

RZ/G3L Group

RZ/G3L TSN Integration and Communication

Introduction

Time-sensitive Network (hereinafter referred to as “TSN”) is a set of networking technologies designed to enable real-time communication over Ethernet. TSN extends traditional Ethernet to enable time-sensitive communication for applications requiring predictable network behavior.

This document provides a guide to prepare test environment of the TSN package on the evaluation boards with the RZ Linux BSP Plus for Kernel 6.12.

This guide provides the following information:

- Building procedure
- Preparation for the test environment for each industrial network

Target Reference Board

- RZ/G3L SMARC EVK

Target Software

- RZ Linux BSP Plus for Kernel 6.12
URL: https://renesas-rz.github.io/rz_linux_bsp_plus/
- TSN
TSN is a set of networking technologies that includes the following protocols and technologies:

Table 1. TSN List

Stack Name	OSS	Version	License	Reference URL
PTP	LinuxPTP	4.1	GPL-2.0	https://sourceforge.net/projects/linuxptp/files/v4.1/
TDMA Driver	iproute2	6.7.0	GPL-2.0	https://github.com/iproute2/iproute2/tree/v6.7.0
LLDP	lldpd	1.0.18	ISC	https://github.com/lldpd/lldpd/tree/1.0.18
YANG	libyang	2.1.148	BSD-3-Clause	https://github.com/CESNET/libyang/tree/v2.1.148
NETCONF	libnetconf2	3.0.8	BSD-3-Clause	https://github.com/CESNET/libnetconf2/tree/v3.0.8
EAP	Freeradius	3.2.6	GPL-2.0 license	https://github.com/FreeRADIUS/freeradius-server/tree/release_3_2_6

Contents

1. Environmental Requirement.....	3
2. Build Instructions.....	5
3. TSN.....	7
3.1 gPTP (IEEE 802.1AS)	7
3.1.1 How to prepare the environment.....	7
3.1.2 How to use gPTP.....	8
3.1.3 Explanation of the parameter in configuration file.....	10
3.2 PTP (IEEE 1588).....	12
3.2.1 How to prepare the environment.....	12
3.2.2 How to use PTP.....	13
3.3 TDMA driver (IEEE 802.1Qbv)	16
3.3.1 How to prepare the environment.....	16
3.3.2 How to use 802.1Qbv for GBETH	17
3.4 TDMA driver (IEEE 802.1Qbu).....	20
3.4.1 How to prepare the environment.....	20
3.4.2 How to use 802.1Qbu for GBETH	20
3.5 LLDP.....	25
3.5.1 How to prepare the environment.....	25
3.5.2 How to use LLDP.....	25
3.6 YANG/NETCONF	27
3.6.1 How to prepare the environment.....	27
3.6.2 How to use YANG/NETCONF	28
3.7 EAP.....	30
3.7.1 How to prepare the environment.....	30
3.7.2 How to use EAP.....	30
4. Appendix.....	35
4.1 Ethernet port.....	35
5. Revision History	36
Website and Support.....	37

1. Environmental Requirement

The environment for building the Board Support Package (hereinafter referred to as “BSP”) is listed in Table 2. Please refer to the documents below for details about setting up the environment:

Figure 1 shows the recommended environment for this package.

A Linux PC is required for building the software.

A Windows PC can be used as a serial terminal interface with software such as TeraTerm.

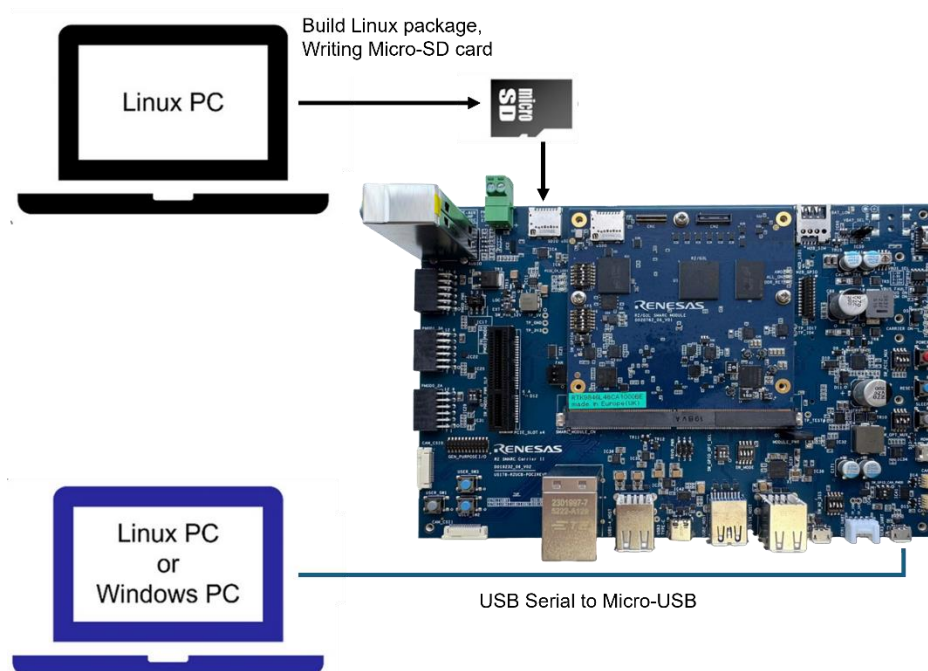


Figure 1. Recommend Environment

Table 2. Equipment and Software for Developing Environments of Linux Platform

Equipment		Description
Linux Host PC		Used as build/debug environment 100GB free space on HDD or SSD is necessary
	OS	Ubuntu 22.04 LTS 64 bit OS must be used. 20.04 inside a docker container also OK.
Windows Host PC		Used as debug environment, controlling with terminal software
	OS	Windows 11
	Terminal software	Used for controlling serial console of the target board Tera Term (latest version) is recommended Available at https://github.com/TeraTermProject/teraterm/releases
	VCP Driver	Virtual COM Port driver which enables to communicate Windows Host PC and the target board via USB which is virtually used as serial port. Available at: http://www.ftdichip.com/Drivers/VCP.htm
USB serial to micro-USB Cable		Serial communication (UART) between the Evaluation Board Kit and Windows PC. The type of USB serial connector on the Evaluation Board Kit is micro USB type B.
micro-SD Card		Use to boot the system, and store applications.

2. Build Instructions

This section describes the instructions to build and enable TSN. **Copy the TSN package into your Linux Host PC prior to the steps below.**

(1) Build RZ Linux BSP plus for Kernel 6.12

Read the following site and build RZ Linux BSP Plus for Kernel 6.12. After that, proceed to the next step.

- https://renesas-rz.github.io/rz_linux_bsp_plus/RZG/getting_started/

(2) Decompress the TSN package

Run the following command to decompress the TSN package.

```
$ cd ${WORK}
$ unzip ./RTK0EF0045Z0052AZJ-rz-tsn-v1_0_0.zip
$ tar zxvf ./RTK0EF0045Z0052AZJ-rz-tsn-v1_0_0/meta-rz-tsn_rzg3l_v1_0_0.tar.gz
```

(3) Build Initialize

Initialize a build using the 'oe-init-build-env' script in Poky and point TEMPLATECONF to platform conf path.

```
$ TEMPLATECONF=${PWD}/meta-renesas/meta-rz-distro/conf/templates/bsp-plus-conf/ \
source poky/oe-init-build-env build
```

(4) Add layers

In this step, you need to enable the layer of the TSN. The command to be executed depends on the specific TSN you intend to use.

```
$ bitbake-layers add-layer ../meta-openembedded/meta-networking
$ bitbake-layers add-layer ../meta-rz-tsn
```

(5) Add TSN settings in local.conf

In this step, you need to add the line to the local.conf file to install TSN. The lines to be added depend on the specific TSN you intend to use.

Add the following lines in the "\${WORK}/build/conf/local.conf".

PTP

```
# PTP
IMAGE_INSTALL:append = " linuxptp "
```

TDMA

```
# TDMA Driver
IMAGE_INSTALL:append = " iproute2 iproute2-tc iperf3 "
```

LLDP

```
# LLDP
IMAGE_INSTALL:append = " lldpd "
```

YANG & NETCONF

```
# YANG & NETCONF
IMAGE_INSTALL:append = " netopeer2-server "
```

EAP

```
# EAP
IMAGE_INSTALL:append = " freeradius "
IMAGE_INSTALL:append = " wpa-supPLICant "
```

For packet capture (Optional)

```
# For packet capture
IMAGE_INSTALL:append = " tcpdump "
```

(6) Start a build

Run the following command to start a build. Building an image can take up to a few hours depending on the user's host system performance. Available build options (<image name>) are:

- core-image-minimal
- core-image-weston

```
$ MACHINE=smarc-rzg3l bitbake <image name>
```

After the building is successfully completed, a similar output will be seen, and the command prompt will return.

NOTE: Tasks Summary: Attempted 6613 tasks of which 0 didn't need to be rerun and all succeeded.

All necessary files listed in the following table will be generated by the bitbake command and will be in the "build/tmp/depLOy/images/smarc-rzg3l" directory.

Table 3. Image files

RZ/G3L SMARC EVK	Linux kernel	Image-smarc-rzg3l.bin
	Device tree file	r9a08g046l48-smarc.dtb
	root filesystem	<image name>-smarc-rzg3l.rootfs.tar.bz2
	Boot loader	bl2_bp_spi-smarc-rzg3l.srec fip-smarc-rzg3l.srec
	SD image	<image name>-smarc-rzg3l.rootfs.wic.gz <image name>-smarc-rzg3l.rootfs.wic.bmap

If you want to know the components installed to the root filesystem, please check the manifest file. The manifest file is created to the following path after building the images:

```
${WORK}/build/tmp/depLOy/images/smarc-rzg3l/<image name>-smarc-rzg3l.rootfs.manifest
```

(7) Boot the evaluation board

Read the following site to know how to boot the evaluation board and boot it.

