



Wireless Time-Sensitive Networking (WTSN) Reference Software for Linux* with Intel® Wireless AC9560

User Guide

January 2022

Intel Padova Engagement



You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting: <http://www.intel.com/design/literature.htm>

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at <http://www.intel.com/> or from the OEM or retailer.

No product or component can be absolutely secure.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2022, Intel Corporation. All rights reserved.

Revision History

Date	Revision	Description
January 2022	1.0	Initial release.

Contents

1.0	Introduction.....	7
1.1	Terminology Usage.....	7
1.2	Scope and Purpose.....	7
1.3	System Overview and Flow Description	8
1.4	Hardware and Software Requirements	10
2.0	Getting Started Guide	11
2.1	Install and Build Wireless TSN features on Host Machines.....	12
	Step 1. Prepare host machines for dependencies.....	12
	Step 2. Download Intel® Wireless TSN Software Stack.....	13
	Step 3. Install and update the kernel version	13
	Step 4. Copy binaries and scripts.....	13
	Step 5. Prepare Intel® Wi-Fi Firmware.....	14
	Step 6. Change the wireless network name.....	14
2.2	Run Wireless TSN stack	15
	2.2.1 Network Topology	15
	2.2.2 Configure Machine A and B.....	15
	2.2.3 Get WTSN Connected between Machine A and B.....	17
2.3	Enable System Time Disciplining.....	21
3.0	Demo 1: 802.1AS Time Synchronization Performance Measurement.....	23
3.1	Hardware Set Up.....	23
3.2	Measure Time Synchronization Performance.....	23
4.0	Demo 2: 802.1Qbv Time Aware Shaper.....	27
4.1	Hardware Set Up.....	27
4.2	Measure Time-Aware Traffic Scheduling.....	27
	4.2.1 Without the Time-Aware Traffic Scheduling	28
	4.2.2 With the Time-Aware Traffic Scheduling Enabled.....	30
5.0	Frequently Asked Questions (FAQ).....	33
5.1	How many Wi-Fi* TX queues could be used for Queuing Disciplines?.....	33
5.2	How to get more debug messages from Wireless TSN components?	33
5.3	Why do I get the error messages when do plot visualization - QXcbConnection: XCB error: 145 (Unknown) with MobaXTerm?.....	35
5.4	How to configure SSH public key Authentication to easier auto login?	35
5.5	How to Enable Soft AP mode at 5GHz in WTSN?.....	37
	5.5.1 Configure wpa_supplicant.conf in Machine A for 5GHz.....	38
	5.5.2 Enable Station mode and connect to 5Ghz Wireless Access Point in Machine A.	39
	5.5.3 Check if Machine A is connected to 5GHz Wi-Fi* AP and which 5GHz channel	39

5.5.4	Configure the hostapd.conf in Machine A for 5GHz soft AP	39
5.5.5	Enable 5GHz softAP mode while Station is active in Machine A	40
5.5.6	Check if we enable 5GHz soft AP successfully by hostapd_cli in Machine A.....	40
5.5.7	Start station mode on Machine B and connect to 5GHz Soft AP on Machine A.	42
5.6	How to measure the end-to-end latency without include application latency.	43
5.6.1	What is included in the measurement tools?	43
5.6.2	Install the necessary component for measurement tool	43
5.6.3	Running the tool and having measurement result.....	45
6.0	Learn More.....	46
6.1	IEEE 1588 and IEEE 802.1AS-2011	46
6.2	IEEE 802.1Qbv	49
6.3	Queue Disciplines	52
6.4	Wireless TSN Application Classes.....	52
6.5	Glossary	54
6.6	Terminology	56



Figures

Figure 1: 802.11v/mc Timing Measurement Protocol to Propagate Reference Time	9
Figure 2: 802.11mc Fine Timing Measurement Protocol	9
Figure 3: Wireless Time Sensitive Network Software Stack Overview	10
Figure 4: Network Topology for Wireless TSN	12
Figure 5: Network Topology for Wireless TSN	15
Figure 6: Hardware Set Up for Wireless TSN.....	23
Figure 7: Hardware Set Up for Wireless TSN.....	27
Figure 8: Network topology of Soft AP at 5Ghz	38
Figure 9: gPTP in Action between Grandmaster and Subordinate Clocks	47
Figure 10: Time Synchronization Demo: Software / Application in Machine A	48
Figure 11: Time Related Parameters for OPC UA Pub/Sub Communication	Error! Bookmark not defined.

Tables

No table of figures entries found.

1.0 Introduction

This user guide describes the procedure to download, install and use Intel® Wireless Time Sensitive Network (WTSN) Reference Software for Linux* on Intel platform with Intel AC 9560 Wireless card. It includes two demos and walks users through the features and capabilities of this reference software.

1.1 Terminology Usage

This document uses the terms primary and secondary which is known as “master” and “slave” as defined in IEEE 802.1AS Std.

1.2 Scope and Purpose

The WTSN reference stack is an effort to enable Time Sensitive Networking (TSN) features on top of Wireless. Currently the stack supports the following features:

- IEEE802.1AS (Time Synchronization)
- IEEE802.1Qbv (Time Based Prioritization)

The WTSN reference stack builds upon existing software components or stack listed as follows and adds functionality to enable the above mentioned TSN specific features.

- OpenAvnu IEEE 802.1AS software stack
- Intel® Wi-Fi* device Software stack (including wpa_supplicant and hostapd)

The functionality indicated by the green boxes above are new additions that have been introduced into the standard software stack.

The WTSN stack contains the following components:

- 1. Intel® Wi-Fi* Device Driver changes**
 - Enable PHC interface
 - Enable sending and receiving of IEEE 802.11v Timing measurement Frames in Wi-Fi*.
 - Propagate ToA (Time of Arrival) and ToD (Time of Departure) up the TSN stack.
- 2. wpa_supplicant and hostapd**
 - Handle exchange and parsing of Timing Measurement Frames (IEEE 802.11v) and payloads.
- 3. IEEE 802.1AS daemon**
 - Implement a wireless port on Linux*

- System Time Disciplining of the secondary system_clock with respect to the primary system_clock.
- 4. **Wi-Fi* aware Qbv Scheduler Script**
 - Reference script showing configuration of the Qbv schedules for Time aware Traffic shaping using Linux Q-disc.
- 5. **Scripts** to operate and configure the above components
- 6. **Scripts and tools** to measure the performance of 802.1AS Time Synchronization and 802.1QBV Traffic shaping - to visualize the data and performance of the WTSN System.

1.3 System Overview and Flow Description

This section will provide high level functional description of the enhancements made to the various components in the WTSN stack and describe the flow of timing related information between the various components of the WTSN stack.

The WTSN stack builds upon the OpenAvnu implementation of the 802.1AS time synchronization protocol to add time synchronization over 802.11 protocol. A functional Linux* Wireless Port implementation was added to affect the 802.1AS time synchronization protocol over 802.11.

This included interfacing with 802.11mc/11v support and PHC clock support in the Intel 802.11 Wireless Stack to effect exchange of timing measurement frames to implement the 802.1AS timing protocol. The 802.1AS protocol propagates a reference time, typically produced by a precise Grand Primary clock, to all the leaf nodes in a network via timing measurement frames.

In the 802.11 domain, the AP acts as a relay and helps extend the reference time from the wired domain to the wireless domain. The 802.1AS protocol uses timing measurement frames to accurately propagate the reference time to all ports connected to the bridge. In the case of 802.11 AP, we have wireless ports, connecting wireless nodes via 802.11 links, and as well wired port connecting to a wired domain.

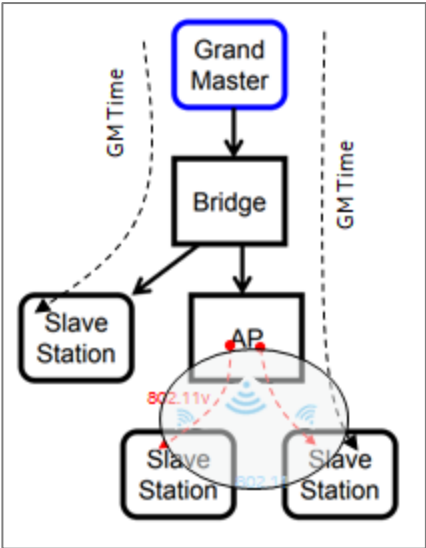


Figure 1: 802.11v/mc Timing Measurement Protocol to Propagate Reference Time

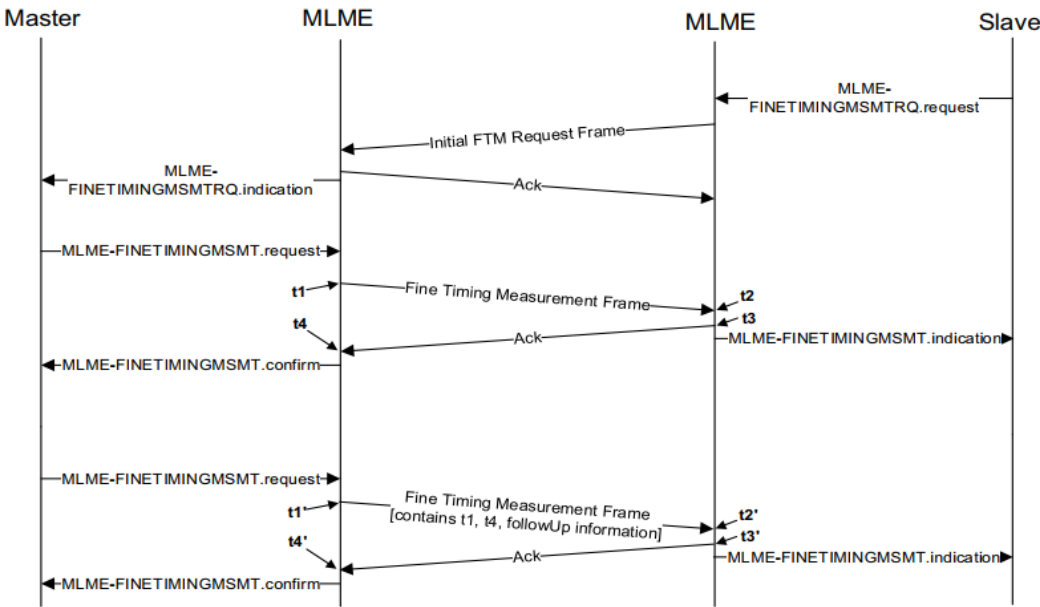



Figure 2: 802.11mc Fine Timing Measurement Protocol

In the 802.11 domain, 802.1AS implementation takes advantage of the 802.11v Timing measurement protocol to propagate the reference time.

1.4 Hardware and Software Requirements

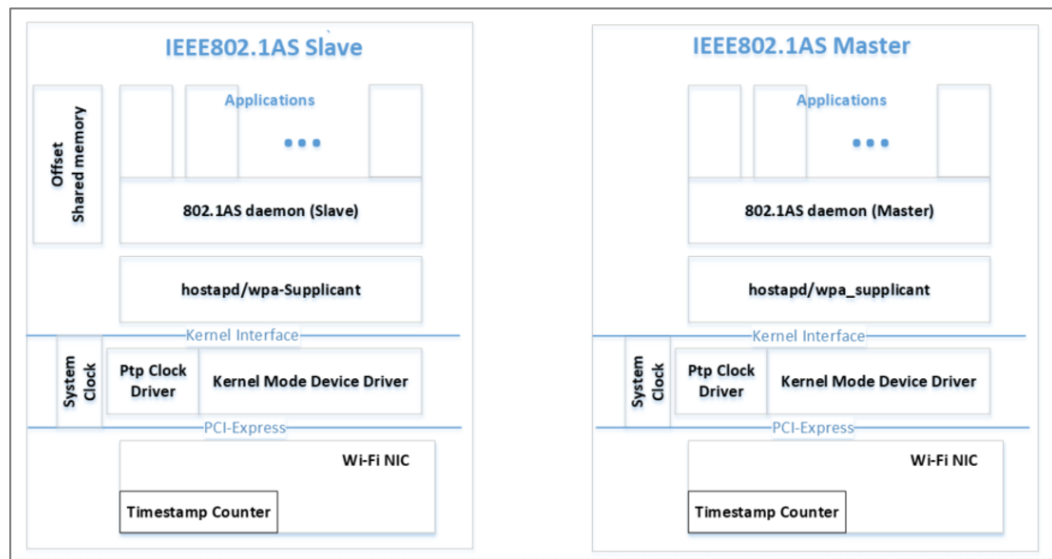
Hardware Requirements

- **Platform:** Two Intel® NUCs - NUC8i7BEH (similar platform with Intel CNVi support). or
 
- Setup one host machine for soft AP operation that acts as the primary system.
- Another machine is for Wi-Fi* Station operation that acts as the secondary system.
- **Wireless:** Intel® Wireless AC9560 M.2 Modules

Software Requirements

- Ubuntu* 18.04 with Linux kernel 5.1.21
- Intel's Wireless Time Sensitive Network software stack

Figure 3: Wireless Time Sensitive Network Software Stack Overview



§

2.0 Getting Started Guide

This chapter describes setting up and installing the wireless components to support Wireless TSN on Ubuntu*. Once the drivers and user space applications are built, users can integrate and enabling Wireless TSN features on Ubuntu*.

