ubuntu machine and platform-2 with Ethernet cable and set Ubuntu IP address

```
# ifconfig ethx 192.168.0.100
```

where ethx stands for the name of NIC on Ubuntu machine

(2) Put captured file tmp.pcap in step-2 of 3.3.2.1 from platform-2 to Ubuntu machine by tftp

```
# ifconfig eth0 192.168.0.10
# tftp 192.168.0.100
# put tmp.pcap
```

(3) Put qdisc configuration file taprio.batch from platform-1 to Ubuntu machine by tftp

```
# ifconfig eth0 192.168.0.10
# tftp 192.168.0.100
# put taprio.batch
```

(4) The last 10000 packets need to be analysis, cut tmp.pcap into two small pcap file by editcap tool

```
# editcap tmp.pcap -c 10000 tmp-cut.pcap
```

(5) Use dump-classifier to analyze traffic on Ubuntu and none of “late” log will show on console

```
# dump-classifier -d tmp-cut_00001_xxx.pcap -f filter -s taprio.batch | grep -v ontime
```

3.4.3.2 Packets Arrive Time Liner at Each Open Gate

That packet arrive time of strict and deadline is liner at open gate0 and gate1 and it is aligned with user space packet sending period 1ms as shown in Figure 8.

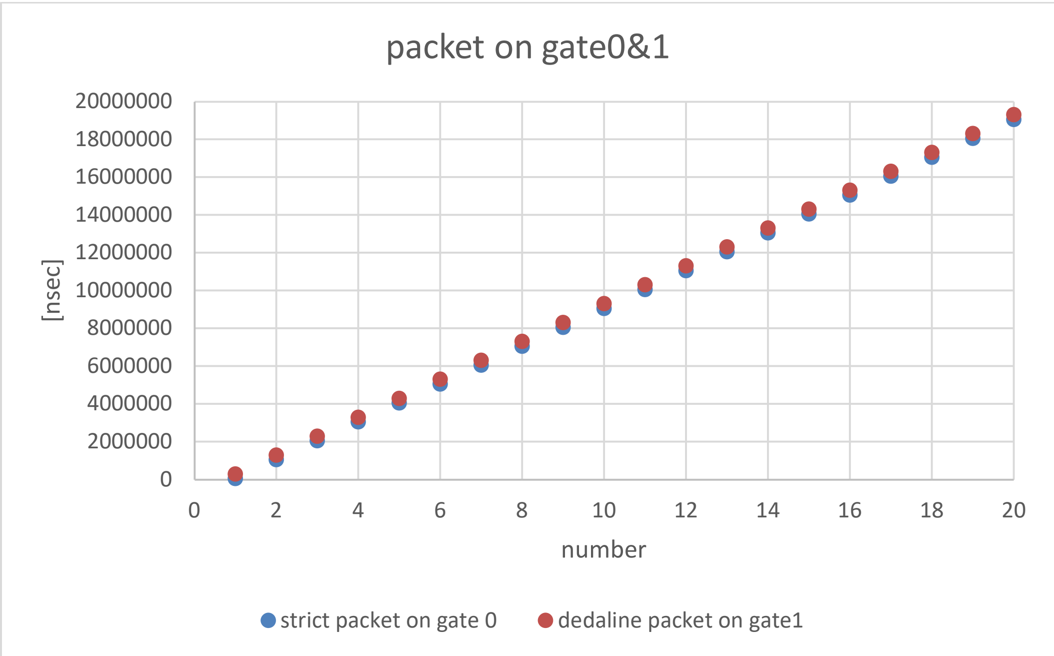


Figure 3-7. Packets Liner at Each Open Gate

3.4.3.3 Packets Arrive Offset from Gate Open

Strict packets arrive offsets from gate-0 open time and the offset is stable around 53us and aligned with setting txtime 50us as shown in Figure 9.

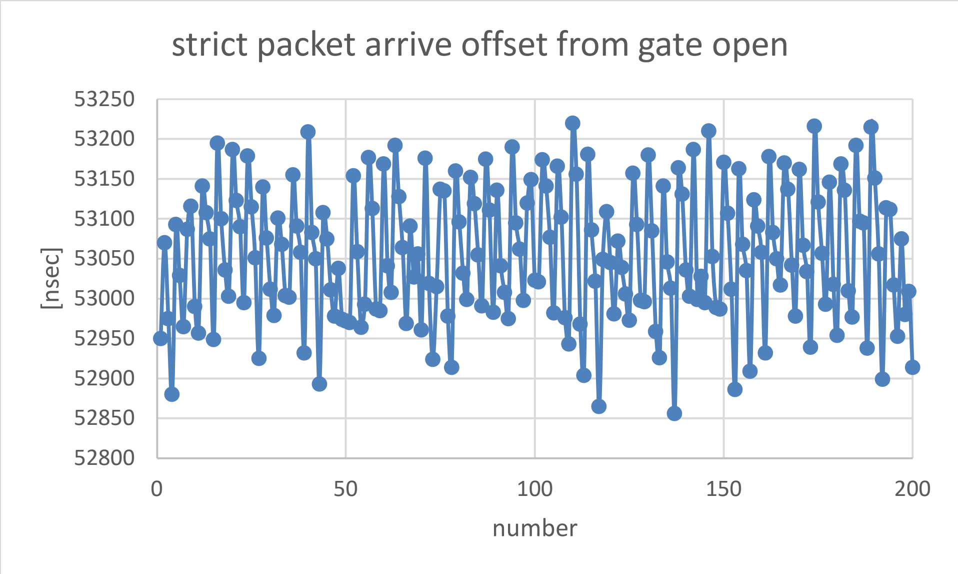


Figure 3-8. Strict Packets Arrive Offset from Gate-0 Open

Deadline packets arrive offsets from gate-1 open time and the offset is stable around 1us as shown in Figure 10. The txtime of deadline packet is modified to be 'now' of CLOCK_TAI when dequeued which is earlier than gate-1 open time, so the deadline packet is sent out by Hardware once the gate-1 is open.

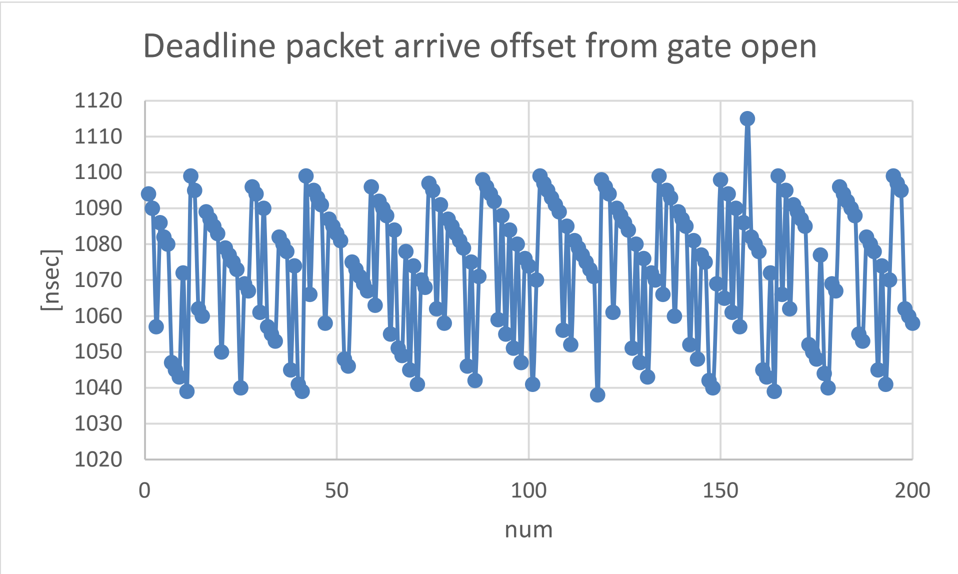


Figure 3-9. Deadline Packets Arrive Offset from Gate-1 Open

3.5 802.1Qbu Test

3.5.1 802.Qbu Introduction

A large, non-time-critical frame may start ahead of time-critical frame transmission. This condition leads to excessive latency for the time-critical frame. The lack of transmission preemption severely inhibits the capabilities of an application that uses scheduled frame transmission to implement a real-time control network. 802.1Qbu define a class of service for time-critical frames that requests the transmitter in a bridged Local Area Network to suspend the transmission of a non-time-critical frame and allow for one or more time-critical frames to be transmitted. When the time-critical frames have been transmitted, the transmission of the preempted frame is resumed. A non-time-critical frame could be preempted multiple times. Here we present the steps taken for setting up a test that the FPE(Frame Preemption) qdisc setting on platform-1, and FPE packets received on platform-1&2 will indicate that FPE qdisc takes effect.

3.5.2 Test Steps

3.5.2.1 Time Synchronization

(1) Execute steps1-9 in 3.1.2 to run basic PTP engine.

(2) Enable Hardware timestamps for RX packets on platform-1&2

```
# hwstamp_ctl -i eth0 -r 1
```

(3) Sync PHC to system time on platform-1 & 2

```
# phc2sys -s eth0 -c CLOCK_REALTIME --step_threshold=1 --transportSpecific=1 -w&
```

(4) Adjust CLOCK_TAI 37s offset on platform-1 & 2

```
# adjust_clock_tai_offset
```

(5) Check CLOCK_TAI is synchronized with PHC or not on platform-1 & 2, and log show as follows, value of phc-tai delta should be smaller than 10000, it means that time synchronization is already done

```
# check_clocks eth0
console:/data/local # ./check_clocks eth0
rt tstamp:      1637576933896599762
tai tstamp:     1637576970896602224
phc tstamp:     1637576970896607395
rt latency:     77
tai latency:    384
phc latency:    837
phc-rt delta:   37000007633
phc-tai delta:  5171
```

3.5.2.2 FPE Configuration and Send Packets

(1) Use iperf3 to receive traffic on platform-2

```
# iperf3 -s -i 2 &
```

(2) Configure FPE Hardware engine qdisc on platform-1&2

```
# sh /etc/tsn-scripts/config-fpe.sh eth0
# tc qdisc show dev eth0
qdisc taprio 100: root tc 4 map 0 1 2 3 0 0 0 0 0 0 0 0 0 0 0 0
queues offset 0 count 1 offset 1 count 1 offset 2 count 1 offset 3 count 1
        clockid invalid flags 0x2       base-time 1600599351000000000 cycle-time 1000000 cy-cle-
time-extension 0
        index 0 cmd SH gatemask 0x8 interval 300000
        index 1 cmd SH gatemask 0x4 interval 300000
        index 2 cmd SH gatemask 0x2 interval 100000
        index 3 cmd SR gatemask 0x1 interval 300000
qdisc etf 8001: parent 100:4 clockid TAI delta 200000 offload on deadline_mode off skip_sock_check
off
qdisc pfifo 0: parent 100:2 limit 1000p
qdisc pfifo 0: parent 100:1 limit 1000p
qdisc etf 8002: parent 100:3 clockid TAI delta 200000 offload on deadline_mode on skip_sock_check
off
```

(3) Use iperf3 to send traffic on platform-1

```
# iperf3 -c 192.168.0.10 -b 10M -i 2 -t 10 &
```

3.5.3 Test Results

(1) Get count of preempted packets on platform-1&2, log shows as follows, and mmc_tx_fpe_fragment_cntr should be equal to mmc_rx_fpe_fragment_cntr

```
# ethtool -S eth0 | grep fpe
platform-1:
mmc_tx_fpe_fragment_cntr: 221
mmc_rx_fpe_fragment_cntr: 0
platform-2:
mmc_tx_fpe_fragment_cntr: 0
mmc_rx_fpe_fragment_cntr: 221
```

4 Quick Start Test Guide

4.1 802.1AS

1. Connect eth0 interface on platform-1 & platform-2

2. Kill NetworkManager process on platform-1&2 to avoid losing IP address and qdisc

```
# ps -aux | grep NetworkManager
# kill xxx
```

where xxx stands for PID of NetworkManger process,

3. Disable EEE on platform-1&2 to reduce path delay caused by PHY enter and exit EEE mode

```
# ethtool --set-eee eth0 eee off advertise 0
```

4. Set platform-2 MAC and IP address

```
# ifconfig eth0 down
# ifconfig eth0 hw ether 00:11:22:33:44:55 up
# ifconfig eth0 192.168.0.10
```

5. Set platform-1 IP address and ping platform-2 to confirm link path is ready

```
# ifconfig eth0 192.168.0.1
# ping 192.168.0.1 -c 3
```

6. Run ptp4l slave on platform-1 with automotive-slave.cfg configuration file and ptp4l master on platform-2 with automotive-slave.cfg configuration file

```
# ptp4l -i eth0 -f /etc/ptp4l_cfg/automotive-slave.cfg  --step_threshold=1 -m &
# ptp4l -i eth0 -f /etc/ptp4l_cfg/automotive-master.cfg  --step_threshold=1 -m &
```

7. After running ptp4l, platform-2 works as master and platform-1 works as slave, and synchronization messages would be printed on platform-1.

