Mälardalen University
School of Innovation, Design and Engineering
Västerås, Sweden

---

Thesis for the Degree of Master of Science in Computer Science with Specialization in Embedded Systems - 30.0 credits

# Integrating 5G Components into a TSN Discrete Event Simulation Framework

Alexander Magnusson
amn16006@student.mdh.se
David Pantzar
dpr16001@student.mdh.se

Examiner: Saad Mubeen
saad.mubeen@mdh.se
Mälardalen University, Västerås, Sweden

Supervisors: Mohammad Ashjaei & Zenepe Satka
mohammad.ashjaei@mdh.se zenepe.satka@mdh.se
Mälardalen University, Västerås, Sweden

June 8, 2021

## Abstract

*Time Sensitive Networking (TSN) has for many years been the staple of reliable communication over traditional switched Ethernet and, has been used to advance the industrial automation sector. However, TSN is not mobile, which is needed to fully enable Industry 4.0. The development of 5G and its promised Ultra Reliable and Low-Latency Communication (URLLC) combined with TSN would give both a mobile and reliable heterogeneous network. The 3GPP has suggested different designs for a 5G and TSN integration. This thesis investigates the different proposed integration designs. Besides the integration design, one of the most essential steps towards validity of the integration is to evaluate the TSN-5G networks based on simulation. Currently, this simulation environment is missing. The investigation in this thesis shows that the most exhaustive work had been done on the Logical TSN Bridge design for simulators, such as the ones based on OMNeT++. Capabilities of the simulator itself are also investigated, where aspects such as the lack of a 5G medium and clock synchronization are presented. In this thesis, we implement the 5G-TSN component that results in a translator which sets different 5G channel parameters depending on the Ethernet packet's priority and its corresponding value. To verify the functionality of the translator that is developed within the simulator, it is tested in a use case inspired by the vehicle industry, containing both TSN and 5G devices. Results from the use case indicate that the translation is performed correctly.*

# Acronyms

**(R)AN** (Radio) Access Network.

**3GPP** 3rd Generation Partnership Project.

**AF** Application Function.

**AMF** Access and Mobility Management Function.

**AN** Access Node.

**ARQ** automatic repeat request.

**AVB** Audio-Video Bridging.

**BD** Bridge Delay.

**BE** Best-Effort.

**CAN** Controller Area Network.

**CBS** Credit-Based Shaper.

**CN** Core Node.

**CNC** Centralized Network Configuration.

**CP** Control Plane.

**CRC** Cyclic Redundancy Check.

**CSMA/CD** Carrier sense multiple access with collision detection.

**CUC** Centralized User Configuration.

**DEI** Drop Eligible Indicator.

**DRB** Data Radio Bearer.

**DS-TT** Device Side TSN Translator.

**FQTSS** Forwarding and Queuing Enhancements for Time-Sensitive Streams.

**FRER** Frame Replication and Elimination for Reliability.

**GBR** Guaranteed Bit Rate.

**GFBR** Guaranteed Flow Bit Rate.

**gNB** new Generation Node B.

**gPTP** generalized Precision Time Protocol.

**HSR** High-availability Seamless Redundancy.

| | |
|---|---|
| **HSS** | Home Subscriber Service. |
| **IEEE** | Institute of Electrical and Electronics Engineers. |
| **IoT** | Internet of Things. |
| **IT** | information technology. |
| **LAN** | Local Area Networks. |
| **LLC** | Logical Link Control. |
| **LLDP** | Link Layer Discovery Protocol. |
| **MAC** | Medium Access Control. |
| **MAN** | Metropolitan Area Networks. |
| **MDBV** | Max data burst volume. |
| **MFBR** | Maximum Flow Bit Rate. |
| **MO** | Managed Objects. |
| **MSRP** | Multiple Stream Registration Protocol. |
| **NED** | Network Description. |
| **NEF** | Network Exposure Function. |
| **NeSTiNg** | Network Simulator for Time-Sensitive Networking. |
| **NIC** | Network Interface Card. |
| **NW-TT** | Network TSN Translator. |
| **OMNeT++** | Objective Modular Network Testbed in C++. |
| **OT** | Operational Technology. |
| **PCF** | Policy Control Function. |
| **PCP** | Priority Code Point. |
| **PDB** | Package Delay Budget. |
| **PDU** | Protocol Data Unit. |
| **PER** | Package Error Rate. |
| **PRP** | Parallel Redundancy Protocol. |
| **PSA** | PDU Session Anchor. |
| **PSFP** | Per-Stream Filtering and Policing. |
| **PTP** | Precision Time Protocol. |

**QFI** QoS Flow Identifier.

**QoS** Quality of Service.

**SMF** Session Management Function.

**SRP** Stream Reservation Protocol.

**TAS** Time-Aware Shaper.

**TCI** Tag Control Information.

**TDMA** Time Division Multiple Access.

**TEID** Tunnel Endpoint Identifier.

**TPID** Tag Protocol Identifier.

**TSN** Time Sensitive Networking.

**TT** TSN Translator.

**UDM** Unified Data Management.

**UE** User Equipment.

**UP** User Plane.

**UPF** User Plane Function.

**URLLC** Ultra Reliable and Low-Latency Communication.

**VID** VLAN Identifier.

**VLAN** Virtual Lan.

**WAN** Wide Area Networks.

# Table of Contents

## 1 Introduction

Real-time embedded systems are computing systems which must be capable of guaranteeing an output or response according to the systems functionality within a specifically bounded time [1]. These systems have to be dependable, as a failure might lead to catastrophic consequences depending on the criticality of the system. Real-time embedded systems exist in almost all industries; nuclear plant control, flight control, telecommunication, robotics and industrial automation for Industry 4.0, to mention a few. These systems are commonly distributed, where sensors and actuators can be separate systems requiring deterministic bounded real-time communication. In traditional industry application networks there is a need for a specifically bounded end-to-end latency. The end-to-end latency is the time it takes for an event to trigger a response. The latency should be within a few microseconds to a few milliseconds [2], one millisecond for Internet [3] and 100 microseconds for wireless networks. In the medical field, for example, telesurgery, a requirement is a set of near to real-time connectivity [4].

Timing constraints put a deadline on the data-transfer where a system should act upon an event within an amount of time after the event has been sensed [1]. Such requirements add to the communication system's complexity, which has to guarantee that the correct data arrives at the right time. There are several predictable network communication protocols, such as Controller Area Network (CAN) [5], EtherCAT [6], and PROFINET [7]. A network is predictable if it is possible to demonstrate at design time that all timing requirements will be satisfied during the execution of the system [8]. All three are low bandwidth protocols, which is an important aspect for today's network as they are transferring more data than ever, over larger distributed systems.

One well-known standard which can handle such bandwidth is the Institute of Electrical and Electronics Engineers (IEEE) Ethernet standard [9], supporting speeds from 1 GBit/s up to 40 GBit/s depending on the hardware used [10]. While the IEEE Ethernet standard can support the throughput, it can not provide predictability guarantees. Timing constraints of the communication scheme is not feasible as the latency can reach infinity due to queues or dropped packages because of overfull buffers [11].

A standard that has the properties of reliable data-transfer with high throughput is needed to meet the requirement of Industry 4.0 for deterministic bounded real-time communication. One of these standards is Time Sensitive Networking (TSN), developed by IEEE [12] 802.1 TSN Task Group. It works as an extension of the traditional Ethernet OSI-Layer 2 so that different standards can share the same infrastructure; creating an environment of no vendor dependencies. TSN utilizes a control node which computes the requirements of the network to provide guarantees of low packet loss, jitter and latency. The controller can provide this by using several features, such as Time-Triggered (TT), Audio-Video Bridging (AVB) as well as Best-Effort (BE), which makes it well suited for the type of real-time communication that requires low-latency with varying priority levels [13]. The challenge lies in configuring critical and non-critical data traffic so that neither real-time characteristics nor performance is impaired. As TSN is built as an extension of the IEEE Ethernet standard, adopting a TSN solution would lead to a lower hardware cost and a higher bandwidth in comparison with other vendor specific solutions [2].

While TSN is a strong candidate to become the standard communication for Industry 4.0, it is lacking in the domain aimed towards mobile systems [2][14]. However, with the integration of 5G, limitations which come from cable installations would be removed and a mobile industry could be possible [15][16]. 5G is designed to handle the communication for consumer devices, and Internet of Things (IoT) but it also shares many features with TSN such as Ultra Reliable and Low-Latency Communication (URLLC). Integrating 5G and TSN would provide a gateway for a more converged network structure, allowing for all types of data to be transferred over a single network. Within the 3GPP release 16 [17] many of the features needed to reduce latency and increase reliability in the 5G architecture has been added, providing the low latency and reliability required for a dependable integrated system. Combining 5G and TSN is seen as a holistic communication solution which would provide a wireless and, URLLC that has the flexibility and dependability needed for future networks within smart factories [2].

Integration of TSN and 5G could be done in different ways. One of these ways is to evaluate the ingetgration in a simulated environment. Simulation is important as it gives a means to see the functionality of a system before it is implemented in a hardware solution which in turn could lead to a reduction in cost of development. By using a simulation approach many different scenarios could be tested before taking the system into a live environment. A simulator for TSN exists, based on Objective Modular Network Testbed in C++ (OMNeT++), the Network Simulator for Time-Sensitive Networking (NeSTiNg) simulator [18], developed by the Distributed Systems group of IPVS, at the University of Stuttgart [19] is capable of modeling several of the TSN characteristics [20]. There are also a few 5G simulation tools in development for the OMNeT++ simulator such as simu5G [21], and Ginthör *et al.*'s simulator presented in [22]. However, as of writing this thesis, neither of these sources have released their network modules yet creating a gap in integration of TSN and 5G networks in a simulated environment.

As the integration of 5G components into a TSN network is a vital part for creating smart factories with mobile capabilities, the main focus of this thesis will be to investigate the main and essential components for integrating 5G and TSN. In order to advance the simulation tools, we design and develop integration components, which will be implemented in an existing simulation tool. Analyzing such a integrated TSN and 5G network would give a better understanding on how adding 5G functionalities to a smart factory could improve the overall performance and provide a more converged network architecture [15]. A close-to-real-world scenario is set up and tested with the 5G-TSN components integrated.

## 1.1 Motivation

Industry 4.0 would require converged network, able to provide guarantees with regards to deterministic bound data transference [16]. This would lead to a more transparent network, allowing parts of the Operational Technology (OT) and information technology (IT) sector of a smart factory to have a homogeneous layout. In today's industrial networks the OT domain is made up of 90% vendor locked wired technologies with limited throughput [23]. In integrating 5G into a TSN an architecture could be created which would lower the amount of specific vendor requirements as well as introduce a greater level of flexibility in the network.

With parts of the smart factory network being wireless one near-term benefit would be the significant reduction in cables used, which in turn would reduce production cost [16]. Especially with regards to factories with mobile platforms, robots or guided vehicles [24]. For example, fully automated mobile platforms, such as mining vehicles requires a lot of intensive and high-bandwidth sensors, LIDARs and video cameras. For handling such communication system the best possible solution is TSN. However, in some use cases a remote control for the vehicles is expected. In this case the best option to communicate with that vehicle is 5G because of the very low-latency communication, which leads to a combination of TSN and 5G networks.

A wireless solution could lead to more efficient factory layouts, granting more effective production as a whole. The components for simulation environment for 5G-TSN integration proposed in this thesis will focus on a translator between the two systems. The translator would act as an intermediary, taking the necessary properties from TSN and assign them to the 5G QoS as per 3GPP and vice versa [17].

## 1.2 Problem Formulation

The main focus of this thesis is to investigate the main and essential components for integrating 5G and TSN, and then implement and test 5G-TSN components in the NeSTiNg simulation tool. We have selected NeSTiNg which currently supports TSN features, as it is developed over a well-known simulation platform, i.e., OMNeT++, as well as because it already supports the TSN features. Firstly, the system architectures and properties required to integrate 5G and TSN has to be investigated. Approaches to a centralized or distributed, time-triggered or credit-based has to be analyzed. Secondly, a study of the current capabilities of the NeSTiNg simulation tool is required to ascertain how to best integrate the translator with respect to the established architecture and properties. Furthermore, the translator would require that the traffic flows, priorities, time synchronization and other properties from the TSN is integrated into the 5G QoS configuration, and vice versa without jeopardizing the properties of either of them. To do so, a scheduling algorithm has to be decided considering the capabilities of the NeSTiNg simulator. In light of the above-mentioned problems, the following research questions will be answered in thesis.

- **RQ1:** *What are the different approaches to integrate TSN and 5G in the research and standardization communities?*
- **RQ2:** *What are the current capabilities and limitations of the NeSTiNg simulation tool in regards to properties regarding 5G and TSN?*

And with the knowledge gained answering RQ1 and RQ2, a third RQ can be presented, which focuses on applying the knowledge gained into the simulation environment.

- **RQ3:** *How can 5G-TSN configuration be implemented to improve the usability of the NeSTiNg simulation tool?*

## 1.3 Expected Outcomes

The expected outcome for this thesis is a functioning 5G addition to the NeSTiNg simulator based on OMNeT++. By integrating TSN and 5G it is also expected that the overall flexibility of the simulated network will increase, however, that the performance in terms of latency will decrease in comparison with a fully wired approach. It is expected that the integration will provide a 5G-TSN component for use in the future development and research of the 5G-TSN integration. Note that the developed TSN-5G simulator based on OMNeT++ can be downloaded from our GitHub repository [1].

## 1.4 Limitations

Performance data will be gathered via the simulating software, and implementations done on the simulated environments in the NeSTiNg simulator based on OMNET++/INET-framework. Performance measurements will be limited to the outputs given by the NeSTiNg

[1]https://gitlab.com/DavidPantzar/5GTSNTranslator

simulator. Networks will be purely simulated, no external hardware will be used within this thesis. Furthermore, no global clock-synchronization or disruptive radio environment for the wireless signals will be established in the current iteration of the project

## 1.5 Thesis Outline

Section 2 outlines background knowledge that is relevant to know for this thesis. Section 3 presents the work related to the thesis in the area of 5G-TSN. Section 4 presents the method of the work in this thesis. Section 5 outlines 5G-TSN concept that are needed for the TSN-5G integration. Section 6 presents an investigation of available 5G-TSN approaches. Section 7 presents an investigation of the simulation environment. Section 8 defines the design of the implementation as well as the design of the usecase. Section 9 presents the results of the implementation with the usecase defined in Section 8. Section 10 outlines the discussion of the investigation into 5G-TSN System Architecture, properties of the simulation tool and 5G-TSN implementation. Section 11 defines the conclusions of the work. Section 12 outline suggestions for future additions that can be made to the work.

# 2 Background

This section will provide the necessary knowledge required to understand the contributions of the Master Thesis. The following sections will present an overview of Real Time Systems, Ethernet, Time Sensitive Networking, and OMNET++ with its respective subsections.

## 2.1 Real Time Systems

A real-time system is defined as a system that can respond to environmental events in a timely fashion [25][26]. The response can be things such as make calculations or actuating on the event. Real-time systems have constraints on them regarding the time they should act, leading to a common misconception that a real-time system has to be fast. While a real time system can be fast, it is not always so; it is the environment that determines the system's speed. For example, based on their task and environment, one system might have to sample the environment once every three days and another every 5 seconds. The two systems have different speeds but are both real-time systems; they are just designed after their respective environments. Buttazzo [1] defines *real* and *time* as follows: "*The word time means that the correctness of the system depends not only on the logical result of the computation but also on the time at which the results are produced...*" "*The word real indicates that the reaction of the systems to external events must occur during their evolution...*"

A real-time system can be either one system or as part of a more extensive distributed system [1]. Guaranteed response time is still the system's most defining characteristic regardless of how the system is designed. This response time can either be the established deadline, E.g., the finish time, or both start and finish time. A typical scenario for a real-time system is as follows. An environmental change or event occurs which triggers a sensor. Data from the sensor is processed and then actuated upon in some way. For example, if a temperature increase is detected and is above a certain threshold, the system tells the fan it has to start spinning faster to cool down the room. In this case, the deadline could be that the system has to cool down the room before whatever is in the room take damage by the increased heat.

A control system such as the temperature regulator can be designed as either periodic, aperiodic or a combination of both [26]:

- **Periodic:** The system performs its set task every pre-defined period of time. This makes the system predictable as it is known when it will interact with its environment. For example, the temperature sensor is polled every pre-defined period of time to check for any changes.
- **Aperiodic:** The system does not know when it will act upon its environment, the trigger could be an event or change in the environment. For example, the temperature sensor only sends data when it detects a certain threshold has been reached.

Real-time systems are deadline-driven; they have to perform their logical and functional correctness within a predefined time. Suppose the temperature regulating system does not act in time. In that case, the product it was meant to keep cool might be damaged, or if an airbag does not expand at just the right time after a collision, it might lead to severe damage or even the loss of life for the person behind the wheel. As there are many real-time systems, their deadlines have been defined as soft, firm, and hard depending on the consequences should they be missed [1].