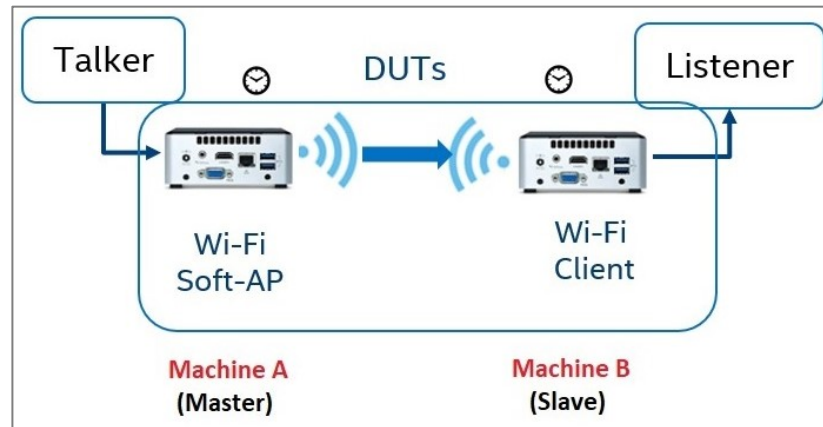
This chapter covers:

Steps	Description
1	Prepare host machine for software dependencies
2	Download Intel® Wireless TSN Software Stack
3	Install and set up Intel® Wi-Fi Driver stack
4	Setup wpa_supplicant and shared library
5	Setup up hostapd
6	Setup 802.1AS daemon
7	Prepare Intel® Wi-Fi* firmware
8	Change the wireless network name

2.1 Install and Build Wireless TSN features on Host Machines

Prepare two host machines – Machine A and B for Wireless TSN environment setup. Setup one host machine for soft AP operation and another machine for Wi-Fi* Station or Client operation. Wireless TSN software components are needed to install on both sides.

Figure 4: Network Topology for Wireless TSN



Step 1. Prepare host machines for dependencies

a) Prepare host machine with Ubuntu* OS 18.04.

- Check Ubuntu* version using this command line

```
$ lsb_release -a
```

```
Last login: Mon Apr 27 15:50:21 2020 from 10.0.0.8
andrew@Andrew-NUCi5:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:   Ubuntu 18.04.4 LTS
Release:      18.04
Codename:     bionic
andrew@Andrew-NUCi5:~$
```

- Steps to upgrade Ubuntu* 16.04 to 18.04 (if needed)

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt dist-upgrade
$ sudo do-release-upgrade
```

- b) Install required software packages.
- Following software packages are needed.
 - libreadline-dev
 - libncurses5-dev
 - libdbus-1-dev
 - libnl-genl-3-dev
 - libssl-dev
 - gawk
 - bison
 - flex

- `sudo apt-get install <package name>`

```
$ sudo apt update
$ sudo apt install -y libreadline-dev libncurses5-dev \
    libdbus-1-dev libnl-genl-3-dev \
    libssl-dev gawk bison flex iw
```

Step 2. Download Intel® Wireless TSN Software Stack

- a) Download the wtsn stack package from the shared drive.
- b) Untar the tarball (`tar -xvzf wtsn.tar.gz`)

```
$ cd <wtsn>
$ tar -xvzf wtsn.tar.gz
```

- c) The WTSN BSP file structures as below. It includes 6 major components:
- QBV and time sync scripts: Sample app that controls underlying 802.1AS (gPTP daemon) and Wi-Fi stack for Time sync and QBV.
 - `iwlwifi-stack-dev_new`: Intel® Wi-Fi Driver source code
 - `iwlwifi-hostap-dev`: `wpa_supplicant` and `hostapd` source code
 - gPTP: 802.1As daemon source code
 - config: sample configuration files

Step 3. Install and update the kernel version

```
$ sudo dpkg -i <wtsn>/kernel/*.deb
```

Step 4. Copy binaries and scripts

```
$ sudo mkdir -p /usr/share/intel/wtsn
$ sudo cp -rf <wtsn>/bin /usr/share/intel/wtsn
$ sudo cp -rf <wtsn>/lib /usr/share/intel/wtsn
$ sudo cp -rf <wtsn>/scripts /usr/share/intel/wtsn
$ sudo ln -s /usr/share/intel/wtsn/bin/wpa_supplicant
    /usr/sbin/wpa_supplicant
```

```
$ sudo ln -s /usr/share/intel/wtsn/bin/wpa_cli /usr/sbin/wpa_cli
$ sudo ln -s /usr/share/intel/wtsn/bin/hostapd /usr/sbin/hostapd
$ sudo ln -s /usr/share/intel/wtsn/bin/hostapd_cli /usr/sbin/hostapd_cli
$ sudo ln -s /usr/share/intel/wtsn/bin/daemon_cl /usr/sbin/daemon_cl
$ sudo ln -s /usr/share/intel/wtsn/bin/startAp.sh /usr/sbin/startAp.sh
$ sudo ln -s /usr/share/intel/wtsn/bin/startSta.sh /usr/sbin/startSta.sh
$ sudo ln -s /usr/share/intel/wtsn/lib/libwpa_client.so /usr/local/lib/libwpa_client.so
```

Step 5. Prepare Intel® Wi-Fi Firmware

- a) Install Wi-Fi* firmware.

```
$ cd <wtsn>/fw
```

- b) Delete any existing AC9560 FWs from the **/lib/firmware** directory.

```
$ sudo rm /lib/firmware/iwlwifi-9000-*.ucode
```

- c) Copy Wi-Fi firmware version .64 and PNVM files to **/lib/firmware**

```
$ cd <wtsn>/fw
$ sudo cp iwlwifi-9000-*.ucode /lib/firmware
```

- d) Copy Wi-Fi drivers to **/lib/modules/5.1.21-tgpio-wtsn-tgpio**

```
$ cd <wtsn>/driver
$ sudo cp -rf updates /lib/modules/5.1.21-tgpio-wtsn-tgpio/.
```

Step 6. Change the wireless network name

After the Ubuntu* installation, the default wireless network interface name may get changed to wlp2s0 from the default name wlan0. If this happens and if you desire to retain the wireless interface name as 'wlan0', please follow up below steps to change.

- a) Configure **/etc/default/grub** and modify below parameters

```
$ sudo vim /etc/default/grub
GRUB_CMDLINE_LINUX="net.ifnames=0 biosdevname=0"
```

```
GRUB_DEFAULT=0
GRUB_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT=0
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX="net.ifnames=0 biosdevname=0"
```

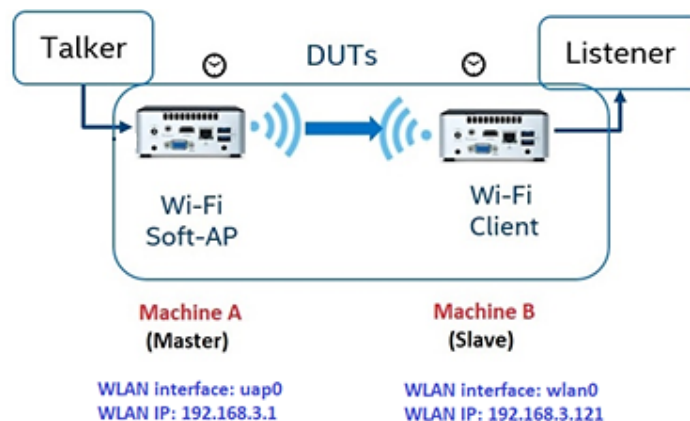
- b) Update the grub.cfg to enable this modification. Then reboot device.

```
$ sudo update-grub
$ sudo reboot
```

2.2 Run Wireless TSN stack

2.2.1 Network Topology

Figure 5: Network Topology for Wireless TSN



2.2.2 Configure Machine A and B

2.2.2.1 Disable Power Save Mode

Ensure that the power save setting on the Wi-Fi* is disabled. When the STA goes into Power Save mode, it misses the Timing Measurement frames sent to it by the AP causing the IEEE802.1AS secondary state machine to time-out and reset. This limitation would be taken care of in the production version of the stack that would be more optimized.

1. Append the following line to `/etc/modprobe.d/iwlwifi.conf`, if already existing in this path, or else to create new file directly in below path and add this line.

```
$ sudo vim /etc/modprobe.d/iwlwifi.conf

options iwlvm power_scheme=1
```

```
andrew@Andrew-NUCi5:/etc/modprobe.d$ cat iwlwifi.conf
# /etc/modprobe.d/iwlwifi.conf
# iwlwifi will dynamically load either iwldvm or iwlvm depending on the
# microcode file installed on the system. When removing iwlwifi, first
# remove the iwl?vm module and then iwlwifi.
remove iwlwifi \
(/sbin/lsmod | grep -o -e ^iwlvm -e ^iwlwifi | xargs /sbin/rmmod) \
&& /sbin/modprobe -r mac80211
options iwlvm power scheme=1
```

2.2.2.2 Ensure Network Manager is NOT Managing the Wireless Interface

NOTES

In case this is not done, then Network manager will also compete for the Wireless interface, which may result in undefined behavior.

1. Install additional net-tools package

```
$ sudo apt install net-tools
```

2. List down the MAC address of wireless interface

```
$ ifconfig -a
```

```
andrew@Andrew-NUCi5:/etc/NetworkManager$ ifconfig -a
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.16 netmask 255.255.255.0 broadcast 10.0.0.255
    inet6 fe80::546:6799:6f68:6633 prefixlen 64 scopeid 0x20<link>
    ether 88:88:88:88:87:88 txqueuelen 1000 (Ethernet)
    RX packets 99731 bytes 11126572 (11.1 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 6080 bytes 731257 (731.2 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 16 memory 0xdff00000-dff20000

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 689 bytes 55607 (55.6 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 689 bytes 55607 (55.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4098<BROADCAST,MULTICAST> mtu 1500
    ether d4:6d:6d:f3:e2:9c 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
```

3. Edit **/etc/NetworkManager/NetworkManager.conf** file (**Note:** In MAC, below should all be in lowercase alphabets):

```
$ sudo vim /etc/NetworkManager/NetworkManager.conf
```



```
[keyfile]
unmanaged-devices=mac:<MAC Address of Wireless Interface>
```

```
[main]
plugins=ifupdown,keyfile

[ifupdown]
managed=false

[device]
wifi.scan-rand-mac-address=no

[keyfile]
unmanaged-devices=mac:d4:6d:6d:f3:e2:9c
```

4. Restart Network Manager to enable above setting.

```
$ sudo service network-manager restart
```

2.2.3 Get WTSN Connected between Machine A and B

Configure *hostapd.conf* in Machine A and *wpa_supplicant.conf* in Machine B. Refer to respective documentation in this regard. From the standpoint of WTSN stack, the only thing that needs to be added to these configurations are the presence of the following setting.

2.2.3.1 Configure *hostapd.conf* in Machine A

1. Configure the *hostapd.conf*. A sample configuration file is located at **<wtsn>/config/hostapd.conf**. And you can reuse this configuration file directly (recommended).
2. Copy **<wtsn>/config/hostapd.conf** to **/etc/hostapd/** directory.

```
$ sudo mkdir -p /etc/hostapd
$ sudo cp <wtsn>/config/hostapd.conf /etc/hostapd/
```

2.2.3.2 Configure *wpa_supplicant.conf* in Machine B

1. Configure *wpa_supplicant.conf*. A sample configuration file is located at **<wtsn>/config/wpa_supplicant.conf**. And you can reuse this configuration file directly (recommended).
2. Copy the templated config or newly created config to **/etc/wpa_supplicant/** directory.

```
$ sudo mkdir -p /etc/wpa_supplicant
$ cd <wtsn>/config/
$ sudo cp <wtsn>/config/wpa_supplicant.conf /etc/wpa_supplicant/
```

2.2.3.2.1 Start using the Machine A acts as primary machine.

1. Install screen package for terminal.

```
$ sudo apt-get install screen
```

2. Start the hostapd (soft Access Point)

```
$ sudo startAp.sh
```

3. Start the time synchronization as Grand Master.

```
$ sudo startGptpM.sh
```

2.2.3.2.2 Start Wi-Fi* Station in the Machine B acts as secondary machine.

1. Install screen package.

```
$ sudo apt-get install screen
```

2. Start the station.

```
$ sudo startSta.sh
```

3. Verify connectivity

```
$ sudo wpa_cli -i wlan0 status
```

4. Start time synchronization as slave

```
$ sudo startGptpS.sh
```

2.2.3.3 Configure and Run IEEE 802.1Qbv Time Aware Shaper between Machine A and B

A traffic generation program capable of generating timed packets and having the ability to measure latency observed can then be used to measure impact in the presence of background traffic.

1. Download the TC from iproute2 in Machine A and B.

Linux* kernel's queue disciplines need TC (traffic control) tool to configure your queue schedule policy. TC is part of iproute2. We need to install the latest version.