4.2 802.1Qav

1. Run steps1-5 in 4.1 to make sure the path link between platform-1 & 2 is ready.

2. Map skb->priority to traffic class on platform-1

3pri -> tc3, 2pri -> tc2, (0,1,4-7)pri -> tc0
Map traffic class to transmit queue on platform-1
tc0 -> txq0, tc2 -> txq2, tc3 -> txq3

```
# tc qdisc replace dev eth0 parent root mqprio num_tc 4 map 0 0 2 3 0 0 0 0 0 0 0 0 0 0
0 0 queues 1@0  1@1 1@2 1@3 hw 0
```

3. Set bandwidth of queue3-class-A and queue2-class-B, such as queue3-20Mbps and queue2-20Mbps on platform-1

```
# tc qdisc replace dev eth0 parent 8001:4 cbs locredit -1470 hicredit 30 send-slope -
980000 idleslope 20000 offload 1
# tc qdisc replace dev eth0 parent 8001:3 cbs locredit -1470 hicredit 61 send-slope -
980000 idleslope 20000 offload 1
```

locredit/hicredit/sendslop/idleslope parameters can be got by calculate_cbs_params.py, such as setting class-A XXXKbps and class-B YYYKbps (pls refer to [4] to get calculate_cbs_params.py)

```
# calculate_cbs_larams.py -A XXX -a 1500 -B YYY -b 1500
```

4. As CBS is based on VLAN, configure VLAN on platform-1 and platform-2

```
# ip link add link eth0 name eth0.100 type vlan id 100
# ip link set eth0.100 type vlan egress 0:0 1:1 2:2 3:3 4:4 5:5 6:6 7:7
# ifconfig eth0.100 up
```

5. Receive specific class-A and class-B traffic on platform-2

```
# tsn_listener -d 00:11:22:33:44:55 -i eth0.100 -s 1500 &
```

6. Send stream to queue3 and get result bandwidth is 19+Mbps

```
# tsn_talker -d 00:11:22:33:44:55 -i eth0.100 -p3 -s 1500&
```

7. Send stream to queue2 and get result total bandwidth is 38+Mbps

```
# tsn_talker -d 00:11:22:33:44:55 -i eth0.100 -p2 -s 1500&
```

4.3 802.1Qbv

1. As 802.1Qbv relies on time synchronization, run steps1-6 in 4.1 to make sure time synchronization is done between platform-1&2.

2. Capture streams on platform-2

```
# tcpdump -c 20000 -i eth0 -w tmp.pcap -j adapter_unsynced -tt --time-stamp-
precision=nano "inbound"&
```

3. Enable Hardware timestamps for RX packets on platform-2

```
# hwstamp_ctl -i eth0 -r 1
```

4. Sync PHC to system clock on platform-1 & 2

```
# phc2sys -s eth0 -c CLOCK_REALTIME --step_threshold=1 --transportSpecific=1 -w&
# adjust_clock_tai_offset
# check_clocks eth0
```

5. Get basetime + 2minutes by shell script

```
# sh /etc/tsn-scripts/base-time.sh
```

6. Set time schedule, open queue3 300us, open queue2 300us, open queue1 100us, open queue0 300us, replace $BASE_TIME in command based on step-5

```
# tc qdisc replace dev eth0 parent root handle 100 taprio num_tc 4 map 0 1 2 3 0 0 0 0 0
0 0 0 0 0 0 0 queues 1@0 1@1 1@2 1@3 base-time $BASE_TIME  sched-entry S 08 300000
sched-entry S 04 300000 sched-entry S 02 100000 sched-entry S 01 300000 flags 0x2
```

7. Send two streams into queue3 and queue2, replace $BASE_TIME in command based on step-5

```
# udp_tai -i eth0 -b $BASE_TIME + 60000050000 -P 1000000 -t 3 -p 90 -d 600000 -u 7788&
# udp_tai -i eth0 -b $BASE_TIME + 60000550000 -P 1000000 -t 2 -p 90 -d 600000 -u 7789&
```

8. Open the pcap file on PC with Wireshark and 0~300us queue3 frame with UDP port-7788 and 300~600us queue2 frame with UDP port-7799 will be got.

4.4 802.1Qbu

1. As 802.1Qbu relies on time synchronization, run steps1-6 in 4.1 to make sure time synchronization is done between platform-1&2.

2. Enable Hardware timestamps for RX packets on platform-1&2

```
# hwstamp_ctl -i eth0 -r 1
```

3. Sync PHC to system clock on platform-1&2

```
# phc2sys -s eth0 -c CLOCK_REALTIME --step_threshold=1 --transportSpecific=1 -w&
# adjust_clock_tai_offset
# check_clocks eth0
```

4. Get basetime + 2minutes by shell script

```
# sh /etc/tsn-scripts/base-time.sh
```

5. Set gate open hold and release on platform-1 & 2, replace $BASE_TIME in command based on step-4

```
# tc qdisc replace dev eth0 parent root handle 100 taprio num_tc 4 map 0 1 2 3 0 0 0 0 0
0 0 0 0 0 0 0 queues 1@0 1@1 1@2 1@3 base-time $BASE_TIME sched-entry SH 08 300000
sched-entry SH 04 300000 sched-entry SH 02 100000 sched-entry SR 01 300000 flags 0x2
```

6. Send stream into queue0 by iperf3

platform-2:

```
# iperf3 -s -i 2 &
```

platform-1:

```
# iperf3 -c 192.168.0.10 -b 10M -i 2 -t 10 &
```

7. Get count of preempted packets on platform-1&2 by checking mmc_tx_fpe_fragment_cntr and mmc_rx_fpe_fragment_cntr

```
# ethtool -S eth0 | grep fpe
```

5 OPC UA

OPC UA is a protocol for industrial communication and has been standardized in the IEC 62541 series. At its core, OPC UA defines

- an asynchronous protocol (built upon TCP, HTTP or SOAP) that defines the exchange of messages via sessions, (on top of) secure communication channels, (on top of) raw connections,
- a type system for protocol messages with a binary and XML-based encoding scheme,
- a meta-model for information modeling, that combines object-orientation with semantic triple-relations, and
- a set of 37 standard services to interact with server-side information models. The signature of each service is defined as a request and response message in the protocol type system.

5.1 open62541

open62541 is an open source and free implementation of OPC UA written in the common subset of the C99 and C++98 languages.
We enable open62541 support by add "libopen62541" to bb file:
meta/recipes-rity/packagegroups/packagegroup-rity-tsn.bb
which build open62541 as a C-based dynamic library (libopen62541.so).

5.2 OPC UA Pub/Sub over TSN

The new part 14 of the OPC UA specification defines an extension of OPC UA based on the Publish / Subscribe communication paradigm. This opens new usage scenarios, including many-to-many communication. In addition, the integration of OPC UA PubSub with Time-Sensitive Networking (TSN) is designated to additionally enable real time communication.

5.3 OPC UA PubSub Sample Application

The OPC/UA PubSub TSN sample application is token from NXP Real-time Edge Software. One acts as Publisher and the other acts as Subscriber.
Briefly speaking:

- Publisher and Subscriber create a PubSubConnection, whose network address URL is opc.eth://01-00-5E-00-00-01, and publisher id is 2234.
- Publisher conveys the CPU temperature/tx hw timestamp/… in the published packet through the PubSubConnection, and subscriber records the rx hw timestamp when the published packet arrives, also extracts CPU temperature /tx hw timestamp/… information from the published packet.
- Both the Publisher and the Subscriber run a OPC UA server, then User can use a OPC UA client running on a host PC to browse the server's Address Space on either the Publisher or the Subscriber.

5.3.1 Test Network topology

A simple back-to-back setup is made as show in the following diagram. One Genio 1200-demo board (platform 1) acts as Publisher and the other (platform 2) acts as Subscriber, and they are connected via eth0 which is TSN capable. Also, the eth1 interface, which is USB Ethernet, on both boards is connected to a router, then we can use OPC UA Client from PC to observe the Data on both Publisher and Subscriber.

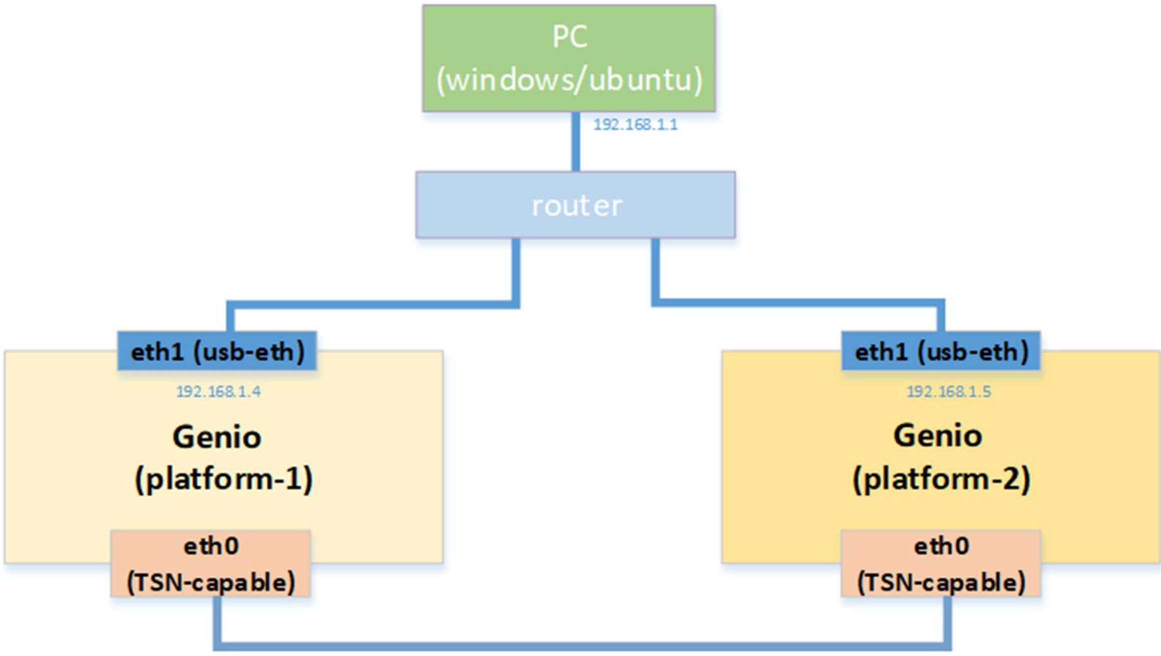


Figure 5-1. Test Network topology

5.3.2 Test Steps

- On both boards, bring up eth0 and disable the EEE features to avoid side effect to PTP (EEE will affect the path delay).

```
// if eth0 is already up, skip
# ip link set eth0 up
# ethtool eth0

// disable eee
# ethtool --set-eee eth0 eee off advertise 0
```

- On the Subscriber (platform 2), set MAC address.

```
# ifconfig eth0 down

# ifconfig eth0 hw ether 00:11:22:33:44:55 up
```

- On the Publisher (platform 1), add a tc filter rule to match OPC UA PubSub packet (EtherType 0xB62C) on eth0 and modify SKB priority to 2.

```
# tc qdisc add dev eth0 clsact

// ether type = opc ua assign priority 2
# tc filter add dev eth0 egress prio 1 u32 match u16 0xb62c 0xffff at -2 action skbedit
priority 2

// show
# tc filter show dev eth0 egress
```

- Run ptp4l for PTP time synchronization and run phc2sys to synchronize PHC clock to Linux system clock (Clock_REALTIME) on Publisher (platform 1), Publisher acts as master.

```
# ptp4l -i eth0 -f /etc/ptp4l_cfg/automotive-master.cfg -m > /var/log/ptp4l.log 2>&1 &
# phc2sys -s eth0 -c CLOCK_REALTIME --step_threshold=1 --transportSpecific=1 -w -m >
/var/log/phc2sys.log 2>&1 &
```

- Run ptp4l for PTP time synchronization and run phc2sys to synchronize PHC clock to Linux system clock (Clock_REALTIME) on Subscriber (platform 2), Subscriber acts as slave.

```
# ptp4l -i eth0 -f /etc/ptp4l_cfg/automotive-slave.cfg -m > /var/log/ptp4l.log 2>&1 &
# phc2sys -s eth0 -c CLOCK_REALTIME --step_threshold=1 --transportSpecific=1 -w -m >
/var/log/phc2sys.log 2>&1 &
```

- On both boards, we can observe the logs of ptp4l and phc2sys to check the time synchronization progress by below commands

```
# tail -f /var/log/ptp4l.log
# tail -f /var/log/phc2sys.log
```

