Proceedings of 2018 IEEE
International Conference on Mechatronics and Automation
August 5 - 8, Changchun, China

A Time-sensitive Networking (TSN) Simulation Model Based on OMNET++

Junhui Jiang, Yuting Li, Seung Ho Hong*
Department of Electronic Systems Engineering
Hanyang University
Ansan, 15588, Korea
Email: 0229ender@gmail.com, yuting2427@gmail.com,
shhong@hanyang.ac.kr*

Aidong Xu, Kai Wang
Key Laboratory of Networked Control Systems
Shenyang Institute of Automation,
Chinese Academy of Sciences
Shenyang, China
Email: {xad, wangkai}@sia.cn

***Abstract* - Industrial and automation control systems require that data be delivered in a highly predictable manner in terms of time. Time-sensitive Networking (TSN), an extension of the Ethernet, is a set of protocols developed and maintained by the IEEE 802.1 Task Group; the protocols deal with time synchronization, traffic scheduling, and network configuration, etc. TSN yields promising solutions for real-time and deterministic networks. Here, we develop a TSN simulation model based on OMNET++; we model a TSN-enabled switch that schedules traffic using gate control lists (GCLs). Simulation verified that the model guaranteed deterministic end-to-end latency.**

Index Terms – Time-sensitive Networking (TSN), OMNET++, Simulation Model, Deterministic, Real-Time, Schedule Traffic

I. INTRODUCTION

The Ethernet is the most widely used general-purpose communication protocol and has excellent bandwidth, scalability, and compatibility [1]. However, it is difficult to meet the real-time deterministic requirements of industrial control systems: although several Ethernet extensions have offered deterministic solutions (e.g., PROFINET, EthernetCAT, and TTEthernet), they are either not mutually compatible or do not integrate with standard Ethernet networks and devices. To address this issue, the IEEE 802.1 Task Group [2] has begun to develop a common real-time Ethernet standard termed the IEEE 802.1 Time-sensitive Networking (TSN) Standard, with a specific focus on industrial automation and automotive networks. The TSN seeks to provide deterministic Ethernet capabilities, thereby enabling automation systems to converge to a single and interconnected network supporting both time-critical and non-time-critical data communication without disrupting time-critical control tasks.

The TSN consists of a set of IEEE 802 Ethernet sub-standards (time synchronization via IEEE 802.1ASrev [3] and IEEE 1588; traffic-shaping via IEEE 802.1Qbv [4]; a centralized network configuration (CNC) using IEEE 802.1Qcc [5]; and time-based ingress policing via IEEE802.1Qci [6]). Using these standards, the TSN will provide a deterministic service guaranteeing packet transmission with a bounded low latency, low packet delay variation, and low packet loss [2]. Thus, the TSN will be of great benefit to industrial control and automation.

In this context, to increase bandwidth utilization, time-critical control data must be transmitted in a time-critical manner with other classes of traffic within the same network. Here, the issue concerns how to schedule traffic to ensure real-time transmission of time-critical traffic. Prioritization was first proposed to solve the issue, but “assigning priorities” alone is insufficient to address the real-time requirements of critical traffic. If low-priority traffic is being transmitted, high-priority traffic cannot access the transmission medium until the existing transmission is complete, creating uncertainty. Thus, 802.1Qbv proposed a method termed Enhancements for Scheduled Traffic [4], which allows a switch to use gate control lists (GCLs) to control the gate state (open or closed) of the corresponding queue in the output port of a switch. Thus, time-triggered (TT) traffic and best-effort (BE) traffic can be scheduled using a globally synchronized clock to ensure real-time, predictable, deterministic, network transmission. In terms of time synchronization, TSN uses IEEE 802.1ASrev, based on the precision time protocol (PTP) (with an accuracy of nanoseconds [3]), to synchronize the global network.