```
$ cd <wtsn>/  
$ mkdir iproute2  
$ git clone git://git.kernel.org/pub/scm/network/iproute2/iproute2.git
```

```
andrew@Andrew-NUCi3:~/workspace/wtsn/iproute2$ git clone git://git.kernel.org/pub/scm/network/iproute2/iproute2.git
Cloning into 'iproute2'...
remote: Enumerating objects: 4655, done.
remote: Counting objects: 100% (4655/4655), done.
remote: Compressing objects: 100% (2419/2419), done.
remote: Total 27835 (delta 3274), reused 3213 (delta 2149), pack-reused 23180
Receiving objects: 100% (27835/27835), 5.62 MiB | 4.15 MiB/s, done.
Resolving deltas: 100% (21042/21042), done.
andrew@Andrew-NUCi3:~/workspace/wtsn/iproute2$
```

2. Build and install the TC.

Linux* kernel's queue disciplines need TC (traffic control) tool to configure your policy. TC is part of iproute2. We need to install the latest version.

```
$ cd <wtsn>/iproute2
$ make
$ sudo make install
$ tc -V
```

```
andrew@Andrew-NUCi3:~/workspace/wtsn/iproute2$ make
lib
CC      libgenl.o
CC      libnetlink.o
libnetlink.c:151:2: warning: #warning "libmnl required for error support" [-Wcpp]
#warning "libmnl required for error support"
~~~~~
AR      libnetlink.a
CC      utils.o
CC      rt_names.o
CC      ll_map.o
CC      ll_types.o
CC      ll_proto.o
CC      ll_addr.o
CC      inet_proto.o
CC      namespace.o
CC      json_writer.o
CC      json_print.o
CC      names.o
CC      color.o
CC      bpf.o
CC      exec.o
CC      fs.o
CC      mpls_ntop.o
CC      mpls_pton.o
AR      libutil.a
```

```
andrew@Andrew-NUCi3:~/workspace/wtsn/iproute2$ tc -V
tc utility, iproute2-ss200330
andrew@Andrew-NUCi3:~/workspace/wtsn/iproute2$
```

3. Install the iPerf version 2, since only this version supports enhanced analysis mode.

- iPerf - Traffic generation program capable of generating timed packets and having the ability to measure latency observed can then be used to measure impact in the presence of background traffic.
- Download iPerf version 2 (iPerf 2.0.13) from <https://sourceforge.net/projects/iperf2/files/> to temp folder.

- Build the iPerf version 2.

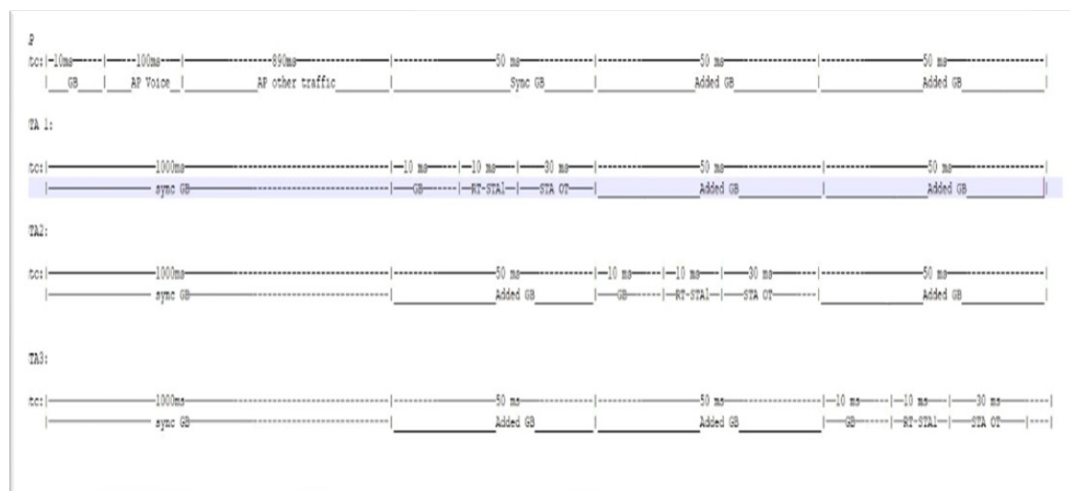
```
$ cd ~/temp/iperf-2.0.13
$ ./config
$ make
$ sudo make install
```

```
andrew@Andrew-NUCi3:~/temp/iperf-2.0.13/src$ ./iperf -v
iperf version 2.0.13 (21 Jan 2019) pthreads
```

```
andrew@Andrew-NUCi3:~/temp/iperf-2.0.13/src$ ./iperf -h
Usage: iperf [-s|-c host] [options]
       iperf [-h|--help] [-v|--version]

Client/Server:
  -b, --bandwidth #[kmgKMG | pps]  bandwidth to send at in bits/sec or packets per second
  -e, --enhancedreports             use enhanced reporting giving more tcp/udp and traffic information
  -f, --format [kmgKMG]            format to report: Kbits, Mbits, KBytes, MBytes
  -i, --interval #                 seconds between periodic bandwidth reports
  -l, --len #[kmKM]                length of buffer in bytes to read or write (Default is: TCP=128K, v4 UDP=1470, v6 UDP=1450)
  -m, --print_mss                  print TCP maximum segment size (MTU - TCP/IP header)
  -o, --output <filename>         output the report or error message to this specified file
  -p, --port #                     server port to listen on/connect to
  -u, --udp                        use UDP rather than TCP
      --udp-counters-64bit          use 64 bit sequence numbers with UDP
  -w, --window #[KM]              TCP window size (socket buffer size)
  -z, --realtime                   request realtime scheduler
  -B, --bind <host>[:<port>][%<dev>] bind to <host>, ip addr (including multicast address) and optional port and device
  -C, --compatibility              for use with older versions does not send extra msgs
  -M, --mss #                     set TCP maximum segment size (MTU - 40 bytes)
  -N, --nodelay                    set TCP no delay, disabling Nagle's Algorithm
```

NOTES



Do not use the default iperf version i.e. iperf3 that is installed with 'sudo apt install iperf' as this **does not** support **enhanced analysis mode** required for WTSN performance measurements.

Figure 6: QBV Scheduling between Machine A and Machine B's

4. Run the 802.1Qbv Time Aware Shaper in **Machine A**

- Select option 2 and then option 1 (ADD) to configure the parameters.
 - Enter the number of Station device connected or to be connected to AP. This is important for properly synching QBV window of each device.
 - Select option to change any of the QBV config, else enter 0 (ZERO) to submit the config.
- NOTE:** The Synchronized Guard band should be equal or greater than Machine B (STA) device total QBV Window.

5. Run the 802.1Qbv Time Aware Shaper in **Machine B.**

- After successful connection with Machine A (AP)
- Select option 2 and then option 1 (ADD) to configure the parameters.
- Enter the number of Station device connected or to be connected to AP. Also enter the order of STA (if requested by APP) on which QBV config is conducted. This is important for properly synching QBV window of each device.
- Select option to change any of the QBV config, else enter 0 (ZERO) to submit the config.
- **NOTE:** The Synchronized Guard band should be equal or greater than Machine A (AP) device total QBV Window

2.3 Enable System Time Disciplining

System time disciplining is a feature that can be enabled at the secondary side, which helps to keep the Secondary system clock synchronized with the primary system clock. On enabling this feature, the gPTP daemon in Secondary machine will set its system clock to closely track the Primary system clock. Any change in the primary system clock will follow with a corresponding change in Secondary system clock.

However, even after enabling the System Time Disciplining feature, there is still a Clock offset that can be seen at the Secondary system side, which is the Time Sync error and is of the order of tens of Microseconds to few milliseconds – depending on how tightly the two clocks are synchronized.

The System time disciplining feature is required to be enabled only on the Secondary system side. To enable this feature start phc2sys as below

```
$ sudo phc2sys -s /dev/ptp1 -O 0 -f /home/intel/phc2sys.cfg -m -i -L 0  
-R 8 -u 8 -I 0.25 | tee /usr/share/intel/wtsn/phc2sys2.log
```

Once the above changes are done, you can now see the Secondary system clock being disciplined by the phc2sys and in turn constantly syncing with the primary clock.

NOTES

Offset (System) – this is the Time Sync Error.

3.0 Demo 1: 802.1AS Time Synchronization Performance Measurement

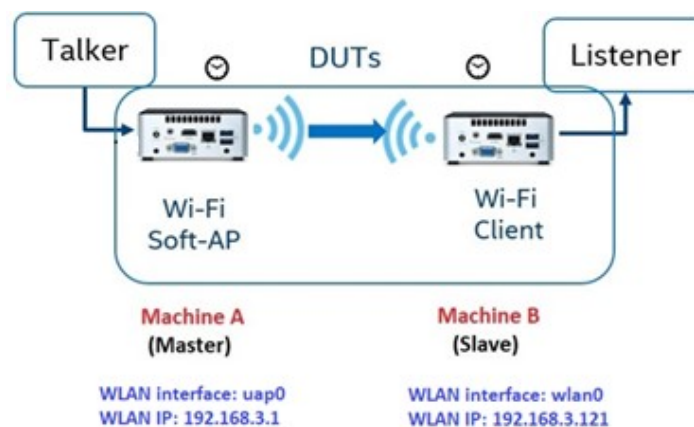
NOTES

Users can check the reference success logs for 802.1AS, in the folder location – 'wtsn/reference_success_logs/802.1AS'.

3.1 Hardware Set Up

This scenario is based on two Intel NUC devices configured to connect wireless as shown in below. 802.1AS time synchronization is run over the Intel Wireless Stack and Time synchronization Error is calculated by the estimated offset.

Figure 7: Hardware Set Up for Wireless TSN



3.2 Measure Time Synchronization Performance

1. **Establish wireless connectivity** using the pre-installed Intel WTSN Stack using the QBV App as noted in section [2.2.3.2.1](#).
2. **Verify the connectivity** via ping (refer section IP assignment) or using the following command.

Node	Command	Notes
------	---------	-------

Machine B	<pre>\$ cd /usr/sbin \$ sudo wpa_cli -i wlan0 status grep bssid</pre>	Starts node A beforehand as the Wireless AP
-----------	---	---

3. Start 802.1As time synchronization as noted in [2.2.3.3](#)

4. Before executes the 8021AS IPC Server, prepare and **build the 8021AS IPC Server**.

a) Install the required software packages.

```
$ cd <wtsn>/measurement/src/8021ASServer/
$ sudo apt-get install libevent-dev
$ sudo apt-get install curl
```

b) Build and install the 8021AS IPC Server.

```
$ cd <wtsn>/measurement/src/8021ASServer/
$ ./build.sh
```

```
andrew@Andrew-NUCi3:~/workspace/wtsn/measurement/src/8021ASServer$ ./build.sh
andrew@Andrew-NUCi3:~/workspace/wtsn/measurement/src/8021ASServer$ ls
build.sh  get_offset  get_offsets.cpp  get_offsets.hpp  ipcdef.hpp  linux_ipc.hpp  ptptypes.hpp  server.cpp
```

c) Copy 8021AS IPC Server to temp folder.

```
$ cd <wtsn>/measurement/src/8021ASServer/
$ mkdir ~/temp
$ cp ./get_offset ~/temp
```

5. **Start the 8021AS IPC Server**, which will query the offset from the shared memory of 802.1AS daemon and then display the offset. Logging this time offset data into file - timesync.dat for analysis.

Node	Command	Notes
Machine B	<pre># Open one new terminal \$ cd ~/temp \$ sudo ./get_offset</pre>	Starts 802.1AS IPC Server to provide the estimated offset.
Machine B	<pre># Open one new terminal \$ cd ~/temp \$ for i in {1..10000}; do curl http://localhost:9051 -s awk - F',' '{print \$3/1000}'; done tee timesync.dat</pre>	Queries the server for time data and save the result in timesync.dat

Demo 1: 802.1AS Time Synchronization Performance Measurement

NOTES

1. [1..10000] – user could decide how many of offset data need to analyze.
2. \$3/1000 – default unit is nanosecond, user could scale down to micro-second by divide 1000 if needed.

```
andrew@Andrew-NUCi3:~/temp$ sudo ./get_offset
[sudo] password for andrew:
DBG: 8021AS update thread launched ..

DBG: Offset: -2540
DBG: Offset: -5780
DBG: Offset: -7330
DBG: Offset: -7810
DBG: Offset: -9230
DBG: Offset: -8490
DBG: Offset: -11360
DBG: Offset: -11260
DBG: Offset: 5520
DBG: Offset: -5550
DBG: Offset: -4630
DBG: Offset: -7000
DBG: Offset: -8270
DBG: Offset: -7980

andrew@Andrew-NUCi3:~/temp$ for i in {1..10000}; do curl http://localhost:9051 -s | awk -F',' '{print $3/1000}'; done | tee timesync.dat
-127291
-127291
-127291
-127291
```

6. Plot the time offset results.

Node	Command	Notes
Machine B	<pre># Open one new terminal \$ cd ~/temp \$ sudo apt-get install gnuplot</pre>	Install the Gnuplot
Machine B	<pre>\$ cd ~/temp \$ cp <wtzn>/measurement/scripts/time_sync.plot . \$ gnuplot -persist time_sync.plot</pre>	<p>1. Please make sure the timesync.dat exists as show in step 5 above.</p> <p>2. Plots the histogram for synchronization error observed.</p>



Refer to the source of the time_sync.plot. You could adjust the x-axis range in xrange or set to autoscale by comment out the xrange.