- On the slave side (Subscriber, platform 2), the rms value reported by ptp4l shows the root mean square of the time offset between the PHC and the GM clock. If ptp4l consistently reports rms lower than 100 ns, the PHC is synchronized. Example ptp4l log below:

```
root@i1200-demo:~# tail -f /var/log/ptp4l.log
ptp4l[1033.279]: selected /dev/ptp0 as PTP clock
ptp4l[1033.324]: port 1: INITIALIZING to SLAVE on INIT_COMPLETE
ptp4l[1033.324]: port 0: INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[1035.225]: rms  301 max  433 freq   -134 +/- 230 delay   786 +/-   0
ptp4l[1036.226]: rms   84 max  128 freq   +101 +/-  88 delay   791 +/-   0
ptp4l[1037.227]: rms  129 max  152 freq   +305 +/-  25 delay   789 +/-   0
ptp4l[1038.227]: rms   70 max  109 freq   +325 +/-  11 delay   793 +/-   0
ptp4l[1039.228]: rms   23 max   36 freq   +297 +/-  18 delay   797 +/-   0
ptp4l[1039.479]: master offset         21 s3 freq    +299 path delay       797
ptp4l[1040.479]: master offset        -34 s3 freq    +250 path delay       797
ptp4l[1041.479]: master offset        -42 s3 freq    +232 path delay       797
ptp4l[1042.479]: master offset         29 s3 freq    +291 path delay       797
ptp4l[1043.479]: master offset         12 s3 freq    +282 path delay       797
ptp4l[1044.479]: master offset         -7 s3 freq    +267 path delay       793
ptp4l[1045.479]: master offset        -14 s3 freq    +258 path delay       793
```

- On both boards, the offset information reported by phc2sys shows the time offset between the PHC
- and the system clock (CLOCK_REALTIME). If phc2sys consistently reports offset lower than 100 ns, the System clock is synchronized. Example phc2sys log below:

```
root@i1200-demo:~# tail -f /var/log/phc2sys.log
phc2sys[1073.532]: CLOCK_REALTIME phc offset       4 s2 freq     +10 delay   1230
phc2sys[1074.532]: CLOCK_REALTIME phc offset      29 s2 freq     +36 delay   1231
phc2sys[1075.533]: CLOCK_REALTIME phc offset     -24 s2 freq      -8 delay   1231
phc2sys[1076.533]: CLOCK_REALTIME phc offset      23 s2 freq     +32 delay   1230
phc2sys[1077.533]: CLOCK_REALTIME phc offset     -36 s2 freq     -20 delay   1230
phc2sys[1078.534]: CLOCK_REALTIME phc offset       5 s2 freq     +10 delay   1230
phc2sys[1079.534]: CLOCK_REALTIME phc offset      51 s2 freq     +57 delay   1230
phc2sys[1080.534]: CLOCK_REALTIME phc offset     -29 s2 freq      -7 delay   1230
phc2sys[1081.534]: CLOCK_REALTIME phc offset       0 s2 freq     +13 delay   1230
phc2sys[1082.535]: CLOCK_REALTIME phc offset      24 s2 freq     +37 delay   1231
phc2sys[1083.535]: CLOCK_REALTIME phc offset     -45 s2 freq     -25 delay   1231
phc2sys[1084.536]: CLOCK_REALTIME phc offset      40 s2 freq     +47 delay   1231
phc2sys[1085.536]: CLOCK_REALTIME phc offset       9 s2 freq     +28 delay   1231
phc2sys[1086.536]: CLOCK_REALTIME phc offset      -3 s2 freq     +18 delay   1231
phc2sys[1087.537]: CLOCK_REALTIME phc offset     -61 s2 freq     -40 delay   1230
phc2sys[1088.537]: CLOCK_REALTIME phc offset     -16 s2 freq     -14 delay   1231
phc2sys[1089.537]: CLOCK_REALTIME phc offset      27 s2 freq     +24 delay   1230
phc2sys[1090.538]: CLOCK_REALTIME phc offset      21 s2 freq     +27 delay   1230
phc2sys[1091.538]: CLOCK_REALTIME phc offset      30 s2 freq     +42 delay   1230
phc2sys[1092.538]: CLOCK_REALTIME phc offset     -33 s2 freq     -12 delay   1231
phc2sys[1093.539]: CLOCK_REALTIME phc offset      -5 s2 freq      +6 delay   1230
phc2sys[1094.539]: CLOCK_REALTIME phc offset     -15 s2 freq      -6 delay   1230
phc2sys[1095.539]: CLOCK_REALTIME phc offset      29 s2 freq     +34 delay   1231
phc2sys[1096.540]: CLOCK_REALTIME phc offset       0 s2 freq     +14 delay   1230
```

After establishing the time synchronization successfully on both the Publisher and the Subscriber, we can configure TSN Qbv and run the OPC UA PubSub sample applications as in the following steps.

- On the Publisher (platform 1), configure TSN Qbv on eth0 to map SKB priority to traffic class to hardware queue as below, set gate control list to have 2 entries and total cycle time of 1ms (queue 3 has 500us for best effort traffic, queue 0 and queue 2 share 500us for OPC UA PubSub and PTP traffic as well as other traffic like ping), also set base time to 1ms so that the schedule is aligned to 1ms. This is just an example configuration for the schedule.
    - SKB priority 0 -> traffic class 0 -> queue 0
    - SKB priority 1 -> traffic class 1 -> queue 1
    - SKB priority 2 -> traffic class 2 -> queue 2
    - SKB priority 3 -> traffic class 3 -> queue 3

```
# tc qdisc replace dev eth0 parent root handle 100 taprio num_tc 4 map 0 1 2 3 queues
1@0 1@1 1@2 1@3 base-time 001000000 sched-entry S 0x8 500000 sched-entry S 0x05 500000
flags 2
```

Together with the "*tc filter rule*" configured previously, the above TSN Qbv configuration on eth0 will distribute:

OPC UA PubSub traffic into Tx hardware queue 2

PTP traffic into Tx hardware queue 0

Best effort traffic to Tx hardware queue 3 (generate by pktgen later)

Other traffic to Tx hardware queue 0

Since the OPC UA PubSub/PTP traffic don't share the same Tx hardware queues and time slot with the best effort traffic, the latter will not influence the former.

- On the Subscriber (platform 2), run "hwstamp_ctl -i eth0 -r 1" to 'timestamp any incoming packet', so can get the RX hardware timestamp of the published packets once arrive in Subscriber.
- Then run the OPC UA PubSub Subscriber sample application. Run the Subscriber application before the Publisher application so that we won't miss any packet sent by the Publisher.

```
# hwstamp_ctl -i eth0 -r 1

# /home/root/open62541_example/opcua_pubsub_subscriber -u opc.eth://01-00-5E-00-00-01 -d
eth0
```

Logs on Subscriber:

```
root@i1200-demo:~# /home/root/open62541_example/opcua_pubsub_subscriber -u opc.eth://01-00
-5E-00-00-01 -d eth0&
[3] 599
root@i1200-demo:~# [2022-04-28 20:11:58.376 (UTC+0000)] info/userland   Transport Profile
  : http://opcfoundation.org/UA-Profile/Transport/pubsub-eth-uadp
[2022-04-28 20:11:58.376 (UTC+0000)] info/userland      Network Address URL : opc.eth://01
-00-5E-00-00-01
[2022-04-28 20:11:58.376 (UTC+0000)] info/userland      Ethernet Interface  : eth0
[2022-04-28 20:11:58.469 (UTC+0000)] warn/server        AccessControl: Unconfigured Access
Control. Users have all permissions.
[2022-04-28 20:11:58.469 (UTC+0000)] info/server        AccessControl: Anonymous login is
enabled
[2022-04-28 20:11:58.469 (UTC+0000)] warn/server        Username/Password configured, but
no encrypting SecurityPolicy. This can leak credentials on the network.
[2022-04-28 20:11:58.469 (UTC+0000)] warn/userland      AcceptAll Certificate Verification
. Any remote certificate will be accepted.
[2022-04-28 20:11:58.469 (UTC+0000)] info/userland      PubSub channel requested
[2022-04-28 20:11:58.469 (UTC+0000)] info/server        Open PubSub ethernet connection.
[2022-04-28 20:11:58.469 (UTC+0000)] info/userland      Subscriber socket FD : 3
[2022-04-28 20:11:58.469 (UTC+0000)] info/userland      Subscriber Rx HW timestamp : Enabl
ed
[2022-04-28 20:11:58.469 (UTC+0000)] info/userland      Subscriber thread priority : 81
[2022-04-28 20:11:58.470 (UTC+0000)] info/userland      Publisher on CPU core : 7
[2022-04-28 20:11:58.470 (UTC+0000)] info/userland      Subscriber thread callback Id: 281
473005334816
[2022-04-28 20:11:58.470 (UTC+0000)] info/userland      Starting the subscriber cycle at 1
651176719.000500000
[2022-04-28 20:11:58.471 (UTC+0000)] info/network       TCP network layer listening on opc
.tcp://i1200-demo:4801/
[2022-04-28 20:11:59.000 (UTC+0000)] info/userland      Packet Sequence Number : 0
[2022-04-28 20:12:00.001 (UTC+0000)] info/userland      Packet Sequence Number : 0
[2022-04-28 20:12:01.001 (UTC+0000)] info/userland      Packet Sequence Number : 0
[2022-04-28 20:12:02.001 (UTC+0000)] info/userland      Packet Sequence Number : 0
[2022-04-28 20:12:03.000 (UTC+0000)] info/userland      Packet Sequence Number : 0
[2022-04-28 20:12:04.000 (UTC+0000)] info/userland      Packet Sequence Number : 1
[2022-04-28 20:12:05.000 (UTC+0000)] info/userland      Packet Sequence Number : 2
[2022-04-28 20:12:06.000 (UTC+0000)] info/userland      Packet Sequence Number : 3
[2022-04-28 20:12:07.000 (UTC+0000)] info/userland      Packet Sequence Number : 4
[2022-04-28 20:12:08.000 (UTC+0000)] info/userland      Packet Sequence Number : 5
```

- On the Publisher (platform 1), run the OPC UA PubSub Publisher sample application.

```
# /home/root/open62541_example/opcua_pubsub_publisher -u opc.eth://01-00-5E-00-00-01 -d
eth0
```

Logs on Publisher:

```
root@i1200-demo:~# /home/root/open62541_example/opcua_pubsub_publisher -u opc.eth://01-00-
5E-00-00-01 -d eth0&
[4] 569
[3]   Done                    /home/root/open62541_example/opcua_pubsub_publisher -u opc.e
th://01-00-5E-00-00-01 -d eth0
[2022-04-28 20:11:58.703 (UTC+0000)] info/userland      Transport Profile   : http://opcfo
undation.org/UA-Profile/Transport/pubsub-eth-uadp
[2022-04-28 20:11:58.703 (UTC+0000)] info/userland      Network Address URL : opc.eth://01
-00-5E-00-00-01
[2022-04-28 20:11:58.703 (UTC+0000)] info/userland      Ethernet Interface  : eth0
root@i1200-demo:~# [2022-04-28 20:11:58.795 (UTC+0000)] warn/server     AccessControl: Unc
onfigured AccessControl. Users have all permissions.
[2022-04-28 20:11:58.795 (UTC+0000)] info/server        AccessControl: Anonymous login is
enabled
[2022-04-28 20:11:58.795 (UTC+0000)] warn/server        Username/Password configured, but
no encrypting SecurityPolicy. This can leak credentials on the network.
[2022-04-28 20:11:58.795 (UTC+0000)] warn/userland      AcceptAll Certificate Verification
. Any remote certificate will be accepted.
[2022-04-28 20:11:58.795 (UTC+0000)] info/userland      PubSub channel requested
[2022-04-28 20:11:58.795 (UTC+0000)] info/server        Open PubSub ethernet connection.
[2022-04-28 20:11:58.795 (UTC+0000)] info/userland      Publisher socket FD : 3
[2022-04-28 20:11:58.795 (UTC+0000)] info/userland      Publisher Tx HW timestamp : Enable
d
[2022-04-28 20:11:58.795 (UTC+0000)] info/userland      Publisher cycle time : 1000.000000
 ms
[2022-04-28 20:11:58.795 (UTC+0000)] info/userland      Publisher thread priority : 78
[2022-04-28 20:11:58.795 (UTC+0000)] info/userland      Publisher on CPU core : 7
[2022-04-28 20:11:58.795 (UTC+0000)] info/userland      Publisher thread callback Id: 2814
73625633056
[2022-04-28 20:11:58.796 (UTC+0000)] info/network       TCP network layer listening on opc
.tcp://i1200-demo:4840/
[2022-04-28 20:11:58.796 (UTC+0000)] info/userland      Starting the publisher cycle at 16
51176723.000000000
```