- **Soft:** A late result might still be useful for the system, and the system does not suffer severe consequences for getting a late result. These systems are typically found in system-user interactions such as using a keyboard, saving data or displaying messages.
- **Firm:** If the deadline of a firm task is missed the results have no use, but no severe consequences occur. Examples of this are found in multimedia, such as when audio and video are unsynched, or in streaming and the video becomes pixelated due to video instances arriving in the wrong order.
- **Hard:** The results after a missed deadline in a hard system has no use, additionally a missed deadline has severe consequences in terms of either economical or in human lives. Examples of these include, acquisition of sensory data, detection and servoing systems.

When designing a real-time system it is the consequences and the penalty of said consequences that defines what the system requirements are [1]. With the requirements of a real-time system comes certain characteristics that the system should have; timeless, predictable, efficient, robust, tolerant and, maintainable. These characteristics are typically provided at the cost of one another, where for example, a predictable system might not be the most efficient.

- **Timeliness:** A system which can guarantee results which are not only correct but also arrives at the right time, regardless of priority level or deadlines of the tasks.
- **Predictable:** A system where all deadlines can be guaranteed, for safety critical systems the tasks have to be scheduled offline to provide the guarantees before run-time.
- **Efficiency:** Real-time systems are typically an embedded system with limited resources. Systems has to be designed to handle such constraints like limited computational power and energy.
- **Robustness:** Real-time systems have to be designed to be able to handle overloads. How should the system act if an outlier occurs outside of normal parameters. Unit-testing can help create a more robust system.
- **Tolerance:** A real-time system has to be able to handle certain levels of failures. Backup systems or secondary hardware could be implemented to increase the tolerance.
- **Maintainability:** A well designed real-time system is modular in nature so future development and handovers can be made. As the nature of a systems environment change it might be necessary, and cheaper, to perform a change to the system rather than try to change the environment as a whole.

## 2.2 Ethernet

Ethernet includes standard for communication in Local Area Networks (LAN), Metropolitan Area Networks (MAN) and Wide Area Networks (WAN). Ethernet includes specifications for both the physical and data link layer of the OSI model. Multiple options for the physical medium are available such as fiber optics, coaxial and twisted pair. The data link layer of ethernet are divided in two sublayers Logical Link Control (LLC) being the upper and

Medium Access Control (MAC) the lower. This because MAC handles the hardware side with each Network Interface Card (NIC) having a unique 48bit address often represented in hexadecimal. MAC handles the hardware responsible for interactions with the physical medium. Mac also handles multiplexing and flow control for the physical medium. In older version of ethernet Carrier sense multiple access with collision detection (CSMA/CD) was also included in the MAC layer, when Ethernet used a shared medium, but in the age of Full duplex and switched ethernet this is no longer needed. LLC controls the logical link between nodes and makes sure that multiple network protocols can be supported on the same multipoint network. It acts as a bridge between the physical, MAC layer and the Network Layer. Error correction such as automatic repeat request (ARQ) is also available in some version of the LLC layer but is not needed in the Ethernet version since bit error are rare in physical medium. In Switched Ethernet all devices are connected to a shared network device called Switch. A switch is a multiport layer 2 device with uses the MAC address to forward data on data link layer and is used to bridge the connection of multiple devices. This is done by keeping a MAC table of all MAC addresses connected to the switch and which port that MAC address is connected to.

## 2.3 Time Sensitive Networking

Time Sensitive Networking (TSN) is a collection of Institute of Electrical and Electronics Engineers (IEEE) 802.1 standards produced by the TSN Task Group focused on transfer of time sensitive transmissions over Ethernet networks. The initial focus area of standardization was Audio-Video Bridging (AVB) and was the original name of the Task Group before the rename and broadening in scope [27][28]. The AVB task group was renamed in 2012 following rising demand from industries for high reliability, high bandwidth traffic over Ethernet. In the original design of Ethernet all data frames were treated as equally important [27]. Traffic classes were not added until IEEE 802.1D-1998 which gave control in eight different classes of priority. Virtual Lan (VLAN) were added in IEEE 802.1Q-1998 allowing for multiple separated data streams on the physical network. This allowed for traffic with higher network requirements such as audio and video traffic to be given priority over other traffic classes. In 2002 IEEE-1588 Precision clock synchronization was standardized, this was later adapted and modified for AV implementations in 802.1AS-2011. With the Time synchronization from 802.1AS-2011 and a Stream Reservation Protocol (SRP) (802.1Qat) using VLAN, and a Credit-Based Shaper (CBS) (802.1Qav) used to enforce the contract created by the reservation protocol. Using this a solution for lossless guaranteed bandwidth for audio over Ethernet for Audio studios was produced using AVB standards [28].

TSN is an end to end solution for deterministic communication on Ethernet, TSN have a centralized management and guaranteed packet delivery with minimized jitter using scheduling mechanics for real-time traffic requiring determinism [27][28]. Both AVB and TSN are Layer 2 technologies meaning they work on the data link layer of the OSI model. A requirement for TSN is that all switches that are connected in the TSN are time synchronized using 802.1AS to be able to honor the timing constraints of the TSN flows. TSN supports multiple flows of data using what they call TSN flows where each flow can have a different timing constraint. Each flow spans from the end device to end device to be able to enforce the full timing constraint specified in the contract created for each TSN flow. TSN flows uses the Priority Code Point (PCP) that was added in 802.1Q-1998 VLAN allowing for eight traffic classes on each physical port on the TSN switches. Scheduling available in TSN is the earlier mentioned CBS from AVB standardized in 802.1Qav or the newer TSN implementation Time-Aware Shaper (TAS) based on Time Division Multiple

Access (TDMA) where critical traffic and non-critical traffic are assigned different time slots. TAS use PCP to guarantee traffic of specific priority exclusive access to the transfer medium during the assigned time slot of traffic of that priority is available. Each time slot represent which gate on the queues of the egress port of the switch should be open at that time. To be able to guarantee that the time schedule is followed and no frame exceeds their time slot into another time slot a guard band of the size of the next frame or maximum frame size is included at the end of each scheduling cycle. CBSs scheduling mechanic is based on a token bucket system where packet that are waiting in queue increase the tokens available, sending packets reduce the amounts of tokens. CBS is useful for minimizing delays and jitter in AVB networks but not suitable for the real time constraints of a TSN network.

TSN includes mechanics to guarantee packet deliveries from end device to end device, this is done using packet replication based on High-availability Seamless Redundancy (HSR) and Parallel Redundancy Protocol (PRP). Multiple copies of a packet are sent along different paths and later removed close or at the destination. Packet can be replicated at source or along the way to the destination depending on configuration and demand for packet redundancy.

The TSN standards also support preemption of non-critical frames by critical frames and was added in 802.1Qbu were this service is standardized [29]. In traditional Ethernet a frame that have started transmission have to complete the whole process. This can be a big problem in time critical networks. As critical frames would be made to wait for the transmission process to be finish before being allowed to start transmission this adding to the delay and jitter. The goal of 802.1Qbu was to minimize delay for time critical transmission while minimizing impact on non-time critical transmission. If an output port is preemptable is controlled on a per port basis. Preempted frames is resumed at the point of preemption when no more higher priority frames are available.

### 2.3.1 TSN Models

The 802.1Qcc [30] introduced improvements to the Stream Reservation Protocol which became the features for resource reservation within a TSN network. There are different types of TSN models, the fully distributed model, the centralized / distributed model, and the fully centralized model [31].

### 2.3.2 Fully Distributed Model

In the fully distributed model each network component shares its properties required to establish a TSN flow with all other components that are part of the flow in order to establish a TSN QoS. The TSN End Stations, i.e. Talkers and Listeners, sends their requirements to the TSN Bridges, which then propagates the requirements along the TSN stream. Each node in the fully distributed model handles their own resource management. A visual representation of the distributed model can be seen in Figure 2.1.

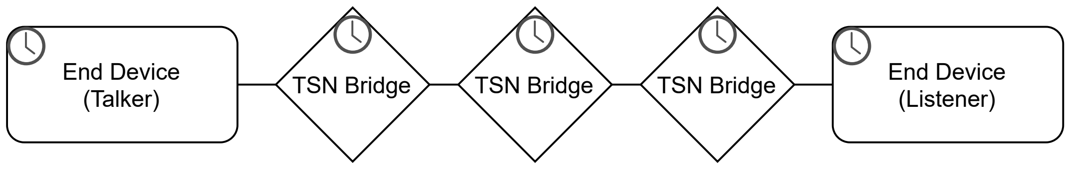


Figure 2.1: Fully Distributed TSN Model

### 2.3.3 Centralized/Distributed Model

In the Centralized / Distributed Model the TSN components still send their properties amongst each other, but also to the Centralized Network Configuration (CNC), which gets an overview of the entire network. The CNC facilitates more complex TSN features such as TAS and preemption of flows.

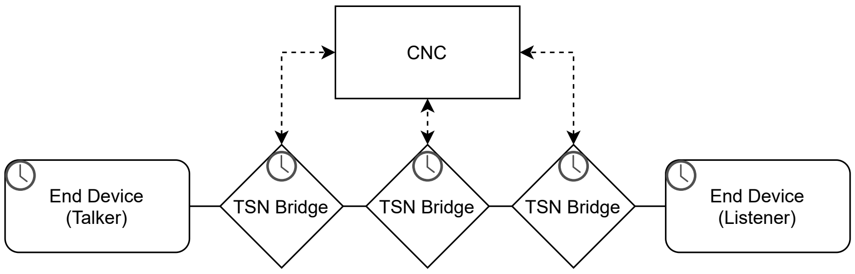


Figure 2.2: Centralized / Distributed TSN Model

### 2.3.4 Fully Centralized Model

The main components in a fully centralized TSN network are the Centralized User Configuration (CUC), Centralized Network Configuration (CNC), the TSN Bridge and the end devices [27]. The CNC gets the complete picture of the network by receiving network information such as capabilities and topology from the bridges, it also receives the requirements set by the End Stations from the CUC who acts as their intermediary. With this information the CNC can schedule the network, sending the TSN Configuration info to the bridges and CUC, which relays the information to the End Stations. A visual representation of the Centralized TSN Model can be seen in 2.3, the flow of information in the centralized TSN model is as follows:

1. The End Devices sends their QoS requirements to the CUC, these requirements are things such as data-rate, traffic class, priorities and worst-case latency.

2. The CUC relays the End Device information to the CNC, however it can in this step adapt them depending on the application of the network.

3. The TSN Bridges convey their capabilities to the CNC, these are things such as, bridge delays per port and class, propagation delays per port and priorities.

4. The CNC calculates a schedule for the network based on the QoS requirements from the End Devices as well as the capabilities of the TSN Bridges. It schedules with regards to start times of flows and the open and closing gate control timings within each of the TSN Bridge.

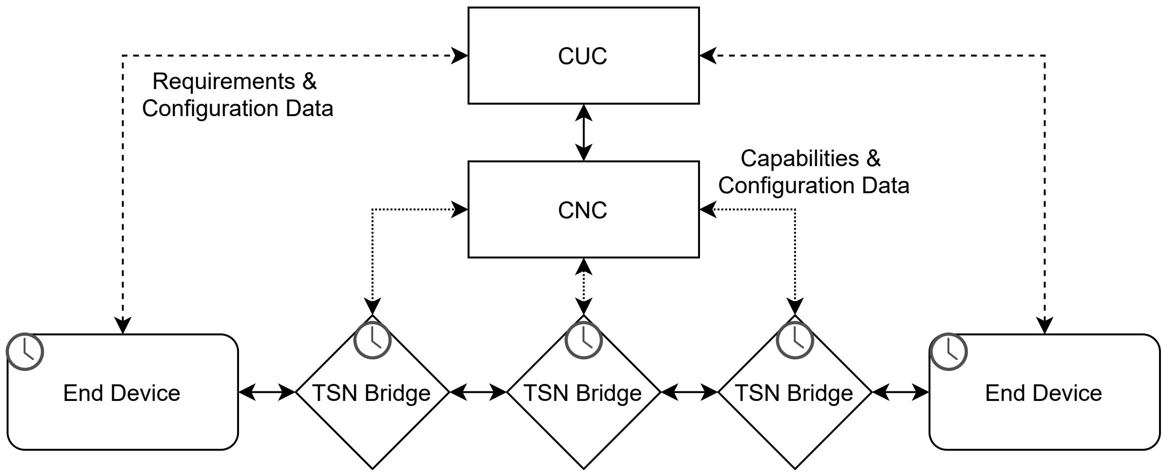

CUC
CNC
Requirements & Configuration Data
Capabilities & Configuration Data
End Device
TSN Bridge
TSN Bridge
TSN Bridge
End Device

Figure 2.3: Fully Centralized TSN Architecture with shown requirement, capabilities and configuration flows. The clocks in each of the End Devices and TSN Bridges indicate that the components are time synchronized as per IEEE 802.1AS.

Figure 2.4 represents a model for the functions of a TSN Bridge. It contains Traffic Class Queues, a Transmission Scheduling Algorithm and a Gate Control List [27]. When traffic arrives to the Bridge, the traffic class is inspected so the package is sent to the correct queue. The queues are numbered from 0 to 7 and BE, with 7 being high priority and 0 being the lowest priority queue. The transmission scheduling algorithm queues the frames depending on the QoS requirements. The Gate Control List determines which gate is currently in the open or closed state. All the configuration of the Bridge is done in the CNC and depends on the requirements and capabilities of the network as a whole.

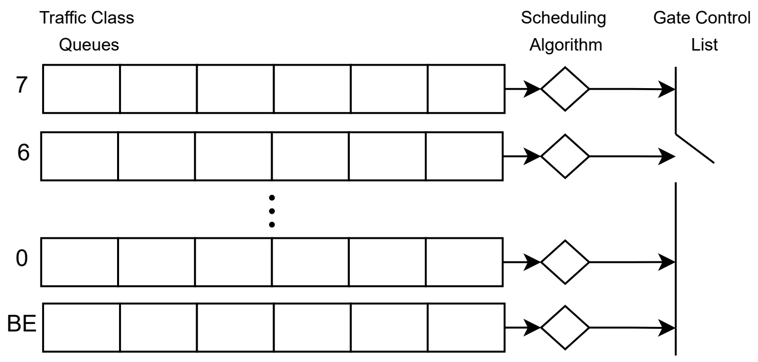


Figure 2.4: A TSN Bridge model containing the Traffic Class Queues, a Transmission Scheduling Algorithm and a Gate Control List

## 2.4 5G

Fifth generation (5G) networks introduce many new functionalities for mobile technologies, not only within traditional voice and mobile broadband but also for domains such as autonomous vehicles, robotic surgery, and industrial control [32]. 5G provides URLLC with extreme data rates, which makes it able to support critical and massive IoT traffic. 3rd Generation Partnership Project (3GPP) is a group of standards developing organization working to develop and maintain the standards in mobile telecommunications. In the release 16 from 3GPP, additions were made to 5G which included increased reliability, adding to its popularity as within the domain of real time systems.

### 2.4.1 URLLC

Ultra Reliable and Low-Latency Communication (URLLC) is a set of features to support new use cases in 5G networks for latency and reliability critical applications[33]. The low latency part of URLLC can be seen as optimisation for a network to be able to transport a large amount of traffic with minimal latency, while still being able to deliver bound latency for critical applications even when the network is under heavy load. URLLC is achieved with centralised scheduling at the base-station using QoS, the scheduler control the wireless resources available in the form of licensed frequencies and the time granularity of the transmission slots. Multiple technologies are applied to increase reliability both for the data and control channel, such as multiple output technologies and packet duplication that are send over individual data channels.

## 2.5 5G-TSN Concepts

The following concepts are a part of the core architecture for a 5G-TSN design; a brief overview is given to understand the function of the individual components better. Firstly concepts regarding the architecture are presented, then the QoS parameters for 5G, know as 5QI. Then the frame structure used for the User Plane (UP) is presented. Lastly, the 5G QoS flow identifier mapping is shown.

- **Control Plane (CP):** Definition of a plane that handles connection management, QoS policies, authentication, and other management functions, separated from the UP.
- **Access and Mobility Management Function (AMF):** Receives information related to 5G sessions within the network and manages handovers between gNB components [34].
- **Unified Data Management (UDM):** Follows the design of the Home Subscriber Service (HSS) in 4G networks. It stores user data information regarding what components are connected to the Network, customer data, and customer information. The 5G additions to the HSS have added cloud functionality and 5G designs.
- **Session Management Function (SMF):** Creates a communication channel between the CP and the UP. It handles the UPF concerning session context by dealing with creating, updating, and deleting PDUs. It communicates with the AF via the PCF, giving the MAC-addresses of the TT per PDU session.
- **Network Exposure Function (NEF):** Provides a secure connection between 5G and third-party applications. Communication with 5G services is done via the NEF.
- **Policy Control Function (PCF):** Receives the QoS information from the AF, and maps the TSN QoS parameters to a 5QI. The 5QI is a scalar reference to certain 5G QoS characteristics, such as priority, Package Delay Budget (PDB), Package Error Rate (PER) and Max data burst volume (MDBV).
- **Application Function (AF):** Communicates with the CNC, with the primary purpose to decide TSN QoS parameters, such as priority and delay based on received configuration information from the CNC.