NOTES

The plotted graph might look different on your test setup and environment.

Just the above point is sufficient. In fact, we are already seeing very optimized result in the order of -10 to +10 usec, which is excellent.

```

clear
reset
stats 'timesync.dat'
set key off
set border 3
# set xrange [-60:60]
set yzeroaxis
set title "WTSN Time Sync Error"

bin_width = 2
#set boxwidth 0.05 absolute
set boxwidth 0.5*bin_width
set style fill solid 0.75 noborder
set offset graph 0.05,0.05,0.05,0.0
set xlabel "Time Sync Error (us)"
set ylabel "Number of Samples"

bin_number(x) = floor(x/bin_width)

rounded(x) = bin_width * ( bin_number(x) + 0.5 )

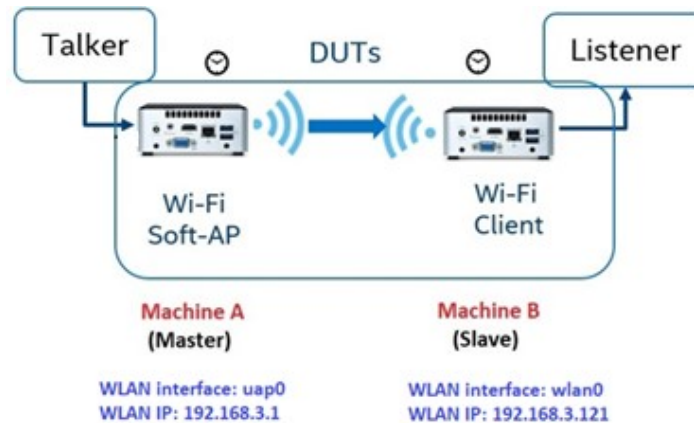
plot 'timesync.dat' using (rounded($1)):(1.0) smooth frequency with boxes
  
```

4.0 Demo 2: 802.1Qbv Time Aware Shaper

This section will describe a simple demo setup and demonstrate configuring IEEE Qbv on top of Wireless TSN stack and measure the performance.

4.1 Hardware Set Up

Figure 8: Hardware Set Up for Wireless TSN



4.2 Measure Time-Aware Traffic Scheduling

This demo measures the performance improvement archived by a single Time Sensitive (TSN) Stream in presence of increasing background traffic using time aware scheduling. The first thing to establish here for this demo or a representative measurement setup is the traffic profile of the TSN traffic as the TSN schedule will depend on the traffic profile.

Time-aware scheduling based on the 802.1Qbv standard enables the configuration of a time windows reserved exclusively for time sensitive traffic leveraging the time synchronization across the network enabled by 802.1AS. All other traffic is blocked during the protected windows, therefore avoiding congestion delay and losses. The TSN schedule will define the duration of the windows based on the traffic requirements and the link capabilities.

In this demo, we assume the TSN traffic profile to be as follows.

Traffic Tag	Packet Size	Rate/Periodicity	Proto	Notes
TSN Traffic 1	100 bytes	50 ms	UDP	
BE Traffic	1500 bytes	Variable 1Mbps to 100 Mbps	UDP	

4.2.1 Without the Time-Aware Traffic Scheduling

1. **Establish wireless connectivity and time synchronization** using the steps in section [2.2](#)
2. **Capture a baseline performance**

Refer section [2.4](#) for performing data transfer between Primary and Secondary devices and collect logs from server screen output.

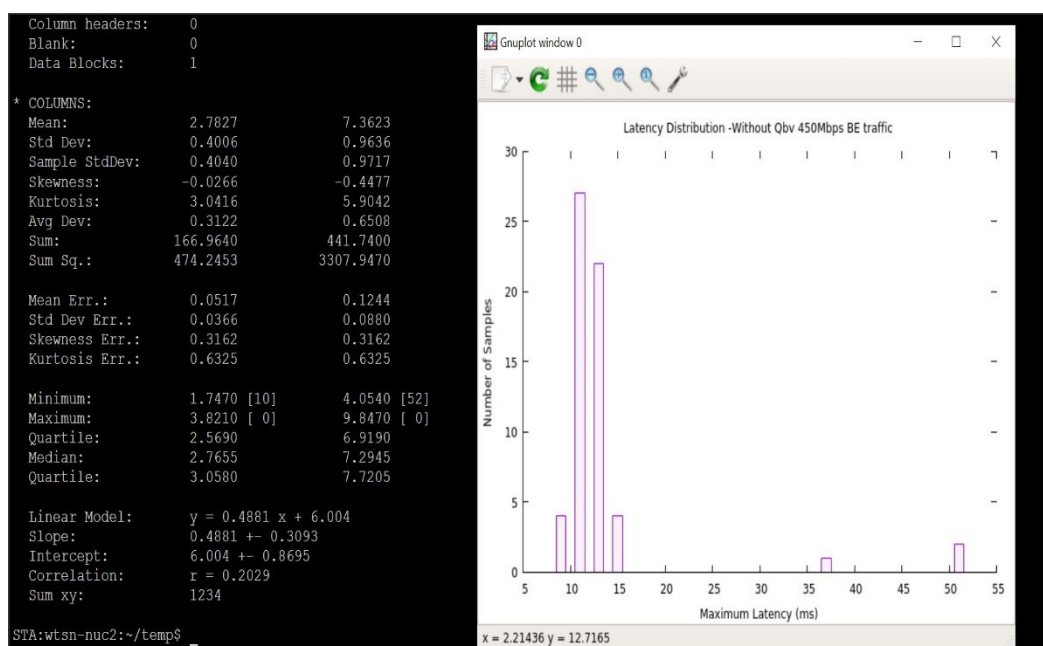
3. **Compile and aggregate all baseline data.**

Node	Command	Notes
Machine A	<pre>\$ cd ~/temp \$ for file in *.log; do csv='latencies_qbv.dat'; cat \$file grep 'pps' awk '{print \$9,\$14\$15\$16}' sed 's/\ /,/g' sed 's/\\/,/g' sed 's/ms//' sed 's/pps//g' >> \$csv; sed -i '/^\$/d' \$csv; done</pre>	Extract the latency data from all iPerf output files (result_qbv_XM.log) as CSV and combine for later parsing. This will create one single file with all latencies data in the format Jitter, average, min and max.

4. **Pilot data and visualize.**

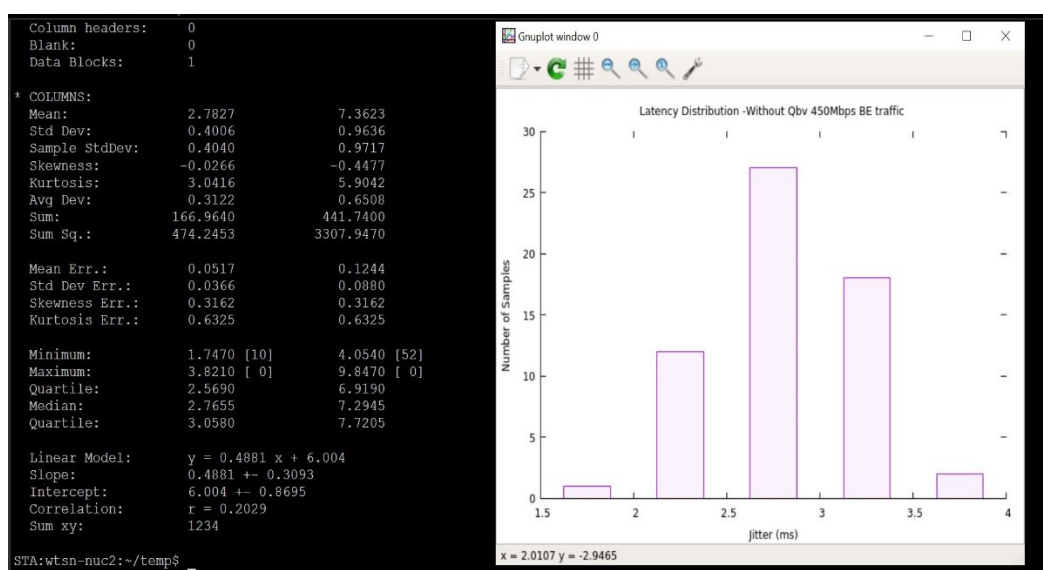
Node	Command	Notes
Machine A	<pre>\$ cd ~/temp \$ cp <wtsn>/measurement/scripts/qbv .plot . \$ gnuplot -p -e "filename='latencies_qbv.dat'; plot_title='Latency Distribution -Without Qbv'" qbv_avg_lat.plot \$ gnuplot -p -e "filename='latencies_qbv.dat'; plot_title='Latency Distribution -Without Qbv'" qbv_jitter.plot \$ gnuplot -p -e "filename='latencies_qbv.dat'; plot_title='Latency Distribution -Without Qbv'" qbv_max_lat.plot</pre>	Use Gnuplot with the supplied qbv_avg_lat.plot / qbv_jitter.plot / qbv_max_lat.plot file to plot the latency distribution.

Demo 2: 802.1Qbv Time Aware Shaper



NOTES

The plot should look different and up on test environment. In here, the major target is to validate the plot command and script if they could plot the latency observed during the testing, it may not be the optimize result.



4.2.2 With the Time-Aware Traffic Scheduling Enabled

1. **Establish wireless connectivity and time synchronization** using the steps in section [2.2](#).

2. **QBV Configuration,**

Configuring QBV can be performed using QBV App, please refer [Section 2.2.3.4](#)

3. **Compile and aggregate all Qbv data.**

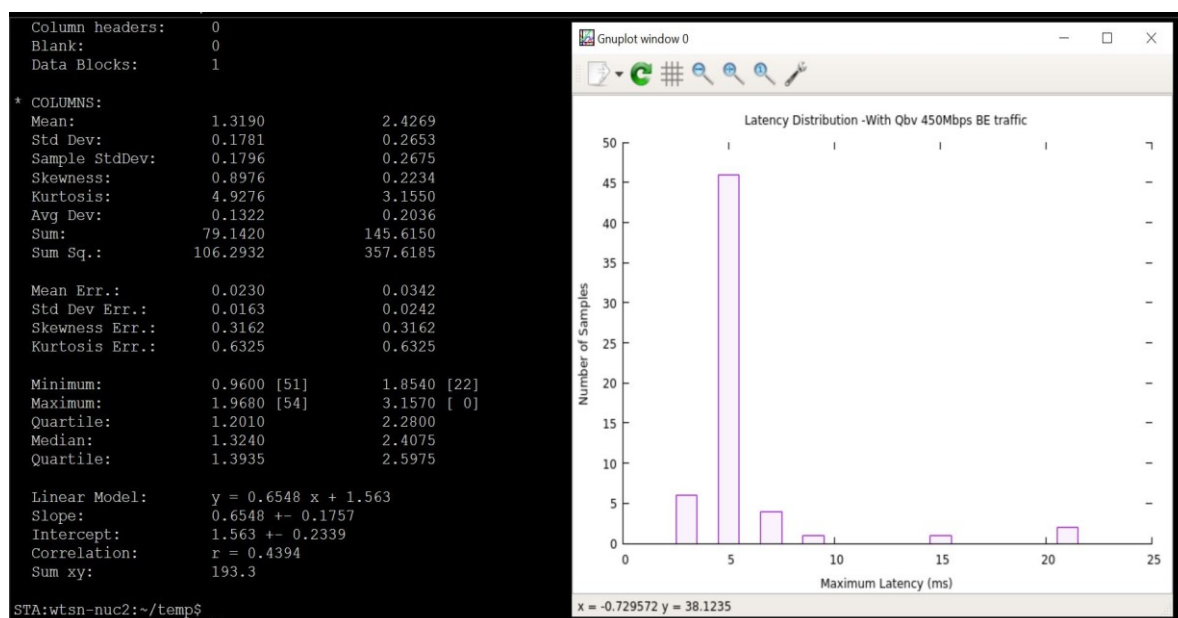
Refer section [2.4](#) for performing data transfer between Primary and Secondary devices and collect logs form server screen output.

Node	Command	Notes
Machine A	<pre>\$ cd ~/temp \$ for file in *.log; do csv='latencies_qbv.dat'; cat \$file grep 'pps' awk '{print \$9,\$14\$15\$16}' sed 's/\ /,/g' sed 's/\\/,/g' sed 's/ms//' sed 's/pps//g' >> \$csv; sed -i '/^\$/d' \$csv; done</pre>	Extract the latency data from all iPerf output files (result_qbv_XM.log) as CSV and combine for later parsing. This will create one single file with all latencies data in the format Jitter, average, min and max.