- On a PC connected to the router and with OPC UA Client installed (e.g. UaExpert), we can browser either the OPC UA server's Address Space on either the Publisher or the Subscriber. (We assume that eth1 has obtained the IP address by DHCP automatically).
- The URL of the OPC UA server on the Publisher is below:
    - opc.tcp://<IP_of_eth1_on_Publisher>:4840/
- The URL of the OPC UA server on the Subscriber is below:
    - opc.tcp://<IP_of_eth1_on_Subscriber>:4801/

Example snapshot of UaExpert connected to the Publisher:

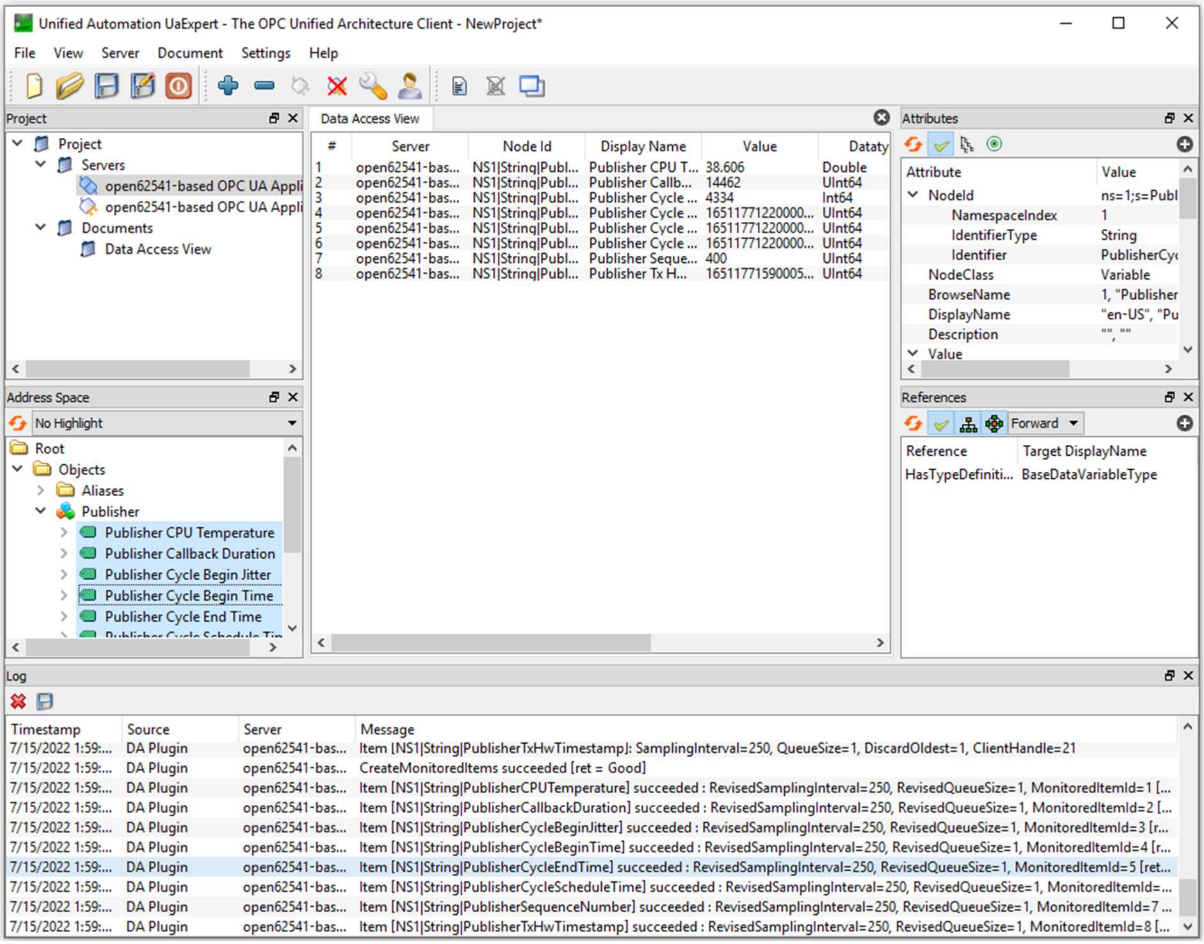


Figure 5-2. UaExpert connected to the Publisher

Example snapshot of UaExpert connected to the Subscriber:

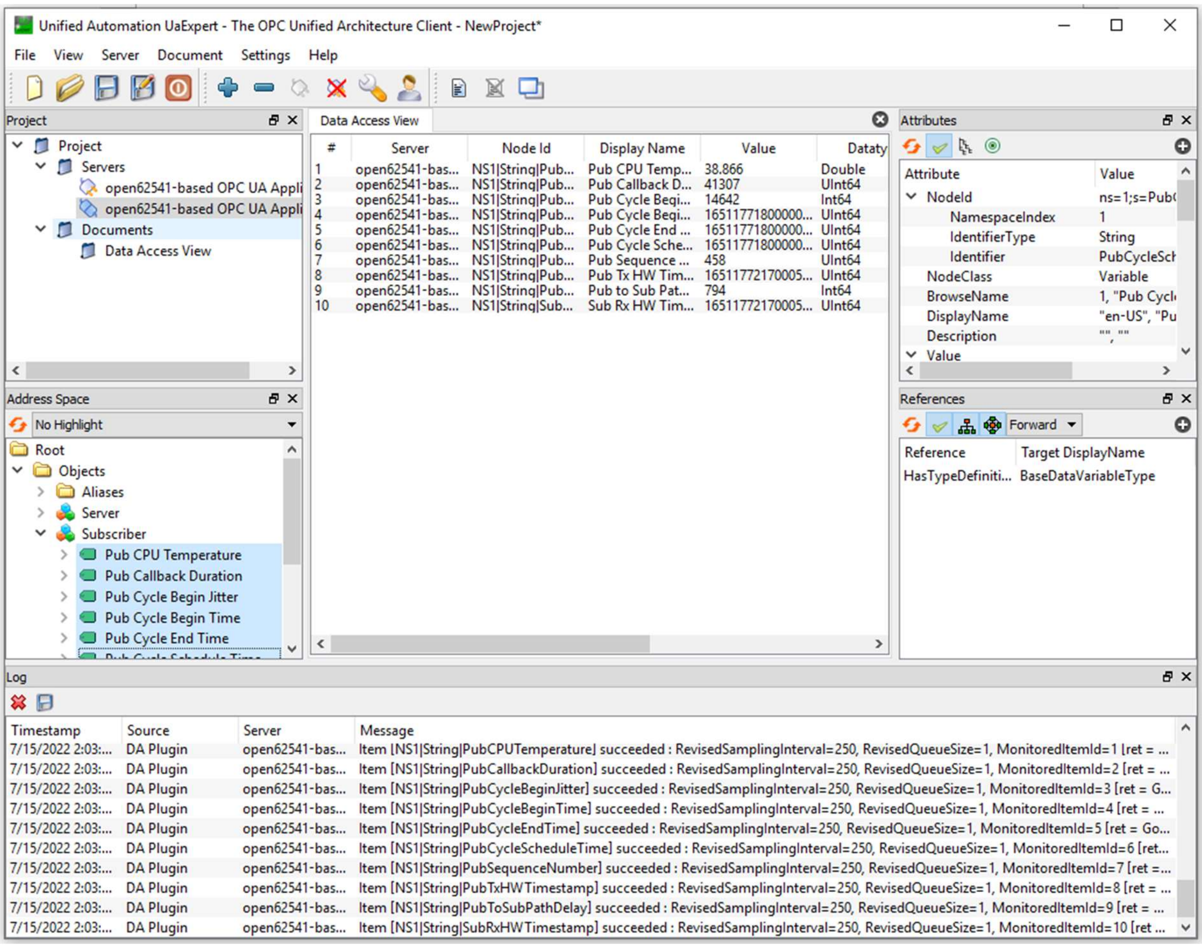


Figure 5-3. UaExpert connected to the Subscriber

On the UaExpert client connected to the Subscriber, we can observe the CPU temperature published by the Publisher and the path delay from Publisher to Subscriber which is close to 800ns.

5.3.3 Taprio effects on PTP/Best Effort Traffic

As previous mentioned, PTP traffic is assigned to Hardware queue 0.

- On the Publisher (platform 1), we can use pktgen to simulate high load best effort traffic, which is sent to queue 3 of eth0, and remove the TAPRIO qdisc by default.

```
# /home/root/pktgen/pktgen_sample01_simple.sh -i eth0 -q 3 -s 1000 -n 0
# tc qdisc del dev eth0 root
```

Observe the PTP synchronization on Publisher:

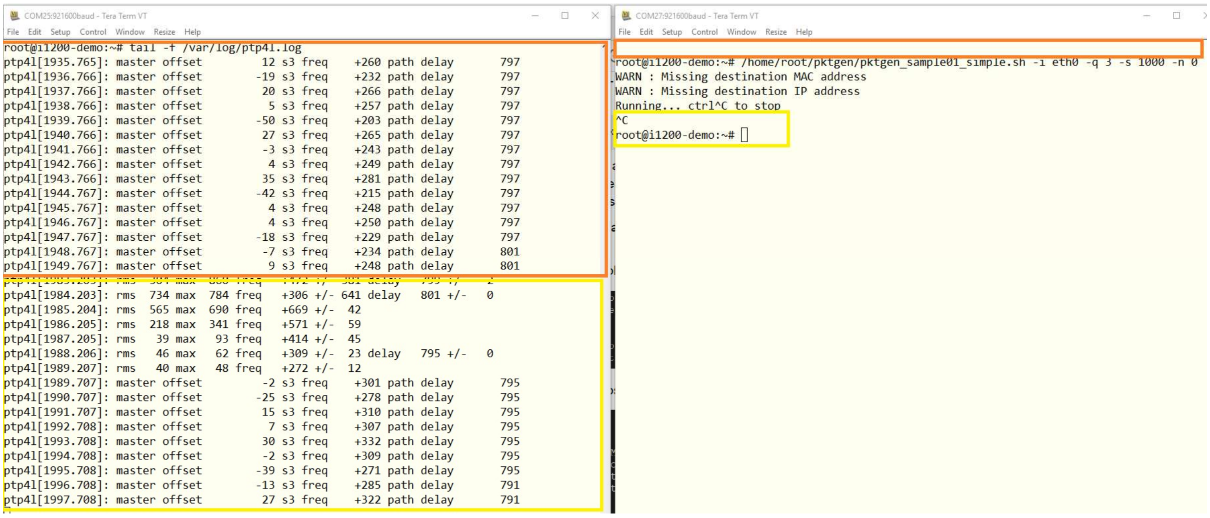
The sync is lost when pktgen is running, restored after pktgen is killed.

The left Tera Term --> *Subscriber*

The right Tera Term --> *Publisher*

Red block --> *before pktgen script runs*

Yellow blocks --> *after pktgen script quilts*



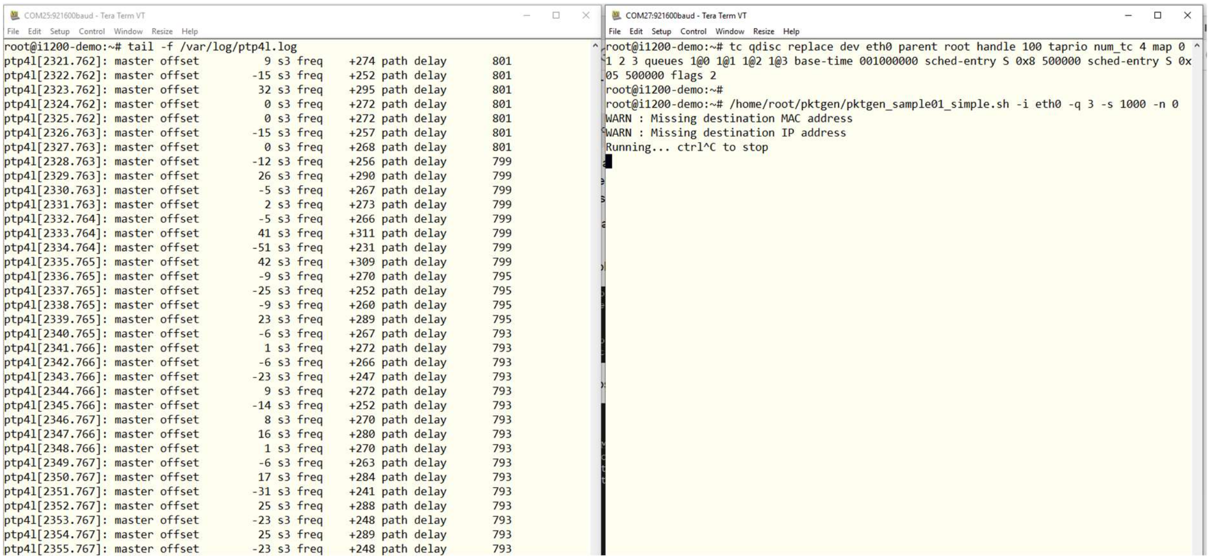
When TAPRIO is not configured, the PTP synchronization is impacted by high load best effort traffic.