- **User Plane (UP):** Definition of the plane which deals with data-traffic forwarding, separated from the CP.
- **TSN Translator (TT):** Both the DS-TT and the NW-TT acts as intermediaries between the UP borders of the logical TSN bridge. The Device Side refers to, e.g., actuators or sensors, while the Network refers to the TSN network on the other side of the logical TSN bridge. In essence, they translate the requirements established by the AF and PCF on the flows traversing the logical TSN Bridge [35].
- **User Plane Function (UPF):** The communication scheme established between the gNB and the NW-TT.
- **User Equipment (UE):** Components capable of communicating with the (R)AN.
- **(R)AN:** A 5G capable device which acts as the access point for the wireless side of the logical TSN bridge, sometimes also know as new Generation Node B (gNB). It communicates with one or many UE.

### 2.5.1 5QI - 5G QoS Parameters

A 5G QoS Identifier (5QI) is a list of 5G QoS parameters representing commonly used values for certain types of traffic. Each of the 5QI entries has the following parameters [17]. These parameters are guidelines for setting up the 5G node, or Logical TSN bridge in a specific way.

- **Resource Type:** Can either be Guaranteed Bit Rate (GBR), Non-GBR or Delay Critical GBR, is a definition which indicates how the PDB, PER and MDBV should be handled.
- **Default Priority Level:** The priority level in 5G QoS characteristics gives the priority of scheduling within a QoS flow. It indicates the highest priority with the lowest value, a different level has to be established on a per flow basis, even if the flow comes from the same UE. The standardized 5QI comes with their own default priority values.
- **Package Delay Budget:** Defines an upper bound of how long a package may be delayed between the UE and the UPF.
- **Package Error Rate:** Defines an upper bound of packages that can be processed and sent by the 5G node, but never arriving at their intended destination.
- **Default Maximum Data Burst Volume:** Defines the amount of data that can be sent within a PDB.
- **Default Averaging Window:** Indicates the calculation time of Guaranteed Flow Bit Rate (GFBR) and Maximum Flow Bit Rate (MFBR)

A 5QI entry can be visually represented as shown in Table 2.1, note that the presented 5QI are all of delay-critical GBR, which are recommended when using TSN by [35]. The 5QI parameters listed are part of a much more extensive list of statically assigned parameters. However, they can be set dynamically as well, creating opportunities for highly specified scenarios. The list is derived from the 3GPP, ”Table 5.7.4-1: Standardized 5QI to QoS characteristics mapping” [17, pp. 140–143].

Table 2.1: 5QI Standardized Delay-Critical GBR parameters [17, pp. 140–143]

| **5QI Value** | **Resource Type** | **Default Priority Level** | **PDB** | **PER** | **MDBV** | **Default Averaging Window** | **Example Services** |
|---|---|---|---|---|---|---|---|
| 82 | Delay Critical GBR | 19 | 10 ms | $10^{-4}$ | 255 bytes | 2000 ms | Discrete Automation |
| 83 | Delay Critical GBR | 22 | 10 ms | $10^{-4}$ | 1354 bytes | 2000 ms | Discrete Automation |
| 84 | Delay Critical GBR | 24 | 30 ms | $10^{-6}$ | 1354 bytes | 2000 ms | Intelligent Transport System |
| 85 | Delay Critical GBR | 21 | 5 ms | $10^{-5}$ | 255 bytes | 2000 ms | Electricity Distribution-high Voltage |

### 2.5.2 5G - GTP-U Frame Structure

5G utilizes GTP to encapsulate the frames sent over a tunnel. In the user-plane, the GTP-U version is used [17]. The GTP-U frame structure contains the GTP destination (DST) and Source (SRC), the QFI, which represents the priority level of the flow, the Tunnel Endpoint Identifier (TEID), indicating the tunnel ID for the PDU session anchor, the IP package with their DST and Source IP, as well as the payload, which in this case is indicated by the TSN-frame. Figure 2.5 is a visual representation of the frame structure.

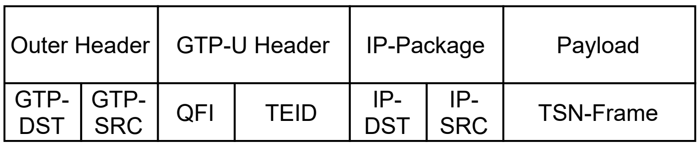


Figure 2.5: GTP-U Frame Structure

### 2.5.3 5G - QoS Flow Identifier Mapping

After a User Equipment has registered with a 5GS Access and Mobility Management Function to authenticate the SIM-card, the UE will first initiate a PDU Session Establishment request to the AMF via the gNB [36]. The parameters exchanged in this setup give the device the default QoS flow. This flow is usually a non-GBR QoS flow without any packet filter; the flow has the lowest precedence and is typically used to signal the AF to set up more QoS demanding flows. To establish a connection between a UE and the receiving network, two things are required; a bi-directional wireless Data Radio Bearer (DRB), and two uni-directional GTP-U tunnels between the gNB and the UPF.

The tunnels, called Access Node (AN) and Core Node (CN), consist of their respective IP-address and the UE's GTP-U TEID, which is a unique identifier for the current flow and used to distinguish the UE's utilizing the tunnel [37]. The IP-address and the TEID are known as the PDU Session Anchor (PSA). Each of the tunnels can have its own established QoS flow; the flows are first established in the SMF. If the UE, gNB, and UPF can satisfy the QoS construct's demands, the QoS Flow Identifier (QFI), which is sent along with the PDU, indicates which parameters the flow should have.

The QFI is set as a value in the GTP header and is the value that will tell the components of the device what parameters they should have while transferring this flow. The header also contains port number 2152, which indicates a GTP-User Plane using either TCP or UDP as the transport protocol [38][39]. This suggests that it's a 3GPP transference and can be used to distinguish 5G traffic being sent. In Figure 2.6 the packaging of the frame is shown. An IP package is generated on the UE side, encapsulated with the GTP-U Header (DRB), containing the QFI and the TEID. On top of this, the outer header is added, including the DST IP of the UPF and the SRC IP of the gNB [40]. Once received by the UPF, the package is de-encapsulated to down to the original IP-Package.

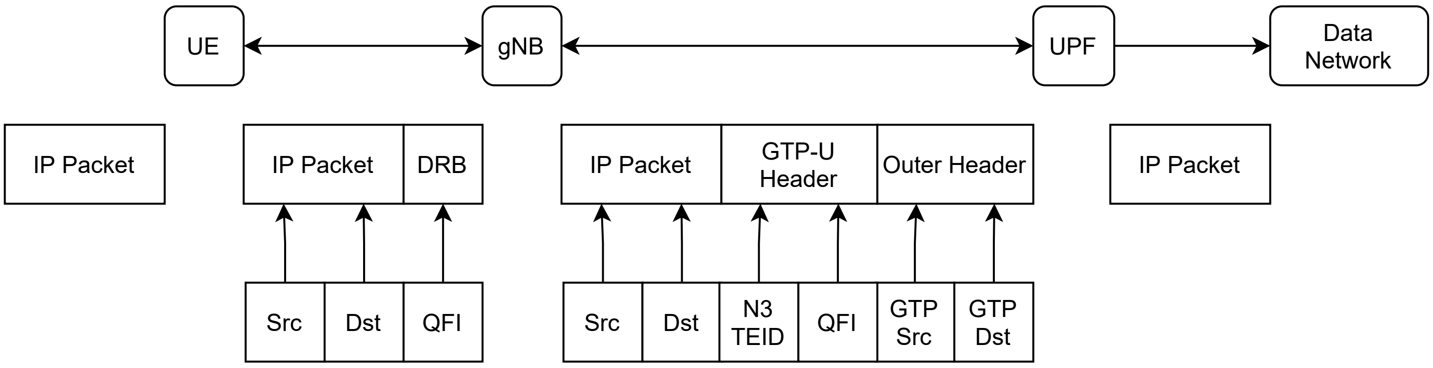


Figure 2.6: 5G QFI flow

## 2.6 OMNeT++

The Objective Modular Network Testbed in C++ (OMNeT++) is a tool which can assist researchers and students in developing network simulators [20]. It includes a framework, library and IDE which allows for building both wired and wireless networks, ad-hoc and on-chip. OMNeT++ is distributed under the Academic Public License, and has a large community behind it, developing various models, extensions and plugins which allows for simulating many different types of networks. The OMNeT++ architecture is based on models which are built from modules connected via gates [41]. These modules are what builds the model, which in turn builds the simulator. The modules are written in C++, making use of the simulation library provided by OMNeT++. See Figure 2.7 for a the hierarchical view how the model or network is built from modules represented by a gray background and compounded modules connected via gates represented via the small boxes connected with arrows.

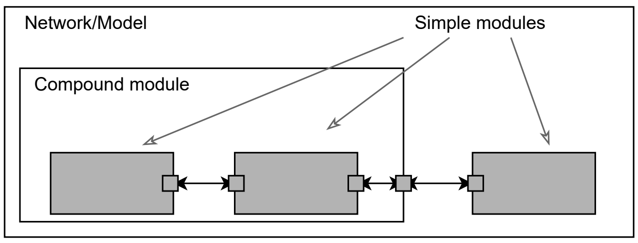


Figure 2.7: Hierarchical view how the OMNeT++ network or model is built from modules programmed in C++.

### 2.6.1 NED

The modules programmed by the users are what creates the model which can then be described in the Network Description (NED) language [41][42]. NED allows a user to define the structure of their network and within OMNeT++ there are built in GUI tools assisting in the process of creating the network structure. In essence, the NED definitions creates the network topology by utilizing the modules.

### 2.6.2 INET Framework

The INET Framework provides a model library to OMNeT++, the library can be utilized for validation purposes and when using the Internet stack, link layer protocols among others [43]. The INET Framework follows the same design concept regarding modules, gates and connections. These can be used to form complex networks consisting of various network devices. As the INET framework is interconnected with OMNeT++ any necessary change to the framework can be done via the OMNeT++ IDE. The framework grants visual support during network simulation, allowing for user to see when packages are dropped, path activity, routing tables and other important aspects useful for a better network overview.

### 2.6.3 NeSTiNg

NeSTiNg is a enhancement of the INET Framework adding a TSN simulation model into OMNeT++ [18]. The NeSTiNg simulator is developed by the Distributed Systems group of IPVS, at the University of Stuttgart [19]. NeSTiNg is built to follow the TSN standards of IEEE, and provide a accurate simulation down to the frame tagging and packet processing on TSN switches. NeSTiNg also implements both the CBS and TAS for scheduling of the traffic in the TSN. NeSTiNg includes other TSN supported features such as preemption and strict priorities that are required to evaluate the timing constraints of a network using TSN.

## 3 Related Work

Martenvormfelde *et al.* [44] have worked on a simulation model for integrating 5G into TSN as a transparent bridge, they utilized OMNeT++, NeSTiNg and a 5G user plane model. Their bridge design is limited to the user plane, derived from the 3GPP 5G architectural model, and is capable of uplink and downlink traffic. Their limited model resulted in that certain characteristics of the New Radio frame structure and sub-carrier spacing affect the end-to-end delay, even in a smaller network. This paper concludes that for larger networks, to establish proper QoS the model has to handle the 5QI, similarly enabling priorities and queues as TSN IEEE 802.1Q. This is something that we propose to do in our thesis by mapping the TSN QoS flows to the 5G QoS.

Ginthör *et al.* [22] presents a system-level simulator that investigates the impact and through it establishes requirements of TSN end-to-end systems. They utilized the OMNeT++ simulator with NeSTiNg model to simulate a TSN-5G network. However, this was done during the Rel. 15 of 3GPP, where certain aspects of Rel. 16 had not yet been fully realized. By converting a 4G architecture, they added characteristics such as; Ethernet PDU sessions and packet filter sets supporting MAC addressing, mini-slots, high-reliability modulation, and 5QI, which are required for a 5G. A conversion from 4G to 5G is something that our thesis also will have to consider. Their simulation setup consisted of multiple UE connected to one base station with a strict prioritization scheme. Through their results, they discovered that integration of 5G and TSN is plausible. Still, the support of TSN functionalities in 5G alone was not enough to meet the requirements of TSN as the inter-arrival time of frames could vary widely. Ginthör *et al.* conclude that a scheduling algorithm that takes these times into account is required to meet the TSN requirements.

Larrañaga *et al.* [35] performs an analysis of and quantifies the 5G Bridge Delay (BD) a closed loop, they showcase how the bridge delay is a relevant component for effective 5G-TSN integration. The example they use indicates some of the open challenges when integrating TSN and 5G. Their work indicates how the mapping between certain TSN Quality of Service (QoS) and 5G QoS parameters is done, as well as showcasing the traversal of the configuration data from the Centralized Network Configuration (CNC) to the Application Function (AF) to Policy Control Function (PCF) and finally to the TSN Translator (TT). Larrañaga *et al.* continues by showing how the BD is calculated and how it has relevancy when integrating TSN and 5G, indicating that all standardized 5QI gives a Package Delay Budget (PDB) of 5 ms or greater. However, similarly Martenvormfelde *et al.* [44] they go on to say that with configured setups the PDB can go down significantly, such as in [45]. While our thesis's goal is not to write a scheduler but rather a translator, the approach regarding BD calculation has to be taken into consideration during implementation. They conclude by giving insight into how the BD is hard to estimate, and a solution where a few packages are sent to establish the BD before TSN-traffic is sent might be a solution.

Ferandez *et al.* [46] proposes a hybrid network, consisting of TSN and IEEE 802.11, designed for industrial control. They establish the TSN with a deterministic access point as a bridge to connect the networks. While their proposed design utilizes IEEE 802.11 as the wireless communication, and ours will use 5G. Their main contribution lies in their soft-handover algorithm, built on a hybrid TSN and IEEE 802.11 MAC scheme, for guaranteed uninterrupted communication. Similarly to our work they utilize the OMNeT++ simulator to evaluate their network.

Bergström [47] has in their Master Thesis "Automatic Generation of Simulated TSN Network Configuration" developed a tool for the NeSTiNg simulator. The tool provides easier generation of TSN network configurations, something that is highly relevant for our thesis as the setup of TSN-networks otherwise can be a lengthy process. His work presents five improvement areas to the simulator; recollection of network properties, manual inputs, visual aid, efficiency and user error.

Donde [48] integrated a 5G system with TSN in a discrete event simulator NS-3 as part of their Master Thesis ”Support for Emulated 5G-System bridge in a Time-Sensitive Bridged Network”. In a similar fashion, our thesis will aim to create a TSN-translator in a different simulator, namely the NeSTiNg based on OMNeT++ In the CNC they established a scheduling algorithm per egress port based on IEEE 8021.Q. The system was tested by measuring jitter, throughput and package loss when the scheduler was active and when it was not. Processing delay was established before run-time and Donde concludes that the overall impact on the network decrease as the variation range for the processing delay is smaller.

# 4 Method

As this Master Thesis will be focused on a system development a method which allows for iterative research has been chosen, to answer the research questions established a systems development research method will be used. The method is derived from Nunamaker and Chen [49], and follows an iterative approach. A visual representation of the method can be seen in Figure 4.1; it allows for greater modularity and backtracking early stages of a project.

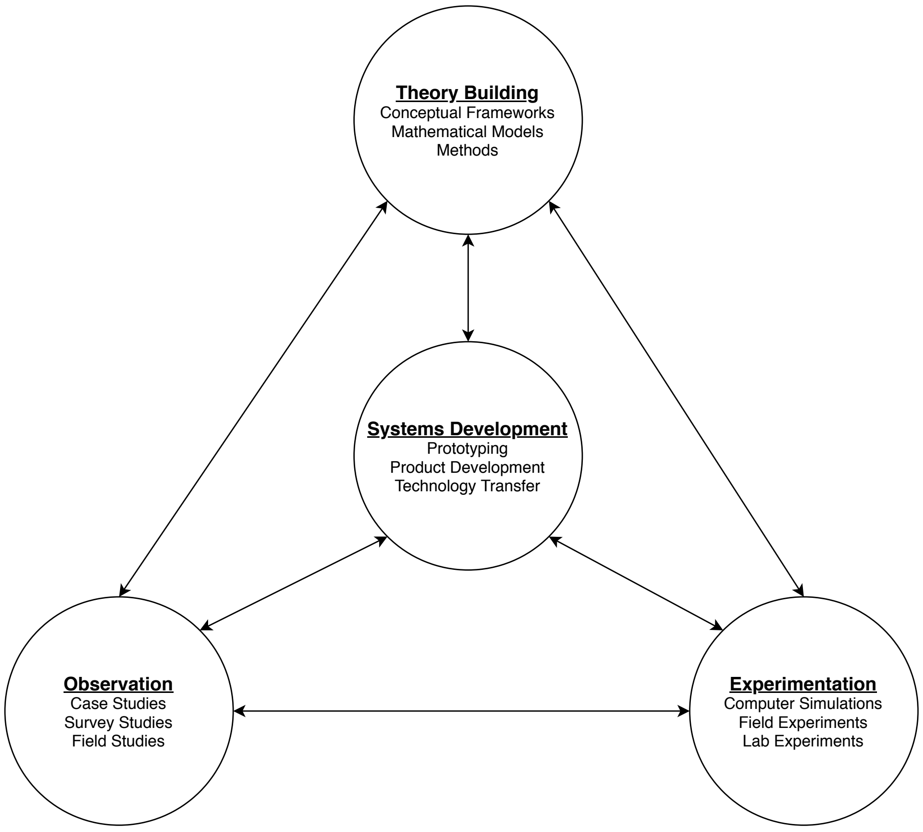


Figure 4.1: Multi Methodical Research Approach [49]

As seen in Figure 4.1, there are many different approaches. To narrow the method down further, the following process will be taken regarding the structure of the system development model. In Figure 4.2 a subset of the model is presented where only relevant parts for the Master Thesis are kept.