- https://renesas-rz.github.io/rz_linux_bsp_plus/RZG/getting_started/

When reading the site, ignore the build step because the build steps were completed by chapter 2 in this application note. After booting the evaluation board, read chapter 3 in this application note to know how to try TSN.

3. TSN

This chapter provides guidance on utilizing TSN on the evaluation board.

3.1 gPTP (IEEE 802.1AS)

This section describes the steps to use the master and slave operation of PTP based on IEEE 802.1AS.

LinuxPTP is open-source software that supports the Precision Time Protocol (PTP) in Linux environments. It also supports gPTP (generalized Precision Time Protocol) according to IEEE 802.1as, a profile of IEEE 1588.

3.1.1 How to prepare the environment

This section describes how to prepare the environment. Set the evaluation board (PTP Master) and the evaluation board (PTP Slave) as below.

When you want to check communication status with tools (Example: Wireshark), establish a connection between your PC and the Throwing Star Lan Tap. Basically, a direct Ethernet cable connection between the evaluation boards is sufficient.

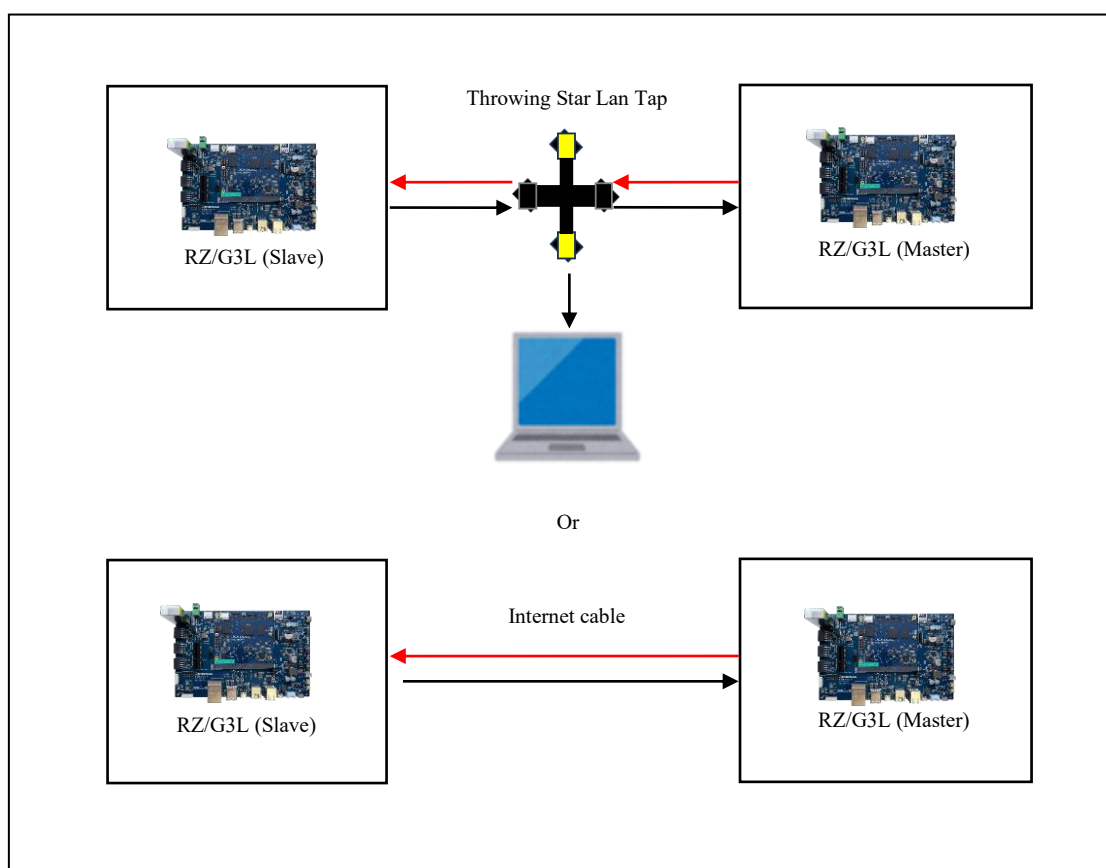


Figure 2. gPTP Environment

Note) There are the marks " $\leftarrow \rightarrow$ ", " $\leftarrow$ ", and " $\rightarrow$ " on the "Throwing Star Lan Tap" device. Please connect the Master and Slave with the mark " $\leftarrow \rightarrow$ ".

3.1.2 How to use gPTP

(1) Create conf files for Master and Slave

Create files with the following contents on Master/Slave. Normally, Master/Slave is determined by the difference in MAC Address (it is not recommended to change other values), but to check the operation of the sample, we have changed the priority2 value (red below) to switch between them.

[/root/gPTP.conf (Master)]

```
[global]
gmCapable          1
priority1          255
priority2          254
logAnnounceInterval 1
logSyncInterval    0
syncReceiptTimeout 3
neighborPropDelayThresh 800000
min_neighbor_prop_delay -200000000
assume_two_step    1
path_trace_enabled 1
#domainNumber      0
follow_up_info     1
transportSpecific  0x1
network_transport  L2
delay_mechanism     P2P
```

[/root/gPTP.conf (Slave)]

```
[global]
gmCapable          1
priority1          255
priority2          255
logAnnounceInterval 1
logSyncInterval    0
syncReceiptTimeout 3
neighborPropDelayThresh 800000
min_neighbor_prop_delay -200000000
assume_two_step    1
path_trace_enabled 1
#domainNumber      0
follow_up_info     1
transportSpecific  0x1
network_transport  L2
delay_mechanism     P2P
```

(2) Run PTP command

For Master:

```
$ ptp4l -i end1 -f /root/gPTP.conf --step_threshold=1 -m
```

Note) For RZ/G3L, only “end1” is usable for PTP synchronization. Because the PHC is shared between GBETH Channel 0 and 1, the value of the system time updated by GBETH Channel 1 is also reflected in GBETH Channel 0.

For Slave: (same as master)

```
$ ptp4l -i end1 -f /root/gPTP.conf --step_threshold=1 -m
```

(3) Check the PTP result by log

For checking Master, please confirm the message "LISTENING to MASTER on ANNOUNCE_RECEIPT_TIMEOUT_EXPIRES" is output to the log, as shown below.

```
root@smarc-rzg3l:/# ptp4l -i end1 -f gPTP.conf --step_threshold=1 -m
ptp4l[513.525]: selected /dev/ptp1 as PTP clock
ptp4l[513.584]: port 1 (end1): INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[513.584]: port 0 (/var/run/ptp4l): INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[513.585]: port 0 (/var/run/ptp4lro): INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[521.216]: port 1 (end1): LISTENING to MASTER on ANNOUNCE_RECEIPT_TIMEOUT_EXPIRES
ptp4l[521.217]: selected local clock 749050.ffff.00005e as best master
ptp4l[521.217]: port 1 (end1): assuming the grand master role
ptp4l[524.738]: port 1 (end1): new foreign master 923dad.ffff.88e58b-1
```

For checking Slave, confirm the message "UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED" is output to the log, as shown below. And the "master offset" should be convergence to the nsec unit.

```
root@smarc-rzg3l:/# ptp4l -i end1 -f gPTP.conf --step_threshold=1 -m
ptp4l[499.921]: selected /dev/ptp1 as PTP clock
ptp4l[499.964]: port 1 (end1): INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[499.964]: port 0 (/var/run/ptp4l): INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[499.965]: port 0 (/var/run/ptp4lro): INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[503.993]: port 1 (end1): new foreign master 749050.ffff.00005e-1
ptp4l[507.511]: port 1 (end1): LISTENING to MASTER on ANNOUNCE_RECEIPT_TIMEOUT_EXPIRES
ptp4l[507.511]: selected local clock 923dad.ffff.88e58b as best master
ptp4l[507.512]: port 1 (end1): assuming the grand master role
ptp4l[507.993]: selected best master clock 749050.ffff.00005e
ptp4l[507.993]: port 1 (end1): MASTER to UNCALIBRATED on RS_SLAVE
ptp4l[508.992]: master offset      -127 s0 freq  +1593 path delay      700
ptp4l[509.992]: master offset      -153 s2 freq  +1567 path delay      702
ptp4l[509.992]: port 1 (end1): UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
ptp4l[510.992]: master offset      -105 s2 freq  +1462 path delay      702
ptp4l[511.992]: master offset       -35 s2 freq  +1501 path delay      704
ptp4l[512.992]: master offset        46 s2 freq  +1571 path delay      703
ptp4l[513.992]: master offset        96 s2 freq  +1635 path delay      701
ptp4l[514.992]: master offset        48 s2 freq  +1616 path delay      701
ptp4l[515.992]: master offset        -2 s2 freq  +1580 path delay      703
...
...
```

By confirming the above logs, we can verify that the Master and Slave are communicating according to the PTP protocol.