- On Publisher, attach the TAPRIO qdisc to eth0, then run pktgen script to simulate high load best effort traffic.

```
# tc qdisc replace dev eth0 parent root handle 100 taprio num_tc 4 map 0 1 2 3 queues
1@0 1@1 1@2 1@3 base-time 001000000 sched-entry S 0x8 500000 sched-entry S 0x05 500000
flags 2

# /home/root/pktgen/pktgen_sample01_simple.sh -i eth0 -q 3 -s 1000 -n 0
```

When TAPRIO is enabled, PTP runs normally, even the pktgen script is running, which demonstrates the power of TAPRIO.



5.3.4 More Experiment

We can also try:

OPC UA traffic → priority 2 → queue2

PTP traffic → priority 1 → queue 1

Best effort traffic → priority 0 → queue 0

```
// OPC UA traffic --> priority 2
# tc filter add dev eth0 egress prio 1 u32 match u16 0xb62c 0xffff at -2 action skbedit
priority 2

// OPC UA traffic --> priority 1
# tc filter add dev eth0 egress prio 1 u32 match u16 0x88f7 0xffff at -2 action skbedit
priority 1

# tc qdisc replace dev eth0 parent root handle 100 taprio num_tc 4 map 0 1 2 3 queues
1@0 1@1 1@2 1@3 base-time 001000000 sched-entry S 0x1 400000 sched-entry S 0x4 300000
sched-entry S 0x2 300000 flags 2

// high load best effort traffic generated by pktgen
# /home/root/pktgen/pktgen_sample01_simple.sh -i eth0 -q 0 -s 1000 -n 0
```

5.4 UaExpert Installation

Below steps shows how to use UaExpert to connect to an OPC UA server on a Window10 PC.

1. Open the UaExpert GUI. Click on the 'Add Server' button.

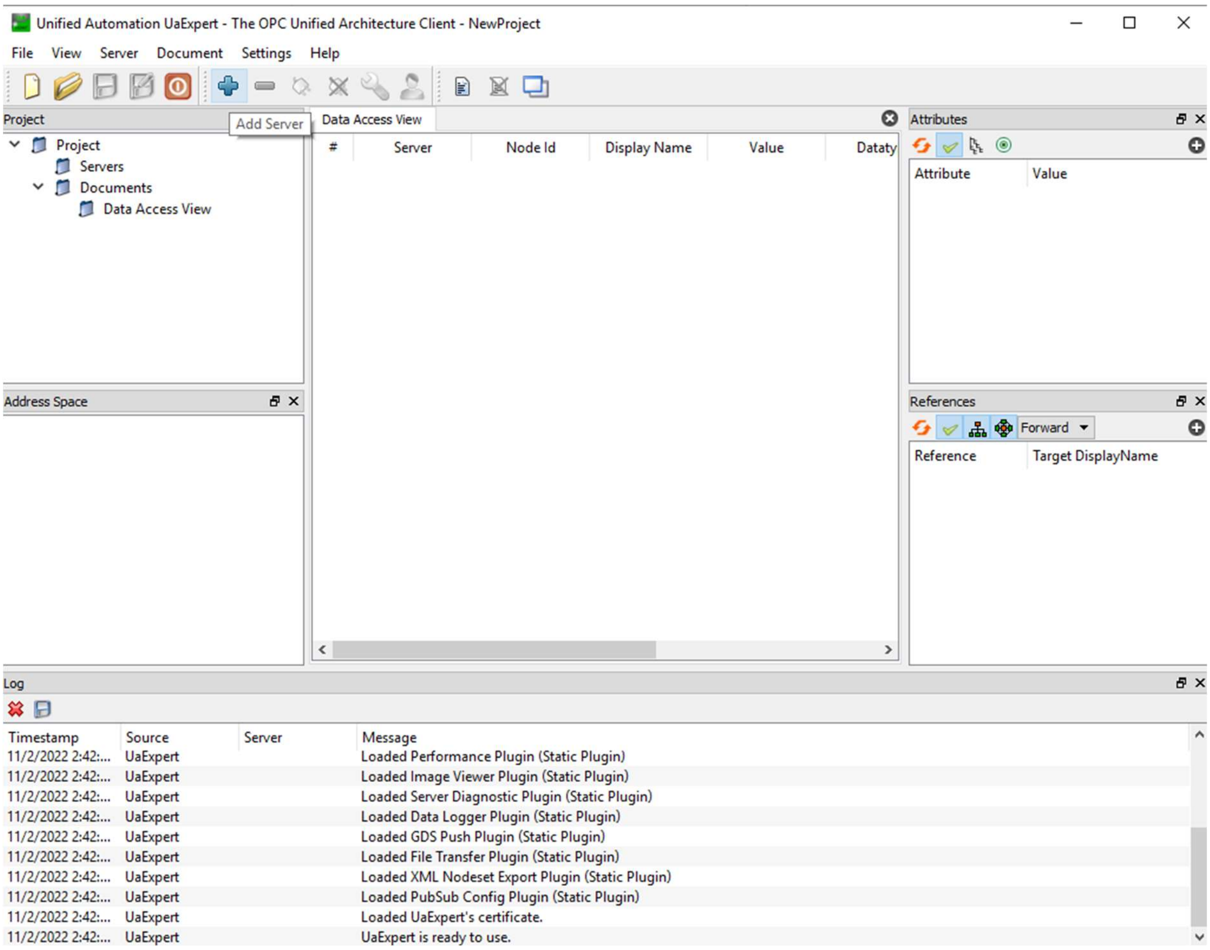


Figure 5-4. Add Server

2. The 'Add Server' window will pop up. Select Custom Discovery and double click '< Double click to Add Server... >'. The 'Enter URL' window will pop up. Input IP address and port number of the OPC UA server separated by colon. For example, the complete URL is ***opc.tcp://192.168.1.3:4840*** in below snapshot. Click OK.

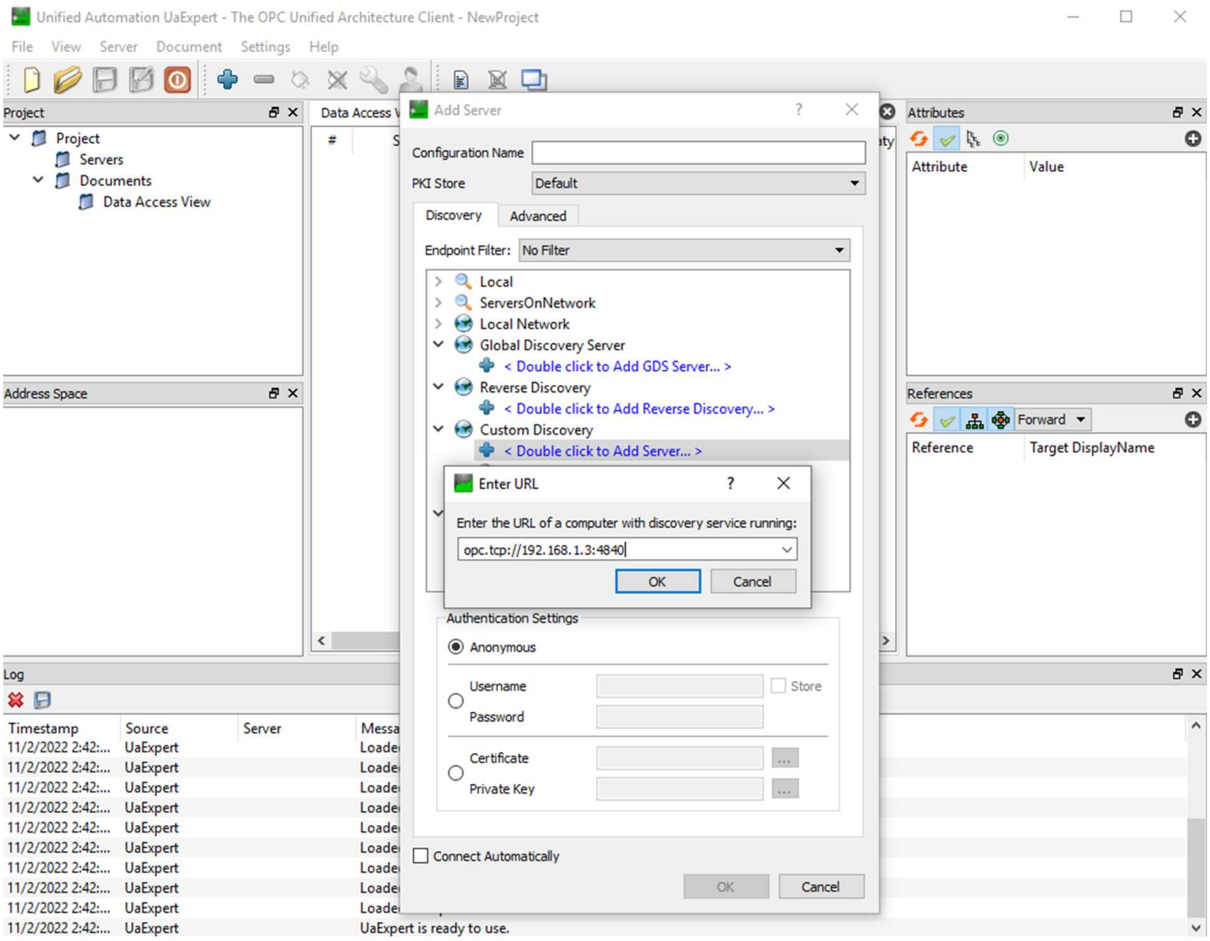


Figure 5-5. Enter URL

3. The new server (i.e., ***opc.tcp://192.168.1.3:4840***) will be listed under Custom Discovery. Click to expand it. Then click to expand 'open62541-based OPC UA Application (opc.tcp)'. A 'Replaced Hostname' window will pop up. Click 'Yes'.

