Time-Aware Traffic Shaper Application - All Text

Discord server: https://bit.ly/OpenEduHub

1. Introduction

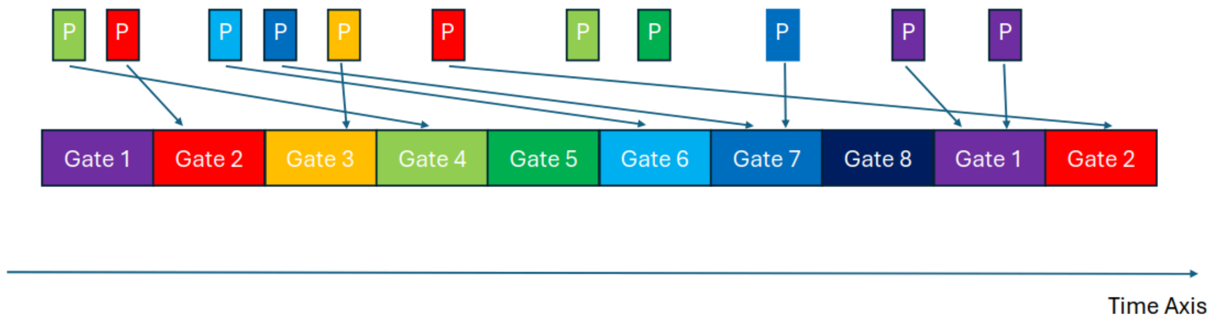
TSN - Time Sensitive Network

TSN is a technology that started to receive more and more attention from networking engineers as it guarantees determinism and ensures stable throughput in Ethernet networks. But accomplishing time synchronization between two or more endpoints is not an easy thing, not when dealing with time critical communications where nanoseconds precision is required.

In this Keysight Challenge we will be looking at some of the problems which TSN networking tries to solve and get familiar with basic TSN concepts.

QBV

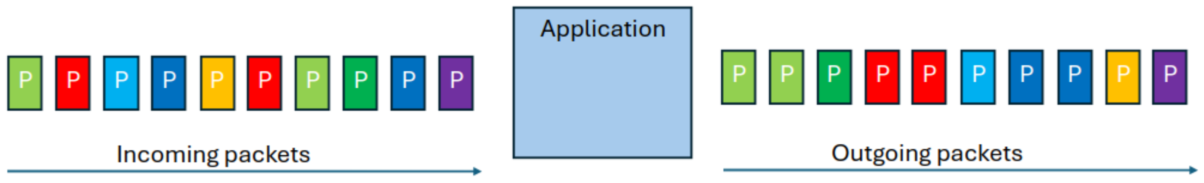
The purpose of the IEEE 802.1**Qbv** time-aware scheduler is to send packets of different priorities in different time slices (called gates). Only when a gate for a specific traffic priority is opened, only then the packets associated with that priority are allowed to be sent on the wire. The standard defines 8 gates, and usually the priority of the packets can be the VLAN priority bit in the VLAN header.



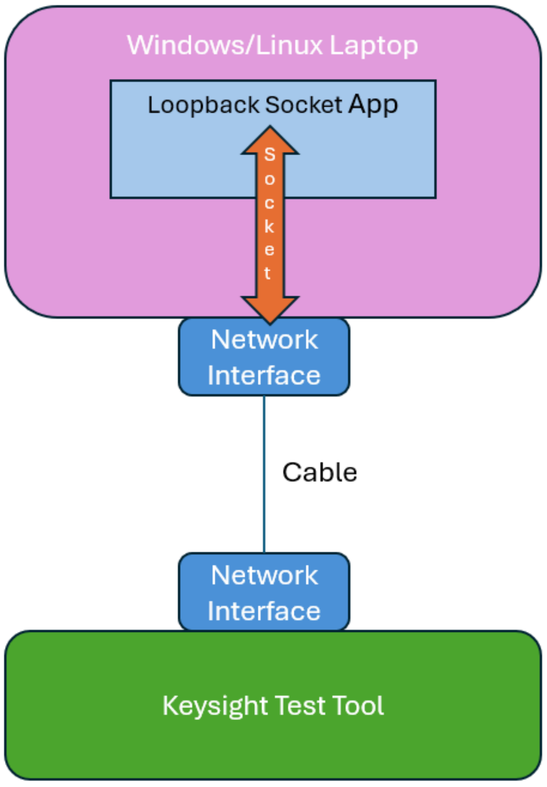
In the above picture we can see how different packets (using different colors for their specific priorities) will only be scheduled for transmission inside their designated gates.

2. Challenge

The challenge is to create an application which will receive raw traffic from a network socket, classify the packets and buffer them until the corresponding gate opens for transmission.