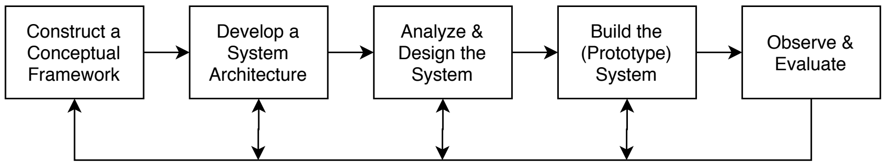

Construct a Conceptual Framework
Develop a System Architecture
Analyze & Design the System
Build the (Prototype) System
Observe & Evaluate

Figure 4.2: Structure of the system development research method

Going from left to right, at first, a literature review and state of the art study was done to identify the appropriate paths to take, which research questions are relevant, and how to further narrow down the work. As there are multiple architectures for a 5G-TSN solution, these had to be investigated in this step, which gave a solid foundation for the second step where each part of the system as a whole was defined in system architecture. Furthermore, as the work was done in the OMNeT++ NeSTiNg simulator, the capabilities of said systems had to be analyzed.

This helped to further limit the system with regards to constraints such as time and scope. A complete picture was made before a proper startup of the build phase by analyzing and designing the system. In this thesis a lot of work was put on the translation of TSN QoS to 5G QoS, before any implementation of such translators could be done, the design had to be established and analyzed. Lastly, the system was evaluated through a comparison method or other relevant methods discovered during the first phase. These steps were done with an iterative approach in mind; going back to a previous step before seeing the current one to completion was something that this method allowed for.
A brief summary of the work carried out in this thesis:

- Literature review and SOTA
- Investigation of existing 5G-TSN architecture solutions
- Current limitations of simulators analyzed
- Design of 5G-TSN translator
- Development of 5G-TSN translator
- Evaluation and Comparison of use-case with or without the 5G-TSN translator

# 5 Investigation of available 5G-TSN approaches

Through investigation, it was determined that there are five different approaches to the 5G-TSN integration. The integrated model has been merged with the Logical Bridge model and is discontinued by the 3GPP [50]. The adapted model requires a lot more specification work on 3GPP and has not been thoroughly investigated. Therefore, the three remaining architectures of note are the TSN Link and the two bridge architectures' designs.

The investigation will have a focus on the 3GPP architectural design of different 5G-TSN approaches. It is essential to distinguish these approaches from a potential simulated approach, which was first considered. However, it was quickly realized that there were no public available simulated approaches to the 5G-TSN approach at the time of writing. Papers that investigated the subject of 5G-TSN integration reference back to the 3GPP design approaches. Thus, in light of there not being any open simulated models, this investigation had to follow the same approach.

The adaption of 5G into TSN is one of the critical issues presented by 3GPP [50]. Integration of 5G into TSN is based on either having 5G with the capabilities of TSN or integrating 5G as a Logical TSN bridge. In the first approach, the 5G is seen as a cable link between the devices, or as integration of specification is added to the 5G network itself [17]. In the second approach, the 5G is seen as a Logical TSN bridge. In both systems, the following needs to be supported to follow both the 5G and TSN requirements. The items on the list are derived from 3GPP TR 23.734 V16.2.0 (2019-06) [50]:

- Ingress and Egress ports connecting TSN end stations or TSN Bridges to the UE;
- Egress and Ingress ports connecting TSN end stations or TSN Bridges to the UPF;
- TSN time synchronization needs to be guaranteed for all Ingress and Egress ports;
- Mapping of TSN QoS requirements to 3GPP QoS requirements and vice versa;
- Mapping of 5GS capabilities to TSN bridge capabilities and vice versa (e.g., TSN bridge Managed Objects (MO);
- TSN bridge capabilities (e.g., bridge-related Managed Objects (MO)) and vice versa (in case of 5GS is modeled as a TSN bridge); or
- TSN link capabilities (e.g., bandwidth, propagation delay) and vice versa (in case of 5GS is modeled as a TSN link);
- TSN bridge self-management for the fully distributed model: handling of TSN information exchange (CUC, CNC, TSN bridges, TSN end stations) according to the TSN models;
- Support of TSN protocols and functionality (e.g., FRER, SRP, MSRP, LLDP, FQTSS, etc.) as defined in IEEE802.1 family for TSN.

## 5.1 5G - Logical TSN Bridge

3GPP presents two approaches to the 5G-TSN system architecture when designing a Logical TSN Bridge. Both system architecture designs have the same general concept where the TSN network sees the 5G network as a black-box TSN bridge, albeit one approach has a more decoupled approach to where functions are located. The first design has the translator located within the UPF, and the second with translators established on the edge of the network. Both Logical TSN approaches consist of the following components in Control Plane (CP): Access and Mobility Management Function (AMF), Unified Data Management (UDM), Session Management Function (SMF), Network Exposure Function (NEF), Policy Control Function (PCF), and Application Function (AF). In the User Plane (UP), the designs are different concerning that the translators are placed differently. One design has the TSN Translator (TT) as part of the User Plane Function (UPF), while the second has decoupled Device Side TSN Translator (DS-TT) and Network TSN Translator (NW-TT) at each end of the network. In the UP, both designs share the following components: User Equipment (UE), (Radio) Access Network ((R)AN) and UPF [17].

## 5.2 5G-TSN System Architecture

A visual representation of the components in an integrated 5G and TSN Network can be seen in Figure 5.1 [50]. In this architecture, the UPF and TT are seen as one component, where the translation of parameters between the TSN and the 5G system is taking place within the UPF itself.

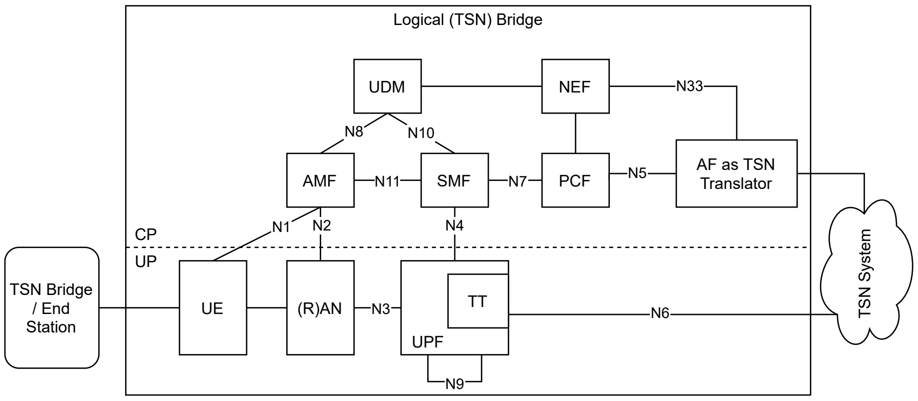


Figure 5.1: System architecture view with 5GS appearing as TSN bridge(TSN Translator (TT) shown within UPF) [50].

In the secondary architecture, the TSN Translators are established on the Device side and Network side of the Logical TSN Bridge. A visual representation of the components in an integrated 5G and TSN Network can be seen in Figure 5.2 [50].

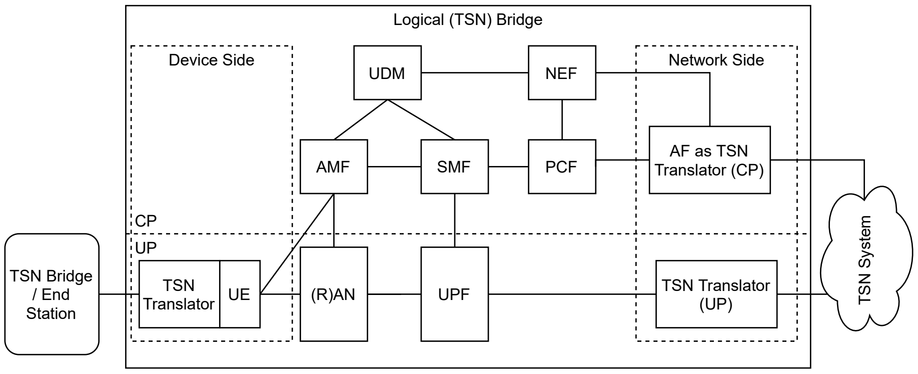


Figure 5.2: System architecture view with 5GS appearing as TSN bridge(TSN Translator shown outside UPF) [50].

The difference between the two architectures lies in where the Translator is located when designing the system. In Figure 5.2 the Translators are modeled on each side of the network, and in Figure 5.1 it is part of the UPF. The introduction of the TSN Translators at the UE side and the network side, makes it possible to reuse many of the existing interfaces defined for 5GS. While having the translator at the UPF would require the 5GS functionalities to communicate via the SMF [50].

To be seen as a Logical TSN Bridge, the parameters and information of the 5G network have to be translated by the AF into TSN standard parameters, and vice versa. In a Logical TSN Bridge integrated into a Fully Centralized TSN architecture, the AF would communicate its capabilities with the CNC, like any other TSN Bridge, albeit translated from the 5G properties to the TSN Bridge properties. The CNC, in turn, would provide the QoS requirements depending on flow and port. In both the architectures, the Logical TSN Bridge is seen as a black box; information within is not seen by the TSN network giving it transparency. Communication flows in and out of the Logical TSN Bridge via the TSN Translators on each side of the flow. On the Device Network side, the translation only needs to be done in the UP, while on the Network Device side, translation is done on both the UP and CP. Depending on the TSN network design, the AF has to support translations of different types of standards. The fully distributed model acts differently compared to the fully centralized model [50].

In essence, seen from the UP's perspective, the Logical TSN Bridge is a virtual tunnel between the UE and the TSN Network. There's a DS-TT, translating the necessary parameters TSN to 5G QoS to establish the message's priority on the Device Side. This is transmitted of the (R)AN to the Network Side, which holds the TSN Network. A NW-TT handles the translation from 5G QoS to TSN QoS so that the flow maintains the correct priority within the integrated network [17].

Both the 5G network and the TSN network maintain time synchronization and are considered time-aware. However, their clocks are not synchronized with each other as they are using different master clocks. To ensure that the QoS regarding deadline criteria is maintained, a bridge delay calculation has to be done within the Logical TSN bridge. This is done by having a timestamp be added to the frame at the ingress port, i.e., at whatever translator is responsible for the flow going from a talker to a listener. After traversing the UP inside the logical TSN bridge, another timestamp is added to the frame as per the generalized precision time protocol in 5G, this time at the egress port. Given these two timestamps, the bridge delay can be added to the TSN time to maintain synchronization [17][35].

## 5.3 5G - TSN Link

In the 5G-TSN link approach, TSN sees the 5G network as a regular cable link between two TSN bridges [50]. It makes use of the already established link model used in TSN and uses a limited number of parameters that are defined by the connected entities. The 5G network would have to propagate its capabilities such as link speed, delay information, and available bandwidth to the link model to establish a link.

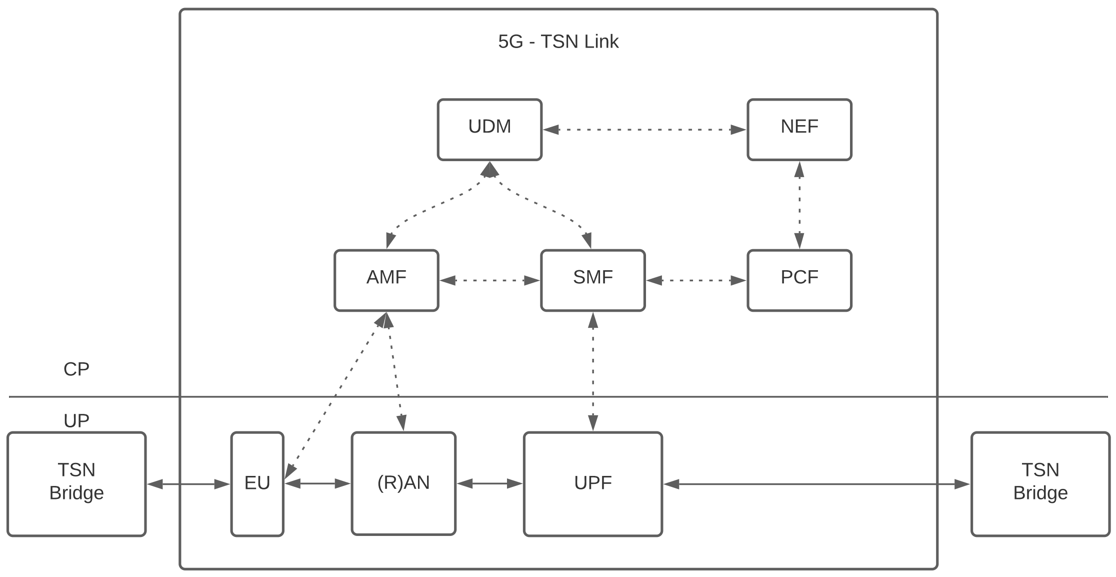


Figure 5.3: Integrated TSN and 5G architecture as a link [50].

## 5.4 Adapted & Integrated TSN Framework

Two architectures that have been either changed or discontinued are the Adapted and the Integrated TSN Framework. The Adapted framework later became part of the Logical TSN Bridge; it utilizes the same idea of a QoS translation between the nodes to ensure QoS requirements are met. However, it does not use the SMF, NEF, and UDM. The Integrated TSN framework allows for each node within a 5G network to communicate with the TSN Network; the SMF, NEF, UDM, gNB, and so on could be seen as a TSN-compatible endpoint. 3GPP writes that such an approach would require an extensive rework of the specification and that it is found to have ”significant difficulties” [50].

# 6 NeSTiNg - Capabilities and Limitations

There are not many choices for a simulator for 5G-TSN integration since not many simulation models for TSN or 5G exist. The ones that do exist are generally not publicly available, or the code is not available to built the extension for 5G-TSN. TSN simulators have been build using discrete-event network simulators such as NS2 and NS3. However, no framework seems to be available. Several projects have no source code available which makes it difficult to move forward with integration of 5G and TSN. The complete TSN framework that is publicly available is NeSTiNg, built over the OMNeT++. At the time of writing this thesis, no publicly available 5G simulators exist.

Integration of 5G into TSN will be done in OMNeT++ making use of the NeSTiNg simulation model for TSN. To make full use of the simulation model, a study on the capabilities and limitations of NeSTiNg is performed. The focus will be on functions of the simulation model of OMNeT++ and NeSTiNg in the area of TSN and 5G.

## 6.1 Capabilities

The base of the OMNeT++ is the NED description language that was designed for extensive scale network simulations [51]. OMNeT++ is also built to be modular, meaning that it is built and designed to be extended with modules of different network protocols, with open data interfaces for input and output, making integration of new technologies possible. NeSTiNg is a module created to integrate a TSN simulation module into OMNeT++ and is built as an enhancement of the Ethernet protocol provided by the INET framework. The main features of the NeSTiNg simulation model include scheduling, gate control, queuing, and frame preemption. Frame preemption is one of the unique problems that was solved since using the INET implementation of Ethernet; it is assumed that any frame that starts transmission is completed before the start of the next frame. This is handled with having a queue for preemptable frames and express frames [18]. NeSTiNg supports both strict-priority, Credit-based scheduling, as well as Time-aware scheduling. Another plugin modules created for OMNeT++ and NeSTiNg is a tool developed to make setting up TSN simulations configurations an automated task, cutting configuration time on TSN simulations [47]. This is to facilitate easier setup and testing of large-scale TSN simulations. This is done by introducing a modular framework to automate offline scheduling.

## 6.2 Limitations

NeSTiNg brings limitations in both missing features and assumptions that are required to test a time-sensitive network thoroughly. NeSTiNg assumes that every clock on the TSN bridges are synchronized and make use of the global simulation clock, instead of simulating time synchronization according to the standard of TSN that make use of IEEE 802.1AS/Asrev. NeSTiNg also does not support IEEE 802.1CB Frame Replication and Elimination for Reliability (FRER) which handles the part of the guarantee that a frame is delivered to the destination. This results in networks with packet loss or delay being affected. Another missing reliability feature is Per-Stream Filtering and Policing (PSFP) which makes it possible for an ingress port to filter frames based on arrival time, rate, and bandwidth, protecting from excess bandwidth usage and limits burst size. But this limitation can be partly negated by using the TAS for scheduling and would only be a problem while using CBS or strict-priority. Another limitation is that NeSTiNg does not support Windows and is only compatible with Linux even when OMNeT++ does support Windows.

# 7 Design & Implementation

This presents the design and implementation of the 5G-TSN translator. First, the plan is presented, and then each necessary component, parameter, or flow is shown. Then all the components are brought together to the finalized design. Finally, the implementation is presented.