4. **Pilot data and visualize.**

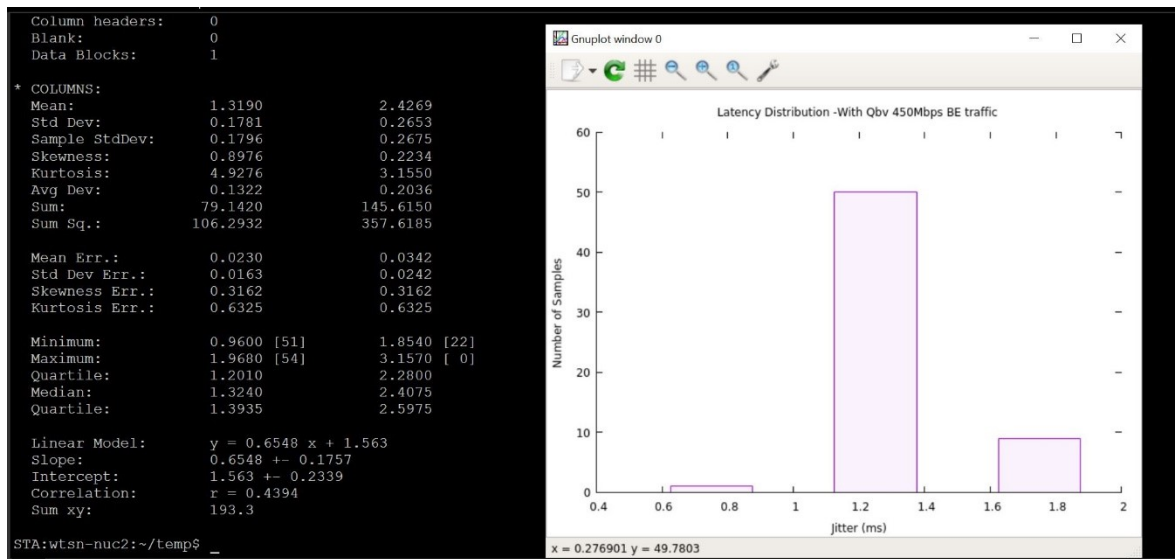
Node	Command	Notes
Machine A	<pre>\$ cd ~/temp \$ cp <wtsn>/measurement/scripts/qbv.p lot . \$ gnuplot -p -e "filename='latencies_qbv.dat';pl ot_title='Latency Distribution - With Qbv'" qbv_avg_lat.plot \$ gnuplot -p -e "filename='latencies_qbv.dat';pl ot_title='Latency Distribution - With Qbv'" qbv_jitter.plot \$ gnuplot -p -e "filename='latencies_qbv.dat';pl ot_title='Latency Distribution - With Qbv'" qbv_max_lat.plot</pre>	Use Gnuplot with the supplied qbv_avg_lat.plot / qbv_jitter.plot / qbv_max_lat.plot file to plot the latency distribution.

Demo 2: 802.1Qbv Time Aware Shaper



NOTES

The plot should look different and up on test environment. In here, the major target is to validate the plot command and script if they could plot the latency observed during the testing, it is not the optimized result.



NOTES

In the package, Intel also provided measure the latency which exclude the latency introduce by application. Please check session 5.6 to for more detail.


```
$ sudo journalctl -f -n 100 // you could also check hostapd log in journal
```

3. wpa_supplicant:

To check *wpa_supplicant* status, you could check if process is active and check logging in log file.

```
$ cd ~/usr/sbin
$ ps -aux | grep wpa_supplicant // check if wpa_supplicant is active
$ sudo cat ~/wpa_supplicant.log // wpa_supplicant log file is located at ~/
$ sudo journalctl -f -n 100 // you could also check wpa_supplicant log in journal
```

4. 802.1AS Daemon:

To check 802.1AS Daemon status, you could check if process is active and check logging in log file.

```
$ cd ~/usr/sbin
$ ps -aux | grep daemon_cl // check if 802.1AS daemon is active
$ sudo cat ~/M_gptp.log // gptp log file is located at ~/ and all logging in terminal console
```

You could adjust the log level of 802.1AS Daemon by below configurations.

```
$ sudo vim <wtsn>/gptp/common/gptp_log.hpp
```

```
#ifndef GTP_LOG_HPP
#define GTP_LOG_HPP

/**@file*/

#include <stdio.h>
#include <stdarg.h>
#include <time.h>

#ifdef GENIVI_DLT
#include "dlt.h"
#endif

#define GTP_LOG_CRITICAL_ON 1
#define GTP_LOG_ERROR_ON 1
#define GTP_LOG_EXCEPTION_ON 1

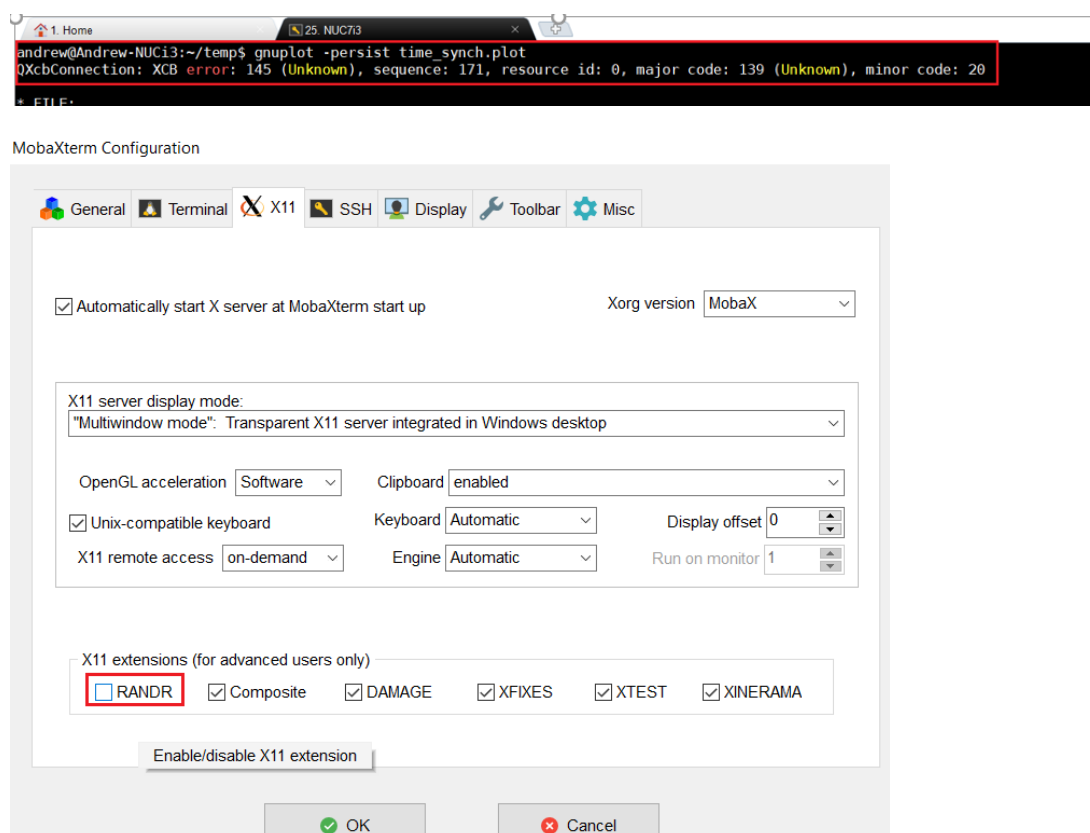
#define GTP_LOG_WARNING_ON 1
#define GTP_LOG_INFO_ON 1
#define GTP_LOG_STATUS_ON 1
#define GTP_LOG_DEBUG_ON 1
#define GTP_LOG_VERBOSE_ON 1

#if 0
#define GTP_LOG_WARNING_ON
#define GTP_LOG_INFO_ON
#define GTP_LOG_STATUS_ON
#define GTP_LOG_DEBUG_ON
#define GTP_LOG_VERBOSE_ON
#endif
```

5.3 Why do I get the error messages when do plot visualization - QXcbConnection: XCB error: 145 (Unknown) with MobaXTerm?

The error message always appears as following - QXcbConnection: XCB error: 145 (Unknown), sequence: 171, resource id: 0, major code: 139 (Unknown), minor code: 20 in MobaXTerm.

It is not cause by metrics and visualization scripts. Please disable the “RANDR” MobaXTerm X11 extension.



5.4 How to configure SSH public key Authentication to easier auto login?

Configure SSH public key authentication to easier auto login into machines. Public key authentication works as follows:

- Generate a keypair.
- Give a server the public key.

- Whenever you want to authenticate, the server asks you to prove you have the private key that corresponds to the public key.
- You need to prove that you have the private key.

1. Generate an SSH Key Pair in Machine A.

```
$ cd ~/.ssh
$ ssh-keygen // generate a key pair
$ cat id_rsa.pub // pass this public key string to Machine B
```

```
andrew@Andrew-NUCi5:~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCA3A+c4MugxWcPQgCbzXWwc97G9lQSBZB085HanR37J
jXG7ow65gplD55sk0xtM6Ty++twXjJ3Lv4p0BlyA1Im5Majpsuwm1/qSiPlQCcehqs0V0qaHnSjd/sn
qa19UIQFtZyQHAaEeSr6d0xBKwDc8f/smCc50nJcUDEp/NuZV4BwjlnkABBSLHqJ3ymnuDHvWqrwRcTE
0AKIb4u0cb/fEys+ImQqLTMYA6+wWUKLYrY68hkEEvYr0hyIws6PN70DrL5vwUD9Hb/LEYA0NcpUhnuy
RihIA6lfc2hePo8wKEjZGoSs2HqQAoX2fVWF4JCI4wkfLYXCBh1AwJ51S7D andrew@Andrew-NUCi5
```

2. Generate an SSH keypair in Machine B.

```
$ cd ~/.ssh
$ ssh-keygen // generate a key pair
$ cat id_rsa.pub //pass this public key string to Machine A
```

```
andrew@Andrew-NUCi3:~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQADNjhTnD0T09GVCZC7FARKxbaeFY8JLIiHa/f1InCyg
6o9FVm5fRYxe5z8r/mFYdaywptB49ZVb+A0Gr6+ME+dUk98IKvu4G4TnK8MVb1k1o70J3FLcpN2oC+0
W3ricluFi7UxFOIkpBrLqrneAi0XD0kEz6Ih+YFrF8FcQJog+dIrbf3AJRG0DVmh8g7WbJ4z+zYfwqv
r++tXkVcvCgZSWgaTpZI3ep2qpbCgzx4WsTaz2M6AYGLSuus5WFmB8dL88p9GPMh6+YUxwjv9RrNTRu
R29bFjgBvVqdUlb9byESNp0VPMr1LDIf9ZGjms075UfFcZJmgHHNCHrx17Z andrew@Andrew-NUCi3
```

3. Copy the own public key string to another side.

```
$ cd ~/.ssh
$ vim authorized_keys // paste the other side's public key string in
```

a) In Machine A, paste Machine B's public key in authorized_keys.

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQADNjhTnD0T09GVCZC7FARKxbaeFY8JLIiHa/f1InCyg
6o9FVm5fRYxe5z8r/mFYdaywptB49ZVb+A0Gr6+ME+dUk98IKvu4G4TnK8MVb1k1o70J3FLcpN2oC+0
W3ricluFi7UxFOIkpBrLqrneAi0XD0kEz6Ih+YFrF8FcQJog+dIrbf3AJRG0DVmh8g7WbJ4z+zYfwqv
r++tXkVcvCgZSWgaTpZI3ep2qpbCgzx4WsTaz2M6AYGLSuus5WFmB8dL88p9GPMh6+YUxwjv9RrNTRu
R29bFjgBvVqdUlb9byESNp0VPMr1LDIf9ZGjms075UfFcZJmgHHNCHrx17Z andrew@Andrew-NUCi3
```

b) In Machine B, paste Machine A's public key in authorized_keys.

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCA3A+c4MugxWcPQgCbzXWwc97G9lQSBZB085HanR37J
jXG7ow65gplD55sk0xtM6Ty++twXjJ3Lv4p0BlyA1Im5Majpsuwm1/qSiPlQCcehqs0V0qaHnSjd/sn
qa19UIQFtZyQHAaEeSr6d0xBKwDc8f/smCc50nJcUDEp/NuZV4BwjlnkABBSLHqJ3ymnuDHvWqrwRcTE
0AKIb4u0cb/fEys+ImQqLTMYA6+wWUKLYrY68hkEEvYr0hyIws6PN70DrL5vwUD9Hb/LEYA0NcpUhnuy
RihIA6lfc2hePo8wKEjZGoSs2HqQAoX2fVWF4JCI4wkfLYXCBh1AwJ51S7D andrew@Andrew-NUCi5
```

4. Log in to remote with keypairs.

```
$ ssh [account]@[IP_address]
```

```
andrew@Andrew-NUCi5:~/.ssh$ ssh andrew@10.0.0.15
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.20.0-042000-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage

* MicroK8s passes 9 million downloads. Thank you to all our contributors!

  https://microk8s.io/

* Canonical Livepatch is available for installation.
- Reduce system reboots and improve kernel security. Activate at:
  https://ubuntu.com/livepatch

25 packages can be updated.
0 updates are security updates.

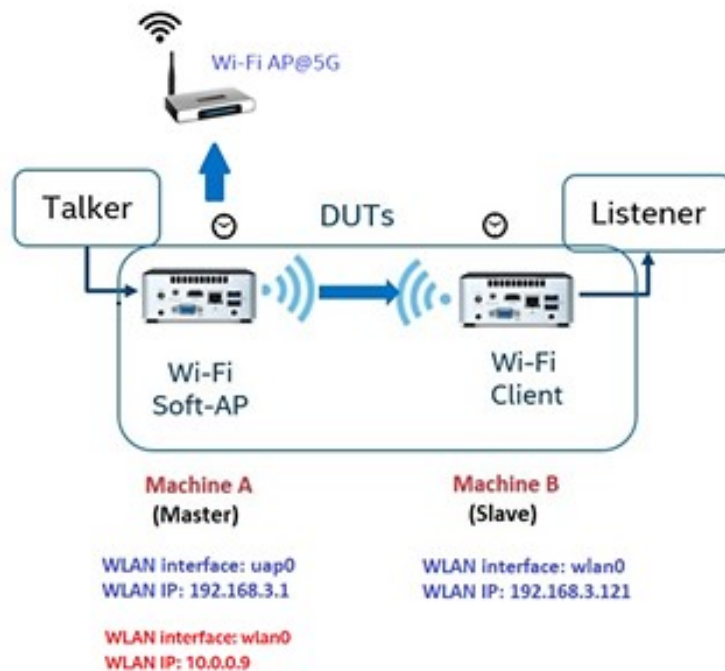
Last login: Sun May 31 19:44:38 2020 from 10.0.0.8
andrew@Andrew-NUCi3:~$
```

5.5 How to Enable Soft AP mode at 5GHz in WTSN?

Since the regulatory limitation at 5GHz and AX210 without radar detection, Intel wireless module will use 802.11d with DRS (Dynamic Regulatory Solution) to automatic apply regulatory profile by country code.

