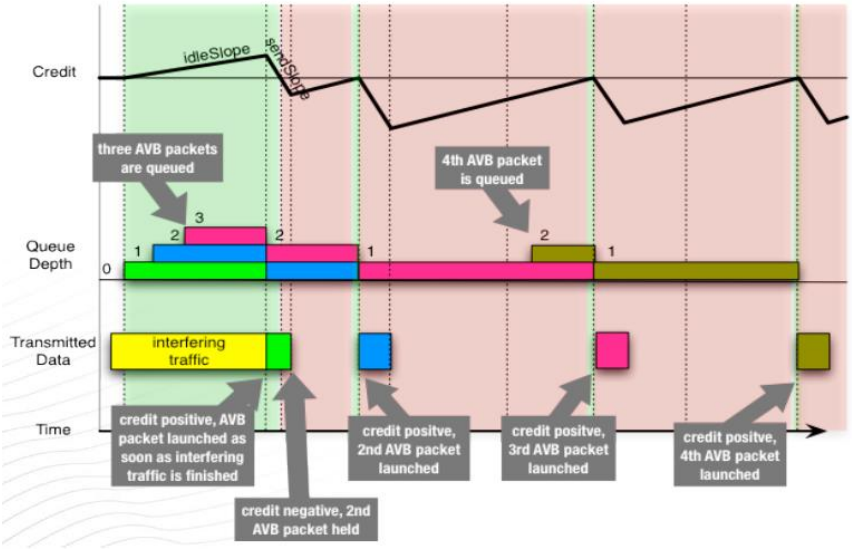
Time Sensitive Networking on LS1028ARDB

Time Sensitive Networking(TSN) is an extension to traditional Ethernet networks, providing a set of standards compatible with IEEE 802.1 and 802.3. TSN aims to provide guarantees for deterministic latency and packet loss under congestion, allowing critical and non-critical traffic to be converged in the same network. This document introduces basic concept of Time Sensitive Networking, LS1028 TSN switch and using TSN features on LS1028ARDB.

TSN Introduction

TSN is all about Layer 2 of the OSI model and is an extension to IEEE 802.1 to make Ethernet deterministic, more robust and
reliable. Time Sensitive Networking (TSN) refers to a set of standards that are being driven and developed by the IEEE 802.1 Time Sensitive Networking Task Group.
TSN standards
P802.1AS-Rev –Timing and Synchronization -Revision
802.1Qbu –Frame Preemption – published
802.1Qbv –Enhancements for Scheduled Traffic–published
802.1Qca –IS-IS Path Control and Reservation (PCR)–published
P802.1Qcc –Stream Reservation Protocol (SRP) Enhancements and Performance Improvements
P802.1Qch –Cyclic Queuing and Forwarding –based on Qci
P802.1Qci –Per-Stream Filtering and Policing
P802.1Qcj –Auto-attach to PBB services
P802.1Qcp –YANG Data Model
P802.1Qcr –Asynchronous Traffic Shaping (ATS)
P802.1CB –Frame Replication and Elimination for Reliability
P802.1CM –Time-Sensitive Networking for Fronthaul
P802.1CS –Link-local Registration Protocol (LRP) –