## 7.1 5G-TSN High Level Design

The focus of this thesis is on the user plane and the translation of the incoming frames so they can function in both a 5G and a TSN network environment. Mapping of QoS flows to maintain the priority of the flow in both network environments is done by checking the PCP values in the TSN-frames. The PCP value indicates how the 5G system should change its parameters and priorities to maintain the 5QI, which are pre-determined and listed in an XML file. Figure 7.1 represents the high-level design of the proposed Logical TSN Bridge's User plane. On the left side, the frames and package structure is listed, in the middle the frame flow, and on the right the translation logic.

Once a frame arrives at the translator, it is first determined what type of translation has to be done by checking the data's interface. A TSN to 5G translation is performed by de-encapsulating the frame and checking the 802.1Q Header for its PCP value; this value is then mapped to the 5QI, established in the XML-file [52]. The appropriate value to maintain QoS is added, which will tell the used channel how to prioritize the frame when traversing through the 5G system.

A translation from 5G to TSN is a matter of de-encapsulating and egress the frame. As the data is stored in the payload, the PCP value is still available, and no reverse mapping is necessary to maintain the QoS flow. The receiving TSN-bridge checks the PCP value and can prioritize the frame as per usual.

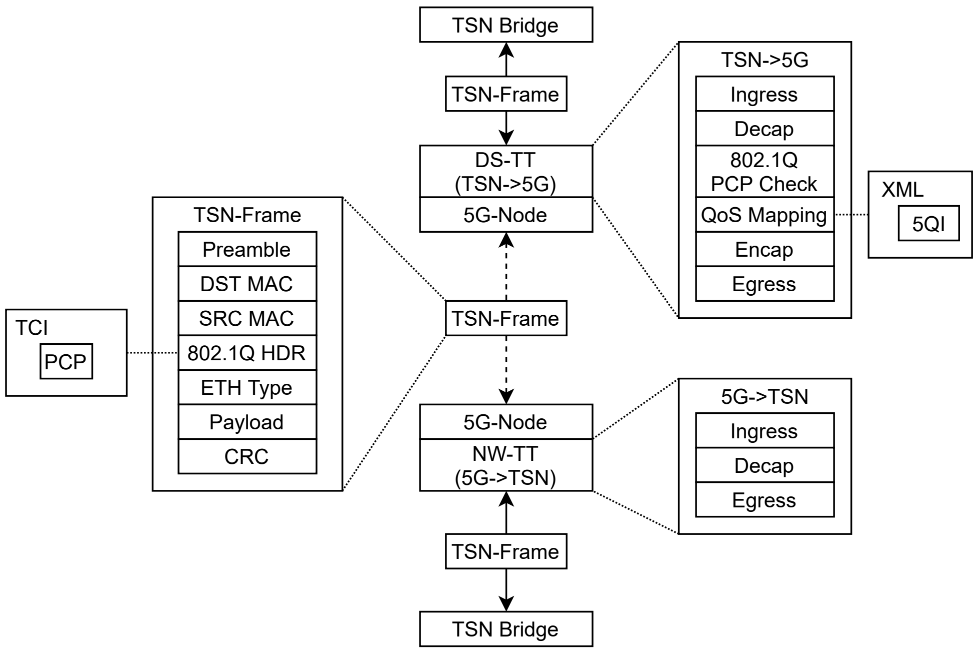


Figure 7.1: 5G-TSN High Level Flow Design

For this version of the translator the payload of the frame is the same as a Ethernet frame of 1500 bytes is utilized. However, for a 5G medium utilizing the standardized

Delay Critical GBR this would not be possible due to the lower max MTU frame size of 1358 bytes [50]. In such a situation the frame would have to be split up and the payload transferred over multiple transmissions.

### 7.1.1 TSN to 5G - Translator Flow

To guarantee that the QoS in both 5G and TSN flows are upheld, there are specific attributes from TSN which has to be mapped to the 5QI and vice-versa. The translator translating from TSN to 5G will have the following design as presented in Figure 7.2.

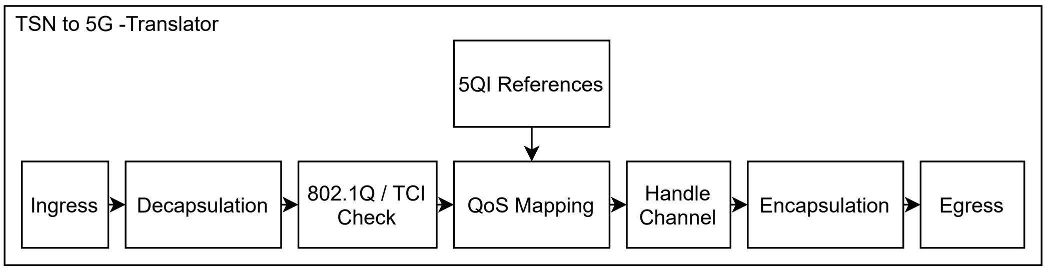


Figure 7.2: TSN to 5G Translator Flow

Each of the fields in Figure 7.2 represents a model or sub-model in the OMNeT++ simulator. The Ingress and Egress modules handle the message by receiving or sending it, which is done in the egressTC or ingressTC sub-modules of OMNeT++. Both de-encapsulation and encapsulation are done in a module called IEtherEncap. Between the IEtherEncap and the medium access control module, a new module has to be integrated, which performs a check to see if the received package is a TSN-Frame, and then see what level of priority it has. From there, the QoS is mapped to the 5QI reference established earlier. This 5QI reference is a pre-configured XML document representing the parameters that should be set on the Logical TSN-bridge. As this implementation does not have access to a 5G medium, the translator instead configures the channel within the Logical TSN-Bridge to act as a 5G medium. This is done in the Handle Channel sub-module. The channel gets the 5QI parameters as listed from the 3GPP Standardized Delay-critical GBR Table 2.1. Once the channel is configured correctly, the translator then encapsulates the package and sends it onward over the channel.

### 7.1.2 5G to TSN - Translator Flow

The 5G to TSN translation is done by de-encapsulating the frame, removing the GTP-U and IP Header, to make the frame function as a TSN frame again. The frame is then sent to the TSN-bridge which can maintain the QoS flow by prioritising the frame appropriately depending on the PCP. The flow can be seen in Figure 7.3.

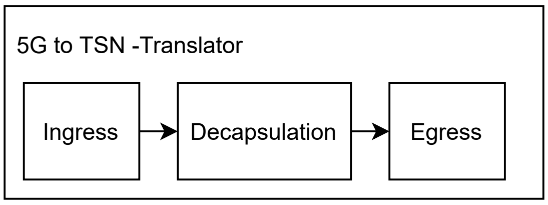


Figure 7.3: TSN to 5G Translator Flow

### 7.1.3 802.1Q - TSN Frame Structure

Mapping a TSN QoS flow to a 5QI flow would first require a controlling system that checks if the frame in question should be translated at all. There might be non-TSN traffic on the network, which should just be sent onward and not handled via specific QoS references. The structure of a TSN frame can be seen in Figure 7.4, where the first 24 bytes consists of a preamble, destination MAC (DST MAC) and, source MAC (SRC MAC) [53]. Then the 802.1Q header, which contains the Tag Control Information (TCI); these are data fields that tell the network how the flow should be prioritized. Lastly the Ethernet Type (ETH Type), the Payload (Data), and the Cyclic Redundancy Check (CRC).

| 8 bytes | 6 bytes | 6 bytes | 4 bytes | 2 bytes | 46-1500 bytes | 4 bytes |
|---|---|---|---|---|---|---|
| Preamble | DST MAC | SRC MAC | 802.1Q HDR | ETH Type | Payload | CRC |

Figure 7.4: TSN Frame Structure

To check if a frame has to be mapped to 5QI, the frame is de-encapsulated down to the 20th byte, i.e., the Preamble, DST MAC and SRC MAC are popped off. Then the next 16 bits are checked; if it is an 802.1Q-frame, the Tag Protocol Identifier (TPID)) field will have the value 0x8100 in it [53]. Once it is established that the frame has the correct TPID, the next 3 bits can be checked. This is the Priority Code Point (PCP) field and contains the TSN Flow's priority value, from 0-7. This field is part of a larger field called Tag Control Information (TCI), which also contains the Drop Eligible Indicator (DEI) and the VLAN Identifier (VID). However, in our proposed implementation, it is assumed that all packets are not drop-able and belong to the same VID; this is done to reduce the number of parameters in the initial stages of implementation. The structure of the 802.1Q Header Frame can be seen in Figure 7.5.

| 16 bits | 3 bits | 1 bit | 12 bits |
|---|---|---|---|
| TPID | TCI | | |
| | PCP | DEI | VID |

Figure 7.5: 802.1Q Header Frame Structure

Once the PCP value has been derived from the frame, it must be mapped to a 5QI parameter. As it is discussed in [35] the TSN flow is listed as a Delay-critical GBR.

## 7.2 OMNeT++ Implementation

The implementation was done in OMNeT++ by integrating modules and sub-modules into a network, some modules were used as is, and others re-written to suit the translator's needs. The 5G Node contains an Ethernet gate (ethg), the TSN-Translator (TT), and the message dispatcher between the two (down). It also has a few sub-modules, the interfaceTable, the filteringDatabase, a scheduleSwap, an oscillator, the legacyClock, and the clock. The interface table and the filteringDatabase indicate where the traffic should go once it has been handled; the indication is done by establishing port and destination mapping rules in an XML document. The oscillator and clock modules deal with the tick time in the module and synchronize itself to the simulation time for time-stamping of logged data. In Figure 7.6 a visual representation of the node can be seen, where each of the squares indicates a submodule, the horizontal line marked as ”down” is the message dispatcher and the arrows are channels connecting the modules. Note that the submodules set on the left side do not require any connectors and should be seen as a way to access information elsewhere in the system.

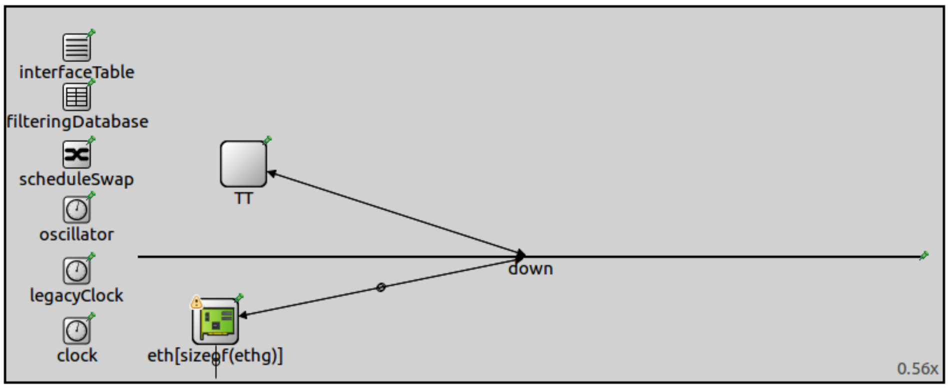


Figure 7.6: 5G Node in OMNeT++

The line underneath the eth submodule leading to the module’s border indicates a way out of the module itself. It gives way for other modules, such as VlanEtherSwitch-Preemptable (TSN Bridges), to connect to the 5G Node. Do Note that the arrows in the 5G node go both ways; both the TT and the eth can send and receive data. The eth has a subclause(sizeof(ethg)), meaning that several eth modules will be generated upon initialization depending on the required amount of gates. Therefore, for example, if there are two connections to the 5GNode, i.e., a connection from the TSN-bridge and one to the 5G medium, there would be two gates established in the node. See Figure 7.7 for a visual representation.

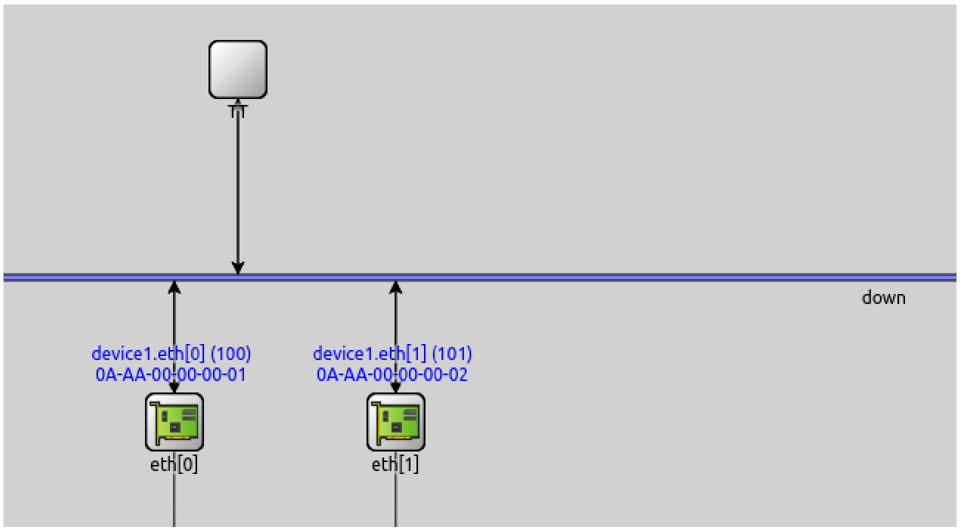


Figure 7.7: 5G Node - Two eth modules

Each of the gates connects to a channel; this channel can be seen as its submodule with established parameters. They are set at initialization to a default value but can then be changed during runtime. These parameters include delay of transfer, packet error rate, or data rate. One of the TSN-Translator (TT) functions is to set this parameter of the channel to the correct value to simulate the max-delay of the PDB to establish a simulated 5G transfer. Similarly, as to how the QFI would change the parameters in an actual 5G node, the TT looks at the incoming transmissions PCP value and changes the channel parameters to pre-determined values set in the XML files.

An example of a network can be seen in Figure 7.8 where two hosts are connected to a TSN bridge and then to a Logical TSN bridge set up by two ”FiveGNodes”. The simulator does not allow a maximum numerical value, thus the naming convention of using characters. The dotted line indicates the 5G channel, which gets manipulated depending on the PCP used by the hosts.

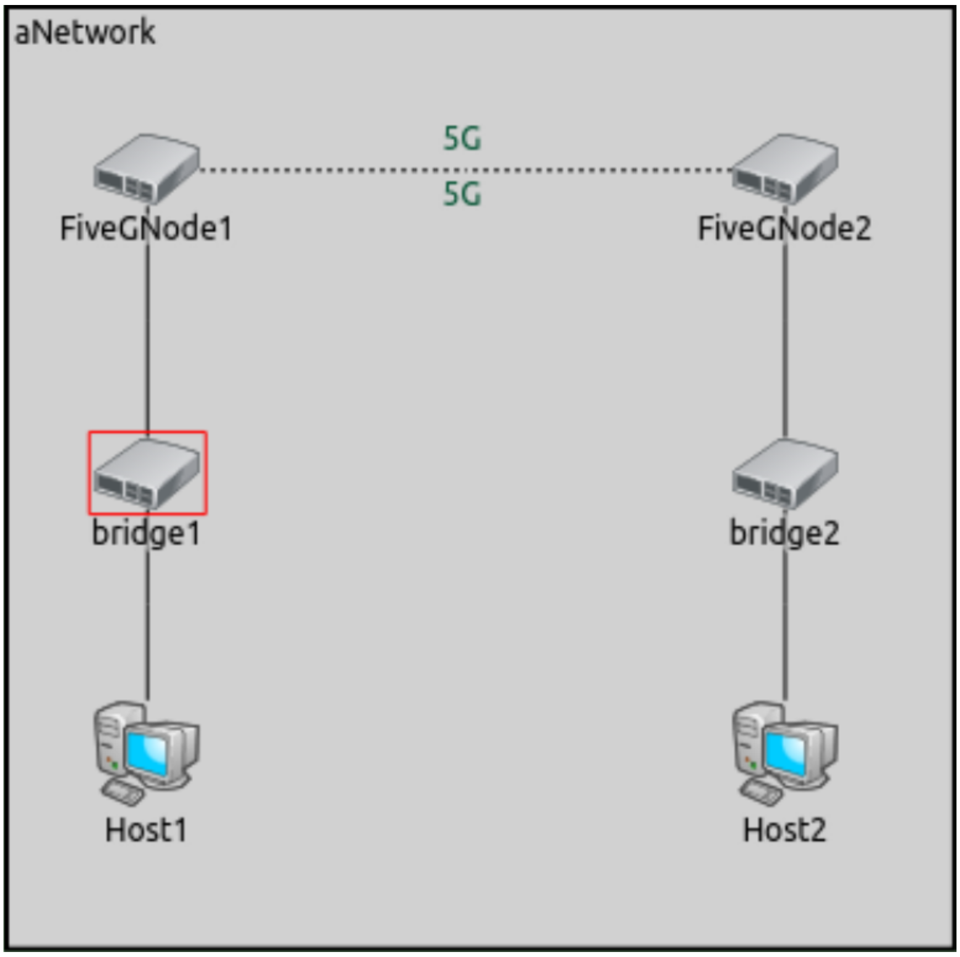


Figure 7.8: 5G Network Example

### 7.2.1 5QI - XML Integration

When the TT is initialized, it gathers the data it requires; one of these data structures is an XML file containing the pre-determined QoS parameters set on a channel. Using an XML file, a user can establish their parameters beforehand for an easy change. This XML file is akin to the 5QI values in an actual 5G node, the loading of such is taken care of by the SMF after negotiating with the AF and PCF. However, as this implementation focuses on the user-plane and the translator, an XML loaded during initialization was an elegant solution. An example of the loaded XML file can be seen in Listing 7.1, which contains the delay, data rate, and PER of the channel. The translator will choose the parameters which correspond with the PCP being sent through the device. In TSN, there are eight different levels of PCP, so technically, eight 5QI parameters could be used; however, only two are listed in the example shown in Listing 7.1. The delayPar sets the delay of the channel to the value indicated, the errorRatePar sets the packet error rate, and the dataratePar sets the datarate. As there are multiple of the same parameter type, the PCP value is used to indicate which of these parameter groups should be chosen.

```xml
<QoSPar>
    <PCP val="1">
        <-- 5QI 83 -->
        <delayPar delay="0.001"/>
        <errorRatePar per="0.00001"/>
        <dataratePar datarate="100000000"/>
    </PCP>
    <PCP val="2">
        <-- 5QI 84 -->
        <delayPar delay="0.003"/>
        <errorRatePar per="0.0000001"/>
        <dataratePar datarate="100000000"/>
    </PCP>
</QoSPar>
```

Listing 7.1: XML-file Parameters

### 7.2.2 TSN-Translator Flow

The TSN-Translator receives the message from the eth submodule; it casts the message to a packet structure and checks the header for an EtherMacHeader, which contains the IEEE8021QHeader. The IEEE8021QHeader has a previously established PCP value set by the Hosts generating the traffic via a TraffGenSchedApp, implemented by the INET package. The Translator reads the PCP and uses it to look up the correct parameters set in the 5QI-XML file. These parameters are then used to change how the channel between the two 5G-nodes acts; for example, a PCP value of 1 will establish a different QoS flow than a PCP value of 2.