(4) Check the PTP result by communication packet (optional)

Run below command before starting.

```
$ tcpdump -i end1 -w <Log File Name>.pcap &
```

Example:

If you want to create a "sample.pcap" log, then the command should be like below:

```
$ tcpdump -i end1 -w sample.pcap &
```

You can find "PTPv2" Protocol packets in the log. The following picture is a snapshot by using Wireshark.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	ba:13:c0:4a:a3:62	LLDP_Multicast	PTPv2	68	Peer_Delay_Req Message
2	1.000037	ba:13:c0:4a:a3:62	LLDP_Multicast	PTPv2	68	Peer_Delay_Req Message
3	2.000075	ba:13:c0:4a:a3:62	LLDP_Multicast	PTPv2	68	Peer_Delay_Req Message
4	3.000116	ba:13:c0:4a:a3:62	LLDP_Multicast	PTPv2	68	Peer_Delay_Req Message
5	3.000586	c2:b6:2b:3c:06:14	LLDP_Multicast	PTPv2	68	Peer_Delay_Resp Message
6	3.000710	c2:b6:2b:3c:06:14	LLDP_Multicast	PTPv2	68	Peer_Delay_Resp Follow Up Message

3.1.3 Explanation of the parameter in configuration file

The explanation of the parameter in configuration file "/root/PTP.conf" is below:

Config parameter name	Value	Description
gmCapable	1	This parameter is required to enable IEEE 802.1AS. When set to 1, the device can become the Grand Master device, Which serves as the time source in the network.
priority1	255	This parameter is used in the BMCA (Best Master Clock Algorithm) to determine the PTP time source device. The device with the smallest Priority1 value becomes the Master device. The setting range is 0 to 255.
priority2	255 or 254	This parameter is used in the BMCA to determine the PTP time source device. The device with the smaller Priority2 value becomes the Master device. The setting range is 0 to 255. Priority2 has a lower priority than Priority1.
logAnnounceInterval	1	Sets the interval for sending Announce messages, which contain the device's clock information. The value is specified as a power of 2, and the unit is seconds.
logSyncInterval	0	Sets the interval for sending Sync and Follow Up messages, which contain time information. The value is specified as a power of 2, and the unit is seconds.
syncReceiptTimeout	3	Sets the allowable number of failures in receiving Sync and Follow Up messages. If the number of failures exceeds this value, time synchronization is considered lost.
neighborPropDelayThresh	800000	Specifies the upper limit for transmission delay time. The unit is nanoseconds. If the calculated transmission delay exceeds this limit, the peer is considered non-compliant with IEEE 802.1AS.
min_neighbor_prop_delay	-20000000	Specifies the lower limit for transmission delay time. The unit is nanoseconds. If the calculated transmission delay falls below this limit, the peer is considered non-compliant with IEEE 802.1AS.
assume_two_step	1	This setting enables IEEE 802.1AS. When set to 1, the Two-Step Clock method is used for time distribution.

		In the Two-Step Clock method, a Follow Up message is sent after the PTP message frame is transmitted. In the One-Step Clock method, the transmission time is embedded in the Correction Field of the message frame header (Follow Up messages are not sent in One-Step Clock mode). IEEE 802.1AS uses the Two-Step Clock method.
path_trace_enabled	1	When set to 1, the Announce message includes data used in IEEE 802.1AS.
domainNumber	0	This parameter is not actually used. This is the value set for the domainNumber parameter in the PTP header.
follow_up_info	1	This setting enables IEEE 802.1AS. When set to 1, the Follow Up message includes data used in IEEE 802.1AS.
transportSpecific	0x1	This is the value set for the transportSpecific parameter in the PTP header. IEEE 802.1AS uses 0x1.
network_transport	L2	Specifies the communication protocol to be used. L2 uses Layer 2 frames to send and receive PTP message frames. IEEE 802.1AS uses L2.
delay_mechanism	P2P	Specifies the mechanism for measuring transmission delay time. The mechanisms available are E2E (End-to-End) and P2P (Peer-to-Peer). IEEE 802.1AS uses P2P.

Note) Below parameters should be set when using an opposite board running a stack compliant to IEEE 802.1AS:
Set logSyncInterval to "-3" (125ms) (otherwise opposite board slaves will report Sync Timeout).
Set ptp_dst_mac to "01:80:C2:00:00:0E" (needed for 802.1as).

3.2 PTP (IEEE 1588)

This section describes the steps to use the master and slave operation of PTP based on IEEE 1588.

3.2.1 How to prepare the environment

This section describes how to prepare the environment. Set the evaluation board (PTP Master) and the evaluation board (PTP Slave) as below.

When you want to check communication status with tools (Example: Wireshark), establish a connection between your PC and the Throwing Star Lan Tap. Basically, a direct Ethernet cable connection between the evaluation boards is sufficient.

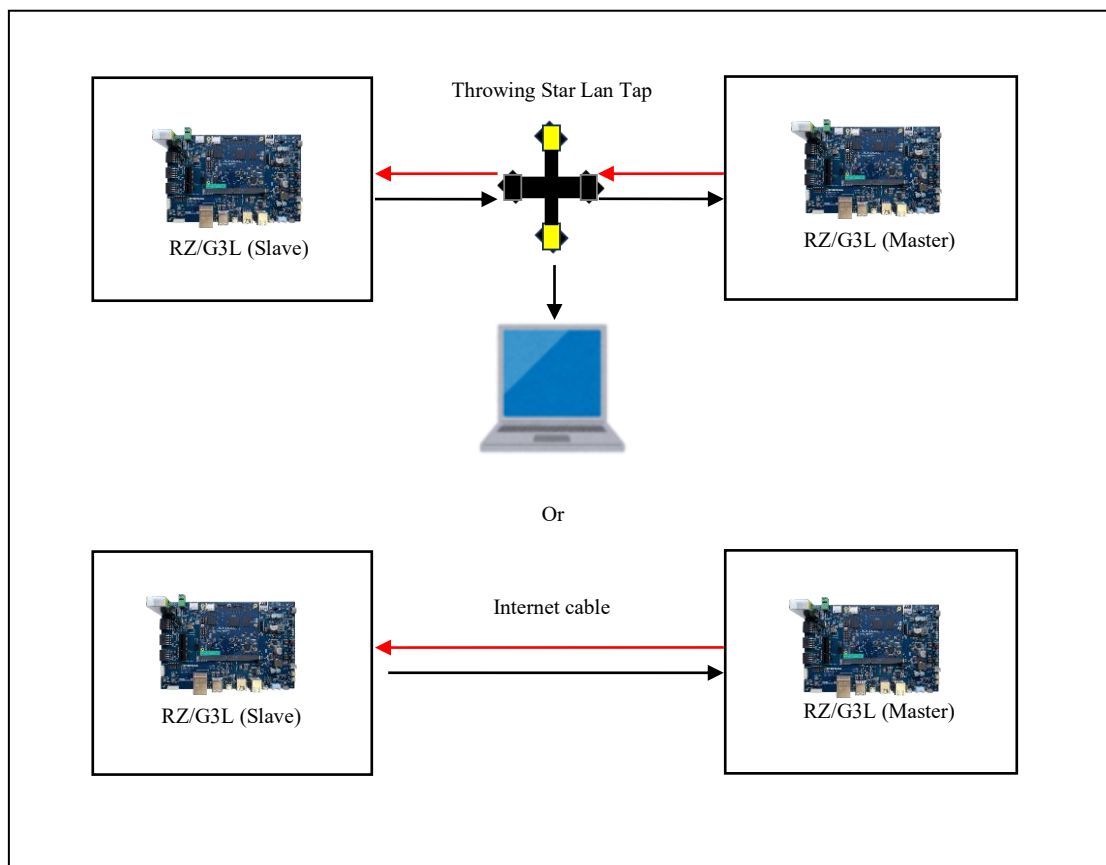


Figure 3. PTP Environment

Note) There are the marks " $\leftarrow \rightarrow$ ", " $\leftarrow$ ", and " $\rightarrow$ " on the "Throwing Star Lan Tap" device. Please connect the Master and Slave with the mark " $\leftarrow \rightarrow$ ".

3.2.2 How to use PTP

(1) Create conf files for Master and Slave

Create files with the following contents on Master/Slave. Normally, Master/Slave is determined by the difference in MAC Address (it is not recommended to change other values), but to check the operation of the sample, we have changed the priority2 value (red below) to switch between them.

[/root/PTP.conf (Master)]

```
[global]
priority1          255
priority2          254
```

[/root/PTP.conf (Slave)]

```
[global]
priority1          255
priority2          255
```

(2) Run PTP command

For Master:

```
$ ptp4l -i end1 -f /root/PTP.conf -m
```

For Slave: (same as master)

```
$ ptp4l -i end1 -f /root/PTP.conf -m
```

Note) **GBETH** Timestamp is used. For RZ/G3L, only "end1" is usable for PTP synchronization.

(3) Check the PTP result by log

For checking Master, please confirm the message "LISTENING to MASTER on ANNOUNCE_RECEIPT_TIMEOUT_EXPIRES" is output to the log, as shown below.

```
root@smarc-rzg3l:/# ptp4l -i end1 -f PTP.conf -m
ptp4l[1437.769]: selected /dev/ptp1 as PTP clock
ptp4l[1437.770]: port 1 (end1): INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[1437.771]: port 0 (/var/run/ptp4l): INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[1437.771]: port 0 (/var/run/ptp4lro): INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[1445.657]: port 1 (end1): LISTENING to MASTER on ANNOUNCE_RECEIPT_TIMEOUT_EXPIRES
ptp4l[1445.657]: selected local clock 749050.ffff.00005e as best master
ptp4l[1445.657]: port 1 (end1): assuming the grand master role
```