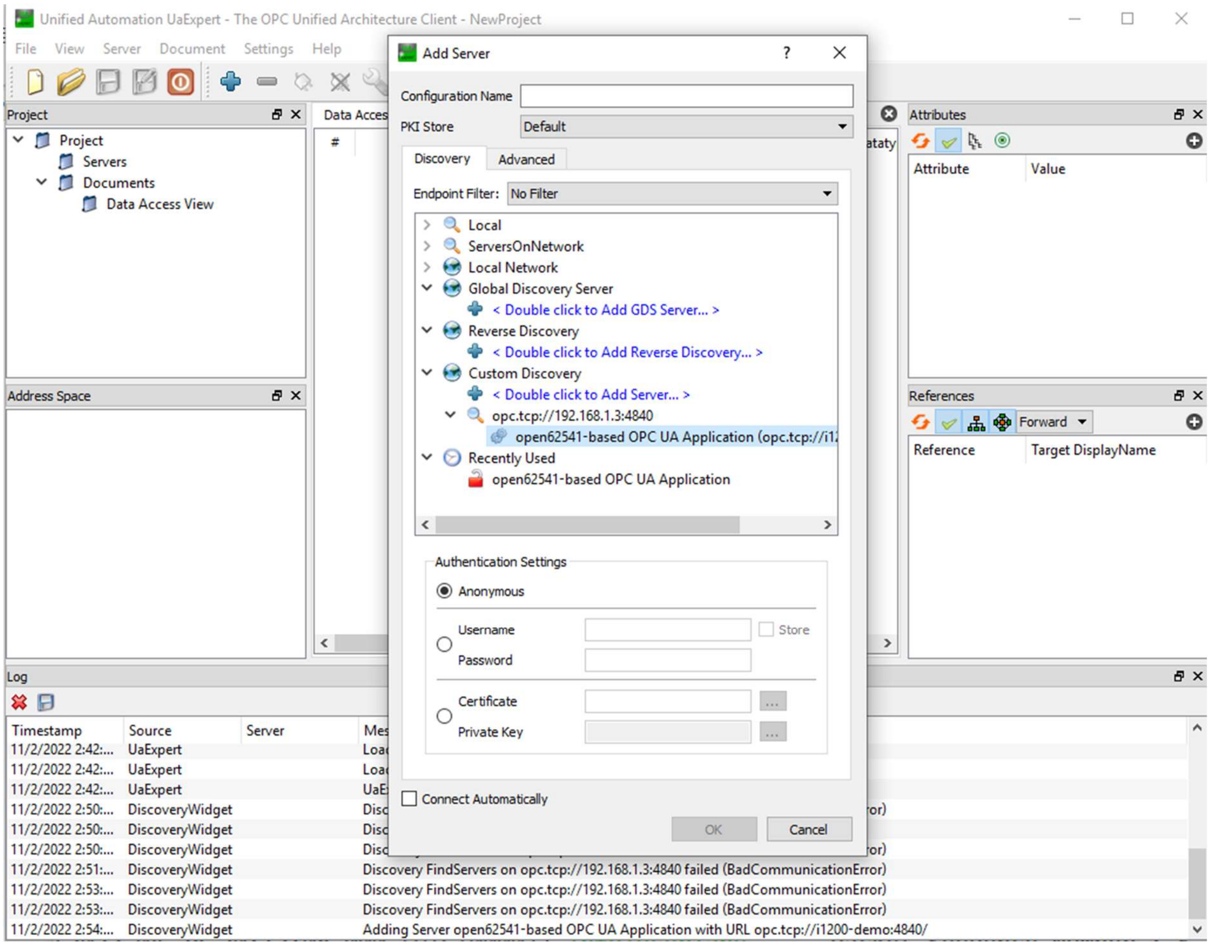


Figure 5-6. New server list

4. Click to select 'None – None (…)' and click OK.

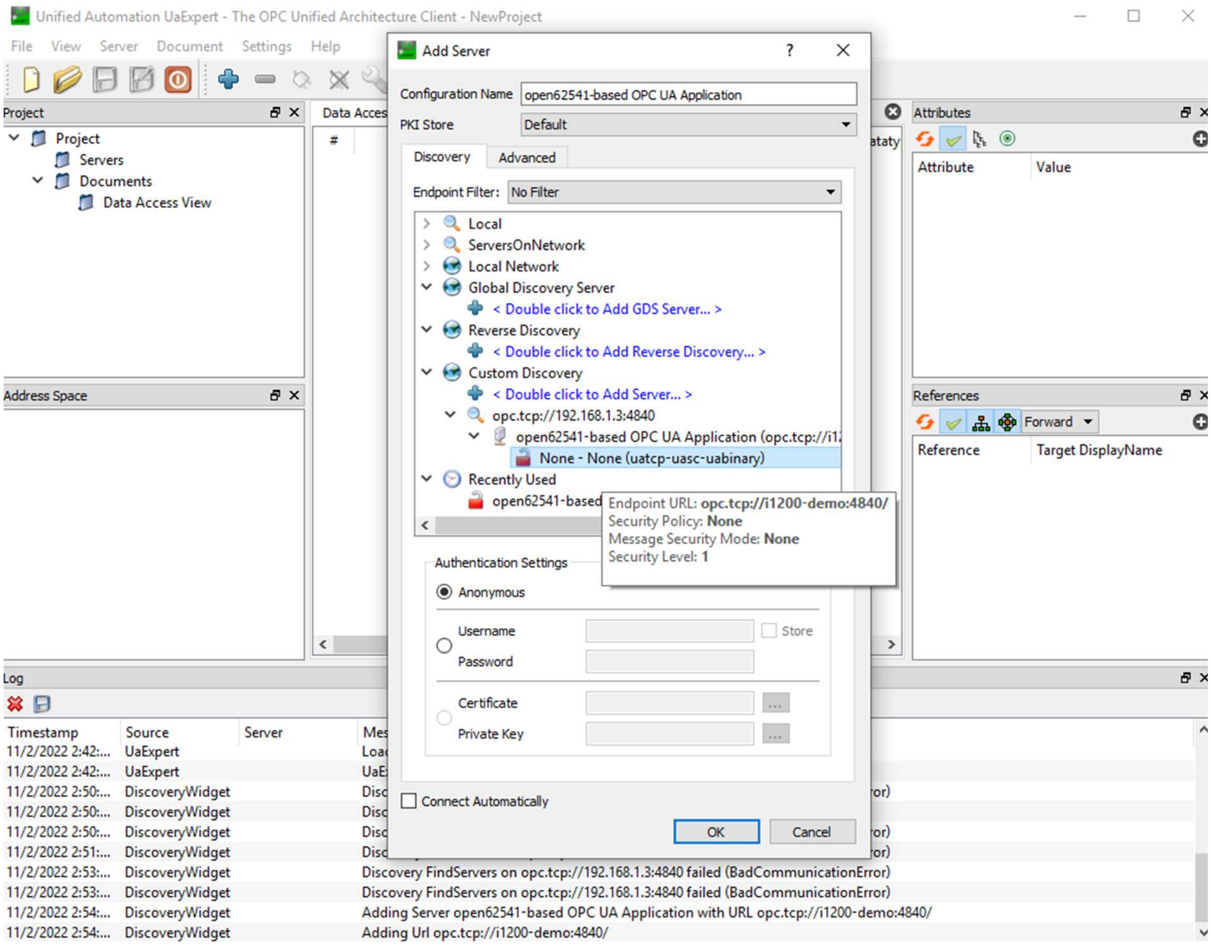


Figure 5-7. Select None – None option

5. Right click on the server listed under ‘Servers’ and click ‘Connect’.

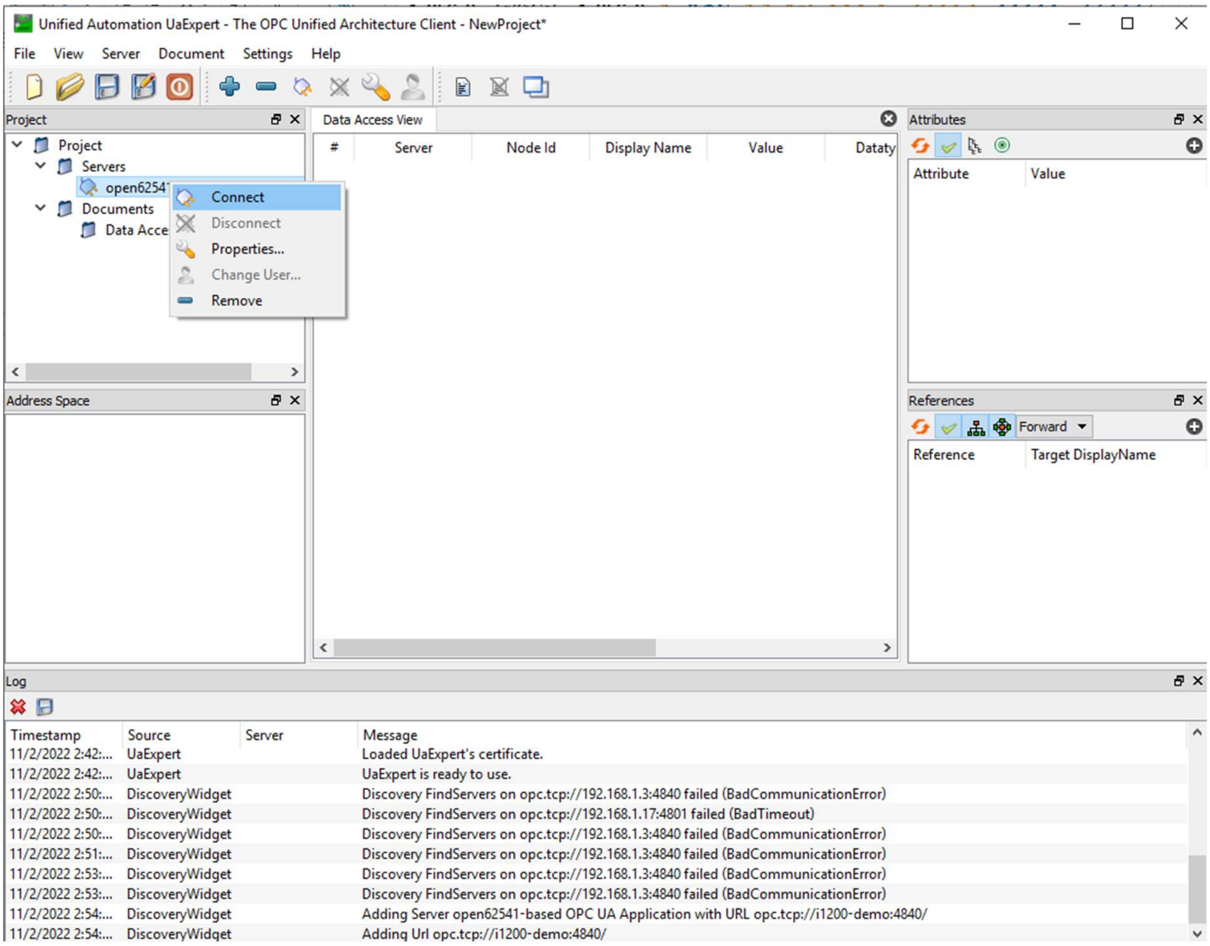


Figure 5-8. Connect servers

6. You are now connected to the OPC UA server and can browse or monitor its object. To monitor the value of an object, you can drag and drop the object to the 'Data Access View' area.

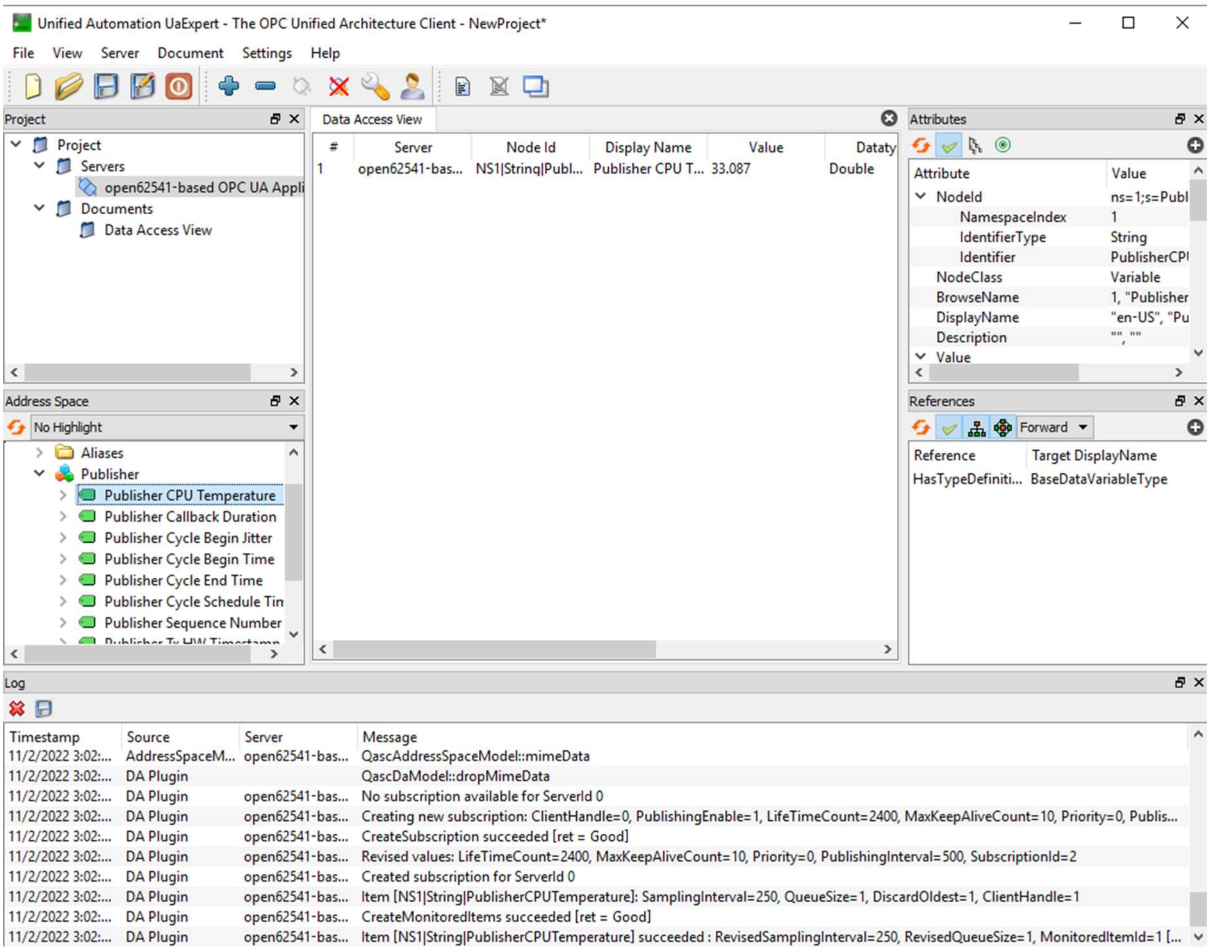


Figure 5-9. Monitor server object

6 NETCONF/YANG

NETCONF provides mechanisms to install, manipulate, and delete the configuration of network devices. Its operations are realized on top of a simple Remote Procedure Call (RPC) layer. The NETCONF protocol uses XML based data encoding for the configuration data as well as the protocol messages.
YANG is a standards-based, extensible, hierarchical data modeling language that is used to model the configuration and state data used by NETCONF operations, remote procedure calls (RPCs), and server event notifications.
A YANG module defines a data model through its data, and the hierarchical organization of and constraints on that data. A module can be a complete, standalone entity, or it can reference definitions in other modules and sub-modules as well as augment other data models with additional nodes. The module dictates how the data is represented in XML.
A YANG module defines not only the syntax but also the semantics of the data. It explicitly defines relationships between and constraints on the data. This enables user to create syntactically correct configuration data that meets constraint requirements and enables user to validate the data against the model before uploading it and committing it on a device [7].