To date, few studies have built TSN simulation models [7][8]. Paper [7] dealt with Audio-Video-Bridging (AVB) traffic, BE traffic, and scheduled traffic in an AVB network, rather than a TSN network. Paper [8] presented a simulation framework featuring frame pre-emption. However, time synchronization was not considered; only non-time-based features were explored. Here, we develop a TSN simulation model based on OMNET++ that schedules network traffic in a time-based manner using an advanced concept, namely GCLs [4]. This simulation model can be verified that the issue of real-time transmission of time-critical traffic in a hybrid network can be addressed by sets of TSN protocols. Our main contributions are as follows: we first build a TSN-enabled switch (SW) that can read GCLs and operate the gate state of the corresponding queue accordingly; second, we build a network topology to verify the switch model; and third, we calculate GCLs based on the network topology and host parameters. Last, we note that many papers [9]–[12] on traffic scheduling algorithms and GCL synthesis have appeared; our model can be used to evaluate the feasibility of those algorithms, e.g., the synthesized GCL can be input into this simulation model and get the verification results.

The remainder of this paper is organized as follows. Section II introduces the overall development environment. Section III presents the detailed development process. Section IV evaluates the simulation model, and Section V concludes the paper.

II. Overview of the OMNET++ Simulation Tool

OMNET++ is an open-source, discrete event simulator that has an extensible, modular, component-based, C++ simulation library and framework, primarily used to build network simulators [13]. The advantages of OMNET++ include its open source nature, ease of operation, and extensibility. OMNET++ also provides numerous extended frameworks, not only supporting many Ethernet protocols and features, but also many IEEE standards. Here, we use two of these frameworks to develop our TSN simulation model, as follows.

We first use an INET framework (an open-source model library of Internet stack, wired, and wireless link-layer protocols [14]). We also employ a Communication over Real-time Ethernet for INET (CoRE4INET) framework, which is an extension of the INET framework allowing event-based simulation of the real-time Ethernet [15]. Currently, CoRE4INET supports the IEEE 802.1Q standard and also provides an example network paradigm for this standard. Fig. 1 shows the developmental environment of our model. In addition to the CoRE4INET and INET frameworks, we applied IEEE 802.1Qbv to schedule traffic and IEEE 802.1QAS for time synchronization.

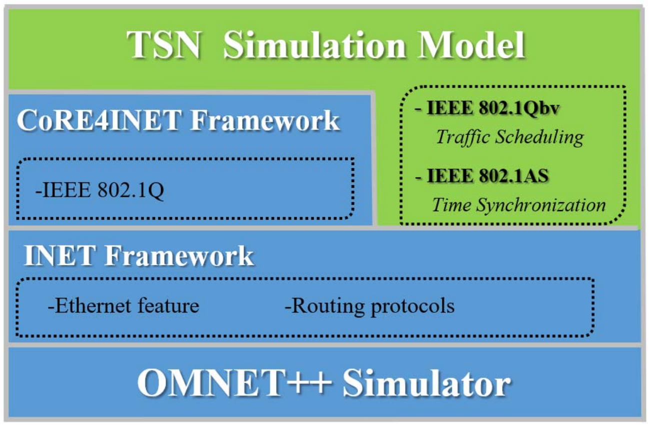


Fig. 1 The developmental environment.

III. Development of the TSN Simulation Model

In this section, the development of the TSN simulation model will be described in two steps: the first is the development of sub-models (the SW and network topology models) and the second is GCL synthesis by reference to traffic parameters and topology. We use GCLs to control the gate state (open or closed) of each queue in the output port of a switch [4].

A. Switch Model

Using the extension to the IEEE 802.1Q switch provided by the CoRE4INET framework, we developed our SW model with the aid of IEEE 802.1Qbv, which not only allows different queues to select different traffic based on their priorities, but also transmission of specific traffic classes at specified times. Fig. 2 shows the function block of our model; both read file and transmission functions are evident. Using these, the switch controls the gate states of queues in the output port by reference to the GCLs, scheduling traffic and guaranteeing deterministic end-to-end delay.

