Task 4: Modify the TSN simulation models

There are many TSN models released by different organizations or individuals, we will choose one of them. We will provide all the source codes of TSN simulation model for you. We only need the "Time-aware shaper" module of TSN simulation model.

You only need to rewrite the configuration interface, do some version adaptation or function fine-tuning with the above model, and disguise the TSN model as the model we developed.

Details will be transferred to you later.

The following contents are updated on Jan. 29th.

INET 4.5.2 provides the IEEE 802.1Qbv simulation models, which is also referred as "TSN Time-aware shaper (TAS)". Link:
https://inet.omnetpp.org/docs/showcases/tsn/trafficshaping/timeawareshaper/doc/index.html

However, the configuration interface of the current version of the TAS model is not a standard one, so the task 4 includes:

1. Modify the configuration interface according to our requirements.
2. Disguise the TAS model as our own developed model, i.e., all the source code about the TAS model needs to be placed in the */src/tsn* directory, and also the file names and function names can be modified appropriately.

Requirement:

However, the configuration interface of the current version of the TAS model is not a standard one, so the task 4 includes:

8.6.8.4 Enhancements for scheduled traffic

A Bridge or an end station may support enhancements that allow transmission from each queue to be scheduled relative to a known timescale. In order to achieve this, a transmission gate is associated with each queue; the state of the transmission gate determines whether or not queued frames can be selected for transmission (see Figure 8-14). For a given queue, the transmission gate can be in one of two states:

a) *Open*: Queued frames are selected for transmission, in accordance with the definition of the transmission selection algorithm associated with the queue.
b) *Closed*: Queued frames are not selected for transmission.

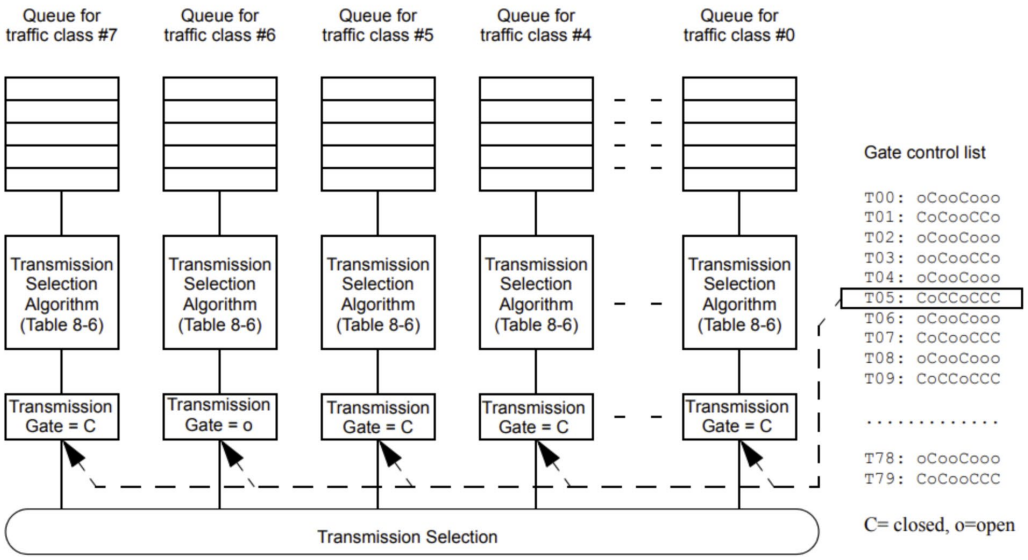


Figure 8-14—Transmission selection with gates

Section 1:

I will briefly introduce the concept of TAS. As shown in the figure, TAS has 8 queues, each of which carries (stores) different traffic. The traffic has different priorities ranging from 0 to 7.

a) Traffic with priority 7 is stored or forwarded by queue 7.
b) Traffic with priority 6 is stored or forwarded by queue 6.
c) …. …
d) Traffic with a priority of 0 will be stored or forwarded by queue 0.

This mechanism is called "flow-queue mapping".

Each queue has a gate, which is controlled by a GCL. the GCL is a control list and is also the output of the TSN scheduling algorithm. TAS is located at egress of TSN switch or end-station, so the GCL controls the traffic shaping at egress.

The GCL consists of two components, the time interval and the gate state. Let's take an example of GCL:

1111 1111,t0.

0000 0000,t1.

1000 0000,t2.

0111 1111,t3.

