Fault-tolerance in Time Sensitive Network with Machine Learning Model

by

Sang Hee Lee

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved November 2021 by the
Graduate Supervisory Committee:

ARIZONA STATE UNIVERSITY

May 2022

ABSTRACT

Nowadays, demand from the Internet of Things (IoT), automotive networking, and video applications is driving the transformation of Ethernet. It is a shift towards time-sensitive Ethernet. As a large amount of data is transmitted, many errors occur in the network. For this increased traffic, a Time Sensitive Network (TSN) is important. Time-Sensitive Network (TSN) is a technology that provides a definitive service for time-sensitive traffic in an Ethernet environment that provides time-synchronization. In order to efficiently manage these errors, countermeasures against errors are required. A system that maintains its function even in the event of an internal fault or failure is called a Fault-Tolerant system. For this, after configuring the network environment using the OMNET++ program, machine learning was used to estimate the optimal alternative routing path in case an error occurred in transmission. By setting an alternate path before an error occurs, I propose a method to minimize delay and minimize data loss when an error occurs. Various methods were compared. First, when no replication environment and secondly when ideal replication, thirdly random replication, and lastly replication using ML were tested. In these experiments, replication in an ideal environment showed the best results, which is because everything is optimal. However, except for such an ideal environment, replication prediction using the suggested ML showed the best results. These results suggest that the proposed method is effective, but there may be problems with efficiency and error control, so an additional overview is provided for further improvement.

DEDICATION

I recently completed my two-year master's program and submitted my master's thesis. The past two years have been both happy and sad. It's not enough, but as I finish my thesis, I would like to express my gratitude to them. I am infinitely grateful to my parents and sisters who have helped me more than anyone and have always been my strength. And I would like to thank my friends and lab colleagues who have always supported me.

ACKNOWLEDGMENTS

I am infinitely grateful to my supervisor Dr. Martin Reisslein who gave me the opportunity and guided me research works with invaluable advice. Also, I am very grateful to the Dr. Robert LiKamWa and Akhilesh Thyagaturu who willingly became members of my committee. I am also grateful to my lab colleagues for their great help and advice on the research. Especially, thank you for Ahmed and Venkat for their help and advice on research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation

As the demand for communication increases and diversifies, the demand for an Ultra-Low Latency network becomes important. However traditional packet networks can handle latencies to the order of tens of milliseconds [1]. Existing networks often change in various ways, such as changes network nodes and changes in network topology or traffic. In case of congestion on the network to prevent performance degradation due to the discarding and retransmission of packets transmitted through the Internet, it is possible to store more packets in the nodes constituting the Internet, such as routers/switches. It became possible to configure the size of the buffer to be large. However, when the buffer is configured to be large, the number of packets stored in the network node also increases [2]. These phenomena cause long latency and can be a big problem in the autonomous vehicle industry [3], where real-time connectivity is required, and in the augmented and virtual reality (AR/VR) industry [4] – [7]. In particular, the tele-surgical and transportation industries require more and more real-time connectivity [8]. Throughput depends on application requirements. Therefore, it is possible to send and receive a small amount of IoT data in the cloud, and it is diverse enough to transmit a large amount of media data [9]. Also, from a vehicle and robot perspective, higher data rates are required because the cameras used to control transmit video data [10]. As such, a mechanism that can generally accommodate a wide range of ULL requirements is highly desirable and useful [11],[54]-[55]. In addition, if the packet discard rate is reduced, and if a node has a problem, it is predicted and responded in advance, the latency may be significantly reduced. Furthermore, for the research, we used fault-tolerance. When building a system, handling faults is very important. No matter what kind of problem occurs over a long period of time, the performance before and after the problem should be the same, and the output should be the same. When these requirements are met, the system is said to be a fault-tolerant system. In handling system failure, it must be able to service normally for a long time, and in order to do that, it must be a system with durability against problems. There are often two types of recovery methods for fault-tolerant operation when a problem occurs. The first is roll forward recovery, and the second is roll back recovery. Roll-forward recovery is a method of recovering the system based on the state after an error occurs, and roll-back recovery restores the system to the checkpoint before the error. Also, there are other techniques for handling failures. Representatively, there are redundant system, operation continuity, and fault isolation. In the redundant system, there is the replication method I used for my research, and there are also redundancy and diversity methods. The replication method prepares a plurality of identical items and executes them in parallel, and the redundancy method prepares a plurality of identical items and uses them as an auxiliary in case of failure. The diversity method is to prepare and operate a plurality of different hardware and software of the same specification.

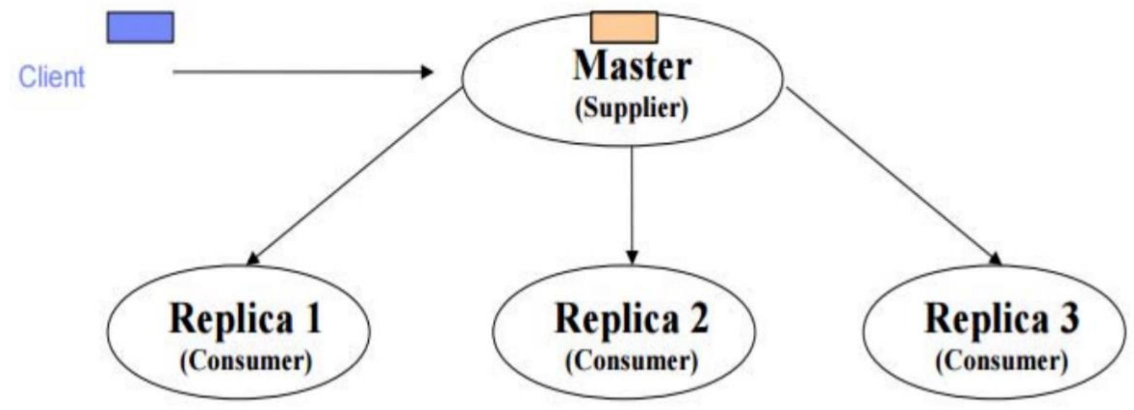


Fig 1 : Simple Replication Example [25]

In addition, there is operation continuation as a technique for fault handling. Operation continuity means that it should continue to operate even if there is a failure, and that the performance should not be degraded during restoration work. Another fault handling technique is fault isolation. Fault isolation means that the system is independent of faults and should not affect normal components. The redundancy system (Replication, Redundancy, Diversity) technique and the fault isolation technique are mainly focused on the design method, and the operation continuity technique is related to the actual implementation. The approximate situation in which fail tolerance occurs in the network is that if a node fails during data transmission, the node in transmission detects the failure and assigns the task to another node. Also, restarting a task does not require communication with other nodes that are working on other parts. When a failed node is restarted, it is automatically connected to the system and assigned a new task. Also, if a node detects that the performance of a particular node is very low, the node assigns the same task to another node.

Other important factor of this research is Time Sensitive Network (TSN). TSN is one of the standards of 802.1 and is a network transport mechanism that accurately transmits important data at the correct time. TSN has very low transmission latency and high transmission capacity. It is very necessary for real-time control capabilities such as real-time audio or factory control equipment, which may be part of a larger network that may involve gateways and wireless network [56]-[59]. In TSN, it is very important to synchronize the time. Data transmitted from one point to another must be time-accurate, and clocks must be synchronized because a common time reference is required to keep it accurate. Based on the synchronized clock, all devices and networks start work at the correct start time, allowing synchronized work at the required time. Due to these characteristics, it is also used for vehicular communication.

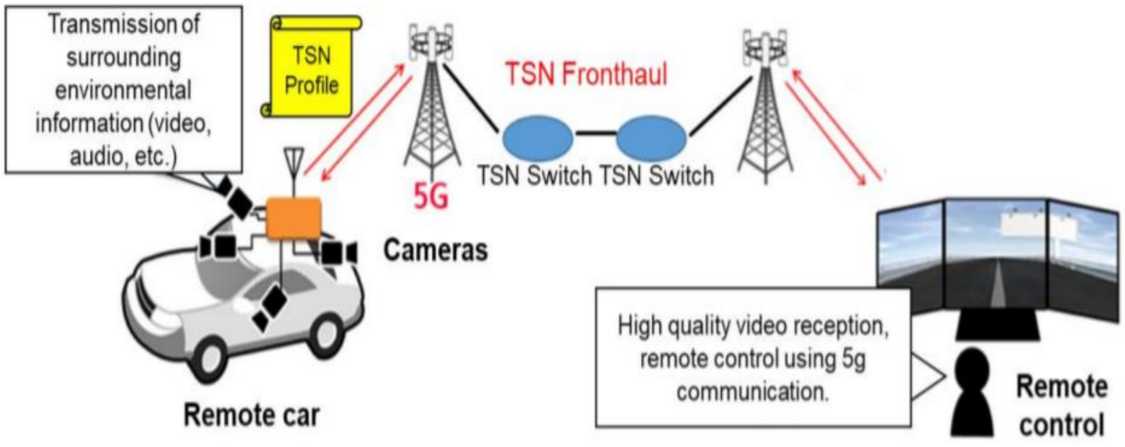

Transmission of surrounding environmental information (video, audio, etc.)
TSN Profile
TSN Fronthaul
5G
TSN Switch TSN Switch
Cameras
High quality video reception, remote control using 5g communication.
Remote car
Remote control

Fig 2 : Example of vehicular communication with TSN [26]

Another feature of Time Sensitive Network (TSN) is that it has priority by using scheduling and traffic shaping and can coexist in the same network with different types of data traffic with different bandwidths and latency [60]-[61]. In addition, although the TSN classifies data by using the priority, this does not guarantee that the priority data is delivered at the correct time. The reason for this is because of buffering inside the switch, no matter how high-priority data is, it is not transmitted without buffering. In a standard switch, it is difficult to avoid such a temporally inaccurate network transmission method. This may not be a big problem for office or home networks, but in places such as the automotive and medical industries, accurate data transmission at the right time is important, so strict priorities need to be strengthened. TSN enhances standard Ethernet communication capabilities by adding mechanisms to accurately communicate real-time transmission requests from software and hardware. TSN basically uses time division multiple access (TDMA) to establish a virtual communication channel at a specific time and for a specific period, allowing time-critical data traffic to be separated from non-critical data traffic. By allowing certain time-critical traffic to use the network first, avoiding the buffering effect of the switch, and ensuring that time-sensitive traffic arrives at its destination on time.

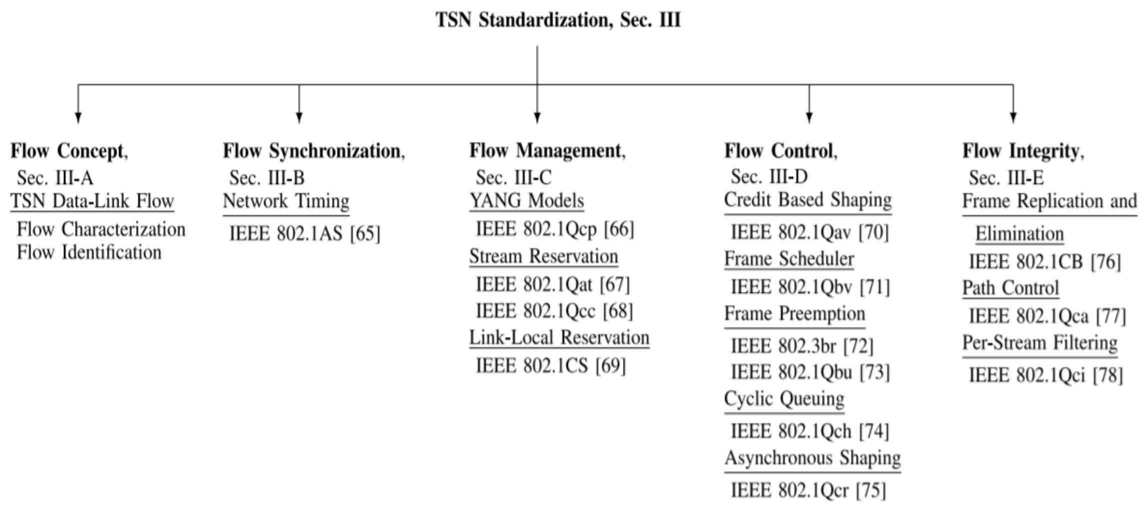


Fig 3 : Time Sensitive Networking (TSN) standardization [1]

Especially in modern industry, Time Sensitive Network is very important in various fields. It is used very importantly in audio and video fields in the media field and is also used in industries that require real-time data transmission, such as autonomous vehicles and factory automation. In the field of autonomous vehicles, each sensor of the vehicle processes data through a real-time network to drive safely, and in the field of factory automation, robots in the factory production line repeat the same tasks at the correct time, thereby minimizing defects.

In addition, machine learning technology was used for the study. Machine learning is a form of Artificial Intelligence (AI) that can learn systems from data rather than through explicit programming. Machine learning is not a simple process, it can collect training data through an algorithm and then create a more accurate model based on that data. A machine learning model is an output value generated when training a machine learning algorithm using data, and if the input is provided to the model after learning, the output result can be checked. In detail, the prediction algorithm creates a prediction model, and when data is provided to the prediction model, it predicts the result based on the data trained on the model. With machine learning, a model can be trained on data before it is deployed. Some machine learning models are made online, and through the iterative process of online models, the types of connections made between data elements are improved. After training the model, it can be used in real time to learn from the data, and then the prediction accuracy is improved due to the training process. Also, different approaches are used depending on the data type to increase the accuracy of the predictive model.

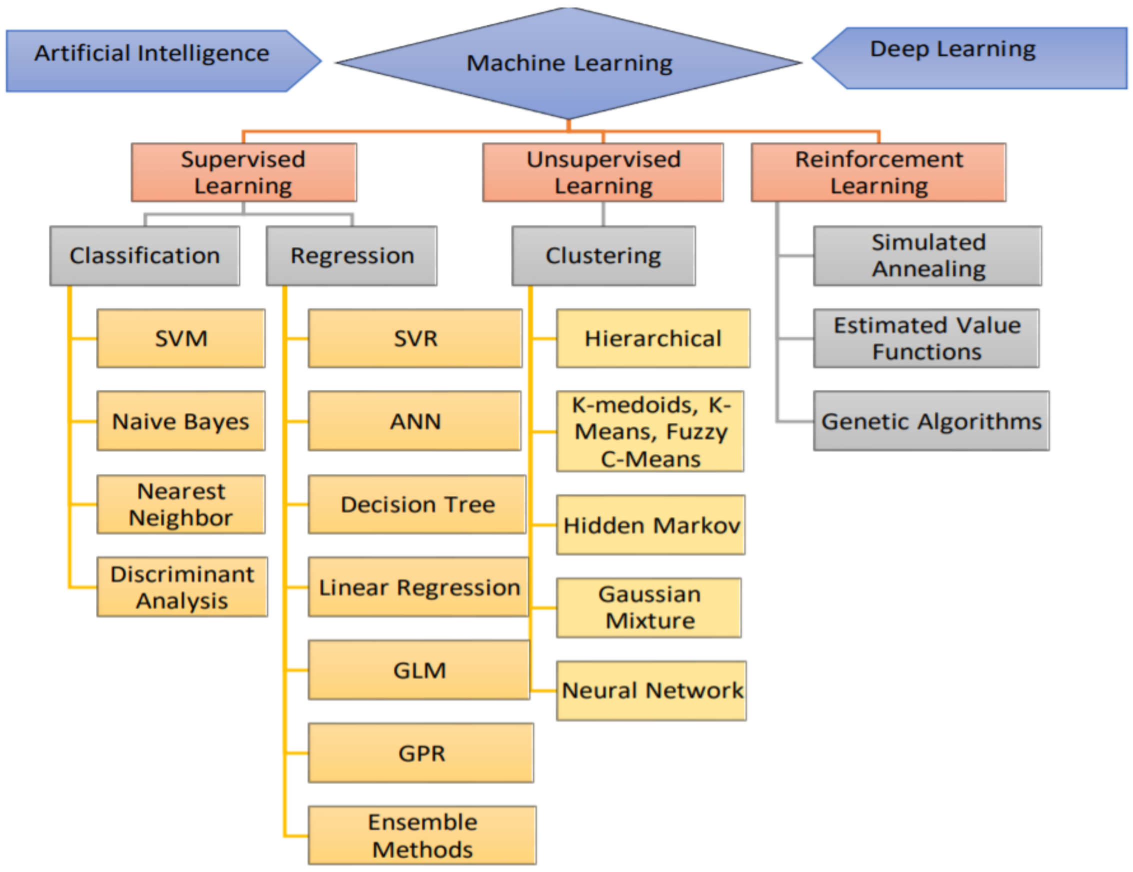


Fig 4 : Machine Learning Techniques Classification [27]

The first type is the supervised learning method. A supervised learning algorithm is a method of making predictions based on a set of cases. It also includes predicting the future, for example, based on past data. Supervised learning is accompanied by input variables composed of previously classified data for learning and desired output variables, and by analyzing the data for learning using an algorithm, a function that maps input variables to output variables can be found. This inferred function maps unknown new cases through generalization from training data and predicts outcomes in invisible situations. In addition, in supervised learning, the desired answer called label must be included in the training data injected into the algorithm, and classification and regression methods are used. Classification refers to supervised learning when data is used to predict categorical variables. For example, in cat images frequently used in machine learning, the case of assigning a cat-like label or indicator is called binary classification, and when there are two or more categories, it is called multi class classification. Also, regression is a method of predicting continuous output values. Regression is to find the relationship between two variables that predict the output value given predictor variables, input and response variables.

Another method is called unsupervised learning. In supervised learning, when training a model, it is a learning method that is used when the correct answer is known in advance, but in unsupervised learning, unlabeled or unknown data is dealt with. Using unsupervised learning techniques, data can be explored to extract meaningful information without knowing the known output values. Clustering of unsupervised learning is a data analysis technique that organizes group information accumulated without prior information into meaningful subgroups or clusters. to form Clustering is important because it organizes information and builds meaningful relationships in data.

Lastly, there is reinforcement learning. Reinforcement learning is characterized by improving agent system performance by interacting with the environment, and since the current state information of the environment includes a reward signal, reinforcement learning can be considered as a method related to supervised learning. However, feedback in reinforcement learning is not a ground truth value, but a value measured as a reward function, and the agent interacts with the environment to learn the behavior that maximizes the reward.

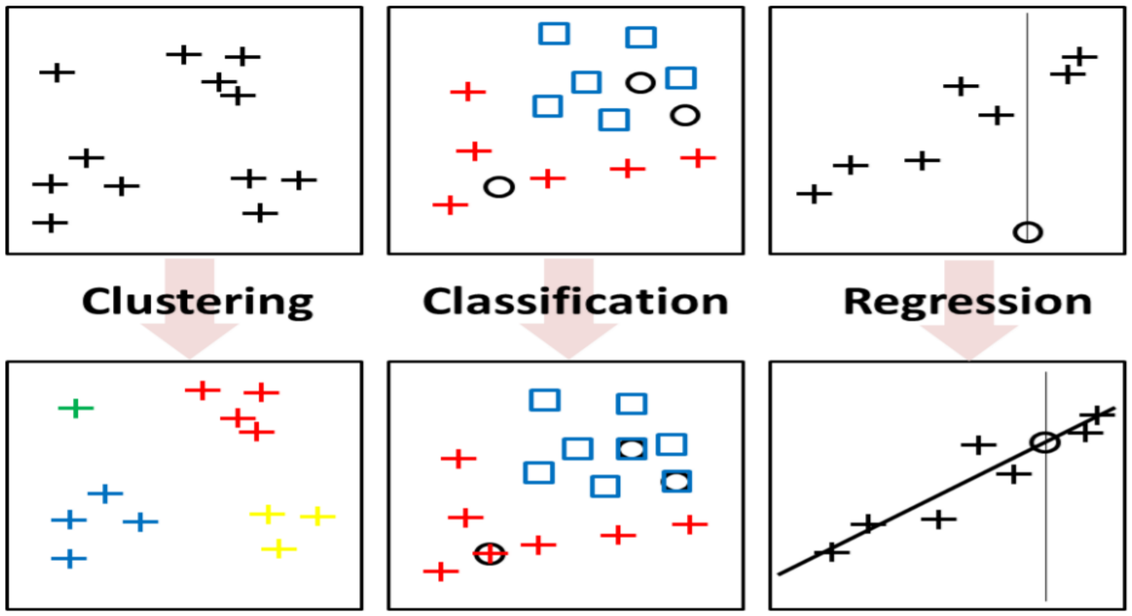


Fig 5 : Clustering, Classification and Regression [28]

Based on these technologies, this graduation thesis will cover in-depth research-related technologies such as latency, Time Sensitive Networking, Fault-tolerant and Machine Learning in the next section.

For research purposes, I have mainly referenced this thesis. The paper is called "Reconfiguration Algorithms for High Precision Communications in Time Sensitive Networks" [53]. Our study was based on this study. In this paper, maximizing the number of ad-mitted flows in a dynamic and volatile environment while maintaining QoS metric guarantees in IEEE 802.1 TSN [53] and evaluates the impact of TAS on its effectiveness with and without CNC. That is, we compare models of centralized and decentralized. In addition, the paper shows that the TAS time slot of IEEE 802.1 Qbv is allocated with high priority to the Scheduled Traffic (ST), and the TAS configuration is designed for centralized model and is a fully distributed model. Also, central network configuration (CNC) is utilized here for configuration exchange and network. In this study, the core technology element uses TAS for flow scheduling, and TAS distributes appropriate slots for each traffic class. Based on this paper we consider the scheduled traffic and best effort packets used in our study. The biggest difference between this paper and our study is the use of machine learning models and our research aimed to measure the data loss according to each run type situation. Our research is reconstructed the model based on the mentioned paper and propose a replication method to reduce packet loss.

1.2 Background and Related works

1.2.1 Latency Terminology

Latency generally refers to the total end-to-end packet delay that the recipient receives completely from the moment the sender starts the transmission [1]. The ULL (Ultra Low Latency) term literally means very short time such as milliseconds. ULL applications generally require deterministic latency, so this latency must be guaranteed as much as possible for proper functioning [12]. Furthermore, the application can require a stochastic latency. In other words, the prescribed delay limit must be met with a high probability such as multi-media services [13],[14], where a small number of bound delay violations have a negligible effect on the quality of multi-media.

1.2.2 Time Sensitive Network

On this section, I will handle a brief explanation of the Time Sensitive Network (TSN). Time-Sensitive Network (TSN) technology, the central technology of the Fourth Industrial Revolution, is a technology that provides definitive services such as low delay, low latency, and low packet loss for time-sensitive traffic such as control traffic or multi-media traffic based on time synchronization. IEEE 802.1Qch called as CQF [15] proposes to periodically adjust the enqueue or dequeue operation within the switch. Figure 1 shows a brief figure of the CQF mechanism.

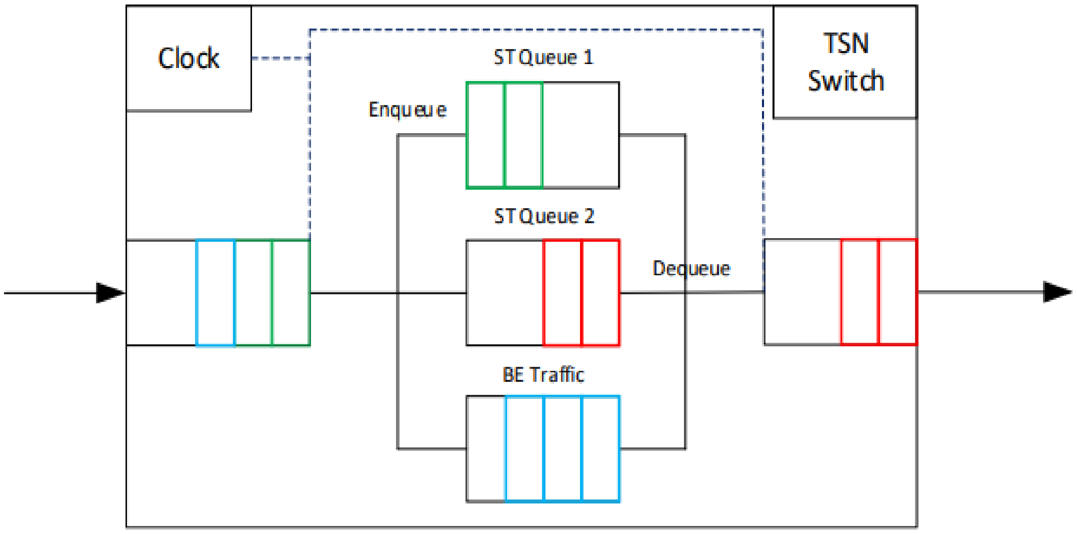


Fig 6 : Brief CQF Mechanism [16]

However, the existing CQF has a disadvantage that frames transmitted during the cycle must be received during the same cycle so that the delay is limited by the cycle time and the hop count. To compensate for the propagation delay exceeding this cycle time, 3-Queue CQF was proposed [17]. As shown in Figure 2, when traffic arrives in an out-of-date cycle, a third queue is needed so that the flow of traffic is not disturbed.

