The Multi-connection Tactile Internet Protocol

UNIVERSIDAD DE MÁLAGA

Tesis Doctoral

Delia Rico Marchena

Programa de Doctorado en Tecnologías Informáticas
Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

Julio 2023

AUTORA: Delia Rico Marchena

https://orcid.org/0000-0002-4310-4729

EDITA: Publicaciones y Divulgación Científica. Universidad de Málaga

The Multi-connection Tactile Internet Protocol

Thesis Autor
Delia Rico Marchena

Thesis Supervisor
Pedro Merino Gómez

Programa de Doctorado en Tecnologías Informáticas
Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

Julio 2023

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD DE LA TESIS PRESENTADA PARA OBTENER EL TÍTULO DE DOCTOR

D./Dña DELIA RICO MARCHENA
Estudiante del programa de doctorado TECNOLOGÍAS INFORMÁTICAS de la Universidad de Málaga, autor/a de la tesis, presentada para la obtención del título de doctor por la Universidad de Málaga, titulada: THE MULTI-CONNECTION TACTILE INTERNET PROTOCOL

Realizada bajo la tutorización de PEDRO MERINO GÓMEZ y dirección de PEDRO MERINO GÓMEZ (si tuviera varios directores deberá hacer constar el nombre de todos)

DECLARO QUE:

La tesis presentada es una obra original que no infringe los derechos de propiedad intelectual ni los derechos de propiedad industrial u otros, conforme al ordenamiento jurídico vigente (Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia), modificado por la Ley 2/2019, de 1 de marzo.

Igualmente asumo, ante a la Universidad de Málaga y ante cualquier otra instancia, la responsabilidad que pudiera derivarse en caso de plagio de contenidos en la tesis presentada, conforme al ordenamiento jurídico vigente.

En Málaga, a 19 de JUNIO de 2023

Fdo.: Doctorando/a	Fdo.: Tutor/a
Fdo.: Director/es de tesis	

Edificio Pabellón de Gobierno. Campus El Ejido. 29071
Tel.: 952 13 10 28 / 952 13 14 61 / 952 13 71 10
E-mail: doctorado@uma.es

Por la presente, el Dr. Pedro Merino Gómez profesor del Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga, CERTIFICA:

Que Delia Rico Marchena ha realizado en el Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga bajo su dirección, el trabajo de investigación correspondiente a su TESIS DOCTORAL titulada:

"The Multi-Connection Tactile Internet Protocol"

En dicho trabajo se han propuesto aportaciones originales en el campo de protocolos de comunicación para aplicaciones críticas. Los resultados expuestos han dado lugar a las siguientes publicaciones en revistas y aportaciones a congresos:

1. D. Rico and P. Merino, "A Survey of End-to-End Solutions for Reliable Low-Latency Communications in 5G Networks" in *IEEE Access*, vol. 8, pp. 192808-192834, 2020, doi: 10.1109/ACCESS.2020.3032726

2. D. Rico, M.M. Gallardo, and P. Merino. "Modeling and verification of the Multi-connection Tactile Internet Protocol" in *Proceedings of the 17th ACM Symposium on QoS and Security for Wireless and Mobile Networks (Q2SWinet '21)*. Association for Computing Machinery, New York, USA, 2021, 105–114, doi: 10.1145/3479242.3487328

3. A. Díaz Zayas, D. Rico, B. García, and P. Merino. "A Coordination Framework for Experimentation in 5G Testbeds: URLLC as Use Case" in *Proceedings of the 17th ACM International Symposium on Mobility Management and Wireless Access (MobiWac '19)*. Association for Computing Machinery, New York, NY, USA, 71–79, doi: 10.1145/3345770.3356742

4. D. Rico, K.-J. Grinemmo, A. Brunstrom and P. Merino, "Implementation and evaluation of the Multi-connection Tactile Internet Protocol and API" in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, Budapest, Hungary, 2022, pp. 1-6, doi: 10.1109/NOMS54207.2022.9789842

5. D. Rico, K.-J. Grinemmo, A. Brunstrom and P. Merino, "Performance Analysis of The Multi-connection Tactile Internet Protocol over 5G" in *Journal of Network and Systems Management*, doi: 10.1007/s10922-023-09737-0

6. A. Díaz Zayas, D. Rico, B. García, and P. Merino. "Marco de Trabajo para la Coordinación de Testbeds 5G: URLLC como Caso de Uso" in XIV Jornadas de Ingeniería Telemática (JITEL 2019), Zaragoza *(España)*, doi: 10.26754/uz.978-84-09-21112-8

Estas publicaciones en coautoría que avalan la tesis no han sido utilizadas en tesis anteriores. Por todo ello, consideran que esta Tesis es apta para su presentación al Tribunal que ha de juzgarla. Y para que conste a efectos de lo establecido, AUTORIZA la presentación de esta Tesis en la Universidad de Málaga.

En Málaga, a 19 de Junio de 2023.

Fdo.: Dr. Pedro Merino Gómez

To Grandma Conchi.

Acknowledgments

As I was reflecting on this acknowledgments section, I was struck by the fact that I have dedicated nearly 20% of my life to working on this thesis. During this time, I have had the privilege of receiving the help and support of so many wonderful people, without whom I would not have made it this far.

Firstly, I would like to thank my supervisor, Pedro Merino, for providing me with the opportunity to pursue this career path and for his guidance over the years, as well as Mª del Mar Gallardo for her help in every occasion we have worked together. Additionally, I want to express my deep gratitude to all my colleagues at ITIS. From those who were present from the beginning to those who came at the end, you have been an invaluable support both inside and outside of work.

I cannot overstate the importance of my stay at Sweden for this thesis. Anna Brunström and Karl-Johan Grinnemo, your support during my stay in Karlstad University and even after I left has meant a lot to me. I truly hope that we have the opportunity to continue collaborating in the future. To all my friends in Sweden, thank you for making me feel at home. You have not just been a chapter in my life, but have also become an integral part of it.

Finally, to all my friends and family, your love and friendship has always meant much more than I normally express with words. Alberto, Inma and Gema, thank you for how you are and how you make me be. Alicia, part of this thesis belongs to you, I believe you can already say that you are a bit of an expert in telecommunications.

This work has been supported by the Ministry of Science, Innovation and Universities of Spain under grant agreement FPU17/04292.

Abstract

Tactile Internet refers to the transmission of touch (forces, vibrations, or motions) and the real-time control of applications like the remote control of industrial machines, drones or vehicles. These applications require low levels of latency combined with reliability, since any deviation could cause catastrophic outcomes. The evolution of the network infrastructure is essential to support such advanced communications. With the arrival of 5G networks, multiple solutions have been developed to enhance latency and reliability. Examples of that are the use of multiple data paths (3GPP and non-3GPP), virtualization, programmable networks, etc. Tactile Internet communications require protocols that can exploit the benefits of such novel networks and, at the same time, can adapt to the specific requirements of each application.

In this context, this thesis presents the Multi-connection Tactile Internet Protocol (MTIP), a transport protocol for the Tactile Internet that uses context awareness and multiple paths to flexibly improve communication. MTIP creates a link between a remote controller and a device constituted by multiple end-to-end connections (sublinks) between interfaces of multi-homed endpoints. MTIP uses application preferences and measurements of network state to perform an intelligent and dynamic selection of the sublinks to send data.

This thesis follows an incremental approach to the development of the Multi-connection Tactile Internet Protocol. First, MTIP is modeled to verify its operation. Two initial PROMELA models abstract the operation of the protocol. The models allow validation of the correctness of MTIP data exchange using the SPIN model tool. Then, the validation is complemented using the UPPAAL Statistical Model Checking (SMC) capability to evaluate the performance of MTIP's use of multiple sublinks.

After formal verification, the first implementation of MTIP was developed in C/C++ and the Linux OS in order to check the behavior of the protocol in the network. The first evaluation consists of a simulated multipath environment between two endpoints. In this evaluation, this thesis assesses the impact of different configurations of MTIP decision-making in selecting paths and, ultimately, in the performance of the protocol. Then,

a second evaluation was performed on a real 5G testbed in UMA. In this scenario, a real industrial robot is controlled using MTIP over multiple 4G and 5G connections. The evaluations confirm MTIP's ability to intelligently select network paths in diverse scenarios, showing how MTIP can be configured to offer reliable, low-latency transport services at the cost of sending some redundant packets in an actual application.

Resumen

Internet Táctil se refiere a la transmisión del tacto (fuerzas, vibraciones o movimientos) y el control en tiempo real de aplicaciones como el control remoto de máquinas industriales, drones o vehículos. Estas aplicaciones requieren bajos niveles de latencia combinados con fiabilidad, ya que cualquier desviación podría provocar consecuencias sumamente perjudiciales. Con la llegada de las redes 5G, se han desarrollado múltiples soluciones para soportar este tipo de comunicaciones avanzadas. Ejemplos de ello son el uso de la multiconectividad, la virtualización o las redes programables. Las comunicaciones del Internet Táctil requieren protocolos que puedan explotar las ventajas de estas nuevas redes y, al mismo tiempo, adaptarse a los requisitos específicos de cada caso de uso.

En este contexto, esta tesis presenta el Protocolo de Multiconectividad para el Internet Táctil (MTIP), un protocolo de transporte para el Internet Táctil que utiliza información de contexto y múltiples caminos para mejorar de forma flexible la comunicación. MTIP crea un enlace entre un controlador remoto y un dispositivo utilizando múltiples conexiones extremo a extremo llamadas subenlaces. MTIP utiliza las preferencias de las aplicaciones y mediciones sobre el estado de la red para realizar una selección inteligente y dinámica de los subenlaces utilizados para enviar datos.

La tesis sigue un enfoque incremental en el desarrollo de MTIP, comenzando con el modelado del protocolo para verificar su funcionamiento. Se desarrollan dos modelos PROMELA iniciales que abstraen el funcionamiento de MTIP y permiten la validación de la corrección del intercambio de datos utilizando la herramienta SPIN. A continuación, la validación se complementa utilizando la herramienta de modelado UPPAAL para analizar el impacto del uso de múltiples subenlaces en MTIP.

Tras la verificación formal, se ha desarrollado una primera implementación de MTIP en C/C++ y el sistema operativo Linux para examinar el comportamiento del protocolo en diferentes topologías de red. La primera evaluación consiste en un entorno de multiconectividad simulado entre dos puntos finales. En esta evaluación, se estudia el impacto de diferentes configuraciones de MTIP en la selección de subenlaces y, en última instancia, en el rendimiento del protocolo. A continuación, se realizó una segunda evalu-

ación en un testbed 5G real en la Universidad de Málaga (UMA). En este escenario, un robot industrial real se controla mediante MTIP a través de múltiples conexiones 4G y 5G. Las evaluaciones confirman la capacidad de MTIP para seleccionar de manera inteligente diferentes subenlaces en diversos escenarios, mostrando cómo MTIP puede configurarse para ofrecer servicios de transporte fiables y de baja latencia al coste de enviar paquetes redundantes.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

CONTENTS

Summary: This chapter serves as an introduction to the thesis work, outlining the motivation behind the research and providing an overview of the research outline.

1.1 Motivation

1.1.1 The Tactile Internet

The IEEE Tactile Internet (TI) Standards Working Group (WG) defines the Tactile Internet as "*a network for remotely accessing, perceiving, manipulating, or controlling real or virtual objects or processes in perceived real time by humans or machines*" [1]. Tactile Internet aims to push the boundaries of wireless communications for remote physical interaction, networked control of highly dynamic processes, and the communication of touch experiences [2, 3]. A Tactile Internet interaction is composed of three main components as depicted in Figure 1.1, i) a controller side, responsible for transmitting commands and processing the feedback; ii) one or multiple TI devices that act as the clients of the remote control information; and iii) the network that interconnects them.

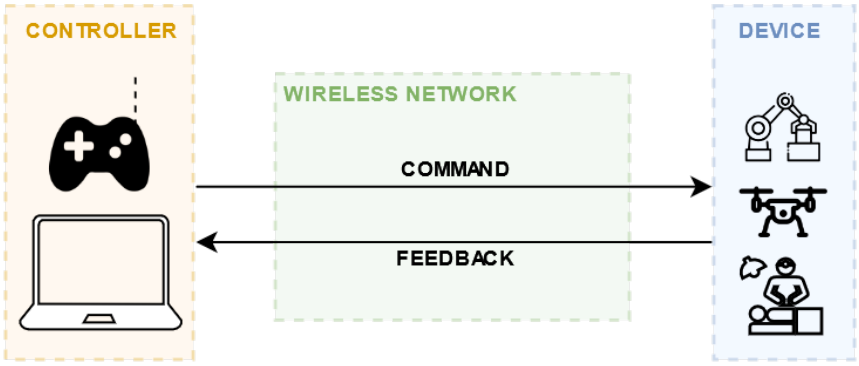


Figure 1.1: Main components of a Tactile Internet interaction

The Tactile Internet has numerous use cases and applications, some of the most representative ones are the remote operation of industrial machinery, the remote control of drones and cars, the remote manipulation of surgical devices, and immersive, virtual, or mixed reality. Remote operation of industrial machinery or *factory automation* is a high-reliability, low-latency, and low-jitter use case [4] that has traditionally relied on wired networks. However, it is increasingly being directed towards wireless and cellular networks for increased deployment flexibility, reduced maintenance costs, and improved long-term reliability through initiatives such as time-sensitive networking [5, 6, 7]. Autonomous connected car communication [8] and the remote control of drones (unmmanned aerial vehicles) [9] require extremely high reliability to prevent misinterpreted control messages, low latency and seamless robust handover to maintain constant connectivity and access to

information about other vehicles to enhance overall performance. Remote surgery includes complex life-saving procedures in health emergencies [10], where reliable communication is required to prevent catastrophic outcomes. Virtual, mixed and immersive reality are technologies that aim to enhance human interaction in virtual and hybrid environments, demanding a seamless connection to provide the user with an immersive interactive experience that can be combined with any of the previously presented use cases.

These TI use cases and their applications are mission-critical, as any errors can have serious consequences for people's lives and valuable assets. This critically translates into the requirements that the underlying network must support. To ensure the support, the IEEE TI WG, as well as the 3GPP through Cyber-Physical Systems [11] and service requirements for the 5G systems [12], have established specific requirements for the Tactile Internet in terms of Key Performance Indicators (KPI) like latency and reliability. The general Tactile Internet requirements demand latencies ranging from below 10 ms to nearly 1 ms, and reliability levels typically reaching close to 99.999%. However, each use case has its own specific requirements and considerations. For instance, the remote operation of industrial machinery may require low levels of jitter (around 1-100us, depending on the scenario), the remote surgery prioritizes availability, and the remote control of drones and cars may need larger service area dimensions (several kilometers). Table 1.1 offers a better understanding of the use cases and the KPI target values extracted from the references [11, 12, 13]. The first column present some of the most representative TI use cases, whereas cloums two and three display their requirements in the two main KPIs of TI, latency and reliability. Finally, the last column presents additional configurations for each use case.

To support such use cases and requirements, the Tactile Internet must rely on a reliable and low-latency network. This network often must offer wireless links due to the mobile nature of the devices involved. As a result, the Tactile Internet is highly tied with current advances on cellular networks, particularly the 5G network.

1.1.2 5G Networks

The 5th Generation (5G) of mobile networks is designed to provide faster, more reliable high-coverage cellular communications [14]. 5G presents an evolutionary step from previous cellular architectures, offering an upgraded architecture with advanced technologies. Such technologies allow for an increased performance in terms of numerous KPIs, like reliability and latency, that could benefit novel use cases such as the ones presented in Tactile Internet. In the next subsections, we describe the principal components of 5G.

Table 1.1: Tactile Internet use cases, KPIs and considerations

TI use case	**Latency**	**Reliability**	**Other considerations**[1]
Industrial Teleoperation	1-10ms (highly dynamic) 1-100ms (dynamic)	99.99%-99.9999%	**PS:** Small (40-250 bytes) **DR:** Low (1-10 Mbps) **SA:** Small (Few hundred meters) **OA:** Low Jitter (1-100us)
Remote Driving	0.5-2ms (life-critical) 5-10ms (dynamic)	99.9%-99.999%	**PS:** Medium (50-1000 bytes) **DR:** Medium (1-25 Mbps) **SA:** High (Some Kms) **OA:** Wireless links
Drone Control	0.5-2ms (kinesthetic) 10ms (tactile)	99.9%-99.999%	**PS:** Small (40-250 bytes) **DR:** Low (1-10 Mbps) **SA:** High (Some Kms) **OA:** Wireless links
Remote Surgery	5-10ms	99.9999%	**PS:** Medium (50-1000 bytes) **DR:** Low (∼1 Mbps) **SA:** High (Some Kms) **OA:** High availability (99.999%)
Virtual Reality	5-10ms	99.99%	**PS:** High (800-1500 bytes) **DR:** High (30-1000 Mbps) **SA:** Variable

[1] Packet Size (PS), Data Rate (DR), Service Area (SA) and Other Aspects (OA)

1.1.2.1 5G Architecture

As in any of the previous cellular architectures, 5G can be separated into two main components: the Radio Access Network (RAN) and the Core Network (CN) [15]. The RAN is responsible for connecting the user equipment (UE) to the 5G network. It is composed of base stations or gNodeBs (gNBs) which transmit information from the UE to the CN and vice versa. The CN is responsible for managing the overall network, processing user data, and providing different network services. It is constituted by multiple subcomponents that conjointly work to offer the 5G service, like the Access and Mobility Management Function (AMF), in charge of the management of registration and mobility, and the User Plane Function (UPF), which routes and forwards user data to the data network. The remaining components, as well as their connections between them and the RAN is depicted in Figure 1.2.

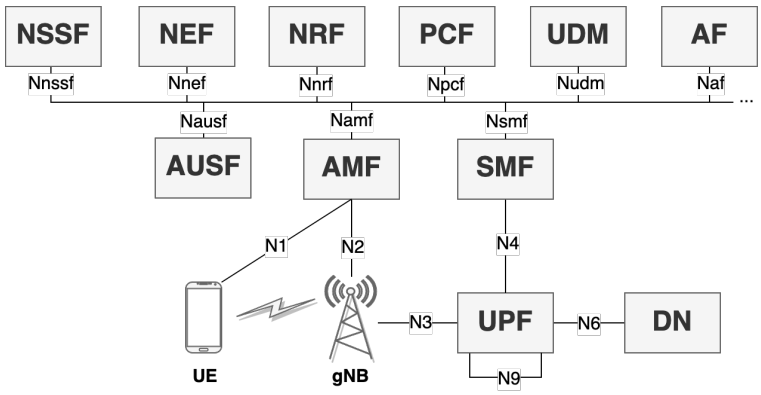


Figure 1.2: 5G system architecture (source [15])

This architecture was initially presented with two deployment options, 5G NSA (Non-Stand Alone) and 5G SA (Stand Alone), as illustrated in Figure 1.3. In a 5G NSA deployment, the 5G RAN is built on top of existing 4G infrastructure. It leverages the existing 4G core network, but uses 5G base stations with the support of 4G base stations to provide 5G connectivity to end-user devices. On the other hand, in a 5G SA deployment the core network and RAN are fully independent and do not rely on any existing 4G infrastructure. This means that all elements of the 5G network are novel components to support 5G technology. The 5G SA deployment is the more complex, comprehensive 5G implementation which offers the full range of 5G capabilities and provides higher levels of performance and reliability for critical applications.

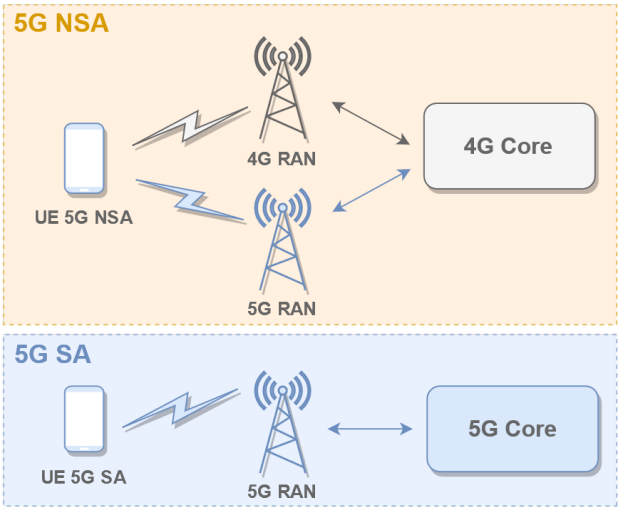


Figure 1.3: 5G NSA and SA architectures

1.1.2.2 5G Categories of Services

Due to its broad spectrum, 5G is divided into three categories of services, Enhanced Mobile Broadband (eMBB), Massive Machine-Type Communications (mMTC) and Ultra-Reliable Low-Latency Communcations (URLLC) which are depicted in Figure 1.4.

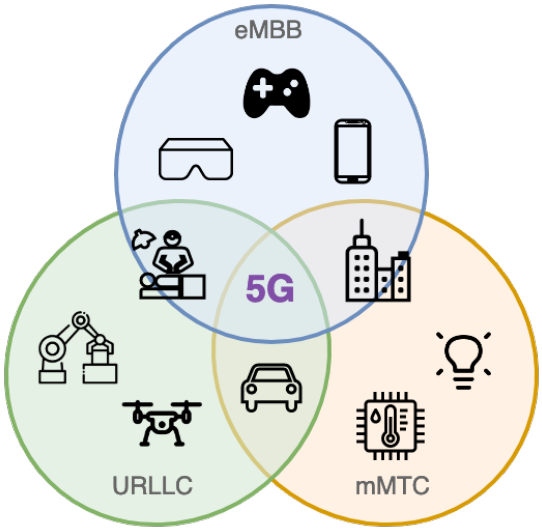


Figure 1.4: 5G categories of services and applications

eMBB is set to improve the speed and capacity of mobile networks, allowing users to download and upload data at higher rates and with lower latency. This makes it ideal for applications that require high-bandwidth connectivity, such as streaming video, online gaming, and virtual reality. mMTC is typically associated with the Internet of Things (IoT), as it enables devices such as sensors, actuators, and other low-power devices to communicate with each other and with the wider network. mMTC is designed to be highly scalable and efficient, allowing it to support a large number of devices with minimal impact on network resources. Ultra-Reliable Low-Latency Communications involve providing high-reliability and low-latency communication for mission-critical applications. URLLC is designed to support use cases that require real-time communication and cannot tolerate any delays or errors. Use cases with similar requirements as those of the Tactile Internet. In order to support each category of service, 5G presents different enabling technologies. The most relevant to the scope of URLLC and, therefore, to TI are presented in next subsection.

1.1.2.3 5G Technological Enablers

5G is supported by different enabling technologies in order to meet the needs of its wide range of service. Three of the most relevant end-to-end solutions for high performance communications and flexibility are Edge computing (EDGE), Software-Defined Networking (SDN) and Network Function Virtualization (NFV) [16]. Figure 1.5 displays how they interconnect in a 5G architecture.

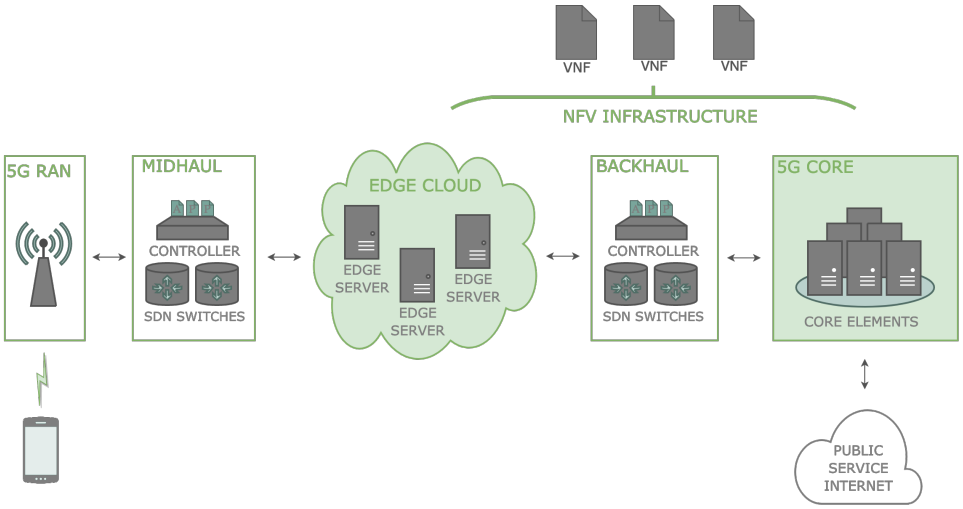


Figure 1.5: Interconnection of technological enablers in the 5G architecture

Edge computing or Multi-Access Edge Computing involves bringing certain network functions in closer proximity to the end-user, enabling the provision of services locally. This allows eluding avoidable transmissions of data that would impact the latency and affect performance. Software-defined networking consists of creating a decoupled architecture that divides the control and data planes. By doing so, SDN enables intelligent routing, adaptability, programmability, and an enhanced management of traffic. Network Function Virtualization aims to virtualize network functions. This is achieved by separating software functionalities from physical hardware to enhance aspects such as flexibility, scalability, latency, reliability, and capacity. One of the most significant outcomes of this virtualization is network slicing, which involves dividing the physical network into multiple logical networks known as slices. The slices are independently configured to optimally serve a particular type of use case in a flexible way.

These and further enabling technologies [17] make the 5G architecture a complex system that requires careful planning and implementation to provide reliable and high-performance network connectivity to end-users. One of the key pillars in charge of managing the data transmission and exposing the benefits of the underlying enabling technologies are the network protocols.

However, the traditional transport solutions such as TCP or UDP expose inefficiently the capabilities of novel networks, since TCP includes retransmission mechanisms that might interfere in the data exchange and, on the contrary, UDP lacks of mechanisms to support novel applications such as TI. Therefore, novel networks and applications, such as 5G and TI, need protocols that address the requirements without interfering with the advancements on the underlying layers. Next Section 1.1.3 deepens into this matter.

1.1.3 End-to-end Protocols for Tactile Internet

The Internet Protocol (IP), the User Datagram Protocol (UDP), and the Transmission Control Protocol (TCP) have been the dominant protocols on the Internet and the transport layer since their standardization in the 1980s. This traditional TCP/IP stack (shown in Figure 1.6) has been sufficient for most of the history of the Internet; however, the increasing demands of novel applications like TI present new challenges for the TCP/IP stack.

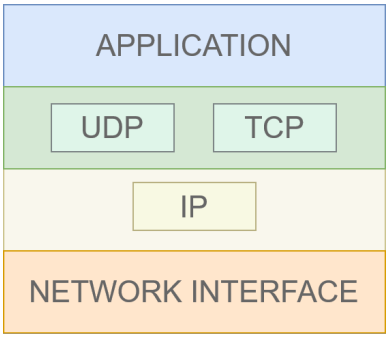


Figure 1.6: The TCP/IP protocol stack

It is clear that traditional TCP and UDP protocols cannot support the TI requirements. For instance, TCP provides 100% reliability, which often leads to significant latency due to poor path quality between endpoints. On the other hand, UDP does not add any specific quality to the underlying path and thus does not offer any reliability improvement. Moreover, to support the operation of such critical remote control applications, various end-to-end solutions have been developed lately for wireless and cellular networks, like the advancements presented in Section 1.1.2 and further novel solutions that include exposing network awareness, using multi-homed devices and creating private network slices, among others listed in [18]. Traditional transport protocols such as TCP and UDP are unable to fully utilize recent advances and impose limitations on novel applications [19].

Protocols such as MPTCP [20] and QUIC [21] are the first steps in the development of a new transport layer that tries to overcome the limitations of TCP and UDP. Multipath TCP (MPTCP) is a variant of TCP that allows

creating TCP connections through multiple paths simultaneously to transmit data. Using multiple paths can increase the reliability and throughput of the connection and can provide a backup path in case of failure or congestion. QUIC (Quick UDP Internet Connections) is a transport protocol based on UDP that is designed to provide low-latency and security-enhanced communication over the Internet. QUIC is intended to address some limitations of both UDP, such as the lack of encryption to secure the data transmitted, and TCP, such as connection setup latency and head-of-line blocking. These protocols, as well as their variants, i.e. PR-MPTCP [22], NC-MPTCP [23], MPQUIC [24], etc. are mature solutions to exploit network advancements that address the importance of overcoming network ossification. However, they are designed to transmit large amounts of available data, where control flow and congestion control are generally mandatory. Scenario that does not necessarily follow the characteristics of the mission-critical applications aforementioned.

Historically, real-time transport protocols have been preferred for mission-critical remote control applications due to their superior suitability compared to TCP, UDP, and their variants. This is the case for real-time protocols like SCTP [25], RTP [26] and extensions such as Multipath RTP (MPRTP) [27] which have been typically used for remote control applications [28, 29]. However, it is arguable that the characteristics of remote control data for critical applications differ from the peculiarities of multimedia real-time streams, since the remote control packets are typically small and transmitted at lower data rates, which may allow for the avoidance of flow control and retransmissions to reduce latency. That is why some extensions of these real-time protocols, like Smoothed SCTP [30], and novel protocols, like IRTP [31] or ETP [32], have been developed for the remote control operation use case, among others [33, 34]. Nonetheless, these protocols for remote control have not undergone sufficient evolution in relation to current trends. Most transport protocols used for this purpose do not take advantage of the aforementioned network advancements, particularly one of the key relevant trends of novel protocols: the ability of managing multiple paths (more information on novel trends in [18, 35]). Additionally, there has been a recent interest in exposing transport protocol features to applications for flexible network communication, with solutions like NEAT [36] and TAPS [37]. However, these protocols lack the necessary flexibility to adapt to the specific requirements of each use case, and they come with extra features that might not be needed and could even be counterproductive due to the added complexity and unnecessary header overhead.