Thus, stand-alone soft AP will not come up by default due to DFS restriction. You need to enable STA role first to get approval in that specific 5GHz channel, then enable soft AP role at same channel.

Figure 9: Network topology of Soft AP at 5Ghz



- Station mode and connect to 5Ghz Wireless Access Point (physical Wi-Fi* AP).
- Get 5Ghz channel for Station mode.
- Enable soft AP in 5Ghz mode while Station is active.
- Disable Station mode. Soft AP will remain beaconing using 5Ghz channel.
- Verify the functionality from the WTSN-client

5.5.1 Configure wpa_supplicant.conf in Machine A for 5GHz

Configure *wpa_supplicant.conf*. Please change the *ssid* and *psk* settings matched your Wi-Fi* access point.

```
$ sudo vim /etc/wpa_supplicant/wpa_supplicant.conf
ctrl_interface=/var/run/wpa_supplicant

ctrl_interface_group=0
update_config=1

network={

    ssid="WIFI_5G_AP"           // your Wi-Fi* access point's SSID

    key_mgmt=WPA-PSK

    proto=RSN

    pairwise=CCMP TKIP
}
```

```
group=TKIP CCMP

psk="1234567890" // your Wi-Fi* access point password
}
```

5.5.2 Enable Station mode and connect to 5Ghz Wireless Access Point in Machine A.

Follow same instructions as per [2.2.3.2.2](#) in Machine A.

5.5.3 Check if Machine A is connected to 5GHz Wi-Fi* AP and which 5GHz channel

```
$ cd /usr/sbin
$ sudo iw dev
```

```
andrew@Andrew-NUCi5:/usr/sbin$ iw dev
phy#0
    Unnamed/non-netdev interface
        wdev 0x2
        addr d4:6d:6d:f3:e2:9d
        type P2P-device
        txpower 0.00 dBm
    Interface wlan0
        ifindex 4
        wdev 0x1
        addr d4:6d:6d:f3:e2:9c
        ssid Mosla32
        type managed
        channel 36 (5180 MHz), width: 80 MHz, center1: 5210 MHz
        txpower 22.00 dBm
```

5.5.4 Configure the hostapd.conf in Machine A for 5GHz soft AP

We use the same channel (frequency=5180, channel 36) to soft AP mode. Please enabled below highlighted lines in your hostapd.conf file.

```
$ sudo vim /etc/hostapd/hostapd.conf
ctrl_interface=/var/run/hostapd

ctrl_interface_group=0

tm_cap=1

interface=uap0

driver=nl80211

ssid=WTSN_Test_AP

channel=36 // 5GHz channel as same as your WIF access point used.
```

```
hw_mode=a           // 802.11ac for 5GHz

macaddr_acl=0

auth_algs=1

ignore_broadcast_ssid=0

ieee80211n=1

wpa=2

wpa_passphrase=12345678

wpa_key_mgmt=WPA-PSK

wpa_pairwise=TKIP CCMP

rsn_pairwise=CCMP


ht_capab=[HT40+][LDPC][SHORT-GI-20][SHORT-GI-40] // Enhance BW to 40M
vht_capab=[RXLDPC][SHORT-GI-80][SU-BEAMFORMEE][RX-STBC1][MAX-MPDU-
7991][MAX-A-MPDU-LEN-EXP7]
```

5.5.5 Enable 5GHz softAP mode while Station is active in Machine A

Follow same instructions as per [2.2.3.2.1](#) in Machine A

5.5.6 Check if we enable 5GHz soft AP successfully by hostapd_cli in Machine A

1. Check 5GHz soft AP status includes frequency and channel#.

```
$ cd /usr/sbin
$ sudo hostapd_cli -i uap0 status
$ sudo iw dev
```


Frequently Asked Questions (FAQ)

```
andrew@Andrew-NUCi5:/usr/sbin$ sudo hostapd_cli -i uap0 status
state=ENABLED
phy=phy0
freq=5180
num_sta_non_erp=0
num_sta_no_short_slot_time=0
num_sta_no_short_preamble=0
olbc=0
num_sta_ht_no_gf=0
num_sta_no_ht=0
num_sta_ht_20_mhz=0
num_sta_ht40_intolerant=0
olbc_ht=0
ht_op_mode=0x0
cac_time_seconds=0
cac_time_left_seconds=N/A
channel=36
secondary_channel=1
ieee80211n=1
ieee80211ac=1
ieee80211ax=0
beacon_int=100
dtim_period=3
vht_oper_chwidth=0
vht_oper_centr_freq_seg0_idx=0
vht_oper_centr_freq_seg1_idx=0
vht_caps_info=03801031
rx_vht_mcs_map=fffa
tx_vht_mcs_map=fffa
ht_caps_info=006f
ht_mcs_bitmask=ffff0000000000000000
supported_rates=0c 12 18 24 30 48 60 6c
max_txpower=22
bss[0]=uap0
bssid[0]=d4:6d:6d:f3:e2:9e
ssid[0]=WTSN_Test_AP
num_sta[0]=0
```

```
andrew@Andrew-NUCi5:/usr/sbin$ iw dev
phy#0
  Interface uap0
    ifindex 6
    wdev 0x3
    addr d4:6d:6d:f3:e2:9e
    ssid WTSN_Test_AP
    type AP
    channel 36 (5180 MHz), width: 40 MHz, center1: 5190 MHz
    txpower 22.00 dBm
  Unnamed/non-netdev interface
    wdev 0x2
    addr d4:6d:6d:f3:e2:9d
    type P2P-device
    txpower 0.00 dBm
  Interface wlan0
    ifindex 5
    wdev 0x1
    addr d4:6d:6d:f3:e2:9c
    ssid Mosla32
    type managed
    channel 36 (5180 MHz), width: 80 MHz, center1: 5210 MHz
    txpower 22.00 dBm
```

2. Now disable station mode on **Machine A**.

```
$ sudo ifconfig wlan0 down
$ sudo pkill -9 wpa_supplicant
```

5.5.7 Start station mode on Machine B and connect to 5GHz Soft AP on Machine A.

1. Start station mode on Machine B and connect to 5GHz soft AP by following same instructions as per [2.2.3.2.2](#) on Machine B
2. Check the network status.

```
$ cd /usr/sbin
$ sudo wpa_cli -i wlan0 status
$ sudo iw dev
```

```
andrew@Andrew-NUCi3:/usr/sbin$ sudo wpa_cli -i wlan0 status
bssid=d4:6d:6d:f3:e2:9e
freq=5180
ssid=WTSN_Test_AP
id=0
mode=station
wifi_generation=5
pairwise_cipher=CCMP
group_cipher=CCMP
key_mgmt=WPA2-PSK
wpa_state=COMPLETED
ip_address=192.168.3.121
p2p_device_address=00:e1:8c:79:58:66
address=00:e1:8c:79:58:65
uuid=4033aa72-9b0e-57ed-b1ad-c2f405279d6d
wmm_assoc=1
ieee80211ac=1
```

```
andrew@Andrew-NUCi3:/usr/sbin$ iw dev
phy#0
    Unnamed/non-netdev interface
        wdev 0x2
        addr 00:e1:8c:79:58:66
        type P2P-device
        txpower 0.00 dBm
    Interface wlan0
        ifindex 4
        wdev 0x1
        addr 00:e1:8c:79:58:65
        ssid WTSN_Test_AP
        type managed
        channel 36 (5180 MHz), width: 40 MHz, center1: 5190 MHz
        txpower 22.00 dBm
```

5.6 How to measure the end-to-end latency without include application latency.

Instead of get the latency result by iperf2. Intel provided the advanced tool which allow you to measure the end-to-end latency without include application latency. This allows you have the latency baseline and helping you to fine tune the application for having lower latency.

In the following, we will guide you how to enable and utilized the tool we provided in the package.

5.6.1 What is included in the measurement tools?

The latency measurement tool located at "measurement/src/LatencyMeasurementTool" in the package.

There are two shell scripts and 1 python script. The shell script. "run_collection_client.sh" and "run_collection_server.sh" these example can help you to start iperf real-time traffic, capture the packets and convert to CSV format for analysis. The python script "parsing_data.py" is the script which can help you to have the histogram and statistics result. Please also note that the tool only support measure the UDP traffic at this point.

5.6.2 Install the necessary component for measurement tool

Instead iperf2 already mentioned in the previous session, There are several tools is required for running the tool tcpdump, tskark, python3. You can install them via apt-get command:

```
$ sudo apt-get install tshark
$ sudo apt-get install tcpdump
$ sudo apt-get install python3
$ sudo apt-get install python3-pip
```

In order to run the python tools in the package, there are several extra python module need to install in advanced.

```
$ python3 -m pip install numpy
$ python3 -m pip install pandas
$ python3 -m pip install matplotlib
```

If you would like to run the test when enable Qbv, you need to modified `run_collection_client.sh` to add TC setting base on your Qbv configuration. Following is an example. It classified the real-time traffic packet to use socket priority 6 and define the traffic gate open time as 50 us.

```
sudo iptables -t mangle -F
sudo tc qdisc del dev $INTERFACE_NAME \
    parent root

sudo iptables -t mangle -A POSTROUTING -p udp --dport 5010 -j
CLASSIFY --set-class 0:6
sudo iptables -t mangle -A POSTROUTING -p udp --dport 5010 -j DSCP --
set-dscp-class CS6

sudo tc -d qdisc replace dev $INTERFACE_NAME \
    parent root \
    handle 100 \
    taprio \
    num_tc 2 \
    map 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 \
    queues 1@0 1@1 \
    base-time $BASE_TIME \
    sched-entry S 01 50000 \
    sched-entry S 02 100000 \
    sched-entry S 00 50000 \
    clockid CLOCK_REALTIME
```

Please also making sure that the `iperf` command also add the “`--txstart-time`” parameter which base on the same “`$BASE_TIME`” you defined in the TC command. Following is an example :

```
sudo iperf -c $DST_IP_ADDR -B $HOST_IP_ADDR -u -e -z -l 20 -b 25pps
-i1 -t30 -p 5010 --txstart-time $BASE_TIME
```

5.6.3 Running the tool and having measurement result

It is very straight forward to run the measurement tool. You can follow steps below to running the measurement.

1.) run the shell script at one of the NUC which received traffic.

```
$ run_collection_server.sh
```

2.) run the shell script at one of the NUC which sending traffic.

```
$ run_collection_client.sh
```

3.) You will get two CSV data named “**net_logs_client.txt**” and “**net_logs_server.txt**”. Copy the CSV data from NUC which received traffic the to another NUC and put these two CSV file at the same folder. Run the

```
$ parsing_data.py
```

§

6.0 *Learn More*

6.1 IEEE 1588 and IEEE 802.1AS-2011

TSN has two important components:

- Time synchronization
- Traffic shaping

For time synchronization, Ethernet controllers shall have IEEE 1588 PTP clock and Ethernet frame receive and transmit time-stamping capability.

IEEE 1588-2008, also known as Precision Time Protocol Version 2 (PTPv2), enhances the accuracy of time synchronization between two networked nodes from millisecond (achievable by Network Time Protocol (NTP)) to microsecond or sub microsecond. This is possible as packet time-stamping is done at the hardware, instead of software, level. The transport of PTP messages can be over UDP/IPv4, UDP/IPv6, IEEE802.3 Ethernet, or IEEE802.11 Wireless.