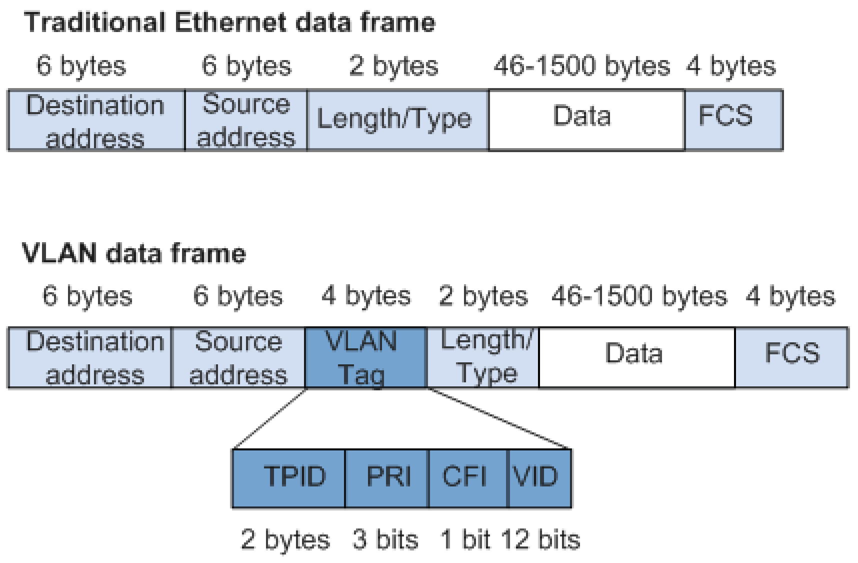


The outgoing traffic will be forwarded to either the same socket or to another socket opened on a second network interface (we will provide an USB network dongle for a second network interface).



The incoming traffic will be classified based on the VLAN priority tag. The priority tag is in the "PRI" field as shown below.

The start of gate 0 will be set to the moment when the first packet with a VLAN tag will be received by the app.



Notes:

- Any packets with no VLAN encapsulation should be ignored by your app

3. Requirements

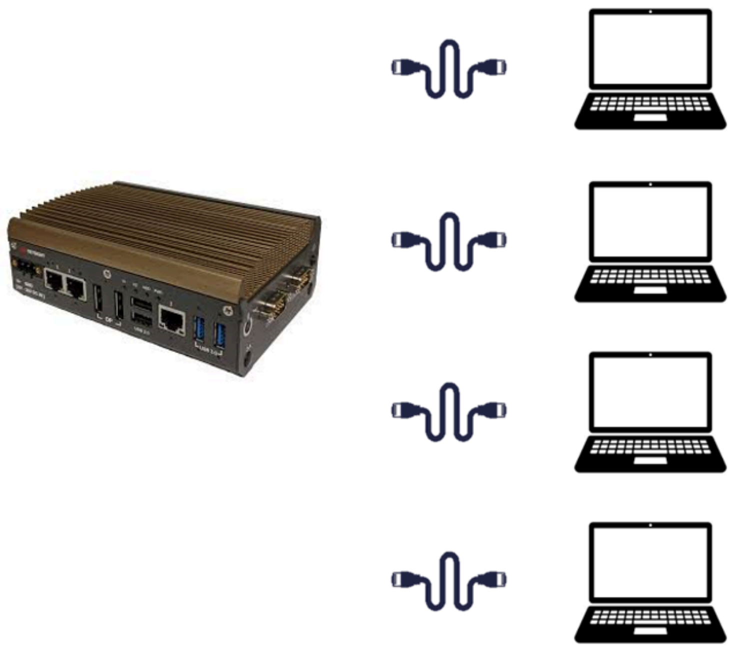
- There will be 8 gates, each gate with an open duration of 100ms
- App should be able to bind to a specific interface (e.g. eth0) by passing the "-i $INTERFACE" argument on the command line
- Maximum frame size set to 1518 bytes
- Using C/C++/Python/Rust

4. Bonus

- Use libpcap to read the packets (binding to an interface) and parsing them
- Periodically read the time while sending packets out, to see if the gate closed

5. Testing

Keysight will bring two or three Novus Mini chassis. Each chassis will have four ports.



- Port 1 - Sending 1 Packet Per Second (PPS), VLAN Priority 0
    - The app should send back the packet instantly, as Gate 0 inside the app opens when the first packet is received
- Port 2 - Sending 1PPS, VLAN Priority 5
    - The app should send back the packet after 500ms, because Gate 5 will open after the previous Gate 0 - 4 and each gate is 100ms long
- Port 3 - Sending 2PPS in a burst, one VLAN Priority 5 and one VLAN Priority 2. The app should send back VLAN Priority 2 after 200ms and VLAN Priority 5 after 500ms
- Port 4 - Sending 1Mbps traffic (1518 length) to see if the app handles buffering well