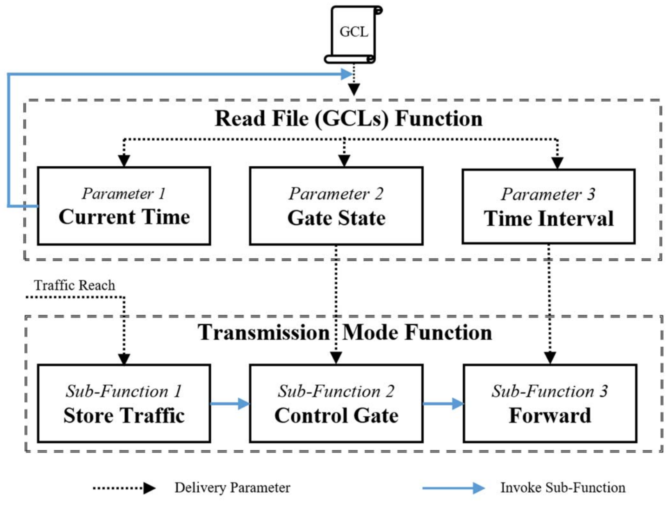


Fig. 2. The function block of the Time-sensitive Networking (TSN)-enabled switch model.

1) Read Gate Control Lists: Table II gives the details of the GCLs utilized in the simulation, and the specific parameters for the three transmission modes ("Protected Window", "Unprotected Window", and" Guard Band") defined by IEEE 802.1Qbv. As shown in Fig. 2, the SW first reads three parameters ("Current Time", "Gate State", and "Time Interval"). The first parameter "Current Time" stands for time-instant, which signifies the moment SW reads GCLs. When the earliest traffic reaches the SW, the SW will be triggered to read the GCLs. Then, the SW will read the GCLs at the time-instant indicated by the following Current Time, i.e., the next entry's parameters. The updated value of Current Time is obtained by accumulating the previous values of Current Time and Time Interval. Refers to the other two parameters when performing the steps below.

2) Traffic Transmission: The transmission function is realized in the MAC layer of the Ethernet switch of the INET framework and is composed of three sub-functions ("Store Traffic", "Control Gate", and "Forward"), as shown at the bottom of Fig. 2. The first sub-function ("Store Traffic") stores the different kinds of traffic when the traffic reach SW. Based on the IEEE 802.1Q standard, CoRE4INET stores traffic in eight queues based on eight priorities [16] (i.e., queue #1 contains traffic of priority 1). The second sub-function ("Control Gate") checks the current gate state (open or closed) and compares it with the read of the Gate State parameter. If they are not the same, the current gate state will be changed, preferring the latter value. The Gate State is an octet, each bit representing the state for a traffic class; the MS bit corresponds to traffic class 7 and the LS bit to traffic class 0. Each gate has only two states: open "1" or closed "0". The third sub-function forwards traffic based on the gate state, constrained by the Time Interval.

Fig. 3 shows a flow chart of gate operation during traffic transmission (sub-function 3: Forward). Specifically, if the gate is closed, the switch will check whether any packet is currently being forwarded; if it is, the switch will allow forwarding to continue until completion, but will stop forwarding of later packets; otherwise, the switch will stop forwarding immediately. However, if the gate is open, the switch will check whether any packets are in the relevant queue. If there are, the switch will forward them immediately; otherwise, the packets will be stored.