For checking Slave, please confirm the message "UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED" is output to the log, as shown below. And the "master offset" should be convergence to the nsec unit.

```

root@smarc-rzg3l:/# ptp4l -i end1 -f PTP.conf -m
ptp4l[1510.810]: selected /dev/ptp1 as PTP clock
ptp4l[1510.812]: port 1 (end1): INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[1510.812]: port 0 (/var/run/ptp4l): INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[1510.812]: port 0 (/var/run/ptp4lro): INITIALIZING to LISTENING on INIT_COMPLE
TE
ptp4l[1511.203]: port 1 (end1): new foreign master 749050.ffffe.00005e-1
ptp4l[1515.203]: selected best master clock 749050.ffffe.00005e
ptp4l[1515.204]: port 1 (end1): LISTENING to UNCALIBRATED on RS_SLAVE
ptp4l[1517.203]: master offset      4672 s0 freq  +1493 path delay      708
ptp4l[1518.203]: master offset      4808 s2 freq  +1629 path delay      684
ptp4l[1518.203]: port 1 (end1): UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
ptp4l[1519.203]: master offset      4768 s2 freq  +6397 path delay      708
ptp4l[1520.203]: master offset      -24 s2 freq  +3035 path delay      708
ptp4l[1521.203]: master offset     -1456 s2 freq  +1596 path delay      708
ptp4l[1522.203]: master offset     -1424 s2 freq  +1191 path delay      708
ptp4l[1523.203]: master offset     -1008 s2 freq  +1180 path delay      708
ptp4l[1524.203]: master offset      -584 s2 freq  +1302 path delay      708
ptp4l[1525.203]: master offset      -212 s2 freq  +1499 path delay      688
ptp4l[1526.203]: master offset      -120 s2 freq  +1527 path delay      708
ptp4l[1527.203]: master offset       -2 s2 freq  +1609 path delay      702
ptp4l[1528.203]: master offset       -2 s2 freq  +1608 path delay      702
ptp4l[1529.204]: master offset       -2 s2 freq  +1608 path delay      702
ptp4l[1530.204]: master offset       -2 s2 freq  +1607 path delay      702
ptp4l[1531.204]: master offset        12 s2 freq  +1621 path delay      696
ptp4l[1532.204]: master offset      -12 s2 freq  +1600 path delay      696
ptp4l[1533.204]: master offset        28 s2 freq  +1637 path delay      696
ptp4l[1534.204]: master offset      -28 s2 freq  +1589 path delay      696
ptp4l[1535.204]: master offset        36 s2 freq  +1645 path delay      696
ptp4l[1536.204]: master offset      -16 s2 freq  +1603 path delay      700
ptp4l[1537.204]: master offset         6 s2 freq  +1621 path delay      710
ptp4l[1538.204]: master offset      -18 s2 freq  +1598 path delay      710
ptp4l[1539.204]: master offset         8 s2 freq  +1619 path delay      700
ptp4l[1540.204]: master offset       -8 s2 freq  +1605 path delay      708
ptp4l[1541.204]: master offset         0 s2 freq  +1611 path delay      708...
...

```

By confirming the above logs, we can verify that the Master and Slave are communicating according to the PTP protocol.

(4) Check the PTP result by communication packet (optional)

Run below command before starting.

```
$ tcpdump -i end1 -w <Log File Name>.pcap &
```

Example:

If you want to create a "sample.pcap" log, then the command should be like below:

```
$ tcpdump -i end1 -w sample.pcap &
```

And the find "PTPv2" Protocol packets in the log. And the "Destination" is "224.0.1.129" which is a standard multicast address for the PTPv2 (IEEE 1588) protocol. The following picture is a snapshot by using Wireshark.

No.	Time	Source	Destination	Protocol	Length	Info
267	65.713950	192.168.15.2	224.0.1.129	PTPv2	86	Delay_Req Message
268	65.714209	192.168.15.1	224.0.1.129	PTPv2	96	Delay_Resp Message
270	66.269127	192.168.15.1	224.0.1.129	PTPv2	86	Sync Message
271	66.269262	192.168.15.1	224.0.1.129	PTPv2	86	Follow_Up Message
273	67.269211	192.168.15.1	224.0.1.129	PTPv2	86	Sync Message
274	67.269363	192.168.15.1	224.0.1.129	PTPv2	86	Follow_Up Message

3.3 TDMA driver (IEEE 802.1Qbv)

This section describes the steps to use the sending and receiving operation of the IEEE 802.1Qbv.

iproute2 is open-source software that supports the IEEE 802.1Qbv in Linux environments.

3.3.1 How to prepare the environment

This section describes how to prepare the environment. Set the evaluation board (sending) and the evaluation board (receiving) as below.

When you want to check communication status with tools (Example: Wireshark), establish a connection between your PC and the Throwing Star Lan Tap. Basically, a direct Ethernet cable connection between the evaluation boards is sufficient.

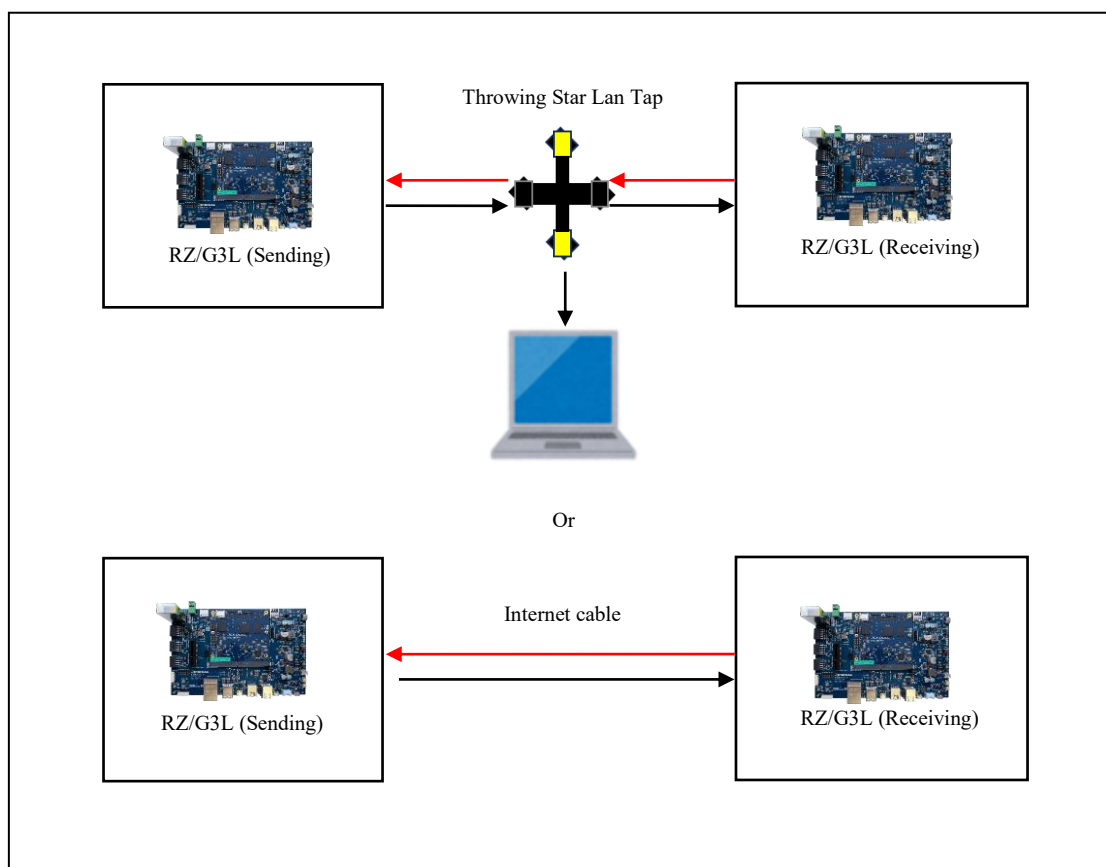


Figure 4. 802.1Qbv Environment

Note) There are the marks " $\leftarrow \rightarrow$ ", " $\leftarrow$ ", and " $\rightarrow$ " on the "Throwing Star Lan Tap" device. Please connect the Master and Slave with the mark " $\leftarrow \rightarrow$ ".

3.3.2 How to use 802.1Qbv for GBETH

(1) Set IP address to evaluation board (sending/receiving)

Note) Enhancements for Scheduled Traffic (IEEE 802.1Qbv) is supported on both GBETH interfaces (**end0** and **end1**). This example uses **end1**.

[sending]

Use GBETH Network Interface **end1** as an example.

```
$ ip link add link end1 name end1.1 type vlan id 1 egress-qos-map 0:0 1:1 2:2
$ ifconfig end1.1 10.0.0.1 netmask 255.255.255.252
```

[receiving]

Use GBETH Network Interface **end1** as an example.

```
$ ip link add link end1 name end1.1 type vlan id 1
$ ifconfig end1.1 10.0.0.2 netmask 255.255.255.252
```

(2) Synchronize the system time

[sending]

```
$ phc2sys -s CLOCK_REALTIME -c end1 -m -O 0 > /dev/null &
$ sleep 10
```

Note) For RZ/G3L, only “**end1**” is usable for PTP synchronization.

(3) Set the 802.1Qbv configuration

[sending]

```
$ tc qdisc add dev end1.1 clsact
#Using tc filter command edit the SKB priority based on udp port number. i.e udp port
5002 -> prio 2 (TC2/Q2), port 5001 -> prio 1 (TC1/Q1), 5000 -> prio 0(TC0/Q0)
$ tc filter add dev end1.1 egress protocol ip prio 1 u32 match ip dport 5000 0xffff \
action skbedit priority 0
$ tc filter add dev end1.1 egress protocol ip prio 1 u32 match ip dport 5001 0xffff \
action skbedit priority 1
$ tc filter add dev end1.1 egress protocol ip prio 1 u32 match ip dport 5002 0xffff \
action skbedit priority 2
```

Configure TDMA 802.1Qbv:

```
#TC0 <-> Q0, TC1 <-> Q1, and TC2 <-> Q2
#TC1 <-> prio1, TC2 <-> prio2
#tdma will start after 50000 nano seconds (base time 50000)
$ tc qdisc replace dev end1 parent root handle 100 taprio \
num_tc 3 \
map 0 1 2 0 0 0 0 0 0 0 0 0 0 0 0 \
queues 1@0 1@1 1@2 \
base-time 50000 \
sched-entry S 0x4 150000 \
sched-entry S 0x0 200000 \
sched-entry S 0x1 200000 \
sched-entry S 0x0 200000 \
sched-entry S 0x2 250000 \
sched-entry S 0x0 200000 \
flags 0x2

$ tc qdisc show dev end1
```

Note) Configure TDMA Qbv on the G3L sending side, which uses VLAN priority to classify traffic into ETHSW TDMA queues, as shown below:

- Traffic class 0: priority 0, queue 0, use port 5000
- Traffic class 1: priority 1, queue 1, use port 5001
- Traffic class 2: priority 2, queue 2, use port 5002
- The cycle time is 1200000 ns:
 - Port 5002 opens at the beginning of the cycle.
 - Port 5000 opens at 350000 (=150000 + 200000) ns in the cycle.
 - Port 5001 opens at 750000 (=150000 + 200000+ 200000+ 200000) ns in the cycle.