In conclusion, Tactile Internet applications need a protocol that can adapt to the specific requirements of each use case while also functioning seamlessly within novel networks without obstructing progress in the underlying layers.

1.2 Research Outline

1.2.1 Research Questions

The main objective of this thesis work is to explore the viability of transport protocols for reliable low-latency communications. For that purpose, the thesis is based on several research questions that are the central focus of the study and provide a framework for the investigation. In this section, we outline the research questions that this thesis seeks to answer.

- **Question 1 (Q1):** Which are the most relevant novel reliable low-latency use cases and what are their specific requirements in terms of KPIs?
- **Question 2 (Q2):** Which are the main end-to-end solutions and enabling technologies in novel cellular networks (such as 5G) to support reliable low-latency communications?
- **Question 3 (Q3):** Are traditional transport protocols ready to support novel reliable low-latency communications?
- **Question 4 (Q4):** How can novel advances enhance a transport protocol to provide a better service for reliable low-latency communications?
- **Question 5 (Q5):** What performance can be achieved with an enhanced transport protocol for reliable low-latency applications?

1.2.2 Contributions

In order to address the research questions, this thesis work has focused on developing a novel transport protocol for the Tactile Internet. The contributions of this thesis build upon and extend the existing literature on the topic, resulting in a set of published works. The relation between the questions, contributions and works is depicted in in Figure 1.7 and explained below.

Contribution 1 (C1): State of the art of end-to-end solutions on low-latency reliable communications.

In order to address Q1, Q2 and Q3, we performed an exhaustive study of the state of the art of end-to-end solutions on low-latency reliable communications. First, we researched the specific requirements for novel reliable low-latency use cases, focusing on the particularities of each application. Then, we identified relevant research areas and enabling technologies, both in the scope of transport protocols and in the scope of wireless network advancements. We performed an exhaustive study on the different novel solutions

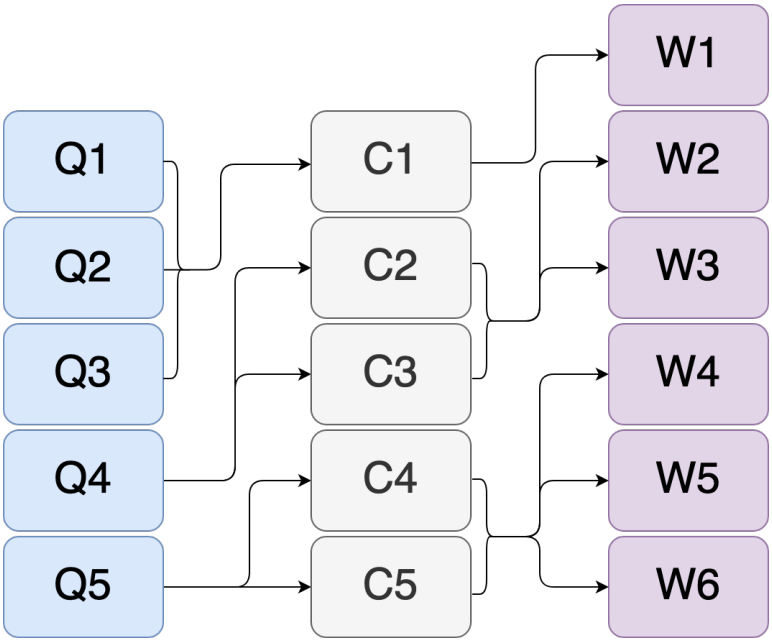


Figure 1.7: Relation between research questions and contributions

for the research areas, identifying the main methods used in each solutions. Finally, we determined promising research topics in terms of protocols like multi-connectivity, context awareness; and in terms of network support, such as edge computing and self-organizing networks. We collected the results of this contribution on the work (W1) *A Survey of End-to-End Solutions for Reliable Low-Latency Communications in 5G Networks.*

Contribution 2 (C2): Design of a transport protocol for low-latency reliable communications (The Multi-Connection Tactile Internet Protocol).

In order to answer Q4, we took our previous knowledge from C1 and designed a transport protocol protocol for low-latency reliable communications: The Multi-Connection Tactile Internet Protocol (MTIP). The protocol is based on the studied relevant enablers, such as multi-connectivity and context awareness. The aim of the protocol is to transport remote control commands of Tactile Internet applications flexibly and maintaining high levels of reliability combined with low latency.

Contribution 3 (C3): Creation and verification of formal models that accurately depict the protocol.

Following the work on C2, we identified the main properties that define the protocol and created formal models to perform an exhaustive verification of its operation. We separate the properties into correctness and performance properties. Correctness properties are used to perform a formal verification of the protocol data transmission, while performance properties take into account the trade-off that exists in multi-connectivity between enhanced performance versus resource waste.

The combination of C2 and C3 resulted in two works, (W2) *Modeling and verification of the Multi-connection Tactile Internet Protocol* and (W3) *Verification of a multi-connectivity protocol for Tactile Internet applications.*

Contribution 4 (C4): Implementation of the protocol and API that exposes the protocol capabilities flexibly for applications.

We developed an implementation of the protocol in a widely used operating system and language, as well as an API that exposes the protocol capabilities flexibly for applications. The protocol is programmed in C++ and works on the Linux operating system. The API offers two versions in C and C++ and offers the application an interface to expose its requirements and flexibly configure the protocol.

Contribution 5 (C5): Evaluation of the performance of the protocol through simulations and real-world experiments.

In order to answer Q5, we took our previous C4 and performed an evaluation of the protocol in a simulated environment and real-world experiments. For the simulated environment, we used the tools Mininet [38] and Netem [39] to create a topology with multiple paths and studied the impact of different configurations of the protocol. For the real-word experiments, we first studied the capabilities of a real 5G testbed at the University of Málaga (UMA) [40] to support remote control communications, and then we created an scenario with multiple cellular paths (such as 4G, and 5G NSA and SA) to deepen into the impact of different configurations of the protocol.

The results of C4 and C5 are the works (W4) *Implementation and evaluation of the Multi-connection Tactile Internet Protocol and API*, (W5) *A Coordination Framework for Experimentation in 5G Testbeds: URLLC as Use Case* and (W6) *Performance Analysis of The Multi-connection Tactile Internet Protocol over 5G.*

1.2.3 Document Structure

This section provides a comprehensive overview of the organization and structure of the document. It outlines the different chapters and appendices

included in the thesis and highlights their objectives and contributions. The document is structured in six main chapters and two appendices.

- **Chapter I - Introduction.** This chapter comprises the motivation and provides the research outline of the thesis.
- **Chapter II - State of the Art.** This chapter presents an overview of end-to-end solutions for reliable low-latency communications in 5G networks.
- **Chapter III - The Multi-connection Tactile Internet Protocol (MTIP)**. This chapter introduces the Multi-connection Tactile Internet Protocol and its architecture, design, and functionality.
- **Chapter IV - Modeling and Verification of MTIP**. This chapter describes the models developed for MTIP in PROMELA and timed automata. It also presents the verification of correctness properties with SPIN and performance properties with UPPAAL.
- **Chapter V - Implementation and Evaluation of MTIP**. This chapter discusses the implementation details of MTIP and presents the evaluations performed in both a simulated environment and a real 5G platform.
- **Chapter VI - Conclusion and Future Work**. This chapter summarizes the thesis and outlines directions for future research.
- **Appendix A - *Resumen en Español***. This appendix provides a summary of the thesis in Spanish, following the regulations of University of Malaga.
- **Appendix B - MTIP Implementation Details**. This appendix includes detailed information on the implementation of MTIP and the API.

This structured approach aims to provide a clear and concise understanding of the thesis to the reader and aid in an efficient navigation through the document.

Chapter 2

State of the Art: End-to-End Solutions for Reliable Low-Latency Communications in 5G Networks

CONTENTS

SUMMARY:

This chapter summarizes end-to-end solutions that enhance reliable, low-latency communications. It evaluates different enabling technologies and their impact on key performance indicators. Moreover, it analyzes common methods to improve the performance and concludes with a discussion on promising research topics.

2.1 Enabling Technologies and Solutions

Works such as that of Elbamby et al. [41] or that of Parvez et al. [42] have helped in the selection of the enabling technologies for low latency and high reliability. Elbamby et al. [41] study the importance of reliability and latency in virtual reality and present multi-connectivity, edge computing and multicasting as enabling solutions; whereas Parvez et al., [42] study some of the increasingly important novel technologies for 5G, such as software-defined networking, network function virtualization and edge computing. Moreover, it also presents network technological enablers with the potential to complement these ones, such as information-centric networking (ICN). Based on these surveys, the enabling technologies and solutions selected for this work are shown jointly in Figure 2.1.

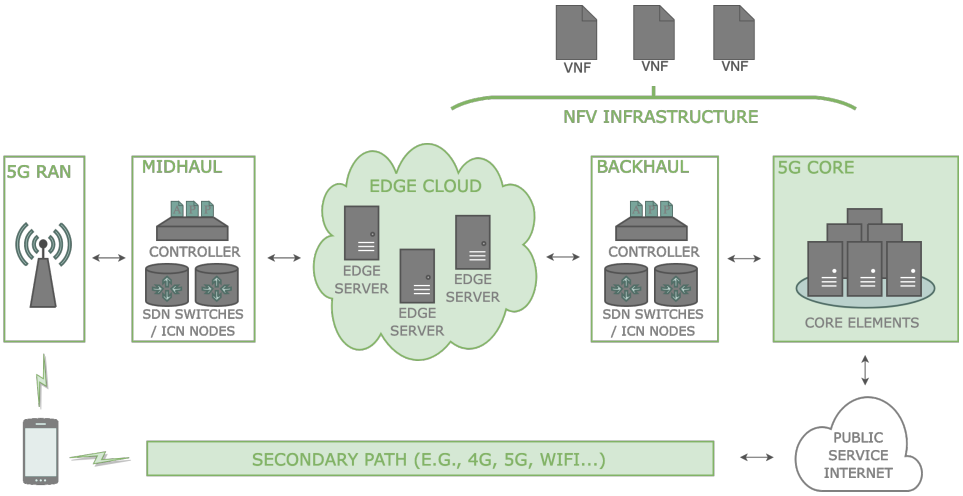


Figure 2.1: Enabling technologies in 5G networks

2.1.1 End-to-end Protocols

Novel communication protocols can be classified into three main categories.

- **Single-path protocols:** Single-path protocols are the most commonly used in current Internet. An inefficient protocol limits the possibility of taking advantage of network capabilities, so enhancements on single-path protocols are the natural evolution of the existing protocols.

- **Multipath protocols:** As Qadir et al. [43] noted, the Internet's future is inherently multipath. Multihoming capabilities, path/interface/network diversity, data centre enhancements and wireless commu-

nications are leading networking into the use of multi-access connectivity. The benefits of multiple connectivity include better reliability, network offloading, improved availability, etc.

- **Multicast protocols:** Poularakis et al. [44] and Araniti et al. [45] reflect on the growth of mobile multicast applications and present multicasting as a key opportunity in future 5G networks. Sending the same copy of information to multiple receivers at a given moment of time can provide lower latencies, higher scalability and network offloading.

2.1.2 Network Support

The network technologies selected that support application/protocol operation to reduce latency and to increase reliability are the following:

- **Edge computing (EDGE):** Edge computing consists of moving the cloud and some network functions closer to the user to provide services locally and consequently improve performance, such as a reduced latency.
- **Software-defined networking (SDN):** SDN entails creating a decoupled architecture that splits the control and data planes. SDN allows intelligent routing, flexibility, programmability and facilitates virtualization [46].
- **Network function virtualization (NFV):** NFV aims to virtualize network functionalities. NFV decouples software functionalities from physical equipment to offer better flexibility, scalability, latency, reliability, capacity, etc.
- **Information-centric networking (ICN):** ICN is an approach to redesign the current Internet infrastructure to leave behind the point-to-point paradigm and embrace data replication, content distribution, naming schemes and catching [47, 48].

2.1.3 Application Programming Interface (API)

The concept of the API is present in most of the enabling technologies. Application programming interfaces are intermediaries that facilitate application layers to manage information of the layers below in order to work flexibly according to the needs. Taking advantage of information outside of a layer's scope of work and managing these functionalities could result in a better service that would benefit both the applications, with performance improvements, and the network, with reduced overload. The traditional socket API is too low-level, simple and inflexible [49, 50, 51] and has been questioned for a long time. This has created an arising interest in the topic.

2.2 Common Methods and Techniques

In this section, we introduce a taxonomy that categorizes common solutions adopted by the enabling technologies to meet the desired requirements in terms of latency or reliability. We anticipate that the taxonomy depicted in Figure 2.2 will be used to classify future papers as well. Section 2.4 includes references that discuss and demonstrate these methods in their contributions.

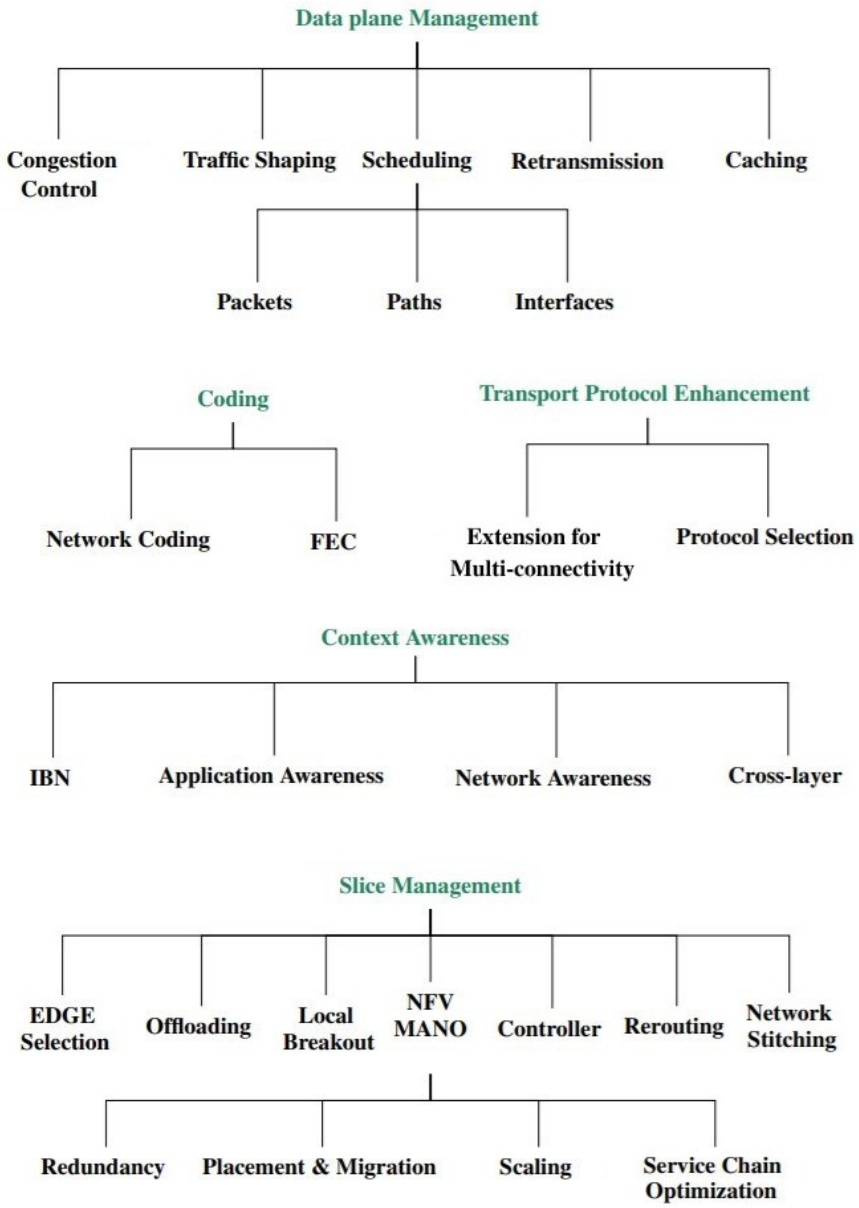


Figure 2.2: Common methods and techniques for latency and reliability

2.2.1 Data Plane Management

There are several ways to address latency and reliability in end-to-end protocols such as TCP, UDP or SCTP over 5G networks. One immediate approach is to modify the fundamental mechanisms for managing the data flow in these protocols. Such modifications include: the vast literature on **congestion control** mechanisms in protocols like TCP for wireless networks, changes in **retransmission** algorithms for early confirmation or deletion of unnecessary ACKs (possible in the lower radio level), or intelligent **traffic shaping** to prioritize certain types of traffic.

Another popular method in the revised literature is using intelligent packet delivery scheduling methods with three different alternatives: a) splitting packets into tasks for one connection (**packet scheduling**), b) using multiple connections on a single interface and selecting one or more to send packets (**path scheduling**), and c) using a multi-homed device with multiple interfaces and selecting interfaces for sending packets (**interface scheduling**), possibly duplicating the packets.

The last relevant method in data plane management is **caching**. Caching involves storing frequently accessed data content and routing popular requests to reduce retrieval delay. This method is commonly applied in the context of Information-Centric Networking (ICN) and has been shown to significantly improve end-to-end latency.

2.2.2 Transport Protocol Enhancement

Two techniques being explored are a) extending a well-known single-path protocol to support multi-connectivity, improving reliability and throughput and further KPIs (we refer to this category as **extension for multi-connectivity**), and using a flexible protocol stack that can select different protocols based on network state, parameters, or requirements (we refer to this category as **protocol selection**). This latter approach is often combined with context awareness information that could help reduce the latency and enhance the reliability.

2.2.3 Coding

In data transmissions, coding refers to sending information with some modifications in order to enhance communication. This is often done by adding redundancy, allowing data to be decoded at the destination in different ways. This work considers two well-known coding forms: **forward error correction (FEC)** and **network coding**. FEC is an end-to-end technique that detects and corrects errors without retransmissions, improving reliability, throughput, and latency. Network coding allows intermediate nodes to send coded data in multiple packets, allowing the original message to be decoded if enough packets arrive at the destination, according to most variants.

2.2.4 Context Awareness

Utilizing information beyond the scope of a protocol or layer to enhance performance and improve specific key performance indicators like latency and reliability is known as context awareness. The 5G network is expected to be fully context-aware [52], generating interest in research groups such as the Path Awareness Networking Research Group [53].

Context information can be extracted from various sources, including higher or lower layers of the protocol stack. **Application awareness** involves monitoring the information flow from the application layer to improve latency, reliability, throughput, or general performance in the lower layers. **Network awareness** considers network conditions or configurations to make decisions about parameters and usage of certain higher layer protocols or applications. **Cross-layer** approaches refer to the exposure of information between layers and working jointly to fulfill different requirements at execution time. Finally, Intent-based Networking (IBN) is a concept that involves taking into account user preferences and applying logical intelligence to map or translate them into policies that can be applied in the current protocol, network, or operating scope [54].

2.2.5 Slice Management

The concept of network slicing in 5G networks has gained significant attention in recent years. In general, network slicing refers to the logical division of a network to isolate resources in order to maintain a certain level of quality, such as latency and reliability, for specific users and services. Since the 5G network slice is end-to-end, most of the common methods related to slice management are connected to the management of the enabling technologies in the network support category, such as EDGE, SDN, and NFV.

One of the most common methods to reduce latency is the EDGE technology. A proper **EDGE selection** would reduce the distance between end users and thus, result in enhanced latencies. **Local breakout (LBO)** is another promising solution that sends data packets through closer destinations, such as EDGE or local nodes, instead of the central core network, to reduce excessive delay in the core network load. In addition, **offloading** is a network solution based on leveraging the processing or execution of tasks to the network that is highly coupled with EDGE since it allows easy deployment in the closer places of the network.

Network function virtualization management and orchestration (NFV MANO) constitutes another series of common methods and techniques for slice management. Optimized selection of VNF **placement and migration** of services would mean offering a better service with enhanced KPIs. Several methods regarding **service chain optimization**, such as refactoring, pipelining, using parallelism, etc. also result in enhanced com-

munication with better latency, throughput or reliability. Moreover, the nature of NFV allows VNFs to be simultaneously deployed in different parts of a network. This **redundancy** provides better service in terms of reliability and even latency. Finally, resources are allocated to VNFs in terms of CPU cores and memory, among others. A proper dynamic allocation or **scaling** of these resources at runtime would enhance the overall communication and reduce potential overload.

Regarding the role of SDN in the network slice, proper **controller placement** would reduce the latency between SDN nodes while **rerouting** or dynamically changing routing tables would provide sufficient network flexibility so as to adapt to network or application requirements. Finally, **network stitching** or slice stitching is another relevant slice management method. This method modifies the functionality of an existing slice by adding and merging the functions of another slice in order to enhance the overall operations or concrete KPIs such as the reliability and latency.

2.3 Parameters Evaluated

To complete the characterization of the literature, on top of the previous taxonomy, we consider a number of parameters that include key performance indicators and other relevant metrics.

2.3.1 Key Performance Indicators

Key performance indicators are specific network measurements that aid in the monitoring, optimization, and characterization of services. Some well-defined 5G-PPP KPIs have been taken from the 5Genesis project [55, 56] and have been set as goals to assess the various contributions as feasible enablers in subsequent sections. The KPIs under study are the enhancement of latency, the increase in reliability and the improvement of the throughput. While the first two KPIs are crucial in the communications under study, the third one is present in a number of critical communications that share video content

2.3.2 Other Parameters

Quality characteristics beyond KPIs are also considered important in characterizing contributions to critical communication networks. Two such characteristics are partial reliability and heterogeneous networks. Partial reliability refers to the ability to achieve low latency without sacrificing reliability entirely. In some cases, certain data transmissions can tolerate some loss in favor of lower latency. Although partial reliability does not meet all critical communication requirements, it can still meet the demands of certain types

of reliable low-latency communications, making it an interesting feature to consider. The support of heterogeneous networks is another key characteristic to consider. Solutions that can work properly, maintain fairness and adapt to changes over various technologies with diverse characteristics, add significant value to critical communication networks.

2.4 Analysis of the State of the Art

This section presents a range of novel solutions organized by the enabling technologies. To provide a clearer understanding of the contributions, several tables are presented summarizing the main methods used for achieving their respective goals and their main impact on KPIs and other parameters.

2.4.1 End-to-end Protocols

Tables 2.1, 2.2, 2.3 present all the contributions studied on single-path, multipath, and multicast protocols. Recent single-path protocol contributions usually prioritize low latency and high throughput over reliability. In contrast, multipath protocol solutions focus on reliability, but also have solutions for increasing throughput and reducing latency. Finally, multicast protocols also focus on reliability through the use of multiple copies of information, which can also improve throughput in some cases, but latency is not typically a major consideration.

Table 2.1: Comparison of protocol solutions: Single-path protocols

	Main Methods	KPI			Other Parameters	
		Low Latency	High Reliability	High Throughput	Partial Reliability	HetNet Support
TCP/SCTP modifications [57]	DP Mgmt: Retransmission	✓			✓	
ER TCP Pert [58]	DP Mgmt: Retransmission	✓			✓	
TCP-ROME [59]	Scheduling: Paths		✓	✓		✓
TCP VEGAS* [60]	DP Mgmt: Congestion Control	✓		✓	✓	✓
TCP 5G mmWave Networks [61]	Ctxt Awa: Cross-layer DP Mgmt: Retransmission		✓	✓		
Advanced 5G-TCP [62]	DP Mgmt: Congestion Control		✓	✓		✓
TCP BBR [63]	DP Mgmt: Congestion Control	✓		✓	✓	
Gambhava et al. [64]	DP Mgmt: Congestion Control		✓	✓		✓
Zhu et al. [65]	Ctxt Awa: Cross-layer DP Mgmt: Congestion Control	✓		✓		✓
Luo et al. [66]	DP Mgmt: Congestion Control	✓		✓		
ASAP [67]	Ctxt Awa: Cross-layer	✓				
STRP [68]	DP Mgmt: Retransmission	✓			✓	
PrefCast [69]	Scheduling: Packets Ctxt Awa: IBN	✓				
Park et al. [70]	DP Mgmt: Retransmission	✓			✓	
SCReAM [71]	DP Mgmt: Congestion Control	✓		✓	✓	✓
Mittal et al. [72]	Ctxt Awa: Cross-layer	✓		✓		
L4S Architecture [73]	DP Mgmt: Congestion Control	✓		✓		✓

Table 2.2: Comparison of protocol solutions: Multipath protocols

	Main Methods	KPI			Other Parameters	
		Low Latency	High Reliability	High Throughput	Partial Reliability	HetNet Support
LISP-HA [74]	Scheduling: Paths		✓	✓		✓
Yap et al. [75]	Scheduling: Paths Ctxt Awa: Application		✓			✓
Singh et al. [76]	Scheduling: Interfaces		✓	✓		✓
Gonzalez-Muriel et al. [77]	Scheduling: Interfaces		✓	✓		✓
MPTCP [78]	TP Enh.: Extension for MC		✓	✓		
NC-MPTPC [23]	Coding: Network Coding Scheduling: Paths		✓	✓		✓
FMTCP [79]	Coding: Network Coding Scheduling: Paths	✓	✓	✓		✓
Hurtig et al. [80]	Scheduling: Paths	✓		✓		✓
QoS-MPTPC [81]	Scheduling: Packets	✓			✓	
ADMIT [82]	Coding: FEC	✓	✓	✓		✓
PR-MPTCP+ [83]	Scheduling: Paths Ctxt Awa: Network	✓	⋆	✓	⋆	✓
MPFlex [84]	Scheduling: Paths		✓	✓		
Lee et al. [85]	Scheduling: Paths		✓	✓		
Multipath PERT [86]	Scheduling: Paths	✓	✓	✓		✓
Multipath QUIC [87]	TP Enh.: Extension for MC	✓	✓	✓		✓
MPRTP [88]	TP Enh.: Extension for MC	✓	✓	✓		
EMSTP [89]	TP Enh.: Extension for MC Coding: FEC	✓			✓	✓
MPMTP [90]	TP Enh.: Extensions for MC Coding: FEC	✓		✓	✓	✓
HMTP [91]	Coding: FEC		✓	✓		✓
MPMTP-AR [92]	Scheduling: Paths		✓	✓		
m²CMT [93]	Scheduling: Paths		✓	✓		✓
A-CMT [94]	Scheduling: Paths		✓	✓		✓

Table 2.3: Comparison of protocol solutions: Multicast protocols

	Main Methods	KPI			Other Parameters	
		Low Latency	High Reliability	High Throughput	Partial Reliability	HetNet Support
Zhu et al. [95]	Scheduling: Paths		✓	✓		✓
Tsimbalo et al. [96]	Coding: Network Coding		✓			✓
Xiong et al. [97]	Scheduling: Paths		✓	✓		
Chi et al. [98]	Coding: Network Coding DP Mgmt: Retransmission		✓			✓
Roger et al. [99]	Ctxt Awa: Cross-layer	✓				
ECast [100]	DP Mgmt: Retransmission		✓			
Mahajan et al. [101]	Ctxt Awa: Cross-layer DP Mgmt: Congestion Control		✓			✓

2.4.2 Network Support

Tables 2.4, 2.5, 2.6 and 2.7 show the solutions presented for the technologies that provide network assistance in current and future networks. Edge computing solutions typically reduce latency, due to the reduced distance between the endpoints of communication, while SDN, NFV and ICN proposals focus also mostly on decreasing the latency, but consider reliability and throughput more extensively.