6.1 Tools

The tools using to demonstrate remote configuration follow that in NXP Real-time Edge Software.
Following diagram shows the high-level architecture of Netopeer2 and sysrepo.

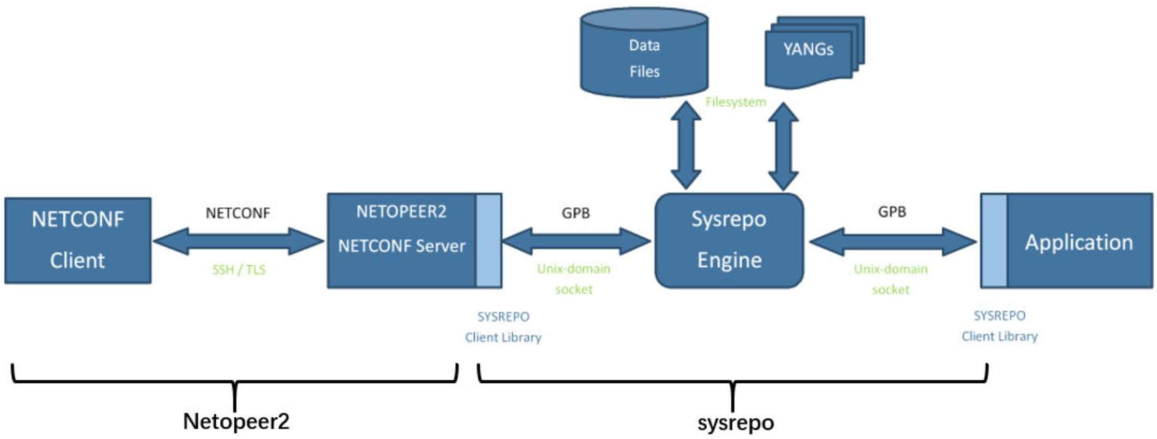


Figure 6-1. Architecture of Netopeer2 and sysrepo

6.1.1 Sysrepo

Sysrepo is a YANG-based datastore for Unix/Linux systems. Applications that have their configuration modelled using YANG can use Sysrepo for its management[8].

6.1.2 Netopeer2

Netopeer2 is a server for implementing network configuration management based on the NETCONF Protocol. This is the second generation, originally available as the Netopeer project. Netopeer2 is based on the new generation of the NETCONF and YANG libraries - libyang and libnetconf2. The Netopeer2 server uses sysrepo as a NETCONF datastore implementation[9].

6.1.3 Tools in meta-mediatek-tsn

The remote configuration tools using NETCONF/YANG in meta-mediatek-tsn layer mainly refers to meta-real-time-edge. Some modifications are made to adapt to MediaTek platform.

6.2 Remote Configuration

6.2.1 Test topology

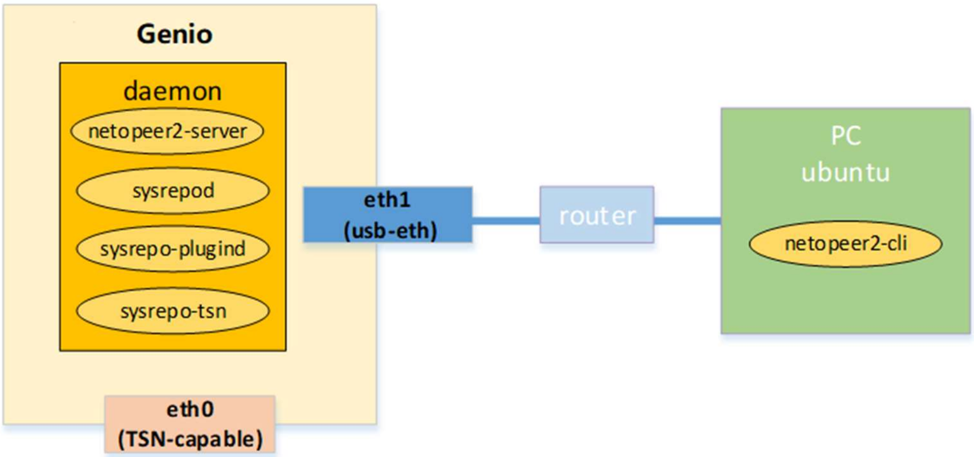


Figure 6-2. Test topology

The package in meta-mediatek-tsn will create a service which will start daemons (netopeer2-server/sysrepod/sysrepo-plugind/sysrepo-tsn) and install required yang models. Then we can configure TSN qdisc from netopeer2 client(netopeer2-cli) by using proper .xml files.

Note:

You can also run netopeer2-cli on Genio board with “connect localhost” for concept demonstration.

6.2.2 Test Steps

- On Genio board, ensure the required daemons already run. These daemons are started by sysrepo-cfg.service when boot up.

```
# ps aux | grep sysrepo
# ps aux | grep neto
```

```
i1200-demo login: root
root@i1200-demo:~# ps aux | grep sysrepo
root         363  1.3  0.5 302728 11120 ?        Ssl  18:16   0:00 /usr/bin/sysrepod
root         372  0.0  0.0   6296   268 ?        Ss   18:16   0:00 /usr/bin/sysrepo-plugind
root         378  0.0  0.1 153900  2308 ?        Ssl  18:16   0:00 /usr/bin/sysrepo-tsn
root         442  0.0  0.0   2948  1312 ttyS0    S+   18:16   0:00 grep sysrepo
root@i1200-demo:~# ps aux | grep neto
root         383  0.9  0.3 387132  6940 ?        Ssl  18:16   0:07 /usr/bin/netopeer2-server
root         453  0.0  0.0   2948  1312 ttyS0    S+   18:30   0:00 grep neto
```

- At the same time, check corresponding YANG modules are installed. These modules are installed by sysrepo-cfg.service when boot up.

```
# sysrepoctl -l
```

```
root@i1200-demo:~# sysrepoctl -l
Sysrepo schema directory: /etc/sysrepo/yang/
Sysrepo data directory:   /etc/sysrepo/data/
(Do not alter contents of these directories manually)

Module Name                                 | Revision   | Conformance | Data Owner          | Permissions | Submodules                     | Enabled Features
---------------------------------------------------------------------------------------------------------------------------------------------
ietf-netconf-notifications                  | 2012-02-06 | Installed   | root:root           | 666         |                                |
ietf-netconf                                | 2011-06-01 | Installed   | root:root           | 666         |                                | writable-running candidate rollback-on-error vali
date startup xpath url
ietf-netconf-acm                            | 2018-02-14 | Installed   | root:root           | 666         |                                |
sysrepo-module-dependencies                 | 2017-04-21 | Installed   | root:root           | 666         |                                |
sysrepo-notification-store                  | 2016-11-22 | Installed   | root:root           | 666         |                                |
sysrepo-persistent-data                     | 2016-03-30 | Installed   | root:root           | 666         |                                |
nc-notifications                            | 2008-07-14 | Installed   | root:root           | 666         |                                |
notifications                               | 2008-07-14 | Installed   | root:root           | 666         |                                |
ietf-x509-cert-to-name                      | 2014-12-10 | Installed   |                     |             |                                |
ietf-keystore                               | 2016-10-31 | Installed   | root:root           | 666         |                                |
ietf-netconf-with-defaults                  | 2011-06-01 | Installed   |                     |             |                                |
ietf-netconf-monitoring                     | 2010-10-04 | Installed   | root:root           | 666         |                                |
ietf-yang-library                           | 2018-01-17 | Installed   | root:root           | 666         |                                |
ietf-datastores                             | 2017-08-17 | Imported    |                     |             |                                |
ietf-netconf-server                         | 2016-11-02 | Installed   | root:root           | 666         |                                | listen ssh-listen tls-listen call-home ssh-call-h
ome tls-call-home
ietf-ssh-server                             | 2016-11-02 | Imported    |                     |             |                                |
ietf-tls-server                             | 2016-11-02 | Imported    |                     |             |                                |
ietf-system                                 | 2014-08-06 | Installed   | root:root           | 666         |                                | authentication local-users
iana-crypt-hash                             | 2014-08-06 | Imported    |                     |             |                                |
ietf-interfaces                             | 2018-02-20 | Installed   | root:root           | 666         |                                |
ietf-yang-types                             | 2013-07-15 | Installed   |                     |             |                                |
ieee802-dot1q-types                         | 2020-10-23 | Installed   |                     |             |                                |
ieee802-dot1q-preemption                    | 2020-07-07 | Installed   |                     |             |                                | frame-preemption
ieee802-dot1q-bridge                        | 2020-11-24 | Installed   | root:root           | 666         |                                |
ieee802-types                               | 2020-10-23 | Installed   |                     |             |                                |
iana-if-type                                | 2020-01-10 | Installed   |                     |             |                                |
ieee802-dot1q-sched                         | 2020-07-07 | Installed   |                     |             |                                | scheduled-traffic
ieee802-dot1q-stream-filters-gates          | 2020-11-06 | Installed   |                     |             |                                | closed-gate-state
ieee802-dot1q-psfp                          | 2020-07-07 | Installed   |                     |             |                                | psfp
ieee802-dot1cb-stream-identification        | 2021-05-06 | Installed   | root:root           | 666         |                                |
ieee802-dot1cb-stream-identification-types  | 2021-06-14 | Installed   |                     |             |                                |
ieee802-dot1q-qci-augment                   | 2019-05-20 | Installed   |                     |             |                                |
ietf-ip                                     | 2018-02-22 | Installed   |                     |             |                                | ipv4-non-contiguous-netmasks
ieee802-dot1cb-frer                         | 2021-05-06 | Installed   | root:root           | 666         |                                |
ieee802-dot1cb-frer-types                   | 2021-05-06 | Imported    |                     |             |                                |
```

The .xml files for TSN qdisc configuration are installed in path: */etc/sysrepo-tsn/Instances*, please copy them to Ubuntu PC before test starts.

6.2.2.1 IP Configuration

- On Ubuntu, run netopeer2 client, and send IP configuration file (ietf-ip-cfg.xml) to “running” datastore, the sysrepo-tsn daemon will monitor file changes, and change the IPv4 address accordingly.

```
# netopeer2-cli

// 192.168.1.50 is the ip address of eth1
> connect --login root --host 192.168.1.50

// configure IP with ietf-ip-cfg.xml
// change the path of ietf-ip-cfg.xml according to your test environment
> edit-config --target running --config=/etc/sysrepo-tsn/Instances/ietf-ip-cfg.xml

// check the IP configure data
> get-config --source running --filter-xpath /ietf-interfaces:interfaces/interface
[name='eth0']/ietf-ip:ipv4
```

Check the result on Genio board, the IPv4 address is same with that in xml.

```
root@i1200-demo:~# ifconfig
eth0: flags=3<UP,BROADCAST>  mtu 1500
        inet 192.168.78.129  netmask 255.255.0.0  broadcast 192.168.255.255
        ether 00:55:7b:b5:7d:f7  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 16  base 0x8000
```

Snippet in ietf-ip-cfg.xml:

```
root@i1200-demo:~# cat /etc/sysrepo-tsn/Instances/ietf-ip-cfg.xml
<interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
            xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
            xmlns:ip="urn:ietf:params:xml:ns:yang:ietf-ip">
        <interface>
                <name>eth0</name>
                <enabled>true</enabled>
                <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd</type>
                <ip:ipv4>
                        <ip:enabled>true</ip:enabled>
                        <ip:forwarding>false</ip:forwarding>
                        <ip:address>
                                <ip:ip>192.168.78.129</ip:ip>
                                <ip:netmask>255.255.0.0</ip:netmask>
                        </ip:address>
                </ip:ipv4>
        </interface>
</interfaces>
```

6.2.2.2 Qbv Configuration

- On Genio platform Kill NetworkManager process.

```
# ps -aux | grep NetworkManager

# kill xxx
```

where xxx stands for PID of NetworkManger process.