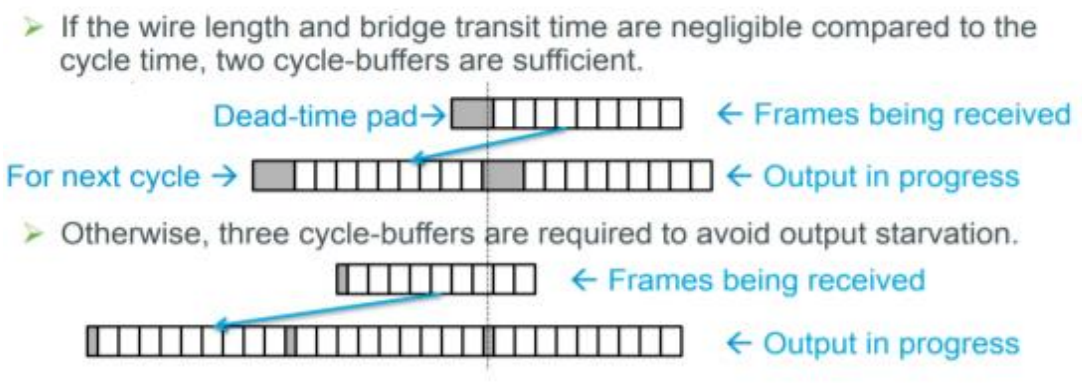

If the wire length and bridge transit time are negligible compared to the cycle time, two cycle-buffers are sufficient.
Dead-time pad→
← Frames being received
For next cycle →
← Output in progress
Otherwise, three cycle-buffers are required to avoid output starvation.
← Frames being received
← Output in progress

Fig 7 : Compared standard CQF and 3-Queue CQF [16]

Also, there is IEEE 802.1 TSN provides standarized framework for assuring ultra low latency (ULL) such as for avionics systems [1],[18]. Especially, IEEE 802.1 Qbv TAS

(Time Aware Sharper) is one of the core tools for the ULL network. One of the most important tasks in TAS is scheduling traffic flows. In order to prevent the interference of low-priority traffic such as best effort (BE) from priority such as scheduled traffic (ST), there is a guard band in ST transmission [19].

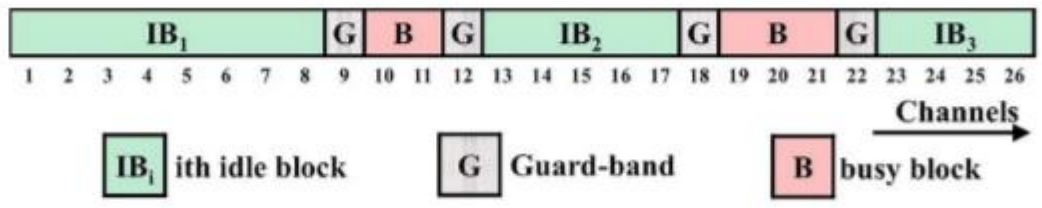


Fig 8 : Example of Guard Band [20]

TAS requires synchronization of trigger windows at all times to be applied for ULL. It means that all bridges from sender to receiver should be synchronized on time. All bridges must be synchronized on time between sender and recipient. For this reason, TAS opens and closes according to the schedule set at each port of the bridge. Overall, IEEE 802.1Qbv transmits frames from a given queue where the gate is open as follows. First, when a frame ready to transmit the next item in the queue. Secondly, when queues with higher priority traffic class gates do not have frames to transfer. Thirdly, when completing frame transfer for a given queue before the gate closes. The transmission conditions allow low priority traffic transmission only when the transmission of high priority traffic is completed. Though this transmission method effectively prevents low-priority traffic from interfering with high-priority traffic by setting up guard band [18].

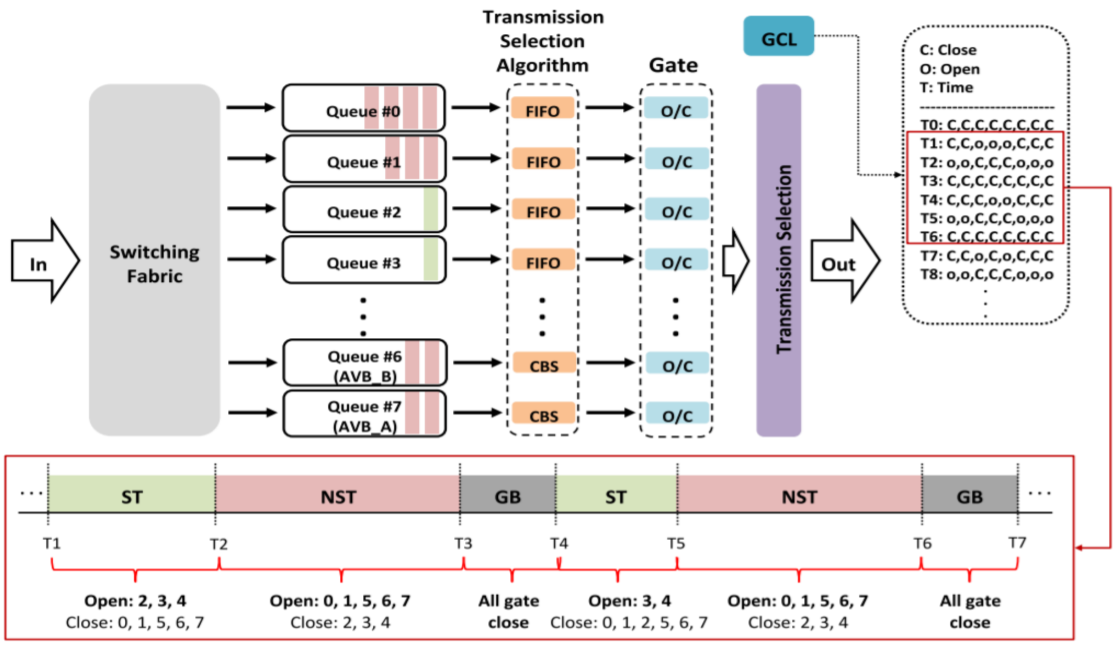


Fig 9 : Time Aware Sharping(TAS) [21]

Another IEEE standard is IEEE 802.1 Qcc. IEEE 802.1 Qcc provides tools for managing and controlling the network as a whole [22]. IEEE 802.1Qcc improves the existing Stream Reservation Protocol (SRP) with User Network Interface (UNI) using Centralized Network Configuration (CNC) node as shown in the Fig. 10 below [19]. In general, User Network Interface (UNI) provides a common way to request layer 2 services. Centralized Network Configuration (CNC) also interacts with User Network Interface (UNI) to offer centralized method for resource assignment and scheduling through remote management protocol like RESTCONF [23] or NETCONF [24]. Therefore, IEEE 802.1 Qcc data modeling language is compatible with the NETCOF language.

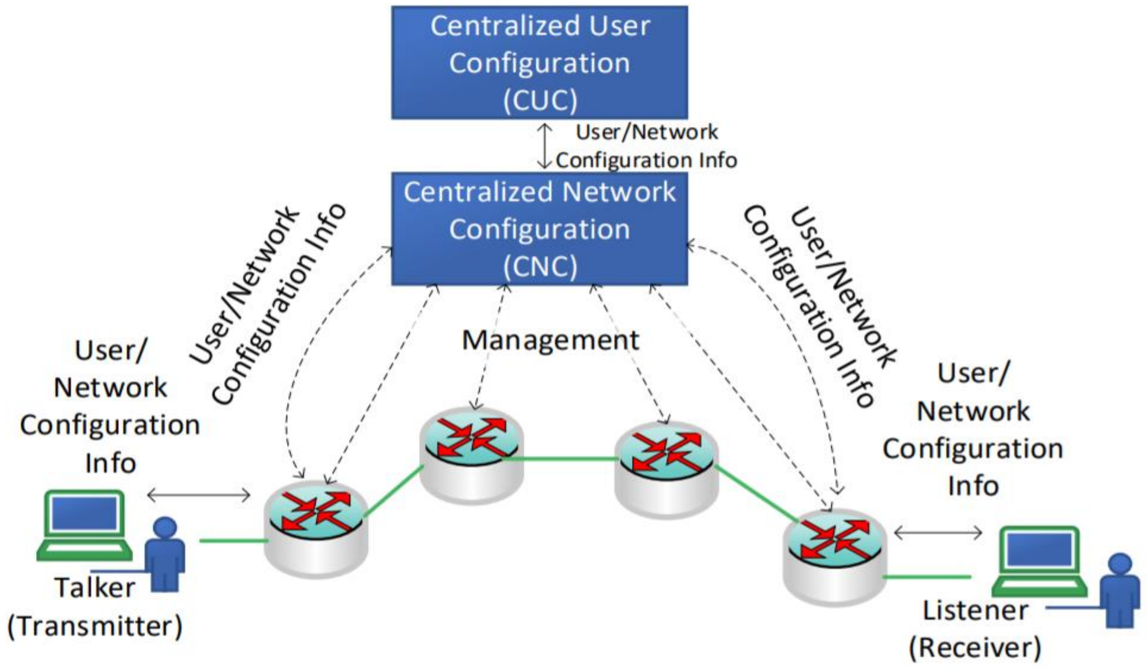

Centralized User Configuration (CUC)
User/Network Configuration Info
Centralized Network Configuration (CNC)
User/Network Configuration Info
User/Network Configuration Info
Management
User/ Network Configuration Info
User/ Network Configuration Info
Talker (Transmitter)
Listener (Receiver)

Fig 10 : Centralized Network Congifuration (CNC) [19]

Also, I will introduce a paper on time-sensitive networks to help with research. In a paper called Reliability-Aware Multipath Routing of Time-Triggered Traffic in Time-Sensitive Networks [52], proposed a heuristic-based reliability-aware routing algorithm that aims to improve the schedulability of time-triggered flows in TSN. This paper [52] demonstrates a new concept of multipath routing using Shortest path routing (SPR), Shortest multipath routing (SMR), and Reliablility Aware Multipath Routing (RAMR). The proposed Reliability-Aware Multipath Routing approach consists of three steps. First, a candidate lighting solution is found, and a cost function is calculated based on the degree of collision and flow latency. Then, the Ant Lion Optimization algorithm (ALO) is used to minimize the cost function [51]. They compared the methods proposed in Python-based frameworks with SPR and SMR methods. The proposed method does not significantly improve stability in low-connection graphs and small topologies. However, compared to conventional routing methods, the scheduling possibilities are greatly improved with higher reliability to handle large networks [51]. In particular, in their RAMR results, you can see that the data streams do not overlap.

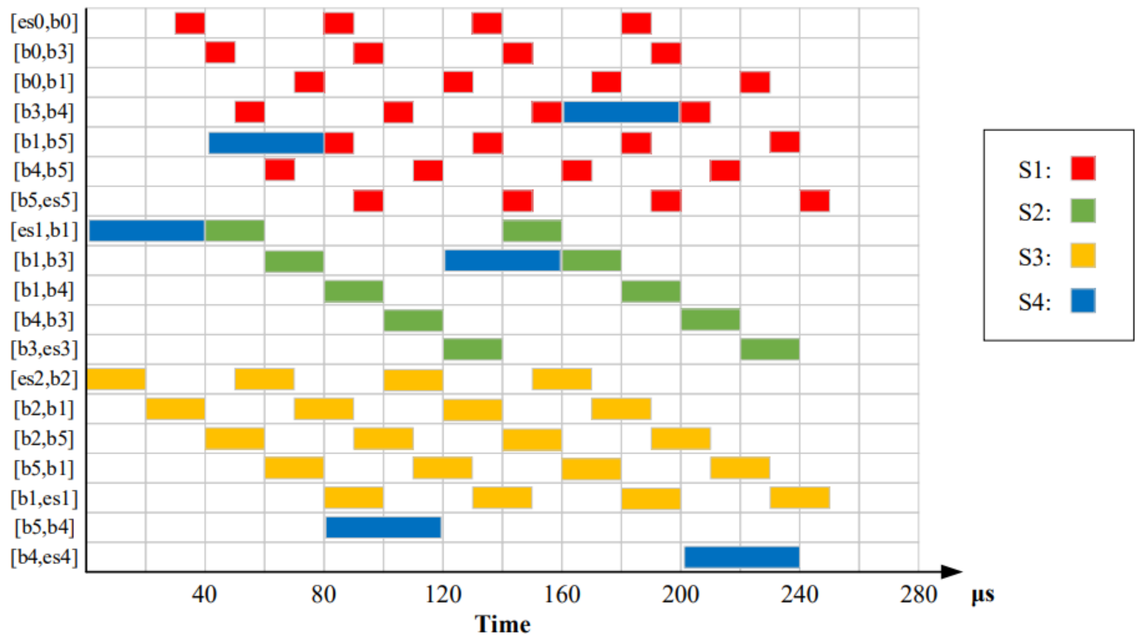


Fig 11 : Result of RAMR routing [51]

With reference to these papers, the research was conducted. The next chapter, 1.2.3, will describe fault-tolerance which was a major goal of the research.

1.2.3 Fault Tolerance

A fault-tolerance system refers to a system that maintains its function even in the event of an internal defect or failure. In modern society, it is widely used in self-driving cars and automated robot factories and high fault tolerance is also required in semiconductor factories, which are very controversial these days. Because of this characteristic of fault tolerance, it is widely applied in the fields related to human safety, national defense, aviation, and medical equipment. The reason that such fault tolerance is necessary is that a fault that occurs due to a physical defect or an external factor may lead to an error in which the expected value and the actual measured value are different, which leads to the possibility of system failure. In order not to lead to system failure, the first thing to do is to quickly detect the occurrence of an error, diagnose the failure, and recover the system through this. The fault-tolerant system aims to prevent the system from failing through these measures and to operate the system so as not to cause great damage to users or companies. Failures can be divided into temporary, permanent, and intermittent according to the continuity of occurrence. In case of temporary failure, it occurs once and the failure state is not maintained. Permanent failure is mainly caused by components or design errors in the system, and since the failure persists permanently, multiple errors occur. Intermittent failure is a failure condition that occurs occasionally, and can be caused by component connections, etc. Various methods are being considered as a configuration for a system for high failure. Our research has been carried out to improve this fault-tolerance and to achieve lower cost and faster error recovery. Since my research is a fault tolerance applied to the Time Sensitive Network, I will introduce some suggested technologies related to this. Fault-tolerant topology and routing synthesis for IEEE time-sensitivity networking [49] suggests determining a fault-resistant network topology consisting of redundant physical links and bridges, minimizing architectural costs, and routing each stream of applications. In addition, Agent based fault tolerance in wireless sensor networks [50] shows fault tolerance in a multi-agent-based method using a set of fixed and mobile agents.

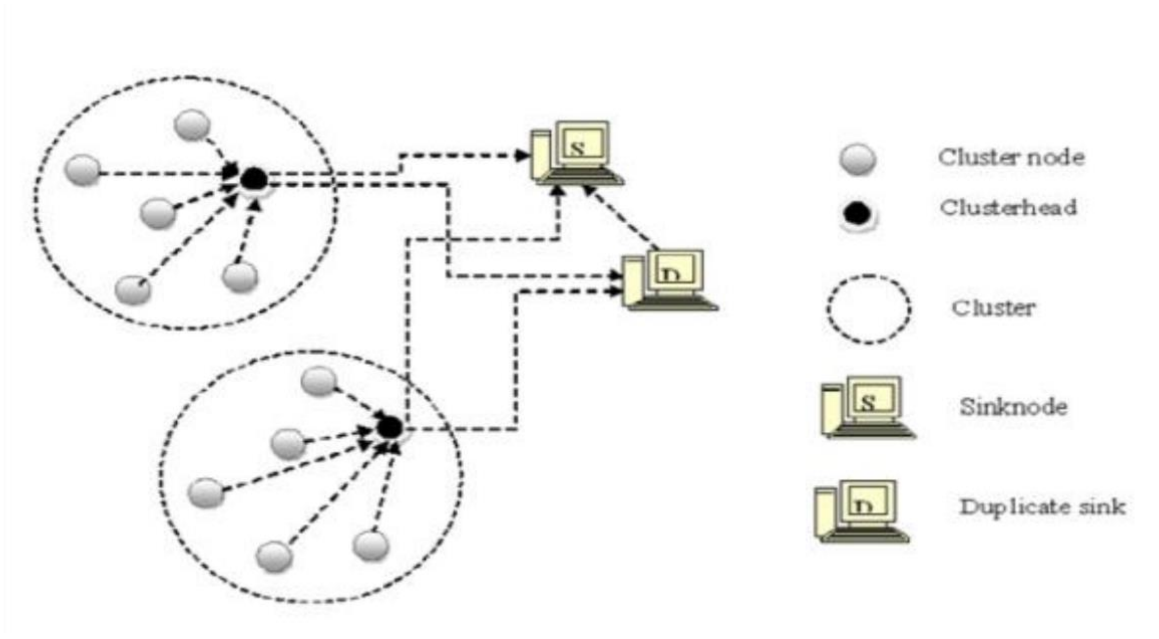


Fig 12 : Example of Fault-tolerance model [50]

The paper, Automated fault-tolerance testing, introduces Screwdriver, a new automation solution that improves the fault tolerance and resilience of the entire system [51]. In the real world, there are many obstacles to consider in system design, such as node failure, network failure, and high service response time. In particular, it is very important to test for such errors and failures, and to check for fault-tolerance of the system because a lot of losses can occur due to system down [51]. As a way to handle these failures, this paper [51] introduce and measure Screwdriver, an automated fault-tolerance tool built by Groupon's test engineering team.

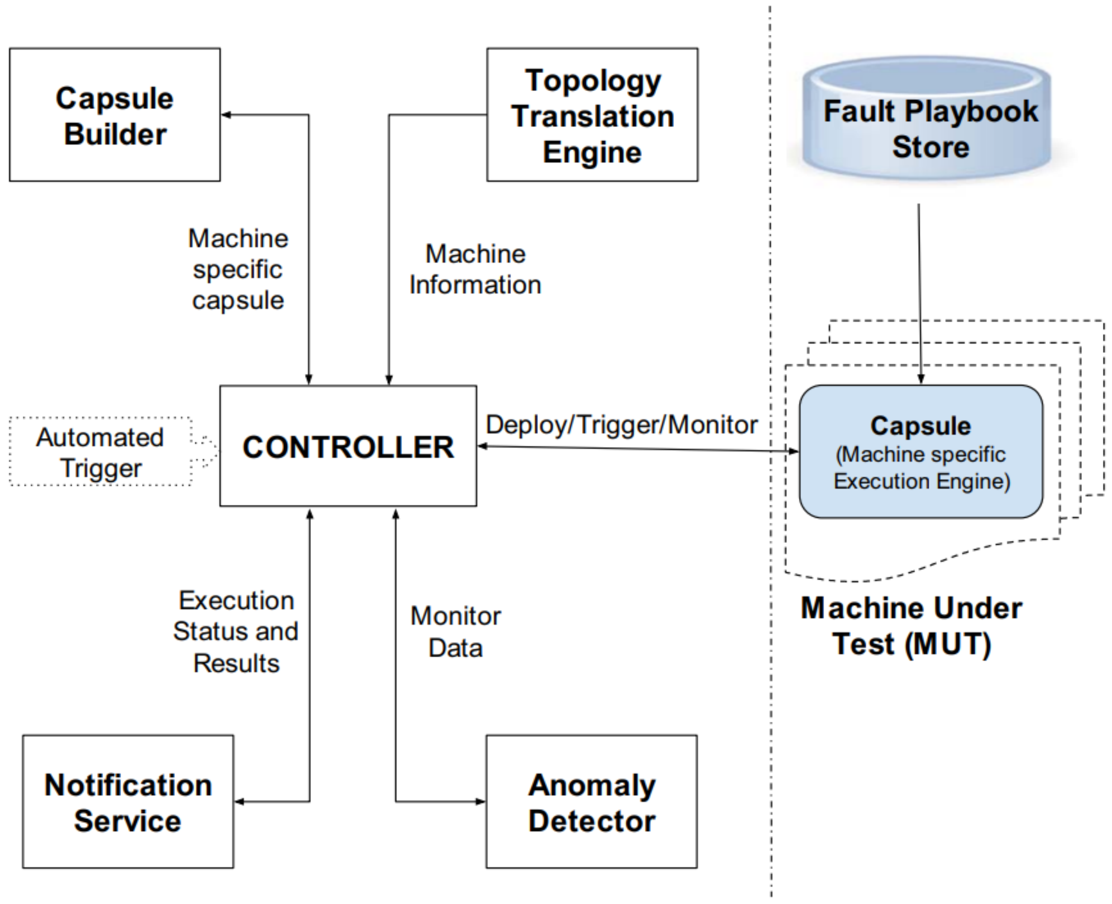


Fig 13 : Screwdriver design [51]

We devised a method using machine learning to improve fault-tolerance based on these ongoing studies. Therefore, in Chapter 1.2.4, we will discuss machine learning.

1.2.4 Machine Leaning

In the modern society, the terms Machine Learning and Artificial Intelligence are becoming very talkative. In particular, machine learning, which is always discussed as a core technology of the 4th industrial revolution, is to obtain new knowledge by giving data to computers to learn just as humans learn. If the previous program entered the input data and the program into the computer to get the desired output, machine learning is to teach the computer the input data and the output result.

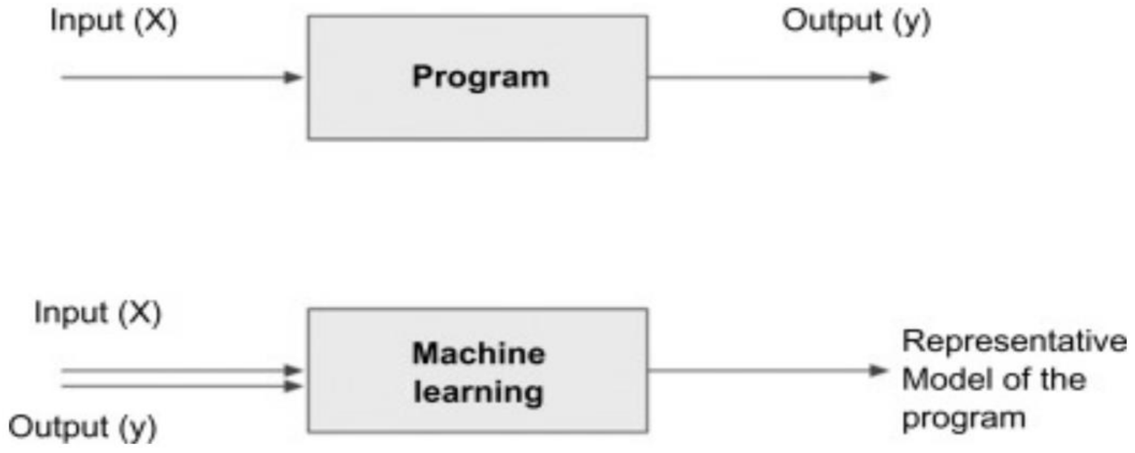


Fig 14 : Traditional program and Machine Learning [30]

Machine learning allows computers to learn on their own without human programming, and creates algorithms that learn from data and make predictions. Through powerful computing, the information and patterns hidden in the dataset are derived through complex mathematics and algorithms.

