CSE 5820 Term Project: Learning Online Schedule for 802.1 Qbv Time-Sensitive Network

Chuanyu Xue

May 7, 2022

1 Introduction

Time-Sensitive Network (TSN) is an extension of Ethernet to achieve deterministic and high reliable communication[1]. One of the most important enhancements is traffic scheduling defined by IEEE 802.1Qbv in Time-Aware Shaper (TAS). The basic idea in TAS is separating traffic into multiple queues and scheduling the transmission period for each queue with a Gate Control List (GCL). Thus the scheduling task for TAS can be summarized as queue assignment and GCL configuration. However, such an off-line generated schedule may suffer from unscheduled traffic or interleaved traffic leading to catastrophic effects. To solve above problem, this project aims to develop an online intelligent agent to provide a robust schedule in deterministic network.

2 Problem formulation

In this project, we choose to learn the schedule for a single egress port. This should be extent as a distributed approach to support the online scheduling for a whole network in the future work. For this egress port, we assume it has **N** queues with maximum queue size **S**. A set of periodic traffic $s_i \in S$ are transmitted through the egress port with properties in terms of transmission duration l_i, queue assignment q_i and period t_i. Each traffic is required to be received within end-to-end delay d_i.

The model of GCL configuration can be abstracted as a cyclic schedule of **K** transmission windows. We define the ϕ_i as the start time and τ_i as the end time of the transmission window assigned to traffic s_i. The period of schedule is set to the least common multiple $T = LCM(t_i)$. Our goal is to learn this cyclic schedule only from the observation without knowing l_i and t_i.

3 Method

3.1 Framework

First, we introduce the state, action and reward representation:

- **State.** Our state is defined as the start and end time of each traffic window for each queue.

- **Action.** We define $a_i^{\phi} = \{-1, 0, 1\}$ as moving traffic s_i's window start time left or right a time granularity or remaining its state, so forth for a_i^{τ}.
- **Reward.** We define the number of frames received within deadline w as reward.

3.2 Agent design

Here, we use Q-learning to estimate the reward based on the state and action. Following the Bellman Equation, the agent can find the action leading to the maximum reward in the long run.

$$q(s_t, a, t) = r_{a,t+1} + \gamma \max q\left(s_{a,t+1}, a', t+1\right) \tag{1}$$

To reduce the number of states, we decoupled the dependency of each frame, window start time, and window end time. Specifically, each frame's window start time and end time has its own Q-table instead of share a large one. Otherwise, the number of states would be huge as nFrame $\times$ nPotentialStarts $\times$ nPotentialEnds. Another general idea to solve this problem is to introduce deep Q-table learning for future work[2].

4 Experiment

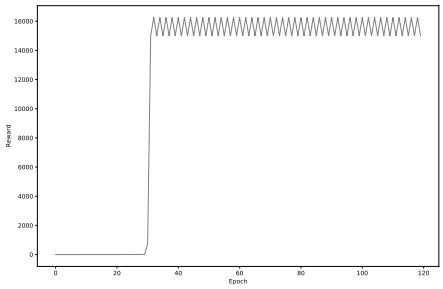
This experiment is conducted in a Python simulation environment[3]. For the environment, we set the each episode contains 10000 iterations and each iteration runs the system 100 nanoseconds. For the Q-learning algorithm, we set the learning rate $\alpha = 0.9$, $\delta = 0.9$, and $\epsilon = 0.8$. The network topology is a simple linear structure with 1 talker, 1 bridge, and 1 listener. The initial schedule is randomly generated with length 100 nanoseconds for each traffic flow. We varies the number of queues within bridge and generates random traffic flows to validate if the schedule can be learned.

To demonstrate the difference between ideal environment and real-world, we creates two types of traffic flows:

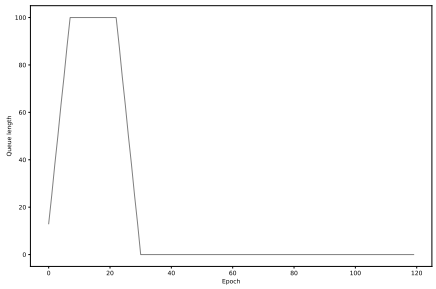
- **Constant flow** is the traffic with constant arrival time.

- **Dynamic flow** is the traffic effected by random jitter in real-world with nondeterministic arrival time. Here, we randomly advance or delay the release time of this type of flow from 0 to 500 nanoseconds.