- On Ubuntu, run netopeer2 client, and send Qbv configuration file (qbv-eth0.xml) to “running” datastore.

```
# netopeer2-cli

// 192.168.1.50 is the ip address of eth1
> connect --login root --host 192.168.1.50

// configure Qbv with qbv-eth0.xml
// change the path of qbv-eth0.xml according to your test environment
> edit-config --target running --config=/etc/sysrepo-tsn/Instances/qbv-eth0.xml

// check the Qbv configure data
> get-config --source running --filter-xpath /ietf-interfaces:interfaces/interface
[name='eth0']/ieee802-dot1q-bridge:bridge-port
```

- Check the result on Genio board, the default qdisc is replaced with taprio Qbv qdisc.

```
root@i1200-demo:~# tc qdisc show dev eth0
qdisc taprio 100: root tc 4 map 0 1 2 3 0 0 0 0 0 0 0 0 0 0 0 0
queues offset 0 count 1 offset 1 count 1 offset 2 count 1 offset 3 count 1
clockid invalid flags 0x2       base-time 5000 cycle-time 100000 cycle-time-extension 0
        index 0 cmd S gatemask 0x5 interval 4000
        index 1 cmd S gatemask 0x7 interval 3000
        index 2 cmd S gatemask 0x7 interval 3000

qdisc pfifo 0: parent 100:4 limit 1000p
qdisc pfifo 0: parent 100:3 limit 1000p
qdisc pfifo 0: parent 100:2 limit 1000p
qdisc pfifo 0: parent 100:1 limit 1000p
```

Snippet in qbv-eth0.xml:

```
<dot1q:bridge-port>
        <sched:gate-parameter-table>
                <sched:queue-max-sdu-table>
                        <sched:traffic-class>0</sched:traffic-class>
                        <sched:queue-max-sdu>1024</sched:queue-max-sdu>
                </sched:queue-max-sdu-table>
                <sched:gate-enabled>true</sched:gate-enabled>
                <sched:admin-gate-states>127</sched:admin-gate-states>
                <sched:config-change>true</sched:config-change>
                <sched:supported-list-max>10</sched:supported-list-max>
                <sched:supported-interval-max>1000000000</sched:supported-interval-max>
                <sched:admin-base-time>
                        <sched:seconds>0</sched:seconds>
                        <sched:nanoseconds>5000</sched:nanoseconds>
                </sched:admin-base-time>
                <sched:admin-cycle-time>
                        <sched:numerator>1</sched:numerator>
                        <sched:denominator>10000</sched:denominator>
                </sched:admin-cycle-time>
                <sched:admin-control-list>
                        <sched:gate-control-entry>
                                <sched:index>0</sched:index>
                                <sched:operation-name>sched:set-gate-states</sched:operation-name>
                                <sched:gate-states-value>5</sched:gate-states-value>
                                <sched:time-interval-value>4000</sched:time-interval-value>
                        </sched:gate-control-entry>
                        <sched:gate-control-entry>
                                <sched:index>1</sched:index>
                                <sched:operation-name>sched:set-gate-states</sched:operation-name>
                                <sched:gate-states-value>7</sched:gate-states-value>
                                <sched:time-interval-value>3000</sched:time-interval-value>
                        </sched:gate-control-entry>
                        <sched:gate-control-entry>
                                <sched:index>2</sched:index>
                                <sched:operation-name>sched:set-gate-states</sched:operation-name>
                                <sched:gate-states-value>7</sched:gate-states-value>
                                <sched:time-interval-value>3000</sched:time-interval-value>
                        </sched:gate-control-entry>
                </sched:admin-control-list>
        </sched:gate-parameter-table>
</dot1q:bridge-port>
```

- Edit the qbv-eth0.xml to set gate-enabled to false, and configure xml again, the qbv qdisc will be deleted.

```
<dot1q:bridge-port>
        <sched:gate-parameter-table>
                <sched:queue-max-sdu-table>
                        <sched:traffic-class>0</sched:traffic-class>
                        <sched:queue-max-sdu>1024</sched:queue-max-sdu>
                </sched:queue-max-sdu-table>
                <sched:gate-enabled>false</sched:gate-enabled>
                <sched:admin-gate-states>127</sched:admin-gate-states>
                <sched:config-change>true</sched:config-change>
                <sched:supported-list-max>10</sched:supported-list-max>
                <sched:supported-interval-max>1000000000</sched:supported-interval-max>
                <sched:admin-base-time>
                        <sched:seconds>0</sched:seconds>
                        <sched:nanoseconds>5000</sched:nanoseconds>
                </sched:admin-base-time>
                <sched:admin-cycle-time>
                        <sched:numerator>1</sched:numerator>
                        <sched:denominator>10000</sched:denominator>
                </sched:admin-cycle-time>
                <sched:admin-control-list>
                        <sched:gate-control-entry>
                                <sched:index>0</sched:index>
                                <sched:operation-name>sched:set-gate-states</sched:operation-name>
                                <sched:gate-states-value>5</sched:gate-states-value>
                                <sched:time-interval-value>4000</sched:time-interval-value>
                        </sched:gate-control-entry>
                        <sched:gate-control-entry>
                                <sched:index>1</sched:index>
                                <sched:operation-name>sched:set-gate-states</sched:operation-name>
                                <sched:gate-states-value>7</sched:gate-states-value>
                                <sched:time-interval-value>3000</sched:time-interval-value>
                        </sched:gate-control-entry>
                        <sched:gate-control-entry>
                                <sched:index>2</sched:index>
                                <sched:operation-name>sched:set-gate-states</sched:operation-name>
                                <sched:gate-states-value>7</sched:gate-states-value>
                                <sched:time-interval-value>3000</sched:time-interval-value>
                        </sched:gate-control-entry>
                </sched:admin-control-list>
        </sched:gate-parameter-table>
</dot1q:bridge-port>
```

```
// netopeer2-cli
// configure Qbv with qbv-eth0.xml with gate-enabled = false
> edit-config --target running --config=/etc/sysrepo-tsn/Instances/qbv-eth0.xml
```

- Check the result on Genio board, the taprio Qbv qdisc is deleted, and restore to default qdisc.

```
root@i1200-demo:~# tc qdisc show dev eth0
qdisc mq 0: root
qdisc pfifo_fast 0: parent :4 bands 3 priomap 1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
qdisc pfifo_fast 0: parent :3 bands 3 priomap 1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
qdisc pfifo_fast 0: parent :2 bands 3 priomap 1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
qdisc pfifo_fast 0: parent :1 bands 3 priomap 1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
```

6.2.2.3 Qbu Configuration

The Qbu configuration is almost same with Qbv, use qbu-eth0.xml instead.

```
// netopeer2-cli
// configure Qbv with qbu-eth0.xml
> edit-config --target running --config=/etc/sysrepo-tsn/Instances/qbu-eth0.xml
```

```
root@i1200-demo:~# tc qdisc show dev eth0
qdisc taprio 100: root tc 4 map 0 1 2 3 0 0 0 0 0 0 0 0 0 0 0 0
queues offset 0 count 1 offset 1 count 1 offset 2 count 1 offset 3 count 1
clockid invalid flags 0x2       base-time 5000 cycle-time 100000 cycle-time-extension 0
        index 0 cmd SH gatemask 0x5 interval 4000
        index 1 cmd SR gatemask 0x7 interval 6000

qdisc pfifo 0: parent 100:4 limit 1000p
qdisc pfifo 0: parent 100:3 limit 1000p
qdisc pfifo 0: parent 100:2 limit 1000p
qdisc pfifo 0: parent 100:1 limit 1000p
```

6.3 Installing Netopeer2-cli on Ubuntu18.04

Use the following steps for installing Netopeer2-cli on Ubuntu18.04 operating systems.

1. Install the following packages:

```
$ sudo apt install -y git cmake build-essential bison autoconf dh-autoreconf flex
$ sudo apt install -y libavl-dev libprotobuf-c-dev protobuf-c-compiler zlib1g-dev
$ sudo apt install -y libgcrypt20-dev libssh-dev libev-dev libpcre3-dev
```

2. Install libyang:

```
$ git clone https://github.com/CESNET/libyang.git
$ cd libyang;
$ git checkout v1.0-r4 -b v1.0-r4
$ mkdir build; cd build
$ cmake -DCMAKE_INSTALL_PREFIX:PATH=/usr ..
$ make
$ sudo make install
```

3. Install sysrepo (v0.7.8):

```
$ git clone https://github.com/sysrepo/sysrepo.git
$ cd sysrepo
$ git checkout v0.7.8 -b v0.7.8
$ mkdir build; cd build
$ cmake -DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX:PATH=/usr ..
$ make
$ sudo make install
```

•

4. Install libnetconf2：

```
$ git clone https://github.com/CESNET/libnetconf2.git
$ cd libnetconf2
$ git checkout v0.12-r2 -b v0.12-r2
$ mkdir build; cd build
$ cmake -DCMAKE_INSTALL_PREFIX:PATH=/usr ..
$ make
$ sudo make install
```

•

5. Install protobuf:

```
$ git clone https://github.com/protocolbuffers/protobuf.git
$ cd protobuf
$ sudo apt-get update
$ sudo apt-get install libtool
$ git checkout v3.21 -b v3.21
$ git submodule update --init --recursive
$ ./autogen.sh
$ ./configure
$ make
$ sudo make install
$ sudo ldconfig # refresh shared library cache.
```

6. Install Netopeer2-cli (v0.7-r2):

```
$ git clone https://github.com/CESNET/Netopeer2.git
$ cd Netopeer2
$ git checkout v0.7-r2 -b v0.7-r2
$ cd cli
$ cmake -DCMAKE_INSTALL_PREFIX:PATH=/usr .
$ make
$ sudo make install
```

7 Reference

[1]. http://linuxptp.sourceforge.net/
[2]. https://gist.github.com/jeez/bd3afeff081ba64a695008dd8215866f
[3]. https://tsn.readthedocs.io/index.html
[4]. https://www.spinics.net/lists/netdev/msg460869.html
[5]. https://patchwork.ozlabs.org/project/netdev/cover/20180703224300.25300-1-jesus.sanchez-palencia@intel.com/
[6]. https://patchwork.ozlabs.org/project/netdev/cover/20180714000536.1008-1-vinicius.gomes@intel.com/
[7]. https://www.nxp.com/docs/en/user-guide/REALTIMEEDGEUG_REV2.2.pdf
[8]. https://netopeer.liberouter.org/doc/sysrepo/master/html/
[9]. https://github.com/CESNET/netopeer2

Exhibit 1 Terms and Conditions

Your access to and use of this document and the information contained herein (collectively this "Document") is subject to your (including the corporation or other legal entity you represent, collectively "You") acceptance of the terms and conditions set forth below ("T&C"). By using, accessing or downloading this Document, You are accepting the T&C and agree to be bound by the T&C. If You don't agree to the T&C, You may not use this Document and shall immediately destroy any copy thereof.

This Document contains information that is confidential and proprietary to MediaTek Inc. and/or its affiliates (collectively "MediaTek") or its licensors and is provided solely for Your internal use with MediaTek's chipset(s) described in this Document and shall not be used for any other purposes (including but not limited to identifying or providing evidence to support any potential patent infringement claim against MediaTek or any of MediaTek's suppliers and/or direct or indirect customers). Unauthorized use or disclosure of the information contained herein is prohibited. You agree to indemnify MediaTek for any loss or damages suffered by MediaTek for Your unauthorized use or disclosure of this Document, in whole or in part.

MediaTek and its licensors retain titles and all ownership rights in and to this Document and no license (express or implied, by estoppels or otherwise) to any intellectual propriety rights is granted hereunder. This Document is subject to change without further notification. MediaTek does not assume any responsibility arising out of or in connection with any use of, or reliance on, this Document, and specifically disclaims any and all liability, including, without limitation, consequential or incidental damages.

THIS DOCUMENT AND ANY OTHER MATERIALS OR TECHNICAL SUPPORT PROVIDED BY MEDIATEK IN CONNECTION WITH THIS DOCUMENT, IF ANY, ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE. MEDIATEK SPECIFICALLY DISCLAIMS ALL WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, COMPLETENESS OR ACCURACY AND ALL WARRANTIES ARISING OUT OF TRADE USAGE OR OUT OF A COURSE OF DEALING OR COURSE OF PERFORMANCE. MEDIATEK SHALL NOT BE RESPONSIBLE FOR ANY MEDIATEK DELIVERABLES MADE TO MEET YOUR SPECIFICATIONS OR TO CONFORM TO A PARTICULAR STANDARD OR OPEN FORUM.

Without limiting the generality of the foregoing, MediaTek makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does MediaTek assume any liability arising out of the application or use of any product, circuit or software. You agree that You are solely responsible for the designing, validating and testing Your product incorporating MediaTek's product and ensure such product meets applicable standards and any safety, security or other requirements.

The above T&C and all acts in connection with the T&C or this Document shall be governed, construed and interpreted in accordance with the laws of Taiwan, without giving effect to the principles of conflicts of law.