Machine learning is divided into three categories: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning learns the mapping relationship between input and target output, and the target output has discrete classification and continuous regression. In addition, unsupervised learning is a case where there is only an input value and there is no target output. There is clustering that classifies input data into N groups, and association analysis that analyzes relationships between data is also included in unsupervised learning. Reinforcement learning is a decision process that mimics the human judgment process, and when an action is taken in a specific state, it is determined according to the reward system. Furthermore, we should also know the term deep learning, which is used a lot these days. Deep learning is a branch of machine learning that gradually learns meaningful expressions in successive layers, and is a new way to learn expressions from data. The term deep used here refers to the depth of the model in how many layers are used, and recent deep learning models have dozens or hundreds of consecutive layers. All of these layers are exposed to training data and automatically trained. Also, we need to know Aritificial Neural Network (ANN) in deep learning. ANN is a machine learning algorithm created by mimicking the principle of a human neural network as a structure. The motif is that neurons in the human brain receive a certain signal or stimulus, and when the stimulus crosses a threshold, the resulting signal is transmitted. ANN uses the input value to calculate the output value of the artificial neuron, and compares the output value calculated by the artificial neuron with the output value expected by the user. In this process, the weights are adjusted to generate an output value. However, it is difficult to find the optimal parameters, and the learning time is relatively long. A supplement to this is the Deep Neural Network (DNN). DNN is a method with more than one hidden layer. The computer creates a classification label by itself and repeats the process of classifying the data to derive the optimal dividing line. It requires a lot of data and iterative learning, and is currently widely used. By applying

DNN, algorithms such as Convolution Neural Network (CNN) and Recurrent Neural Network (RNN) have been derived.

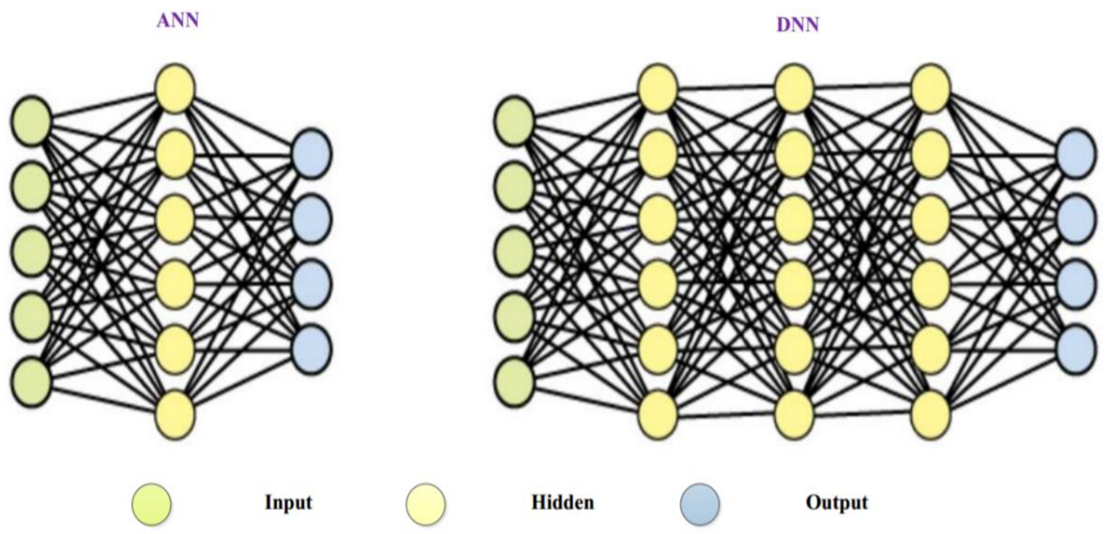


Fig 15 : Traditional ANN and DNN architecture [31]

Convolution Neural Network (CNN) is a method to identify patterns of features by extracting features from data. The CNN algorithm proceeds through convolution and pooling. In the process of extracting the characteristics of the data, the characteristics are identified by examining the adjacent components of each component of the data, and the identified characteristics are derived. The layer derived from this is called a convolution layer. This does an effective job of reducing the number of parameters. Also, pooling is a process of reducing the size of the convolutional layer. Intuitively, it reduces the size of data and has the effect of canceling out noise. CNN is widely used in fields such as information extraction and face recognition.

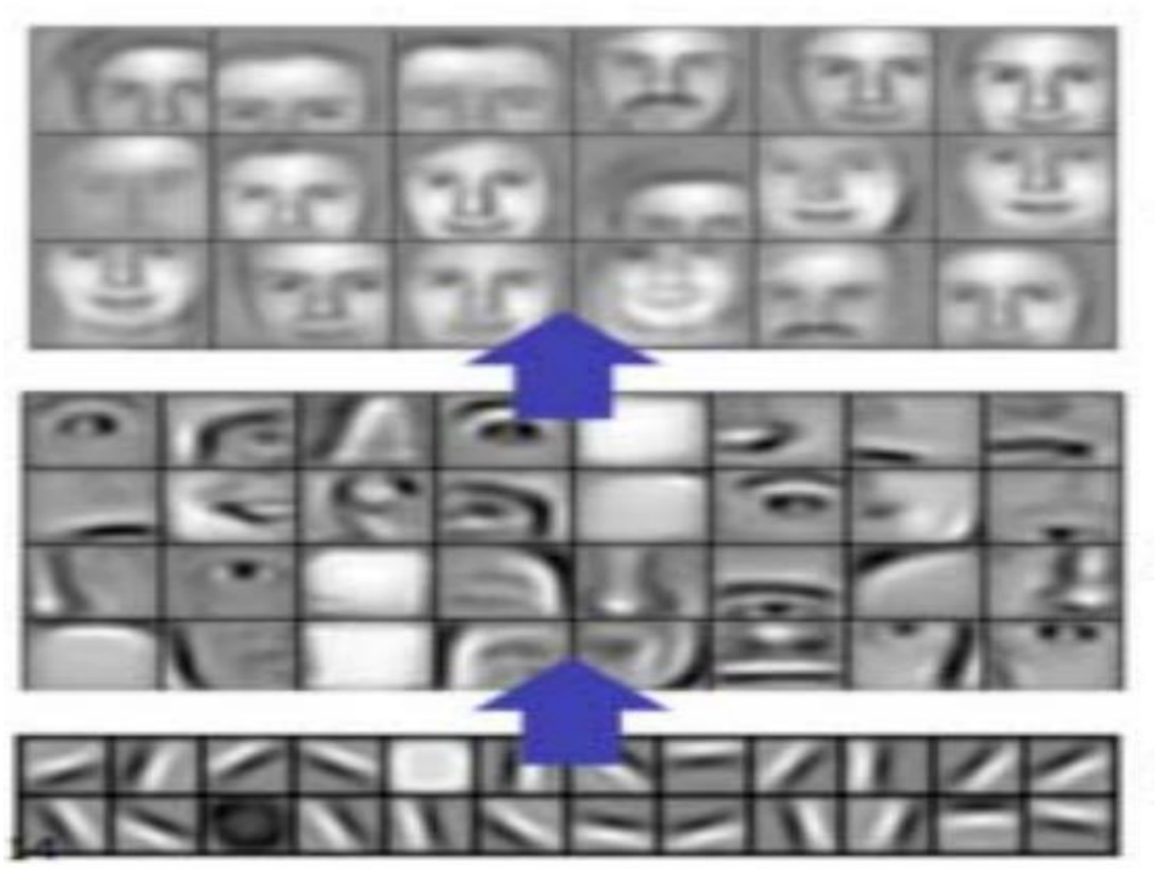


Fig 16 : Learned features from CNN [32]

Also, there is one more method called a Recurrent Neural Network (RNN). The RNN algorithm is a type of artificial neural network specialized in iterative and sequential data learning, and has a characteristic that it contains an internal cyclic structure. Using a circular structure, past learning is reflected in current learning through weights. It is an algorithm that solves the limitations of the existing continuous, iterative and sequential data learning. It makes it possible to connect the learning of the present with the learning of the past, and has a characteristic that is dependent on time. In addition, many techniques have been developed in addition to ANN, DNN, CNN, and RNN described above. It is also very important to know how to use the appropriate technique because each technique has different pursuits and different results. We changed the machine learning technique several times to get the right results and also changed the number of hidden layers. In the chapter 3, I will describe the model and results for the simulation that I researched. First, the simulation model will be explained for the researchand the results according to each run type will be explained.

CHAPTER 2

FAULT-TOLERANT TSN BASED ON MACHINE LEARNING

In modern networks, they vary dynamically due to changes in traffic patterns, network nodes. To deal with, IEEE 802.1 Qbv TAS shows excellent performance with deterministic ultra-low latency. TAS allocates gate times for ST and BE. TAS is giving exclusive access to transport media and devices to time-critical traffic classes, the buffering effect of Ethernet switch transport buffers can be avoided. The common concept for allocation system is to control network. It provides timely and orderly to access service with maximum of streams. We approached fault tolerance by overlapping with TAS. Our goal is to minimize packet loss in a dynamically changing TSN network. For this purpose, we applied the ML model.

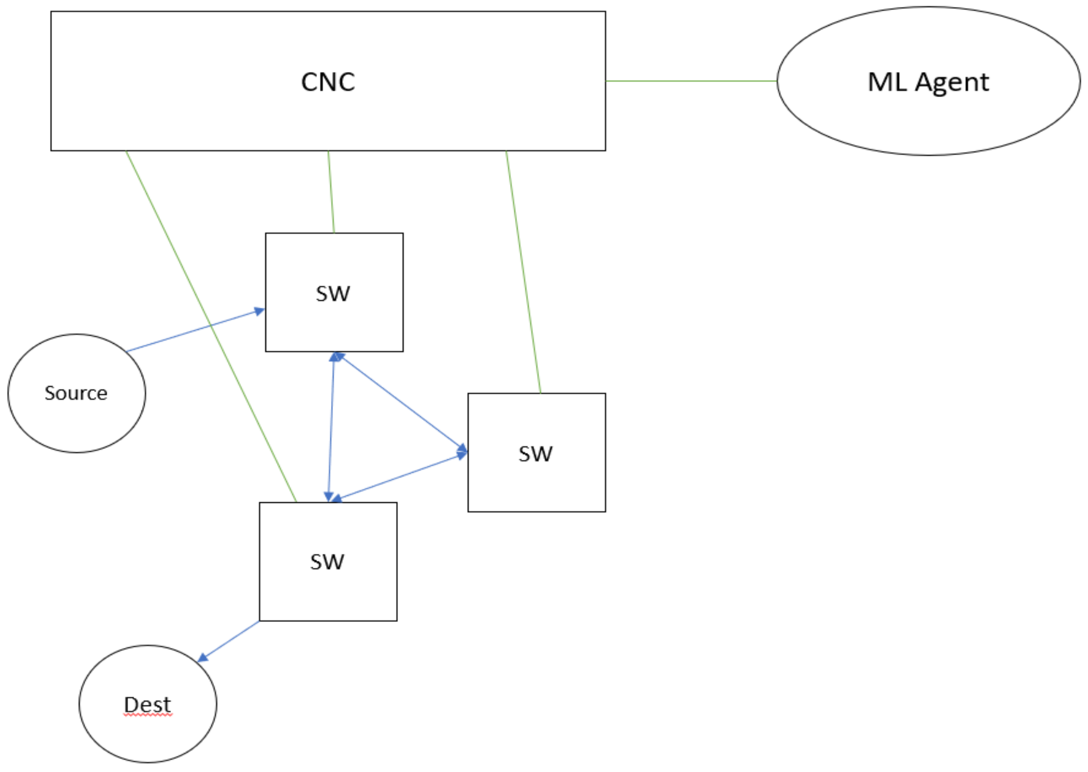


Fig 17 : Modeling

In this figure6 modeling, the first step is to send a packet from the source to the switch. At this time, all switches are connected to the CNC, and the CNC controls the flow. The CNC gives the ML Agent information about the data of each switch, and the ML Agent calculates and informs the best place to replicate data packets. Through this process, we can send the correct data packet to the destination without any problem. We conducted research with this model. Specially, compared to previous research data, our contribution did not have a method to determine the replication location using Machine Leaning like the method we conducted. I tried changing the methods as CNN and FNN comparing the results while changing the number of hidden nodes.

Also, in the modeling of our research, one-way transmission between the source and the switch, and the two-way transmission between switches (duplex). The node between the CNC and the switch is also a two-way transmission node, and the one-way transmission is set between the switch and Dest. The section between CNC and ML Agent is also set for bidirectional transmission, so data can be exchanged with each other. More detail for process, CNC keep monitoring network condition. After degradation happened, if packet error rate (PER) is bigger than threshold, switch send notice to CNC. CNC send to ML agent and ML agent uses port throughput, number of disconnections, port packet loss and packet error rate (PER).

CHAPTER 3

3.1 SIMULATION ARCHITECTURE

This chapter describes the overall architecture of the simulation of the study. After data transmission starts, the access method varies depending on the run type. In Run type 0, 1, 2 degradation occurs, and disconnection occurs. However, in the run type 3 ML model, after checking the port condition before disconnection occurs, disconnection is predicted. Then, by performing replication before disconnection occurs, packet loss is reduced. However, in run type 2, after disconnection occurs, replication starts and replication cannot be delivered to the destination, resulting in relatively large packet loss. In run type 1, since it is an ideal replication situation, packet loss is small, but it should be noted that this is an ideal situation that is difficult to occur. Also, since run type 0 is a no replication situation, there is no replication, so packet loss is large. Also, due to run type 3 uses machine learning, data is required. For the ML model, we use number of disconnections, port throughput, port packet loss, and packet error rate (PER). For this, we parsed the data obtained from OMNET++ using MATLAB. Below, figure 17 shows the overall architecture as a flow chart.

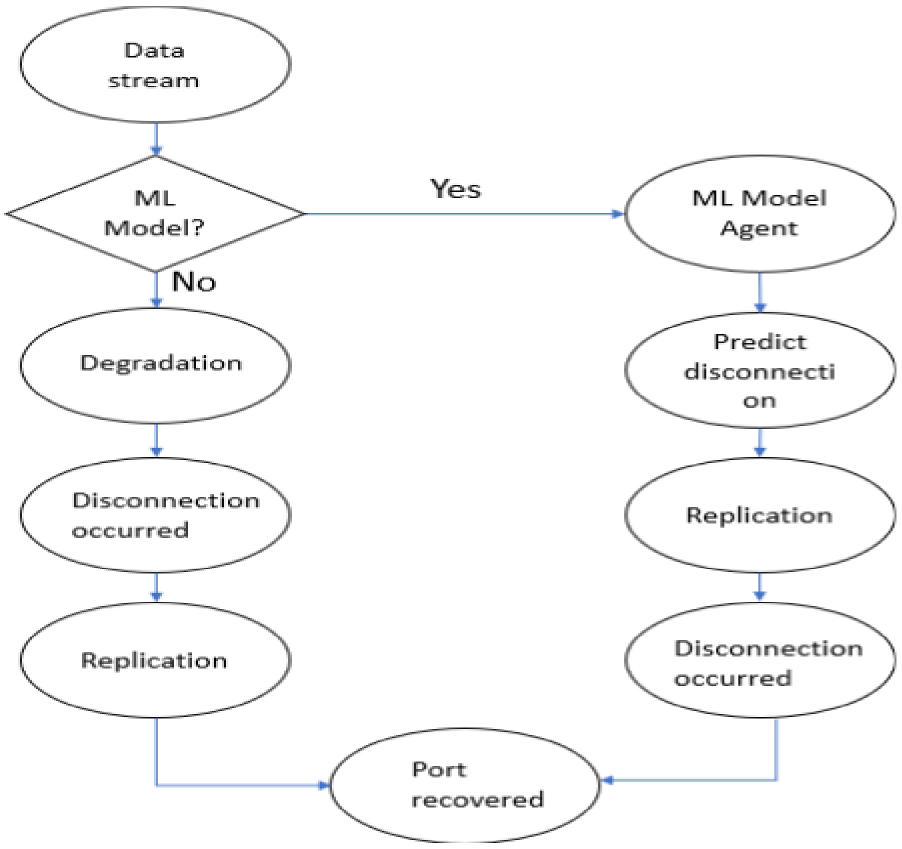


Fig 18 : Simulation Architecture

Studies were conducted based on these structures. Unlike previous studies, this study applied machine learning for replication. This is different from existing research and I think it can be helpful for future technology development. The next chapter 2.2 will describe the simulation model and explain in which model the research was conducted.

3.2 SIMULATION MODEL

In this research, the topology was designed using OMNET++ and for the machine learning part, Python (Spyder) was used with Keras and TensorFlow. The ultimate goal of the study is to reduce data loss by performing replication in advance by predicting disconnection in advance using machine learning. The study was run in four environments for comparison. Run type 0 = No replication environment, Run type 1: Ideal replication environment, Run type 2: Random replication environment, and finally Run type 3: ML replication environment. The data for machine learning used simulation values of OMNET++. The CNC used in the model is responsible for the overall control of the topology as a central network controller. SC is a best effort (BE) traffic generator as a source. Also, SCP is a scheduled traffic (ST) generator with source priority. Sink is a traffic collector. For the study, 1 CNC, 6 SCs, 6 SWs and 6 SCPs were used, and 1 sink was also used. The study conducted an experiment on how effective replication using machine learning is in such an environment. For comparison, we did not use only the method using machine learning, but also derived the results of the various methods described above. In the next section, we will compare the results of each of the different methods, and lastly describe the future work with conclusions.

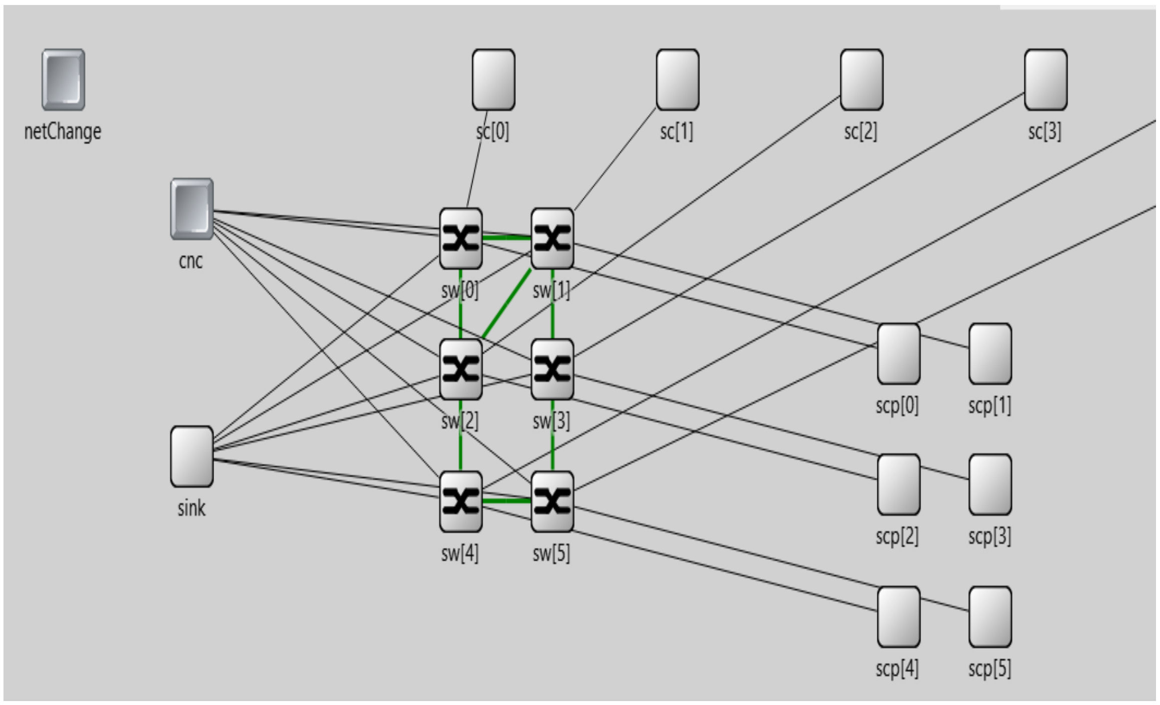
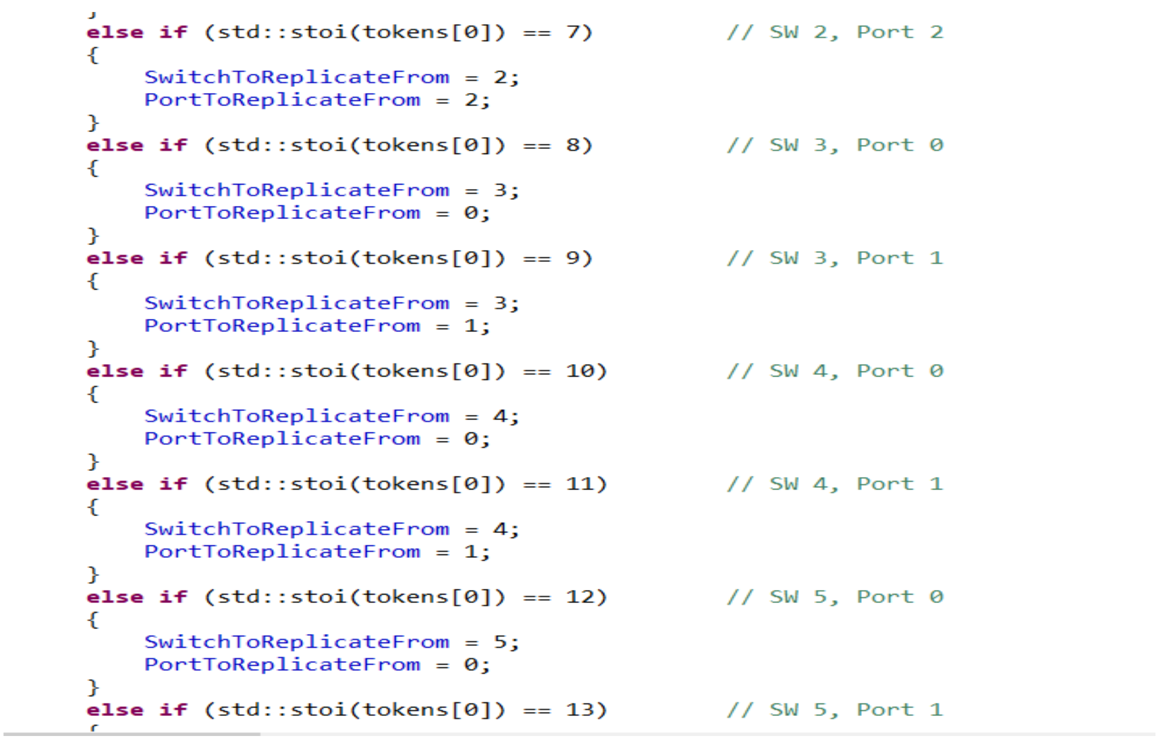