IEEE 802.1Qav – Forwarding and queuing for Time-Sensitive stream



802.1Qbv Time-Aware Shaper

Make switches aware of the cycle time for control traffic

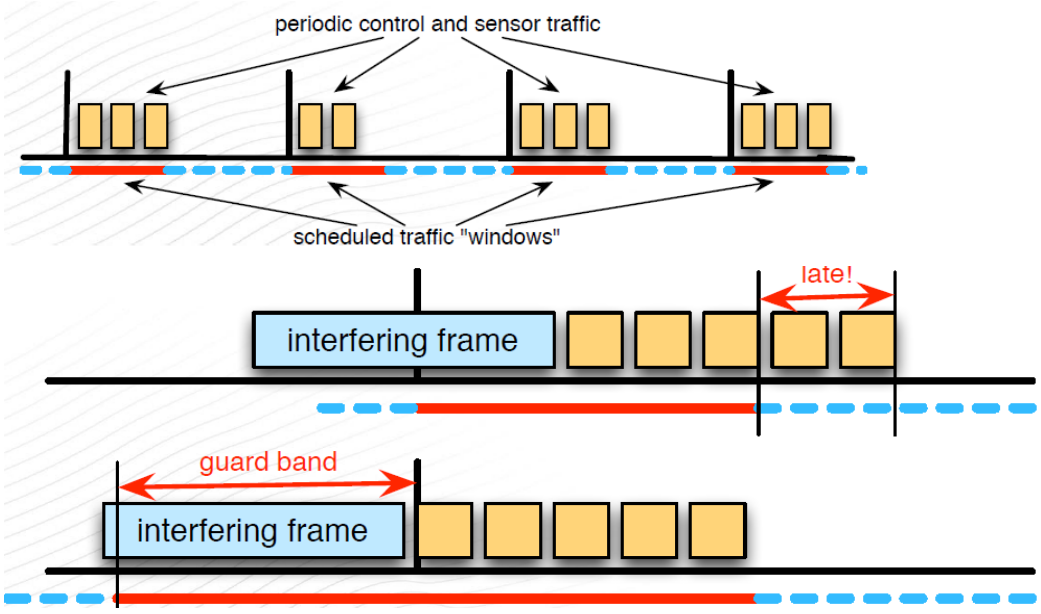
Block non-control traffic during particular windows of time to ensure that the egress port for a control stream is idle when the control traffic is expected.

Each egress port would have a separate schedule.

Nontrivial calculation in nontrivial networks

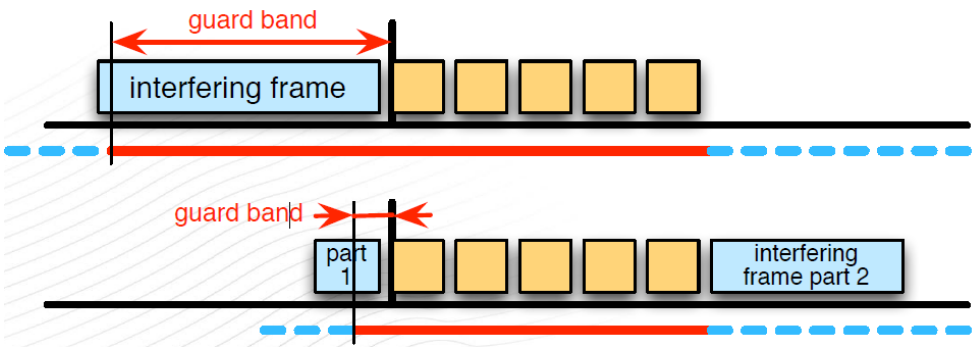
Requires a fully managed network.

This is a well-understood but difficult problem currently implemented in proprietary networks such as Siemens' "Profinet."



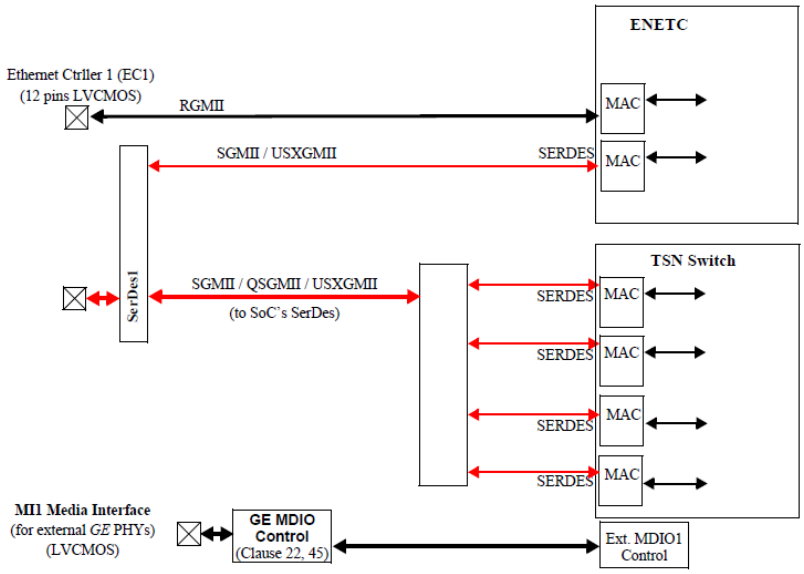
802.1Qbu/802.3br - Preemption

If preemption is used, the guard band needs to be only as large as the largest possible interfering fragment instead of the largest possible interfering frame.