## 7.3 Use case Setup - Sensor to Actuator Flow

To demonstrate the TSN-Translator, two different use-cases are created with varying criteria of testing. The use case aims to test the channel propagation time in a camera to actuator flow. In the use case, a network with an actuator(A) controlled by a camera input(S) is implemented. The camera is connected to an aggregation(Agg) for control and calculations; the three devices are connected with a TSN bridge. On a remote TSN network, a Remote Computer (RC) is in charge of deciding the state of the (A). The Two networks are connected with 5G-nodes. The traffic flow is as follows. The (S) sends its data to the (Agg), which does some calculations with it before sending it to the (RC) through the 5G network, the (RC) will then send a message back through the 5G network to (A).

A visual representation of the use case can be seen in Figure 7.9. The parameters can be seen in Table 7.1 where the deadline of the entire end-to-end transmission is set to 50ms.

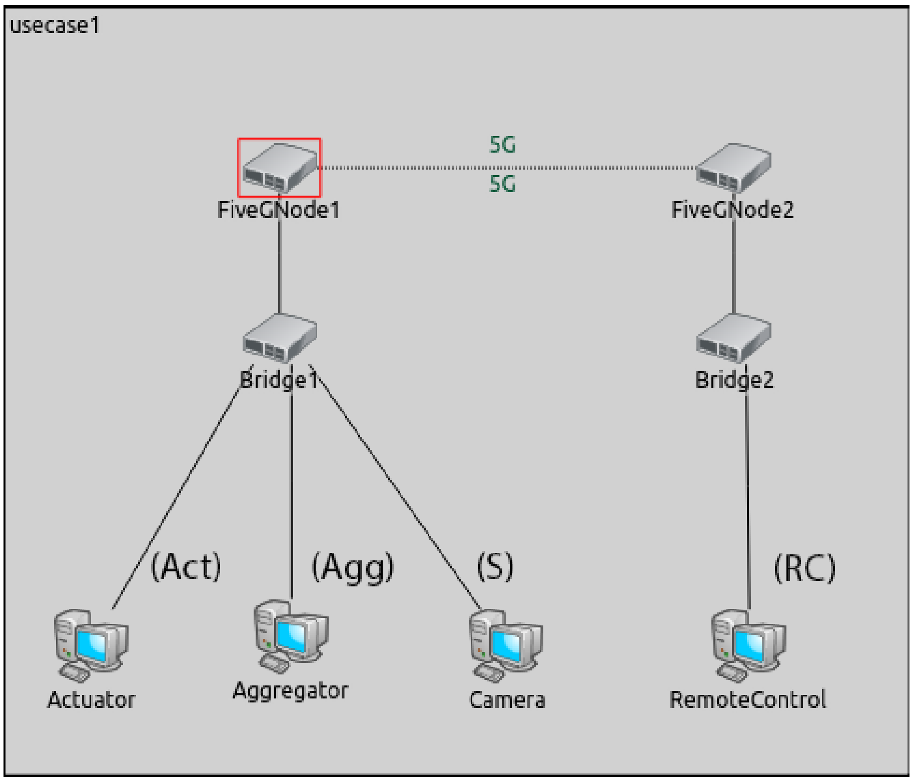


Figure 7.9: Network setup of Use Case

| ID | Path | Class | Period($\mu s$) | Lenght(byte) | Transmission ($\mu s$) |
|---|---|---|---|---|---|
| M1 | S->Agg | TSN | 10000 | 1500 | 123 |
| M2 | Agg -> RC | TSN | 10000 | 1500 | 123 |
| M5 | RC->A | TSN | 10000 | 1500 | 123 |
| M6 | G1->G2 & G2->G1 | 5G | | 5G | |

Table 7.1: 5G-TSN use case Flow Table

### 7.3.1 Simulator Parameters

The parameters were set where appropriate, either on the channels or in the omnetpp.ini file. Each wired channel has a 100Mbit bandwidth and a delay of 123$\mu s$. Each TSN device has a processing delay of 20$\mu s$, while the processing delay in the FiveGNodes is set to 0.5ms to simulate max latency of L2/L3 flow as per 3GPP to fulfill URLLC requirements in the UP [54]. The channel delay time of the 5G channel was set to the parameters indicated by the XML-file value in Listing 7.1 presented earlier. The TSN-Bridges were set with a pre-configured schedule MAC-addressing before run-time. The gates on the TSN bridges were set to StrictPriority, which indicates that they both check if a gate is open and the PCP value of the incoming traffic to determine which flow to prioritize. An overview of the parameters set in the omnetpp.ini file for the use case can be seen in Listing 7.2.

```
[General]
network = usecase1
sim-time-limit = 1s

#Mac Addresses
**.Aggregator.eth.address = "00-00-00-00-00-01"
**.RemoteControl.eth.address = "00-00-00-00-00-02"
**.Actuator.eth.address = "00-00-00-00-00-03"
**.Camera.eth.address = "00-00-00-00-00-04"

# Delays in Switch and 5G
**.processingDelay.delay = 20us
**.FiveGNode*.TT.delayTime = 0.5ms

# MAC Address Tables and Schedule
*.Bridge*.filteringDatabase.database = xmldoc("xml/route.xml", "/
    ↪ filteringDatabases/")
*.Bridge*.filteringDatabase.cycle = xmldoc("xml/sched.xml", "/schedule/")
*.FiveGNode*.filteringDatabase.database = xmldoc("xml/5Groute.xml", "/
    ↪ filteringDatabases/")

# 5QI Channel Parameters
**.FiveGNode*.TT.QoSPar = xmldoc("xml/QoSPar.xml")

# Parameters for Actuator, required for build but will not send anything
**.**.trafGenApp.packetLength = 1500Byte-4Byte # MTU-Size - VLAN-Tag-Size
**.Actuator.trafGenApp.sendInterval = 0ms

# Parameters for schedule and gates
**.gateController.initialSchedule = xmldoc("xml/sched.xml")
**.gateController.enableHoldAndRelease = false
**.Bridge*.eth[*].queuing.tsAlgorithms[0].typename = "StrictPriority"
**.Bridge*.eth[*].queuing.tsAlgorithms[1].typename = "StrictPriority"
**.Bridge*.eth[*].queuing.tsAlgorithms[2].typename = "StrictPriority"
**.Bridge*.eth[*].queuing.tsAlgorithms[3].typename = "StrictPriority"
**.Bridge*.eth[*].queuing.tsAlgorithms[4].typename = "StrictPriority"
**.Bridge*.eth[*].queuing.tsAlgorithms[5].typename = "StrictPriority"
**.Bridge*.eth[*].queuing.tsAlgorithms[6].typename = "StrictPriority"
**.Bridge*.eth[*].queuing.tsAlgorithms[7].typename = "StrictPriority"

# Initial schedule for Hosts
**.trafGenSchedApp.initialSchedule = xmldoc("xml/sched.xml")
```

Listing 7.2: Simulation Parameters

### 7.3.2 MAC-Addressing

The addressing used for the simulator was based on the flow presented earlier, and the submodule required for the mapping was already provided by INET [43]. This addressing is done via an XML document which is then read by the devices. The mapping is done via MAC-address to Port so the devices can make the correct forwarding decision when sending packets onward. In Listing 7.3 a partial XML-file is being shown, the full version can be seen in Appendix B.

```
<filteringDatabases>
 <filteringDatabase id="Bridge1">
     <static>
         <forward>
           <!-- Forward packets addressed to Aggregator -->
           <individualAddress macAddress="00-00-00-00-00-01" port="0" />
               .
               .
               .
```

Listing 7.3: Routing Parameters

### 7.3.3 Schedule Parameters

The schedule used for this use case was created to showcase that the tool functions even with gates not being open at all times. The (S) and (Agg) assume the same priority and set their PCP value to 1, while the (RC) node has its traffic assigned to a PCP of 2. Each of the devices was set with a period of 10ms and varying offsets depending on the arrival time of the previous message in the flow. For example, the (RC) has an offset to 2500 $\mu s$, which is when it takes for the message from the (Agg) to arrive. Table 7.2 shows the period and offset of each of the hosts. The scheduling sub-module was already part of the NeSTiNg packet in OMNeT++ and was derived from the examples provided in their simulation environment [18]. The schedule is designed so that the gates are open as the message arrives at Bridge1 but have a slight delay before they open when they arrive at Bridge2. This was done to simulate a potential configuration where other flows go through the port, which has a different priority. The scheduling works by having gates open or closed for a certain length over a set cycle. The gates are represented by bit-vectors where a 0 is a closed gate, and a 1 is an open gate. In Listing 7.4 part of the schedule can be seen for the (RC) host and the Bridge2 node, these values are used for both initial schedules and for running schedule as indicated by the omnetpp.ini file shown earlier in Listing 7.2. A complete version of the schedule implemented in the use case can be found in Appendix B.

| FlowId | Name | StartTime($\mu s$) | Period($\mu s$) |
|---|---|---|---|
| 1 | Camera | 10 | 10000 |
| 2 | Aggregator | 300 | 10000 |
| 3 | Remote Control | 2500 | 10000 |

Table 7.2: 5G-TSN use case Flow Table

```
    .
    .
    .
  </host>
    <host name="RemoteControl">
    <cycle>10000us</cycle>
    <entry>
      <!-- us -->
      <start>2500us</start>
      <queue>2</queue>
      <dest>00-00-00-00-00-03</dest>/*Remote Control TO Actuator*/
      <!-- byte -->
      <size>1500B</size>
      <flowId>3</flowId>
    </entry>
  </host>
<switch name="Bridge2">
  <cycle>10000us</cycle>
        <port id="0">/*Aggregator TO Remote Control */
                <entry>
                    <length>1700us</length>
                    <bitvector>00000000</bitvector>
                </entry>
        <entry>
                    <length>300us</length>
                    <bitvector>00000001</bitvector>
                </entry>
                <entry>
                    <length>8000us</length>
                    <bitvector>00000000</bitvector>
                </entry>
                .
                .
                .
```

Listing 7.4: Schedule Parameters

# 8 Results

Two main aspects of importance were analyzed during the simulation. The first was the end-to-end delay. The second was that the translator performed channel manipulation during run-time. Figure 8.1 shows the transference time of packages throughout 10ms and a 1ms channel delay time going from FiveGNode1 to FiveGNode2 and a 3ms delay for return traffic. The events indicated on the x-axis are each event of the transfer, where event 1 is the channel between the (S) and Bridge1. The blue line indicates the measured delay for each hop in the simulator, the value of which is seen on the y-axis; this delay is less than the actual delay due to encapsulation delays not being part of the measurements given by the simulator. On events 3, 10, and 17, the total delay is shown for the (S) to (Agg), (Agg) to (RC), and (RC) to (Act), respectively, where the total end-to-end delay is 5.57ms.

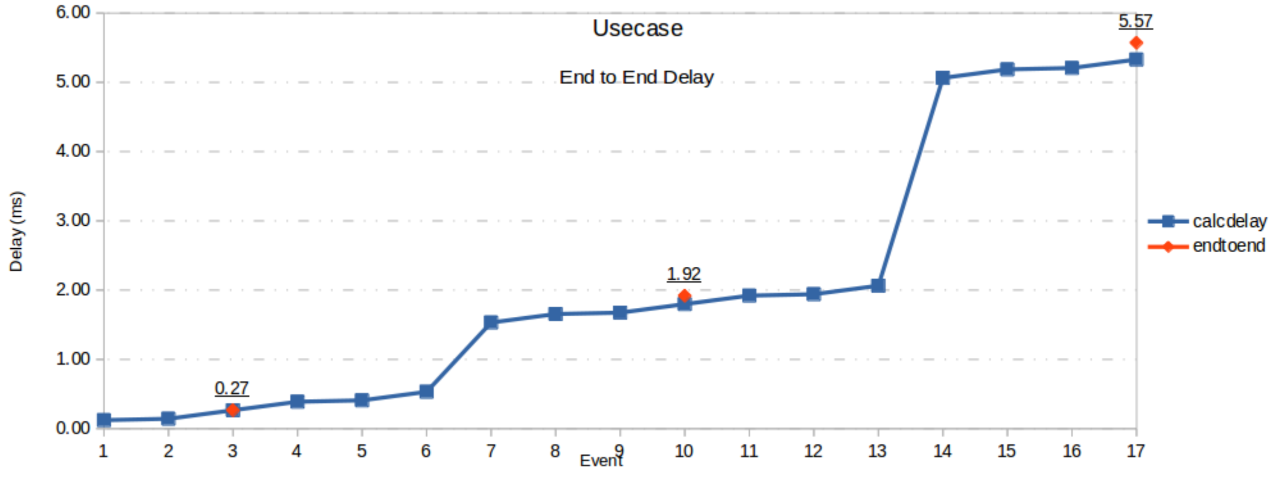
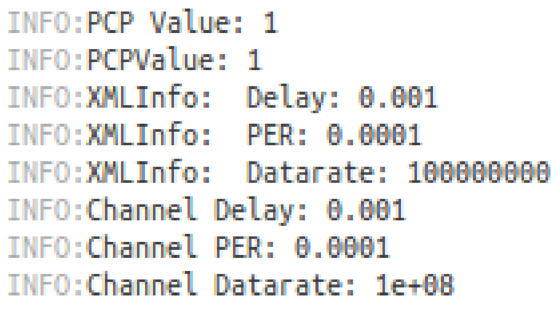


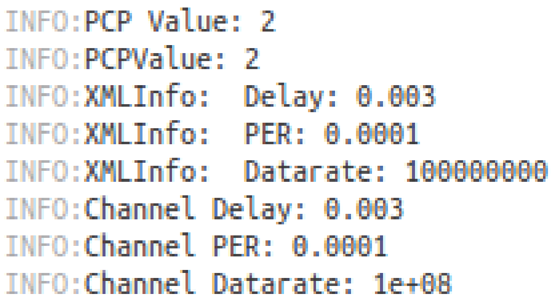
Figure 8.1: Usecase end-to-end Delay

At both events 6 to 7 and events 13 to 14, there's a more significant jump on the y-axis; these jumps are the transference time over the 5G channel. The manipulation of the channel is indicated by first writing the PCP value and then the corresponding XML value, shown by an output of XMLInfo. To make sure the values are set, the channel parameters are also output to the console. This flow of output can be read in Figure 8.2 and Figure 8.3 which shows that two different PCP values give different outputs and corresponding channel delay.



```
INFO:PCP Value: 1
INFO:PCPValue: 1
INFO:XMLInfo:  Delay: 0.001
INFO:XMLInfo:  PER: 0.0001
INFO:XMLInfo:  Datarate: 100000000
INFO:Channel Delay: 0.001
INFO:Channel PER: 0.0001
INFO:Channel Datarate: 1e+08
```

Figure 8.2: PCP 1 Output



```
INFO:PCP Value: 2
INFO:PCPValue: 2
INFO:XMLInfo:  Delay: 0.003
INFO:XMLInfo:  PER: 0.0001
INFO:XMLInfo:  Datarate: 100000000
INFO:Channel Delay: 0.003
INFO:Channel PER: 0.0001
INFO:Channel Datarate: 1e+08
```

Figure 8.3: PCP 2 Output

PCP 1 is used from (Agg) to (RC) and is indicated by events 6 to 7; PCP 2 is used from (RC) to (Act) and is denoted by events 13 to 14. In Figure 8.2 the PCP Value is first to read, then matched to the PCPValue of the XML document as listed by the XMLInfo. This is applied to the channel, indicated by the Channel Delay, PER, and Datarate. For PCP 1 and 2, the delay of 0.001 and 0.003 correspond to the delay-time increase when looking at events 6 to 7 and 13 to 14.