Fig 19 : OMNET++ Topology

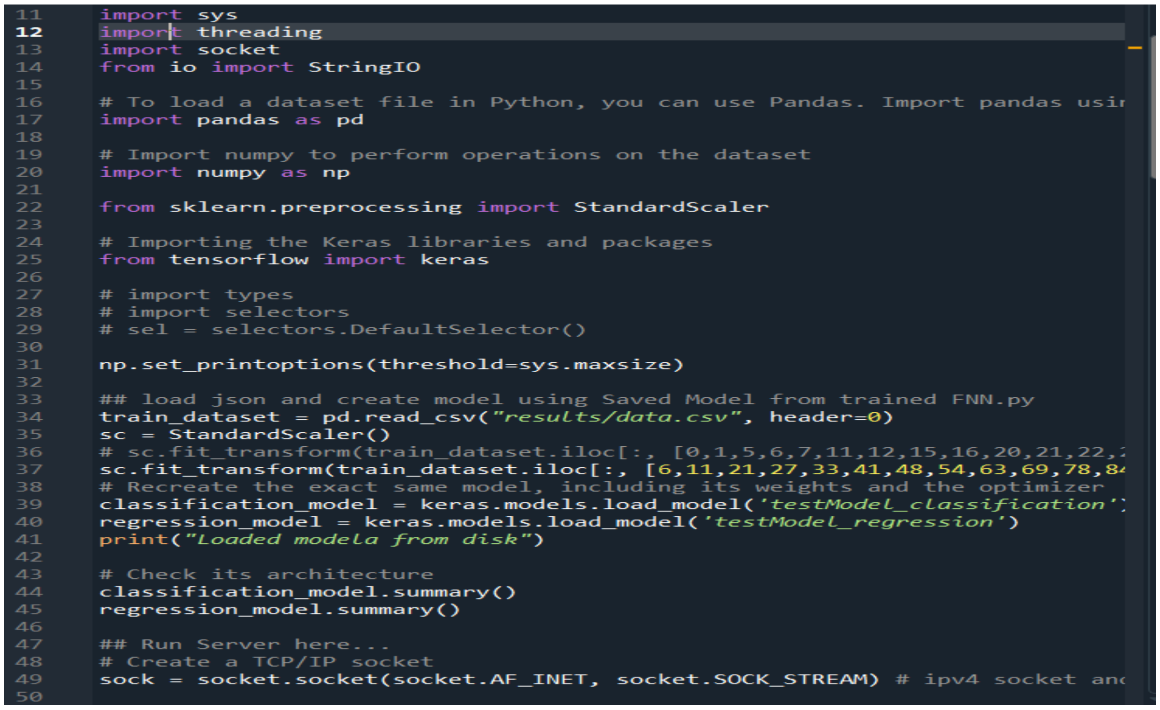
ST(Scheduled Traffic)	0
BE(Best Effort)	1
Qsize	512*1024
upSTLimit	0.91
downSTLimit	0.10
Degradation Value	0.1
Run type	0,1,2,3

Table 1 : Network Parameters



```
else if (std::stoi(tokens[0]) == 7)         // SW 2, Port 2
{
    SwitchToReplicateFrom = 2;
    PortToReplicateFrom = 2;
}
else if (std::stoi(tokens[0]) == 8)         // SW 3, Port 0
{
    SwitchToReplicateFrom = 3;
    PortToReplicateFrom = 0;
}
else if (std::stoi(tokens[0]) == 9)         // SW 3, Port 1
{
    SwitchToReplicateFrom = 3;
    PortToReplicateFrom = 1;
}
else if (std::stoi(tokens[0]) == 10)        // SW 4, Port 0
{
    SwitchToReplicateFrom = 4;
    PortToReplicateFrom = 0;
}
else if (std::stoi(tokens[0]) == 11)        // SW 4, Port 1
{
    SwitchToReplicateFrom = 4;
    PortToReplicateFrom = 1;
}
else if (std::stoi(tokens[0]) == 12)        // SW 5, Port 0
{
    SwitchToReplicateFrom = 5;
    PortToReplicateFrom = 0;
}
else if (std::stoi(tokens[0]) == 13)        // SW 5, Port 1
```

Fig 20 : Example of OMNET++ Code

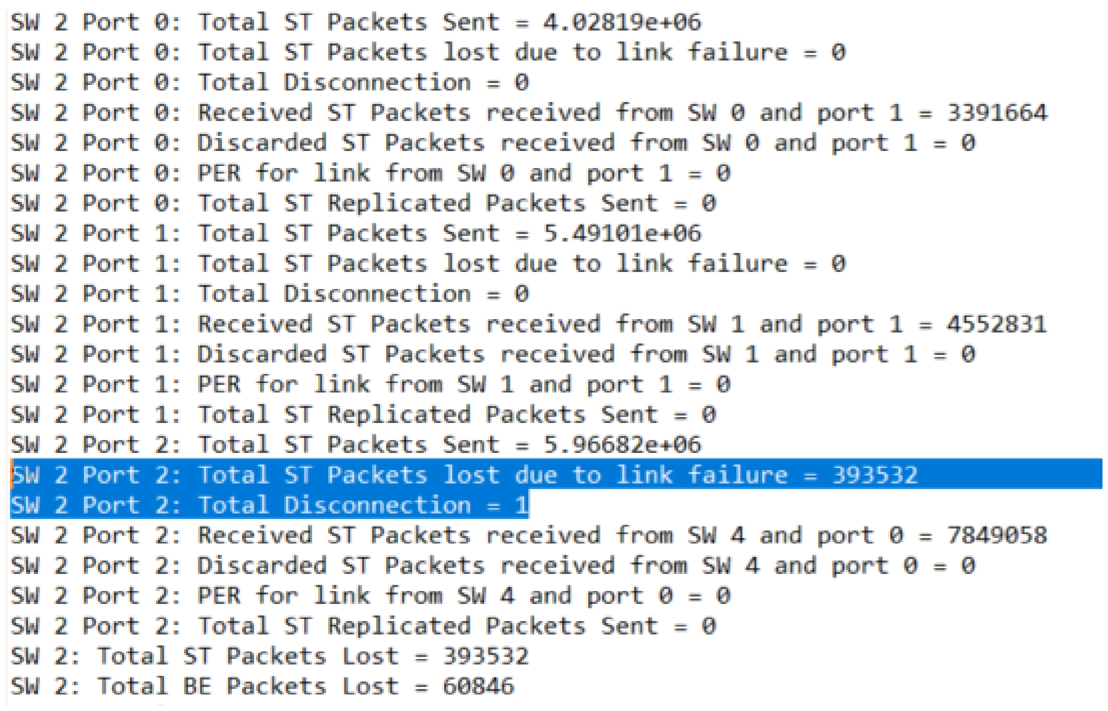


```
import sys
import threading
import socket
from io import StringIO

# To load a dataset file in Python, you can use Pandas. Import pandas usin
import pandas as pd

# Import numpy to perform operations on the dataset
import numpy as np

from sklearn.preprocessing import StandardScaler

# Importing the Keras libraries and packages
from tensorflow import keras

# import types
# import selectors
# sel = selectors.DefaultSelector()

np.set_printoptions(threshold=sys.maxsize)

## load json and create model using Saved Model from trained FNN.py
train_dataset = pd.read_csv("results/data.csv", header=0)
sc = StandardScaler()
# sc.fit_transform(train_dataset.iloc[:, [0,1,5,6,7,11,12,15,16,20,21,22,
sc.fit_transform(train_dataset.iloc[:, [6,11,21,27,33,41,48,54,63,69,78,8
# Recreate the exact same model, including its weights and the optimizer
classification_model = keras.models.load_model('testModel_classification')
regression_model = keras.models.load_model('testModel_regression')
print("Loaded modela from disk")

# Check its architecture
classification_model.summary()
regression_model.summary()

## Run Server here...
# Create a TCP/IP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM) # ipv4 socket an
```

Fig 21 : Example of Python Code

3.3 SIMULATION RESULT

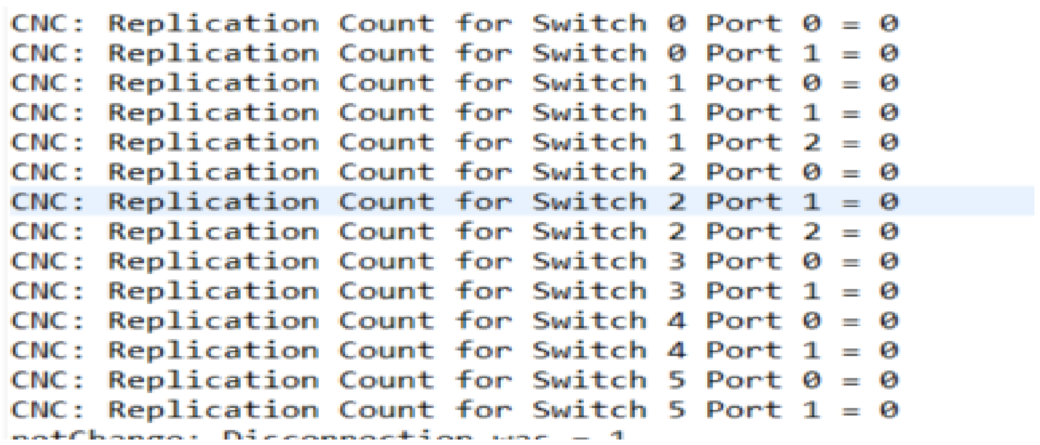
3.3.1 RUN TYPE 0



```
SW 2 Port 0: Total ST Packets Sent = 4.02819e+06
SW 2 Port 0: Total ST Packets lost due to link failure = 0
SW 2 Port 0: Total Disconnection = 0
SW 2 Port 0: Received ST Packets received from SW 0 and port 1 = 3391664
SW 2 Port 0: Discarded ST Packets received from SW 0 and port 1 = 0
SW 2 Port 0: PER for link from SW 0 and port 1 = 0
SW 2 Port 0: Total ST Replicated Packets Sent = 0
SW 2 Port 1: Total ST Packets Sent = 5.49101e+06
SW 2 Port 1: Total ST Packets lost due to link failure = 0
SW 2 Port 1: Total Disconnection = 0
SW 2 Port 1: Received ST Packets received from SW 1 and port 1 = 4552831
SW 2 Port 1: Discarded ST Packets received from SW 1 and port 1 = 0
SW 2 Port 1: PER for link from SW 1 and port 1 = 0
SW 2 Port 1: Total ST Replicated Packets Sent = 0
SW 2 Port 2: Total ST Packets Sent = 5.96682e+06
SW 2 Port 2: Total ST Packets lost due to link failure = 393532
SW 2 Port 2: Total Disconnection = 1
SW 2 Port 2: Received ST Packets received from SW 4 and port 0 = 7849058
SW 2 Port 2: Discarded ST Packets received from SW 4 and port 0 = 0
SW 2 Port 2: PER for link from SW 4 and port 0 = 0
SW 2 Port 2: Total ST Replicated Packets Sent = 0
SW 2: Total ST Packets Lost = 393532
SW 2: Total BE Packets Lost = 60846
```

Fig 22 : Result of run type 0 (1)

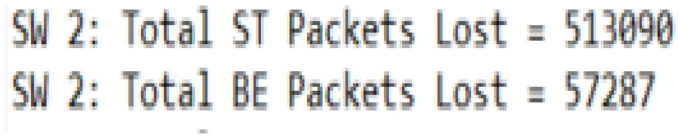
Figure 22 shows the result when run type 0 is in the no replication situation. When disconnection occurs in SW2 Port 2, it shows packet lost. Result indicated that many packets are lost and this is because replication was not performed.



```
CNC: Replication Count for Switch 0 Port 0 = 0
CNC: Replication Count for Switch 0 Port 1 = 0
CNC: Replication Count for Switch 1 Port 0 = 0
CNC: Replication Count for Switch 1 Port 1 = 0
CNC: Replication Count for Switch 1 Port 2 = 0
CNC: Replication Count for Switch 2 Port 0 = 0
CNC: Replication Count for Switch 2 Port 1 = 0
CNC: Replication Count for Switch 2 Port 2 = 0
CNC: Replication Count for Switch 3 Port 0 = 0
CNC: Replication Count for Switch 3 Port 1 = 0
CNC: Replication Count for Switch 4 Port 0 = 0
CNC: Replication Count for Switch 4 Port 1 = 0
CNC: Replication Count for Switch 5 Port 0 = 0
CNC: Replication Count for Switch 5 Port 1 = 0
```

Fig 23 : Result of run type 0 (2)

Also, Figure 23 shows the result of no replication taking place anywhere on the central network controller (CNC). Because run type 0 is a no replication model, replication does not happen anywhere.



```
SW 2: Total ST Packets Lost = 513090
SW 2: Total BE Packets Lost = 57287
```

Fig 24 : Result of run type 0 (3)

Figure 24 shows the overall scheduled traffic (ST) packets lost and best effort (BE) packets lost in SW2 where disconnection occurred. Result indicates that there was a very large packet loss overall.

3.3.2 RUN TYPE 1

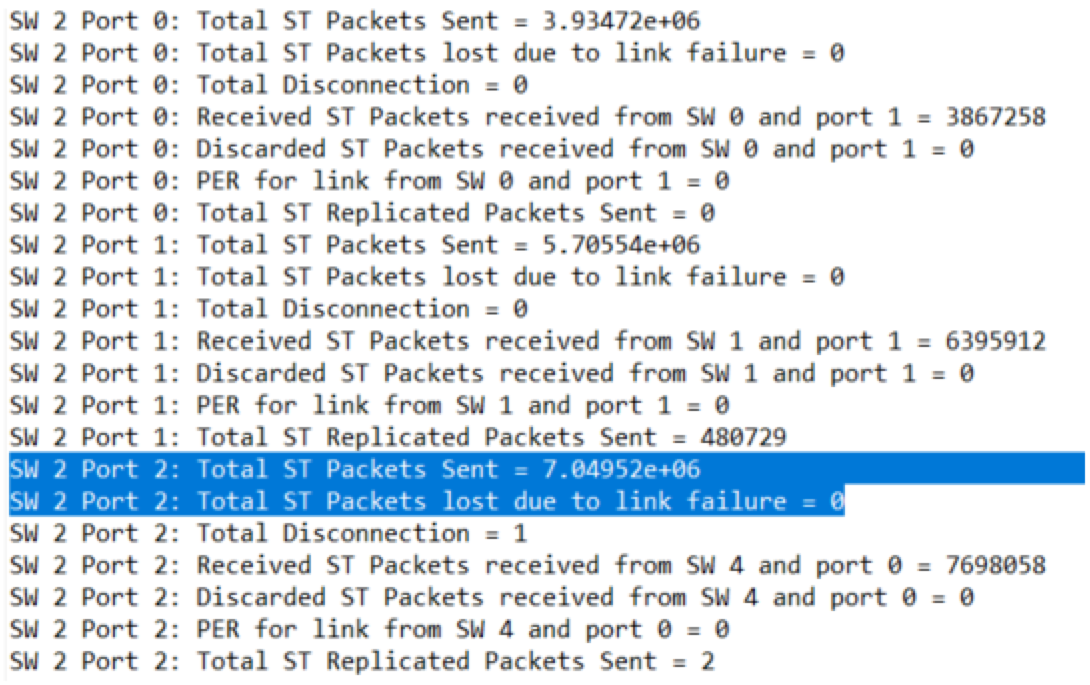
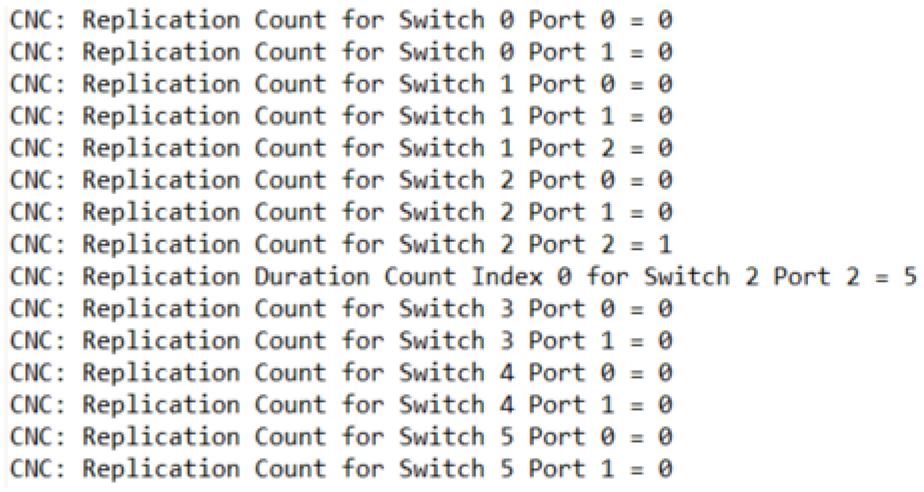

```
SW 2 Port 0: Total ST Packets Sent = 3.93472e+06
SW 2 Port 0: Total ST Packets lost due to link failure = 0
SW 2 Port 0: Total Disconnection = 0
SW 2 Port 0: Received ST Packets received from SW 0 and port 1 = 3867258
SW 2 Port 0: Discarded ST Packets received from SW 0 and port 1 = 0
SW 2 Port 0: PER for link from SW 0 and port 1 = 0
SW 2 Port 0: Total ST Replicated Packets Sent = 0
SW 2 Port 1: Total ST Packets Sent = 5.70554e+06
SW 2 Port 1: Total ST Packets lost due to link failure = 0
SW 2 Port 1: Total Disconnection = 0
SW 2 Port 1: Received ST Packets received from SW 1 and port 1 = 6395912
SW 2 Port 1: Discarded ST Packets received from SW 1 and port 1 = 0
SW 2 Port 1: PER for link from SW 1 and port 1 = 0
SW 2 Port 1: Total ST Replicated Packets Sent = 480729
SW 2 Port 2: Total ST Packets Sent = 7.04952e+06
SW 2 Port 2: Total ST Packets lost due to link failure = 0
SW 2 Port 2: Total Disconnection = 1
SW 2 Port 2: Received ST Packets received from SW 4 and port 0 = 7698058
SW 2 Port 2: Discarded ST Packets received from SW 4 and port 0 = 0
SW 2 Port 2: PER for link from SW 4 and port 0 = 0
SW 2 Port 2: Total ST Replicated Packets Sent = 2
```

Fig 25 : Result of run type 1 (1)

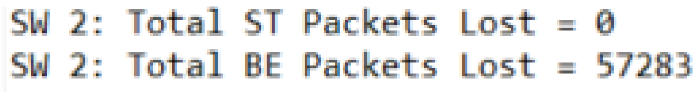
Figure 25 shows that the result when run type 1 ideal replication situation. Even if disconnection occurred in SW2 Port2 like run type 0, run type1 result shows that scheduled traffic (ST) packets lost is 0 because it is an ideal replication situation. It should be borne in mind that these results are perfect replication situations that are unlikely to occur in reality.



```
CNC: Replication Count for Switch 0 Port 0 = 0
CNC: Replication Count for Switch 0 Port 1 = 0
CNC: Replication Count for Switch 1 Port 0 = 0
CNC: Replication Count for Switch 1 Port 1 = 0
CNC: Replication Count for Switch 1 Port 2 = 0
CNC: Replication Count for Switch 2 Port 0 = 0
CNC: Replication Count for Switch 2 Port 1 = 0
CNC: Replication Count for Switch 2 Port 2 = 1
CNC: Replication Duration Count Index 0 for Switch 2 Port 2 = 5
CNC: Replication Count for Switch 3 Port 0 = 0
CNC: Replication Count for Switch 3 Port 1 = 0
CNC: Replication Count for Switch 4 Port 0 = 0
CNC: Replication Count for Switch 4 Port 1 = 0
CNC: Replication Count for Switch 5 Port 0 = 0
CNC: Replication Count for Switch 5 Port 1 = 0
```

Fig 26 : Result of run type 1 (2)

In addition, Figure 26 shows that disconnection occurred in SW2 Port2 in run type1 ideal replication and that replication was measured in central network controller (CNC). Because run type 0 is a no replication model, replication does not happen anywhere. Unlike run type 0, in run type 1 ideal replication situation, it shows that replication has occurred.

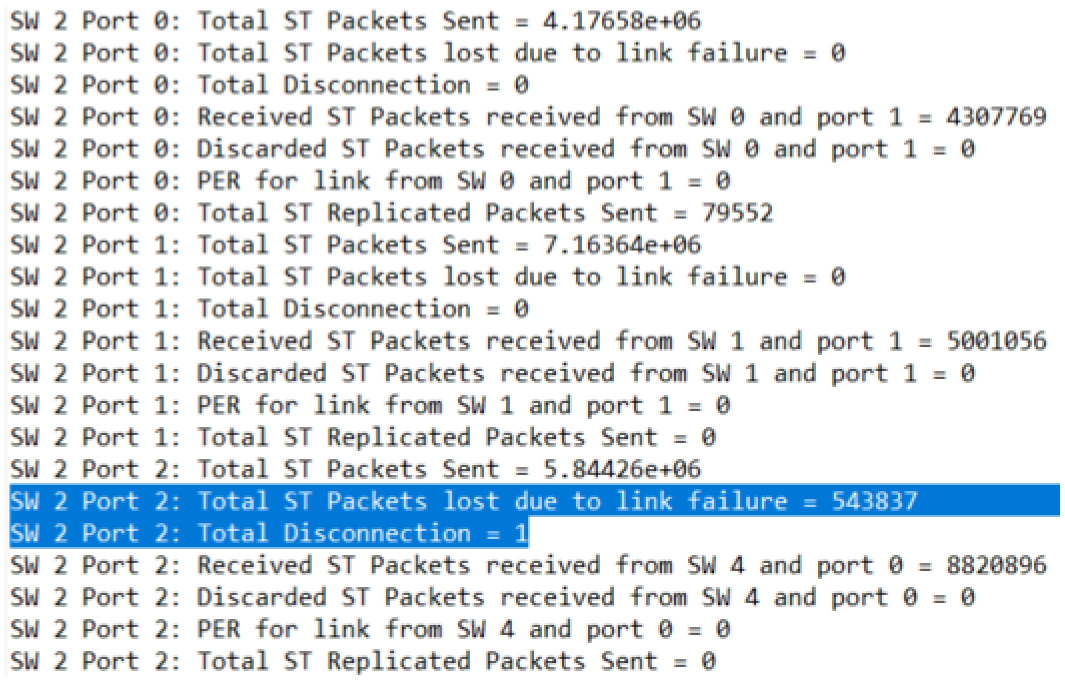


```
SW 2: Total ST Packets Lost = 0
SW 2: Total BE Packets Lost = 57283
```

Fig 27 : Result of run type 1 (3)

In a Time Sensitive Network, since scheduled traffic (ST) packets have higher priority than best effort (BE) packets, it can be said that less ST lost is more important. In Figure 27, we can see that the ST packet lost is zero. It should be noted that these results were obtained under ideal circumstances.