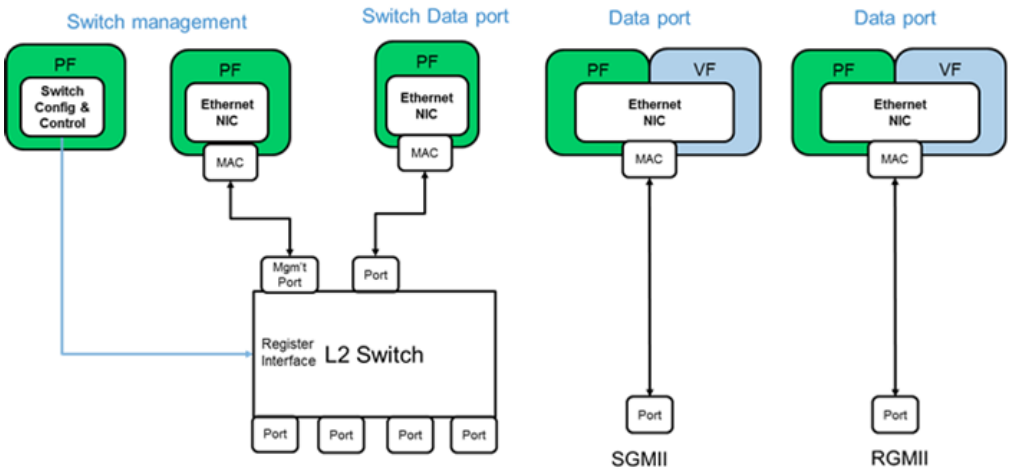


TSN in LS1028

The diagram below is intended to show the external-facing Ethernet interfaces, and therefore does not show the MAC-to-MAC connections between ENETC and the TSN Switch.

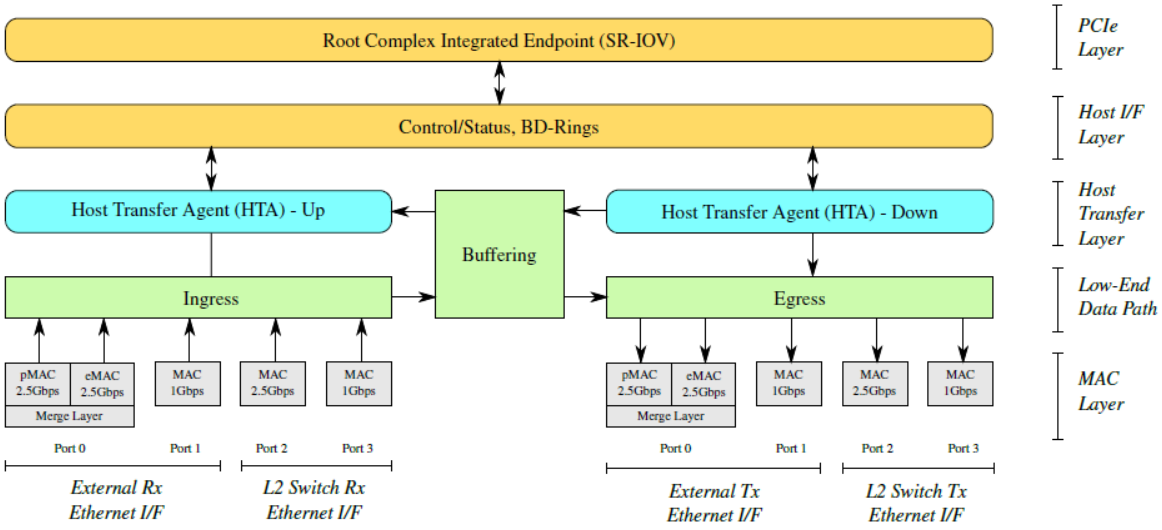


ENETC and switch management, SoC sees 4 different network interfaces (instances of our Ethernet controller) 2 of which are connected to ports of the switch using back-to-back MAC connections



The TSN Switch IP is a **6-port** TSN-capable Ethernet switch. 4 of the 6 ports operate at

speeds up to 2.5Gbps, for connection to external PHYs. 2 of the 6 ports are for connection to MACs of the ENETC IP (for packets terminated or initiated by the host and/or forwarded to/from ENETC MACs). One of these MACs to ENETC is for control traffic (operating at 1Gbps) and the other for data (operating at 2.5Gbps).
Autonomous switching is supported within the TSN switch.
ENETC TSN features