(4) Check the 802.1Qbv results using communication packets

Run the commands as below:

[receiving]

```
$ vi receive_lan_qbv.sh
```

- Create a receive_lan_qbv.sh with the following contents:

```
#!/bin/sh

iperf3 -s -p5000 &
iperf3 -s -p5001 &
iperf3 -s -p5002 &
tcpdump -i end1 -w <log file name>.pcap --time-stamp-precision nano
```

```
$ chmod 777 receive_lan_qbv.sh
$ ./receive_lan_qbv.sh
```

[sending]

```
$ vi send_lan_qbv.sh
```

- Create a send_lan_qbv.sh with the following contents:

```
#!/bin/sh

iperf3 -c 10.0.0.2 -u -b1G -p 5000 &
iperf3 -c 10.0.0.2 -u -b1G -p 5001 &
iperf3 -c 10.0.0.2 -u -b1G -p 5002 &
```

```
$ chmod 777 send_lan_qbv.sh
$ ./send_lan_qbv.sh
```

Note) Check the result when testing the 802.1Qbv based on the <log file name>.pcap file, as shown below:

- The following picture is a snapshot of the pcap file in 1 cycle:

610	16.256330	10.0.0.1	10.0.0.2	UDP	1494 33097 → 5001 Len=1448
611	16.256778	10.0.0.1	10.0.0.2	UDP	1494 41010 → 5002 Len=1448
612	16.256901	10.0.0.1	10.0.0.2	UDP	1494 41010 → 5002 Len=1448
613	16.257129	10.0.0.1	10.0.0.2	UDP	1494 36762 → 5000 Len=1448
614	16.257251	10.0.0.1	10.0.0.2	UDP	1494 36762 → 5000 Len=1448
615	16.257530	10.0.0.1	10.0.0.2	UDP	1494 33097 → 5001 Len=1448
616	16.257978	10.0.0.1	10.0.0.2	UDP	1494 41010 → 5002 Len=1448
617	16.258102	10.0.0.1	10.0.0.2	UDP	1494 41010 → 5002 Len=1448

- The cycle starts at 16.256778 and ends at 16.257978 → The cycle time is 1200 us = 1200000ns which matches the configured setting.
- Port 5000 opens at 350000 ns (16.257129 – 16.256778 ~ 350 us) in the cycle, which matches the configured setting.

- Port 5001 opens at 750000 ns (16,257530 – 16,256778 ~ 750 us) in the cycle, which matches the configured setting.

(5) Clear the 802.1Qbv configuration

To clear the 802.1Qbv configuration, run the following commands:

[sending]

```
$ tc qdisc del dev end1 parent root handle 100 taprio
$ tc qdisc del dev end1.1 clsact
```

3.4 TDMA driver (IEEE 802.1Qbu)

This section describes the steps to use the sending and receiving operation of the IEEE 802.1Qbu.

iproute2 is open-source software that supports the IEEE 802.1Qbu in Linux environments.

3.4.1 How to prepare the environment

This section describes how to prepare the environment. Set the evaluation board as below. PC is optional. Please connect it if you want to monitor the communication.

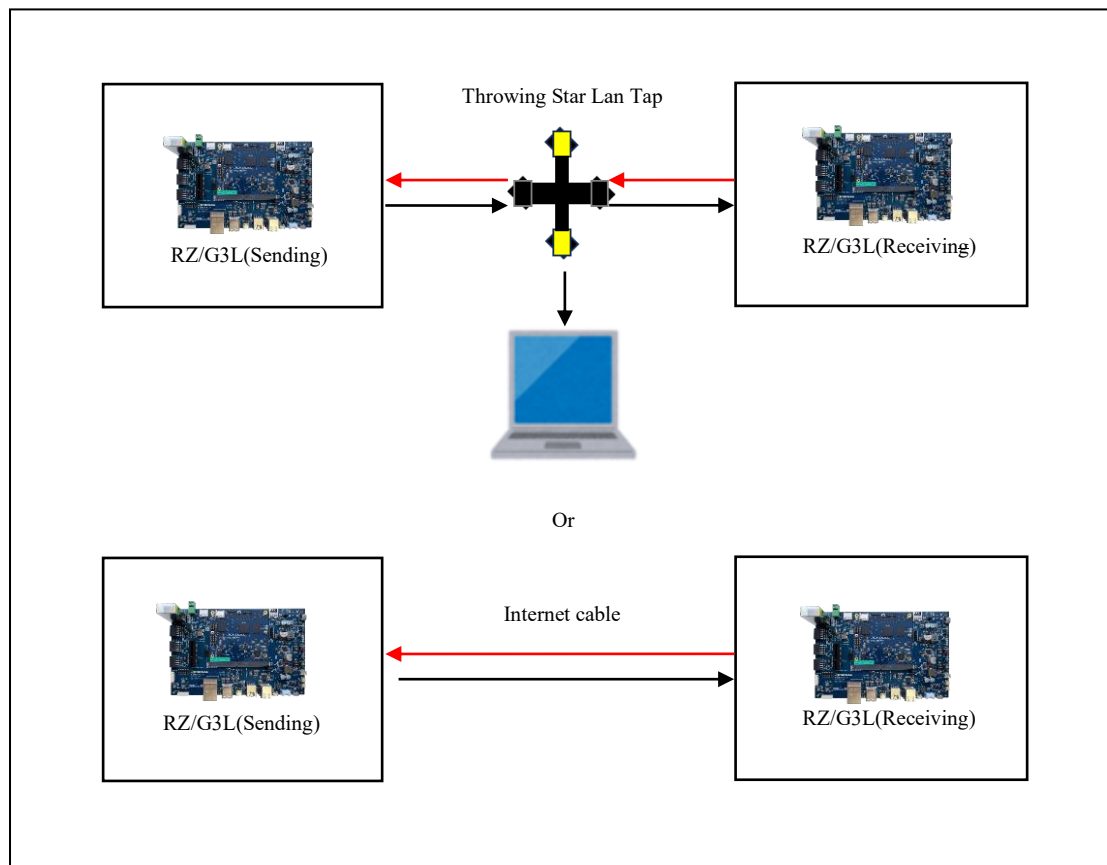


Figure 5. 802.1Qbu Environment

3.4.2 How to use 802.1Qbu for GBETH

(1) Check the current PMAC Merge status and verification state

Note) Frame Preemption (IEEE 802.1Qbu) is supported on both GBETH interfaces (**end0** and **end1**). This example uses **end1**.

Check the current PMAC Merge status and verification state on both RZ/G3L evaluation boards (sending and receiving) using the following command:

```
$ ethtool --include-statistics --show-mm end1
```

Check the status of the frame preemption and the verify handshake between devices before configuring 802.1 Qbu testing by using ethtool:

[Sending]

Example output:

```
MAC Merge layer state for end1:
pMAC enabled: off
TX enabled: off
TX active: off
TX minimum fragment size: 60
```

```

RX minimum fragment size: 60
Verify enabled: off
Verify time: 128
Max verify time: 128
Verification status: DISABLED
Statistics:
  MACMergeFrameAssErrorCount: 0
  MACMergeFrameSmdErrorCount: 0 (*1)
  MACMergeFrameAssOkCount: 0
  MACMergeFragCountRx: 0
  MACMergeFragCountTx: 0
  MACMergeHoldCount: 0

```

[receiving]

Example output:

```

MAC Merge layer state for end1:
pMAC enabled: off
TX enabled: off
TX active: off
TX minimum fragment size: 60
RX minimum fragment size: 60
Verify enabled: off
Verify time: 128
Max verify time: 128
Verification status: DISABLED
Statistics:
  MACMergeFrameAssErrorCount: 0
  MACMergeFrameSmdErrorCount: 0 (*1)
  MACMergeFrameAssOkCount: 0
  MACMergeFragCountRx: 0
  MACMergeFragCountTx: 0
  MACMergeHoldCount: 0

```

(*1) It should be “0”. A non-zero value may appear during initial link-up or PHY stabilization; however, it should remain unchanged during verification

(2) Configure the IP address for the RZ/G3L evaluation boards (sending and receiving)

[sending]

Note) Use GBETH Network Interface end1 as an example.

```
$ ifconfig end1 10.0.0.1 netmask 255.255.255.0
```

[receiving]

Note) Use GBETH Network Interface end1 as an example.

```
$ ifconfig end1 10.0.0.2 netmask 255.255.255.0
```

(3) Enable the Preemptible PMAC on the “endX” interface for the G3L evaluation boards (sending and receiving)

[sending]

```
$ ethtool --set-mm end1 pmac-enabled on
```

[receiving]

```
$ ethtool --set-mm end1 pmac-enabled on
```

(4) Configure PMAC merge transmission and verification for the G3L evaluation boards (sending and receiving)

[sending]

```
$ ethtool --set-mm end1 tx-enabled on  
$ ethtool --set-mm end1 verify-enabled on
```

[receiving]

```
$ ethtool --set-mm end1 tx-enabled on  
$ ethtool --set-mm end1 verify-enabled on
```

(5) Set the 802.1Qbu configuration

[sending]

```
$ tc qdisc replace dev end1 parent root handle 100 taprio \  
num_tc 2 \  
map 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 \  
queues 1@0 1@1 \  
base-time 1000000000 \  
sched-entry S 01 600000 \  
sched-entry S 02 400000 \  
fp E P \  
flags 0x2  
  
$ tc qdisc add dev end1 clsact  
$ tc filter add dev end1 egress protocol ip prio 1 u32 match ip dport 5000 0xffff \  
action skbedit priority 0  
$ tc filter add dev end1 egress protocol ip prio 1 u32 match ip dport 5001 0xffff \  
action skbedit priority 1  
$ tc qdisc show dev end1
```

Note)

- Queue 0 is configured as an express queue
- Queue 1 is configured as a preemptable queue

(6) Verify the frame preemption by statistics

Frame preemption can be verified by checking: Tx fragment count on the sender side and Rx fragment count on the receiver side. These counters indicate that preemptable traffic has been fragmented on the transmitter side and received/reassembled on the receiver side.