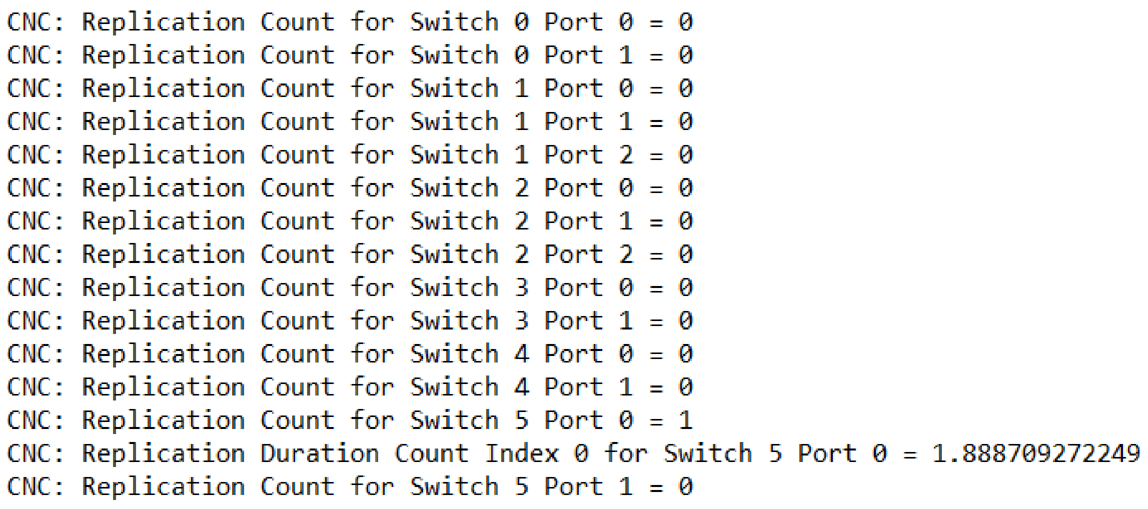
3.3.3 RUN TYPE 2



```
SW 2 Port 0: Total ST Packets Sent = 4.17658e+06
SW 2 Port 0: Total ST Packets lost due to link failure = 0
SW 2 Port 0: Total Disconnection = 0
SW 2 Port 0: Received ST Packets received from SW 0 and port 1 = 4307769
SW 2 Port 0: Discarded ST Packets received from SW 0 and port 1 = 0
SW 2 Port 0: PER for link from SW 0 and port 1 = 0
SW 2 Port 0: Total ST Replicated Packets Sent = 79552
SW 2 Port 1: Total ST Packets Sent = 7.16364e+06
SW 2 Port 1: Total ST Packets lost due to link failure = 0
SW 2 Port 1: Total Disconnection = 0
SW 2 Port 1: Received ST Packets received from SW 1 and port 1 = 5001056
SW 2 Port 1: Discarded ST Packets received from SW 1 and port 1 = 0
SW 2 Port 1: PER for link from SW 1 and port 1 = 0
SW 2 Port 1: Total ST Replicated Packets Sent = 0
SW 2 Port 2: Total ST Packets Sent = 5.84426e+06
SW 2 Port 2: Total ST Packets lost due to link failure = 543837
SW 2 Port 2: Total Disconnection = 1
SW 2 Port 2: Received ST Packets received from SW 4 and port 0 = 8820896
SW 2 Port 2: Discarded ST Packets received from SW 4 and port 0 = 0
SW 2 Port 2: PER for link from SW 4 and port 0 = 0
SW 2 Port 2: Total ST Replicated Packets Sent = 0
```

Fig 28 : Result of run type 2 (1)

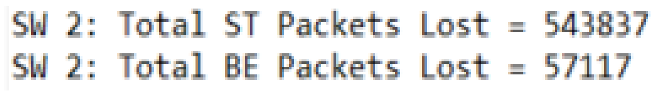
Figure 28 shows the results in run type 2 random replication environment. As with run type 0 and 1 described above, disconnection occurred in SW2 Port2 and it shows that scheduled traffic (ST) packets were considerably lost. In run type 2, the overall efficiency is low because replication occurs randomly. If replication occurs in a node that is already transmitting many packets, the effect of replication is inevitably small.



```
CNC: Replication Count for Switch 0 Port 0 = 0
CNC: Replication Count for Switch 0 Port 1 = 0
CNC: Replication Count for Switch 1 Port 0 = 0
CNC: Replication Count for Switch 1 Port 1 = 0
CNC: Replication Count for Switch 1 Port 2 = 0
CNC: Replication Count for Switch 2 Port 0 = 0
CNC: Replication Count for Switch 2 Port 1 = 0
CNC: Replication Count for Switch 2 Port 2 = 0
CNC: Replication Count for Switch 3 Port 0 = 0
CNC: Replication Count for Switch 3 Port 1 = 0
CNC: Replication Count for Switch 4 Port 0 = 0
CNC: Replication Count for Switch 4 Port 1 = 0
CNC: Replication Count for Switch 5 Port 0 = 1
CNC: Replication Duration Count Index 0 for Switch 5 Port 0 = 1.888709272249
CNC: Replication Count for Switch 5 Port 1 = 0
```

Fig 29 : Result of run type 2 (2)

Figure 29 also shows that the disconnection occurred at SW2 Port2. Unlike the run types tested before, in the situation of random replication, the central network controller (CNC) measured the replication.

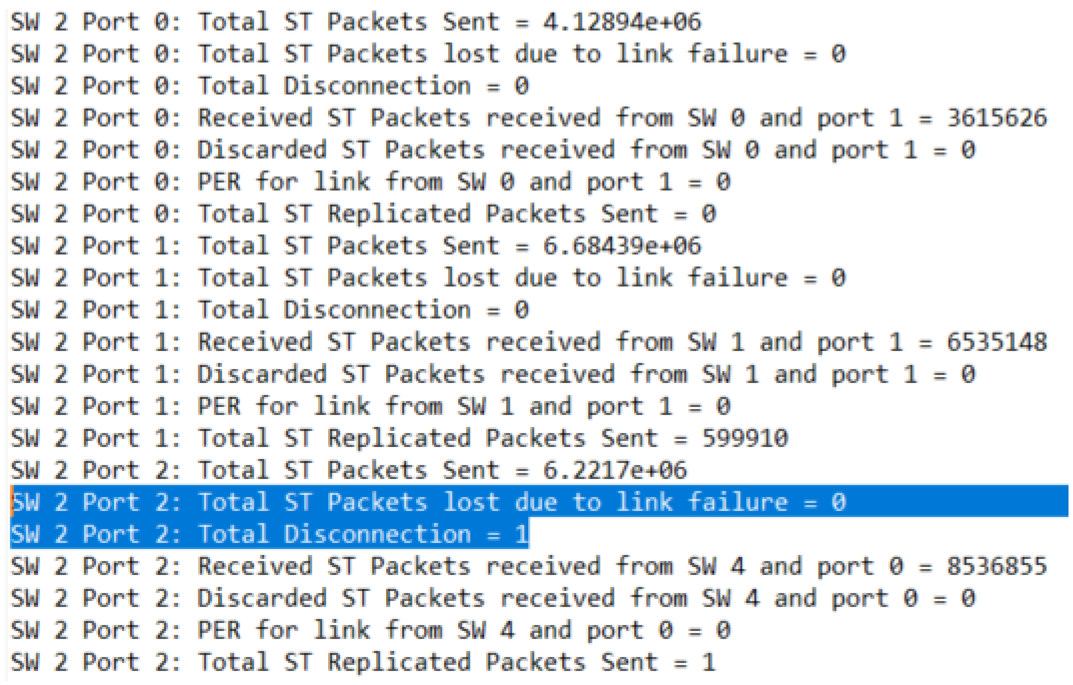


```
SW 2: Total ST Packets Lost = 543837
SW 2: Total BE Packets Lost = 57117
```

Fig 30 : Result of run type 2 (3)

Figure 30 shows that the overall scheduled traffic (ST) packets lost and best effort (BE) packets lost in SW2 where disconnection occurred. Result shows that when random replication situation, scheduled traffic (ST) packets and best effort (BE) packets have large lost.

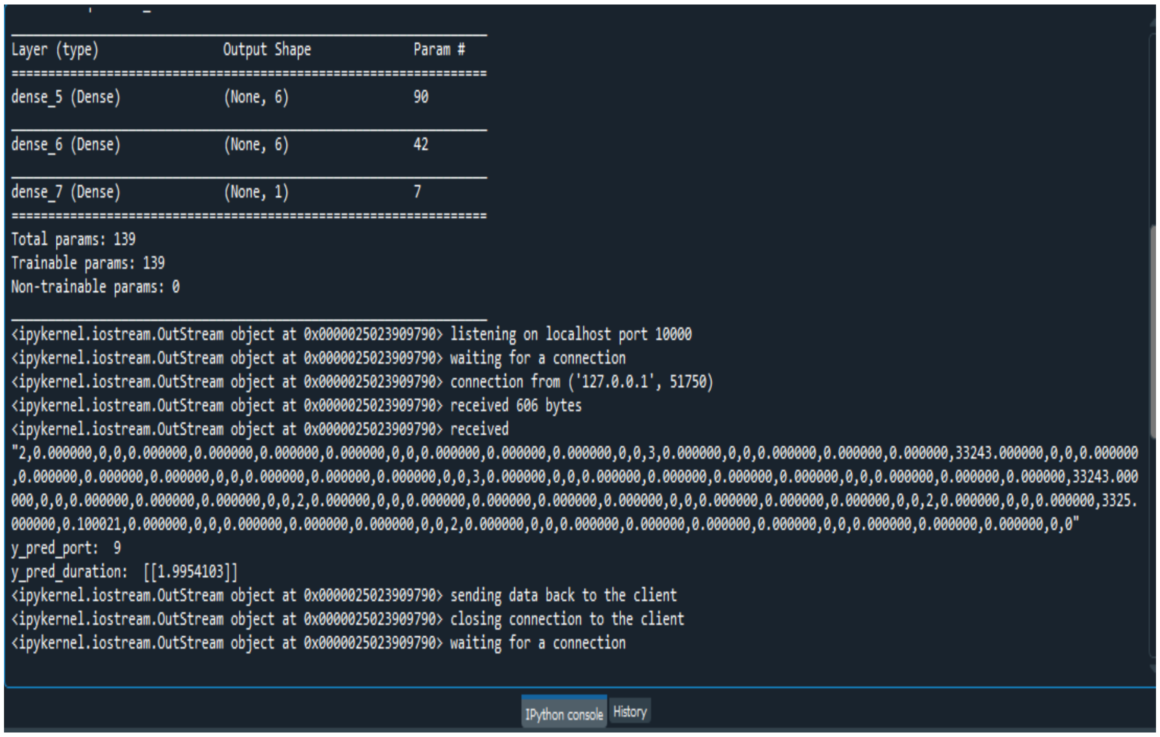
3.3.4 RUN TYPE 3



```
SW 2 Port 0: Total ST Packets Sent = 4.12894e+06
SW 2 Port 0: Total ST Packets lost due to link failure = 0
SW 2 Port 0: Total Disconnection = 0
SW 2 Port 0: Received ST Packets received from SW 0 and port 1 = 3615626
SW 2 Port 0: Discarded ST Packets received from SW 0 and port 1 = 0
SW 2 Port 0: PER for link from SW 0 and port 1 = 0
SW 2 Port 0: Total ST Replicated Packets Sent = 0
SW 2 Port 1: Total ST Packets Sent = 6.68439e+06
SW 2 Port 1: Total ST Packets lost due to link failure = 0
SW 2 Port 1: Total Disconnection = 0
SW 2 Port 1: Received ST Packets received from SW 1 and port 1 = 6535148
SW 2 Port 1: Discarded ST Packets received from SW 1 and port 1 = 0
SW 2 Port 1: PER for link from SW 1 and port 1 = 0
SW 2 Port 1: Total ST Replicated Packets Sent = 599910
SW 2 Port 2: Total ST Packets Sent = 6.2217e+06
SW 2 Port 2: Total ST Packets lost due to link failure = 0
SW 2 Port 2: Total Disconnection = 1
SW 2 Port 2: Received ST Packets received from SW 4 and port 0 = 8536855
SW 2 Port 2: Discarded ST Packets received from SW 4 and port 0 = 0
SW 2 Port 2: PER for link from SW 4 and port 0 = 0
SW 2 Port 2: Total ST Replicated Packets Sent = 1
```

Fig 31 : Result of run type 3 (1)

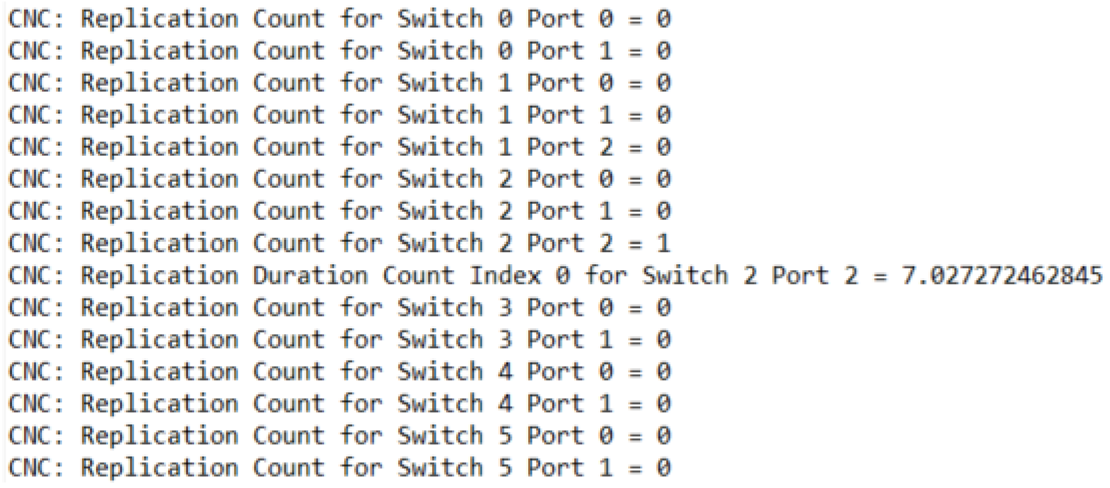
Figure 31 shows the results in run type 3 replication with Machine Learning (ML) environment. As with run type 0 and 1 and 2 described above, disconnection occurred in SW2 Port2. Figure 30 result shows that scheduled traffic (ST) packets were 0. In run type 3 which uses machine learning to derive the location where replication will occur, shows better results than the previous random replication or no replication situation. In the study, the results were derived by machine learning using Python, and it was used with Keras and TensorFlow.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_5 (Dense)              (None, 6)                 90        
_________________________________________________________________
dense_6 (Dense)              (None, 6)                 42        
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 7         
=================================================================
Total params: 139
Trainable params: 139
Non-trainable params: 0
_________________________________________________________________
<ipykernel.iostream.OutStream object at 0x0000025023909790> listening on localhost port 10000
<ipykernel.iostream.OutStream object at 0x0000025023909790> waiting for a connection
<ipykernel.iostream.OutStream object at 0x0000025023909790> connection from ('127.0.0.1', 51750)
<ipykernel.iostream.OutStream object at 0x0000025023909790> received 606 bytes
<ipykernel.iostream.OutStream object at 0x0000025023909790> received
"2,0.000000,0,0,0.000000,0.000000,0.000000,0.000000,0,0,0.000000,0.000000,0.000000,0,0,3,0.000000,0,0,0.000000,0.000000,0.000000,33243.000000,0,0,0.000000
,0.000000,0.000000,0.000000,0,0,0.000000,0.000000,0.000000,0,0,3,0.000000,0,0,0.000000,0.000000,0.000000,0.000000,0,0,0.000000,0.000000,0.000000,33243.000
000,0,0,0.000000,0.000000,0.000000,0,0,2,0.000000,0,0,0.000000,0.000000,0.000000,0.000000,0,0,0.000000,0.000000,0.000000,0,0,2,0.000000,0,0,0.000000,3325.
000000,0.100021,0.000000,0,0,0.000000,0.000000,0.000000,0,0,2,0.000000,0,0,0.000000,0.000000,0.000000,0.000000,0,0,0.000000,0.000000,0.000000,0,0"
y_pred_port:  9
y_pred_duration:  [[1.9954103]]
<ipykernel.iostream.OutStream object at 0x0000025023909790> sending data back to the client
<ipykernel.iostream.OutStream object at 0x0000025023909790> closing connection to the client
<ipykernel.iostream.OutStream object at 0x0000025023909790> waiting for a connection
```

Fig 32 : Example result from Python(Spyder)

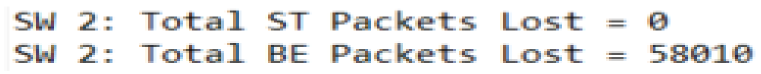
In order to obtain the results shown in Figure 32, the study determined the optimal replication location and replication duration using Feed Forward Neural Network (FNN). The result obtained is expressed as a port number and duration in Python. These port numbers are the values set from 0 to 13 in the CNC and can be seen in Figure 20.



```
CNC: Replication Count for Switch 0 Port 0 = 0
CNC: Replication Count for Switch 0 Port 1 = 0
CNC: Replication Count for Switch 1 Port 0 = 0
CNC: Replication Count for Switch 1 Port 1 = 0
CNC: Replication Count for Switch 1 Port 2 = 0
CNC: Replication Count for Switch 2 Port 0 = 0
CNC: Replication Count for Switch 2 Port 1 = 0
CNC: Replication Count for Switch 2 Port 2 = 1
CNC: Replication Duration Count Index 0 for Switch 2 Port 2 = 7.027272462845
CNC: Replication Count for Switch 3 Port 0 = 0
CNC: Replication Count for Switch 3 Port 1 = 0
CNC: Replication Count for Switch 4 Port 0 = 0
CNC: Replication Count for Switch 4 Port 1 = 0
CNC: Replication Count for Switch 5 Port 0 = 0
CNC: Replication Count for Switch 5 Port 1 = 0
```

Fig 33 : Result of run type 3 (2)

Figure 33 also shows that the disconnection occurred at SW2 Port2. Like the previous run types, it can be seen in Figure 32 that the replication model using Machine Learning properly detects disconnection in the CNC.