Table 2.4: Network support proposals: EDGE

	Main Methods	KPI			Other Parameters	
		Low Latency	High Reliability	High Throughput	Partial Reliability	HetNet Support
Garcia et al. [102, 103]	Ctxt Awa: Cross-layer Slice Mgmt: Local Breakout	✓				
Zhang et al. [104]	Slice Mgmt: Offloading	✓				✓
CASH [105]	Ctxt Awa: Application Ctxt Awa: Network	✓		✓		✓
Lee et al. [106]	Slice Mgmt: Local Breakout	✓				
Cattaneo et al. [107]	Slice Mgmt: Local Breakout	✓				✓
Huang et al. [108]	Ctxt Awa: Network Ctxt Awa: Cross-layer	✓				
Heinonen et al. [109]	Slice Mgmt: EDGE Selection	✓				✓
Schiller et al. [110]	DP Mgmt: Caching	✓		✓		✓
Yang et al. [111]	VNF MANO: Scaling	✓				✓
Cziva et al. [112]	VNF MANO: Plcmt & Migr.	✓				✓
Nunna et al. [113]	Ctxt Awa: Network	✓				
Dutta et al. [114]	Slice Mgmt: Offloading	✓				
Taleb et al. [115]	Slice Mgmt: EDGE Selection VNF MANO: Plcmt & Migr.	✓				
Edgent [116]	Slice Mgmt: Offloading	✓				
Maier et al. [117]	Slice Mgmt: Offloading	✓	✓	✓		✓
Liu et al. [118]	Slice Mgmt: Offloading	✓	✓			✓

Table 2.5: Network support proposals: SDN

	Main Methods	KPI			Other Parameters	
		Low Latency	High Reliability	High Throughput	Partial Reliability	HetNet Support
Pagé et al. [119]	DP Mgmt: Traffic Shaping	✓				
Costa-Requena et al. [120]	DP Mgmt: Traffic Shaping	✓	✓	✓		
Wang et al. [121]	Slice Mgmt: EDGE Selection Slice Mgmt: Offloading	✓	✓			
Lakiotakis et al. [122]	Ctxt Awa: Network Slice Mgmt: Rerouting	✓				
Awobuluyi et al. [46]	Ctxt Awa: Network Slice Mgmt: Rerouting	✓	✓	✓		
Garg et al. [123]	Slice Mgmt: EDGE Selection Slice Mgmt: Offloading	✓	✓			
Han et al. [124]	Slice Mgmt: Controller Placement	✓				
G. Wang et al. [125]	Slice Mgmt: Controller Placement	✓				
Yap et al. [126]	Slice Mgmt: Network stitching Scheduling: Paths	✓	✓	✓		✓
RLMD [127]	Slice Mgmt: Controller Placement	✓	✓			

Table 2.6: Network support proposals: NFV

	Main Methods	KPI			Other Parameters	
		Low Latency	High Reliability	High Throughput	Partial Reliability	HetNet Support
Raza et al. [128, 129]	NFV MANO: SC Optimization	✓	✓			
Qu et al. [130]	NFV MANO: Plcmt & Migr.	✓	✓	✓		
Mekikis et al. [131]	NFV MANO: Scaling	✓	✓	✓		✓
Ding et al. [132]	NFV MANO: Redundancy		✓			
Nascimento et al. [133]	NFV MANO: SC Optimization	✓				
Cho et al. [134]	NFV MANO: Plcmt & Migr.	✓				✓
Sun et al. [135]	NFV MANO: SC Optimization	✓		✓		
Fan et al. [136]	NFV MANO: Redundancy NFV MANO: SC Optimization		✓			
Bekkouche et al. [137]	Slice Mgmt: EDGE Selection Slice Mgmt: Offloading NFV MANO: Plcmt & Migr.	✓			✓	
Valsamas et al. [138]	Slice Mgmt: Network Stitching	✓				✓
Yao et al. [139]	Slice Mgmt: Network Stitching		✓			✓

Table 2.7: Network support proposals: ICN

	Main Methods	KPI			Other Parameters	
		Low Latency	High Reliability	High Throughput	Partial Reliability	HetNet Support
Liang et al. [140]	NFV MANO: Plcmt & Migr. DP Mgmt: Caching	✓				✓
Carofiglio et al. [141, 142, 143]	DP Mgmt: Caching	✓				
Zhang et al. [144]	DP Mgmt: Caching	✓				
Sardara et al. [145]	Ctxt Awa: Cross-layer	✓		✓	✓	✓
Dannewitz et al [146]	DP Mgmt: Caching	✓				✓
Wang et al. [147]	Ctxt Awa: Cross-layer	✓	✓	✓		
Vakilina et al. [148]	DP Mgmt: Congestion control	✓	✓	✓		✓

2.4.3 Application Programming Interface

Table 2.8 presents API contributions. The majority of API contributions help reducing the latency, increasing the throughput and enhancing reliability, sometimes at the cost of another KPIs.

Table 2.8: API proposals[1]

	Main Methods	KPI			Other Parameters	
		Low Latency	High Reliability	High Throughput	Partial Reliability	HetNet Support
Jones et al. [49]	TP Enh.: Protocol Selection Ctxt Awa: IBN	⋆	⋆	⋆	⋆	
Trammell et al. [50]	TP Enh.: Protocol Selection	⋆	⋆	⋆	⋆	⋆
Scharf et al. [149]	Ctxt Awa: Cross-layer		✓	✓		
Hesmans et al. [150]	Scheduling: Paths		✓	✓		✓
Hesmans et al. [151]	Scheduling: Paths		✓	✓		✓
NEAT [152, 153]	Ctxt Awa: IBN Ctxt Awa: Network TP Enh.: Protocol Selection	⋆	⋆	⋆	⋆	⋆
TAPS [154]	TP Enh.: Protocol Selection	⋆	⋆	⋆	⋆	⋆
Nielsen et al. [155]	Scheduling: Paths	✓	✓			✓
Multi-sockets [156]	Ctxt Awa: Cross-layer TP Enh.: Protocol Selection	✓		✓	✓	✓
Msocket [157]	TP Enh.: Protocol Selection	⋆	⋆	⋆	⋆	⋆
Socket Intents [158]	Ctxt Awa: IBN Scheduling: Paths	✓		✓		
Chronos [159]	Ctxt Awa: Cross-layer	✓		✓		
Belay et al. [160, 161]	Ctxt Awa: Cross-layer	✓		✓		
Siddiqui et al. [162]	Ctxt Awa: IBN	⋆	⋆	⋆	⋆	⋆

[1] A ⋆ indicates that the parameter can be addressed at the cost of another parameter

2.5 Evaluation and Challenges

In this evaluation, we follow three classification criteria to report some conclusions: first, we analyse the combination of different enabling technologies; then, we study the KPI concentrations and coverage according to each approach; and finally, we reflect on the different methods used in the literature.

2.5.1 Combination of Enabling Technologies

First, due to the overlap experienced while presenting different enabling technologies, Table 2.9 was generated to achieve a better understanding. It identifies the relationships between the enabling technologies and shows the contributions that combine multiple enabling technologies. This table does not include the category "single-path protocols" since it would not contribute to the evaluation and would add a large number of unnecessary entries in the table (every communication needs a protocol; thus, when it is not a multipath or multicast, it is usually single path). We also analyse the impact of API in the contributions separately.

Table 2.9: Proposals combining multiple enabling technologies

References	Multipath Protocols	Multicast Protocols	EDGE	SDN	NFV	ICN
[111] [120] [110] [163] [131] [109]			✓	✓	✓	
[106] [107] [112][114] [137]			✓		✓	
[101] [95][97][100][127]		✓		✓		
[102] [108] [121] [123]			✓	✓		
[126][46]	✓			✓		
[130]	✓			✓	✓	
[133]				✓	✓	
[118]	✓		✓			
[143]		✓	✓			✓
[144]			✓			✓
[148]			✓	✓		✓
[145]		✓				✓
[140]					✓	✓

The table content highlights the strong association between EDGE, SDN, and NFV, with five contributions each on EDGE and SDN, EDGE and NFV, and two on SDN and NFV, as well as six additional contributions combining all three. There is also a strong coupling between multicast protocols and SDN, with five contributions combining both technologies. ICN shows a strong link with other technologies, with five out of seven contributions combined with other technologies, especially with EDGE.

API is commonly combined with many enabling technologies due to its ubiquitous nature. In multipath protocols, API provides better control of different paths and protocol decisions. Examples of such contributions are NEAT [152, 153], TAPS [154] and Schmidt et al. [158]. Many other contributions include API as a means of achieving the requirements without being the main topic, such as Nunna et al. [113], Taleb et al. [115], Mahajan et al. [101] and Sardara et al. [145]. Overall, APIs have proven to be a valu-

able approach to enhance latency and reliability. Many papers on the topic assess increased reliability and decreased latency while also considering support for other KPIs and parameters as evaluated in Section 2.5.2. Therefore, we consider API to be a promising topic for discussion in Section 2.6.

2.5.2 KPI Coverage

Figure 2.3 summarizes the enabling qualities of each technology based on the analyzed contributions in this work. Each technology is presented in a radar chart that sets the line closer to the edges proportional to the percentage of contributions that focus on that KPI. In general, the literature has a high level of treatment of the latency KPI, followed by a medium-high level of treatment of reliability. Throughput and HetNet support are parameters frequently addressed, while partial reliability is not commonly a main point of study in contributions aimed at enhancing the latency or reliability.

As enablers, APIs and multipath protocols are the approaches most frequently used to address multiple KPIs. APIs address all parameters with at least a medium-high level of coverage, while in multipath protocols, there is a high level of work aimed at improving reliability, throughput, with a medium level on addressing HetNet support and latency. Then, single-path protocols and NFV also present a wide scope. In single-path protocols, latency and throughput are the main points, but HetNet support and partial reliability are also addressed a number of times; whereas NFV solutions focus mostly on latency but also address reliability and HetNet support. Each remaining technology has one or two differentiated main topics: reliability in multicast protocols, latency and HetNet support in EDGE and ICN, and latency and reliability in SDN.

All things considered, each enabling technology has its strengths and weaknesses, which can be combined to achieve the desired requirements, as will be discussed in Section 2.6.

2.5.3 On the Common Methods and Techniques

Figure 2.4 provides a clear overview of the methods and techniques used to achieve low latency and/or high reliability in existing literature One thing learned from this picture is that they reuse and enhance existing mechanisms to improve 5G, but there is some small novelty in the new mechanisms for 5G. Many of the methods are classic versions of basic mechanisms coming from fixed networks (congestion control or cross-layer) or techniques used to optimize end-to-end communications in previous 4G mobile communication networks (coding, context awareness, EDGE enhancements and multi-connectivity). Only a small number are specifically focused on 5G, like NFV or SDN methods, but they are also widely used in cloud (not wireless) environments. This first observation confirms that the core methods for

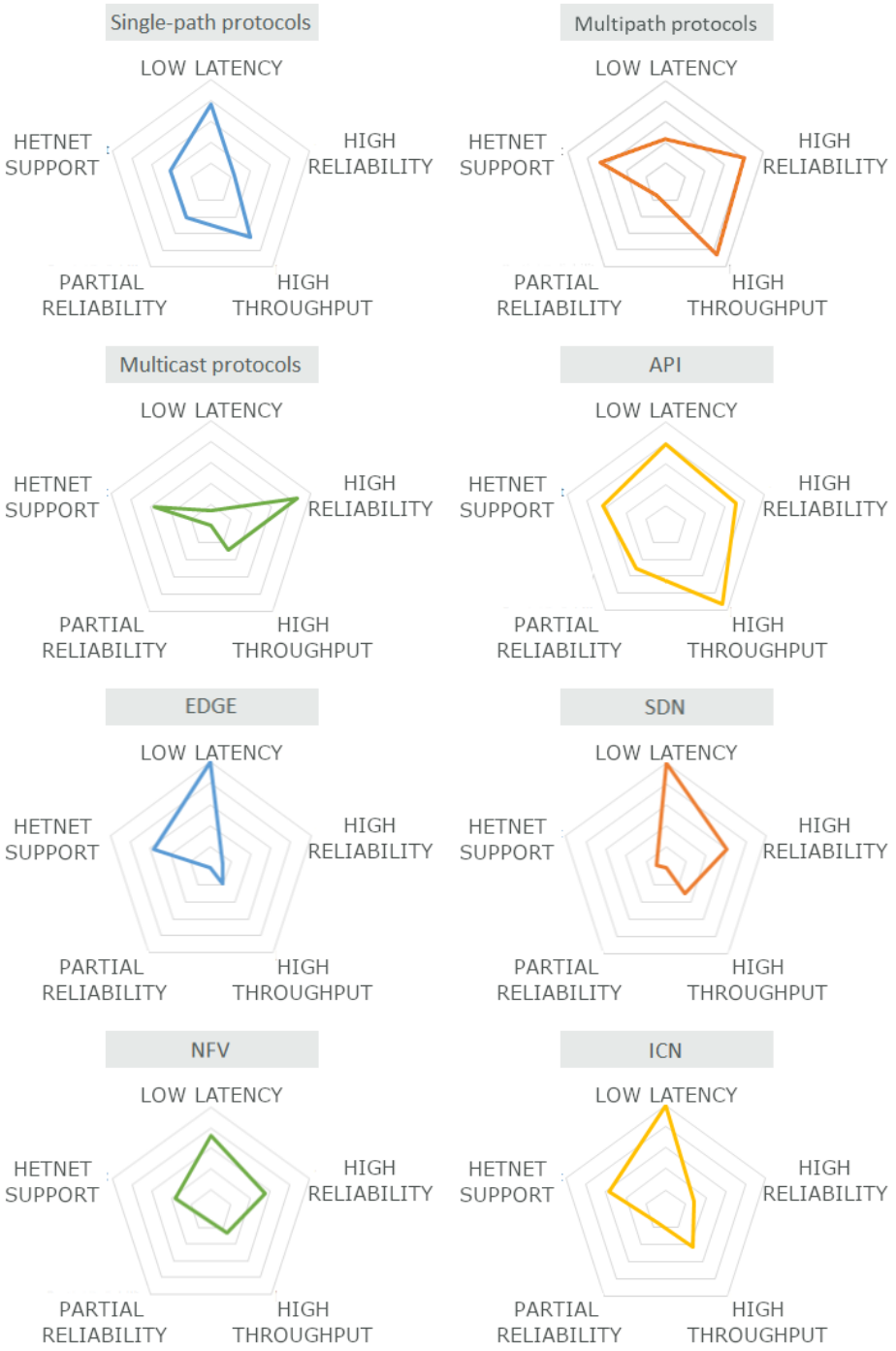


Figure 2.3: Mapping of the enabling technologies used to classify parameters based on their contributions

end-to-end solutions in different network technologies have some continuity and there are improvements and adaptations to new networks, but 5G does not mean a hard break when considering the latency and reliability. More revolutionary techniques are probably in the radio access part, but they are not part of the survey because we limit this survey to end-to-end solutions.

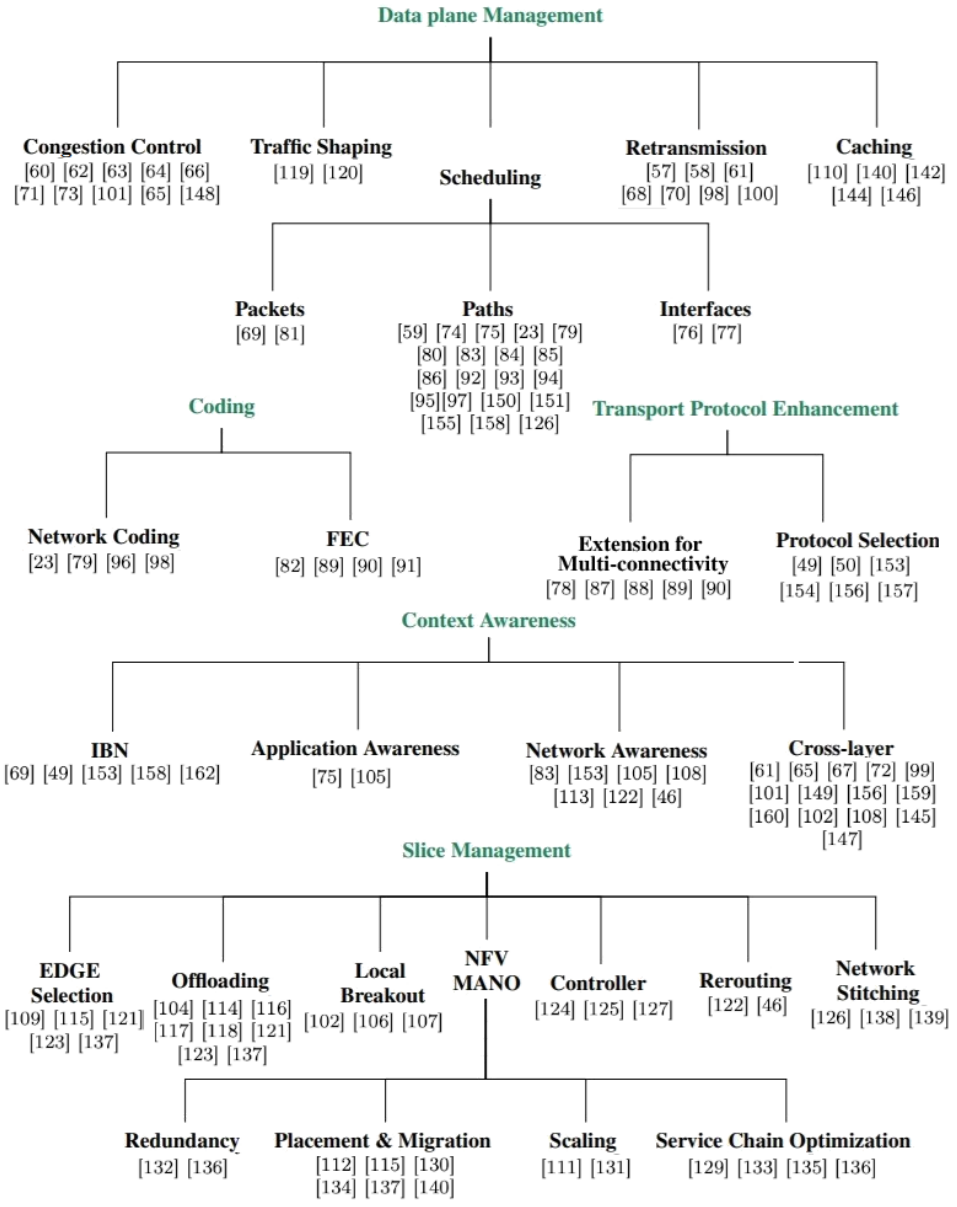


Figure 2.4: Classification of the literature according to the proposed taxonomy

The figure highlights the most popular topics, which include scheduling paths, cross-layer techniques, congestion control, and EDGE support for both offloading and selection. However, there are also some methods with only a few related papers that we believe have more potential, especially when combined with some of the more popular methods, such as automatic scaling in the NFV domain. We discuss some of these promising research topics from our perspective in the research section (Section 2.6).

2.6 Promising Research Topics

We have identified potential areas of research that may pose future challenges. One such area is edge computing, which plays a crucial role in reducing latency in communication. Despite advancements in protocols, technologies, and physical layers, there is still an inherent latency that comes with distance. To minimize this latency, applications and services need to deploy their instances in edge clouds closer to the user. However, this deployment must be done efficiently, which calls for further technological innovation.

Another promising area of research is the development of a 5G end-to-end self-organizing network. This would build upon the concept of self-organizing networks, where networks configure themselves to adapt to user conditions. The proposed solution involves a 5G network that can organize itself, with applications self-organizing and selecting where to be deployed (e.g., in the EDGE), and the network self-organizing and configuring NFV technologies and SDN paths. This would require significant development across several areas, including the creation of an orchestrator to reorganize services, flexible protocols to change connections to different endpoints without affecting user experience, and APIs to collect and offer contextual information to relevant entities. Some interesting contributions that are leading the research in this direction are Balevi et al. [164], Morocho-Cayamcela et al. [165], and the aforementioned work of Eurecom [110].

The third main area for research is end-to-end protocols. In this context, there are two primary areas of research that show great promise: multi-connectivity and the use of more expressive APIs. One of the key methods used for enhancing novel protocols is the scheduling of multiple paths. The potential of multi-connectivity in enhancing different KPIs, including reliability and latency, and the expansion of multi-homed devices has created a rising interest for this advancement. However, existing multi-connectivity solutions primarily rely on schedulers that determine how to use these paths. We foresee potential in using multi-connectivity in a more intelligent way, adapting the appropriate scheduling to the application requirements in each case. In order to do so, we present the second promising topic, the use of more expressive APIs that allow sharing context-information between layers. A more expressive API can expose network state information to the

application and application preferences to the protocol. This exchange of information, combined with an intelligent algorithm, can assist in the selection of key protocol configurations like the use of multiple paths, benefiting both the application by meeting the specific requirements and the network underneath by avoiding wasting resources. This thesis makes a contribution to this research line on end-to-end protocols through the creation of a multi-connectivity protocol that uses context awareness information to enhance its operation. The upcoming chapters will provide a comprehensive overview of the protocol and its development.

Chapter 3

The Multi-connection Tactile Internet Protocol (MTIP)

CONTENTS

Summary: In this chapter, the Multi-connection Tactile Internet Protocol (MTIP) is introduced. It describes the protocol packet structure, packet types and data transmission to support multiple connections. Moreover, this chapter also covers the recycling assumption that MTIP considers for sequence numbers, as well as the MTIP API and context information it supports.

3.1 Overview of MTIP

The Multi-connection Tactile Internet Protocol is a transport protocol for the remote control of Tactile Internet applications. The foundation of MTIP is the use of multiple paths to send data in a redundant way. MTIP combines this capacity with the use of context information to send latency-aware data in a more reliable and flexible way. Figure 3.1 presents an overview of the protocol in the TCP/IP stack.

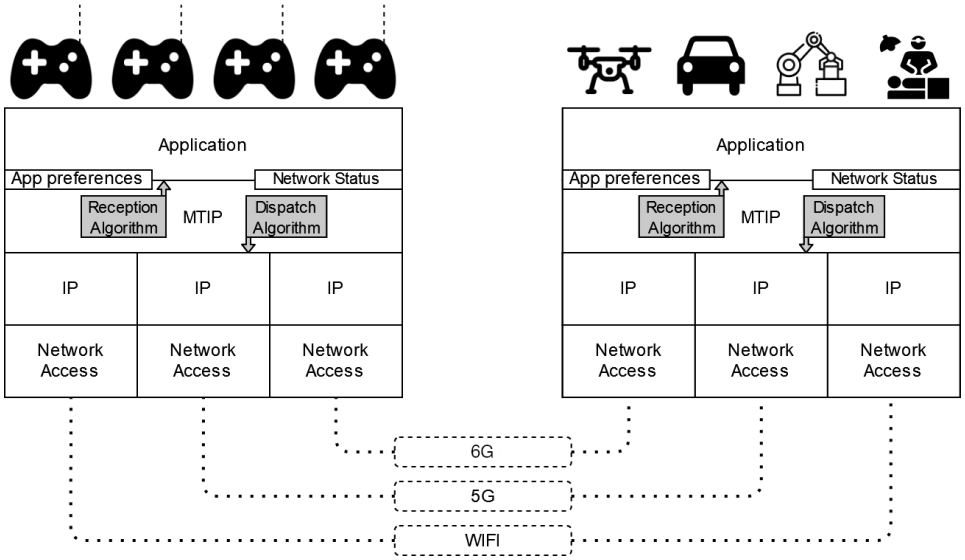


Figure 3.1: Sample scenario for MTIP applications

MTIP creates a link between a Tactile Internet device and a remote controller. This link consists of multiple sublinks that represent independent end-to-end connections through different wireless communication networks such as 5G or 6G, as shown in Figure 3.2. MTIP is envisioned for current modern networks with multi-homed devices and multiple path options [43]. Specifically, it is envisioned for large private networks, the typical environment of Tactile Internet use cases such as the remote control of industrial machinery or manufacturing. MTIP makes use of the specific mechanisms that private networks offer to function properly. These mechanisms include security, such as ciphering at the air interface and secure GTP tunnels [166] in the fixed part of the network, and synchronization, such as the Global Navigation Satellite System (GNSS) and the Precision Time Protocol (PTP) [167], which complement MTIP operation.

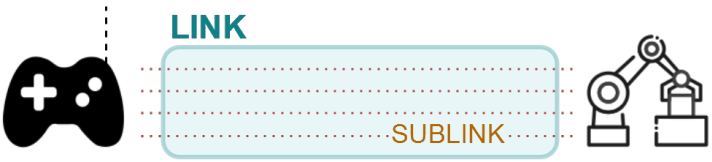


Figure 3.2: Use of multiple sublinks in MTIP

3.2 The MTIP Packet

The packet structure of MTIP in Figure 3.3 builds on a lightweight UDP-like header and includes additional fields to optimize it for the Tactile Internet (shown in blue). Timestamps are used to manage packet deadlines, sequence numbers ensure proper ordering and detect duplicates, and flags distinguish between the two types of packets used on MTIP: data and control packets.

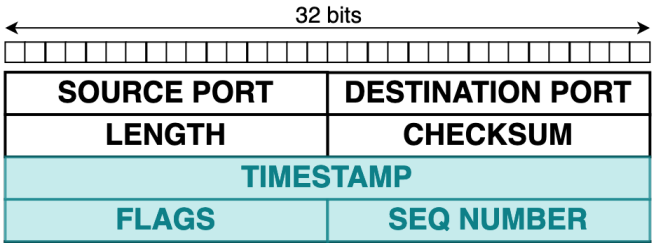


Figure 3.3: The MTIP header

3.2.1 Data Packets

Control packets contain the application data in their payload. They can be sent from one or multiple sublinks, depending on application preferences. On the receiving end, these packets go through the reception algorithm to determine if they are relevant to the application layer. If DATA packets carry the flag PRIO, they are sent directly to the application.

3.2.2 Control Packets

Control packets are identified by different flags and manage link operation. They are sent from all available sublinks in two or three way handshakes which are considered successful if at least one message of each type is received at the correspondent endpoint. They do not go through the reception algorithm to determine if they are relevant to the application layer. A summary of the different MTIP operations is shown in Figure 3.4.

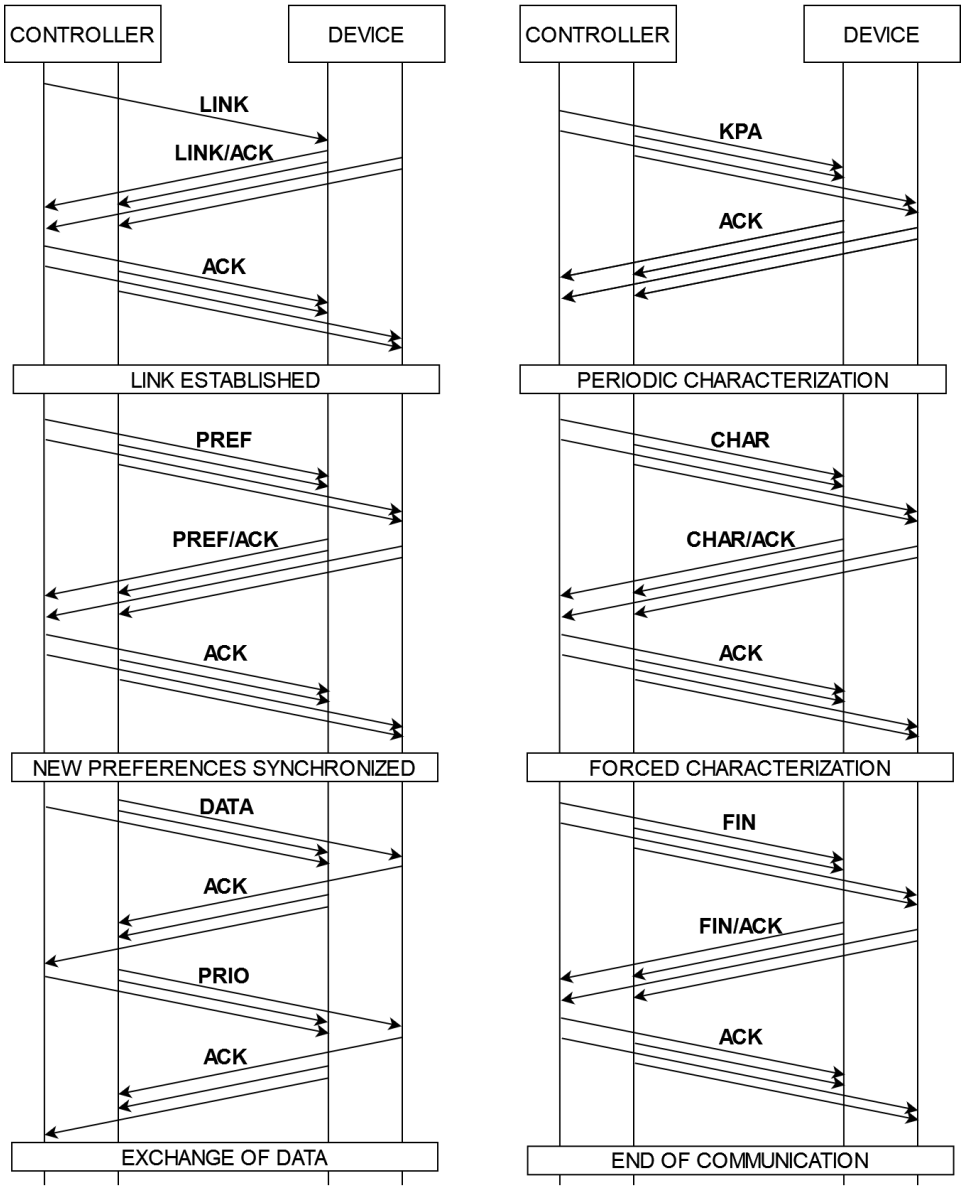


Figure 3.4: Main MTIP operations

- **LINK**: Packets with the flag link share information about the available source interfaces (IP and port pairs), in order to establish a link with the remote endpoint. The result is the creation of a link, constituted by multiple sublinks.

- **PREF**: Preference packets transmit information about the application's preferences to the other endpoint to ensure that they are synchronized.

- **KPA**: Keep alive packets are sent periodically to perform measurements in the sublinks and keep the KPI databases updated in the sublinks that are not in use.

- **CHAR**: Characterization packets also perform measurements of the state of the network. However, these messages perform an exhaustive characterization that can create a high overload due to the large amount of packets sent, so the application is responsible for choosing when to perform this kind of measurement.