Run the following commands:

[receiving]

```
$ iperf3 -s -p5000 &  
$ iperf3 -s -p5001 &
```

[sending]

```
$ iperf3 -c 10.0.0.2 -u -b1G -p 5000 & iperf3 -c 10.0.0.2 -u -b1G -p 5001 &
```

Run the following commands to verify the frame preemption by statistics on both RZ/G3L evaluation boards (sending and receiving):

```
$ ethtool --include-statistics --show-mm end1
```

[Sending]

Example output:

```
MAC Merge layer state for end1:
pMAC enabled: on (*1)
TX enabled: on (*1)
TX active: on (*1)
TX minimum fragment size: 60 (*3)
RX minimum fragment size: 60 (*3)
Verify enabled: on
Verify time: 128
Max verify time: 128
Verification status: SUCCEEDED (*2)
Statistics:
  MACMergeFrameAssErrorCount: 0 (*5)
  MACMergeFrameSmdErrorCount: 0 (*6)
  MACMergeFrameAssOkCount: 0
  MACMergeFragCountRx: 0
  MACMergeFragCountTx: 9327 (*4)
  MACMergeHoldCount: 89154
```

Observation:

- MACMergeFragCountTx = 9327 indicates fragmented frames were transmitted
- MACMergeFragCountRx = 0 indicates no fragmented frames were received on this side
- Verification status SUCCEEDED indicates MAC Merge verification completed successfully

[receiving]

Example output:

```
MAC Merge layer state for end1:
pMAC enabled: on (*1)
TX enabled: on (*1)
TX active: on (*1)
TX minimum fragment size: 60 (*3)
RX minimum fragment size: 60 (*3)
Verify enabled: on
Verify time: 128
Max verify time: 128
Verification status: SUCCEEDED (*2)
Statistics:
  MACMergeFrameAssErrorCount: 0 (*5)
  MACMergeFrameSmdErrorCount: 0 (*6)
  MACMergeFrameAssOkCount: 9327 (*4)
  MACMergeFragCountRx: 9327 (*4)
  MACMergeFragCountTx: 0
  MACMergeHoldCount: 0
```

Observation:

- MACMergeFragCountRx = 9327 indicates fragmented frames were received
- MACMergeFrameAssOkCount = 9327 indicates successful frame reassembly
- MACMergeFragCountTx = 0 indicates this side did not transmit fragmented frames

Note)

*1 Should be “on”

*2 Should be “SUCCEEDED”

- *3 Should be not “0”
- *4 All values should be the same and note “0”
- *5 Should be “0”
- *6 Should remain unchanged during verification from “step (1)”

3.5 LLDP

This section describes the steps to use the LLDP.

lldpd is open-source software that supports the LLDP in Linux environments.

3.5.1 How to prepare the environment

This section describes how to prepare the environment. Set the evaluation board as below. PC is optional. Please connect it if you want to monitor the communication.

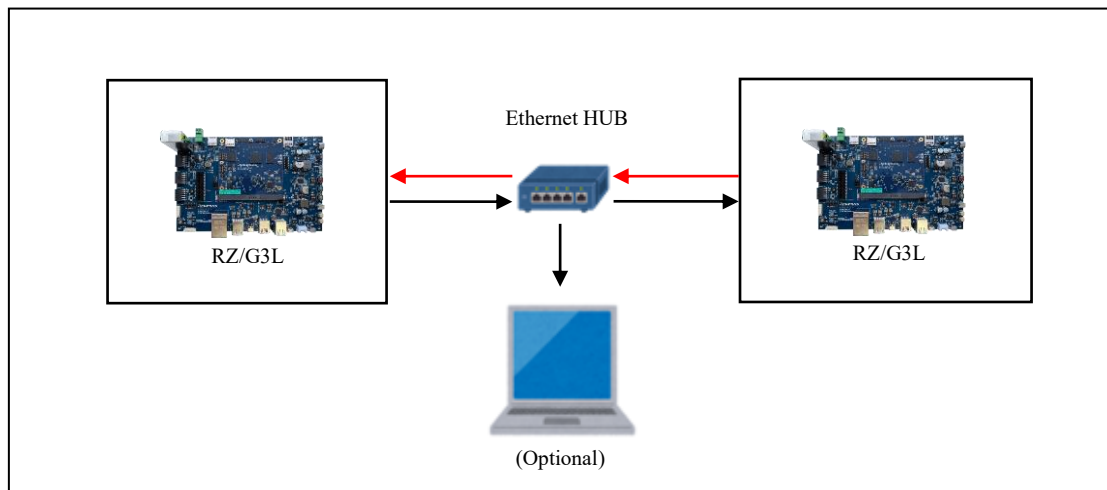


Figure 6. LLDP Environment

Note) There are the marks " $\leftarrow \rightarrow$ ", " $\leftarrow$ ", and " $\rightarrow$ " on the "Throwing Star Lan Tap" device. Please connect two boards with the mark " $\leftarrow \rightarrow$ ".

3.5.2 How to use LLDP

(1) Run LLDP command

Run the command below for both RZ/G3L board.

```
$ lldpd
```

(2) Check the LLDP result by log

The device information appears after running the **LLDP** command when checking **GBETH**.

```

root@smarc-rzg3l:/# lldpctl
Interface:  end1, via: LLDP, RID: 2, Time: 0 day, 00:00:19
Chassis:
  ChassisID:  mac 74:90:50:00:00:37
  SysName:    smarc-rzg3l
  SysDescr:   Poky (Yocto Project Reference Distro) 5.0.14 (scarthgap) Linux 6.12.
46-cip8-yocto-standard-ge6c894b7436f #1 SMP PREEMPT Thu Jul  9 07:58:28 UTC 2026 aarch
64
  MgmtIP:     192.168.5.75
  MgmtIface:  6
  MgmtIP:     fe80::7690:50ff:fe00:37
  MgmtIface:  6
  Capability: Bridge, off
  Capability: Router, off
  Capability: Wlan, off
  Capability: Station, on
  
```

```

Port:
  PortID:      mac 42:8b:f1:60:64:ec
  PortDescr:   end1
  TTL:        120
  PMD autoneg: supported: yes, enabled: yes
  Adv:         10Base-T, HD: no, FD: yes
  Adv:         100Base-TX, HD: no, FD: yes
  Adv:         1000Base-T, HD: no, FD: yes
  MAU oper type: 1000BaseTFD - Four-pair Category 5 UTP, full duplex mode
  VLAN:       1, pvid: no vlan1

```

Note) If the above log is not shown, please wait and run the command again.

By confirming the above log, we can verify that the Master and Slave are communicating according to the LLDP protocol.

(3) Check the LLDP result by communication packet (optional)

Run below command before starting.

```
$ tcpdump -i <Network Interface> -w <Log File Name>.pcap &
```

Example:

If the <Network Interface> is "end1" and you want to create a "sample.pcap" log, then the command should be like below:

```
$ tcpdump -i end1 -w sample.pcap &
```

You can find "LLDP" Protocol packets in the log. The following picture is a snapshot by using Wireshark.

No.	Time	Source	Destination	Protocol	Length	Info
152	73.982392	ba:ac:55:6c:71:8d	LLDP_Multicast	LLDP	253	MA/be:3f:71:30:93:87 MA/ba:ac:55:6c:71:8d 120 SysN=rzt2h-dev Sys
178	90.995976	ce:34:6f:3a:2e:fd	LLDP_Multicast	LLDP	253	MA/f6:b8:bf:42:7f:f9 MA/ce:34:6f:3a:2e:fd 120 SysN=rzt2h-dev Sys
194	103.989457	ba:ac:55:6c:71:8d	LLDP_Multicast	LLDP	253	MA/be:3f:71:30:93:87 MA/ba:ac:55:6c:71:8d 120 SysN=rzt2h-dev Sys
217	121.006956	ce:34:6f:3a:2e:fd	LLDP_Multicast	LLDP	253	MA/f6:b8:bf:42:7f:f9 MA/ce:34:6f:3a:2e:fd 120 SysN=rzt2h-dev Sys
235	133.996373	ba:ac:55:6c:71:8d	LLDP_Multicast	LLDP	253	MA/be:3f:71:30:93:87 MA/ba:ac:55:6c:71:8d 120 SysN=rzt2h-dev Sys
270	151.013936	ce:34:6f:3a:2e:fd	LLDP_Multicast	LLDP	253	MA/f6:b8:bf:42:7f:f9 MA/ce:34:6f:3a:2e:fd 120 SysN=rzt2h-dev Sys
292	164.003262	ba:ac:55:6c:71:8d	LLDP_Multicast	LLDP	253	MA/be:3f:71:30:93:87 MA/ba:ac:55:6c:71:8d 120 SysN=rzt2h-dev Sys
311	180.731761	ce:34:6f:3a:2e:fd	LLDP_Multicast	LLDP	60	MA/f6:b8:bf:42:7f:f9 MA/ce:34:6f:3a:2e:fd 0 (Shutdown LLDPDU)
315	182.958719	ba:ac:55:6c:71:8d	LLDP_Multicast	LLDP	60	MA/be:3f:71:30:93:87 MA/ba:ac:55:6c:71:8d 0 (Shutdown LLDPDU)
335	200.264597	ba:ac:55:6c:71:8d	LLDP_Multicast	LLDP	253	MA/be:3f:71:30:93:87 MA/ba:ac:55:6c:71:8d 120 SysN=rzt2h-dev Sys

3.6 YANG/NETCONF

This section describes the steps to use the YANG/NETCONF.

libyang and libnetconf2 are open-source software for supporting YANG/NETCONF on Linux environments. To add sample codes for testing, open-source software netopeer2 and sysrepo are also added.

3.6.1 How to prepare the environment

This section describes how to prepare the environment. Set the evaluation board as below. PC is optional. Please connect it if you want to monitor the communication.