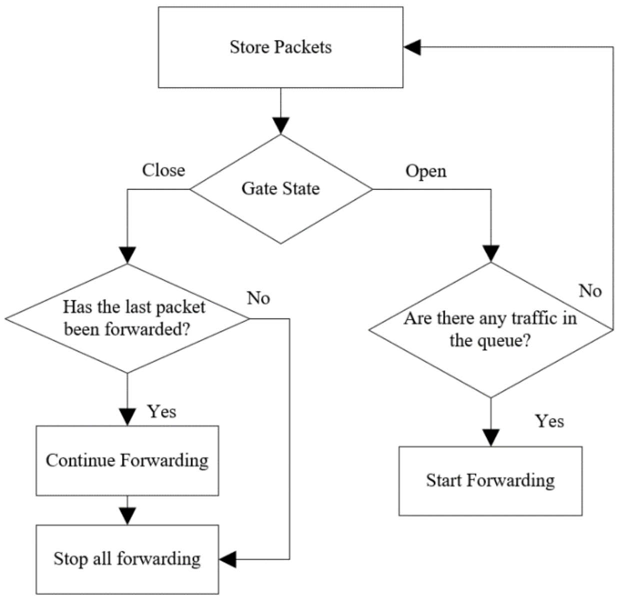


Fig. 3 Flow chart of gate operation during traffic transmission.

B. Architecture

TSN guarantees real-time deterministic transmission of time-triggered traffic in a mixed traffic scenario [2] that combines critical and BE traffic flows. However, under worst-case conditions (i.e., the physical link is completely blocked), can TSN still provide deterministic services for time-triggered traffic? To explore this issue, and to test our SW, we built a TSN topology involving mixed traffic. As shown in Fig. 4, two SWs are TSN network elements, and the other nodes are end-points including a TSN talker and listener, a BE sender (BS) and receiver (BR), and a huge traffic generator (TG) and traffic receiver (TR). The green line shows the direction of data flow. The nodes are connected with 1-Gbit/s full duplex physical links; the nodes can communicate in both directions. The characteristics of traffic sent by the TSN talker, the BS, and the TG are shown in Table I; the time-triggered and BE traffic are of the same size and have the same transmission interval. Thus, below, we compare their end-to-end delays when they are affected by TG traffic. The principal role of the TG is to generate huge packets transmitted at $\geqslant$ 1-G bits/s, thus blocking the network. This is the worst-case scenario; we use it to test if the TSN switch can cope or not. In other words, if the switch works appropriately, time-triggered traffic sent from the TSN talker will be sent to the destination after a deterministic end-to-end delay; otherwise, the end-to-end delay cannot be guaranteed.

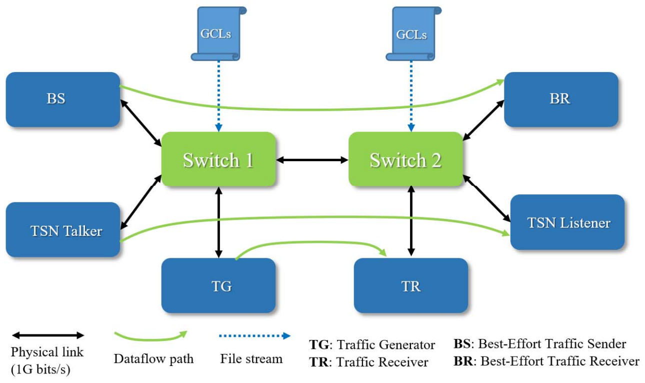


Fig. 4. Topology of the TSN simulation model.

TABLE I
TRAFFIC PARAMETERS

Type	Traffic Type	Interval	Priority	Payload Size (bytes)
TSN Talker	Time-triggered Traffic	1ms	3	1,000
BS	Best-effort Traffic	1ms	1	1,000
TG	Interference Traffic	1µs	0	1,200