- **FIN**: Fin packets close the communication. On receiving these messages, MTIP closes the link at both endpoints and frees all resources.

- **ACK**: Every time a data or control packet different than ACK is sent, MTIP sends an ACK back to the same sublink, with the same sequence number. These packets carry two types of information in the payload: information about the success or failure in the processing of the packet and information about the latency in order to have synchronized databases at each endpoint.

3.3 Data Transmission

MTIP offers various methods for data transmission. It can replicate data across all available sublinks, leave the choice to the application, or use an intelligent selection of sublinks through the MTIP sending algorithm. Using all sublinks simultaneously could maximize certain KPIs such as reliability; however, it would in most cases also result in a waste of network resources. On the receiving end, packets go through a reception algorithm to evaluate their relevance to the application layer. AN example Message Sequence Chart (MSC) of MTIP's data transmission is shown in Figure 3.5.

MTIP does not define explicit methods for flow control or congestion control beyond ensuring the order of messages to the applications. TI applications usually use short, latency-aware TI packets in a periodic pattern where including artificial waiting periods for the packets could negatively impact the low latency needs for TI.

3.3.1 The MTIP Sending Algorithm

The MTIP sending algorithm is able to adapt to conditions maximising the KPIs and minimising the waste of resources. As shown in Figure 3.6, it makes two decisions i) the amount of sublinks to use and ii) which sublinks to use.

MTIP starts sending data packets on all sublinks. Upon the reception of ACK packets, it evaluates if too many duplicate packets are being received

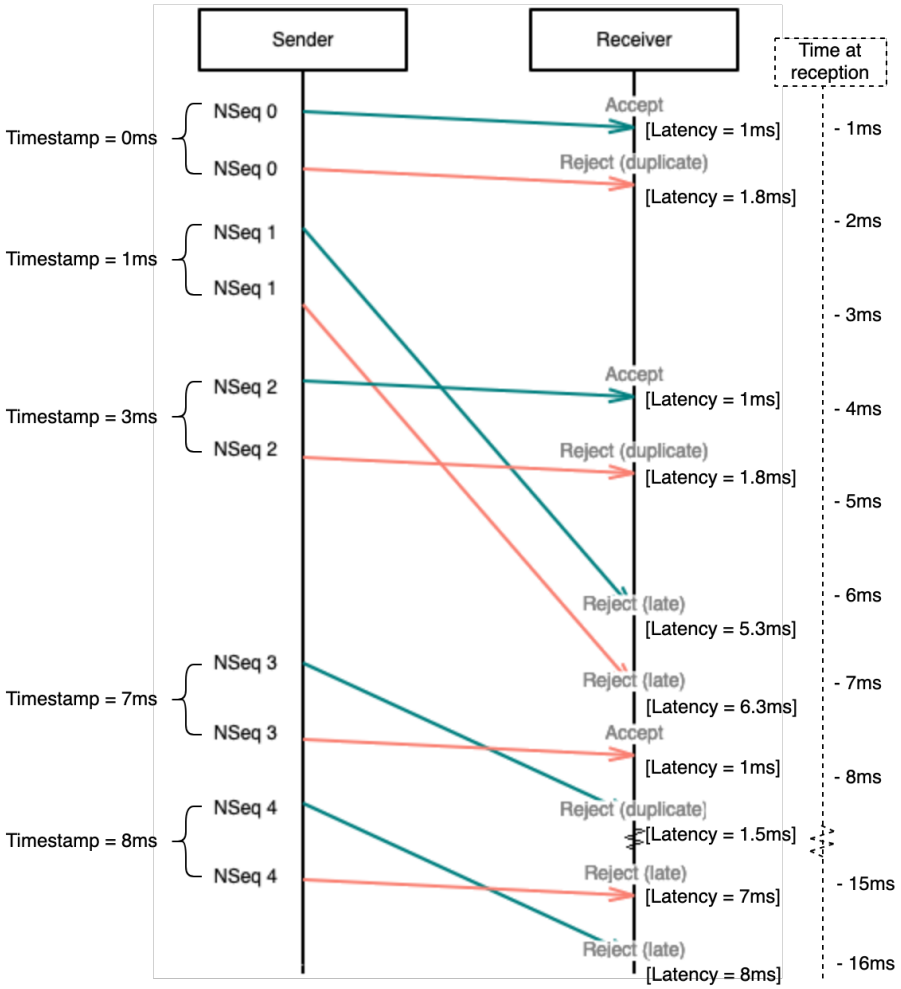


Figure 3.5: Example MSC of MTIP's data transmission

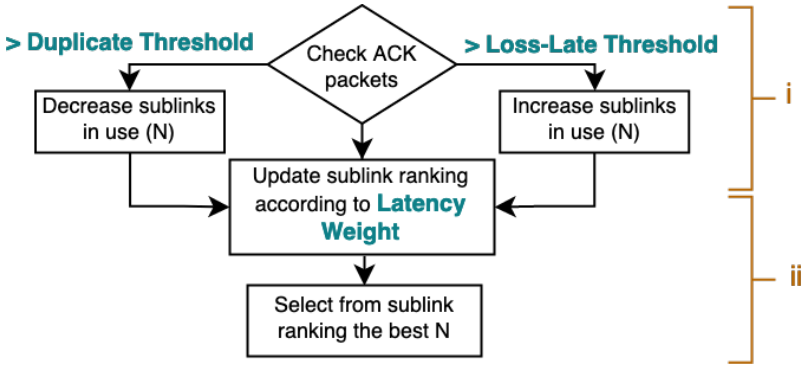


Figure 3.6: The MTIP sending algorithm

or if too many packets are lost or late. It makes so comparing the received packets with the percentage thresholds defined by the application in the `duplicate threshold` and `loss-late threshold`. Based on this, MTIP reduces or increases the number of sublinks in use, respectively. Then, MTIP assess which specific sublinks should be used for transmission by ranking the available sublinks. The application indicates whether this ranking should be done taking into account just the latency of the sublinks (100% `latency weight`), just the reliability (0% `latency weight`), or a weighed mean using both KPIs. The latency and the reliability of the sublinks are characterized by network measurements as defined in Section 3.4.1.

3.3.2 The MTIP Reception Algorithm

The MTIP reception algorithm is presented in Figure 3.7. This algorithm checks three main points before sending the data to the application: i) the data does not arrive late, ii) the data has not already been received, and iii) the data is in the expected order.

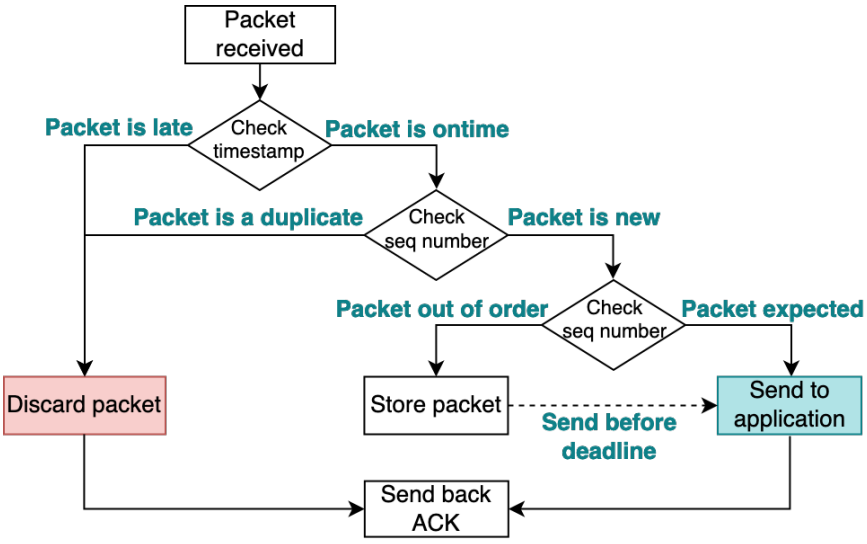


Figure 3.7: The MTIP reception algorithm

To check i), MTIP uses the timestamp on the packets to calculate the latency of each packet and compares it to the `maximum latency` that the application supports. To check ii) and iii), MTIP checks the sequence number of the packets, since duplicate packets share the same sequence number and new packets increment the number sequentially. It is important to note that MTIP saves information about the packets that has already received, so when a packet arrives at the receiving side, its sequence number is checked against a list that holds the sequence numbers of already received packets. Additionally, if a packet is out of order, MTIP stores it until the missing

packet is received or until it is considered lost. A packet is considered lost when the deadline for the next packet is close to be met (refer to [168] for further information about multipath packet expiration). The deadline of a packet is calculated using the timestamp of the packet and the application preference on the `maximum latency` supported.

3.3.3 Recycling assumption

The specific purpose of sequence numbers in MTIP is to avoid reorder and duplication when sending the data to the consumer application. Like in all protocols that use sequence numbers, their range is finite due to the actual limit of the protocol header, and some recycling is needed. We must ensure that the recycled numbers cannot disturb the correct functioning of the protocol, avoiding a negative impact in low-latency applications. Hence, MTIP cannot benefit from the sliding window feature of other protocols like TCP to avoid a quick recycling of sequence numbers. We need the following premise when designing the cyclic sequence numbers in MTIP (the **recycling assumption**): *the valid maximum latency supported by a packet is always much lower than the time needed to recycle the sequence numbers.* So, even if two packets with the same sequence number are present at the consumer buffer, one of them will be discarded as a late packet. Note that this is a realistic assumption as shown in Table 1.1, where the worst scenario for recycling would be the one with the maximum supported latency of 100ms. With 16-bit sequence numbers, MTIP can support a maximum of 2^{16} control actions in that 100ms slot to avoid two packets on time with the same sequence number. This translates into sending a maximum of one action every 0.0015ms (655360 packets/s). In [1], we observe that this is a reasonable data rate for TI applications like tele-operation or automotive, which present the most extreme case with an action every 0.25ms (4000 packets/s). In any case, longer sequence numbers could be used in specific applications at the cost of increasing the header.

3.4 Context Awareness

MTIP uses external information to offer an enhanced service. This information is separated into two categories: network and application awareness.

3.4.1 Network Awareness

MTIP collects information about the state of the network through data packets and control packets (Keep alive and Characterization). This information is stored as network KPIs for each sublink (both downlink and uplink). Latency and reliability are the main KPIs due to their importance in most TI

communications; however, context information itself can be particularized in different implementations.

- **Latency KPI**: When a packet arrives, MTIP calculates on the go the latency of the sublink using the timestamp on the packet and current time and synchronizes this information through ACK packets. Due to possible instability in the network, this information is stored using a weighted mean and a factor α.

$$latency = \frac{stored\_latency + (new\_latency * \alpha)}{1 + \alpha}$$

- **Reliability KPI**: MTIP can measure packet loss using sequence numbers and ACK packets. Due to possible instability in the network, the information is stored following a cumulative moving average, so packet loss is measured by the percentage of packet losses in the last packets.

3.4.2 Application Awareness

MTIP is aware of application preferences through the application programming interface. These preferences can be separated into two main categories.

- **Communication preferences**: Communication preferences inform MTIP about application transport-service requirements, e.g., the maximum acceptable one-way latency, the duplicate threshold, the loss-late threshold and the latency weight.

- **Protocol configurations**: Protocol configurations define how MTIP works internally. These configurations are part of MTIP implementation but MTIP can choose to expose them to the application. Some protocol configurations are the interval between Keep Alive packets, the number of relevant packets that are used for the algorithms and the factor α used in the latency KPI.

3.4.3 The API

MTIP presents a Socket-like API to manage both MTIP operation and context awareness. Table 3.1 presents the set of functions used to manage the link and the data transmission. These functions follow traditional TCP and UDP socket conventions. Table 3.2 presents the set of functions used to describe preferences and obtain feedback. The operation of the functions is described in the following subsections. However, the specific details of the current implementation developed for this thesis can be found in Section 5.1 and Appendix B.

Table 3.1: MTIP API functions to manage link and data

Function	Description
mtip_socket()	Creation of the socket.
mtip_bind()	Bind of the IP and port pairs.
mtip_link()	Establishment of a link.
mtip_send()	Data sending.
mtip_receive()	Reception of data.
mtip_close()	Closing the link (active).
mtip_finished()	Closing the link (passive).

Table 3.2: MTIP API functions to manage preferences and feedback

Function	Description
mtip_preferences()	Set communication preferences.
mtip_feedback()	Get information on network measurements.

3.4.3.1 Creation of the socket

To start a connection, the function `mtip_socket` needs to be executed. This function returns a descriptor that will be used to identify the socket in future operations. It supports an optional parameter to express any socket options exposed by the implementation.

```
1 int mtip_socket(int options)
```

Listing 3.1: mtip_socket

3.4.3.2 Bind sublinks

To receive packets in a socket, IP addresses and ports need to be bound. Since MTIP can support multiple sublinks, a bind for each origin address and port should be made.

```
1 int mtip_bind(int sd, char *addr, int port)
```

Listing 3.2: mtip_bind

3.4.3.3 Link establishment

In order to establish a link between endpoints and exchange the information about the available IP addresses and ports, both endpoints should execute `mtip_link`. The controlled device must execute `mtip_link` in `DEVICE` mode, to wait and listen to a connection, while the remote controller must initiate

in `CONTROLLER` mode, to initiate the connection. The outcome of this function is a socket successfully linked and ready to exchange data.

```
1 int mtip_link(int sd, uint8_t mode, char *addr, int port)
```

Listing 3.3: mtip_link

3.4.3.4 Reception of data

MTIP offers a function `mtip_receive` to set a callback for the reception of data. This callback is a function that will be executed each time a new message is received in the application. The information that can be retrieved in this call consists of the data of the message and the timestamp information. The timestamp data can be used optionally in cases that the application require it.

```
1 int mtip_receive(int sd, void (*callback)(char *, double))
```

Listing 3.4: mtip_receive

3.4.3.5 Data sending

MTIP uses the function `mtip_send` to send data from its set of source interfaces to the previously linked destination. Specific selection of sublinks will be performed using the preferences exposed by the application with `mtip_preferences`. Application can select flag PRIO to mark the packet as urgent as described in Section 3.3.

```
1 int mtip_send(int sd, char *message, int flag)
```

Listing 3.5: mtip_send

3.4.3.6 Closing the link

When an endpoint closes the link, the connection closes in both endpoints, freeing all resources. This operation is started using the function `mtip_close`. However, MTIP also offers the function `mtip_finished` to define a callback that will be executed if the other endpoint closes the connection, in the cases where the application needs to act accordingly.

```
1 int mtip_close(int sd)
```

Listing 3.6: mtip_close

```
1 int mtip_finished(int sd, void (*callback)())
```

Listing 3.7: mtip_finished

3.4.3.7 Preferences

Application can set preferences on the socket according to its demands. For this purpose, MTIP offers the function *mtip_preferences*. The preferences that can be exposed with this function includes both `communication preferences` of tha application and `protocol configurations` that depend on the implementation.

```
1 int mtip_preferences(int sd, char *prefs)
```

Listing 3.8: mtip_finished

3.4.3.8 Feedback

MTIP offers a function to obtain information about the link and the characterization of the sublinks. The application can demand different types of feedback information depending on the implementation.

```
1 int mtip_feedback(int sd, char *&feedback,  int type)
```

Listing 3.9: mtip_finished

3.4.3.9 Examples

In Listings 3.10 and 3.11, we provide a simplified example of the use of MTIP's API in C language. Listing 3.10 presents a remote controller application that tunes the MTIP algorithm and sends commands to the other endpoint, while Listing 3.11 represents a controlled device that is receiving and processing these commands. Extended examples with additional considerations on current MTIP implementation are presented in Appendix B and accessible in [169].

```
#define IP_1 "10.0.0.2"
#define IP_2 "11.0.0.2"
#define IP_REMOTE "10.0.0.1"
#define PORT 1111
#define SIZE 100

int main(){
 /* Creation of the socket */
 int socket_descriptor = mtip_socket();

 /* Binding interfaces */
 mtip_bind(socket_descriptor,IP_1,PORT);
 mtip_bind(socket_descriptor,IP_2,PORT);

 /* Configuring link preferences */
 char preferences_json[SIZE] = "{\n"
                               "  \"maxLatency\": 1e+7,\n"
                               "  \"duplicateThreshold\": 20,\n"
                               "  \"losslateThreshold\": 5,\n"
                               "  \"latencyWeight\": 50"
                               "}";
 mtip_preferences(socket_descriptor, preferences_json);

 /* Linking socket */
 mtip_link(socket_descriptor, MTIP::Mode::CONTROLLER,
    ↪ IP_REMOTE, PORT);

 /* Sending information */
 char msg[SIZE] = "COMMAND";
 mtip_send(socket_descriptor, msg, MTIP::Flag::DATA);

 /* Further execution... */

 /* Obtaining feedback */
 char feedback[SIZE];
 mtip_feedback(socket_descriptor, feedback, MTIP::GENERAL);
 printf("Feedback: %s\n", feedback);

 /* Closing link */
 mtip_close(g_socket_descriptor);
}
```

Listing 3.10: Example controller application using MTIP

```
#define IP_1 "10.0.0.1"
#define IP_2 "11.0.0.1"
#define IP_REMOTE "10.0.0.1"
#define PORT 1111
#define SIZE 100

void reception_callback(char *order);

int main(){
 /* Creation of the socket */
 int socket_descriptor = mtip_socket();

 /* Binding interfaces */
 mtip_bind(socket_descriptor,IP_1,PORT);
 mtip_bind(socket_descriptor,IP_2,PORT);

 /* Configuring reception function */
 mtip_receive(socket_descriptor, reception_callback);

 /* Linking socket */
 mtip_link(socket_descriptor, MTIP::Mode::DEVICE, NULL, 0);

 /* Further execution... */
}

void reception_callback(char *order) {
  /* Processing command... */
  printf("Received: %s\n", order);
}
```

Listing 3.11: Example device application using MTIP

Chapter 4

Modeling and Verification of MTIP

CONTENTS

SUMMARY: This chapter provides a formal analysis and verification of MTIP. It includes an overview of the modeling and analysis tools used, as well as the models created in PROMELA and timed automata (code accesible in [169]). The chapter also covers the verification of MTIP's correctness properties with SPIN and MTIP's performance properties with UPPAAL.

4.1 Formal Analysis and Verification of MTIP

The main challenge after the design of MTIP is to guarantee the correctness and performance of the algorithms to send and to receive packets in MTIP. We address the automated analysis and verification of correctness and statistical performance combining the two well known tools SPIN and UPPAAL as complementary environments to cover all the analysis and verification work.

The tool SPIN is a model checker developed by G. Holzmann [170] for the logical verification of concurrent software. Since its origin, it has been utilized in a number of applications and protocols [171, 172, 173], still being nowadays a powerful tool used to verify modern systems [174, 175, 176]. SPIN's effectiveness lies in its ability to verify the absence of deadlocks, non-progress loops, and complex properties described using Linear Temporal Logic (LTL). However, SPIN does not support performance analysis. UPPAAL is also a model checking tool that has been used to analyse both traditional and novel communication protocols [177, 178, 179, 180]. UPPAAL supports stochastic timed automata, which allows the analysis performance-related properties defined using TCTL (Timed Computation Tree Logic). In order to perform a comprehensive formal analysis and verification, we evaluate MTIP from two perspectives, the correctness evaluation with SPIN and the evaluation of statistical performance with UPPAAL.

4.2 Background: Tools for Modeling and Analysis

In this section, we provide the description of the tools SPIN and UPPAAL.

4.2.1 PROMELA: the Modeling Language of SPIN

In what follows, we summarize the main characteristics of PROMELA, the modeling language of SPIN. A more extensive description of the language can be found in [181]. We use a simplified example of MTIP's PROMELA code in Listing 4.1 and refer to its lines of code to present the language.

Typically, a PROMELA program consist of several processes that execute concurrently interleaving their instructions. Processes communicate via shared variables or channels. Listing 4.1 contains an example of how global constants, variables and channels can be defined in PROMELA. As can be observed, the syntax is similar to that of C, although new data types appear such as `bool` and `byte` (lines 3 and 4). Since the use of memory in models is vital for the later analysis process, PROMELA provides data types able to adapt to the real variable sizes. In contrast, some other types (such pointers and floating point) are not allowed. Line 5 show how enumerated types are declared in PROMELA. Additionally, lines 6 and 7 of show, respectively, an array of asynchronous channels (`sublink` is an array of `MAXSBL` buffers of size

`MAXSEQ+1`) and a global synchronous channel (`source` of size 0). PROMELA uses the CSP symbols "!" and "?" to respectively denote the operators for sending and receiving data via channels (as in line 12 of and line 24 for example). Processes are declared using the reserved word `proctype` (as in Line 10). The word `active` in the `proctype` declaration means that there must be an instance of this type of process from the beginning of the model execution.

We now discuss the subset of PROMELA sentences used in the MTIP model. PROMELA sentences can be classified as *basic* or *composed*. *Basic* sentences are the atomic instructions of the language such as assignments, Boolean expressions and operations on channels. Boolean expressions in PROMELA are used differently from other languages. A Boolean expression is an *instruction*, but it is only executable if it is evaluated to `true`. This special way of dealing with Boolean expressions makes it easy to model process synchronization by shared variables in the language. Basic instruction `skip` is commonly used in the language as being equivalent to `true`. In addition, assignments are always executable and operations on channels are executable if it is allowed by the state of channels. The executability of basic instructions is key to synchronizing processes in the language.

The rest of PROMELA sentences are *composed* by combining several basic sentences. For instance, control flow sentences are composed of the reserved word `if`/`do` followed by several branches (each one starting with symbols "::") and the reserved word `fi`/`od` at the end. Each branch starts with a basic sentence which determines the branch executability (see lines 12 and 40). Usually, we say that a branch is open when its first basic sentence is executable. When a control sentence is to be executed, one of its open branches is selected in a non-deterministic manner to continue the execution. If no branch is open, then the control sentence suspends. Thus, for instance, the `do` instruction of line 24 is executable iff instruction `source?data` is executable or, equivalently, iff the synchronization via channel `source` can be carried out. In case it cannot, process `sender` blocks at line 24 until it can proceed with the communication. As shown in line 47, the last branch of a control `if`/`do` sentence can start with the reserved word `else`. In this case, the sentence never suspends since the `else` branch is selected when all the other branches are closed. `if`/`do` sentences can also be combined with channel pool operations, as in line 42. The channel poll operation resembles a receive statement, but with square brackets enclosing the message field list. It returns true or false, indicating the executability of the associated receive operation. The pool operation might use the underscore symbol _ to represent a global, predefined, write-only, integer variable, which serves as temporary storage for scratch values.

```
#define MAXSEQ 3
#define MAXSBL 2
bool rr, rb
byte nr, nb, nlr, nlb
mtype = { none, red, white }
chan sublink[MAXSBL] = [MAXSEQ+1] of {mtype,byte,byte}
chan source = [0] of { mtype }
byte timebuffer[MAXSEQ+1];

active proctype Source(){
    do
    :: source!white
    :: source!red -> break;
    od
    do
    :: source!white
    od
}

active proctype Sender() {
    byte SeqNum, x;
    mtype data;
    do
    :: source?data ->
        atomic{ (timebuffer[SeqNum]==FREE) ->
            timebuffer[SeqNum]=1;
            sublink[0]!data, SeqNum;
            sublink[1]!data, SeqNum;
            increasetime();
        }
        progress: inc(SeqNum);
    od
}

active proctype Receiver() {
    byte ExpectedNum, rcpbuffer[MAXSEQ+1];
    do
    :: atomic{
        if
        :: ExpectedNum == 0 -> rcpbuffer[MAXSEQ]=none;
            if
            :: sublink[0]?[_,0,ONTIME] -> SublinkNum=0;
            :: sublink[1]?[_,0,ONTIME] -> SublinkNum=1;
            :: timeout ->
               increase: ExpectedNum = ExpectedNum = +1
            fi
        :: else -> rcpbuffer[ExpectedNum-1]=none;
        fi
        (...)
    }
}
```

Listing 4.1: Example of PROMELA syntax

`Atomic` is a very useful composed sentence provided by the language. It is used to define as atomic a sequence of instructions, as shown in line 25. In this case, if the Boolean expression `timeBuffer[seqNum]==Free` is `true`, the rest of instructions nested in the atomic are executed as an atomic block. It is important to clarify that if some instruction inside the `atomic` suspends, the `atomic` breaks to avoid the system deadlock.

SPIN provides the `timeout` sentence, which is dealt with as a Boolean expression that is `true` only when no other program instruction is executable. It is useful to model situations where a process detects inactivity and decides to act to avoid the system deadlock. It is worth noting that SPIN does not support time; however `timeout` can be used as a high level abstraction of a timer. The second tool used in this paper, UPPAAL, supports modelling timed automata.

Finally, PROMELA sentences can be *labelled* using an identifier followed by symbol "`:`". For instance, `increase` in Line 45 is a label of the assignment sentence in that line. Labels are useful to describe properties that need to express whether a process has reached a certain program counter as shown in Section 4.2.3. There are some predefined labels. For instance, labels that start with `progress` (Line 31 of Listing 4.7) are used to detect non-progress cycles. A non-progress cycle is an unproductive interleaving of processes that never reach an instruction labelled as `progress`.

4.2.2 Timed Automata: the Modelling Language of UPPAAL

UPPAAL is a toolbox for verification of real-time systems jointly developed by Uppsala University and Aalborg University [182]. An UPPAAL model is a network of the so-called *timed automata* which execute in parallel, communicate and synchronize each other via shared variables and synchronous channels. Channels may also be declared as *broadcast* to distribute a message to more than one automaton. Each automaton is composed of *locations* (the automaton states) and transitions. It can also have local variables to store the local automata configuration. Besides the discrete variables common in computational systems, *timed automata* of UPPAAL may also contain the so-called *clock* variables which are real-valued and dense variables whose values evolve synchronously with time.

Figure 4.1 shows the timed automaton of a MTIP `sender` process in UPPAAL to exemplify the main components of the language. Locations are written as circles in the automaton (for instance, `Initial`, or `WaitingForReset`) while the arrows are the transitions. Observe that the initial automaton location (`Initial`) is represented with a double circle. In this automaton, variable `clockTime` is a global clock used to register the global time. Locations can have *invariants* which correspond to Boolean expressions that have be `true` while the automaton is at the corresponding location. For instance, the location `ReadyToSend` has `clockTime<=TIMELIMIT && clockTime<=sndTime`

`+ 1` as invariant. This means that the automaton may stay at `ReadyToSend` while `clockTime` is less or equal to `TIMELIMIT` and `sndTime + 1`. Thus, the automaton has to transit to another location at the latest when expressions `clockTime=TIMELIMIT` or `clockTime =sndTime + 1` become `true`; otherwise, the automaton blocks.

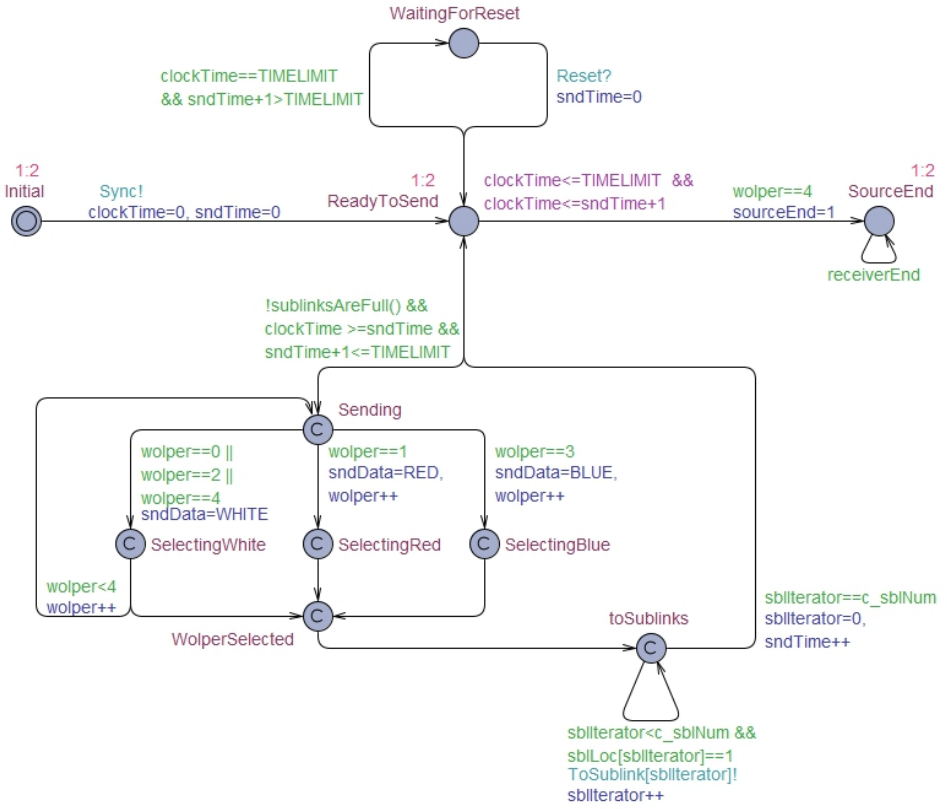


Figure 4.1: Example of UPPAAL syntax

In addition, automata transitions can be decorated with different elements. A transition may have a *guard* that is a Boolean condition. An automaton can only take a transition if its guard evaluates to `true`. For instance, in Figure 4.1, transition from `ToSublinks` to `ReadyToSend` can be executed only if the guard `sblIterator==c_sblNum` is `true`. A transition may also have a *synchronized* condition which is expressed via an operation on a synchronized channel. For instance, the transition from location `WaitingForReset` to `ReadyToSend` can only take place if the synchronization via channel `Reset` is available. Observe that the operations on channels are written using the CSP notation as in PROMELA. As commented above, this is the natural UPPAAL mechanism for synchronizing different timed automata in the system. Finally, transitions can also be used to *update* some variables.