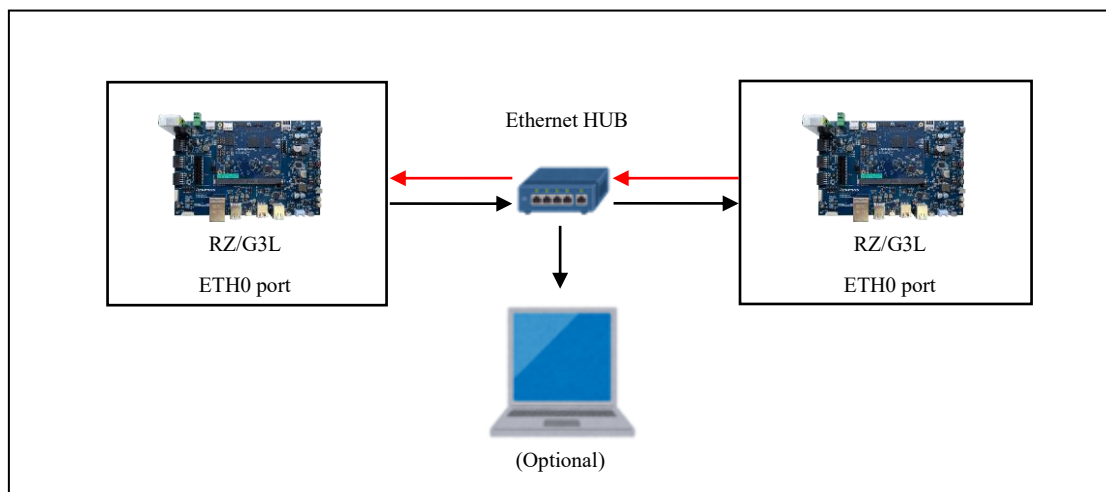


Figure 7. YANG/NETCONF Environment

Note) There are the marks " $\leftarrow \rightarrow$ ", " $\leftarrow$ ", and " $\rightarrow$ " on the "Throwing Star Lan Tap" device. Please connect Server and Client device with the mark " $\leftarrow \rightarrow$ ".

3.6.2 How to use YANG/NETCONF

(1) Run YANG/NETCONF command

Set the <Network Interface> on the Server side.

Note) Use GBETH Timestamp by setting the <Network Interface> as "endX".

Use the <Network Interface> as "end0" for example.

```
$ ifconfig end0 192.168.15.1 netmask 255.255.255.0 up
$ route add default gw 192.168.15.1
```

Set the Network Interface on the Client side.

```
$ ifconfig end0 192.168.15.2 netmask 255.255.255.0 up
$ route add default gw 192.168.15.1
```

Start the server on the Server side.

```
$ systemctl start sysrepor
$ netopeer2-server -d -v2
```

Note) Those commands maybe take a lot of time (about 3 minutes) only at the first time. Please wait the command successfully.

Run the connect command on the Client side. Then run the red command for testing. When the red commands can be run successfully, we can verify that the Master and Slave are communicating according to the YANG/NETCONF protocol.

```
$ netopeer2-cli
> connect --ssh --host <Server IP Address> --port <Port> --login root
The authenticity of the host '<Server IP Address>' cannot be established.
ssh-rsa key fingerprint is *****.
Are you sure you want to continue connecting (yes/no)? yes
> root@<Server IP Address> password:<Password>
> get
DATA
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
...
...
</data>
> get-config --source startup
DATA
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"/>
...
...
</data>
> delete-config --target startup
OK
> get-config --source startup
DATA
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"/>
> copy-config --source running --target startup
OK
> get-config --source startup
DATA
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"/>
...
...
</data>
```

```
> disconnect
> exit
```

Note) About **<Server IP Address>**, in this sample, you can set it as 192.168.15.1 .

Note) About **<Port>**, the default is 830 .

Note) About **<Password>**, the default is empty. If there is no password, then this section will not show up.

Note) After ran "netopeer2-cli ", please type the command manually instead of Copy-Paste, because if you use Copy-Paste, it will make the TeraTerm freeze.

Note) Please input the "yes " and **<Password>** within 5 seconds.

You can close the client connection after running "exit" command.

(2) Check the YANG/NETCONF result by communication packet (optional)

Run below command before starting.

```
$ tcpdump -i <Network Interface> -w <Log File Name>.pcap &
```

Example:

If the **<Network Interface>** is "end0". And you want to create a "sample.pcap" log, then the command should be like below:

```
$ tcpdump -i end0 -w sample.pcap &
```

You can find high frequency TCP communication packets in the log. The following picture is a snapshot by using Wireshark.

No.	Time	Source	Destination	Protocol	Length	Info
107	47.777513	192.168.15.2	192.168.15.1	TCP	66	56390 → 50830 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3284623449 TSecr=3176743848
108	47.777767	192.168.15.2	192.168.15.1	TCP	88	56390 → 50830 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=22 TSval=3284623449 TSecr=3176743848
109	47.777800	192.168.15.1	192.168.15.2	TCP	66	50830 → 56390 [ACK] Seq=1 Ack=23 Win=65152 Len=0 TSval=3176743849 TSecr=3284623449
110	47.850995	192.168.15.1	192.168.15.2	TCP	776	50830 → 56390 [PSH, ACK] Seq=1 Ack=23 Win=65152 Len=710 TSval=3176743922 TSecr=3284623449
111	47.851326	192.168.15.2	192.168.15.1	TCP	66	56390 → 50830 [ACK] Seq=23 Ack=711 Win=64128 Len=0 TSval=3284623522 TSecr=3176743922
112	47.851910	192.168.15.2	192.168.15.1	TCP	738	56390 → 50830 [PSH, ACK] Seq=23 Ack=711 Win=64128 Len=672 TSval=3284623523 TSecr=3176743922
113	47.851940	192.168.15.1	192.168.15.2	TCP	66	50830 → 56390 [ACK] Seq=711 Ack=695 Win=64512 Len=0 TSval=3176743923 TSecr=3284623523
114	47.866980	192.168.15.2	192.168.15.1	TCP	114	56390 → 50830 [PSH, ACK] Seq=695 Ack=711 Win=64128 Len=48 TSval=3284623538 TSecr=3176743923
115	47.867002	192.168.15.1	192.168.15.2	TCP	66	50830 → 56390 [ACK] Seq=711 Ack=743 Win=64512 Len=0 TSval=3176743938 TSecr=3284623538
116	47.910128	192.168.15.1	192.168.15.2	TCP	682	50830 → 56390 [PSH, ACK] Seq=711 Ack=743 Win=64512 Len=616 TSval=3176743981 TSecr=3284623538
117	47.914396	192.168.15.1	192.168.15.2	TCP	82	50830 → 56390 [PSH, ACK] Seq=1327 Ack=743 Win=64512 Len=16 TSval=3176743986 TSecr=3284623538
118	47.925263	192.168.15.2	192.168.15.1	TCP	82	56390 → 50830 [PSH, ACK] Seq=743 Ack=1343 Win=64128 Len=16 TSval=3284623596 TSecr=3176743981
119	47.925296	192.168.15.1	192.168.15.2	TCP	66	50830 → 56390 [ACK] Seq=1343 Ack=759 Win=64512 Len=0 TSval=3176743996 TSecr=3284623596
120	47.926274	192.168.15.2	192.168.15.1	TCP	130	56390 → 50830 [PSH, ACK] Seq=759 Ack=1343 Win=64128 Len=64 TSval=3284623597 TSecr=3176743996
121	47.926297	192.168.15.1	192.168.15.2	TCP	66	50830 → 56390 [ACK] Seq=1343 Ack=823 Win=64512 Len=0 TSval=3176743997 TSecr=3284623597
122	47.926942	192.168.15.1	192.168.15.2	TCP	258	50830 → 56390 [PSH, ACK] Seq=1343 Ack=823 Win=64512 Len=192 TSval=3176743998 TSecr=3284623597
123	47.934393	192.168.15.1	192.168.15.2	TCP	130	50830 → 56390 [PSH, ACK] Seq=1535 Ack=823 Win=64512 Len=64 TSval=3176744006 TSecr=3284623597
124	47.934753	192.168.15.2	192.168.15.1	TCP	66	56390 → 50830 [ACK] Seq=823 Ack=1599 Win=64128 Len=0 TSval=3284623606 TSecr=3176743998

3.7 EAP

This section describes the steps to use the EAP.

freeradius is open-source software that supports EAP in Linux environments.

3.7.1 How to prepare the environment

This section describes how to prepare the environment. Set the evaluation board as below. PC is optional. Please connect it if you want to monitor the communication.

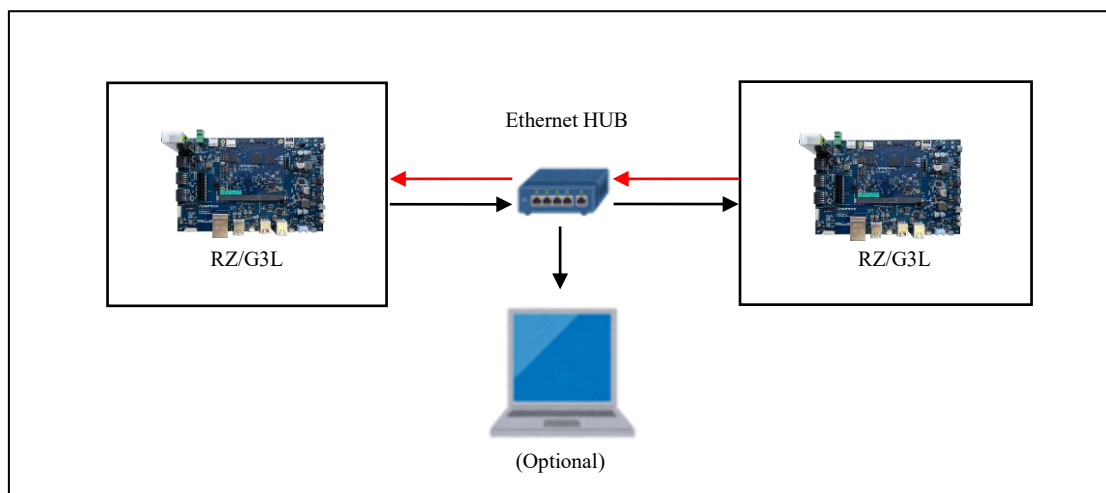


Figure 8. EAP Environment

Note) There are the marks " $\leftarrow \rightarrow$ ", " $\leftarrow$ ", and " $\rightarrow$ " on the "Throwing Star Lan Tap" device. Please Server and Client device with the mark " $\leftarrow \rightarrow$ ".