TSN, Time-sensitive networking; BS, best-effort sender; TG, traffic generator

C. Network Schedule Calculation

Here, we explain in detail how to synthesize GCLs by reference to network topology and traffic parameters. GCLs can ensure that, at specific times, only one specified traffic class (or a set of classes) can access certain channels. We use an octet to present the 8 queues' state (open or closed) [4]. As mentioned, different queues will store different traffic based on their priority. In our simulation case, we have three different kinds of traffic and use three specified queues to store them: queue #3, queue #1, and queue #0 will respectively store time-triggered traffic, best-effort traffic, and interference traffic (TG traffic). The three coloured rectangles in Fig. 6 describe the three traffic are stored in the corresponding queues. To ensure real-time deterministic transmission of time-triggered traffic in a hybrid network (containing both time-triggered and BE traffic), the GCLs are used to create three transmission modes (Protected Window, Unprotected Window, and Guard Band). As shown in Fig. 5, the switch executes the three transmission modes in a time-sensitive manner. The green rectangle in Fig. 5 is the first transmission mode (Protected Window), used for transmission of time-triggered traffic only and the other kinds of traffic will not be transmitted during this period [4]. Thus, time-triggered traffic is not influenced by other traffic classes. In such a period, only time-triggered traffic is allowed to go through, so we set Gate State to "0000 1000", i.e., only allow the gate of queue #3 to open to transmit the time-triggered traffic. Fig. 6 shows this "Protected Window" transmission

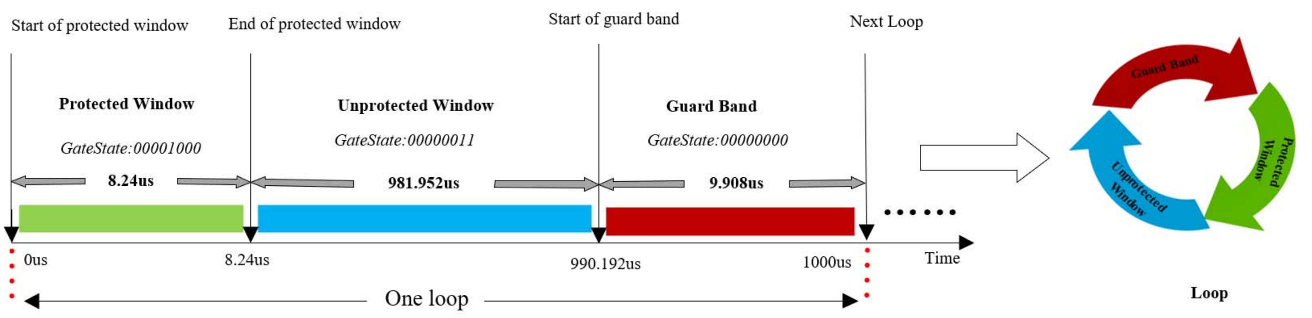


Fig. 5 The channel transmission mode.

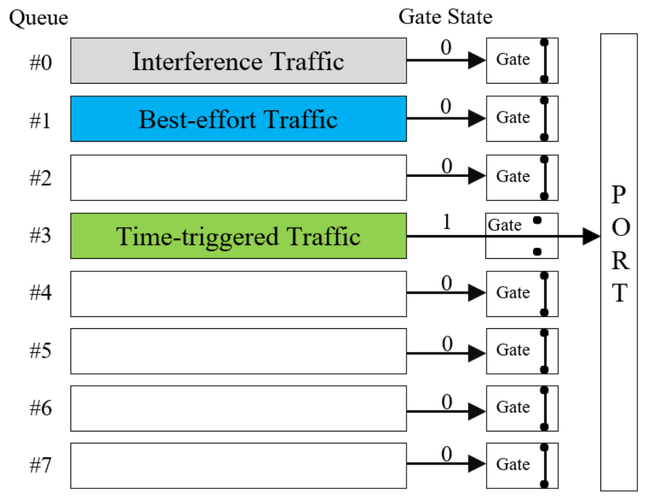


Fig. 6 Description of queues and its gate in the egress port.