IEEE 802.1AS-2011, also known as generalized Precision Time Protocol (gPTP), is based on IEEE 1588-2008 and, being an 802.1 standard, can be applied to a wide range of heterogeneous networks, such as Ethernet, Wireless, Media over Coax Alliance and Home Plug. The primary components of gPTP are:

- **Best primary clock selection:** Uses a version of the IEEE 1588 "best primary clock algorithm" but with improvements:
 - All grandmaster (GM) capable devices announce their capabilities to all their time domain neighbors.
 - Only the "best" capability is retransmitted by bridges/switches to all their neighbors.
 - A GM-capable device that receives "better" capability will stop transmitting its capability announcement.
 - Finally, only the best GM-capable device will continue sending capability announcements and be the source of grandmaster clock to all devices within the time domain.
- **Path delay measurement:** The path delay between connected devices over bridges/switches is a slowly varying value due to physical condition such as temperature of network cables. gPTP device measures the path delay by exchanging Pdelay request, Pdelay response, and Pdelay response follow-up messages between the initiator and responder as shown in the figure below. In fact, all devices including the device that has grandmaster clock can be the path delay initiator.

- Time distribution:** The device with the grandmaster clock periodically sends the SYNC packet that contains time of the day (TAI clock) along with a timestamp of when the SYNC message was actually sent. gPTP specifies the use of IEEE 1588 two-step processing where the said timestamp value is sent on a subsequent message called "Follow Up" message. Through SYNC and Follow-Up messages and with its known path delay, the device with the [subordinate clock](#) will constantly adjust the PTP clock to keep synchronized with the time of the grandmaster clock.

To measure the quality of time synchronization between subordinate clock and grandmaster clock, there are several methods recommended by Avnu Alliance in its publication "802.1AS Recovered Clock Quality Testing Revision 1.0" dated 18 October 2016. See:

To measure the quality of time synchronization between subordinate clock and grandmaster clock, there are several methods recommended by Avnu Alliance in its publication "802.1AS Recovered Clock Quality Testing Revision 1.0" dated 18 October 2016. See: http://avnu.org/wp-content/uploads/2014/05/Avnu-Testability-802.1ASRecovered-Clock-Quality-Measurement-1.0_Approved-for-Public-Release.pdf and the corresponding:

- Chapter 4.2 of this publication specifies Time Error = Time (measured on DUT) - Time (reported at reference).
- Chapter 5.1 explains the 1PPS method of observing the rise time of a signal transmitted out from the subordinate clock and the rise time of a signal transmitted out from the grandmaster clock in lab oscilloscope.

Figure 10: gPTP in Action between Grandmaster and Subordinate Clocks

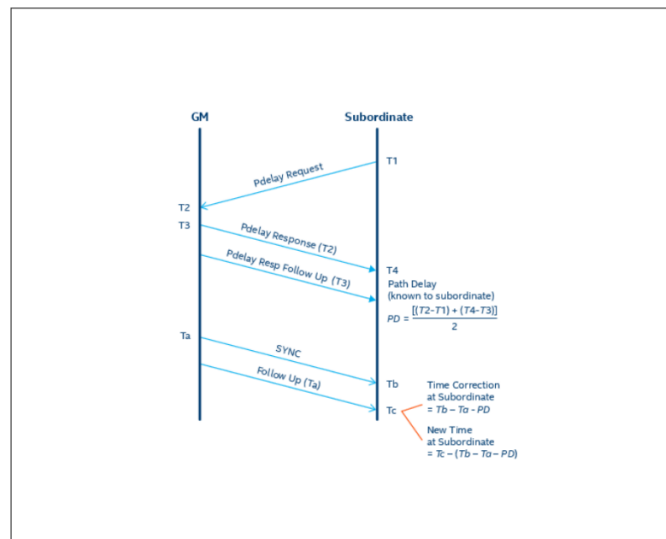


Figure 11: Time Synchronization Demo: Software / Application in Machine A

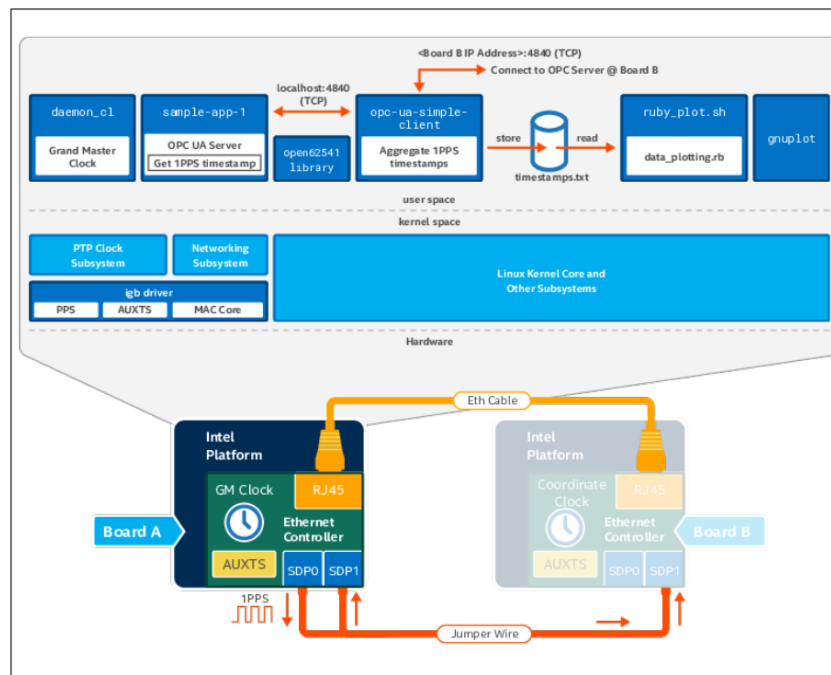
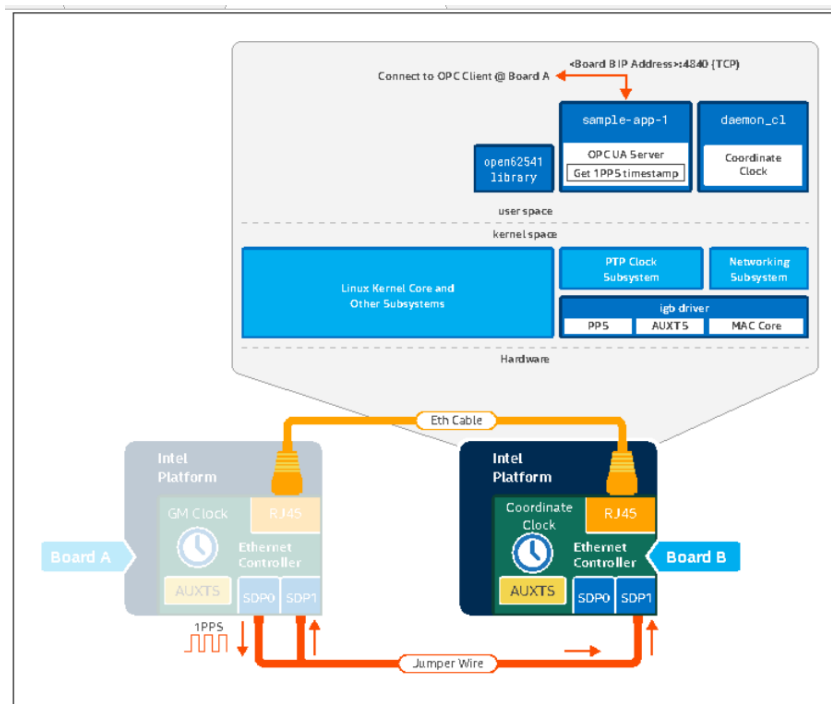


Figure 10: Time Synchronization Demo: Software / Application in Machine B



6.2 IEEE 802.1Qbv

Periodic control applications in automotive and industrial networks require much lower and bounded latencies compared to AV applications. In IEEE 802.1Qbv terminology, this type of traffic pattern is known as *scheduled traffic*. In contrast, conventional best-effort Ethernet networking does not guarantee low transmission latency and timely delivery. Other interfering traffic can affect critical control data used in industrial automation (TSN traffic). Since control data is usually short in frame length and periodically, the bandwidth used by control data is low. Therefore, it is possible to share the same medium with other traffic in the same network for better bandwidth utilization.

To identify and segregate different types of Ethernet traffic, IEEE 802.1Q introduces VLAN header (contains VLAN ID and VLAN priority) to mark different types of Ethernet frames. By using VLAN priority, Ethernet frames can be queued into different transmit queues, known also traffic class transmit queues. As a technology extension to IEEE 802.1Q, IEEE 802.1Qbv describes Time Aware Shaper (TAS) which has time-controlled transmission gates, which are associated with the above-mentioned traffic class transmit queues. TAS uses time from the PTP clock in Ethernet MAC controller. As a result, we use time synchronization technology (IEEE 1588 or IEEE 802.1AS) to synchronize PTP clocks in all networked appliances across the network. In addition to time synchronization, it is important to make sure all the network appliances have a well-coordinated TAS transmission schedule so that end-to-end scheduled traffic transmission achieves a very small and tightly bounded transmission latency.

The figure below shows the components in IEEE 802.1Qbv TAS (marked in blue).

The transmission schedule (also known as transmission windows) is programmed by using a gate control list (GCL). The GCL is a list of gate control entries (gate command, gate open/close state, interval in nanosecond). The open/close state of the gate is coded in bits: 1 means open and 0 means closed. For example, a value of 0x35 (0011 0101) means TxQ0, TxQ2, TxQ4 & TxQ5 are open. As defined in IEEE 802.1Qbv, there is only one gate command, which is *SetGates*. The execution of the GCL starts at base time, repeats itself after a duration of cycle time has lapsed. The cycle extension time is useful to ensure a smooth transition from the old GCL to the new GCL.

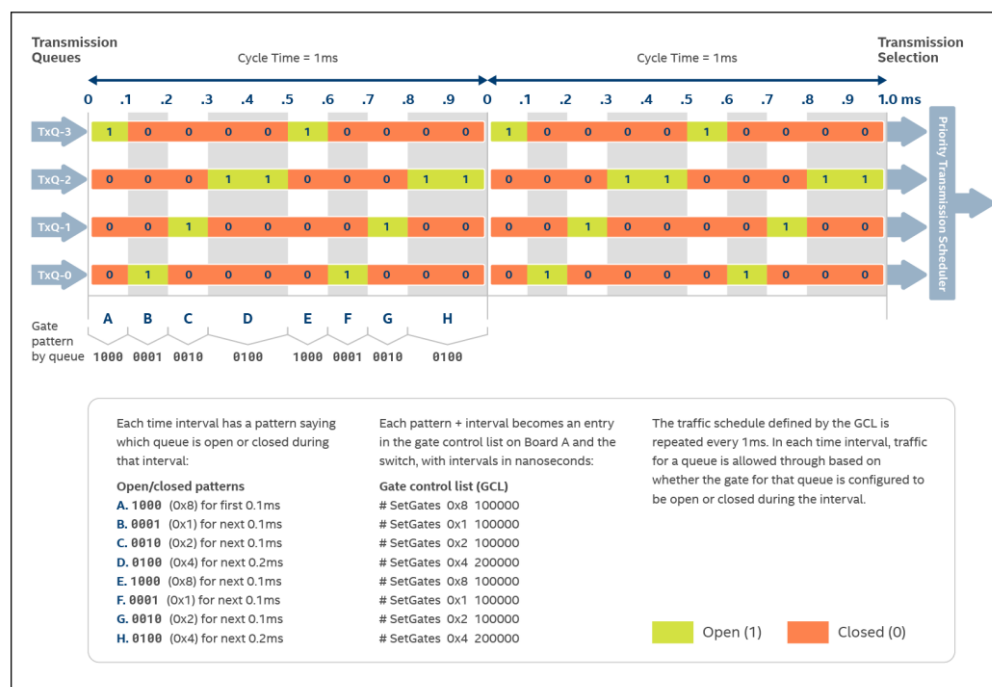
By programming the TAS GCL and setting its associated time-related parameters, we define transmission windows for various types of Ethernet traffic. Taking scheduled traffic (VLAN priority = 7) as an example, Ethernet frames for scheduled traffic are expected to be transmitted within the two green transmission windows:

- (Base Time + N x Cycle Time +0.1) ms to (Base Time + N x Cycle Time +0.2) ms
- (Base Time + N x Cycle Time +0.6) ms to (Base Time + N x Cycle Time +0.7) ms

NOTES

The transmission windows for other scheduled traffic (VLAN priority = 5), marked as blue, PTP frames (marked as red) and best effort traffic (marked as gray) are closed when the transmission window of scheduled traffic (VLAN priority = 7) is open. As a result, the transmission window for the scheduled traffic is protected.

Figure 40: Time Aware Shaper (in IEEE 802.1Qbv)



One important characteristic of TAS is that a frame is not selected for transmission unless adequate transmission gate open time is available to ensure an entire frame is transmitted. As a result, network administrators do not need to set up guard bands in the transmission schedule to prevent interfering frames from further delaying the transmission of scheduled traffic. Use of guard bands was a common technique before TAS since network transmission is not allowed within the guard band. With TAS, setting up a guard band is no longer needed as there is no unnecessary loss of network bandwidth.