If you use different Client board (device) the you need to copy the "certs" files

From: Build PC direcorey:

[~/build/tmp/work/cortexa55-poky-linux/freeradius/3.2.6/image/etc/raddb](#)

To: Clinet board (device) root filesystem:

[/etc/raddb](#)

3.7.2 How to use EAP

(1) Run EAP command

Set the **<Network Interface>** on the Server side.

Note) Use **GBETH** Timestamp by setting the **<Network Interface>** as "endX".

Use the **<Network Interface>** as "end0" for example.

```
$ ifconfig end0 192.168.15.1 netmask 255.255.255.0 up
$ route add default gw 192.168.15.1
```

Set the Network Interface on the Client side.

```
$ ifconfig end0 192.168.15.2 netmask 255.255.255.0 up
$ route add default gw 192.168.15.1
```

Set the date and time for both Server and Client board. For pass the certificate, we need set a validity period. The validity period is one year from the time of certificate files creation.

For example, if the certificate files are created (built) at 18:00 Aug. 30, 2026, then the validity period should be

[18:00 Aug. 30, 2026] ~ [18:00 Aug. 30, 2027]

If you want to set the date and time as 18:05 Aug. 30, 2026, the command is below.

```
$ date 083018052026.00
```

Set the config file on Server side.

1. File "/etc/raddb/clients.conf" around Line 310. <RADIUS Client shared key> can be defined by yourself.

```
client private-network-1 {
    ipaddr      = <RADIUS Client IP address>
    secret      = <RADIUS Client shared key>
}
```

Example:

Set the <RADIUS Client IP address> as "192.168.15.2".

And the <RADIUS Client shared key> as "testing123", then the command should be like below:

```
client private-network-1 {
    ipaddr      = 192.168.15.2
    secret      = testing123
}
```

2. File "/etc/raddb/users" around Line 90. (Add below content)

```
<Username>   Cleartext-Password := <Userpassword>
              Reply-Message := "Hello, %{User-Name}"
```

Set the <Username> as "renesas".

And the <Userpassword> as "mypassword123", then the command should be like below:

```
renesas Cleartext-Password := "mypassword123"
        Reply-Message := "Hello, %{User-Name}"
```

Note: There are 8 spaces before "Reply-Message".

Run the sample app on the Server side.

```
$ systemctl stop radiusd
$ radiusd -X
```

Create the config file name "eapol.conf" on Client side.

The file content is below:

```
network={
    key_mgmt=<Encryption method>
    eap=<EAP Encryption method>
    identity=<Username>
    password=<Userpassword>

    # RADIUS server CA
    ca_cert=<Root certificate file>
    phase1="peapver=0"
    phase2="auth=MSCHAPV2"
}
```

Example:

As the default root file system, they should be set as below:

```
network={
    key_mgmt=WPA-EAP
    eap=PEAP
    identity="renesas"
    password="mypassword123"

    # RADIUS server CA
    ca_cert="/etc/raddb/certs/ca.pem"
    phase1="peapver=0"
    phase2="auth=MSCHAPV2"
}
```

Run the sample app on the Client side.

```
$ eapol_test -c eapol.conf -a <Server IP Address> -s <RADIUS Client shared key>
```

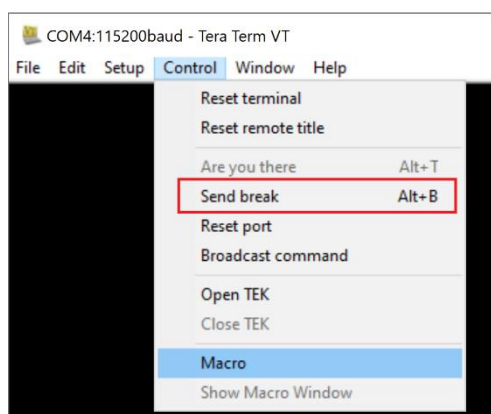
Example:

Set the <Server IP Address> as "192.168.15.1".

And the < RADIUS Client shared key > as "testing123", then the command should be like below:

```
$ eapol_test -c eapol.conf -a 192.168.15.1 -s testing123
```

Note) There are lot of log output during above command that may cause Tera Term freeze. Please send break like below when the Tera Term freezes. You also can send the log to a log file by using command "> test.log".



(2) Check the EAP result by log

Both the Server and Client board can run commands successfully.

At Server side, you should see a lot of text printed on the screen as it starts up. If the server says, "**Ready to process requests**", then it is running properly.

At Client side, you should see message "**Hello, <Username>**" (like "**Hello, renesas**" if you use the example), then it is running properly.

For Server side:

```
root@smarc-rzg3l:/# radiusd -X
FreeRADIUS Version 3.2.5
Copyright (C) 1999-2023 The FreeRADIUS server project and contributors
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE
You may redistribute copies of FreeRADIUS under the terms of the
GNU General Public License
For more information about these matters, see the file named COPYRIGHT
Starting - reading configuration files ...
including dictionary file /usr/share/freeradius/dictionary...
...
Waking up in 0.3 seconds.
(10) Cleaning up request packet ID 10 with timestamp +18 due to cleanup_delay was reac
hed
Ready to process requests
```

For Client side:

```
root@smarc-rzg3l:/# eapol_test -c eapol.conf -a 192.168.15.2 -s testing123
Reading configuration file 'eapol.conf'
Line: 1 - start of a new network block...
...
RADIUS message: code=11 (Access-Challenge) identifier=1 length=80
RADIUS message: code=2 (Access-Accept) identifier=10 length=175
  Attribute 80 (Message-Authenticator) length=18
    Value: ff7485a548aac9301d273b328edf0f3a
  Attribute 26 (Vendor-Specific) length=58
    Value: 00000137113482c416197ee94c6a5d18a3861b525db9a6f13bb8f7eb74b6a37c042da4b77
a4561311ade52e77dabcd87e6b31ff626514a26
  Attribute 26 (Vendor-Specific) length=58
    Value: 0000013710348cd35d2988df9e134f4acb9c7b3a69be2766bda2aba9fdd5e8308395bb83d
47f5d60828ee2216608c679693e6f95538f363a
  Attribute 79 (EAP-Message) length=6
    Value: 03690004
  Attribute 1 (User-Name) length=9
    Value: 'renesas'
  Attribute 12 (Framed-MTU) length=6
    Value: 994
...
WPA: Clear old PMK and PTK
EAP: deinitialize previously used EAP method (25, PEAP) at EAP deinit
ENGINE: engine deinit
MPPE keys OK: 1 mismatch: 0
SUCCESS
```

By confirming the above log, we can verify that the Master and Slave are communicating according to the EAP protocol.

(3) Check the EAP result by communication packet (optional)

Run below command before starting.

```
$ tcpdump -i <Network Interface> -w <Log File Name>.pcap &
```

Example:

If the <Network Interface> is "end0". And you want to create a "sample.pcap" log, then the command should be like below:

```
$ tcpdump -i end0 -w sample.pcap &
```

You can find "RADIUS" Protocol packets in the log. The following picture is a snapshot by using Wireshark.

No.	Time	Source	Destination	Protocol	Length	Info
85	46.889840	192.168.15.1	192.168.15.2	RADIUS	122	Access-Challenge id=0
88	47.223887	192.168.15.2	192.168.15.1	RADIUS	191	Access-Request id=1
89	47.226144	192.168.15.1	192.168.15.2	RADIUS	106	Access-Challenge id=1
90	47.563838	192.168.15.2	192.168.15.1	RADIUS	375	Access-Request id=2
92	47.915112	192.168.15.1	192.168.15.2	RADIUS	1110	Access-Challenge id=2
93	48.249767	192.168.15.2	192.168.15.1	RADIUS	191	Access-Request id=3
94	48.251991	192.168.15.1	192.168.15.2	RADIUS	976	Access-Challenge id=3
101	50.265539	192.168.15.2	192.168.15.1	RADIUS	123	Access-Request id=4
103	50.601428	192.168.15.1	192.168.15.2	RADIUS	106	Access-Challenge id=4

4. Appendix

4.1 Ethernet port

(1) Check the Ethernet ports on the evaluation board and the interfaces in Linux

Please note that the Ethernet ports on the evaluation board are assigned in Linux as shown in the table below.

Table 4. Interfaces GBETH on Linux

Ethernet Port on the evaluation board	Channel	Interface on Linux
ETH0	GBETH0	end0
ETH1	GBETH1	end1

If you want to check it on Linux, run the ifconfig command after booting Linux.

```

root@smarc-rzg3l:/# ifconfig
end0      Link encap:Ethernet  HWaddr 74:90:50:00:00:37
          inet addr:192.168.5.75  Bcast:192.168.5.255  Mask:255.255.255.0
          inet6 addr: fe80::7690:50ff:fe00:37/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:33911 errors:0 dropped:0 overruns:0 frame:0
          TX packets:20248 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:43717223 (41.6 MiB)  TX bytes:3162640 (3.0 MiB)
          Interrupt:47

end1      Link encap:Ethernet  HWaddr 74:90:50:00:00:36
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:51

```

5. Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Sep. 3, 2026	-	First edition issued.

Website and Support

Renesas Electronics Website

<http://www.renesas.com/>

Inquiries

<http://www.renesas.com/contact/>

All trademarks and registered trademarks are the property of their respective owners.