# 9 Discussion

A discussion will be held in this section regarding RQ1 and RQ2 and how they tie into RQ3. The main focus is to showcase why confident design choices were made. While discussing RQ3, the focus will be on the results and their implication. This thesis had two major phases, consisting of investigation and implementation, which follow the summary of the work listed in the method.

## 9.1 5G-TSN System Architecture

An investigation was carried out, which looked into the various system of 5G-TSN architectures. Five different approaches were found and investigated for a 5G-TSN integration [50]—the integrated model, the adapted model, the TSN-Link model, and two Logical TSN Bridge models. The link model was shown to be a more simplistic model with no current support regarding how the link's capabilities could be reported into the TSN Network. The link is seen as an Ethernet cable from the TSN Network side; however, it does not have an Ethernet cable's capabilities, potentially a problem. It would not guarantee the TSN aspects of an integrated system. The Logical TSN Bridge architectures' difference lay in where the translators were located; in the first design, the translator was situated inside the UPF, meaning that the communication scheme also dealt with translating the parameters. In the second design, the translators were decoupled from the flow itself, integrated into the Device Side or the Network Side of the Logical TSN Bridge.

In investigating the current architectures available for a 5G-TSN integration, it has been shown that, in general, the designs are in an early stage of development or discontinued altogether [17]. The integrated TSN model has become part of the Logical TSN bridge as part of this development. The adopted model would require changes to either TSN or 5G to implement the solution. The 5G-TSN link has problems regarding parameters that have to be known for the TSN network side to establish a QoS. Still, it cannot be sent in the architecture's current format as there is no sharing of QoS parameters in either the Control Plane (CP) or the User Plane (UP). A possible solution to this problem is integrating a managed object into a distributed TSN design model. The managed object could be seen as a wireless link instead of as an Ethernet cable, and its respective characteristics could thereby be handled appropriately. In the fully centralized model, the TSN-link capabilities would have to be advertised to establish a CNC schedule; this would, just as in the distributed model, require a change in the 5G or TSN structure.

The Logical TSN Bridge architectures differ in their placements of the TSN translators, where one model has the translator as part of the UPF, and the other has their translators on each side of the bridge [50]. The decoupled translator version gave access to greater reuse of already existing 5GS interfaces on the bridge's Device and Network side in comparing the two bridge architectures. According to 3GPP, the Logical TSN Bridge is a more straightforward approach to implement. It adopts the specified 3GPP 5G QoS frameworks, and minor changes required to the node itself are needed.

## 9.2 Properties of Simulation Tool

Once an architecture was found, namely the Logical TSN Bridge, the limitations of the simulator had to be investigated to see if an implementation of the mentioned architecture was possible and what capabilities and limitations the simulator had [20]. In investigating RQ2, it was found that one of the significant limitations lies in the global clock synchronization, which assumes for a perfect clock. This would not be the case in a real-world

5G-TSN system, where the two networks use different master-clocks, which would have to be handled [17]. However, the fact that OMNeT++ is built to be modular and that most plugin and modules for the simulator follow this design choice lowers the complexity of creating new modules integrated with existing ones. Some functions for reliability that are missing in the NeSTiNg simulator can affect the outcome for large-scale simulations.

## 9.3 5G-TSN Implementation

The investigation of the various architectures and the capabilities of the simulator led to an implementation phase. During this phase, a translator-node was implemented, which acts as the NW/DS-Translator as shown in the Logical TSN Bridge architecture [17]. This architecture was chosen as it was the one that had the most groundwork already laid out. Most of the related works seen during the investigation also pointed at the Logical TSN Bridge being the most frequently used. As there was no 5G medium available at the time of writing, and the creation of such would've been too large of a project for this thesis, it was instead deemed appropriate to create a channel and change its attributes depending on the values of the PCP. Further limitations in the simulator were that the clock was perfectly synced, which is not an aspect that is true to 5G-TSN integration. TSN uses a version of PTP, and 5G uses a gPTP; as they are different systems, they'll use other master clocks. To fully integrate the two, a way to share time is necessary. However, this system has not been deemed a priority for our work and focused on writing the translator. By changing the channel attributes, the timing of a 5G-TSN transmission could be measured, and the translator can, in the future, be used to change other variables within a more developed 5G network environment.

To get a more realistic end-to-end delay, processing of 0.5ms was added to the 5G node, which indicates the encapsulation time required for L2/L3 processing time while still maintaining URLLC, which is deemed necessary for 5G-TSN integration [54]. Attributes added to the translator were the XML file, which acts as the 5QI; the values listed in the 3GPP standardized version include PDB, MDBV, and PER. Out of these, only PDB and PER were used, which acted as the delay and drop rate of packets for the channel. At first, the PDB was always set to the max-value, giving the channel a delay of 10ms; this was later changed in the use case as it did not match the given periods of the host devices. The MDBV could not be used in our implementation as it requires a channel that can split packets into multiple components and then re-assemble them on the receiving side. This is again part of not having access to a 5G medium and should be considered when both the UP and CP are implemented for the 5G node.

The logic and design behind the translator were done by investigating the 5G and TSN designs, the 3GPP Logical TSN-Bridge architecture, and discussions with our supervisors. As the mapping to 5QI parameters required the PCP value to be read, it became necessary to access it. This would be done in five steps, starting the de-encapsulation, then checking the frame to make sure it's a TSN frame. If it is a TSN frame, the PCP value is read, and the corresponding 5QI parameters are added to the channel; the frame is then re-encapsulated and sent onto the 5G channel.

To test that the translator worked, a network consisting of both TSN and 5G devices was created. The network was a typical Sensor to Actuator scenario provided by MDH. The period value of the transmissions set for the use-case did not line up with the PBD-value, which therefore changed from 10ms to 1ms. This change meant that the 5QI values no longer matched the standardized values from 3GPP; however, it is stated that they can be set manually should it be required [50].

The results from the scenario indicate that the translator functions while sending messages with different PCP values over the 5G channel. With the given parameters, an end-to-end delay of 5.57ms could be read. However, this delay was slightly more significant than the calculated one; this has to do with the encapsulation time of the frame not being calculated as an event within the simulator. The scenario showcases a change of channel parameters depending on the read PCP value and is, therefore, the first step towards an improved 5G-TSN integration within the NeSTiNg simulation tool.

The use-case presented in the thesis is a good start for testing the translator, however, it does not give any indication for how well the translator would do in larger scale testing. The parameters for the test where manually set, so to do more thorough testing a more efficient method to set up scenarios would have to be established.

# 10 Conclusion

The purpose of this thesis was to add components to a 5G-TSN simulator in OMNET++. Through an investigation of the 5G and TSN techniques and the study of the 3GPP suggested architectures, the essence of what is required for such a translator was discovered. To apply this theoretical knowledge of 5G and TSN, the capabilities of the OMNET++ simulator, the NeSTiNg and, INET package were also investigated. These aspects lead to this work's main contributing factor, which has been in the development of the 5G-TSN Translator prototype. These prototypes aimed to implement the elements of a 5G-TSN translation that are required to simulate the two networks working together, namely the delay and PER aspects. Through the investigation, it was discovered what requirements have to be supported for a combined 5G-TSN network. Out of these, the Ingress and Egress ports and the QoS mapping have been implemented in the translator. Aspects such as time synchronization, self-management, and protocol support are more advanced and are therefore left for future iterations where a 5G medium is established.

The tool was tested via a sensor to actuator communication flow, utilizing both TSN and the developed translator within a 5G node. The use case was set up with the assigned parameters and a gate schedule in the TSN Bridges. Results indicated that the translator changed the channel parameters during run-time, matching the channel parameters established in the XML file representing 5QI to the PCP values given during network setup.

The translator is at the moment in an early prototype stage. However, the ideas behind it can hopefully be used in future iterations of the project. Should a 5G medium be implemented, the translator might instead manipulate the parameters of the transceiver rather than the channel.The translator can add the encapsulation of IP and GTP rather than utilizing an L2/3 delay value, giving it a more realistic timing aspect. There is a lot to be done in the future, but hopefully, this 5G-TSN translator can be utilized as a stepping stone for a more robust combined network.

## 11 Future Work

While the implementation of a Translator component to the 5G node is a good beginning for a TSN to 5G implementation, there are more functionality that is required in order to have a modular simulator.

The current simulator assumes that the clock synchronization is perfect between the devices in use; this is not the case in a real-world scenario where TSN and 5G have their clock synchronization schemes. The 5G system acts as a time-aware system via gPTP according to IEEE 802.1AS; something that enables synchronization of network nodes and end devices to a grandmaster clock over 5G [55]. The TSN time synchronization is done via Precision Time Protocol (PTP), with its profile IEEE 1588 PTP designed by the IEEE 802.1AS workgroup. Industrial systems that communicate with each other in real-time need a shared understanding of time to agree on corrective actions recognize each other's state, and cooperate. Time synchronization is a vital part of a functioning system for it to be predictable. The sharing of time information between the TSN and 5G system is a required step for a fully synchronized system. 3GPP recommends a transparent relay over the 5G network, where they are measured and compensated for [17][56]. The first step towards this implementation could be by looking at ingress and egress time of initial packets to the 5G Node and using this calculates a processing time used for the time compensation. Alternatively, a PTP to gPTP timing translation could be done; however, this might cause two grandmaster clocks in the system.

At the time of implementation, the simulator did not have access to a 5G medium; this lead to a novel approach where a channel was treated as the 5G medium. This was done to limit the work; otherwise, the entire 5G node had to be implemented. However, the workgroup Simu5G is, as of writing, working on a 5G package to the simulator [21]. Using Simu5G and NeSTiNg together with the translator would give a more true to real-world simulation as there are aspects in the medium that a channel cannot take into consideration [18]. Setting up the 5G medium would give a user access to roaming, environmental disturbances, effects of multi-path propagation, etc. The 5G nodes from Simu5G would have to be combined with our Translator module, which would then check either the PCP or QFI value of the incoming packets and take the appropriate action in QoS flow. With the 5G node, the flow could be established regarding different TEID, and aspects such as tunnel re-use when sending to the same UE depending on QoS flow parameters could give an improved overhead due to less re-forming of tunnels [39]. An implementation of the 5G node and radio medium could also mean that the various functions of 5G are to the disposal of the user, meaning that the 5QI values which are now implemented as an XML file could instead be generated by the PDU session establishment message sequence [17][37].

A thoroughly combined TSN and 5G network would require a translator and intercommunication between the two systems to set up the correct properties. 3GPP lists this as being done via communication of the CNC and the AF. The CNC and CUC collect the requirements and properties from all the devices in the 5G-TSN network, and then from these values calculates a schedule. The generated schedule and priorities would then have to be integrated into the 5G by the SMF via the AF, and pre-created 5QI parameters would then do matching. The interchanging of such parameters and the schedule generation would significantly improve the combined 5G-TSN structure and a vital role in its modularity. Such an improvement would, in essence, enable the network to generate a schedule on its own depending on the requirements and properties of the devices in the network.

When looking at the implementation done in this work, several improvements can be made. The current translator can only handle one output; this is not viable when it comes to a 5G medium as there will be more than one QoS flow leaving the system. Should the Simu5G be implemented, this might not be required to change as the translator would then use a different egress than it currently does. The change to Simu5G will also need another encapsulation than what is presently being used, as there's then necessary to establish an IP and a GTP-U header onto the package before sending it along. The package is currently an EthernetMacHeader as the topmost encapsulation; this would be required to be encapsulated further to function in an upgraded system. The IP would require a destination and source IP address, and the GTP-U header would contain the QFI and the TEID of the established tunnels. These encapsulations can be defined as messages in the simulator and then added to the package before sending it onward. Said implementation is not done currently as it serves no purpose for the translator and channel-changing approach taken in this thesis.

As no 5G medium was usable, there was also no way to change the properties required in a 5G system; instead, the changes happen on the channel being used by the TSN Translator. Should a 5G system be implemented, the properties that have to be changed would also have to be swapped as the channel is no longer in use. Instead, the change should happen on the medium parameters. Depending on the implementation, this can be done in a few different ways, but the general approach could be to read either the PCP or the QFI value out of the package after de-encapsulating it. And it is then traversing the sender module and sub-modules trees to reach the appropriate parameter to change before letting the packet traverse the medium. The medium could then be altered to follow established 5QI rules by implementing them in the SMF or following the current scheme by having them in a separate XML file. With access to the 5G medium and wireless transmission, it'll also be possible to package the frame into the appropriate Max data burst volume (MDBV). The MDBV can only handle frames up to 1354 bytes, but an Ethernet Frame is 1500 bytes large; hence the frame has to be split to fulfill the requirements set by 5G when traversing the 5G medium. This also then means that the package has to be re-ensemble on the receiving end of the communication. The overhead of this could be an essential aspect to analyze when doing the initial setup of a 5G-TSN system as it could be significant enough for the deadlines not to be met, especially when taking dropped packages into consideration.

## References

[1] G. C. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*, 3rd. Springer Publishing Company, Incorporated, 2011.

[2] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0," *IEEE Industrial Electronics Magazine*, vol. 11, no. 1, pp. 17–27, 2017.

[3] G. P. Fettweis, "The tactile internet: Applications and challenges," *IEEE Vehicular Technology Magazine*, vol. 9, no. 1, pp. 64–70, 2014.

[4] A. Finzi, A. Mifdaoui, F. Frances, and E. Lochin, "Incorporating tsn/bls in afdx for mixed-criticality applications: Model and timing analysis," in *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, 2018, pp. 1–10.

[5] A. S. Siddiqui, Y. Gui, J. Plusquellic, and F. Saqib, "Secure communication over canbus," in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2017, pp. 1264–1267.

[6] EtherCAT Technology Group, *Ethercat the ethernet fieldbus*, accessed: 2021-01-21. [Online]. Available: https://www.ethercat.org/default.html.

[7] Siemens, *The pulse of the digital enterprise – intelligent networking with profinet*, accessed: 2021-01-21. [Online]. Available: https://new.siemens.com/global/en/products/automation/industrial-communication/profinet.html.

[8] S. Mubeen, E. Lisova, and A. Vulgarakis Feljan, "Timing predictability and security in safety-critical industrial cyber-physical systems: A position paper," *Applied Sciences*, vol. 10, no. 9, 2020. [Online]. Available: https://www.mdpi.com/2076-3417/10/9/3125.

[9] IEEE, *Ieee 802.3 ethernet working group*, accessed: 2020-11-24. [Online]. Available: https://www.ieee802.org/3.

[10] FLUKE networks, *White paper: Category 8 cable's role in 25g & 40g dc ethernet*, accessed: 2021-01-21. [Online]. Available: https://www.donet.se/upload/documents/Fluke/Category_8_Cable_039_s_Role_in_25G_amp_40G_DC_Ethernet-219421-7001607.pdf.

[11] J. D. Decotignie, "Ethernet-based real-time and industrial communications," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1102–1117, 2005.

[12] IEEE, *Time-sensitive networking (tsn) task group*, accessed: 2020-11-24. [Online]. Available: https://1.ieee802.org/tsn/.

[13] V. Gavriluţ and P. Pop, "Scheduling in time sensitive networks (tsn) for mixed-criticality industrial applications," in *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, 2018, pp. 1–4.

[14] M. Ashjaei, L. L. Bello, M. Daneshtalab, G. Patti, S. Saponara, and S. Mubeen, "Time-sensitive networking in automotive embedded systems: State of the art and research opportunities," *Journal of Systems Architecture*, vol. 117, p. 102 137, 2021.

[15] J. Farkas, B. Varga, G. Miklós, and J. Sachs, *5g-tsn integration meets networking requirements for industrial automation*, accessed: 2021-01-21. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/5g-tsn-integration-for-industrial-automation.

[16] ——, *Boosting smart manufacturing with 5g wireless connectivity*, accessed: 2021-01-21. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/boosting-smart-manufacturing-with-5g-wireless-connectivity.

[17] 3GPP, *3rd generation partnership project;technical specification group services and system aspects; system architecture for the 5g system (5gs); stage 2 (release 16)*, accessed: 2021-01-21. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144.

[18] J. Falk, D. Hellmanns, B. Carabelli, N. Nayak, F. Dürr, S. Kehrer, and K. Rothermel, "NeSTiNg: Simulating IEEE time-sensitive networking (TSN) in OMNeT++," in *Proceedings of the 2019 International Conference on Networked Systems (NetSys)*, Garching b. München, Germany, March 2019.

[19] University of Stuttgart, *Institute for parallel and distributed systems (ipvs)*, accessed: 2021-01-21. [Online]. Available: https://www.ipvs.uni-stuttgart.de/.

[20] OMNET++, *Omnet++ discret event simulator*, accessed: 2020-11-25. [Online]. Available: https://omnetpp.org/.