mode. For this mode, the transmission time which transmits the frame through the egress port of SW can be calculated using (1),

$$T_{trans} = \frac{L_{frame}}{R_{link}} = \frac{L_{header}+L_{payload}+L_{CRC}}{R_{link}} \quad (1)$$

where L_{frame} represents the length of the traffic, which equals the sum of L_{header}, $L_{payload}$, and L_{CRC}. In this Protected Window mode, the time-triggered traffic which is the IEEE 802.1Q frame format will be transmitted alone. Hence, L_{header} and L_{CRC} equals 26 bytes and 4 bytes, respectively. $L_{payload}$ is 1000 bytes defined in Table I. R_{link} denotes the link rate, which equals 1-Gbit/s defined in Sub-Section B of Section III. Thus, based on (1) the transmission time in Protected Window denoted by T_{pw} equals 8.24 µs; this is the Time Interval for the Protected Window. Next, unprotected traffic are transmitted. The blue rectangle in Fig. 5 is the "Unprotected Window" transmission mode, used for transmission of unprotected traffic (best-effort and interference traffic). We allow the two queues which store these two kinds unprotected traffic to forward through the SW egress port during this period, so set the Gate State to "0000 0011". The cycle time of the SW operating GCL is set to 1ms which equals the send interval of time-triggered traffic defined in Table I. For Unprotected Window, the transmission time denoted by T_{Upw} can be calculated using (2),

$$T_{Upw} = T_{cycle} - T_{pw} - T_{gb} \quad (2)$$

where T_{cycle} represents the cycle time (1 ms), T_{pw} and T_{gb} represents the transmission duration of Protected Window and Guard Band, respectively. Therefore, the Time Interval of the Unprotected Window is 981.952 µs calculated using (2).

It is important to ensure that residual unprotected traffic does not affect transmission of time-triggered traffic; we thus created a Guard Band (the red rectangle in Fig. 5) for stopping all transmissions before the Protected Window. Therefore, we set the Gate State to "0000 0000" at this time. Guard Band can ensure that the link is idle enough when the gate of the queue storing time-triggered traffic is opened for transmission, thus we should make the gates associated with BE and TG traffic close sufficiently far in advance of the protected time slot [4]. This method can certain that the last unprotected traffic which starts transmission in Unprotected Window has finished its transmission and there is no traffic transmission on the link before the Protected Window. In the worst case, this would mean that the last unprotected transmission would require the maximum frame transmission time before the Protected Window opens. Thus, the Guard Band is the time needed to transmit a maximum frame; the last unprotected traffic will be completely transmitted before transmission of time-triggered traffic. In the model, the maximum frame is set to 1,200 bytes (i.e., a frame generated by the TG node). Similarly, the Time Interval parameter of Guard Band, i.e., the transmission time of TG traffic can be calculated using (1). TG traffic is the IEEE 802.3 format. Thus, in this case L_{header} and L_{CRC} equals 22 bytes and 4bytes, respectively. $L_{payload}$ is 1200 bytes defined in Table I. R_{link} is 1-Gbit/s as mentioned. Hence, the Time Interval of Guard Band can be calculated as 9.808 µs, i.e., the transmission time of TG traffic.

The time-scheduling traffic table (GCL) is shown in Table II. The three transmission modes form one single loop illustrated in the right panel of Fig. 5; After executing these three transmission modes, one loop ends, and the switch will start a new loop with a new timescale until the simulation is terminated.

TABLE II
GCLS PARAMETERS

Type	Current Time (μs)	Gate State	Time Interval (μs)
Protected Window	0	0000 1000	8.24
Unprotected Window	8.24	0000 0011	981.952
Guard Band	990.192	0000 0000	9.808