For instance, the transition from `Sending` to `SelectingRed` updates variable `sndData` to `RED` and increases variable `wolper` by one. Clock variables can be reset in transitions (for instance, variable `clockTime` is initialized in transition from location `Initial` to `ReadyToSend`). Finally, it is worth noting that a transition cannot be executed if the invariant of the remote location is `false`.

Intuitively, the execution of a network of timed automata proceeds as follows. All network automata start at their initial location (there must be exactly one for each automaton). The network can evolve in two ways: (1) by means of a *continuous network transition*, where the time of all clocks increases simultaneously and (2) by means of a *discrete network transition*, when one (or several) automaton executes a transition. It is important to note that discrete transitions are *instantaneous*, meaning that their execution involves no time passing. Observe that when a transition contains a synchronization via a channel, it has to be executed simultaneously by all the automata involved in the synchronization. The network blocks when it is not possible to execute either a continuous or a discrete transition.

Note that in a timed automata network, there are several sources of non-determinism that may produce a non-numerable set of different behaviors. For instance, an automaton may take a transition at any moment during the time interval in which both the invariant and the transition guard are `true` and the transition synchronization, if it exists, is available. In addition, if several discrete transitions are possible, the system may evolve selecting any of them.

UPPAAL provides several mechanisms to manage non-determinism. One of them is to declare locations as *committed* which is denoted placing a `C` letter inside the location. When some automaton of the network is at a committed location, no continuous transition can be carried out, since the transitions from committed locations are executed with priority. Committed locations allow executing a sequence of transitions atomically (as sentence `atomic` in SPIN). For instance, in previous Figure 4.1, the sequence of discrete transitions from `Sending` to `SelectingRed`, then to `WolperSelected`, to `ToSubLinks`, etc. are atomically executed since all these locations are committed.

4.2.3 Specification of Properties with Temporal Logic

Tools SPIN and UPPAAL accept the properties described using two different subsets of the temporal logic CTL*. Properties in SPIN are written using the *linear temporal logic*, while in UPPAAL, a simplified version of the *timed computational tree logic* (TCTL) is used. TCTL is a real-time variant of CTL aimed at expressing properties of timed automata. Both LTL and TCTL share the Boolean and temporal operators and quantifiers, although they differ in how formulae can be constructed.

Formulae in LTL and TCTL are constructed from a generic set of *atomic* propositions which can be evaluated on system states. In practice, the atomic propositions used in SPIN and UPPAAL are the Boolean expressions allowed by their respective input modelling languages. Anyway, if s is a state and p an atomic proposition, we denote with $s \vdash p$ that s satisfies proposition p. Atomic propositions can be combined by Boolean operators such as negation !, disjunction ||, conjunction && and implication $\rightarrow$. The formulae constructed only using atomic propositions and Boolean operators (for instance, $p||q$ or $p\&\&!q$) are called *state formulae*, since they can be evaluated on single states. In LTL, a trace π satisfies a state formula when its initial state satisfies it. Similarly, in TCTL, a computational tree satisfies a state formula when it is satisfied by its root state s.

Regarding temporal operators, we first introduce temporal operators in the context of LTL and then explain how they are used in TCTL. Operator *eventually*, written as `<>`, is used to describe properties that must hold *at least once* in the future. Operator *always*, written as `[]`, is used to describe properties that must *always* hold in the future. With operator *always*, it is possible to write the *safety* and *invariant* properties to be held on systems. In contrast, operator *eventually* is used to build *liveness* properties. For instance, the property "variable `x` is never negative" is an invariant which can be written as `[](x >= 0)`. Similarly, "variable `x` takes sometimes a non negative value" is a liveness property written as `<>(x >= 0)`. In LTL, we can nest temporal operators to construct more sophisticated properties. For instance, "whenever the producer sends a message, then the receiver will read it eventually" is a liveness property written as `[](send` $\rightarrow$ `<> read)`. In SPIN, it is very common to use labels to detect whether an execution has passed through a given instruction. For instance, LTL property `<>receiver@o_r` expresses that process `receiver` has to execute instruction `o_r` at least one. Property `[]<>receiver@o_r` is a bit more complicated. It is a liveness property that is held by the system only if process `receiver` executes instruction `o_r` infinitely often. LTL operator *until* (written as `U`) is used to express more complicated properties that may involve both safety and liveness. In particular, ϕ_1 `U` ϕ_2 holds iff formula ϕ_2 occurs in the future, from the current state (the liveness component) and, meanwhile, ϕ_1 is always `true` (the safety component). For instance, the property `net_state=INIT U ([] net_state == ONTIME)` can be used to describe a network that once it has been initialized, all packets always arrive on time.

Universal `A` and existential `E` quantifiers are dealt with differently in LTL and TCTL. LTL does not explicitly use quantifiers since it is assumed that all formulae are universally quantified implicitly. In contrast, in TCTL, temporal operators must always be written along with a quantifier. Thus, only considering the temporal operators eventually (`<>`) and always (`[]`), we have four possible operators in TCTL: `A<>`, `E<>`, `A[]`, `E[]`. The other combined opera-

tors are defined similarly. In TCTL, invariants are written using operator `A[]`, while liveness can be expressed using both `A<>` and `E<>`. For instance, the property "in all paths, variable `x` sometimes takes a non-negative value" could be specified as `A<> (x>= 0)`. In addition, property "in all paths, whenever the producer sends a message then, in all possible future paths the receiver will eventually read it" can be written as `A[](send` $\rightarrow$ `A<>read)`. It is worth noting that LTL and TCTL are temporal logics with different expressiveness. For instance, TCTL property `A[](send` $\rightarrow$ `E<>read)` ("in all paths, whenever the producer sends a message then there exists at least a possible future path where the receiver will eventually read it.") cannot be expressed in LTL due to the use of quantifier `E`. Inversely, LTL property `<>[](x>=0)` that expresses that "in each execution, there exists an state from which variable `x` is always non-negative" cannot be written in TCTL due to the use of the two nested temporal operators. TCTL formula `A<>A[](x>=0)` is similar, but it is not equivalent to LTL formula `<>[](x>=0)`.

To finish the section, it is important to note that both SPIN and UPPAAL limit the type of temporal properties to be analysed on systems. In the case of SPIN, operator *next* is not included since it is mostly useless due to the non-deterministic interleaving of processes. UPPAAL supports the analysis of a subset of TCTL formulae. Thus *safety* properties can be described only using formulae such as `A[]`φ, `E[]`φ; *reachability* properties can be written as `E<>`φ; and *liveness* properties can only have two forms: unconditional liveness `A<>`φ and conditional liveness `A[](`$\varphi_1 \rightarrow$ `A<>`φ_2`)` written as φ_1`-->`φ_2 in UPPAAL. In all these formulae, φ stands for a state formula. The analysis of deadlock is implemented as a verification option in SPIN, while UPPAAL defines the keyword `deadlock` that is satisfied by all deadlocked states. Thus, formula `A[](not deadlock)` can be used to verify that systems are deadlock-free in UPPAAL.

4.2.4 UPPAAL SMC

We now discuss the main features of UPPAAL SMC, an extension of tool UPPAAL able to carry out statistical model checking (SMC) on networks of timed automata supported by UPPAAL.

Model checking is an exhaustive technique which may not be suitable to analyse complex systems that are subject to a stochastic nature. This is the case of most communication protocols which display an inherent probabilistic behavior with respect to various aspects such as delays in the message emission or possible message loss. For these systems, the correctness of some properties, such as performance, cannot be guaranteed 100%. Statistical model checking combines model checking and testing techniques with statistic results to conclude whether a system satisfies a property with a *degree of confidence* which may be enough to be sure that the system behaves as expected.

UPPAAL automata and TCTL properties can be enriched with statistical annotations before being simulated and analysed by UPPAAL SMC. We now summarize the features used to model MTIP. One of the key points to simulate a stochastic timed automaton (STA) is to decide the automaton delay in each location. When the location has a time bounded invariant, UPPAAL SMC associates it with an uniform probability distribution, meaning that it is equally probable to leave the location at any point of the location time interval. However, when the location has an unbounded time invariant, UPPAAL SMC gives it an exponential distribution whose rate is provided by a rational number of the form `a:b` decorating the location. For instance, location `ReadyToSend` of the automaton of Figure 4.1 has associated value $\frac{1}{2}$; which is the rate of the location's exponential probability distribution. In an STA network, each automaton races with the rest to leave its current location following its probability distribution function. The winner automaton jumps to another location. Transitions can also be decorated with constant values, meaning the probability that the automaton chooses the transition to leave the location. For instance, in the part of a `sublink` automaton represented in Figure 4.2, values 100−`c_lossTendency` and `c_lossTendency` have been added to the two transitions from `NonEmpty` to `PacketSent` to represent that the probability of *losing* and *not losing* a packet are respectively $\frac{\texttt{c\_lossTendency}}{100}$ and $\frac{\texttt{100-c\_lossTendency}}{100}$, `c_lossTendency` being a constant defined in the model.

Regarding the properties, UPPAAL SMC accepts all the TCTL formulae accepted by UPPAAL discussed above, along with new formulae related to the stochastic interpretation of timed automata. In particular, in this paper, we have used properties of the form `Pr[<=N]<>`ϕ which constitute the so-called *hypothesis testing*, in the sense that they are asking if the probability of formula ϕ is true before the global system clock reaches time `N`. To reach this value, the implementation of UPPAAL SMC is based on the probability estimation algorithm [183] that computes the number of runs needed to produce an approximated value of the probability, with an error of $\pm\epsilon$ and a confidence of $1-\alpha$, ϵ and α being two constants that can be configured in the tool.

4.3 Modeling Decisions

The main difficulty to use model checking is how to build an abstract model of the protocol that represents the relevant behaviors, but at the same time can be analyzed with the available resources (mainly system memory). As explained by G. Holzmman in [170], the protocol model for verification should only reflect the details to conclude that properties verified in the model are also satisfied in the real protocol. In addition, removing unnecessary details contributes to mitigate the state explosion problem. Practical rules for mod-

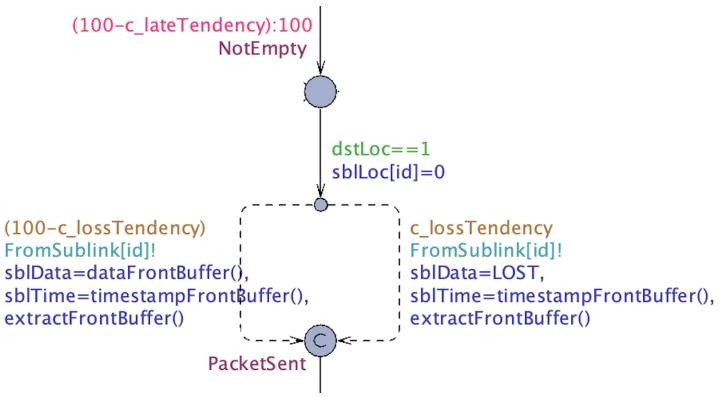


Figure 4.2: Example of UPPAAL syntax for SMC

eling include avoiding redundancy, sinks, sources, filters, or counters [184].

In our models of MTIP, such abstractions should additionally consider the size of relevant parameters such as application layer data, sequence numbers, timestamps and sublinks. This section provides our design decisions regarding these aspects.

4.3.1 Wolper Test for Application Data

One major challenge when modeling MTIP is to decide how much application layer data should be sent over MTIP to preserve the correctness properties. This problem was already addressed by G. Holzmann [170] using previous results by P. Wolper [185]. Wolper developed the "data independence test" which reduces the amount of different data to be used to analyze the correctness of one algorithm. He demonstrated that verification results are valid for a given set of user data if the behavior of the underlying system managing these data does not depend on the specific set. G. Holzmann adapted Wolper results to the case of end to end communication protocols in order to verify correctness properties. Holzmann stated that three distinct data items suffice for the typical correctness properties of flow control protocols (e.g., red data, blue data and white data), if they are sent in a sequence consisting of one red and one blue message inserted randomly in an infinite sequence of white messages. This number of minimum distinct data is independent of other factors, such as the sequence number range or the window size.

An example of the semantics of a PROMELA code to implement the Wolper test is presented in Listing 4.2. The first loop determines whether to send white or red data in a non-deterministic manner. If red data is chosen, the second loop is triggered. In the second loop, the decision is analogous, this time with the option of sending white or blue data. If blue data is selected, the process proceeds to the third loop, where the Wolper test is concluded by sending white data.

Figure 4.3 represents the equivalent in UPPAAL. It presents a loop that starts sending white messages (*SelectingWhite*) until a non-deterministic selection of UPPAAL increases the variable `wolper`. Then, it sends a red message (*SelectingRed*) and returns to sending white messages (*SelectingWhite*). At some point, the non-deterministic interleaving will increase the variable `wolper` again, and it will send a blue message (*SelectingBlue*). Finally, it concludes with an infinite sequence of white messages (*SelectingWhite*) and the Wolper test is considered completed. As for model checking, this code, as well as the PROMELA code, will be executed in an exhaustive manner (interleaving with the rest of the MTIP model), to ensure that all possible infinite sequences of white, red and blue messages are sent.

```
1  mtype { red, white, blue };
2  char source = [0] of { mtype };
3
4  active proctype Source(){
5        do
6        :: source!white
7        :: source!red -> break;
8        od
9        do
10       :: source!white
11       :: source!blue -> break;
12       od
13 end:  do
14       :: source!white
15       od
16 }
```

Listing 4.2: The Wolper test in PROMELA

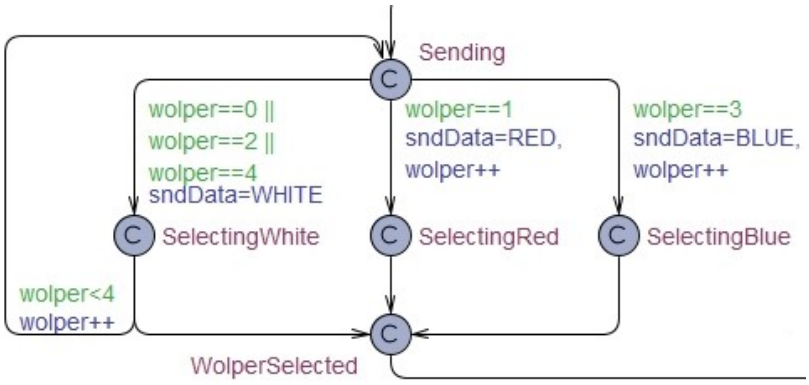


Figure 4.3: Part of the `sender` automaton in UPPAAL: the Wolper test

4.3.2 Sequence Numbers

Following the **recycling assumption**, we need to find a valid abstraction to reduce the 2^{16} available sequence numbers in MTIP to a reasonable size for a formal verification. Our goal is to accurately model and manage various scenarios that can occur in an MTIP link, including duplicate packets, packet reordering, packet loss, and late packets.

To achieve this, we take an incremental approach to determine a reasonable size for the sequence numbers. Firstly, we need at least one sequence number to detect duplicate packets. If the receiving side receives this sequence number twice, it can flag the packet as a duplicate. If we want to model packet reordering, we need at least two differentiated sequence numbers to enable the protocol to identify unexpected orders during processing. Additionally, to represent gaps in received data, we need an extra sequence number to abstract the previous situation plus some application data being lost. Finally, if we want to incorporate late application data, we need an additional sequence number. Therefore, to model the most extreme case where different application data is duplicated, reordered, lost, and late, we need a minimum of four sequence numbers to ensure that the protocol can recover from this scenario effectively. Figure 4.4 displays an example that depicts this abstraction. It shows a sequence of white, red and blue data sent over a link (constituted by multiples sublinks) and received at a destination endpoint. In green, it shows duplicated, reordered, lost and late data using 16-bit sequence numbers, while in yellow it shows how this scenario will translate into 4-bit sequence numbers in order to be effectively abstracted.

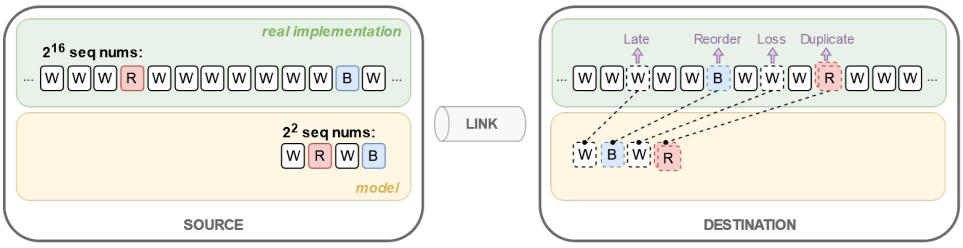


Figure 4.4: Example of the abstraction of sequence numbers for the models

4.3.3 Sublinks

The use of several sublinks can provoke packet reorders (unexpected sequence number received) and duplicates (same sequence number received from several sublinks) at the receiving side. It is clear that we need at least two sublinks to model reorders and duplicates. The question is whether more than two sublinks add any added value compared to two sublinks. Regard-

ing the correctness properties, adding more sublinks would generate longer reorders and more duplicates, but the states generated with two sublinks are enough to check MTIP's ability to manage these situations. However, in terms of performance properties, even if two sublinks are enough, it is interesting to explore the impact of additional sublinks in the protocol's trade-off. Therefore, in our UPPAAL model, we use three sublinks.

4.3.4 Modeling Time

MTIP uses timestamps to control its operation, so some kind of time must be used when modeling. The approach taken to model time in SPIN different from the one in UPPAAL. Since SPIN does not support timers, some kind of operation must be implemented to model time. This operation uses variables to model time and is described in detail in Section 4.4.2. On the other hand, UPPAAL supports clock variables that effectively model time. However, these clock variables cannot be read or written from them, so we need to use additional variables to extract the timestamps. In both cases, even if the model of time differs, the variables should not grow uncontrollably to avoid state explosion but, at the same time, should be sufficient to allow the presence of packets that are still on time and packets that are late in the sublinks.

We use an incremental approach for modelling time. In the PROMELA Model A, we use a static approach with two defined timestamps `ONTIME` and `LATE`. In the PROMELA Model B, we use a counter that increases the timestamp of each active packet in a non-deterministic manner every time a new packet is sent. We use three values 1, 2 and 3 as the units to represent time (0 means that the tag is free and can be reused). Value 1 represents no further time progress than the minimum round trip time, modeling ideal speed in the channel for a while. Value 3 means the channel substantially worsens for a while, provoking reaching the deadline (packets arrive late). Value 2 means a chance for normal transmission, allowing packets to accumulate delay gradually. In the UPPAAL model, we use incremental variables for the timestamps (`sndTime` and `rcvTime`) to model both time and sequence numbers and we set the `TIMELIMIT` to the limit studied for sequence numbers of four. This abstraction does not limit behaviors, since losses would simulate periods without sending data and replicates the behavior of TI applications that normally send data in a constant and periodic manner [1] (e.g., 1KHz). However, in a real implementation where timestamps are used to measure the specific end-to-end latency of each packet and packets might be sent with small variations in the period, it is necessary to use the complementary sequence numbers to ensure MTIP operation in a fail-safe manner.

4.4 Correctness Analysis of MTIP with SPIN

In this section, we present the correctness properties of MTIP, the models developed PROMELA language and the verification of the correctness properties of the protocol with the SPIN tool.

4.4.1 MTIP Correctness Properties

Regarding correctness properties, MTIP operation can be separated into four premises that must be satisfied at all times. For a better understanding, in this section, data at the application level will be referred as *messages*, while data at the transport level will be referred to as *packets*. The four correctness properties of MTIP are the following.

- **P1**: Application never receives duplicate *messages*.
- **P2**: Application never receives *messages* that have met their deadline (*packets* that are considered late).
- **P3**: Application never receives out-of-order *messages*.
- **P4**: Every *packet* that is not a duplicate or late is forwarded to the application.

4.4.2 The PROMELA Models for MTIP

We follow an incremental approach to modeling and verification with two models of MTIP that exhibit different levels of abstraction. The first model (A) is suitable to verify properties **P1**, **P2** and **P3**. The second model (B) is oriented to property **P4**, but also can be used for the others. The structure of the two models is represented in Figure 4.5. Both models contain the two processes `sender` and `seceiver`, two unidirectional communication channels representing two `sublinks` and one application sending user data (process `source`). Note that the models contain some elements not used in a real implementation, but they are feasible abstractions that do not modify the 's real logic. In particular, we need to address the problem of modeling time in a language that does not support timers but instead the `timeout` construction to recover from packet lost. We also need to reduce the range of variables like sequence numbers to keep the model verifiable with the available memory. The main difference between the models resides in how they address this modeling of time, with model B implementing a more rich and complex management of time. In the following, we provide the details of the two models, the justification of the abstraction decisions and a simulation trace.

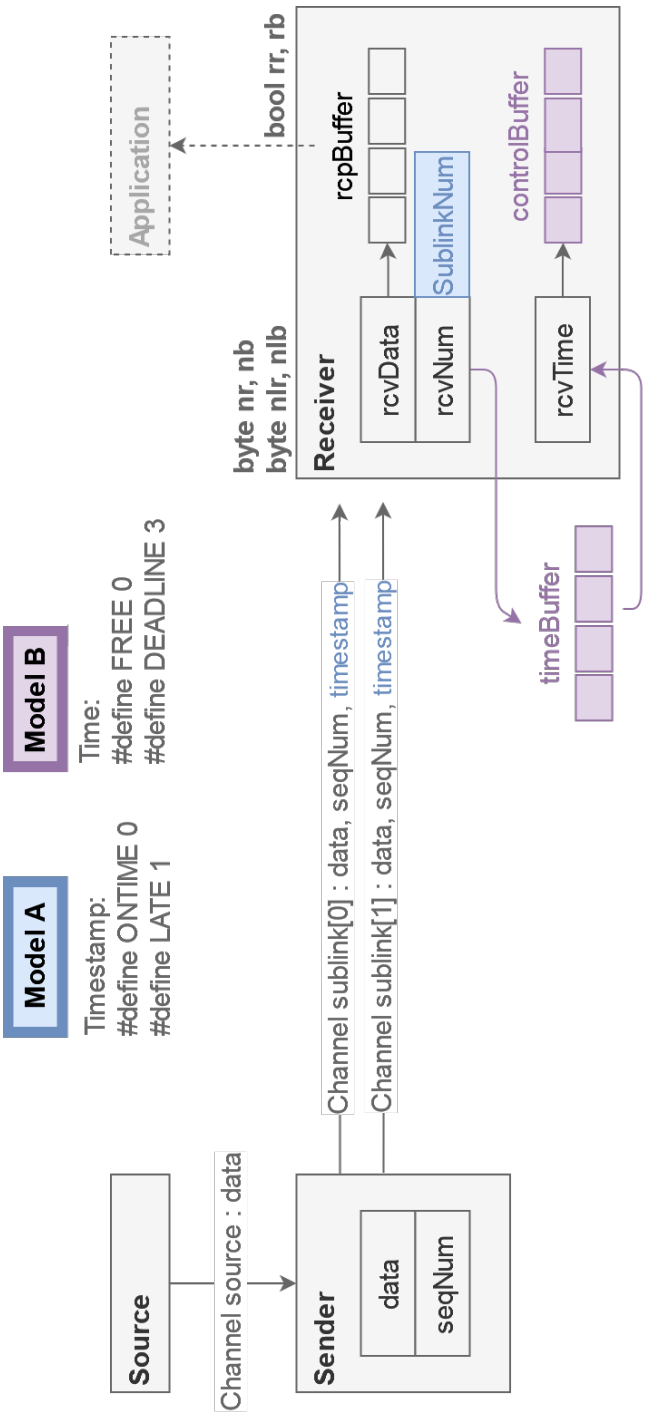


Figure 4.5: Simplified graphical view of the PROMELA models A and B

4.4.2.1 Description of the Model A

The main components of the model are the two processes `sender` and `receiver` using two unidirectional sublinks. The data structures, constants and the `sender` process are defined in Listing 4.3. `sender` is modeled with a loop getting data from the channel `source` and sending it in a non-deterministic way using only sublink 0 (lines 20 to 23), only sublink 1 (lines 24 to 27), the two sublinks (lines 28 to 35) or none of them (line 36). Selecting no sublinks when sending is a low cost way to model packet loss. With this non-deterministic selection at the `sender`, we are offering, in practice, any combination to the `receiver`. Each packet sent includes its corresponding sequence number (range 0 to 3) and the abstraction of the timestamp information (`ONTIME` or `LATE`). With the loop in the `sender`, each packet is managed to have any potential behavior when travelling in the network, two `ONTIME` packets being the best case and two lost packets the worst.

There are some details in the model to save resources during verification, like the use of `atomic` blocks, the use of the synchronous channel `source` that models the Wolper test (see Listing 4.4), or the inclusion of the sentence `(ExpectedNum != SeqNum)` in line 39 as blocking point. This last sentence acts as a guard to suspend the `sender` process in case of attempt to send a new packet with the same sequence number already in transit to the `receiver`. This not feasible in a real implementation, but it is a valid abstraction due to the **recycling assumption**.

The process `receiver` executes an infinite loop with two parts. In the first part (Listing 4.5), it executes one of the 4 blocks depending on the expected sequence number. Each block has the same behavior as the one in lines 18 to 28: if the expected packet is available `ONTIME`, the sublink is selected (in a non-deterministic manner if it is available on both). If `ONTIME` is not available, then the `timeout` in line 21 moves to read `LATE` packets. Again, if no `LATE` packets arrive, a second `timeout` in line 25 informs on packet lost, increases the counter of the expected sequence number and continues the loop to receive a new packet (line 11).