The first column belongs to the gated state, which consists of 8 bits of binary. 1 means the gate is open, 0 means the gate is closed. 8-bit gate states correspond from left to right to queue 7, queue 6, queue 5... and queue 0. For example, "1000 0000" opens queue 7 and closes the doors of the other queues.

The second column represents the time interval, which means how long the state of the gate will be in effect. For example, "0111 1111,t3;" means that closing the gate of queue 7 and opening all the other gates will last for t3.

The GCL is executed periodically, which we call a hyperperiod. the cumulative sum of t0, t1, t2, and t3 is equal to the length of the hyperperiod. We assume that the cumulative sum of t0, t1, t2, t3 is equal to 100 milliseconds, which means that this GCL is executed repeatedly every 100ms. ***Please note that the unit of time-interval should be set as nanosecond.***

Meanwhile, different interface of a TSN switch requires different GCL. TSN end-station also requires a GCL.

Section 2:

There are two key parameters for INET TAS model, duration and offset. They are not GCL, so you need to implement the GCL configuration interface. And the implementation should follow the guide as specified in Section 1.

Also, you should check if the “traffic-queue mapping” works properly.

I found that GCL interface has been already implemented. So your workload will be greatly reduced. You are free to referred the implemented modules, please see the following posts:

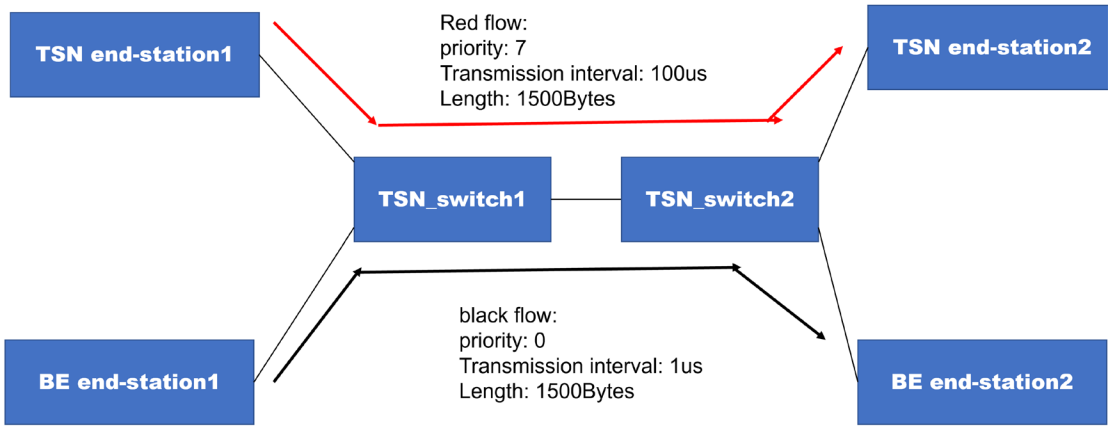
https://github.com/inet-framework/inet/issues/897

https://github.com/inet-framework/inet/issues/771

I am sure that these two posts will be greatly helpful.

Section 3: Test

The simulation scenario should be established as below:



The GCL should be configured for the egress of TSN_swtich1, which is connected to the TSN_switch2.

The GCL content is: The units (nanoseconds, ns) should be omitted.

01111 1111, 8500ns

0000 0000, 12240ns

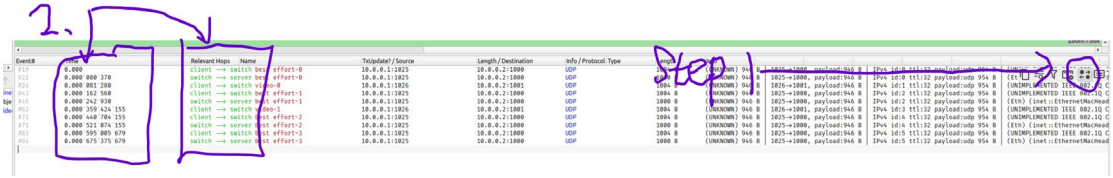
1000 0000, 12336ns

01111 1111, 66924ns

Hyper-period = 8500 + 12240 + 12336 + 66924 = 100 us, which is equal to the transmission interval of red flow.

Two key-indicators should be achieved:

1. Observing the window of the simulation (the figure is an example), the red stream should be sent from TSN_switch1 to TSN_switch2 at moment 20.74us.



2. The end-to-end delay of red flow should be a line, without any jitter.