[21] G. Nardini, D. Sabella, G. Stea, P. Thakkar, and A. Virdis, "Simu5g–an omnet++ library for end-to-end performance evaluation of 5g networks," *IEEE Access*, vol. 8, pp. 181 176–181 191, 2020.

[22] D. Ginthör, J. von Hoyningen-Huene, R. Guillaume, and H. Schotten, "Analysis of multi-user scheduling in a tsn-enabled 5g system for industrial applications," in *2019 IEEE International Conference on Industrial Internet (ICII)*, 2019, pp. 190–199.

[23] HMS - Connecting Devices, *Industrial ethernet is now bigger than fieldbuses*, accessed: 2021-01-21. [Online]. Available: https://www.hms-networks.com/news-and-insights/2018/02/27/industrial-ethernet-is-now-bigger-than-fieldbuses.

[24] R. Sabella, A. T. M. C. Carrozza, and M. Ippolito, *Industrial automation enabled by robotics, machine intelligence and 5g*, accessed: 2021-01-21. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/industrial-automation-enabled-by-robotics-machine-intelligence-and-5g.

[25] M. Barr and A. Massa, *Programming Embedded Systems*. October 2006.

[26] Q. Li and C. Yao, *Real-Time Concepts for Embedded Systems*. January 2003.

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

[28] J. L. Messenger, "Time-sensitive networking: An introduction," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 29–33, 2018.

[29] J. Wen-Kang, L. Gen-Hen, and C. Yaw-Chung, "Performance evaluation of ieee 802.1qbu: Experimental and simulation results," October 2013, pp. 659–662.

[30] "Ieee standard for local and metropolitan area networks–bridges and bridged networks – amendment 31: Stream reservation protocol (srp) enhancements and performance improvements," *IEEE Std 802.1Qcc-2018 (Amendment to IEEE Std 802.1Q-2018 as amended by IEEE Std 802.1Qcp-2018)*, pp. 1–208, 2018.

[31] M. Pahlevan, J. Schmeck, and R. Obermaisser, "Evaluation of tsn dynamic configuration model for safety-critical applications," in *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.

[32] A. Zaidi, *5G physical layer : principles, models and technology components*. London: Academic Press, 2018.

[33] A. Zaidi, A. Bränneby, A. Nazari, M. Hogan, and C. Kuhlins, *Cellular iot in the 5g era: Realizing cellular iot for industrial transformation*, June, 6. 2019. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/white-papers/cellular-iot-in-the-5g-era.[Accessed: MAR 12, 2021].

[34] Ericsson, *Network architecture domains*, accessed: 2021-02-18. [Online]. Available: https://www.ericsson.com/en/future-technologies/architecture/network-architecture-domains.

[35] A. Larrañaga, M. d. C. Lucas-Estañ, I. Martinez, I. Val, and J. Gozalvez, "Analysis of 5g-tsn integration to support industry 4.0," Sep. 2020.

[36] D. Cheung, *5g core part 1 — architecture overview*, accessed: 2021-03-19. [Online]. Available: https://derekcheung.medium.com/5g-core-pdu-session-and-qos-part-1-a12852e1b342.

[37] ——, *5g core part 1 — architecture overview*, accessed: 2021-03-19. [Online]. Available: https://derekcheung.medium.com/5c-core-pdu-session-and-qos-part-2-52bb72cb0ff7.

[38] Internet Assigned Numbers Authority, *Gtp-user plane (3gpp)*, accessed: 2021-03-19. [Online]. Available: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml?search=2152.

[39] D. Cheung, *5g core part 1 — architecture overview*, accessed: 2021-03-19. [Online]. Available: https://derekcheung.medium.com/5gc-part-3-user-plane-and-gtp-u-tunnels-53319463967.

[40] Netmania, *5g traffic flow*, accessed: 2021-03-19. [Online]. Available: https://www.netmanias.com/en/?m=view&id=oneshot&no=14104&page=2.

[41] Omnet++, *Simulation manual omnet++ version 5.6.1*, accessed: 2021-01-21. [Online]. Available: https://doc.omnetpp.org/omnetpp/manual/#sec:introduction:what-is-omnetpp.

[42] OMNeT++ Discrete Event Simulation System, *Ned language overview*, accessed: 2021-01-21. [Online]. Available: https://www.ewh.ieee.org/soc/es/Nov1999/18/ned.htm.

[43] INET Framework, *An open-source omnet++ model suite for wired, wireless and mobile networks. inet evolves via feedback and contributions from the user community.* accessed: 2021-01-21. [Online]. Available: https://inet.omnetpp.org/.

[44] L. Martenvormfelde, A. Neumann, L. Wisniewski, and J. Jasperneite, "A simulation model for integrating 5g into time sensitive networking as a transparent bridge," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2020, pp. 1103–1106.

[45] D. Pannell, *Choosing the right tsn tools to meet a bounded latency in ieee sa ethernet & ip*, accessed: 2021-02-02. [Online]. Available: https://www.allaboutcircuits.com/industry-articles/choosing-the-right-tsn-tools-to-meet-a-bounded-latency/.

[46] Z. Fernández, Ó. Seijo, M. Mendicute, and I. Val, "Analysis and evaluation of a wired/wireless hybrid architecture for distributed control systems with mobility requirements," *IEEE Access*, vol. 7, pp. 95 915–95 931, 2019.

[47] A. Bergström, *Automatic Generation of Network Configuration in Simulated Time Sensitive Networking (TSN) Applications*, Master Thesis, School of Innovation, Design and Engineering, Mälardalen University, Sweden, 2020.

[48] S. Donde, *Support for Emulated 5G-System Bridge in a Time-Sensitive Bridged Network, Master Thesis*, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Sweden, 2020.

[49] J. F. Nunamaker, M. Chen, and T. D. Purdin, "Systems development in information systems research," *Journal of Management Information Systems*, vol. 7, no. 3, pp. 89–106, 1990.

[50] "Technical specification group services and system aspects; study on enhancement of 5g system (5gs) for vertical and local area network (lan) services (release 16)," en, 3rd Generation Partnership Project, 650 Route des Lucioles - Sophia Antipolis Valbonne - FRANCE, Technical report, Jun. 2019. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3487.

[51] V. András and H. Rudolf, "An overview of the omnet++ simulation environment," in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, ser. Simutools '08, Marseille, France: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, ISBN: 9789639799202.

[52] iTecTec, *5.7 qos model*, accessed: 2021-03-30. [Online]. Available: https://itectec.com/spec/5-7-qos-model/.

[53] Cisco, *Inter-switch link and ieee 802.1q frame format*, accessed: 2021-03-15. [Online]. Available: https://www.cisco.com/c/en/us/support/docs/lan-switching/8021q/17056-741-4.html.

[54] 3GPP Tech. Rep. 38.913, "Study on Scenarios and Requirements for Next Generation Access Technologies (Release 14)," v14.2.0, 2017.

[55] Intel, *Adopting time-sensitive networking (tsn) for automation systems*, accessed: 2021-04-21. [Online]. Available: https://software.intel.com/content/www/cn/zh/develop/articles/adopting-time-sensitive-networking-tsn-for-automation-systems-0.html.

[56] Ericsson, *5g synchronization requirements and solutions*, accessed: 2021-04-21. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/5g-synchronization-requirements-and-solutions.

## Appendices

## A Simulation Environment Setup Guide

This simulator setup guide is meant as a joining of to guides provided in https://www.diva-portal.org/smash/get/diva2:1438082/FULLTEXT01.pdf and https://doc.omnetpp.org/omnetpp/InstallGuide.pdf.

### A. Initial Setup

This setup is used for setup of the simulation environment on a Ubuntu 16.04.7 LTS (Xenial Xerus) https://releases.ubuntu.com/16.04/, the guide might not work for other systems.

After installation of Ubuntu 16.04.7, open a terminal and do the following prerequisite setup:

```
$ sudo apt-get update
$ sudo apt-get install build-essential gcc g++ bison flex perl gedit \
    python python3 qt5-default libqt5opengl5-dev tcl-dev tk-dev \
    libxml2-dev zlib1g-dev default-jre doxygen graphviz libwebkitgtk-3.0-0
```

Install the visualization tool Qtenv, first set up the Universe software repository:

```
$ sudo add-apt-repository ppa:ubuntugis/ppa
$ sudo apt-get update
$ sudo apt-get install openscenegraph-plugin-osgearth libosgearth-dev
```

### B. OMNeT++

First, extract the OMNeT++ 5.6.1 folder where you want to install it. /home works well for this. Then cd into the folder and set it as the working environment.

```
tar xvfz omnetpp-5.6.1-src.tgz
$ cd omnetpp-5.6.1
$ . setenv
$ gedit ~\.bashrc
```

Add the following line at the end of the document:

```
export PATH=$HOME/omnetpp-5.6.1/bin:$PATH
```

Then close the terminal and open it again to make the line take effect.
Navigate to the OMNeT++ directory again and start the installation process.

```
$ cd omnetpp-5.6.1
$ ./configure
```

The configure process will create the necessary makefiles, and also warn you if some prerequisites are missing. Should you run into errors here, please resolve them before moving on. Usually Akaroa is missing, this is normal and is optional software. Move on by typing:

```
$ make
```

Wait for OMNeT++ to the built, this is a fairly long process depending on your system. After OMNeT++ has been built, you can use the terminal to launch the program:

```
$ omnetpp
```

The program will ask you to create a workspace, name it what you wish. We will continue to refer it as workspace in this guide. OMNeT++ will prompt you to install INET and an example program. Decline both. Next up we will install INET.

## C. INET and NeSTiNg

To install INET, do the following:

```
$ cd omnetpp-5.6.1/workspace/
$ git clone --branch v4.1.2 --depth 1 https://github.com/inet-framework/
   ↪ inet.git
```

Then, download the 0.9.x version of NeSTiNg from https://gitlab.com/ipvs/nesting/-/tree/v0.9.x, extract the zip in your workspace folder.
Your workspace should now look like:

```
 <workspace>
   |-- nesting
   |-- inet
```

Next, open OMNeT++ and open your workspace. From there, rightclick in the project explorer, choose *Import->General->Existing Project into Workspace*. Select the root directory, and select the workspace you created. The INET and NeSTiNg should show up. Select both and then click Finish. Right click the projects in the project explorer and chose Build Project. If setup has been done correctly both projects should build.

## D. TSN Autoconfiguration

To set up the TSN Autoconfiguration tool [47] used in this thesis. Download the ***org.plugin.tsnsched_1.0.0.202007131134.jar*** from https://gitlab.com/abbelini/TSN-plugin/-/releases. Add it to the "<the_omnet_directory>/ide/plugins" directory.

Then, navigate too
”<the_omnet_directory>/ide/configuration/org.eclipse.equinox.simpleconfigurator/bundles.info”
and add the line

```
org.plugin.tsnsched,1.0.0.202007131134,plugins/org.plugin.tsnsched_1
   ↪ .0.0.202007131134.jar,4,false
```

For more info on how to use the tool, please refer to https://www.diva-portal.org/smash/get/diva2:1438082/FULLTEXT01.pdf

## E. 5G-TSN Translator

To use the Translator first it has to be downloaded. Clone or go directly to the repository at https://gitlab.com/DavidPantzar/5GTSNTranslator and extract it to your workspace. The workspace should now look like.

```
<workspace>
  |-- nesting
  |-- inet
  |-- FiveGTSNTranslator
```

The package has the following folder structure, not listing the must have folders and files such as out, simulations, includes and binaries which are auto-generated by OMNET++.

```
<FiveGTSNTranslator>
  |-- Src
      |-- examples
          |-- UseCase1
              |-- results
              |-- xml
      |-- nodes
```

Src contains two folders, namely examples and nodes. Within nodes there are the .ned files and the sourcecode for the nodes themselves. There are two NED files which can be used by implementing them as one would any other within a network structure for OMNET++ [41]. The CC and H file contains the source-code for the written TSN-Translator. They contain the initialization, handleChannel, handleMessage and getPCP functions. They also set up the requirements for addressing and reading of the 5QI XMLElement.

UseCase1 contains the usecase presented in this report in section 8. The results folder contains any results generated after running a simulation. The XML folder contains the XML files required to run the simulation, namely the QoSPar, route, sched and 5Groute. These files can be used as references when setting up your own network. In UseCase1 there's also the omnetpp.ini and usecase1.ned file. These contains the parameters and network used respectively.

## B Simulation Parameters

The parameters used in the simulation usecase will be listed in the following Apendix.

### A. Usecase Schedule Parameters

The entire uchedule implemented in the usecase.

```xml
<?xml version="1.0" ?>
<schedule>
<defaultcycle>10000us</defaultcycle>
  <host name="Camera">
    <cycle>10000us</cycle>
    <entry>
      <!-- us -->
      <start>10us</start>
      <queue>1</queue>
      <dest>00-00-00-00-00-01</dest> /* Camera TO aggregator*/
      <!-- byte -->
      <size>1500B</size>
      <flowId>1</flowId>
    </entry>
  </host>
  <host name="Aggregator">
  <cycle>10000us</cycle>
  <entry>
      <!-- us -->
      <start>300us</start>
      <queue>1</queue>
      <dest>00-00-00-00-00-02</dest>/*Aggregator TO Remote Control*/
      <!-- byte -->
      <size>1500B</size>
      <flowId>2</flowId>
    </entry>
  </host>
    <host name="RemoteControl">
    <cycle>10000us</cycle>
    <entry>
      <!-- us -->
      <start>2500us</start>
      <queue>2</queue>
      <dest>00-00-00-00-00-03</dest>/*Remote Control TO Actuator*/
      <!-- byte -->
      <size>1500B</size>
      <flowId>3</flowId>
    </entry>
  </host>
  <switch name="Bridge1">
  <cycle>10000us</cycle>
        <port id="0"> /* Camera TO aggregator*/
        <entry>
          <length>0us</length>
          <bitvector>00000000</bitvector>
                </entry>
                <entry>
          <length>300us</length>
          <bitvector>00000001</bitvector>
                </entry>
                <entry>
                   <length>9700us</length>
          <bitvector>00000000</bitvector>
                </entry>
      </port>
    <port id="1">/*Aggregator TO Remote Control*/
        <entry>
          <length>300us</length>
          <bitvector>00000000</bitvector>
                </entry>
        <entry>
          <length>300us</length>
          <bitvector>00000001</bitvector>
        </entry>
        <entry>
          <length>9400us</length>
          <bitvector>00000000</bitvector>
        </entry>
      </port>
    <port id="3">/*Remote Control TO Actuator*/
                <entry>
                    <length>3400us</length>
                    <bitvector>00000000</bitvector>
                </entry>
                <entry>
                    <length>6600us</length>
                    <bitvector>00000100</bitvector>
                </entry>
    </port>
  </switch>
  <switch name="Bridge2">
  <cycle>10000us</cycle>
        <port id="0">/*Aggregator TO Remote Control */
                <entry>
                    <length>1700us</length>
                    <bitvector>00000000</bitvector>
                </entry>
        <entry>
                    <length>300us</length>
                    <bitvector>00000001</bitvector>
                </entry>
                <entry>
                    <length>8000us</length>
                    <bitvector>00000000</bitvector>
                </entry>
        </port>
    <port id="1"> /*Remote Control TO Actuator */
                <entry>
                    <length>2500us</length>
                    <bitvector>00000000</bitvector>
                </entry>
        <entry>
                    <length>300us</length>
                    <bitvector>00000100</bitvector>
                </entry>
                <entry>
                    <length>7200us</length>
                    <bitvector>00000000</bitvector>
                </entry>
        </port>
    </switch>
</schedule>
```

Listing B.1: Schedule Parameters Full Version

## B. Mac-Addressing

The full version of the MAC-Addressing used in the usecase.

```
<filteringDatabases>
 <filteringDatabase id="Bridge1">
     <static>
         <forward>
           <!-- Forward packets addressed to Aggregator -->
           <individualAddress macAddress="00-00-00-00-00-01" port="0" />
           <!-- Forward packets addressed to Camera -->
           <individualAddress macAddress="00-00-00-00-00-04" port="2" />
           <!-- Forward packets addressed to Actuator -->
           <individualAddress macAddress="00-00-00-00-00-03" port="3" />
           <!-- Forward packets addressed to RemoteControl to FiveGNode1
   ↪ -->
           <individualAddress macAddress="00-00-00-00-00-02" port="1" />
         </forward>
     </static>
 </filteringDatabase>
 <filteringDatabase id="Bridge2">
     <static>
         <forward>
           <!-- Forward packets addressed to RemoteControl -->
           <individualAddress macAddress="00-00-00-00-00-02" port="0" />
           <!-- Forward packets addressed to Aggregator to FiveGNode2 -->
```

```
            <individualAddress macAddress="00-00-00-00-00-01" port="1" />
            <!-- Forward packets addressed to Camera to FiveGNode2-->
            <individualAddress macAddress="00-00-00-00-00-03" port="1" />
            <!-- Forward packets addressed to Actuator to FiveGNode2-->
            <individualAddress macAddress="00-00-00-00-00-04" port="1" />
    ↪
          </forward>
      </static>
  </filteringDatabase>
</filteringDatabases>
```

Listing B.2: Routing Parameters Full Version