In the second part (Listing 4.6), the received packet is processed. The packets on time are inserted in the structure `rcpbuffer[]` (line 8), in order to control duplicate copies, and they are consumed provided there are no subsequent copies of a previous received packet with the same sequence number (line 9). Note that we also update a number of variables (`nr, nb, rb, nlr, nlb`) that refer to the reception of `blue` or `red` data and that will only be used for checking properties as described in Section 4.4.3.

```
#define MAXSEQ  3
#define MAXSBL 2
#define ONTIME 0
#define LATE 1
#define inc(x)  x = (x+1)%(MAXSEQ+1)
bool rb
byte nr, nb, nlr, nlb
mtype = { none, red, white, blue }

byte ExpectedNum;
chan source = [0] of { mtype }
chan sublink[MAXSBL] = [MAXSEQ+1] of {mtype,byte,byte}

active proctype Sender(){
    byte SeqNum; mtype data;
    atomic{
    do
    :: source?data ->
        if
        ::  if
            :: sublink[0]!data, SeqNum, ONTIME;
            :: sublink[0]!data, SeqNum, LATE;
            fi
        ::  if
            :: sublink[1]!data, SeqNum, ONTIME;
            :: sublink[1]!data, SeqNum, LATE;
            fi
        ::  if
            :: sublink[0]!data, SeqNum, ONTIME;
            :: sublink[0]!data, SeqNum, LATE;
            fi
            if
            :: sublink[1]!data, SeqNum, ONTIME;
            :: sublink[1]!data, SeqNum, LATE;
            fi
        :: true -> skip;
        fi
        inc(SeqNum);
        (ExpectedNum!=SeqNum) ->
    od
    }
}
```

Listing 4.3: Model A (Global variables and sender)

```
active proctype Source(){
    do
    :: source!white
    :: source!red -> break;
    od
    do
    :: source!white
    :: source!blue -> break;
    od
    do
    :: source!white
    od
}
```

Listing 4.4: Model A (Source)

```
active proctype Receiver() {
    byte RecvNum, RecvTime, LastNum, SublinkNum;
    mtype RecvData;
    int i;
    byte rcpbuffer[MAXSEQ+1];
    d_step{for(i : 0 .. MAXSEQ){rcpbuffer[i]=none;}}
    do
    :: atomic{
        rr=false;
        rb=false;
        Receving:
        if
        :: ExpectedNum == 0 -> rcpbuffer[3]=none;
        :: else -> rcpbuffer[ExpectedNum-1]=none;
        fi
        if
        :: ExpectedNum==0 ->
            if
            :: sublink[0]?[_,0,ONTIME] -> SublinkNum=0;
            :: sublink[1]?[_,0,ONTIME] -> SublinkNum=1;
            :: timeout ->
                if
                :: sublink[0]?[_,0,LATE] -> SublinkNum=0;
                :: sublink[1]?[_,0,LATE] -> SublinkNum=1;
                :: timeout -> ExpectedNum =
                (ExpectedNum+1)%(MAXSEQ+1); goto Receving;
                fi
            fi
        :: ExpectedNum==1 -> (...) //Analogous
        :: ExpectedNum==2 -> (...) //Analogous
        :: ExpectedNum==3 -> (...) //Analogous
        fi
        sublink[SublinkNum]?RecvData,RecvNum,RecvTime
    }
    (...)
```

Listing 4.5: Model A (Receiver part 1: Variables and Receiving)

```
1     (...)
2     if
3     :: RecvTime==ONTIME -> //Case: Ontime
4         if
5         :: rcpbuffer[RecvNum] != none -> //Case: Duplicate
6             rcpbuffer[RecvNum]=none;
7         :: else ->
8             rcpbuffer[RecvNum]=RecvData;
9             if //Consume Data
10            ::RecvData == red ->
11                nr++;
12                assert(!rb);assert(nlr==0);assert(nr<=1);
13            :: RecvData == blue ->
14                nb++; rb=true;
15                assert(nlb==0);assert(nb<=1);
16            :: else -> skip;
17            fi
18        fi
19    :: else -> //Case: Late
20        if
21        :: RecvData == red -> nlr++;
22        :: RecvData == blue -> nlb++;
23        :: else -> skip;
24        fi
25    fi
26    od
27 }
```

Listing 4.6: Model A (Receiver part 2: Algorithm)

4.4.2.2 Description of the Model B

Model B is an alternative to Model A exposing more details on the management of time and potential reorder. Structure `timebuffer[]` in Model B includes a counter per active sequence number that abstracts the time that a packet has been on the network until it reaches the receiving side. Such counter is increased every time a new packet is sent until a deadline is reached (when no more updates are performed in order to avoid states in the model checking process) or until the packet is consumed by the `receiver` (when the entries in this structure become free). Listing 4.7 presents the `sender` process that always sends the packets using the two sublinks. Listing 4.8 presents the operation to increase time. The sentence `(timebuffer[SeqNum]==FREE)` in line 17 of Listing 4.7 controls that a free slot in the structure is available and could suspend the `sender` until the `receiver` frees space.

```
1  #define MAXSEQ  3
2  #define MAXSBL 2
3  #define FREE 0
4  #define DEADLINE 3
5  #define inc(x)  x = (x+1)%(MAXSEQ+1)
6  bool rr, rb
7  byte nr, nb, nlr, nlb
8  byte timebuffer[MAXSEQ+1];
9  mtype = { none, red, white, blue, consumed}
10 chan sublink[MAXSBL] = [MAXSEQ+1] of { mtype, byte }
11 chan source = [0] of { mtype }
12
13 active proctype Sender() {
14     byte SeqNum, x; mtype data;
15     do
16     :: source?data ->  atomic{
17         (timebuffer[SeqNum]==FREE) ->
18         timebuffer[SeqNum]=1;
19         sublink[0]!data, SeqNum;
20         sublink[1]!data, SeqNum;
21         increasetime();}
22         inc(SeqNum);
23     od
24 }
```

Listing 4.7: Model B (Global Variables and sender)

Listing 4.9 contains the reception on the `receiver` process. The process executes an infinite loop with a non-deterministic selection of the sublink to read from every iteration (lines 18 to 21). Unlike the previous Model A, where non-deterministic losses were implemented in the `sender` process, in Model B we introduce an alternative approach where losses are optional and implemented in the `receiver` process. Listing 4.10 shows this non-deterministic selection and MTIP's reception algorithm. Once a message is successfully received from the channel, the sequence number `RecvNum` is used to check if the packet is still on time (lines 9 and 41). The packets on time are inserted in the structure `rcpbuffer[]` (line 18) in order to control duplicate copies, and they are consumed provided there are no subsequent copies of a previous received packet with the same sequence number (line 16 and 17).

Finally, `controlbuffer[]` is used to clean the previous `rcpbuffer[]` and `timebuffer[]` when any sequence number is going to be reused by the model's `sender` process (see Listing 4.11). This structure and control operations such as the one in line 17 of Listing 4.7 are needed in the model due to the limited amount of sequence numbers available in the models.

```
inline increasetime(){
    atomic{ int time;
    if
    :: time=0;
    :: time=1;
    :: time=2;
    fi
    if
    :: time>0 -> for (x : 0 .. MAXSEQ) {
     if
     :: timebuffer[x]>0 && timebuffer[x]<DEADLINE->
        if
        :: (timebuffer[x]+time)>DEADLINE->
                timebuffer[x]=DEADLINE;
        :: else -> timebuffer[x]= timebuffer[x]+time;
        fi
     :: else;
     fi }
    :: else;
    fi }
}
```

Listing 4.8: Model B (Modeling time)

```
active proctype Receiver() {
    byte RecvNum, RecvTime;
    byte rcpbuffer[MAXSEQ+1], controlbuffer[MAXSEQ+1];
    mtype RecvData;
    #IFDEF LOSSES
    byte LastNum=3;
    #ENDIF LOSSES
    int i;
    d_step{
    for (i : 0 .. MAXSEQ) {
            rcpbuffer[i]=none;
    }}

    do
    :: atomic{
        rr=false;
        rb=false;
        if
        :: sublink[0]?RecvData,RecvNum
        :: sublink[1]?RecvData,RecvNum
        fi
        controlbuffer[RecvNum]++; }
        (...)
```

Listing 4.9: Model B (Receiver part 1: New Variables and Receiving)

```
(...)
 if
 #IFDEF LOSSES
 :: //LOST
 #ENDIF
 :: //NO LOST
     RecvTime=timebuffer[RecvNum];
     if
     :: RecvTime<DEADLINE -> //Case: Ontime
         if
         :: RecvData == red -> ontime_red: skip;
         :: RecvData == blue -> ontime_blue: skip;
         :: else -> skip;
         fi
         if
         :: rcpbuffer[RecvNum] != none -> //Case: Dup
         :: else ->
             rcpbuffer[RecvNum]=RecvData;
             #IFDEF LOSSES
             if
             :: (LastNum+1)%(MAXSEQ+1)==RecvNum ->
                 LastNum=RecvNum; //No gap: consume
                 if
                 :: RecvData == red ->
                     nr++; rr=true; skip;
                 :: RecvData == blue ->
                     nb++; rb=true; skip;
                 :: else -> skip;
                 fi
                 rcpbuffer[RecvNum]=consumed;
             :: else -> skip; //Gap: Do not consume yet
             fi
             #ELSE
             if //Consume data
             :: RecvData==red-> nr++;rr=true; skip;
             :: RecvData==blue->nb++;rb=true; skip;
             :: else -> skip;
             fi
             #ENDIF
         fi
     :: else -> //Case: Late
         if
         :: RecvData==red->late_red: nlr++; skip;
         :: RecvData==blue->late_blue: nlb++; skip;
         :: else -> skip;
         fi
     fi
 fi
 (...)
```

Listing 4.10: Model B (Receiver part 2: Reception Algorithm)

```
1      (...)
2      if //Clean buffers for next round (reset)
3      :: controlbuffer[RecvNum]== 2 ->
4          controlbuffer[RecvNum]= 0;
5          #IFDEF LOSSES
6          if
7          :: else; skip;
8          :: (LastNum+1)%(MAXSEQ+1)==RecvNum ->
9              LastNum=(LastNum+1)%(MAXSEQ+1);
10             if
11             :: rcpbuffer[LastNum]!=none &&
12                rcpbuffer[LastNum]!=consumed ->
13                     if
14                     :: rcpbuffer[LastNum] == red ->
15                         nr++; rr=true; skip;
16                     :: rcpbuffer[LastNum] == blue ->
17                         nb++; rb=true; skip;
18                     :: else -> skip;
19                     fi
20                     rcpbuffer[LastNum]=none;
21             :: else -> skip;
22             fi
23         fi
24         #ENDIF
25         rcpbuffer[RecvNum]=none;
26         timebuffer[RecvNum]=FREE;
27     :: else;
28     fi
29 od
30 }
```

Listing 4.11: Model B (Receiving part 3: Clean Buffers)

4.4.3 Simulation and Formal Verification

We use the tool SPIN to analyse the correct functioning of the models. Figure 4.6 show examples of message sequence charts produced by SPIN simulations of the PROMELA models. After the simulations, we use SPIN to exhaustively analyse if the models satisfy the expected properties. In Model A, we use assertions while in Model B, we use the linear temporal logic language. The reason behind this decision is the cost of checking temporal formulas over Model A due to the huge number of states produced when making the interaction of the internal automata representing the formula and the actual processes. Another practical reason is to first use a more invasive but closer way to define requirements in software designs. In particular, properties **P1**, **P2** and **P3** are represented in Model A in lines 12 and 15 of Listing 4.6. All these properties are satisfied, as reported in Table 4.1 in the row "Asserts". For Model B, we translate the properties presented in Section 4.4.1 into LTL and present them in Listing 4.12. The formulas use variables and labels defined to represent different states. Variables `rr` and `rb`

represent that a red or blue message, respectively, has been forwarded to the application. The labels `receiver@ontime_red` and `receiver@ontime_blue` represent when packets are received on time by the protocol, while variables `nr` and `nb` are used to count the amount of red or blue data on time received by the protocol and the variables `nlr` and `nlb` are used to count the amount of red or blue data that arrives late to the protocol.

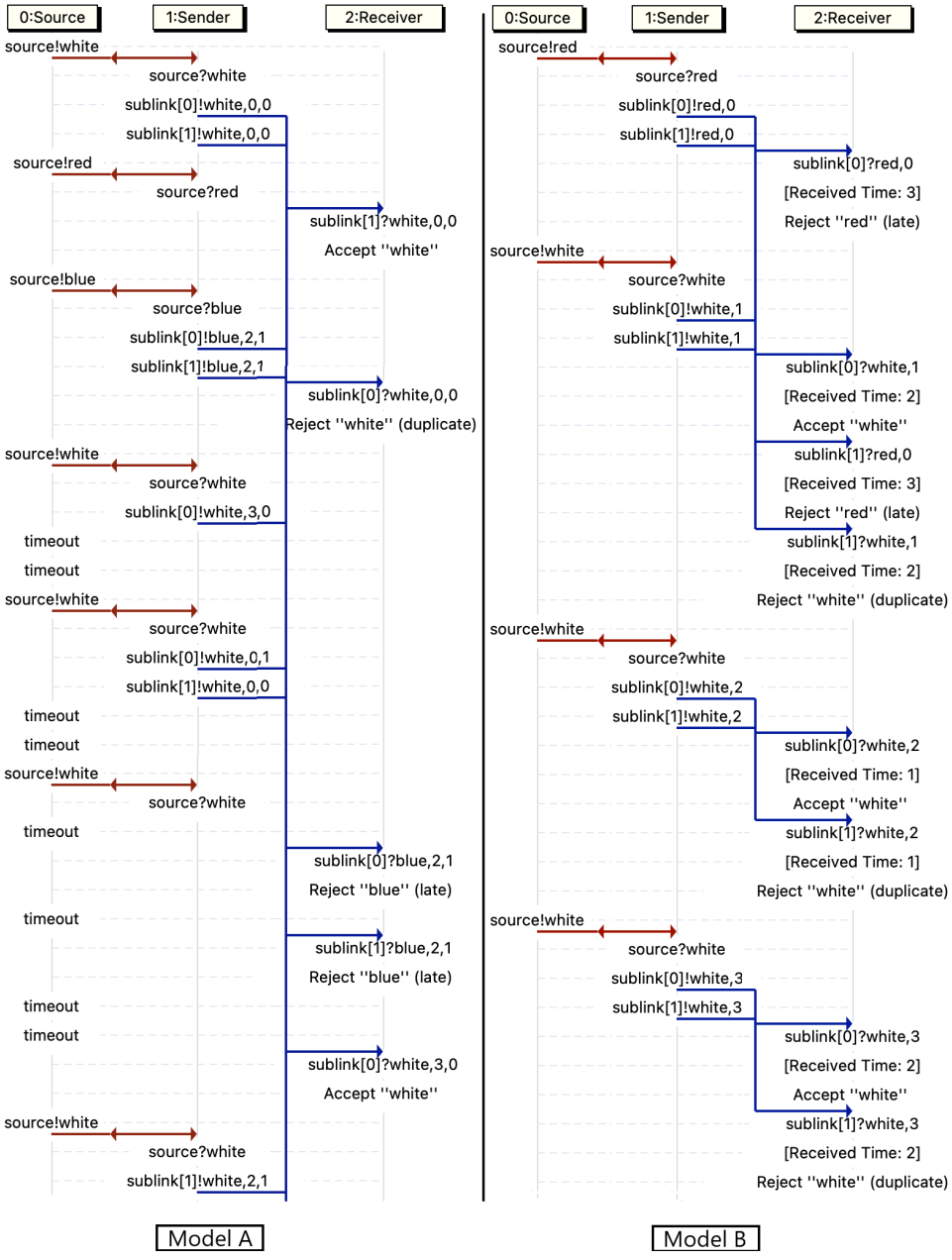


Figure 4.6: MSC generated from the PROMELA models by SPIN

The LTL properties check red and blue data to be able to prove MTIP properties, following the Wolper test. **P1** checks the absence of duplicate *messages* in the application, controlling that the application never receives more than one red or blue data. **P2** proves that the application never receives *messages* that have met their deadline by checking that if all red or blue data is late, they will not be sent to application. Then, **P3** checks that no red data reaches the application after the reception of a blue *message*, using the Wolper test to prove the absence of out-of-order. Finally, **P4** assesses that if a packet arrives on time and no message of that color has reached the application layer, the application will eventually receive that message.

```
ltl P1{ [](( nr<=1) && (nb<=1)) }
#IFDEF LOSSES
ltl P2{ [](( nlr==2 -> [](nr==0)) && [](nlb==2 -> [](nb==0)) }
#ELSE
ltl P3{ [](nlr>0 -> [](!rr) && [](nlb>0 -> [](!rb) }
#ENDIF
ltl P3{ [](rb -> [](!rr) }
ltl P4{ [](( receiver@ontime_red && nr==0) -> <>rr) &&
        [](( receiver@ontime_blue && nb==0)-> <>rb)}
```

Listing 4.12: Correctness properties in LTL

We run the properties in SPIN to check the absence of errors and show the results of the successful evaluations in Table 4.1. The first column (States, stored) shows the number of unique states identified by the verification process (representing the actual size of the model in terms of states). The second column (States, matched) displays the number of repeated states, or states that have already been encountered. The combination of these two columns (stored + matched) provides the total number of distinct transitions executed, which can be used as a measure of the amount of work done to complete the verification. The final column displays the total memory used during each verification.

Table 4.1: Verification results of SPIN

Full statespace search		**States, stored**		**States, matched**		**Memory usage (MB)**	
		No losses	**Losses**	**No losses**	**Losses**	**No losses**	**Losses**
Model B	**No LTL**	478333	6868279	453290	4224732	58.390	838.413
	P1	815041	10808408	692337	6483193	111.929	1484.309
	P2	2496867	34606828	3646870	29482333	342.893	4752.525
	P3	696317	8454045	628802	5280674	90.312	1096.487
	P4	1198662	15651815	1043621	13331607	164.611	2149.450
Model A	**No LTL**	12106416		10814529		1293.105	
	Asserts	18499903		11939937		2117.146	

4.5 Performance Analysis of MTIP with UPPAAL

In this section, we present the performance properties of MTIP, the model developed in the UPPAAL tool and verification of the performance properties of the protocol.

4.5.1 MTIP Performance Properties

MTIP performance properties define the performance of the protocol according to application preferences. While using all sublinks to send duplicate data at all times is usually the most beneficial in terms of maximising some KPIs, especially those related to reliability, it would also incur in an excessive use of the resources that could affect both the network and the Tactile Internet devices. This trade-off can be separated in two performance properties:

- **P5**: The use of more sublinks tends to result in more correct *messages* received.
- **P6**: The use of more sublinks tends to result in more duplicate *packets* received.

In order to address this trade-off between resource consumption and the target KPIs, MTIP exposes an API where the application can indicate its preferences. MTIP uses this information, plus network measurements, to select which sublinks should be used to send data. In this section, we study the impact on the trade-off of using a different number of sublinks.

4.5.2 The UPPAAL Model

The UPPAAL model developed for MTIP is represented in Figure 4.7. The model is equivalent to the last model presented in PROMELA, with the main difference of not using additional variables to abstract time, since UPPAAL directly supports real-time clocks.

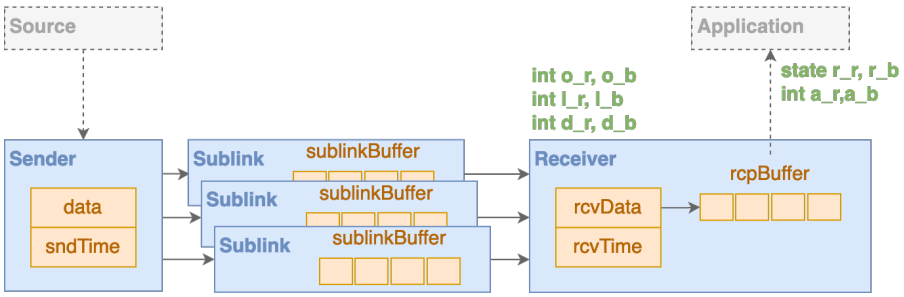


Figure 4.7: Simplified graphical view of the UPPAAL model

The `sender` automaton is shown in Figure 4.8. It is constituted by a main loop (bottom part) that continuously sends data through the different sublinks (indicated by the configurable value of `c_sblNum`). This data follows the Wolper test, which is directly implemented in the `sender` automaton, and includes a timestamp `sndTime` in each packet. In the top part of the automaton, we can see the functionality to reset the value of `sndTime` when the `TIMELIMIT` is reached in all parts of the model.

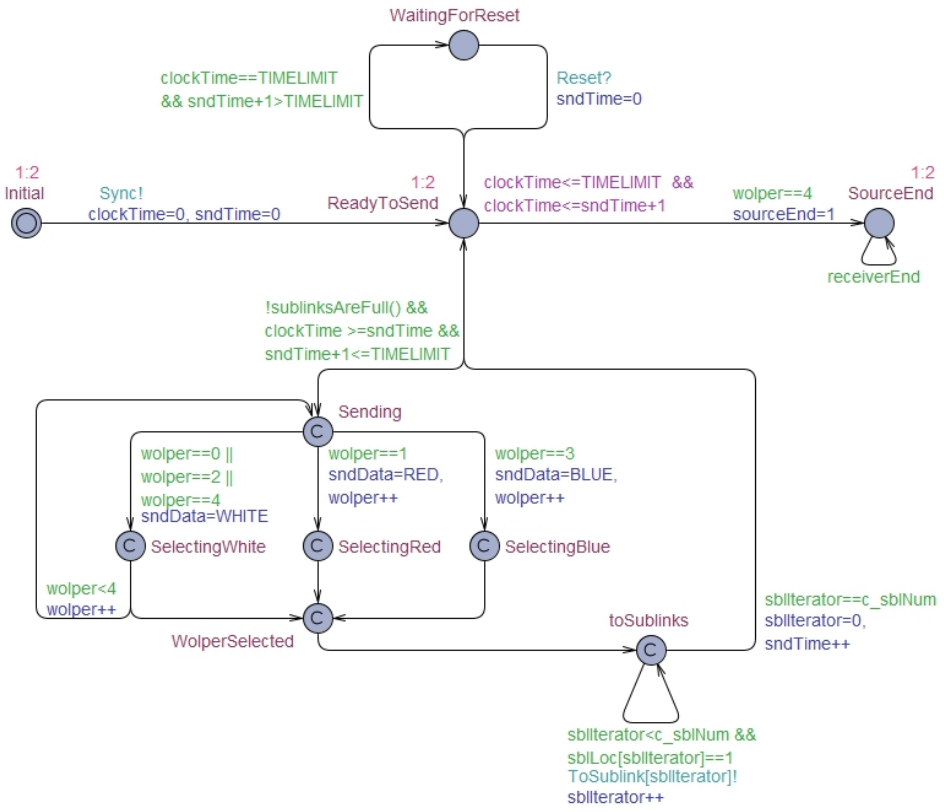


Figure 4.8: `Sender` automaton

The `sublink` automaton, presented in Figure 4.9, is responsible for simulating channel operation. It has two main locations, `Empty` and `notEmpty`. The `sublink` initiates synced with all the other automata and stays at the location `Empty`. When the `sender` wants to send some data to the other endpoint, it uses the `sublink` automaton, which receives the data and stores it with its timestamp in a circular buffer. Then, the `sublink` automaton progresses to the `notEmpty` location, where it can keep receiving data until it is full or has the opportunity to send the packet (data and timestamp) to the `receiver` automaton in a non-deterministic manner. The time it takes the automaton to leave this location is configured by the exponential

rate `c_lateTendency`, a variable that indicates the probability that the automaton will leave the location. The data can be successfully sent to the `receiver` automaton, or the packet can be lost during the sending process. The selection of how data are sent is also non-deterministic at first; however, a variable `c_lossTendency` has been added in order to control the losses in terms of probability and study the impact of the use of multiple sublinks. Note that variables such as `dstLoc` or `sblLoc` are necessary since UPPAAL SMC only supports broadcast channels for synchronization, as described in [186]. The `sublink` automaton also has an operation that forces discarding every packet that is left in the queues when the clock needs to be reset by the `receiver` endpoint, since those packets will have already reached the deadline.

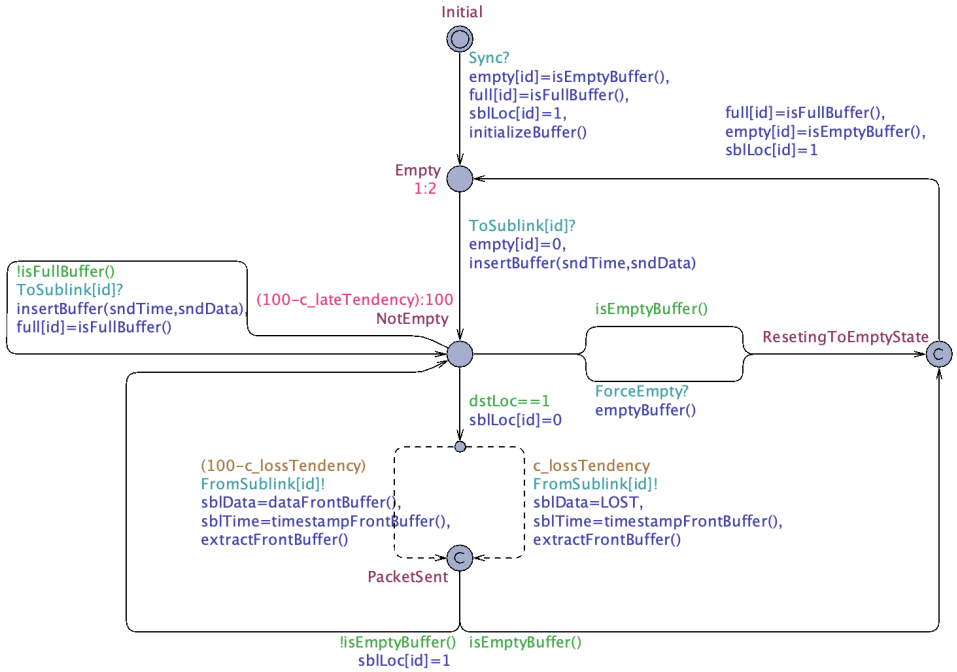


Figure 4.9: `Sublink` automaton

The `receiver` automaton (Figure 4.10) has the following tasks. First, it initiates the synchronization with all the automata (left part). Then, it waits for packets to be received from any sublink. When packets are received (bottom part), it checks if the packet is a duplicate, on time, late or lost and it only saves it in the buffer if it is an accepted packet according to the MTIP receiving algorithm explained in Section 4.4.1. Note that there are additional variables and labels in the form of location names that will be used to check formal properties in Section 4.5.3. The locations `r_r` and `r_b` check for red and blue data in the application while the variables `a_r` and `a_b` are used to count the amount of red and blue messages in the

application. Then, the variables `o_r`, `l_r` and `d_r` are used respectively to check the amount of on time, late or duplicate red data in the protocol, with the equivalent variables for blue data `o_r`, `l_r` and `d_r`. Finally, the `receiver` automaton is in charge of resetting the clock when it reaches the `TIMELIMIT` plus the maximum `DEADLINE` allowed for the packets (right-top part). This operation begins with the application consuming the buffer and saving relevant information for the formal validation. When this operation concludes, the clock is reset, along with other complementary variables. All automata keep working following the Wolper test.

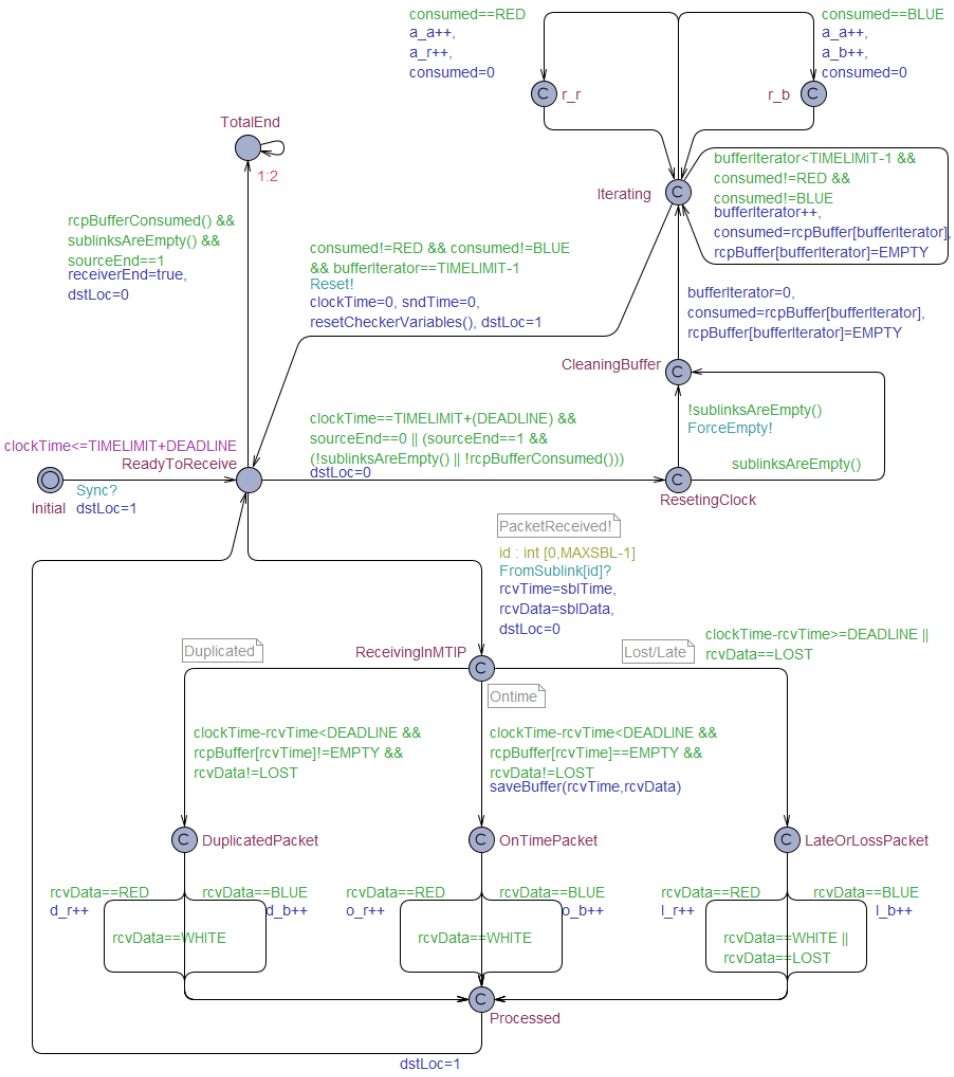


Figure 4.10: `Receiver` automaton

Listing 4.13 contains the values of the constants used in the model, variables for the formal verification (denoted by underscores) and the initial values of the configurable variables (starting with $c\_$) that will be used later to study the performance properties with UPPAAL SMC. The different functions shown in the model have self-explanatory names; however, the full code of the model can be found in [169].

```
1  //Constants (Wolper)
2  const int LOST = 0;
3  const int WHITE = 1;
4  const int RED = 2;
5  const int BLUE = 3;
6
7  //Constants (and limits)
8  const int TIMELIMIT  = 4;
9  const int MAXBUF = TIMELIMIT;
10 const int MAXSBL = 3;
11 const int DEADLINE = 3;
12 const int EMPTY = 0;
13
14 //Variables for formal verification
15 int a_r, a_b;
16 int o_r, o_b;
17 int l_r, l_b;
18 int d_r, d_b;
19
20 //Configurations for SMC
21 int c_sblNum=3;
22 int c_lossTendency=50;
23 int c_lateTendency=50;
```

Listing 4.13: Global declarations

4.5.3 Simulation and Formal Verification

We first run a simulation, shown in Figure 4.11, to check the model's expected behavior (i.e., Figure 3.5). Then, we proceed with the formal verification. Although the model is based on the previous models in PROMELA which have already been checked, we translate the correctness properties of MTIP to TCTL to assure that the behaviors of this new model correlates with the one of the protocol. In Listing 4.14, we show the properties in TCTL language (with `r` being the `receiver`) and in Table 4.2 the verification of the formulas. The translation of the properties is made to comply with the differences of the languages (e.g., A[] instead of []), and the limitations of the language used by UPPAAL, since `lead to` (`-->` or `A[](p imply A<> q)`) is the only TCTL operator allowed in UPPAAL with two nested temporal operators. Specifically, properties 2 and 4 are straightforward equivalent, but property 1 formula now expresses that if all packets arrive late, the application will not receive them, and the formula for property 3 checks that no red data is received in the application if blue data has been received.