Using TSN features on LS1028ARDB

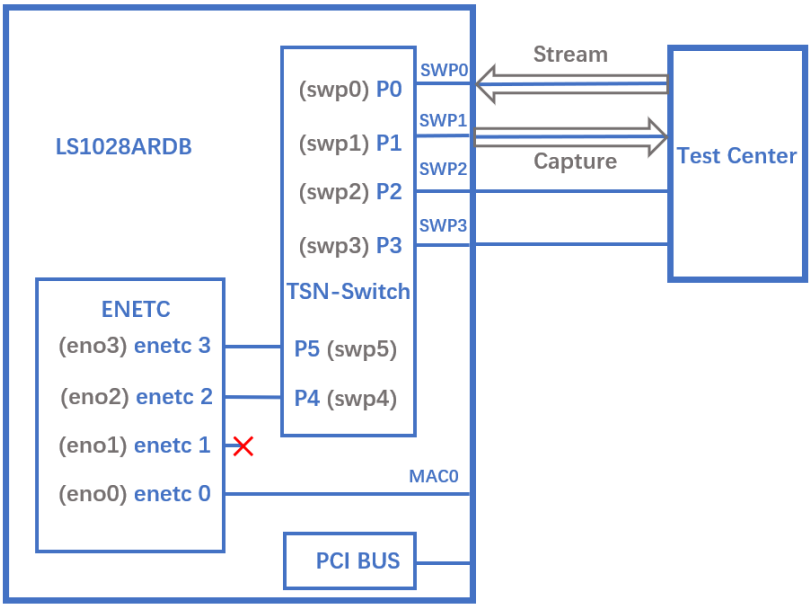
Tsntool is a tool to set the TSN capability of the Ethernet ports of TSN Endpoint and TSN switch.
Before compiling the Linux kernel, we need to configure it. In the kernel, select the configuration settings displayed below:

Symbol: TSN [=y]
| Type : boolean
| Prompt: 802.1 Time-Sensitive Networking support
| Location:
| -> Networking support (NET [=y])
| -> Networking options
| Depends on: NET [=y] && VLAN_8021Q [=y] && PTP_1588_CLOCK [=y]
|
Symbol: ENETC_TSN [=y]
| Type : boolean
| Prompt: TSN Support for NXP ENETC driver
| Location:
| -> Device Drivers
| -> Network device support (NETDEVICES [=y])
| -> Ethernet driver support (ETHERNET [=y])
| -> Freescale devices (NET_VENDOR_FREESCALE [=y])
| Defined at drivers/net/ethernet/freescale/enetc/Kconfig:41
| Depends on: NETDEVICES [=y] && ETHERNET [=y] && NET_VENDOR_FREESCALE [=y]

&& FSL_ENETC [=m] &&
TSN [=y]
│
Symbol: FSL_ENETC_PTP_CLOCK [=y]
│ Type : tristate
│ Prompt: ENETC PTP clock driver
│ Location:
│ -> Device Drivers
│ -> Network device support (NETDEVICES [=y])
│ -> Ethernet driver support (ETHERNET [=y])
│ -> Freescale devices (NET_VENDOR_FREESCALE [=y])
│
Symbol: FSL_ENETC_HW_TIMESTAMPING [=y]
│ Type : boolean
│ Prompt: ENETC hardware timestamping support
│ Location:
│ -> Device Drivers
│ -> Network device support (NETDEVICES [=y])
│ -> Ethernet driver support (ETHERNET [=y])
│ -> Freescale devices (NET_VENDOR_FREESCALE [=y])
|
Symbol: MSCC_FELIX_SWITCH_TSN [=y]
| Type : tristate
| Prompt: TSN on FELIX switch driver
| Location:
| -> Device Drivers
| -> Network device support (NETDEVICES [=y])
| -> Ethernet driver support (ETHERNET [=y])
| -> Microsemi devices (NET_VENDOR_MICROSEMI [=y])
| -> Ocelot switch driver (MSCC_OCELOT_SWITCH [=y])
| -> FELIX switch driver (MSCC_FELIX_SWITCH [=y])
| Defined at drivers/net/ethernet/mscc/Kconfig:38
Symbol: NET_PKTGEN [=y]
| Type : tristate
| Prompt: Packet Generator (USE WITH CAUTION)
| Location:
| -> Networking support (NET [=y])
| -> Networking options
| -> Network testing
| Defined at net/Kconfig:325
| Depends on: NET [=y] && INET [=y] && PROC_FS [=y]
Symbol: MSCC_FELIX_SWITCH_PTP_CLOCK [=y]
| Type : boolean
| Prompt: FELIX switch PTP clock support