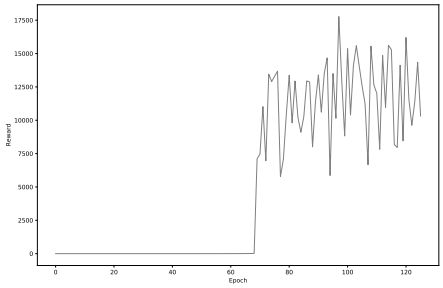
Figure 1 is the experiment result that each device is associate with 1 egress queue and only a single traffic is within network. From figure 1.(b) and figure 1.(d), we can see the schedule for both constant traffic and dynamic traffic is well learned, as the queue size is converged and stabilized to 0 in the end. By comparing figure 1.(a) and figure 1.(c), it is obvious that the schedule in real-world takes more iterations to be learned.



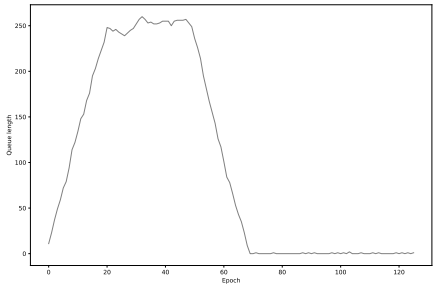
(a) Constant reward



(b) Constant Q-size



(c) Dynamic reward

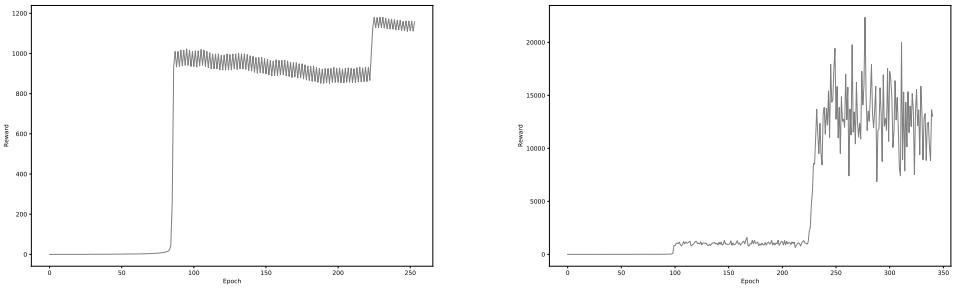


(d) Dynamic Q-size

Figure 1: Simulation network with 1 queue and 1 traffic.

To understand how the proposed algorithm deals with traffic collision, we also conduct the experiments on the network with multiple queues and multiple flows. As expectation, the size of both queues are stabilized to 0

after the 86 and 224 epoch for constant flow, after 100 and 321 epoch for dynamic flow[4, 5].



(a) Constant reward (b) Dynamic reward

Figure 2: Simulation network with 2 queue and 3 traffic.

Interestingly, we found the algorithm merges the transmission windows for the flows in the first queue to reduce jitter demonstrated in Figure 3. Because traffic 1 and traffic 2 share a same transmission window, when one traffic arrives late, the other one can still utilize the window to finish transmit. It can also explain why the reward can still go up after being stable in Figure 2. This is also a more advanced scheduling approach which is so called "time-based" scheduling.

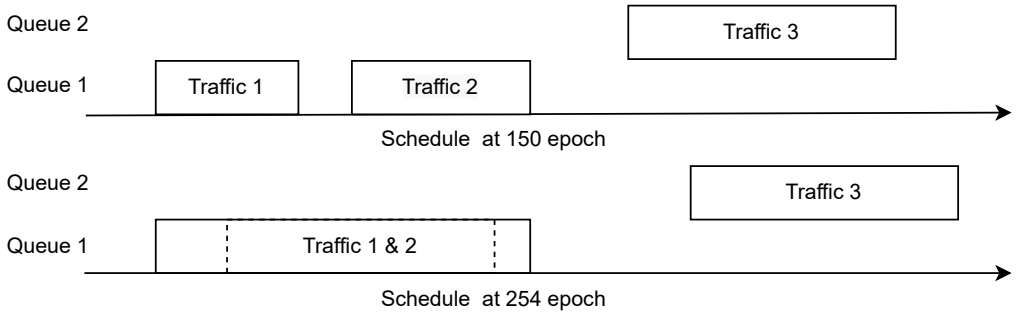


Figure 3: Overlapped schedule to improve reward

References

[1] "Time-Sensitive Networking (TSN) Task Group." [Online]. Available: https://1.ieee802.org/tsn/

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] "GitHub - ChuanyuXue/CSE-5820-CourseProject: The term project of CSE 5820 Advanced machine learning course." [Online]. Available: https://github.com/ChuanyuXue/CSE-5820-CourseProject

[4] R. S. Oliver, S. S. Craciunas, and W. Steiner, "Ieee 802.1 qbv gate control list synthesis using array theory encoding," in *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2018, pp. 13–24.

[5] N. Reusch, L. Zhao, S. S. Craciunas, and P. Pop, "Window-based schedule synthesis for industrial ieee 802.1 qbv tsn networks," in *2020 16th IEEE International Conference on Factory Communication Systems (WFCS)*. IEEE, 2020, pp. 1–4.