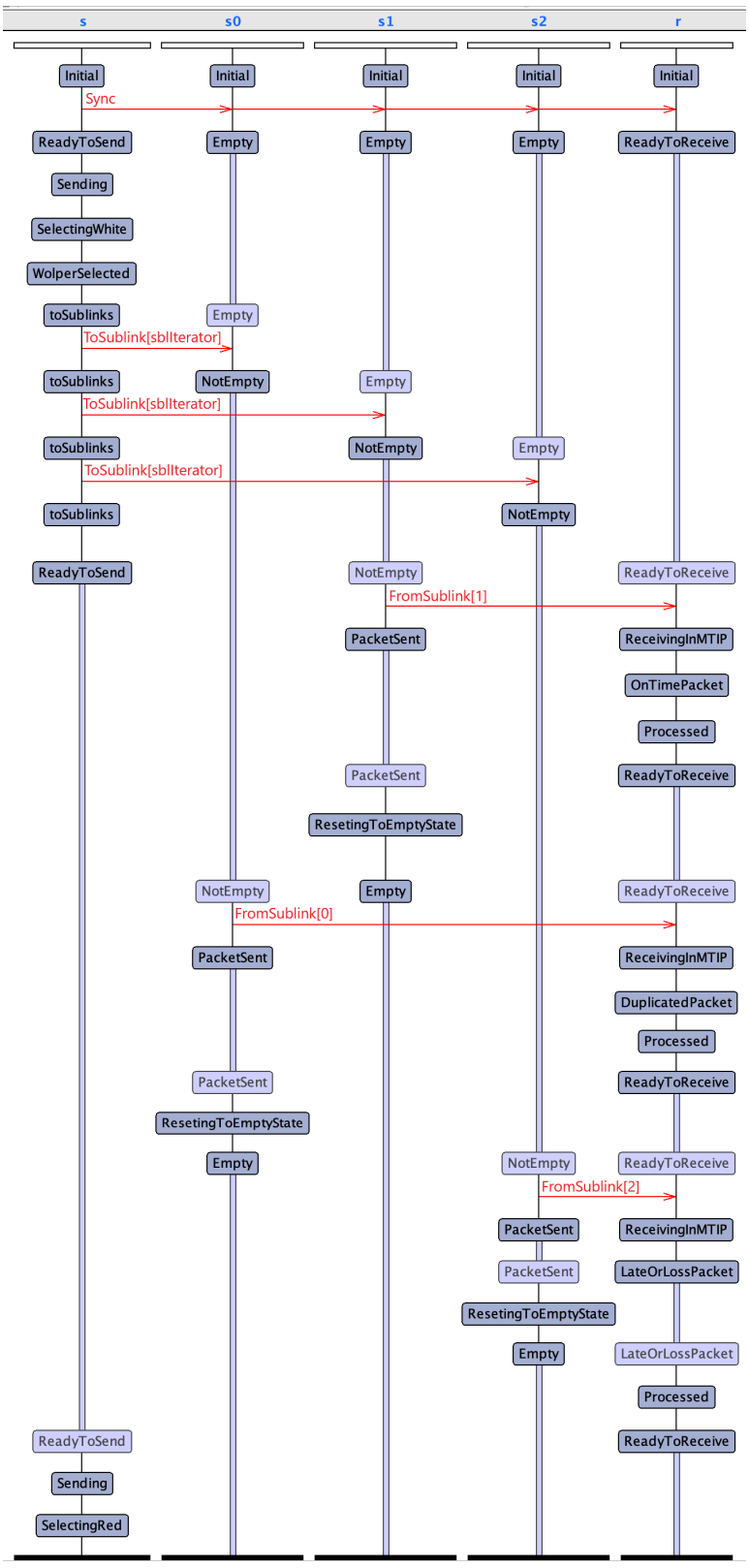


Figure 4.11: MSC extracted from UPPAAL

```
tctl P1       { A[] not ((l_r == c_sblNum && a_r > 0)
                      || (l_b == c_sblNum && a_b > 0)) }
tctl P2       { A[] ((a_r <= 1) && (a_b <= 1)) }
tctl P3       { A[] not (r.r_r && a_b>0) }
tctl P4_red   { (o_r == 1 && a_r == 0) --> r.r_r }
tctl P4_blue  { (o_b == 1 && a_b == 0) --> r.r_b }
```

Listing 4.14: Correctness properties in TCTL

Table 4.2: Verification results in UPPAAL

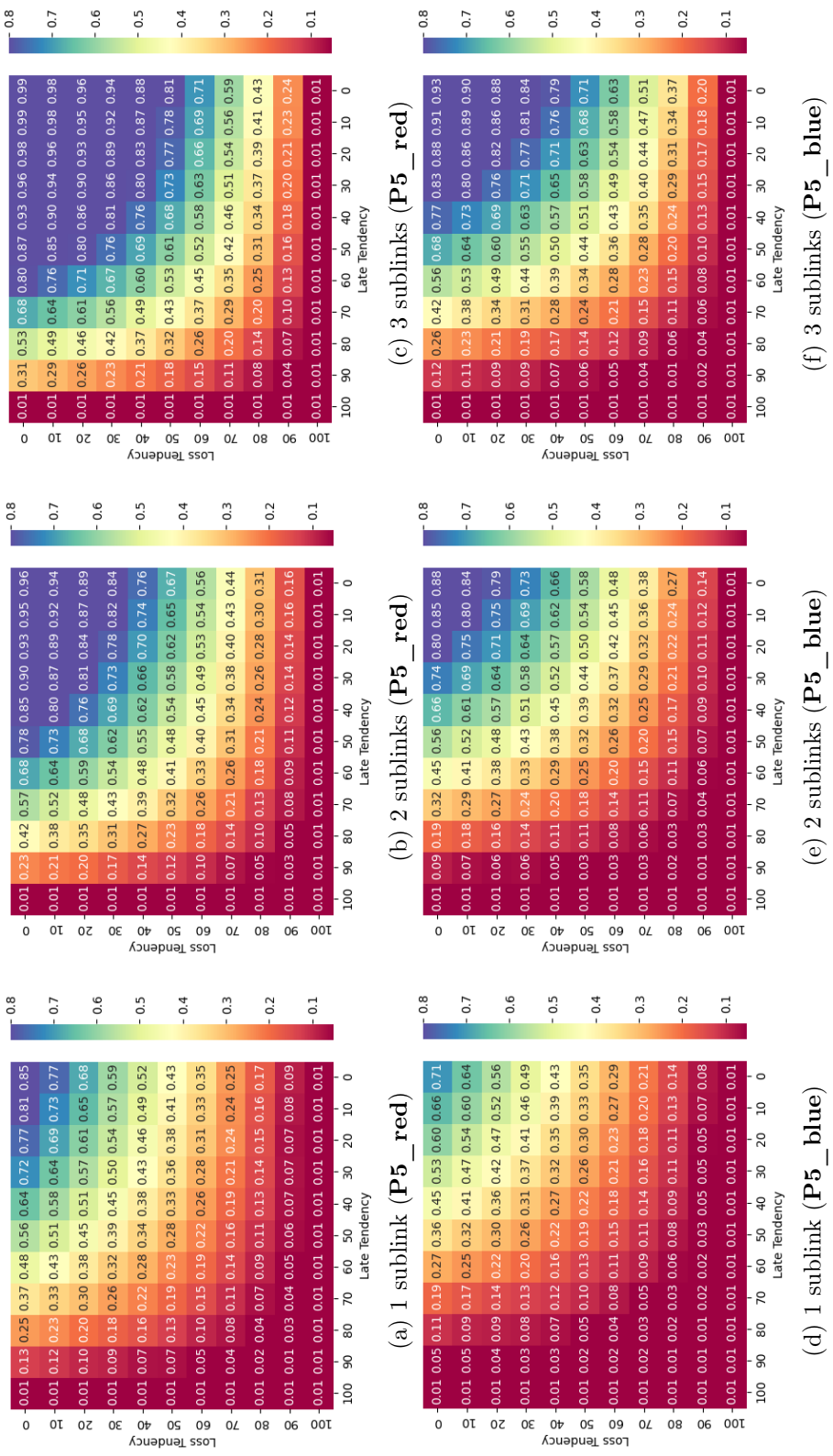
	States, stored	**States, matched**	**Memory usage**
General	42105198	133566430	7581.232 MB
P1	42105198	133566430	7588.732 MB
P2	42105198	133566430	7588.732 MB
P3	42105198	133566430	7588.944 MB
P4_red	42105198	227928967	10735.344 MB
P4_blue	42105198	235998863	10933.564 MB

After evaluating the correctness of MTIP with UPPAAL, we proceed to study the performance properties. We measure the impact of the configurable properties (i.e., `c_sblNum`, `c_lateTendency` and `c_lossTendency`), modifying the number of sublinks and the probability of losses and late packets. The first set of properties (**P5**) are meant to test the probability of receiving the messages successfully, while the second set (**P6**) measure the resource waste in terms of duplicate messages. The properties are shown in Listing 4.15.

```
tctl P5_red   { Pr[<=100] (<> a_r > 0) }
tctl P5_blue  { Pr[<=100] (<> a_b > 0) }
tctl P6_red   { Pr[<=100] (<> d_r > 0) }
tctl P6_blue  { Pr[<=100] (<> d_b > 0) }
```

Listing 4.15: Performance properties in TCTL

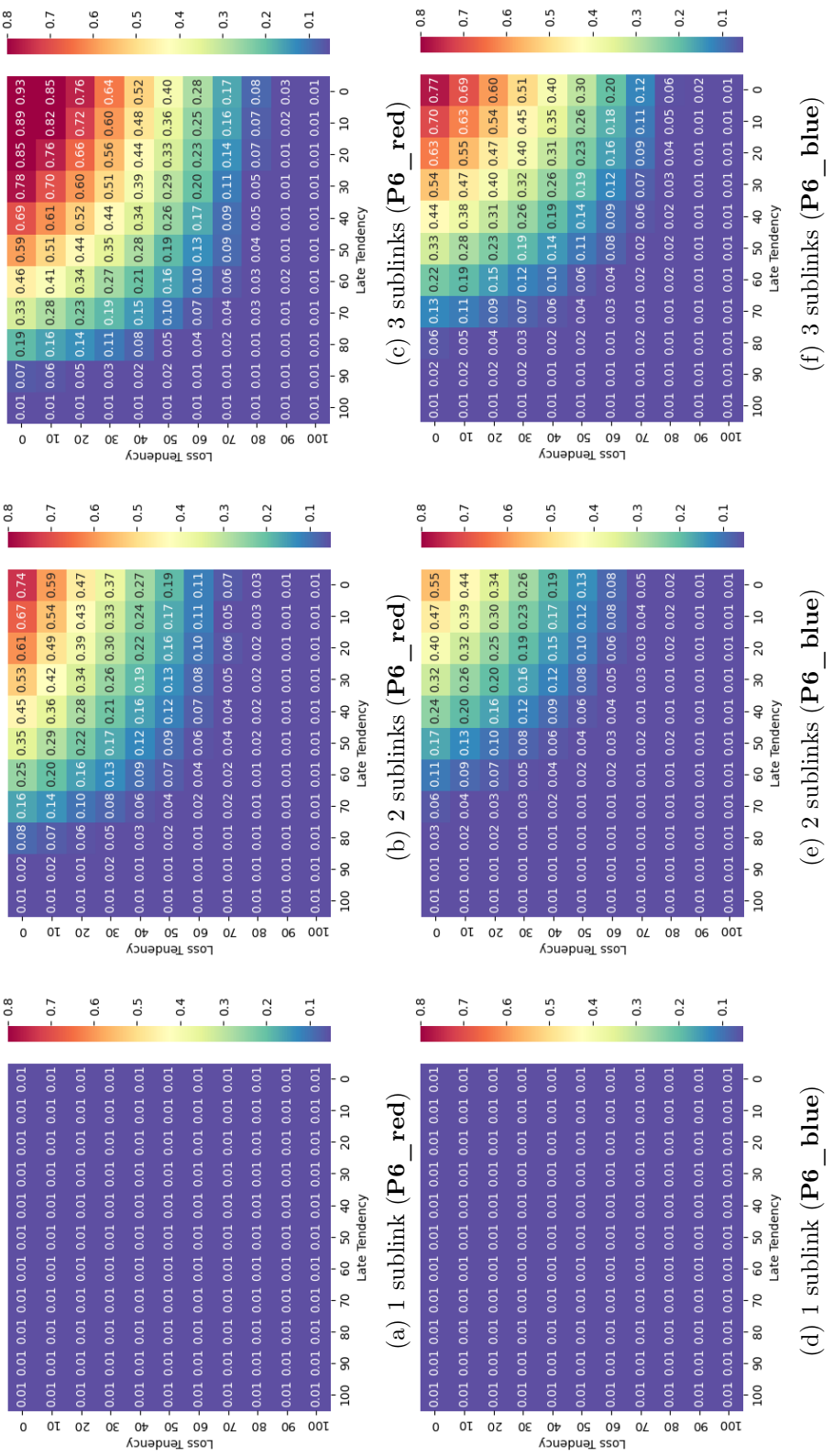
We evaluate the use of 1, 2 or 3 sublinks under configurations of 0 to 100 loss and late tendencies (in 10-sized steps). The heatmaps displaying the results of the first property evaluation can be found in Figure 4.12, and the heatmaps for the second property evaluation can be found in Figure 4.13. These graphs present the average results of validating the properties in a time that assures that the completion of the experiment (`Pr[<=100] (<> receiverEnd == 1)`) occurs in [0.990031,1] with confidence 0.99. To facilitate comparison between the two sets of heatmaps, blue has been used to indicate a positive outcome for the performance property, while red indicates a negative outcome.



(a) 1 sublink (**P5_red**) (b) 2 sublinks (**P5_red**) (c) 3 sublinks (**P5_red**)

(d) 1 sublink (**P5_blue**) (e) 2 sublinks (**P5_blue**) (f) 3 sublinks (**P5_blue**)

Figure 4.12: Evaluation of performance property **P5**



(a) 1 sublink (**P6_red**) (b) 2 sublinks (**P6_red**) (c) 3 sublinks (**P6_red**)

(d) 1 sublink (**P6_blue**) (e) 2 sublinks (**P6_blue**) (f) 3 sublinks (**P6_blue**)

Figure 4.13: Evaluation of performance property **P6**

By evaluating the performance properties, we can extract the following conclusions. First, in performance property **P5**, the colours of the graphs suggest that the use of more sublinks enhances the communication. More green and blue colours can be seen with two and three sublinks. However, if the scenario is poor (see `LossTendency 100%` and `LateTendency 100%`) the communication cannot be improven by the use of more sublinks, since all sublinks in use appear to be faulty. Performance property **P6** shows an analogous behavior. In this case, we see that using more sublinks generally worsen the property. This is due to the fact that using more sublinks usually reflects in receiving more duplicates and having more resource waste. The trade-off of the performance properties of MTIP resides then in finding a sufficient configuration for the application requirements. For instance, if the requirements set a minimum of 0.9 probability of arrival of a red data under a scenario of 0% loss and 20% late tendencies, the graphs shown in Figure 4.12 displays that this can be achieved with two and three sublinks. However, if the the maximum resource waste is set to 0.7, the graphs in Figure 4.13 show that the use of two sublinks is the only one that can comply with both requirements, even if the use of three sublinks provides more reliability.

Chapter 5

Implementation and Evaluation of MTIP

CONTENTS

Summary: This chapter presents the implementation and evaluations performed on MTIP, which were conducted in both a simulated environment and a real 5G testbed at University of Malaga [187]. The chapter also details the use cases and scenarios under test and provides an analysis of the results obtained.

5.1 Implementation

An MTIP link begins in the CLOSED state. Once the sublinks have been bounded and the link handshake has been completed, it enters the LINKED state, where the data exchange can start. At the end of a transport session, the MTIP link is closed, which is done similarly to TCP, i.e., via a FIN_WAIT state.

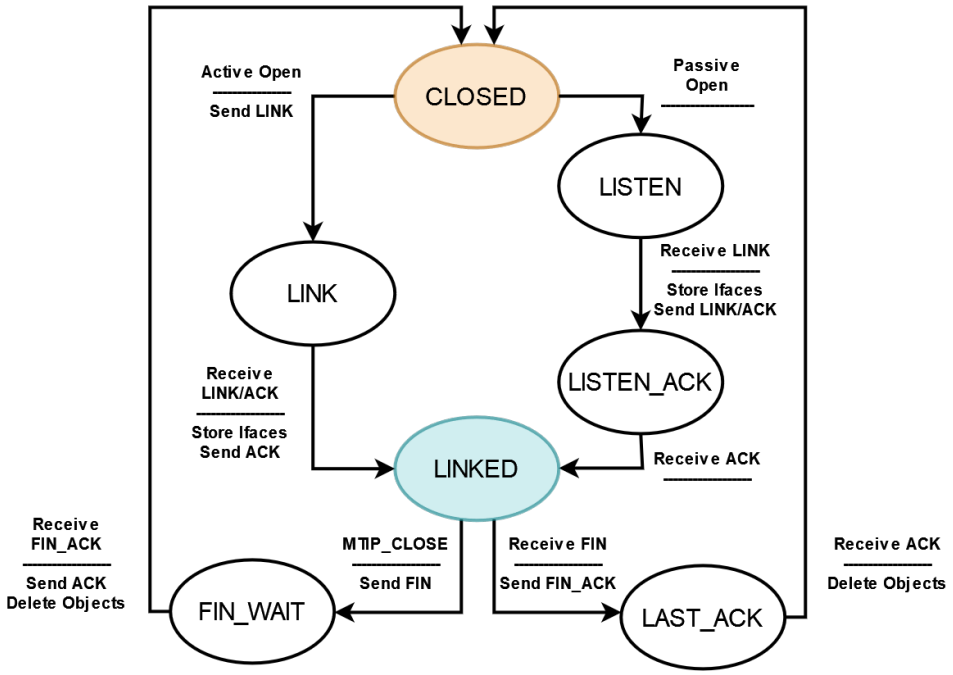


Figure 5.1: The MTIP state machine

The implementation structure of MTIP is shown in Figure 5.2 and can be summarized as follows. When a link has been established and both endpoints have synchronized their information about the sublinks, the data plane uses the different data sublinks (DS) to send the application data (following the algorithm presented in Section 3.3 with the application preferences presented in Section 3.4). The control plane or monitoring entity (MTICP) uses different ports to monitor the status of the sublinks and sends *Keep Alive* messages through control sublinks (CS) when the data sublinks are idle, to keep network KPIs up to date.

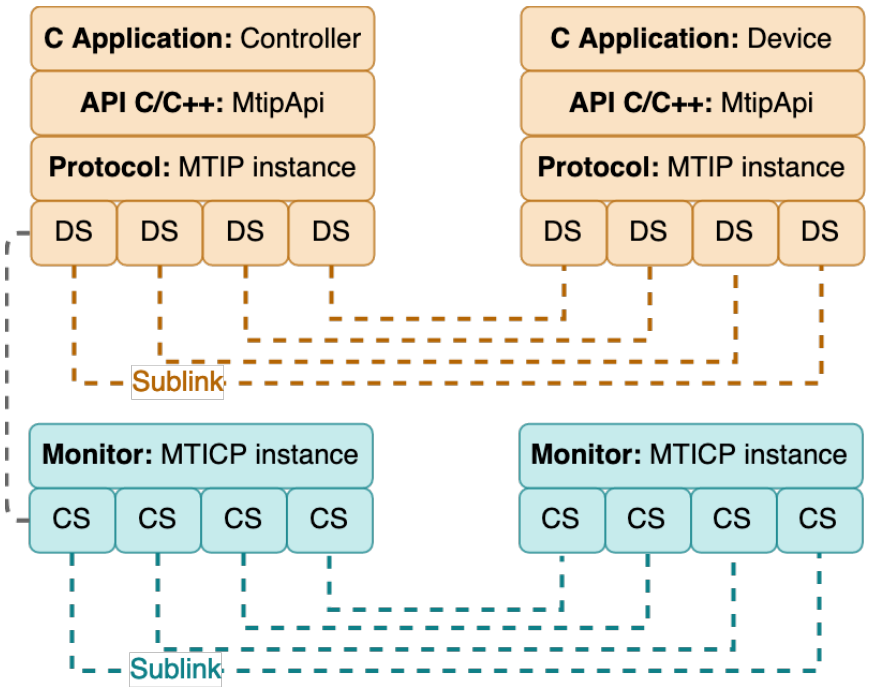


Figure 5.2: MTIP implementation structure

Figures 5.3, 5.4, 5.5 and 5.6 present the interconnection of these classes with the main MTIP data structures. It is important to note that all classes inherit from QObject, since they rely on Qt inter-process communication [188]. Moreover, they also inherit from Debug, a class developed to show debug messages in a cohesive way. An example of a MTIP log is provided in Figure 5.7. We observe similarities between MTIP and MTICP classes. Both classes execute in parallel and use communication preferences and protocol configurations to tune their operation. These similarities are also shown between DS and CS classes, since both are used to send packets through different interfaces. DS are managed by MTIP to send information and are in charge of measuring and storing the status of the sublink, while CS are managed by MTICP and make use of the application preferences stored in MTIP class to send KPA packets.

5.1.1 Context awareness

The code developed also implements the API presented in Section 3.4.3 and detailed in Appendix B. Table 5.1 lists the communication preferences that the application can expose, a short description (further information in Section 3.4.2), the accepted values and the specific default values defined in current implementation. Table 5.2 enumerates the characterization information that the application can retrieve from MTIP in current implementation.

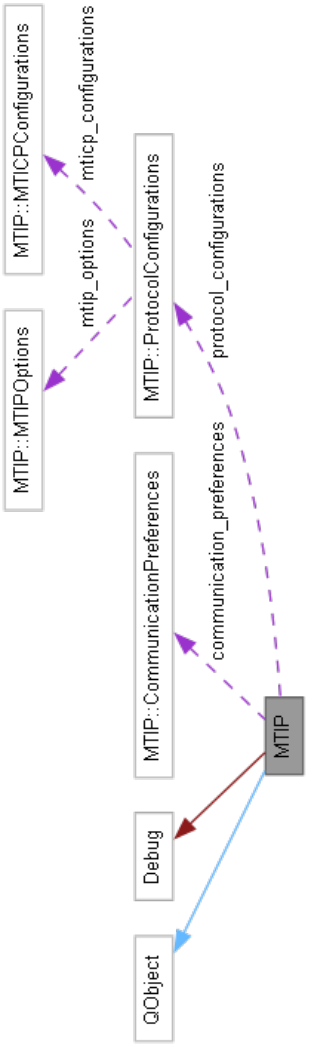


Figure 5.3: MTIP collaboration diagram

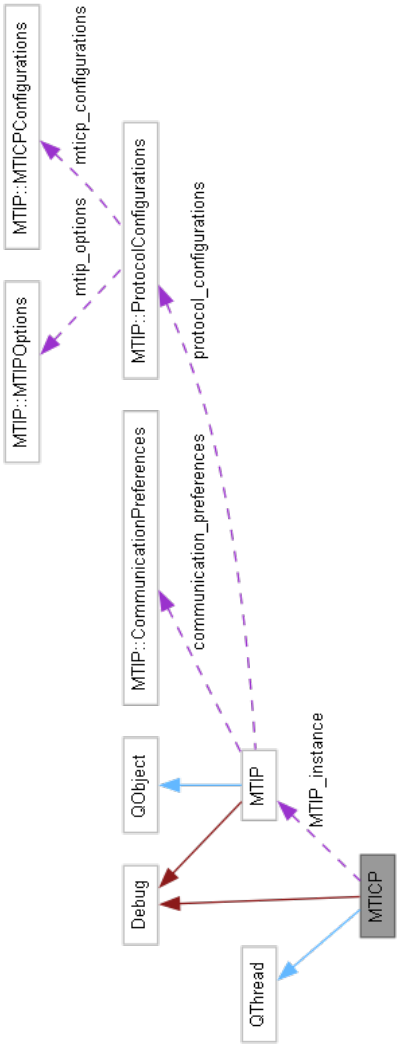


Figure 5.4: MTICP collaboration diagram

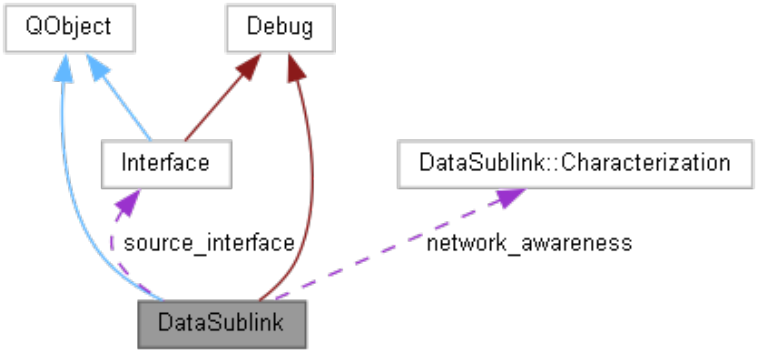


Figure 5.5: DS collaboration diagram

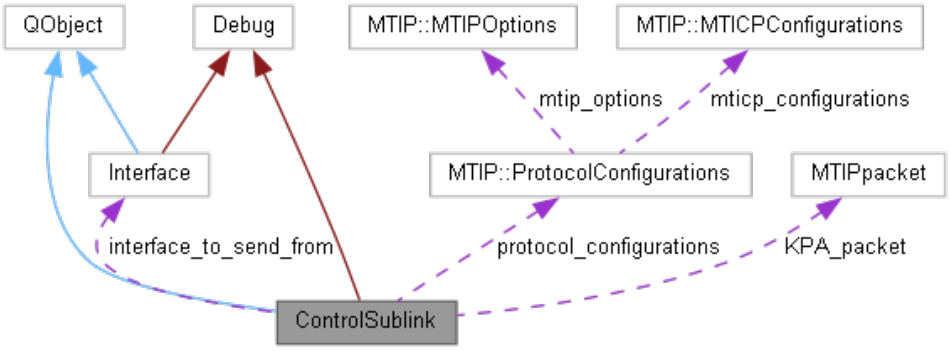


Figure 5.6: CS collaboration diagram

```
[MTIP] ---------- MTIP Initialization
[MTIP] ---------- BIND Interface
[PATH:INTERFACE] - Bind:  "10.11.64.193" 3333
[MTIP] ---------- BIND Interface
[PATH:INTERFACE] - Bind:  "10.11.64.207" 3333
[MTIP] ---------- BIND Interface
[PATH:INTERFACE] - Bind:  "10.11.64.187" 3333
[MTIP] ---------- BIND Interface
[PATH:INTERFACE] - Bind:  "10.11.80.1" 3333
[MTIP] ---------- Preferences Updated
[MTIP] ---------- LINK Starting
[MTIP] ---------- LINK active
[MTIP] ---------- Created -> [SeqNum]  0  [Timestamp]  1667465417740574590  [Flag]
FLAG_LINK  [Payload]  "[\n    {\n        \"CONTROL_PORT\": 4444,\n        \"IP\": \"10.11.64.
193\",\n        \"PORT\": 3333\n    },\n    {\n        \"CONTROL_PORT\": 4444,\n
\"IP\": \"10.11.64.207\",\n        \"PORT\": 3333\n    },\n    {\n        \"CONTROL_PORT\":
4444,\n        \"IP\": \"10.11.64.187\",\n        \"PORT\": 3333\n    },\n    {\n
\"CONTROL_PORT\": 4444,\n        \"IP\": \"10.11.80.1\",\n        \"PORT\": 3333\n    }\n]\n"
[PATH:INTERFACE] - Sent from:  "10.11.64.193" 3333  to: "10.11.23.5" 3333
[PATH:INTERFACE] - Sent from:  "10.11.64.207" 3333  to: "10.11.23.5" 3333
[PATH:INTERFACE] - Sent from:  "10.11.64.187" 3333  to: "10.11.23.5" 3333
[PATH:INTERFACE] - Sent from:  "10.11.80.1" 3333  to: "10.11.23.5" 3333
[PATH:INTERFACE] - Received Message from:  "10.11.23.5" 3333  to: "10.11.64.193" 3333
[MTIP] ---------- Received LINK/ACK, sending ACK
[MTIP] ---------- Received -> [SeqNum]  0  [Timestamp]  1667465418730095998  [Flag]
FLAG_LINKACK  [Payload]  "[\n    {\n        \"CONTROL_PORT\": 4444,\n        \"IP\": \"10.11.
23.5\",\n        \"PORT\": 3333\n    },\n    {\n        \"CONTROL_PORT\": 4444,\n
\"IP\": \"10.11.28.5\",\n        \"PORT\": 3333\n    },\n    {\n        \"CONTROL_PORT\": 4444,
\n        \"IP\": \"10.11.64.5\",\n        \"PORT\": 3333\n    },\n    {\n
\"CONTROL_PORT\": 4444,\n        \"IP\": \"10.11.66.5\",\n        \"PORT\": 3333\n    }\n]\n"
[PATH:DATASBL] --- Creation of new Data Sublink  "10.11.64.193" 3333 "10.11.23.5" 3333
[PATH:DATASBL] --- Creation of new Data Sublink  "10.11.64.207" 3333 "10.11.28.5" 3333
[PATH:DATASBL] --- Creation of new Data Sublink  "10.11.64.187" 3333 "10.11.64.5" 3333
[PATH:DATASBL] --- Creation of new Data Sublink  "10.11.80.1" 3333 "10.11.66.5" 3333
[MTIP] ---------- Calculating latency: [Received   1667465418730095998 ] [Timenow
1667465418736453004 ] [Latency  6357248 ] [ 6.35725e+06  ;  -6.35725e+06 ]
[MTIP] ---------- Created -> [SeqNum]  0  [Timestamp]  6357248  [Flag] ACK
[PATH:DATASBL] --- Sending...
[PATH:INTERFACE] - Sent from:  "10.11.64.193" 3333  to: "10.11.23.5" 3333
[PATH:DATASBL] --- Sending...
[PATH:INTERFACE] - Sent from:  "10.11.64.207" 3333  to: "10.11.28.5" 3333
[PATH:DATASBL] --- Sending...
[PATH:INTERFACE] - Sent from:  "10.11.64.187" 3333  to: "10.11.64.5" 3333
[PATH:DATASBL] --- Sending...
[PATH:INTERFACE] - Sent from:  "10.11.80.1" 3333  to: "10.11.66.5" 3333
[MTIP] ---------- LINK completed
```

Figure 5.7: Example of MTIP debugging log

Table 5.1: Communication preferences

Communication preference	**Description**	**Accepted values in current implementation**	**Default value in current implem.**
Algorithm mode (AM)	It selects the algorithm that is going to be used for the selection of sublinks. The sublinks might be selected by the application (fixed selection) or dynamically by the protocol.	0: Use all sublinks (fixed selection) 1: Use one sublink (fixed selection) 2: Use best (N) sublinks (dynamic selection) 3: Use the MTIP Algorithm (dynamic selection)	3
Number of sublinks (N)	It selects the sublink that is in use (in the case of AM 1) or the number of sublinks that should be in use (in the case of AM 2).	0 to the maximum number of sublinks available	0
Maximum latency (deadline)	It defines the maximum end-to-end latency of the packets, namely the deadline.	0 to 1e9 nanoseconds (ns)	1e7 ns
Duplicate threshold (DT)	It defines the maximum percentage of duplicate packets that the MTIP algorithm considers reasonable (only used in AM 3).	0 to 100 %	50
Loss-late threshold (LT)	It defines the maximum percentage of loss or late packets that the MTIP algorithm considers reasonable(only used in AM 3).	0 to 100 %	10
Latency weight (LW)	It defines how the sublink ranking must be calculated, using just reliability measurements (LW 0), using only latency measurements (LW 100), or using a weighted mean of both measurements (LW from 0 to 100).	0 to 100 %	100

Table 5.2: Feedback information

Feedback	**Description**	**Value**
General	Network information, measured latency (ingress, egress) and measured reliability (ingress, egress) for each sublink.	Network information (sublink ID, IP and ports), latency (nanoseconds) and reliability (% of packets successfully forwarded to the application).
Sending	Status of each packet sent.	Packet on time, duplicate, late or ack not received.
Receiving	Status of each packet received.	Packet on time, duplicate, late or ack not received.