```
SW 2: Total ST Packets Lost = 0
SW 2: Total BE Packets Lost = 58010
```

Fig 34 : Result of run type 3 (3)

As mentioned earlier, in a Time Sensitive Network, it is very important that scheduled traffic (ST) packets reach their destination without being lost. In TSN, especially ST, it can cause sensitive issues if it arrives incompletely due to large lost at its destination. For example, if this problem occurs in telemedicine and industrial factories, it can cause fatal human and financial damage. The results of Figure 34 show that the application of Machine Learning to reduce the loss of scheduled traffic packets, which was the purpose of the study can have effective results.

3.3.5 OVERALL RESULT GRAPH

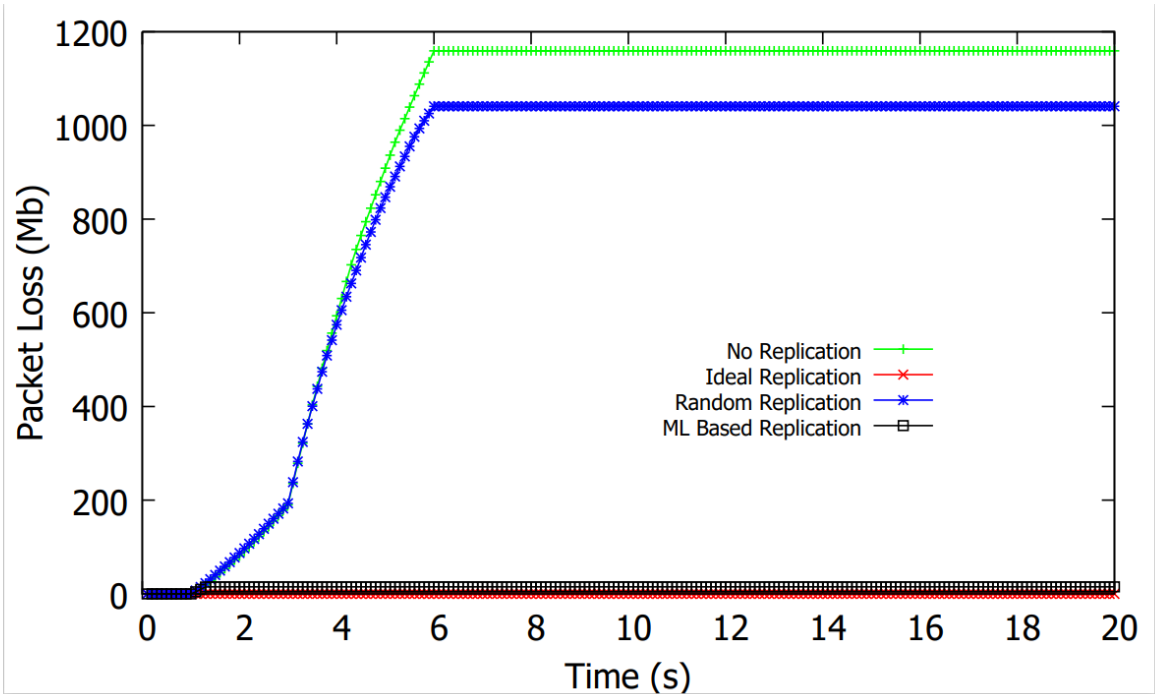


Fig 35 : Comparison of packet loss of each run type

Figure 35 is a graph showing the results in terms of packet loss of the previous run types. The size unit of packet loss is megabytes and the unit of time is seconds. As we can see from the graph above, the packet loss is greatest in the case of no replication, and the next is in the case of random replication. Also, it can be seen that the method using Machine Learning is not significantly different from the ideal replication situation.

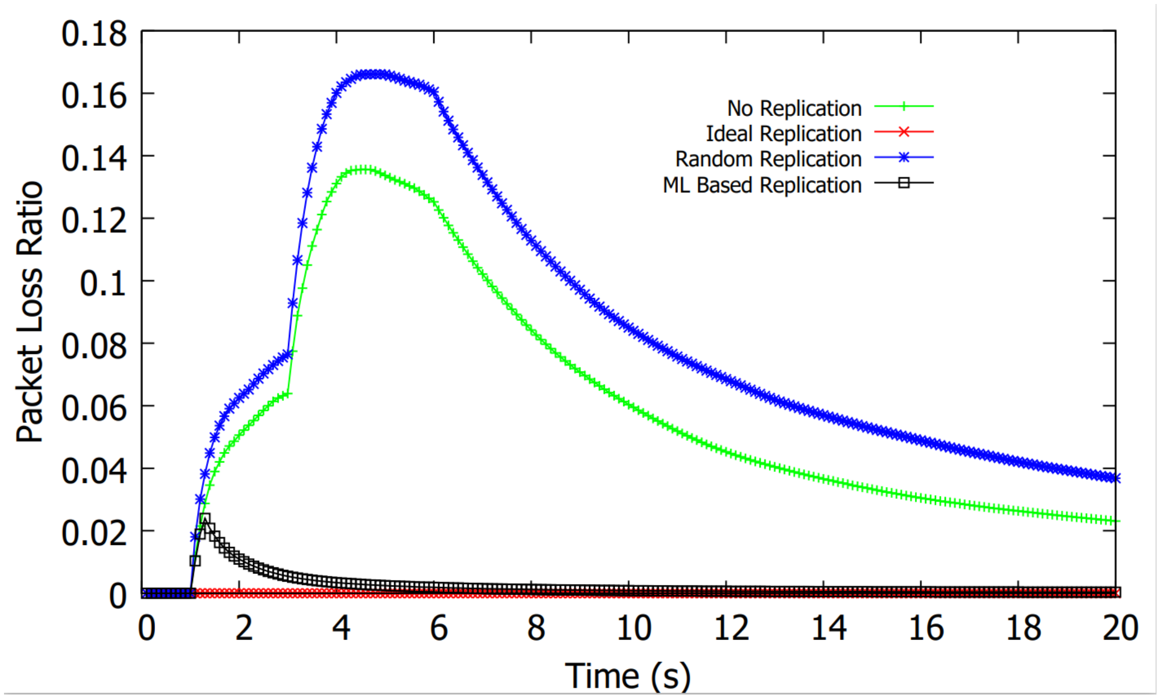


Fig 36 : Comparison of data loss ratio of each run type

Figure 36 shows the packet loss ratio based on Figure 35. It can be seen that degradation starts at about 1 second, and disconnection occurs at about 3 seconds, increasing the data loss ratio. Main point of this figure 35 is that the replication of the Machine Learning method can see that the data loss ratio rises and falls for a while when degradation starts. This is because the method using machine learning predicted disconnection in advance and performed replication.

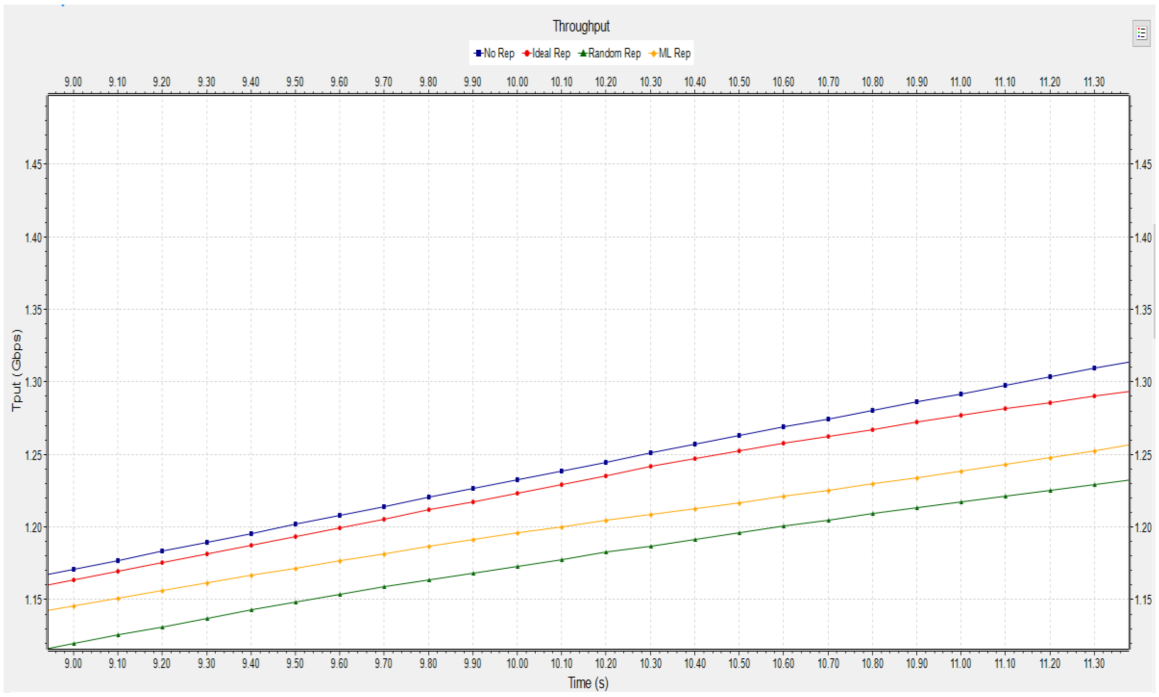


Fig 37 : Comparison of throughput of each run type

Figure 37 is a graph in terms of overall throughput for each run type. The unit of throughput is Gbps, and time is second, which is the same as the graph above. From a throughput perspective, you can see the most in the no replication situation. This is because the packet was transmitted without replication. These results can be expected straightforwardly. In addition, the ideal replication situation was higher than the Machine Learning method or random replication situation.

CHAPTER 4

4.1 Summary

As interest in Time Sensitive Network increases in modern society, the need for a more effective data transmission method has emerged. In particular, with time-sensitive fields such as autonomous vehicles, remote surgery, and automated factories, fault tolerance has become more and more important. Until now, previous studies did not have a way to figure out the location of replication, using machine learning, as we did. For the research, we proposed a method to improve fault-tolerance by combining Machine Leaning. In this graduation thesis, I briefly explained the motivation that started the research in Chapter 1 and the concepts used for research. Also, in Chapter 1.2, works related to research such as Latency, Time Sensitive Network (TSN), Fault-tolerance, and Machine Learning were explained in depth. And in Chapter 3, the overall design and results of the study were described. The main purpose of the study is to use machine learning to predict the ideal replication location and how long replication (duration) should take place, and to test whether this is an effective method. We found that the results of our study can have more ideal results than in the case of no replication or random replication as shown above.

4.2 Future Work

The research proposed in this graduation thesis is the result of a virtual environment in OMNET++. In a real system, there can be many unexpected variables. Also, the challenge of this study is to have enough data to use it to build models and improve accuracy and prediction. Furthermore, as mentioned earlier, it was difficult to compare the results with other studies. This is because there were no data for research papers that proceeded in the same way. Therefore, many studies will have to be conducted to apply it in a real situation. Also, as a future work, through comparing the results with previous papers [33]-[48], and adjust network parameters, use of various machine learning techniques, I think it is desirable to proceed with a method for more accurate replication. In addition, referring to research papers [33]-[40] related to Time Sensitive Network and papers [41]-[48] related to Fault-tolerance, it would be good to find ways to improve in terms of accuracy and latency in the future.

REFERENCES

[1] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. ElBakoury, “Ultra-low latency (ULL) networks: The IEEE TSN and IETF DetNet standards and related 5G ULL research,” IEEE Commun. Surv. & Tut., vol. 21, no. 1, pp. 88–145, 2019.

[2] J. Gettys and K. Nichols, “Bufferbloat: Dark Buffers in the Internet”, ACM Queue, vol. 9, issue 11, 2011.

[3] M. Maier, M. Chowdhury, B. P. Rimal, and D. P. Van, “The tactile Internet: Vision, recent progress, and open challenges,” IEEE Commun.Mag., vol. 54, no. 5, pp. 138–145, May 2016.

[4] D. Delaney, T. Ward, and S. McLoone, “On consistency and network latency in distributed interactive applications: A survey—Part I,”Presence Teleoper. Virtual Environ., vol. 15, no. 2, pp. 218–234, Apr. 2006

[5] D. Delaney, T. Ward, and S. McLoone, “On consistency and network latency in distributed interactive applications: A survey—Part II,” Presence Teleoper. Virtual Environ., vol. 15, no. 4, pp. 465–482, Aug. 2006.

[6] C. S. V. Gutiérrez, L. U. S. Juan, I. Z. Ugarte, and V. M. Vilches, “Timesensitive networking for robotics,” arXiv preprint arXiv:1804.07643, 2018.

[7] F. Prinz, M. Schoeffler, A. Lechler, and A. Verl, “Dynamic real-time orchestration of I4.0 components based on time-sensitive networking,” Procedia CIRP, vol. 72, pp. 910–915, Jun. 2018.

[8] A. Finzi, A. Mifdaoui, F. Frances, and E. Lochin, “Incorporating TSN/BLS in AFDX for mixed-criticality avionics applications: Specification and analysis,” arXiv preprint arXiv:1707.05538, 2017.

[9] P. Schulz et al., “Latency critical IoT applications in 5G: Perspective on the design of radio interface and network architecture,” IEEE Commun. Mag., vol. 55, no. 2, pp. 70–78, Feb. 2017.

[10] C. Bachhuber, E. Steinbach, M. Freundl, and M. Reisslein, "On the minimization of glass-to-glass and glass-to-algorithm delay in video communication," IEEE Trans. Multimedia, vol. 20, no. 1, pp. 238–252, Jan. 2018.

[11] E. Gardiner, "The Avnu alliance theory of operation for TSN-enabled industrial systems," IEEE Commun. Stand. Mag., vol. 2, no. 1, p. 5, Mar. 2018

[12] K. Qian, F. Ren, D. Shan, W. Cheng, and B. Wang, "XpressEth: Concise and efficient converged real-time Ethernet," in Proc. IEEE/ACM 25th Int. Symp. Qual. Service (IWQoS), 2017, pp. 1–6.

[13] M. Amjad, M. H. Rehmani, and S. Mao, "Wireless multimedia cognitive radio networks: A comprehensive survey," IEEE Commun. Surveys Tuts., vol. 20, no. 2, pp. 1056–1103, 2nd Quart., 2018.

[14] R. Trestian, I.-S. Comsa, and M. F. Tuysuz, "Seamless multimedia delivery within a heterogeneous wireless networks environment: Are we there yet?" IEEE Commun. Surveys Tuts., vol. 20, no. 2, pp. 945–977, 2nd Quart., 2018.

[15] "IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks–Amendment 29: Cyclic Queuing and Forwarding," IEEE 802.1Qch-2017 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd(TM)-2015, IEEE Std 802.1Q-2014/Cor 1-2015, IEEE Std 802.1Qbv-2015, IEEE Std 802.1Qbu-2016, IEEE Std 802.1Qbz-2016, and IEEE Std 802.1Qci2017), pp. 1–30, Jun. 2017.

[16] A. Nasrallah, V. Balasubramanian, A. Thyagaturu, M. Reisslein and H. ElBakoury, "Large scale deterministic networking: A simulation evaluation", 2019.

[17] N. Finn, J.-Y. L. Boudec, E. Mohammadpour, J. Zhang, B. Varga, and J. Farkas, "DetNet Bounded Latency," Internet Engineering Task Force, Internet-Draft draft-finn-detnet-bounded-latency-03, Mar. 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/ html/draft-finn-detnet-bounded-latency-03

[18] N. Finn, "Introduction to time-sensitive networking," IEEE Communications Standards Magazine, vol. 2, no. 2, pp. 22–28, 2018.

[19] A. Nasrallah, V. Balasubramanian, A. Thyagaturu, M. Reisslein and H. ElBakoury, "Reconfiguration algorithms for high precision communications in time sensitive networks", *Proc. IEEE Globecom Workshops (GC Wkshps)*, pp. 1-6, Dec. 2019.

[20] M. A. Rahman and M. Krunz, "Stochastic guard-band-aware channel assignment with bonding and aggregation for DSA networks", *IEEE Trans. Wireless Commun.*, vol. 14, no. 7, pp. 3888-3898, Jul. 2015.

[21] M. Kim, J. Min, D. Hyeon and J. Paek, "TAS Scheduling for Real-Time Forwarding of Emergency Event Traffic in TSN," *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, 2020, pp. 1111-1113, doi: 10.1109/ICTC49870.2020.9289458.

[22] "IEEE Draft Standard for Local and metropolitan area networks–Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks Amendment: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements," IEEE P802.1Qcc/D2.0, October 2017, pp. 1–207, Jan. 2017.

[23] A. Bierman, M. Bjorklund, and K. Watsen, "RESTCONF Protocol," RFC 8040, Jan. 2017. [Online]. Available: https://rfc-editor.org/rfc/ rfc8040.txt

[24] R. Enns, M. Bjorklund, A. Bierman, and J. Schnwlder, "Network Configuration Protocol (NETCONF)," RFC 6241, Jun. 2011. [Online]. Available: https://rfc-editor.org/rfc/rfc6241.txt

[25] R. Patil and M. Zawar, "Improving replication results through directory server data replication," *2017 International Conference on Trends in Electronics and Informatics (ICEI)*, 2017, pp. 677-681, doi: 10.1109/ICOEI.2017.8300788.

[26] J. Lee and S. Park, "Time-Sensitive Network Profile Service for Enhanced In-Vehicle Stream Reservation," *2019 4th International Conference on Control, Robotics and Cybernetics (CRC)*, 2019, pp. 133-136, doi: 10.1109/CRC.2019.00035.

[27] Çınar ZM, Abdussalam Nuhu A, Zeeshan Q, Korhan O, Asmael M, Safaei B. Machine Learning in Predictive Maintenance towards Sustainable Smart Manufacturing in Industry 4.0. *Sustainability*. 2020; 12(19):8211. https://doi.org/10.3390/su12198211

[28] Nielsen, Frank. (2016). Parallel Linear Algebra. 10.1007/978-3-319-21903-5_5.

[29] S. Sommer et al., "RACE: A centralized platform computer based architecture for automotive applications," in Proc. IEEE Int. Elect. Veh. Conf., Santa Clara, CA, USA, 2013, pp. 1–6.

[30] Vijay Kotu, Bala Deshpande, in Data Science (Second Edition), 2019

[31] Aslam, Sheraz & Herodotou, Herodotos & Ayub, Nasir & Mohsin, Syed Muhammad. (2019). Deep Learning Based Techniques to Enhance the Performance of Microgrids: A Review. 10.1109/FIT47737.2019.00031.

[32] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186.

[33] I. Álvarez, J. Proenza, M. Barranco and M. Knezic, "Towards a time redundancy mechanism for critical frames in time-sensitive networking," *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2017, pp. 1-4, doi: 10.1109/ETFA.2017.8247721.

[34] Álvarez, Inés & Proenza, Julian & Barranco, Manuel. (2018). Mixing Time and Spatial Redundancy Over Time Sensitive Networking. 63-64. 10.1109/DSN-W.2018.00031.

[35] I. Álvarez, M. Barranco and J. Proenza, "Towards a Fault-Tolerant Architecture Based on Time Sensitive Networking," *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2018, pp. 1113-1116, doi: 10.1109/ETFA.2018.8502614.

[36] L. Su *et al.*, "Synthesizing Fault-Tolerant Schedule for Time-Triggered Network Without Hot Backup," in *IEEE Transactions on Industrial Electronics*, vol. 66, no. 2, pp. 1345-1355, Feb. 2019, doi: 10.1109/TIE.2018.2833022.

[37] A. A. Atallah, G. B. Hamad and O. A. Mohamed, "Fault-Resilient Topology Planning and Traffic Configuration for IEEE 802.1Qbv TSN Networks," *2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*, 2018, pp. 151-156, doi: 10.1109/IOLTS.2018.8474201.

[38] I. Álvarez, D. Čavka, J. Proenza and M. Barranco, "Simulation of the Proactive Transmission of Replicated Frames Mechanism over TSN," *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2019, pp. 1375-1378, doi: 10.1109/ETFA.2019.8868997.

[39] M. Pahlevan, J. Schmeck and R. Obermaisser, "Evaluation of TSN Dynamic Configuration Model for Safety-Critical Applications," *2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable*

Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), 2019, pp. 566-571, doi: 10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00086.

[40] A. A. Atallah, G. B. Hamad and O. A. Mohamed, "Routing and Scheduling of Time-Triggered Traffic in Time-Sensitive Networks," in *IEEE Transactions on Industrial Informatics*, vol. 16, no. 7, pp. 4525-4534, July 2020, doi: 10.1109/TII.2019.2950887.

[41] N. Desai and S. Punnekkat, "Enhancing Fault Detection in Time Sensitive Networks using Machine Learning," *2020 International Conference on COMmunication Systems & NETworkS (COMSNETS)*, 2020, pp. 714-719, doi: 10.1109/COMSNETS48256.2020.9027357.

[42] O. Daniel, G. E. Juan, C. Lua and O. Roman, "Failure Detection in TSN Startup Using Deep Learning," *2020 IEEE 23rd International Symposium on Real-Time Distributed Computing (ISORC)*, 2020, pp. 140-141, doi: 10.1109/ISORC49007.2020.00028.

[43] D. Onwuchekwa, J.-J. Foo and R. Obermaisser, Fault injection framework for time triggered ethernet, apr 2018, [online] Available: https://doi.org/10.5281/zenodo.1485392.

[44] Time-Sensitive Networking Task Group, 11 2016, [online] Available: http://www.ieee802.org/1/pages/tsn.html.

[45] M. Gutiérrez, W. Steiner, R. Dobrin and S. Punnekkat, "A configuration agent based on the time-triggered paradigm for real-time networks", *2015 IEEE World Conference on Factory Communication Systems (WFCS)*, pp. 1-4, May 2015.

[46] L. Lo Bello and W. Steiner, "A perspective on ieee time-sensitive networking for industrial communication and automation systems", *Proceedings of the IEEE*, vol. 107, pp. 1094-1120, June 2019.

[47] T. Gerhard, T. Kobzan, I. Blöcher and M. Hen-del, "Software-defined Flow Reservation: Configuring IEEE 802.1Q Time-Sensitive Networks by the Use of Software-Defined Networking", *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 216-223, Sep. 2019.

[48] S. Jha, S. Banerjee, T. Tsai, S. K. S. Hari, M. B. Sullivan, Z. T. Kalbarczyk, et al., "ML-Based Fault Injection for Autonomous Vehicles: A Case for Bayesian Fault

Injection", *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 112-124, June 2019.

[49] Gavrilut, Voica & Zarrin, Bahram & Pop, Paul & Samii, Soheil. (2017). Fault-tolerant topology and routing synthesis for IEEE time-sensitive networking. 267-276. 10.1145/3139258.3139284.

[50] A. V. Sutagundar, V. S. Bennur, A. M. Anusha and K. N. Bhanu, "Agent based fault tolerance in wireless sensor networks," *2016 International Conference on Inventive Computation Technologies (ICICT)*, 2016, pp. 1-6, doi: 10.1109/INVENTIVE.2016.7823265.

[51] A. Nagarajan and A. Vaddadi, "Automated Fault-Tolerance Testing," *2016 IEEE Ninth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2016, pp. 275-276, doi: 10.1109/ICSTW.2016.34.

[52] Huang, Kai & Wan, Xinming & Wang, Ke & Jiang, Xiaowen & Chen, Junjian & Deng, Qingtang & Xu, Wenyuan & Peng, Yonggang & Liu, Zhili. (2021). Reliability-Aware Multipath Routing of Time-Triggered Traffic in Time-Sensitive Networks. Electronics. 10. 125. 10.3390/electronics10020125.

[53] Nasrallah, Ahmed & B, Venkatraman & Thyagaturu, Akhilesh & Reisslein, Martin & Elbakoury, Hesham. (2019). Reconfiguration Algorithms for High Precision Communications in Time Sensitive Networks. 1-6. 10.1109/GCWkshps45667.2019.9024705.

[54] Importance of Internet Exchange Point (IXP) infrastructure for 5G : Estimating the impact of 5G use cases. / Hoeschele, Thomas; Dietzel, Christoph; Kopp, Daniel; Fitzek, Frank H.P.; Reisslein, Martin.
In: Telecommunications Policy, Vol. 45, No. 3, 102091, 04.2021.

[55] Fitzek, Frank HP and Li, Shu-Chen and Speidel, Stefanie and Strufe, Thorsten and Simsek, Meryem and Reisslein, Martin. Tactile internet: With human-in-the-Loop. (2021). Academic Press.

[56] P. Shantharama, A. S. Thyagaturu and M. Reisslein, "Hardware-Accelerated Platforms and Infrastructures for Network Functions: A Survey of Enabling Technologies and Research Studies," in IEEE Access, vol. 8, pp. 132021-132085, 2020, doi: 10.1109/ACCESS.2020.3008250.

[57] P. Shantharama, A. S. Thyagaturu, N. Karakoc, L. Ferrari, M. Reisslein and A. Scaglione, "LayBack: SDN Management of Multi-Access Edge Computing (MEC) for Network Access Services and Radio Resource Sharing," in IEEE Access, vol. 6, pp. 57545-57561, 2018, doi: 10.1109/ACCESS.2018.2873984.

[58] A. S. Thyagaturu, Y. Dashti and M. Reisslein, "SDN-Based Smart Gateways (Sm-GWs) for Multi-Operator Small Cell Network Management," in IEEE Transactions on Network and Service Management, vol. 13, no. 4, pp. 740-753, Dec. 2016, doi: 10.1109/TNSM.2016.2605924.

[59] J. Rischke, P. Sossalla, S. Itting, F. H. P. Fitzek and M. Reisslein, "5G Campus Networks: A First Measurement Study," in IEEE Access, vol. 9, pp. 121786-121803, 2021, doi: 10.1109/ACCESS.2021.3108423.

[60] B, Venkatraman & Aloqaily, Moayad & Reisslein, Martin. (2020). An SDN architecture for time sensitive industrial IoT. Computer Networks. 186. 107739. 10.1016/j.comnet.2020.107739.

[61] V. Balasubramanian, M. Aloqaily, M. Reisslein and A. Scaglione, "Intelligent Resource Management at the Edge for Ubiquitous IoT: An SDN-Based Federated Learning Approach," in IEEE Network, vol. 35, no. 5, pp. 114-121, September/October 2021, doi: 10.1109/MNET.011.2100121.

]

APPENDIX A

[OMNET++ PROGRAM SOURCE CODE]

```
/*
 * This code is SW source code. There are many other source code such as SC, CNC,
SINK.
 * Since the length of the code is very long, I will attach only the code related to SW.
 * Thank you.
 */
#include "scMessage_m.h"
#include "netChangeMsg_m.h"
#include "perMessage_m.h"
#include "sw.h"
#include "channel_c.h"

//the number of queues per egress SW port (channel)
#define trafficClasses 2
#define ST 0
#define BE 1
#define Qsize 512*1024

using namespace omnetpp;

Define_Module(sw);

Define_Channel(channel_c);

sw::sw()            // constructor
{

}

sw::~sw()           // destructor
{
   cOwnedObject *Del=NULL;
   int OwnedSize = this->defaultListSize();
   for(int i = 0; i < OwnedSize; i++)
   {
      Del = this->defaultListGet(0);
      this->drop(Del);
      delete Del;
   }
   return;
}

void sw::initialize()
{
   std::string g = getParentModule()->par("resultFile");
```

```
    outfile.open("../results/"+g, std::ofstream::out | std::ofstream::app);
    maxWindowTime = (simtime_t) par("maxWindowTime");                    // Window
Time for transmission for each gate
    capacity = (double) par("maxCableBitrate");                  // Max Link Bandwidth in
bits per second 1 Gbps
    reconfig = (bool) par("reconfig");
    perThresh = (double) par("perThresh");
    if ((int)par("runType") == 1)
        perThresh = 0;
    // initializing queues
    switchChannelQueue.resize(gateSize("sw_sw"));
    priR.resize(gateSize("sw_sw"));
    priRchange.resize(gateSize("sw_sw"));
    for (int j = 0; j < gateSize("sw_sw"); j++)     // for each port do...
    {
        for (int i = 0; i < trafficClasses; i++)       // for each traffic class do...
        {
            switchChannelQueue[j].queue.push_back(new cPacketQueue(("SW_Q-
"+std::to_string(i)+"-Gate_Index "+std::to_string(j)).c_str()));          // switch channel
queue
            switchChannelQueue[j].sizeQ.push_back(Qsize);
            switchChannelQueue[j].packetLoss.push_back(0);
        }
        switchChannelQueue[j].timer = new cMessage("swTimer");          // corresponding
timer for queue
        switchChannelQueue[j].timer->setKind(j);                       // Port identifier
        switchChannelQueue[j].timer->setName(std::string("SW-
"+std::to_string(getIndex()) + "-swTimer-Port " + std::to_string(j)).c_str());
        priR[j] = par("priR");               // initializing the ST slot ratio from ini config file
        priRchange[j] = registerSignal(std::string("priRchange-port-" +
std::to_string(j)).c_str());
        getEnvir()->addResultRecorders(this, priRchange[j], std::string("priRchange-port-
" + std::to_string(j)).c_str(), getProperties()->get("statisticTemplate", "priRchange"));
        emit(priRchange[j], priR[j]);
        switchChannelQueue[j].duplicate = false;    // duplication ID for port

        switchChannelQueue[j].stPacketTput = 0;
        switchChannelQueue[j].numOfstPacketsLostDueToLinkFailure = 0;
        switchChannelQueue[j].numOfDisconnections = 0;
        switchChannelQueue[j].cnc_st_pkt_tput = 0;
        switchChannelQueue[j].cnc_st_pkt_loss_due_to_disconnection = 0;
        switchChannelQueue[j].cnc_num_of_disconnections = 0;
        switchChannelQueue[j].received_st_packet_lost = 0;
        switchChannelQueue[j].received_total_st_packet = 0;
        switchChannelQueue[j].received_per = 0;
```

```
      switchChannelQueue[j].cnc_received_st_packet_lost = 0;
      switchChannelQueue[j].cnc_received_total_st_packet = 0;
      switchChannelQueue[j].cnc_received_per = 0;

    }
    for (int i = 0; i < trafficClasses; i++)
      sinkChannelQueue[i] = new cPacketQueue("sinkQ");          // sink channel queue
    lossPktST = 0;                              // packet loss for ST init to 0
    lossPktBE = 0;                              // packet loss for BE init to 0
    SINKmsg = new cMessage("SINKmsg");          // timer for switch to sink channel port
    bsActive = (bool) par("bsActive");          // Bandwidth sharing activation
    SThigh = (simtime_t) par("SThigh");         // ST delay high threshold
    STlow = (simtime_t) par("STlow");           // ST delay low threshold
    /* Qcc parameters */
    cncQ = new cPacketQueue("cncQ");
    cncTimer = new cMessage("cncTimer");
    scpTimer.resize(gateSize("inout_sw_scp"));
    scpQ.resize(gateSize("inout_sw_scp"));
    for (int i = 0; i < scpTimer.size(); i ++)
    {
      scpTimer[i] = new cMessage("scpTimer");
      scpTimer[i]->setKind(i);
      scpQ[i] = new cPacketQueue("scpQ");
    }
    myAddress = par("address");
    numSW = par("numSW");
    numStreams = 0;

    cnc_stPktLost = 0;
    cnc_bePktLost = 0;

    rtable.push_back({std::make_pair(0,0)});
    updateRouting();

    // Create signals for notification method of switch parameters
    stTputSignalId =
registerSignal(std::string("stTput"+std::to_string(getIndex())).c_str());
    stLostDueToLinkSignalId =
registerSignal(std::string("stLostLink"+std::to_string(getIndex())).c_str());
    stNumDisconnectSignalId =
registerSignal(std::string("stDisconnect"+std::to_string(getIndex())).c_str());
    stPktLossSignalId =
registerSignal(std::string("stPktLoss"+std::to_string(getIndex())).c_str());
    bePktLossSignalId =
registerSignal(std::string("bePktLoss"+std::to_string(getIndex())).c_str());
```