| Location:
| -> Device Drivers
| -> Network device support (NETDEVICES [=y])
| -> Ethernet driver support (ETHERNET [=y])
| -> Microsemi devices (NET_VENDOR_MICROSEMI [=y])
| -> Ocelot switch driver (MSCC_OCELOT_SWITCH [=y])
| -> FELIX switch driver (MSCC_FELIX_SWITCH [=y])
| Defined at drivers/net/ethernet/mscc/Kconfig:38
| Depends on: NETDEVICES [=y] && ETHERNET [=y] && NET_VENDOR_MICROSEMI
| Selects: PTP_1588_CLOCK [=y]

Basic TSN configuration examples on the switch
The following figure describes the setup for Qbv test on LS1028ARDB.



Use the set of commands below for basic gate closing.
echo “t0 00000000b 20000” > qbv0.txt
#Explaination:
 # 'NUMBER' : t0

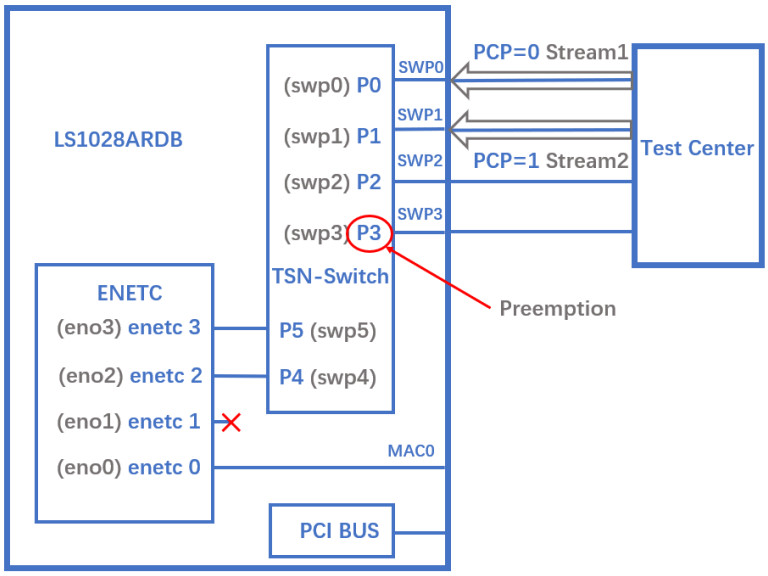
 # 'GATE_VALUE' : 00000000b
 # 'TIME_LONG' : 20000 ns
cp libtsn.so /lib
./tsntool
tsntool> verbose
tsntool> qbvset --device swp1 --entryfile ./qbv0.txt
#Send one broadcast frame to swp0 from TestCenter.
ethtool -S swp1
#Should not get any frame from swp1 on TestCenter.
echo “t0 11111111b 20000” > qbv0.txt

tsntool> qbvset --device swp1 --entryfile ./qbv0.txt
#Send one broadcast frame to swp0 on TestCenter.
ethtool -S swp1
#Should get one frame from swp1 on TestCenter.

For the basetime test, first get the current second time:
#Get current time:
tsntool> ptptool -g -d /dev/ptp1
#add some seconds, for example you get 200.666 time clock, then set 260.666 as result
tsntool> qbvset --device swp1 --entryfile ./qbv0.txt --basetime 260.666
#Send one broadcast frame to swp0 on the Test Center.
#Frame could not pass swp1 until time offset.