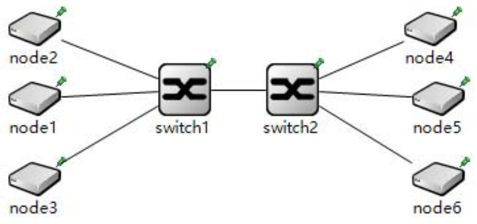
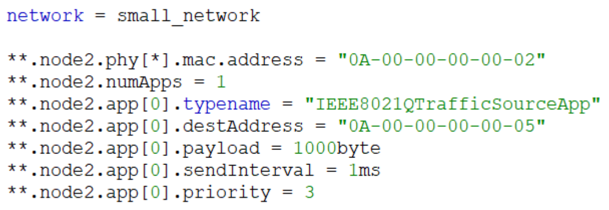


Fig. 7 Deployment topology in OMNET++.



```
network = small_network

**.node2.phy[*].mac.address = "0A-00-00-00-00-02"
**.node2.numApps = 1
**.node2.app[0].typename = "IEEE8021QTrafficSourceApp"
**.node2.app[0].destAddress = "0A-00-00-00-00-05"
**.node2.app[0].payload = 1000byte
**.node2.app[0].sendInterval = 1ms
**.node2.app[0].priority = 3
```

Fig. 8 Configuration of node 2.

TABLE III
END-TO-END LATENCIES

Type	Min	Mean	Max
TT latency (μs)	40.845	40.845	40.845
BE latency (ms)	1.46	79.2	98.02

TT, time-triggered; BE, best-effort

IV. EVALUATION RESULTS

We programed the functions using C++ and built the TSN topology of Section III using OMNET++ (Fig. 7); nodes 1, 2, and 3 represent the BE traffic sender, the TSN talker (generating time-triggered traffic), and the huge TG, respectively. The rest of the nodes receive traffic. Fig. 8 showed the parameters used to configure the TSN Talker (node 2) in the ".ini" format of OMNET++; these include "mac address", "traffic type", "size", and "send interval". At the beginning of the simulation, nodes 1–3 were configured to start traffic transmission at the same time, to allow us to better observe whether BE and TT traffic would be affected by TG traffic. Finally, we used the Signal Mechanism function of OMNET++ to obtain latency statistics; the details are in Chapter 12 of reference [17].

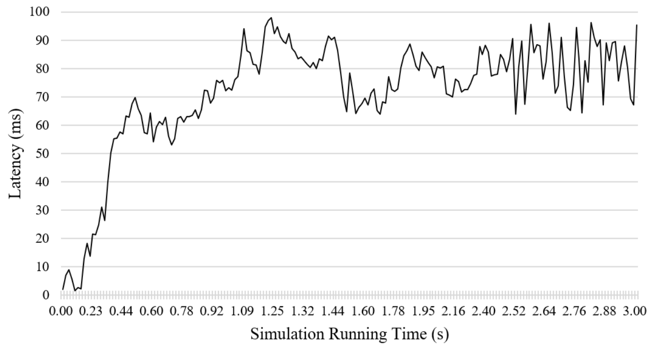


Fig. 9 Best-effort Traffic Simulation Results.

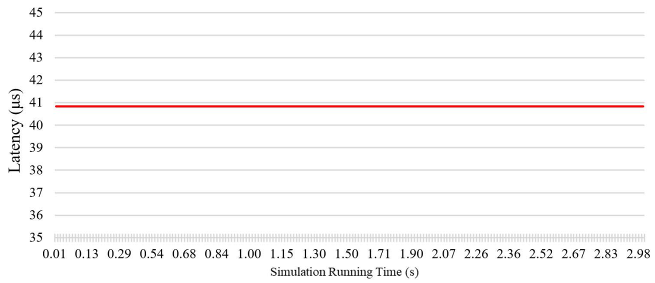


Fig. 10 Time-triggered Traffic Simulation Results.

The simulation allows calculation of the end-to-end latencies for TT and BE traffic. Table III shows the minimum, mean, and maximum end-to-end latencies. The end-to-end latencies of BE traffic and TT traffic are plotted in Fig.9 and Fig.10 respectively. Fig. 9 shows that the BE traffic is disturbed by the TG traffic; the BE end-to-end latency keeps rising and is uncertain values; the maximum value reached 98.02 ms. However, the end-to-end latency of TT traffic shown in Fig.10 was never disturbed by TG traffic, remaining stable at 40.845 μs; Thus, the simulation results can indicate that the TSN protocol can guarantee a deterministic and low end-to-end latency compared to standard Ethernet protocols.