Figure 41: TAS Transmit Schedule

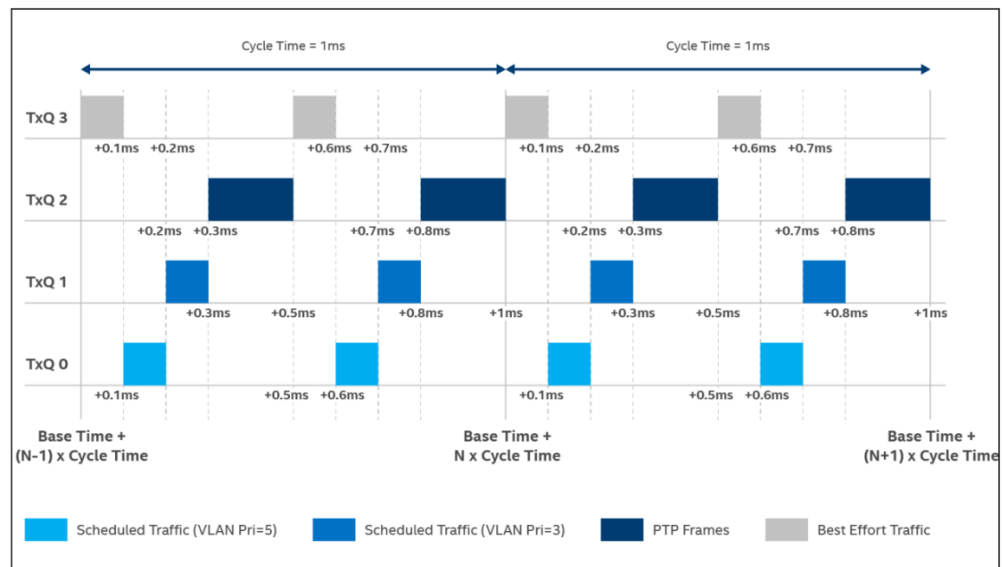
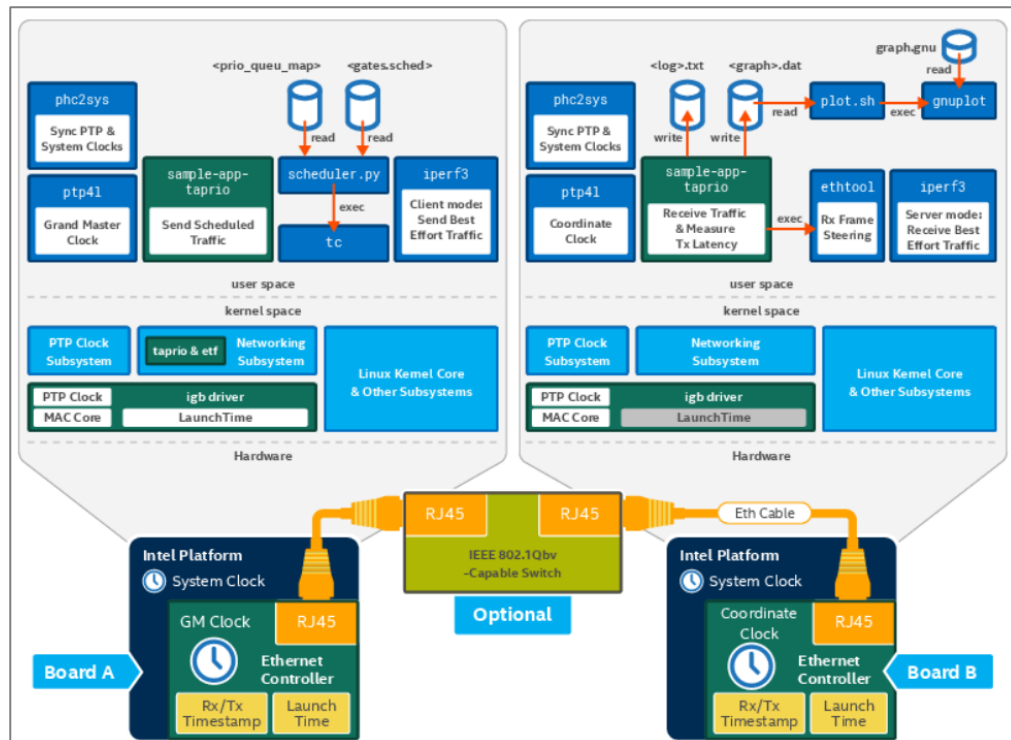


Figure 42: IEEE 802.1Qbv Time Aware Shaper Software Components



6.3 Queue Disciplines

Queueing discipline, or qdisc, is used when the kernel needs to send a packet to an interface. It is enqueued to the qdisc configured for that interface. Immediately afterward, the kernel tries to get as many packets as possible from the qdisc to the network adapter driver.

A simple QDISC is 'pfifo', which does no processing at all and is a pure first-in, first-out queue. It does however store traffic when the network interface cannot handle it.

TSN uses the following qdiscs:

QDISC	Description
CBS	<p>This is a simple rate-limiting shaper aimed at TSN applications on systems with known traffic workloads. Its primary use is to apply bandwidth reservation to user-defined traffic classes, which are mapped to different queues via the mqprio qdisc.</p> <p>Refer to CBS documentation to learn more about this qdisc.</p>
MQPRIO	<p>The MQPRIO qdisc is a simple queuing discipline that allows mapping traffic flows to hardware queue ranges using priorities and a configurable priority to traffic class mapping.</p> <p>Refer to MQPRIO documentation to learn more about this qdisc.</p>
TAPRIO	<p>This scheduler allows the network admin to configure schedules for classes of traffic. This qdisc borrows a few concepts from mqprio and most of the parameters are like mqprio.</p> <p>Refer to TAPRIO documentation to learn more about this qdisc.</p>
ETF	<p>The ETF qdisc is used to schedule traffic transmission based on absolute time. For some workloads, just bandwidth enforcement is not enough and precise control of the transmission of packets is necessary.</p> <p>Refer to ETF documentation to learn more about this qdisc.</p>

6.4 Wireless TSN Application Classes

Target use cases with Wireless TSN software stack focus on Class A and Class B.

	Class A	Class B	Class C
Applications	High quality AV, soft-real-time control, mobile robotics, Automated Guided Vehicles (AGV)	AR/VR, Professional AV, gaming, HMI, hard-real-time cyclic control, machine tools, production lines	Hard-real-time isochronous control, motion control, printing, packaging
Time sync	10-1 μ s	\sim 1 μ s	\sim 1 μ s
Latency	50 - 10 <u>ms</u>	10 - 1 <u>ms</u>	1ms - 250 μ s
Reliability	99% - 99.9%	99.9% - 99.99%	>99.999%

References:

S. Ashraf et al., "Ultra-Reliable and Low-Latency Communication for Wireless Factory Automation: From LTE to 5G," *IEEE*, 2016
 Industrial Communication Protocols, [Springer Handbook of Automation](#)
 Consolidated Use cases for the Tactile Internet, IEEE P1918.1 working group

Class of Service*	Class A	Class B	Class C
Latency	50 - 10 <u>msec</u>	10 - 1 <u>msec</u>	1 <u>msec</u> - ...
Reliability	99% - 99.9%	99.9% - 99.99%	99.999% - ...
Application Examples	High quality AV, soft-real-time control, mobile robots, AGVs	AR/VR, Professional AV, gaming, HMI, hard-real-time cyclic control	Hard-real-time isochronous control, motion control
Potential 802.11 Roadmap	<div> <div>Phase 1 (802.11ac)</div> <div>Phase 2 (802.11ax)</div> <div>Phase 3 (EHT)</div> </div>		
802.11 TSN Capabilities	<div>Time synchronization (802.1AS over 802.11) - DONE</div> <div> <div>Admission control, latency-optimized scheduling, Time-Aware shaping (<u>Qbv</u> over 802.11), redundancy (FRE) Trigger-based access (11ax), 6 GHz (11ax)</div> <div>Scheduled access, Low overhead PHY, frame-preemption, Multi-AP ...</div> </div>		
Standardization activities	<div>WFA, IEEE 802.11ax</div> <div>RTA-TIG, EHT</div>		

6.5 Glossary

Term	Definition
Best Primary Clock Algorithm	<p>A key to the resiliency of the Precision Time Protocol (PTP) is the BMCA. A new grandmaster clock is selected automatically when the previous grandmaster gets disconnected from the network. The selection criteria of the Best Primary Clock, in order of precedence, is listed below:</p> <ol style="list-style-type: none"> 1. Priority1: This is an integer from the range of 0 to 255. The smallest number wins. For subordinate only devices, this is set to 255. 2. Clock class: This denotes the traceability of the synchronous time distributed by a ClockMaster when it is grandmaster. 3. Clock accuracy: This is an enumerated list of ranges of time accuracy. 4. Offset scaled log variance: This field characterizes the precision and frequency stability of the ClockMaster. 5. Priority2: This field, together with the Priority1 field, is a user settable field. This allows system integrators to identify primary and backup clocks among identical redundant grandmasters. 6. Clock identity: This is usually set to the Wi-Fi* MAC address.
Clock Synchronization	No two oscillators can run at the same frequency precisely. As a result, the clocks tend to drift away from each other after running for some time. Clock synchronization is a general term that includes syntonization and synchronization.
gPTP Daemon	<p>OpenAvnu provides an example implementation of Precision Time Protocol (gPTP) daemon, called <code>daemon_cl</code>, to establish high accuracy clock synchronization among network nodes via IEEE standard 802.1AS-2011, also known as generalized gPTP.</p> <p>The daemon also implements Best Primary Clock Algorithm (BMCA), which allows the clocks to automatically select the best primary clock among them. Network interface (for example, <code>wlan0</code>) is the only required parameter to <code>daemon_cl</code>. Other useful parameters are <code>-S</code> and <code>-R</code>. <code>-S</code> performs clock syntonization, besides time synchronization. <code>-R</code> is used to change the priority1 value and can be used to set the endpoint as grandmaster by setting a lower value.</p>
Iproute2 TC	TC is utility/command used to configure Traffic Control in the Linux* kernel.
Open Platform Communications Unified Architecture (OPC UA)	OPC UA provides an extensible framework for a machine-to-machine communication protocol for industrial automation with an extensible framework that is open, cross-platform, cross-operating system and based on a service-oriented architecture.
Preempt RT	<p>The real-time (also known as PREEMPT_RT) patch for making Linux* into a true real-time operating system (RTOS).</p> <p>It was first introduced in 2004, and since then multiple attempts have been made to fully integrate PREEMPT_RT patch into Linux* mainline.</p>
ptp4l Daemon	Another commonly used PTP daemon is <code>ptp4l</code> from The Linux* PTP Project, which implements PTP according to IEEE standard 1588 for Linux*. Same as gPTP daemon, the <code>-I</code> switch together with the network interface (for example, <code>wlan0</code>) is the only

Learn More

	required parameter to ptp4l. Other useful parameters are -f to select configuration file, -m to print messages on the display and -s to select subordinate-only mode.
qdiscs	Queueing discipline, or qdisc, is used when the kernel needs to send a packet to an interface.
Synchronization	Synchronization is generally referred to time synchronization. Two clocks are said to be synchronized when they both agree precisely on the time of the day.
Syntonization	Syntonization is also known as frequency synchronization. Two clocks are said to be syntonized when the measured time passing between them is precisely at the same frequency.
Time slice (or quantum)	In a general-purpose operating system (GPOS), round-robin (RR) time-sharing process scheduling allows each process to use the processor for a short period of processing time, known as a time slice or quantum.

6.6 Terminology

Term	Description
AP	Access Point
API	Application Program Interface
AVB	Audio Video Bridge
BE	Best Effort
BIOS	Basic Input / Output System
BKC	Best Known Configuration
BSP	Board Support Package
CBS	Credit Based Shaper
CPU	Central Process Unit
CRB	Customer Reference Board
EEE	Energy-Efficient Ethernet
ETF	Earliest TxTime First
FPE	Floating Point Exception
GB	Gigabyte
GbE	Gigabit Ethernet
gPTP	Generalized Precision Time Protocol
GUI	Graphical User Interface
IAFW	Intel Architecture Firmware
I/O	Input / Output
LAN	Local Area Network
LTS	Long Term Support
M2M	Machine-to-Machine
MR	Milestone Release
MQPRIO	Multi-Queue Priority
Intel® NUC	Intel® Next Unit of Computing
NIC	Network Interface Card
NTP	Network Time Protocol
OPC UA	Open Platform Communications Unified Architecture
OS	Operating System
PHC	PTP Hardware Clock
POSIX	Portable Operating System Interface
PPS	Pulse-Per-second
PTP	Precision Time Protocol
QDISC	Queuing Disciplines

Learn More

RAM	Random Access Memory
SDK	Software Development Kit
SDP	Software Defined Pin
SOC	System-on-a-Chip
SRP	Stream Reservation Protocol
SSH	Secure Shell
TAI	French “Temps atomique international” which also means International Atomic Time.
TAS	Time Aware Shaper
TAPRIO	Time Aware Priority Shaper
TBS	Time Based Scheduling
TC	Traffic Control
TCC	Time Coordinated Computing
TSN	Time Sensitive Networking
TSO	TCP Segmentation Offloading
ThP2	Typhoon Peak 2
UDP	User Datagram Protocol
USB	Universal Serial Bus
UTC	Coordinated Universal Time
WTSN	Wireless Time Sensitive Network