```
waiting_for_reply = false;

// Watch Parameters (hard-coded for current topology considered)
switch (getIndex())
{
case 0:
   createWatch("PriR for index 0", priR[0]);
   createWatch("PriR for index 1", priR[1]);
   createWatch("ST Packet Loss index 0: ",switchChannelQueue[0].packetLoss[ST]);
   createWatch("ST Packet Loss index 1: ",switchChannelQueue[1].packetLoss[ST]);
   break;
case 1:
   createWatch("PriR for index 0", priR[0]);
   createWatch("PriR for index 1", priR[1]);
   createWatch("PriR for index 2", priR[2]);
   createWatch("ST Packet Loss index 0: ",switchChannelQueue[0].packetLoss[ST]);
   createWatch("ST Packet Loss index 1: ",switchChannelQueue[1].packetLoss[ST]);
   createWatch("ST Packet Loss index 2: ",switchChannelQueue[2].packetLoss[ST]);
   break;
case 2:
   createWatch("PriR for index 0", priR[0]);
   createWatch("PriR for index 1", priR[1]);
   createWatch("PriR for index 2", priR[2]);
   createWatch("ST Packet Loss index 0: ",switchChannelQueue[0].packetLoss[ST]);
   createWatch("ST Packet Loss index 1: ",switchChannelQueue[1].packetLoss[ST]);
   createWatch("ST Packet Loss index 2: ",switchChannelQueue[2].packetLoss[ST]);
   break;
case 3:
   createWatch("PriR for index 0", priR[0]);
   createWatch("PriR for index 1", priR[1]);
   createWatch("ST Packet Loss index 0: ",switchChannelQueue[0].packetLoss[ST]);
   createWatch("ST Packet Loss index 1: ",switchChannelQueue[1].packetLoss[ST]);
   break;
case 4:
   createWatch("PriR for index 0", priR[0]);
   createWatch("PriR for index 1", priR[1]);
   createWatch("ST Packet Loss index 0: ",switchChannelQueue[0].packetLoss[ST]);
   createWatch("ST Packet Loss index 1: ",switchChannelQueue[1].packetLoss[ST]);
   break;
case 5:
   createWatch("PriR for index 0", priR[0]);
   createWatch("PriR for index 1", priR[1]);
   createWatch("ST Packet Loss index 0: ",switchChannelQueue[0].packetLoss[ST]);
   createWatch("ST Packet Loss index 1: ",switchChannelQueue[1].packetLoss[ST]);
```

```
        break;
    }
    createWatch("Number of Streams: ", numStreams);

}

void sw::updateRouting()
{
    /* Build Routing Table here...
    // Brute force approach -- every node does topology discovery on its own,
    // and finds routes to all other nodes independently, at the beginning
    // of the simulation. This could be improved: (1) central routing database,
    // (2) on-demand route calculation
     */
    cTopology *topo = new cTopology("topo");
    topo->extractByParameter("address");
    EV << "cTopology found " << topo->getNumNodes() << " nodes\n";
    cTopology::Node *thisNode = topo->getNodeFor(getParentModule()-
>getSubmodule(getName(), getIndex()));
    // find and store next hops
    for (int i = 0; i < topo->getNumNodes(); i++)
    {
        if (topo->getNode(i) == thisNode)
            continue;  // skip ourselves
        topo->calculateUnweightedSingleShortestPathsTo(topo->getNode(i));
        if (thisNode->getNumPaths() == 0)
            continue;  // not connected
        cGate *parentModuleGate = thisNode->getPath(0)->getLocalGate();
        int gateIndex = parentModuleGate->getIndex();
        int address = topo->getNode(i)->getModule()->par("address");
        rtable[0][address] = gateIndex;
        EV << " towards SW " << address << " Port is " << rtable[0][address] << endl;
    }
    delete topo;
}

void sw::getRouting(int switchIndex, int portIndexReplicateFrom)
{
    // need to disable the previous port and find a new path , i.e., see which port is selected
to the destination switches that were previously used for the disabled source port chosen
    cTopology *topo = new cTopology("topo");
    topo->extractByParameter("address");       // Extract all the switches and their ports
in a graph style G(V,E)...
    rtable.push_back({std::make_pair(0,0)});      // creating a new empty route table for
new path with disabled port chosen to replicate
```

```
    if (topo->getNode(switchIndex)->getModule()->gate("sw_sw$o",
portIndexReplicateFrom)->isConnected())
    {
        if (topo->getNodeFor(topo->getNode(switchIndex)->getModule())-
>getNumOutLinks() != getParentModule()->getSubmodule("sw", switchIndex)-
>gateSize("sw_sw$o"))
        {
            // if out links in cTopo is different from the number of gates associated with the
switch module, then we need to find the right out link to disable...
            int nextAdd = topo->getNode(switchIndex)->getModule()->gate("sw_sw$o",
portIndexReplicateFrom)->getNextGate()->getOwnerModule()->getIndex();
            for (int i = 0; i < topo->getNodeFor(topo->getNode(switchIndex)-
>getModule())->getNumOutLinks(); i++)
            {
                if (topo->getNodeFor(topo->getNode(switchIndex)->getModule())-
>getLinkOut(i)->getRemoteGate()->getOwnerModule()->getIndex() == nextAdd)
                {
                    EV << "Disabling cTopo Node " << topo->getNode(switchIndex)-
>getModule()->getIndex() << " with link out/port " << topo->getNodeFor(topo-
>getNode(switchIndex)->getModule())->getLinkOut(i)->getLocalGate()->getIndex()
<< endl;
                    topo->getNodeFor(topo->getNode(switchIndex)->getModule())-
>getLinkOut(i)->disable();
                    break;
                }
            }
        }
        else
        {
            topo->getNodeFor(topo->getNode(switchIndex)->getModule())-
>getLinkOut(portIndexReplicateFrom)->disable();      // Disable the port in the topology
        }
    }
    // Get new routing table with disabled port
    cTopology::Node *thisNode = topo->getNodeFor(this);
    // find and store next hops
    for (int i = 0; i < topo->getNumNodes(); i++)
    {
        if (topo->getNode(i) == thisNode)
            continue;  // skip ourselves
        topo->calculateUnweightedSingleShortestPathsTo(topo->getNode(i));
        if (thisNode->getNumPaths() == 0)
            continue;  // not connected
        cGate *parentModuleGate = thisNode->getPath(0)->getLocalGate();
        int gateIndex = parentModuleGate->getIndex();
        int address = topo->getNode(i)->getModule()->par("address");
        rtable[rtable.size()-1][address] = gateIndex;
        EV << " towards SW " << address << " Port is " << rtable[rtable.size()-1][address]
<< endl;
    }
    delete topo;
}

void sw::forwardCNC(cMessage *msg)
{
    if (gate("inout_sw_cnc$o")->getTransmissionChannel()-
>getTransmissionFinishTime() <= simTime())
    {
        send(msg, "inout_sw_cnc$o");
    }
    else
    {
        cncQ->insert(dynamic_cast<cPacket*>(msg));
        if (!cncTimer->isScheduled())
            scheduleAt(gate("inout_sw_cnc$o")->getTransmissionChannel()-
>getTransmissionFinishTime(), cncTimer);
    }
}

void sw::forwardSCP(ScMessage *msg)
{
    if (gate("inout_sw_scp$o", msg->getSCPindex())->getTransmissionChannel()-
>getTransmissionFinishTime() <= simTime())
    {
        send(msg, "inout_sw_scp$o", msg->getSCPindex());
    }
    else
    {
        scpQ[msg->getSCPindex()]->insert(msg);
        if (!scpTimer[msg->getSCPindex()]->isScheduled())
            scheduleAt(gate("inout_sw_scp$o", msg->getSCPindex())-
>getTransmissionChannel()->getTransmissionFinishTime(), scpTimer[msg-
>getSCPindex()]);
    }
}

void sw::handleCNC(cMessage *msg)
{
    std::vector<std::tuple<int,int,u_int>>::iterator p;              // a pointer to the local
stream table
```

```
   ScMessage *resMsg = check_and_cast<ScMessage *>(msg);       // cast received new
message to source message
   if (strcmp(resMsg->getMsgType(), "Duplicate Port") == 0)       // sets the duplication
flag for a port to true
   {
      // set the replicated port flag to true
      switchChannelQueue[resMsg->getPortSrc()].duplicate = true;
      if (gate("sw_sw$o",resMsg->getPortSrc())->isConnected())
         check_and_cast<channel_c *>(gate("sw_sw$o",resMsg->getPortSrc())-
>getChannel())->setDuplicated(true);
      waiting_for_reply = false;
      EV << "Setting Port " << resMsg->getPortSrc() << " duplication flag to true" <<
endl;
      delete resMsg;
   }
   else if (strcmp(resMsg->getMsgType(), "Duplicate Port End") == 0)       // sets the
duplication flag for a port to false
   {
      // set the replicated port flag to false
      switchChannelQueue[resMsg->getPortSrc()].duplicate = false;
      if (gate("sw_sw$o",resMsg->getPortSrc())->isConnected())
         check_and_cast<channel_c *>(gate("sw_sw$o",resMsg->getPortSrc())-
>getChannel())->setDuplicated(false);
      EV << "Setting Port " << resMsg->getPortSrc() << " duplication flag to false" <<
endl;
      delete resMsg;
   }
   else if (strcmp(resMsg->getMsgType(), "Replication Message") == 0)      // adds a
replicated stream to the replication table at the source of the replicated stream's path with
initial sequence 0
   {
      frerSrcStreamTable.push_back(std::make_tuple(resMsg->getFlowID(),resMsg-
>getSWindex(),0));
      delete resMsg;
   }
   else if (strcmp(resMsg->getMsgType(), "Elimination Message") == 0)      // adds a
replicated stream to the elimination table at the destination of the replicated stream's path
with initial sequence 1
   {
      frerDestStreamTable.push_back(std::make_tuple(resMsg->getFlowID(),resMsg-
>getSWindex(),1));
      delete resMsg;
   }
   else if (strcmp(resMsg->getMsgType(), "Add Route") == 0)       // adds a new routing
table for the replicated streams to take (2nd shortest path usually)
```

```
    {
        // Need to get the new route similar to CNC by disabling the link which I want to
replicate from and find new path
        getRouting(resMsg->getSWindex(), resMsg->getPortSrc());
        delete resMsg;
    }
    else if (strcmp(resMsg->getMsgType(), "Remove Route") == 0)        // removes last
routing table for the replicated streams to take (2nd shortest path usually)
    {
        rtable.pop_back();
        delete resMsg;
    }
    else if (strcmp(resMsg->getMsgType(), "Duplicate Stream") == 0)     // adjusts ST
gating ratio since we added duplicate streams along some nodes
    {
        if (reconfig && resMsg->getPriR() != priR[resMsg->getKind()])
        {
            priR[resMsg->getKind()] = resMsg->getPriR();
            emit(priRchange[resMsg->getKind()], priR[resMsg->getKind()]);
        }
        delete resMsg;
    }
    else if (strcmp(resMsg->getMsgType(), "Net Change") == 0)       // similar to
duplicate stream, usually just adjust the ST gating ratio
    {
        if (reconfig && resMsg->getPriR() != priR[resMsg->getKind()])
        {
            priR[resMsg->getKind()] = resMsg->getPriR();
            emit(priRchange[resMsg->getKind()], priR[resMsg->getKind()]);
        }
        delete resMsg;
    }
    else if (strcmp(resMsg->getMsgType(), "Reset SW") == 0)     // reset cnc_sw
parameters for ML model
    {
        for (int i = 0; i < gateSize("sw_sw"); i++)
        {
            switchChannelQueue[i].cnc_st_pkt_tput = 0;
            switchChannelQueue[i].cnc_st_pkt_loss_due_to_disconnection = 0;
            switchChannelQueue[i].cnc_num_of_disconnections = 0;
            switchChannelQueue[i].cnc_received_st_packet_lost = 0;
            switchChannelQueue[i].cnc_received_total_st_packet = 0;
            switchChannelQueue[i].cnc_received_per = 0;
        }
        cnc_stPktLost = 0;
        cnc_bePktLost = 0;
    }
    else
    {
        // if approved by cnc, then we add flow ID to local table of source Gateway switch...
        if (resMsg->getApproval())
        {
            if (resMsg->getSrcAddr() == getIndex())
            {
                localStreamTable.push_back(std::tuple<int,int,long>(resMsg->getFlowID(),
resMsg->getSrcAddr(), -1));        // push back stream < Flow ID, Switch ID, Sequence
Number >
                numStreams++;
                if (reconfig && resMsg->getPriR() != priR[rtable[0][resMsg-
>getDestAddr()]])
                {
                    priR[rtable[0][resMsg->getDestAddr()]] = resMsg->getPriR();
                    emit(priRchange[rtable[0][resMsg->getDestAddr()]], priR[rtable[0][resMsg-
>getDestAddr()]]);
                }
                // Notify attached source
                forwardSCP(resMsg);
            }
            else
            {
                if (reconfig && resMsg->getPriR() != priR[rtable[0][resMsg-
>getDestAddr()]])
                {
                    priR[rtable[0][resMsg->getDestAddr()]] = resMsg->getPriR();
                    emit(priRchange[rtable[0][resMsg->getDestAddr()]], priR[rtable[0][resMsg-
>getDestAddr()]]);
                }
                delete resMsg;
            }
        }
        else if (!resMsg->getApproval())            // Not Approved
        {
            if (strcmp(resMsg->getMsgType(), "Stream Complete") == 0)
            {
                if (resMsg->getDuplicated())    // if stream is a duplicated stream...
                {
                    if (reconfig && resMsg->getPriR() != priR[resMsg->getKind()])
                    {
                        priR[resMsg->getKind()] = resMsg->getPriR();
                        emit(priRchange[resMsg->getKind()], priR[resMsg->getKind()]);
                    }
                    // Need to remove the frer dest and src table record for the finished replicated
stream
                    for (int i = 0; i < frerSrcStreamTable.size(); i++)
                    {
                        if (resMsg->getFlowID() == std::get<0>(frerSrcStreamTable[i]) &&
resMsg->getSWindex() == std::get<1>(frerSrcStreamTable[i]))
                        {
                            EV << "Src FRER Table -- Removing Flow " <<
std::get<0>(frerSrcStreamTable[i]) << " Gateway " <<
std::get<1>(frerSrcStreamTable[i]) << endl;
                            frerSrcStreamTable.erase(frerSrcStreamTable.begin() + i);
                            break;
                        }
                    }
                    for (int i = 0; i < frerDestStreamTable.size(); i++)
                    {
                        if (resMsg->getFlowID() == std::get<0>(frerDestStreamTable[i]) &&
resMsg->getSWindex() == std::get<1>(frerDestStreamTable[i]))
                        {
                            EV << "Dest FRER Table -- Removing Flow " <<
std::get<0>(frerDestStreamTable[i]) << " Gateway " <<
std::get<1>(frerDestStreamTable[i]) << endl;
                            frerDestStreamTable.erase(frerDestStreamTable.begin() + i);
                            break;
                        }
                    }
                    delete resMsg;
                }
                else    // else CDT message for original stream
                {
                    // If CDT message source is current switch Index
                    if (resMsg->getSrcAddr() == getIndex())
                    {
                        // Find the stream in the local table and remove it then notify attached
source and finally adjust port gating ratio...
                        for (p = localStreamTable.begin(); p != localStreamTable.end(); p++)
                            if (std::get<0>(*p) == resMsg->getFlowID() && std::get<1>(*p) ==
getIndex())
                            break;
                        localStreamTable.erase(p);
                        numStreams--;
                        forwardSCP(resMsg);
                        if (reconfig && resMsg->getPriR() != priR[rtable[0][resMsg-
>getDestAddr()]])
```

```
                    {
                        priR[rtable[0][resMsg->getDestAddr()]] = resMsg->getPriR();
                        emit(priRchange[rtable[0][resMsg->getDestAddr()]],
priR[rtable[0][resMsg->getDestAddr()]]);
                    }
                }
                else        // else switch in intermediate one, so we just adjust port gating ratio
for the stream CDT given
                {
                    if (reconfig && resMsg->getPriR() != priR[rtable[0][resMsg-
>getDestAddr()]])
                    {
                        priR[rtable[0][resMsg->getDestAddr()]] = resMsg->getPriR();
                        emit(priRchange[rtable[0][resMsg->getDestAddr()]],
priR[rtable[0][resMsg->getDestAddr()]]);
                    }
                    delete resMsg;
                }
            }
        }
        else            // Stream/frame not approved by CNC... notify source node
            forwardSCP(resMsg);
    }
  }
}

void sw::handleMessage(cMessage *msg)
{
  if(!msg->isSelfMessage())       // new packet received will be queued (if not corrupted)
in appropriate port and queue (ST in queue 0, and BE in queue 1)
  {
      std::vector<std::tuple<int,int,u_int>>::iterator p;            // a pointer to the local
stream table

      /* If CDT packet arrived from attached source, then we check flowID and compare
it to our map table.
       * IF the flowID does not exists, then we sent the packet to the CNC for
configuration on TAS delay analysis,
       * else if data packet, then we queue and forward it as usual if the flow ID is in our
stream table */
      if (strcmp(msg->getArrivalGate()->getName(), "inout_sw_sc$i") == 0)
      {
          ScMessage *resMsg = check_and_cast<ScMessage *>(msg);          // cast
received new message to source message
```

```
            resMsg->setSWindex(getIndex());                    // set the SW address for
ID
            resMsg->setSCPindex(resMsg->getArrivalGate()->getIndex());     // set the
source SC node gate address
            resMsg->setSrcAddr(myAddress);                    // Set source
router/address
            if (resMsg->getDestAddr() == -1)
            {
                resMsg->setDestAddr((getIndex() + resMsg->getKind()) % numSW);    // set
destination address based on hopCount..
            }
        }
        else if (strcmp(msg->getArrivalGate()->getName(), "inout_sw_scp$i") == 0)
        {
            ScMessage *resMsg = check_and_cast<ScMessage *>(msg);          // cast
received new message to source message
            resMsg->setSWindex(getIndex());                    // set the SW address for
ID
            resMsg->setSCPindex(resMsg->getArrivalGate()->getIndex());     // Set source
port/gate ID so that we can return control messages if needed to right port...
            resMsg->setSrcAddr(myAddress);                    // Set source
router/address, note that getIndex() and myAddress are the same value...
            if (resMsg->getDestAddr() == -1)
            {
                resMsg->setDestAddr((getIndex() + resMsg->getKind()) % numSW);    // set
destination address based on hopCount..
            }
            if (strcmp(resMsg->getMsgType(), "st_data") == 0)          // Data Traffic
            {
                // Find Flow ID... If found, then forwards as usual (in this case, its TAS
operation), Note that the local stream table is for every port, i.e., all streams are saved in
<FlowID,SwitchID> pairs...
                for (p = localStreamTable.begin(); p != localStreamTable.end(); p++)
                    if (std::get<0>(*p) == resMsg->getFlowID() && std::get<1>(*p) ==
getIndex())
                        break;
                if (p == localStreamTable.end())          // Not found... illegal data packet...
                {
                    delete resMsg;
                    return;
                }
                // Else continue forwarding process...
            }
            else    // Control Traffic
            {
```

```
            forwardCNC(resMsg);
            return;
         }
      }
      else if (strcmp(msg->getArrivalGate()->getName(), "inout_sw_cnc$i") == 0)
// IF packet came from attached CNC
      {
         handleCNC(msg);
         return;
      }
      else if (strcmp(msg->getArrivalGate()->getName(), "ch_net") == 0)          // IF
msg came from net Change module
      {
         NetChangeMsg *resMsg = check_and_cast<NetChangeMsg *>(msg);       // cast
received new message to network change message
         //          updateRouting();

         for (int i = 0; i < switchChannelQueue.size(); i++)     // for each port, do...
         {
            if (!gate("sw_sw$o", i)->isConnected())
            {
               switchChannelQueue[i].numOfDisconnections++;
               switchChannelQueue[i].cnc_num_of_disconnections++;
               switchRecordObject tmp;
               tmp.module = this;
               tmp.swIndex = getIndex();
               tmp.port = i;
               tmp.value = switchChannelQueue[i].cnc_num_of_disconnections;
               if ((int)par("runType") == 3)
                  emit(stNumDisconnectSignalId, &tmp);
            }
         }
         delete resMsg;
         return;
      }

      /********** Usual TAS SW Mechanism starting HERE for DATA PACKETS
**********/
      ScMessage *resMsg = check_and_cast<ScMessage *>(msg);       // cast received
new message to source packet message
      resMsg->setDelay(simTime()-resMsg->getTimestamp());        // set running delay
for received packet based on time stamp

      if (strcmp(resMsg->getArrivalGate()->getName(), "sw_sw$i") == 0)       // only
take into account SW to SW links since these are of interest to us in Fault Tolerance
```

```
        {
          if (resMsg->getPriority() == ST)
          {
            // This is for traffic that is queued in the Future Event Queue that had their
sending channel disconnected while in transit... so need to remove when it arrives
            if (!resMsg->getArrivalGate()->isConnected())
            {
              delete msg;
              return;
            }
            switchChannelQueue[resMsg->getArrivalGate()-
>getIndex()].received_total_st_packet++;
            switchChannelQueue[resMsg->getArrivalGate()-
>getIndex()].cnc_received_total_st_packet++;
            if (resMsg->hasBitError())
            {
              switchChannelQueue[resMsg->getArrivalGate()-
>getIndex()].received_st_packet_lost++;
              switchChannelQueue[resMsg->getArrivalGate()-
>getIndex()].cnc_received_st_packet_lost++;
              if (resMsg->getArrivalGate()->getPreviousGate()->isConnected() &&
check_and_cast<channel_c *>(resMsg->getArrivalGate()->getPreviousGate()-
>getChannel())->getDuplicated() == false)
                switchChannelQueue[resMsg->getArrivalGate()-
>getIndex()].packetLoss[ST]++;
              EV << "resMsg received at gate port/index " << resMsg->getArrivalGate()-
>getIndex() << " from stream ID " << resMsg->getFlowID() << " and gateway " <<
resMsg->getSWindex() << " has error... discarding" << endl;

              //calculate per for the receiving port
              switchChannelQueue[resMsg->getArrivalGate()->getIndex()].received_per
= (double)switchChannelQueue[resMsg->getArrivalGate()-
>getIndex()].received_st_packet_lost / (double)switchChannelQueue[resMsg-
>getArrivalGate()->getIndex()].received_total_st_packet;
              switchChannelQueue[resMsg->getArrivalGate()-
>getIndex()].cnc_received_per = (double)switchChannelQueue[resMsg-
>getArrivalGate()->getIndex()].cnc_received_st_packet_lost /
(double)switchChannelQueue[resMsg->getArrivalGate()-
>getIndex()].cnc_received_total_st_packet;
              EV << "PER for sending switch " << resMsg->getArrivalGate()-
>getPreviousGate()->getOwnerModule()->getIndex() << " and sending port " <<
resMsg->getArrivalGate()->getPreviousGate()->getIndex() << " is " <<
switchChannelQueue[resMsg->getArrivalGate()->getIndex()].received_per << endl;
```