V. CONCLUSIONS

We presented a TSN simulation model based on the OMNET++ simulation tool. We built an SW and a TSN topological model. Furthermore, we detail how to perform network scheduling calculations and how to derive the GCLs. Based on the simulation results, we found that even when the link was completely blocked, the model guaranteed deterministic end-to-end latency.

ACKNOWLEDGMENT

This work was supported by two foundations, one is Ansan-Si hidden champion fostering and supporting project funded by Ansan city. The project name: "Development of Industrial Communication-IoT Gateway & Industrial Wireless IoT Sensor". The other is the framework of international cooperation program (Korea-China) managed by National Research Foundation, the Republic of Korea (NRF-2016K2A9A2A11938310).

References

[1] IEEE, “802.3 Standard for Ethernet,” 2015.

[2] IEEE, “Time-Sensitive Networking Task Group,” http://www.ieee802.org/1/pages/tsn.html, 2016.

[3] IEEE, “802.1ASrev—Timing and Synchronization for Time-Sensitive Applications,” http://www.ieee802.org/1/pages/802.1AS-rev.html, 2017.

[4] IEEE, “802.1Qbv—Enhancements for Scheduled Traffic,” http://www.ieee802.org/1/pages/802.1bv.html, 2015.

[5] IEEE, “P802.1Qcc – Stream Reservation Protocol (SRP) Enhancements and Performance Improvements”, https://1.ieee802.org/tsn/802-1qcc/, 2017

[6] IEEE, “802.1Qci—Per-Stream Filtering and Policing,” http://www.ieee802.org/1/pages/802.1ci.html, 2016.

[7] C. Park, J. Lee, T. Tan, and S. Park, “Simulation of scheduled traffic for the IEEE 802.1 time sensitive networking,” in Information Science and Applications, LNEE Vol. 376. Springer, Singapore, 2016, pp. 75–83.

[8] P. Heise, F. Geyer, R. Obermaisser, “TSimNet: An Industrial Time Sensitive Networking Simulation Framework Based on OMNeT++” in *IEEE NTMS Larnaca Cyprus*: November 2016.

[9] F. Smirnov, M. Glaß, F. Reimann, and J. Teich, “Formal timing analysis of non-scheduled traffic in automotive scheduled TSN networks,” in Design, Automation & Test in Europe Conference & Exhibition, March 2017.

[10] A. M. Kentis, M. S. Berger, J. Soler, “Effects of port congestion in the gate control list scheduling of time sensitive networks,” in *the 8th International Conference on Network of the Future*, November 2017.

[11] L. X. Zhao, and P. Pop, “Timing Analysis of AVB Traffic in TSN Networks using Network Calculus” in *2018 IEEE Real-Time and Embedded Technology and Applications Symposium*, April 2018

[12] S. S. Craciunas, R. S. Oliver, M. Chmelik, and W. Steiner, “Scheduling Real-Time Communication in IEEE 802.1Qbv Time Sensitive Networks,” in *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, ACM, pp. 183-192, 2016.

[13] “OMNeT++.” [Online]. Available: http://www.omnetpp.org/

[14] “INET Framework.” [Online]. Available: http://inet.omnetpp.org/

[15] “CoRE4INET Framework.” [Online]. Available: https://core4inet.core-rg.de

[16] IEEE, “802.1Q - Bridges and Bridged Networks,” http://www.ieee802.org/1/pages/802.1Q.html, 2014

[17] “OMNeT++ Simulation Manual.” [Online]. Available: https://www.omnetpp.org/doc/omnetpp/manual/