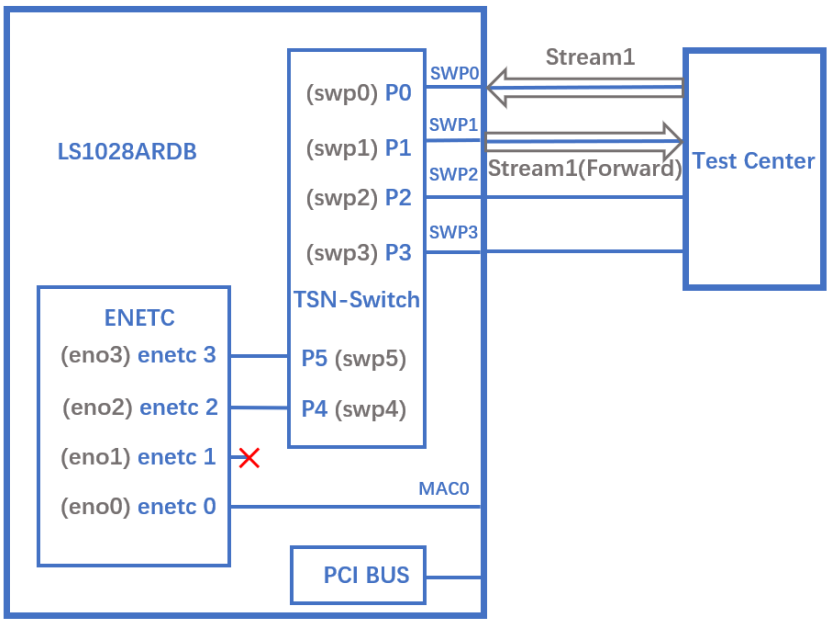
Use the following commands for the QBv performance test:
cat > qbv5.txt << EOF
t0 11111111b 1000000
t1 00000000b 1000000
EOF
qbvset --device swp1 –entryfile
qbv5.txt

Qbu test



Set queue 1 to be preemptable.
tsntool> qbuset --device swp3 --preemptable 0x02
2. Send two streams from TestCenter, then check the number of additional mPackets transmitted by PMAC:
devmem 0x1fc010e48 32 0x3 && devmem 0x1fc010280

Qci test case



Stream gate control

1. Use the following commands for stream gate control:

echo "t0 1b 3 50000 200" > sgi.txt

tsntool> qcisgiset --device swp0 --enable --index 1 --initgate 1 --initipv 0 --gatelistfile sgi.txt --basetime 0x0

2. Send one frame on TestCenter.

ethtool -S swp1

Note that the frame could pass, and green_prio_3 has increased.

3. Now run the following commands:

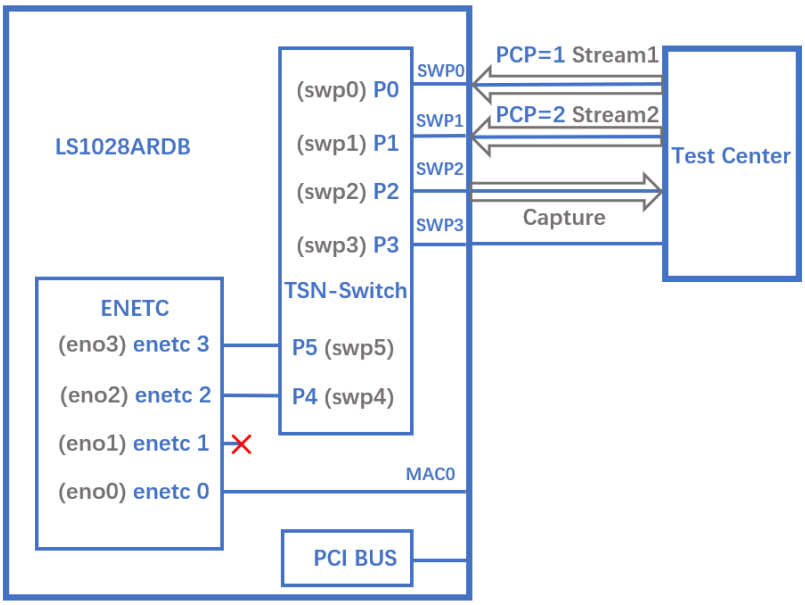
echo "t0 0b 3 50000 200" > sgi.txtx

tsntool> qcisgiset --device swp0 --enable --index 1 --initgate 1 --initipv 0 --gatelistfile sgi.txt --basetime 0x0

4. Next, send one frame on TestCenter.

ethtool -S swp1

Qav test case



1. Set the percentage of two traffic classes:

tsntool> cbsset --device swp2 --tc 1 --percentage 20

tsntool> cbsset --device swp2 --tc 2 --percentage 40

2. Send two streams from Test center, then check the frames count.

ethtool -S swp2

Note that the frame count of queue1 is half of queue2.

Stream rate must lager than bandwidth limited of queue.

3. Capture frames on swp2 on TestCenter.

 # The Get Frame sequence is: (PCP=1), (PCP=2), (PCP=2), (PCP=1), (PCP=2), (PCP=2),…