```
                // Need to check if the per is close to some threshold which then prompts
switch to contact CNC for any recommended actions
                if ((int)par("runType") == 3 && switchChannelQueue[resMsg-
>getArrivalGate()->getIndex()].cnc_received_per > perThresh && waiting_for_reply
== false)
                {
                  waiting_for_reply = true;
                  PerMessage *alert = new PerMessage("perMessage");
                  alert->setPer(switchChannelQueue[resMsg->getArrivalGate()-
>getIndex()].cnc_received_per);
                  alert->setCnc_received_total_st_packet(switchChannelQueue[resMsg-
>getArrivalGate()->getIndex()].cnc_received_total_st_packet);
                  alert->setCnc_received_st_packet_lost(switchChannelQueue[resMsg-
>getArrivalGate()->getIndex()].cnc_received_st_packet_lost);
                  alert->setGiven_sw(resMsg->getArrivalGate()->getOwnerModule()-
>getIndex());
                  alert->setGiven_port(resMsg->getArrivalGate()->getIndex());
                  forwardCNC(alert);
                }
                delete msg;
                return;
              }
            }
            else if (resMsg->getPriority() == BE && resMsg->hasBitError())
            {
              switchChannelQueue[resMsg->getArrivalGate()-
>getIndex()].packetLoss[BE]++;
              delete msg;
              return;
            }
          }

          // check FRER replication and elimination table if stream is part of it and
increment/check according to expected sequence number... packet with -1 sequence
indicate that it is not part of FRER stream
          std::vector<std::tuple<int,int,u_int>>::iterator q;
          if (resMsg->getPriority() == ST && resMsg->getSeq() != -1)     // -1 indicates a
non-replicated stream (could be original or duplicated packets of the same stream)
          {
            for (q = frerDestStreamTable.begin(); q != frerDestStreamTable.end(); q++)
            {
              if (std::get<0>(*q) == resMsg->getFlowID() && std::get<1>(*q) == resMsg-
>getSWindex())
              {
                EV << "Checking Duplicate Packet with Flow ID " << resMsg-
>getFlowID() << " and gateway " << resMsg->getSWindex() << " with Destination
Address " << resMsg->getDestAddr() << endl;
                EV << "Expected sequence number " << std::get<2>(*q) << "; Received
packet sequence number " << resMsg->getSeq() << endl;
                if (resMsg->getSeq() != std::get<2>(*q))
                {
                   EV << "Removing Duplicate Packet with Flow ID " << resMsg-
>getFlowID() << " and gateway " << resMsg->getSWindex() << " with Destination
Address " << resMsg->getDestAddr() << endl;
                   delete resMsg;
                   return;
                }
                else
                {
                   std::get<2>(*q) += 1;       // incrementing the next sequence number packet
for the replicated stream
                }
                break;
             }
          }
          // If the received packet is a duplicated packet that is not part of the FRER
destination Table.. then we need to remove it and not send it to sink
          //           if (getIndex() == resMsg->getDestAddr() && q ==
frerDestStreamTable.end() && resMsg->getDuplicated())
          //           {
          //              delete resMsg;
          //              return;
          //           }
       }
       for (q = frerSrcStreamTable.begin(); q != frerSrcStreamTable.end() && resMsg-
>getPriority() == ST; q++)
       {
          if (std::get<0>(*q) == resMsg->getFlowID() && std::get<1>(*q) == resMsg-
>getSWindex())
          {
             std::get<2>(*q) += 1;       // incrementing the next sequence number packet for
the replicated stream
             EV << "Setting Sequence " << std::get<2>(*q) << " to Packet with Flow ID "
<< resMsg->getFlowID() << " and gateway " << resMsg->getSWindex() << " with
Destination Address " << resMsg->getDestAddr() << endl;
             resMsg->setSeq(std::get<2>(*q));
             break;
          }
       }
```

```
        // use routing table to check destination if we have not arrived yet..
        if (getIndex() == resMsg->getDestAddr())     // queue the newly received packet to
the switch to sink port
        {
            // according data priority insert queue
            if(resMsg->getPriority() == ST)
            {
                if(STTotalLenQ2 - resMsg->getByteLength() >= 0)
                {
                    sinkChannelQueue[ST]->insert(resMsg);
                    STTotalLenQ2 -= resMsg->getByteLength();
                }
                else
                {
                    lossPktST++;
                    delete resMsg;
                    return;
                }
            }
            else
            {
                if(BETotalLenQ2 - resMsg->getByteLength() >= 0)
                {
                    sinkChannelQueue[BE]->insert(resMsg);
                    BETotalLenQ2 -= resMsg->getByteLength();
                }
                else
                {
                    lossPktBE++;
                    delete resMsg;
                    return;
                }
            }
            if (!SINKmsg->isScheduled())
                scheduleAt(simTime(), SINKmsg);
        }
        else     // queue the newly received packet to the switch to switch port(s)
        {
            // If the corresponding gate is set to duplicate, we get a copy and queue in the
other ports as needed.. other port being rtable[rtable.size()-1] routes
            if (resMsg->getPriority() == ST && switchChannelQueue[rtable[0][resMsg-
>getDestAddr()]].duplicate)
            {
                for (p = frerSrcStreamTable.begin(); p != frerSrcStreamTable.end(); p++)
                if (std::get<0>(*p) == resMsg->getFlowID() && std::get<1>(*p) ==
resMsg->getSWindex())
                    break;
            if (p != frerSrcStreamTable.end())      // if the stream is found in the Source
FRER table, then we need to replicate and queue in the appropriate port
            {
                ScMessage *dupMsg = resMsg->dup();      // duplicate the packet...
                if (switchChannelQueue[rtable[rtable.size()-1][dupMsg-
>getDestAddr()]].sizeQ[dupMsg->getPriority()] - dupMsg->getByteLength() >= 0)     //
check if space is available
                {
                    EV << "Duplicating packet with Flow ID " << dupMsg->getFlowID() <<
" and Switch ID " << dupMsg->getSWindex() << " from Port " << rtable[0][dupMsg-
>getDestAddr()] << " to " << rtable[rtable.size()-1][dupMsg->getDestAddr()] << endl;
                    dupMsg->setDuplicated(true);
                    dupMsg->setSeq(resMsg->getSeq());
                    dupMsg->setName(std::string("SCP-"+std::to_string(dupMsg-
>getSWindex()) + "-ST_DataTraffic_Duplicated_FlowID-" + std::to_string(dupMsg-
>getFlowID())).c_str());
                    // Queuing using new routing table for duplicated streams
                    switchChannelQueue[rtable[rtable.size()-1][dupMsg-
>getDestAddr()]].queue[dupMsg->getPriority()]->insert(dupMsg);
                    switchChannelQueue[rtable[rtable.size()-1][dupMsg-
>getDestAddr()]].sizeQ[dupMsg->getPriority()] -= dupMsg->getByteLength();

                    if (!switchChannelQueue[rtable[rtable.size()-1][dupMsg-
>getDestAddr()]].timer->isScheduled())
                        scheduleAt(simTime(), switchChannelQueue[rtable[rtable.size()-
1][dupMsg->getDestAddr()]].timer);
                    else
                    {
                        if (switchChannelQueue[rtable[rtable.size()-1][dupMsg-
>getDestAddr()]].timer->getArrivalTime() == priTopTime ||
switchChannelQueue[rtable[rtable.size()-1][dupMsg->getDestAddr()]].timer-
>getArrivalTime() == topTime)
                        {
                            cancelEvent(switchChannelQueue[rtable[rtable.size()-1][dupMsg-
>getDestAddr()]].timer);
                            scheduleAt(simTime(), switchChannelQueue[rtable[rtable.size()-
1][dupMsg->getDestAddr()]].timer);
                        }
                    }
                }
                else
                    delete dupMsg;
            }
            else          // Stream not found in FRER Source Table for the Duplicated Port
            {
               EV << "Stream ID " << resMsg->getFlowID() << " and Gateway " <<
resMsg->getSWindex() << " not found in FRER Source Table for Duplicated Port " <<
rtable[0][resMsg->getDestAddr()] << endl;
            }
         }
         // If the received packet is a duplicated packet, then we enqueue in to a different
queue based on the new routing path given by CNC from duplication message
         if (resMsg->getPriority() == ST && resMsg->getDuplicated())
         {
            EV << "Received duplicated packet with Flow ID " << resMsg->getFlowID()
<< " and Switch ID " << resMsg->getSWindex() << endl;
            if (switchChannelQueue[rtable[rtable.size()-1][resMsg-
>getDestAddr()]].sizeQ[resMsg->getPriority()] - resMsg->getByteLength() >= 0)
            {
               switchChannelQueue[rtable[rtable.size()-1][resMsg-
>getDestAddr()]].queue[resMsg->getPriority()]->insert(resMsg);
               switchChannelQueue[rtable[rtable.size()-1][resMsg-
>getDestAddr()]].sizeQ[resMsg->getPriority()] -= resMsg->getByteLength();
            }
            else
            {
               switchChannelQueue[rtable[rtable.size()-1][resMsg-
>getDestAddr()]].packetLoss[resMsg->getPriority()]++;
               cnc_stPktLost++;
               switchRecordObject tmp;
               tmp.module = this;
               tmp.swIndex = getIndex();
               tmp.port = rtable[rtable.size()-1][resMsg->getDestAddr()];
               tmp.value = cnc_stPktLost;
               if ((int)par("runType") == 3)
                  emit(stPktLossSignalId, &tmp);
               delete resMsg;
               return;
            }
         }
         else
         {
            //according data priority and route, we insert into queue... this is used by
shortest path routing.. i.e., rtable[0]
            if (switchChannelQueue[rtable[0][resMsg->getDestAddr()]].sizeQ[resMsg-
>getPriority()] - resMsg->getByteLength() >= 0)
            {
                switchChannelQueue[rtable[0][resMsg->getDestAddr()]].queue[resMsg-
>getPriority()]->insert(resMsg);
                switchChannelQueue[rtable[0][resMsg->getDestAddr()]].sizeQ[resMsg-
>getPriority()] -= resMsg->getByteLength();
            }
            else
            {
                switchChannelQueue[rtable[0][resMsg-
>getDestAddr()]].packetLoss[resMsg->getPriority()]++;
                if (resMsg->getPriority() == ST)
                {
                    cnc_stPktLost++;
                    switchRecordObject tmp;
                    tmp.module = this;
                    tmp.swIndex = getIndex();
                    tmp.port = rtable[0][resMsg->getDestAddr()];
                    tmp.value = cnc_stPktLost;
                    //                  if ((bool)par("frerMLmodel") == true)
                    if ((int)par("runType") == 3)
                        emit(stPktLossSignalId, &tmp);
                }
                else if (resMsg->getPriority() == BE)
                {
                    cnc_bePktLost++;
                    switchRecordObject tmp;
                    tmp.module = this;
                    tmp.swIndex = getIndex();
                    tmp.port = rtable[0][resMsg->getDestAddr()];
                    tmp.value = cnc_bePktLost;
                    //                  if ((bool)par("frerMLmodel") == true)
                    if ((int)par("runType") == 3)
                        emit(bePktLossSignalId, &tmp);
                }
                delete resMsg;
                return;
            }

            if (!switchChannelQueue[rtable[0][resMsg->getDestAddr()]].timer-
>isScheduled())
                scheduleAt(simTime(), switchChannelQueue[rtable[0][resMsg-
>getDestAddr()]].timer);
            else
            {
                // If new packet arrives and timer set to end of ST or BE slot, then cancel and
schedule new packet
```

```
                    if (switchChannelQueue[rtable[0][resMsg->getDestAddr()]].timer-
>getArrivalTime() == priTopTime || switchChannelQueue[rtable[0][resMsg-
>getDestAddr()]].timer->getArrivalTime() == topTime)
                    {
                        cancelEvent(switchChannelQueue[rtable[0][resMsg-
>getDestAddr()]].timer);
                        scheduleAt(simTime(), switchChannelQueue[rtable[0][resMsg-
>getDestAddr()]].timer);
                    }
                }
            }
        }
    }
    else
    {
        if (strcmp(msg->getName(), "cncTimer") == 0)        // CNC timer in SW module
        {
            if (gate("inout_sw_cnc$o")->getTransmissionChannel()-
>getTransmissionFinishTime() <= simTime())
            {
                if (!cncQ->isEmpty())
                {
                    send(cncQ->pop(), "inout_sw_cnc$o");
                    if (!cncQ->isEmpty())
                        scheduleAt(gate("inout_sw_cnc$o")->getTransmissionChannel()-
>getTransmissionFinishTime(), cncTimer);
                }
            }
            else
                scheduleAt(gate("inout_sw_cnc$o")->getTransmissionChannel()-
>getTransmissionFinishTime(), cncTimer);
        }
        else if (strcmp(msg->getName(), "scpTimer") == 0)   // one of the SCP timers in
SW module
        {
            if (gate("inout_sw_scp$o", msg->getKind())->getTransmissionChannel()-
>getTransmissionFinishTime() <= simTime())
            {
                if (!scpQ[msg->getKind()]->isEmpty())
                {
                    send(scpQ[msg->getKind()]->pop(), "inout_sw_scp$o", msg->getKind());
                    if (!scpQ[msg->getKind()]->isEmpty())
                        scheduleAt(gate("inout_sw_scp$o", msg->getKind())-
>getTransmissionChannel()->getTransmissionFinishTime(), scpTimer[msg-
>getKind()]);
```

```
            }
        }
        else
            scheduleAt(gate("inout_sw_scp$o", msg->getKind())-
>getTransmissionChannel()->getTransmissionFinishTime(), scpTimer[msg-
>getKind()]);
    }
    else if (strstr(msg->getName(),"swTimer") != nullptr)
    {
        timeCon = simTime() / maxWindowTime;          // Which CT you are on based
on clock
        topTime = ceil(timeCon) * maxWindowTime;       // Top Time for CT given
        botTime = floor(timeCon) * maxWindowTime;      // Bottom Time for CT given
        priTopTime = botTime + priR[msg->getKind()] * maxWindowTime;    // Fraction
ratio for ST traffic slot from CT with respect to port number (msg->getKind() indicates
port number)

        // If gate port is not connected (since it is possible to have link failures)
        if (!gate("sw_sw$o", msg->getKind())->isConnected())
        {
            switchRecordObject tmp;
            EV << "Port " << msg->getKind() << " Not connected... and duplicate flag is "
<< switchChannelQueue[msg->getKind()].duplicate << endl;
            // Drop ST packets in queue since port is disconnected and count lost packets
only if duplication is false for disconnected gate
            if (!switchChannelQueue[msg->getKind()].duplicate)
            {
                switchChannelQueue[msg->getKind()].packetLoss[ST] +=
switchChannelQueue[msg->getKind()].queue[ST]->getLength();
                cnc_stPktLost += switchChannelQueue[msg->getKind()].queue[ST]-
>getLength();
                tmp.module = this;
                tmp.swIndex = getIndex();
                tmp.port = msg->getKind();
                tmp.value = cnc_stPktLost;
                //                      if ((bool)par("frerMLmodel") == true)
                if ((int)par("runType") == 3)
                    emit(stPktLossSignalId, &tmp);
                switchChannelQueue[msg-
>getKind()].numOfstPacketsLostDueToLinkFailure += switchChannelQueue[msg-
>getKind()].queue[ST]->getLength();
                switchChannelQueue[msg-
>getKind()].cnc_st_pkt_loss_due_to_disconnection += switchChannelQueue[msg-
>getKind()].queue[ST]->getLength();
                tmp.module = this;
                tmp.swIndex = getIndex();
                tmp.port = msg->getKind();
                tmp.value = switchChannelQueue[msg-
>getKind()].cnc_st_pkt_loss_due_to_disconnection;
                //                      if ((bool)par("frerMLmodel") == true)
                if ((int)par("runType") == 3)
                   emit(stLostDueToLinkSignalId, &tmp);
             }
             switchChannelQueue[msg->getKind()].queue[ST]->clear();
             switchChannelQueue[msg->getKind()].sizeQ[ST] = Qsize;
             switchChannelQueue[msg->getKind()].packetLoss[BE] +=
switchChannelQueue[msg->getKind()].queue[BE]->getLength();
             cnc_bePktLost += switchChannelQueue[msg->getKind()].queue[BE]-
>getLength();
             tmp.module = this;
             tmp.swIndex = getIndex();
             tmp.port = msg->getKind();
             tmp.value = cnc_bePktLost;
             //                      if ((bool)par("frerMLmodel") == true)
             if ((int)par("runType") == 3)
                emit(bePktLossSignalId, &tmp);
             switchChannelQueue[msg->getKind()].queue[BE]->clear();
             switchChannelQueue[msg->getKind()].sizeQ[BE] = Qsize;
             return;
          }

          if (gate("sw_sw$o", msg->getKind())->getTransmissionChannel()-
>getTransmissionFinishTime() <= simTime())          // check if switch-to-switch
channel is idle
          {
             if (simTime() >= botTime && simTime() < priTopTime)      // ST slot
             {
                if (!switchChannelQueue[msg->getKind()].queue[ST]->isEmpty())   // Not
empty Q
                {
                   if (simTime() + (switchChannelQueue[msg->getKind()].queue[ST]-
>front()->getBitLength()/capacity) <= priTopTime)       // Guard Band
                   {
                      switchChannelQueue[msg->getKind()].sizeQ[ST] +=
switchChannelQueue[msg->getKind()].queue[ST]->front()->getByteLength();
                      send(switchChannelQueue[msg->getKind()].queue[ST]->pop(),
"sw_sw$o", msg->getKind());
                      switchChannelQueue[msg->getKind()].stPacketTput++;
                      switchChannelQueue[msg->getKind()].cnc_st_pkt_tput++;
                      switchRecordObject tmp;
```

```
                    tmp.module = this;
                    tmp.swIndex = getIndex();
                    tmp.port = msg->getKind();
                    tmp.value = switchChannelQueue[msg->getKind()].cnc_st_pkt_tput;
                    //                  if ((bool)par("frerMLmodel") == true)
                    if ((int)par("runType") == 3)
                        emit(stTputSignalId, &tmp);
                    if (!switchChannelQueue[msg->getKind()].queue[ST]->isEmpty()
|| !switchChannelQueue[msg->getKind()].queue[BE]->isEmpty())
                        scheduleAt(gate("sw_sw$o", msg->getKind())-
>getTransmissionChannel()->getTransmissionFinishTime(),
switchChannelQueue[msg->getKind()].timer);
                    else
                        scheduleAt(priTopTime, switchChannelQueue[msg-
>getKind()].timer);
                }
                else
                    if (!switchChannelQueue[msg->getKind()].queue[ST]->isEmpty()
|| !switchChannelQueue[msg->getKind()].queue[BE]->isEmpty())
                        scheduleAt(priTopTime, switchChannelQueue[msg-
>getKind()].timer);
            }
            else
                if (!switchChannelQueue[msg->getKind()].queue[ST]->isEmpty()
|| !switchChannelQueue[msg->getKind()].queue[BE]->isEmpty())
                    scheduleAt(priTopTime,switchChannelQueue[msg->getKind()].timer);
        }
        else if (simTime() >= priTopTime && simTime() < topTime) // BE slot
        {
            if(!switchChannelQueue[msg->getKind()].queue[BE]->isEmpty())   // Not
empty Q
            {
                if(simTime() + (switchChannelQueue[msg->getKind()].queue[BE]-
>front()->getBitLength()/capacity) <= topTime)   // Guard Band
                {
                    switchChannelQueue[msg->getKind()].sizeQ[BE] +=
switchChannelQueue[msg->getKind()].queue[BE]->front()->getByteLength();
                    send(switchChannelQueue[msg->getKind()].queue[BE]->pop(),
"sw_sw$o", msg->getKind());
                    if(!switchChannelQueue[msg->getKind()].queue[BE]->isEmpty()
|| !switchChannelQueue[msg->getKind()].queue[ST]->isEmpty())
                        scheduleAt(gate("sw_sw$o", msg->getKind())-
>getTransmissionChannel()->getTransmissionFinishTime(),
switchChannelQueue[msg->getKind()].timer);
                    else
```

```
                        scheduleAt(topTime, switchChannelQueue[msg->getKind()].timer);
                    }
                    else
                        if (!switchChannelQueue[msg->getKind()].queue[ST]->isEmpty()
|| !switchChannelQueue[msg->getKind()].queue[BE]->isEmpty())
                        scheduleAt(topTime, switchChannelQueue[msg->getKind()].timer);
                }
                else
                    if (!switchChannelQueue[msg->getKind()].queue[ST]->isEmpty()
|| !switchChannelQueue[msg->getKind()].queue[BE]->isEmpty())
                        scheduleAt(topTime,switchChannelQueue[msg->getKind()].timer);
            }
        }
        else
            scheduleAt(gate("sw_sw$o", msg->getKind())->getTransmissionChannel()-
>getTransmissionFinishTime(), switchChannelQueue[msg->getKind()].timer);
    }
    else if (msg == SINKmsg)
    {
        if(gate("inout_sw_sink$o")->getTransmissionChannel()-
>getTransmissionFinishTime() <= simTime())      // check if switch-to-sink channel is
idle
        {
            if (!sinkChannelQueue[ST]->isEmpty())
            {
                STTotalLenQ2 += sinkChannelQueue[ST]->front()->getByteLength();
                send(sinkChannelQueue[ST]->pop(),"inout_sw_sink$o");
                if(!sinkChannelQueue[ST]->isEmpty() || !sinkChannelQueue[BE]-
>isEmpty())
                    scheduleAt(gate("inout_sw_sink$o")->getTransmissionChannel()-
>getTransmissionFinishTime(), SINKmsg);
            }
            else if (!sinkChannelQueue[BE]->isEmpty())
            {
                BETotalLenQ2 += sinkChannelQueue[BE]->front()->getByteLength();
                send(sinkChannelQueue[BE]->pop(),"inout_sw_sink$o");
                if(!sinkChannelQueue[BE]->isEmpty() || !sinkChannelQueue[ST]-
>isEmpty())
                    scheduleAt(gate("inout_sw_sink$o")->getTransmissionChannel()-
>getTransmissionFinishTime(), SINKmsg);
            }
        }
        else
            scheduleAt(gate("inout_sw_sink$o")->getTransmissionChannel()-
>getTransmissionFinishTime(), SINKmsg);
```

```
        }
    }
    return;
}

void sw::finish()
{
    double packetLossST = 0, packetLossBE = 0;
    outfile << "SW " << getIndex() <<  ": Total ports = " << switchChannelQueue.size()
<< endl;
    for (int i = 0; i < switchChannelQueue.size(); i++)
    {
        packetLossST += switchChannelQueue[i].packetLoss[ST];
        packetLossBE += switchChannelQueue[i].packetLoss[BE];
        outfile << "SW " << getIndex() <<  " Port " << i << ": Total ST Packets Sent = " <<
switchChannelQueue[i].stPacketTput << endl;
        outfile << "SW " << getIndex() <<  " Port " << i << ": Total ST Packets lost due to
link failure = " << switchChannelQueue[i].numOfstPacketsLostDueToLinkFailure <<
endl;
        outfile << "SW " << getIndex() <<  " Port " << i << ": Total Disconnection = " <<
switchChannelQueue[i].numOfDisconnections << endl;
        outfile << "SW " << getIndex() <<  " Port " << i << ": Received ST Packets
received from SW " << this->gate("sw_sw$i", i)->getPreviousGate()-
>getOwnerModule()->getIndex() << " and port " << this->gate("sw_sw$i", i)-
>getPreviousGate()->getIndex() << " = " <<
switchChannelQueue[i].received_total_st_packet << endl;
        outfile << "SW " << getIndex() <<  " Port " << i << ": Discarded ST Packets
received from SW " << this->gate("sw_sw$i", i)->getPreviousGate()-
>getOwnerModule()->getIndex() << " and port " << this->gate("sw_sw$i", i)-
>getPreviousGate()->getIndex() << " = " <<
switchChannelQueue[i].received_st_packet_lost << endl;
        outfile << "SW " << getIndex() <<  " Port " << i << ": PER for link from SW " <<
this->gate("sw_sw$i", i)->getPreviousGate()->getOwnerModule()->getIndex() << "
and port " << this->gate("sw_sw$i", i)->getPreviousGate()->getIndex() << " = " <<
switchChannelQueue[i].received_per << endl;
    }
    packetLossST += lossPktST;
    packetLossBE += lossPktBE;
    outfile << "SW " << getIndex() <<  ": Total ST Packets Lost = " << packetLossST <<
endl;
    outfile << "SW " << getIndex() << ": Total BE Packets Lost = " << packetLossBE <<
endl;
    outfile.close();
    return;
}
```