5.2 Evaluation in a Simulation Environment

First, we study the impact of different configurations of the MTIP algorithm in a simulation environment. This environment comprises a virtual topology generated by the Mininet tool [38] and different scenarios impaired with Netem rules with the tc tool [39].

5.2.1 Topology and Scenarios

The topology under test consist on two endpoints connected by four sublinks as presented in previous Figure 3.2. Packets are sent at the typical Tactile Internet rate of 1kHz [1] with a latency limit of 10ms. We study the impact of the main parameters of the MTIP algorithm, i.e. the `loss-late threshold`, the `duplicate threshold` and the `latency weight` (as previously presented in Figure 3.6 of Section 3.3). We measure the impact on the main trade-off of the algorithm: the reliability, counting lost and late packets, versus the resource waste, in terms of duplicate packets. In order to showcase this trade-off, we have created the metric `trade-off` which shows a lower value when the case is more beneficial. The metric is configurable, in this case we have considered the values of the trade-off metric with $Weight = 0.05$. This value represents the importance of wasting resources, shown by the number of duplicate packets, set to a 5% compared to the importance of receiving correct messages. However, the decision on the concrete value of the metric would depend on the specific target application and the importance of reliability versus network overload that the developer assigns to that case.

$$Trade\text{-}off = \frac{Losses + (Duplicates * Weight)}{1 + Weight}$$

We consider three scenarios that differ in the choice of the impaired KPI:

- **Delay scenarios** emulate delays in the lower layers of the protocol stack. These delays could be caused by different configurations, such as the acknowledged mode of the Radio Link Control protocol (RLC) [189], which provides higher reliability at the cost of longer delays. Delay scenarios explore cases with significant delay variation and no losses.

- **Loss scenarios** emulate lower-layer losses, e.g., losses caused by the unacknowledged mode of RLC. Loss scenarios explore cases with losses but with no significant delay variations.

- **Mixed scenarios** are the most realistic of the three types of scenarios. They present the case of having both delay variation and losses caused by balanced configurations of lower layers.

All scenarios change their conditions every second. However, we create three variants of each scenario: one with good conditions on all paths, one with dynamic conditions on all paths, i.e., paths that change from good to poor conditions and vice versa, and one with poor conditions on all paths, in order to showcase a extremely bad situation. To have some reference values on the worst and best-case scenarios, we characterize the scenarios using MTIP over the paths separately, showing the average value, and then using MTIP redundantly on all paths simultaneously. We show the results of this characterization in Table 5.3. It is important to note that the percentage shown in the table is the amount of late and lost packets that do not reach the application layer. In delay scenarios, the percentage represents the fraction of packets whose transfer time is larger than 10 ms, typical TI deadline [1]; in loss scenarios, the percentage denotes actual packet losses; while in mixed scenarios, the percentage is a combination of packets that are late and lost.

Table 5.3: Characterization of the testing scenarios

	Delay Scenarios			**Loss Scenarios**			**Mixed Scenarios**		
	Good	**Dyn.**	**Poor**	**Good**	**Dyn.**	**Poor**	**Good**	**Dyn.**	**Poor**
One Path	0.12%	9.72%	16.30%	0.86%	3.89%	10.52%	1.92%	11.58%	25.02%
All Paths	0.01%	0.02%	0.02%	0.00%	0.01%	0.02%	0.07%	0.06%	0.89%

5.2.2 Results and Discussion

5.2.2.1 Loss-late and Duplicate Thresholds

In Figure 5.8, the loss-late threshold is modified with a fixed duplicate threshold in the top part, and then, the duplicate threshold is modified with a fixed loss-late threshold in the bottom part.

The behavior of the modification of the two thresholds is analogous, with more restrictive thresholds (low loss-late thresholds or high duplicate thresholds) causing causing less losses at the cost of more redundancy and vice-versa. In terms of the trade-off, we observe that, in general, if the scenario is good, it benefits from reducing duplicates, while in worse scenarios, it depends on the specific case. For instance, in the mixed scenario (poor), both 0% and 5% loss-late threshold configurations are considered better than the 10% configuration. This is because the scenario is so impaired that setting a 10% configuration leads to a disproportionate increase in lost-late packets, outweighing the reduction in duplicates.

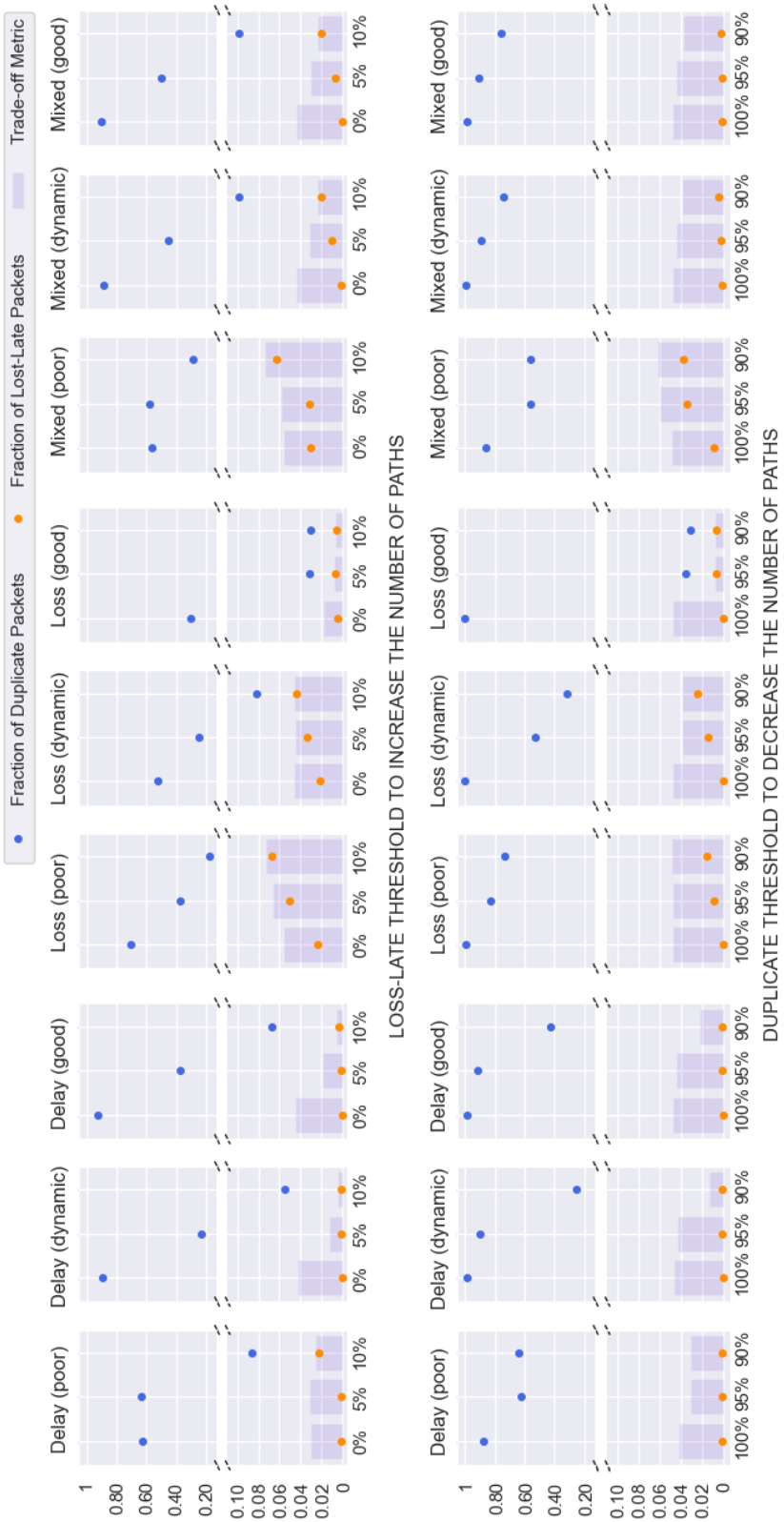


Figure 5.8: Effect of the thresholds in different scenarios

5.2.2.2 Latency Weight

In Figure 5.9, we fix the values of the loss-late and duplicate thresholds to have a scenario prone to losses and evaluate the effect of the latency weight parameter. The results indicate that selecting sublinks based on latency alone (100%) or a combination of latency and reliability is more advantageous than solely considering reliability (0%). Since reliability is already improved by using multiple sublinks, it is usually beneficial to include the latency KPI as well in the selection of the sublinks.

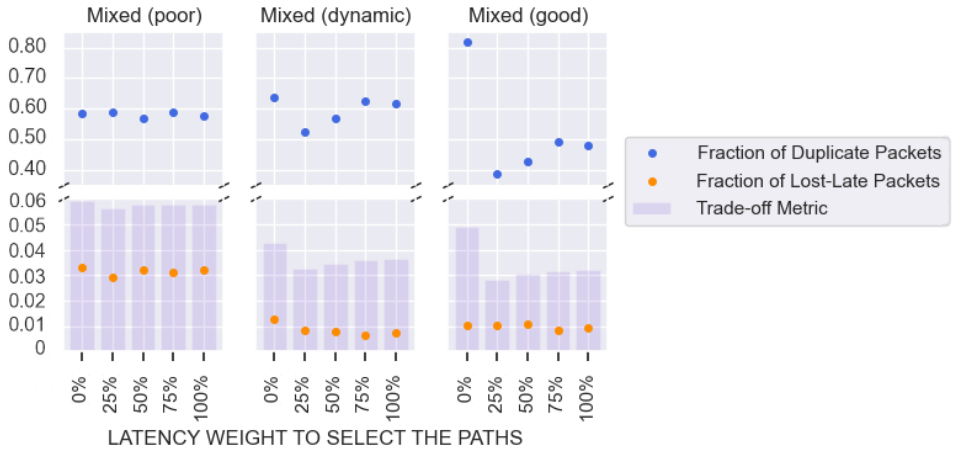


Figure 5.9: Effect of the latency weight on lost-late and duplicate packets

In Figure 5.10, we can see why this effect has a larger impact in good scenarios. In situations where conditions are less favorable, more sublinks tend to be used, making the specific ranking less crucial.

5.3 Evaluation in a Real Environment

In this section, we present a more realistic evaluation in a 4G/5G testbed with a real industrial robot in order to check the validity of our previous simulated evaluation. First, we describe the testbed, then the scenario under evaluation and finally, the results and discussion.

5.3.1 5G Testbed and Topology

For our evaluations, we make use of the experimental platform for 5G and 6G mobile networks Victoria Network [187], located at the University of Malaga. The platform is managed by the MORSE group at the Institute of Technology and Software Engineering of the University of Malaga (ITIS) and is a key European reference for the success of 16 projects of the FP7, H2020 and Horizon Europe programmes, as well as support for numerous

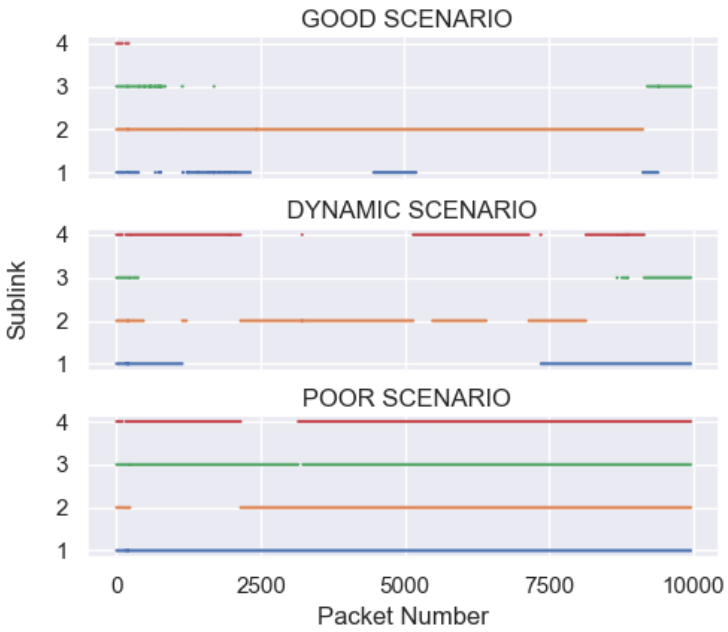


Figure 5.10: Sublinks selected during a communication in different scenarios

national projects. Currently, this infrastructure covers part of the Teatinos campus of the University of Malaga and has connections to different areas of the province of Malaga thanks to agreements with Telefonica and Malaga City Council. Victoria Network consists of all the elements to create end-to-end private 5G networks with commercial and experimental equipment, including terminals, base stations, virtualized 5G cores, application servers, measurement instruments, and the capacity to create SIM cards, among other features.

The architecture of the testbed can be abstracted in two main layers, the physical infrastructure layer and the management and orchestration layer. The physical infrastructure layer involves the the end-to-end components and the underlying network. In our evaluation, these are constituted by the end devices, multi-radio access technologies modems, a multi-probe anechoic chamber and multiple antennas and core networks with synchronization and security capabilities. These network elements are displayed in Figures 5.11 and 5.12. The management and orchestration layer enables the management of virtualized elements and these traditional network systems, configuring independent connections through shared components. Specifically, it allows the creation of the four differentiated sublinks that will be used in our tests: 4G, 5G NSA, 5G NSA with special QoS class identifier priority and 5G SA.

Figure 5.11: Antennas and core networks used for our evaluation

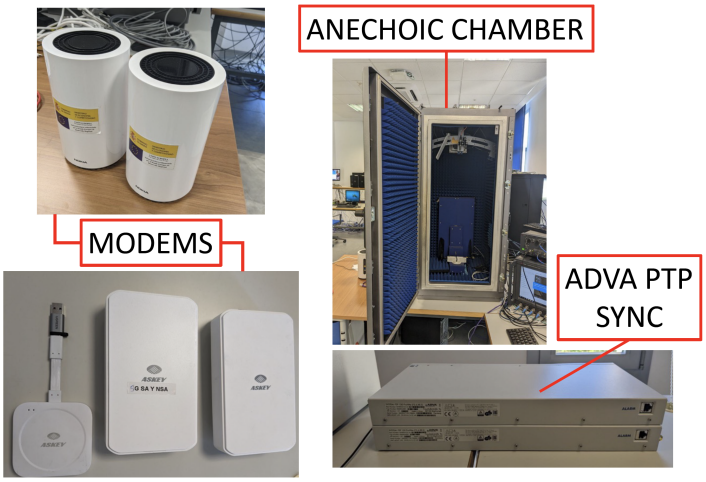


Figure 5.12: Additional elements used for our evaluation

5.3.2 Use Case: Remote Control of an Industrial Robot

The case study in our tests represents a Tactile Internet remote control of an industrial robot that needs to move within a specific time frame in coordination with other robots on the assembly line. It must work reliably but always within a time frame, as any movement after a deadline can lead to accidents or damage to important devices. The data characteristics of this application are similar to those of the emulated evaluation, where an order to move the robot is sent with a 10 ms deadline. In our evaluations, we use the Husky Clearpath [190] shown in Figure 5.13. This robot runs the the

Robot Operating System (ROS) [191] to send and control the commands, so a proxy setup has been created to encapsulate ROS messages over MTIP for the communication. Figure 5.14 shows the resulting communication setup.

Figure 5.13: The Husky Clearpath robot used in the testbed evaluation

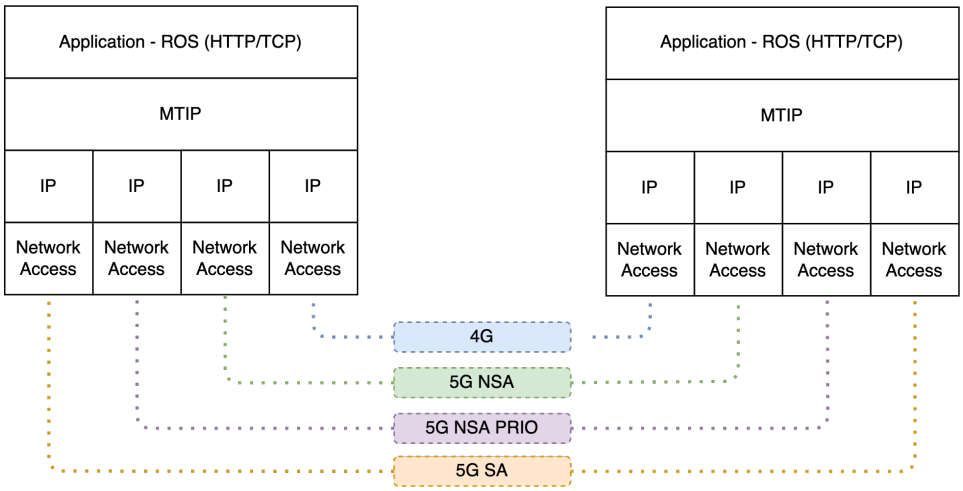


Figure 5.14: Communication setup between the controller and the robot

The scenarios and measurements used to evaluate MTIP in the real environment present a similar approach as in the emulated environment. Regarding the scenarios, there is **regular scenario** that introduces the normal operation of the network without any kind of additional impairments. Since the network has stably low losses but variability in the delay, this scenario is comparable to the previous *delay scenarios* of the emulated evaluation. Then, there is a **impaired scenario** which is more comparable to the *mixed scenarios*. This scenario includes additional losses impairments generated by a multi-Probe Anechoic Chamber to create more challenging radio conditions and replicated for multiple tests with the network emulation tool `netem` [39]. Both scenarios are dynamic, since a real network always presents certain variability. The characterization of the scenarios are presented in Figures 5.15 and 5.16.

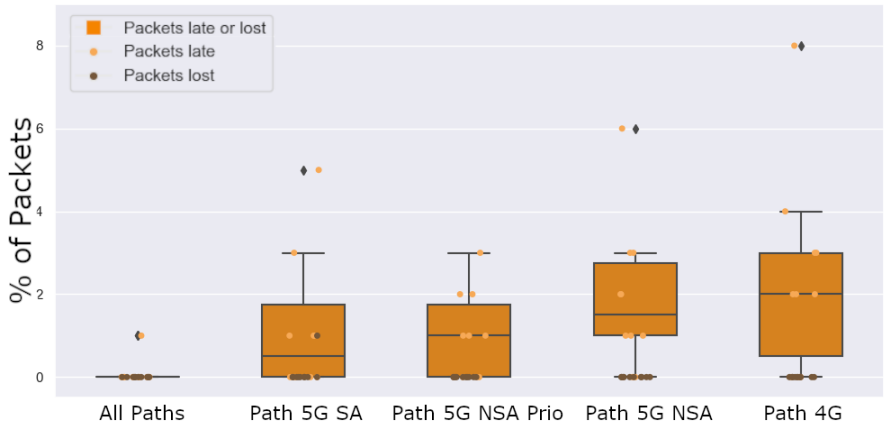


Figure 5.15: Characterization of the regular scenario

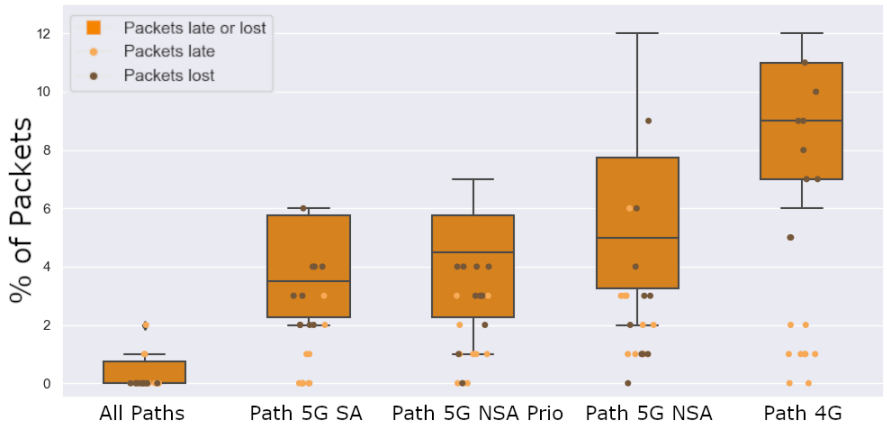


Figure 5.16: Characterization of the impaired scenario

We evaluate the performance of MTIP at the transport and application layers. The transport layer measurements are similar to those conducted in the emulated environment, assessing the impact of MTIP's algorithm parameters on duplicates, late and lost packets, and the trade-off metric. At the application layer, accuracy is measured in terms of the percentage of robot actions correctly processed by the robot, while efficiency is measured through the trade-off metric. The combination of both measurements provides a comprehensive analysis of the most favorable cases. The results of the evaluation are provided in the following subsection.

5.3.3 Results and Discussion

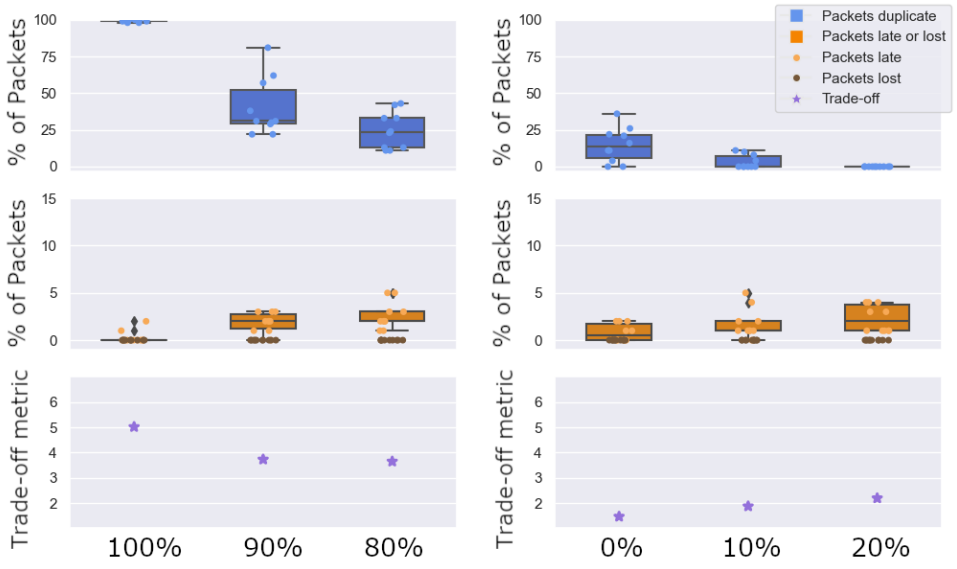
5.3.3.1 Loss-late and Duplicate Thresholds

Figures 5.17a and 5.17b show the number of duplicates and lost or late packets when modifying the thresholds in the regular scenario. The behavior is similar to that observed in the emulated scenario, where an increase in duplicate packets leads to a decrease in loss-late packets due to redundancy, and vice versa. As the scenario is not ideal, some redundancy shows to be beneficial. Both evaluations indicate that keeping duplicate packets at around 20/25% can be advantageous (i.e., the case of 80% Duplicate threshold and the case of 0% Loss-late threshold). In the impaired scenario (see Figures 5.18a and 5.18b), decreasing the number of duplicate packets too much in both evaluations leads to a higher and less desirable trade-off metric (such as with an 80% Duplicate threshold or a 20% Loss-late threshold). In this scenario, having a higher number of duplicate packets (around 50%) appears to be more beneficial since the scenario is more susceptible to losses and can benefit from a higher level of redundancy.

5.3.3.2 Latency Weight

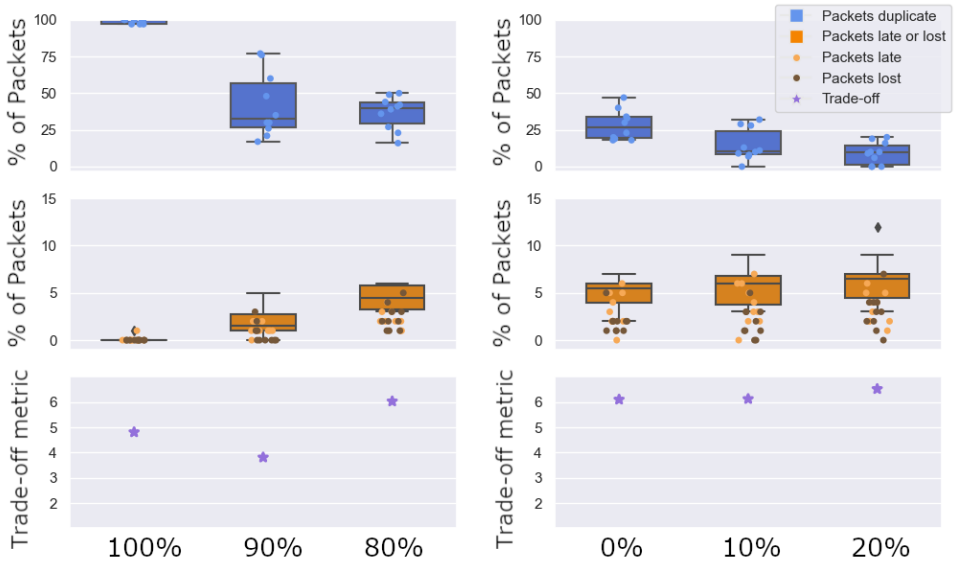
The result of the evaluation of the latency weight presented in Figure 5.19 shows that the different configurations of the parameter do not greatly affect the trade-off metric in general. In the impaired scenario, using latency measurements (25%-100%) can reduce the number of duplicate packets, but this does not enhances significantly the trade-off metric since it incurs in few, but some losses that also impact in the metric. Moreover, as shown in Figure 5.20, the impaired scenario tends to use more sublinks, so the selection of the latency weight and thus, the ranking of sublinks is not critical in this case. In the regular scenario, the difference is more noticeable in the trade-off metric, showing that using only reliability measurements (0%) can be less convenient than using latency measurements as well. However, since the scenario is dynamic and the losses and late packets are not numerous and can occur in any of the sublinks, once a sublinks is generally stable, the

ranking is no longer as critical. This is shown in Figure 5.21, which presents examples of different iterations of the tests in the regular scenario.



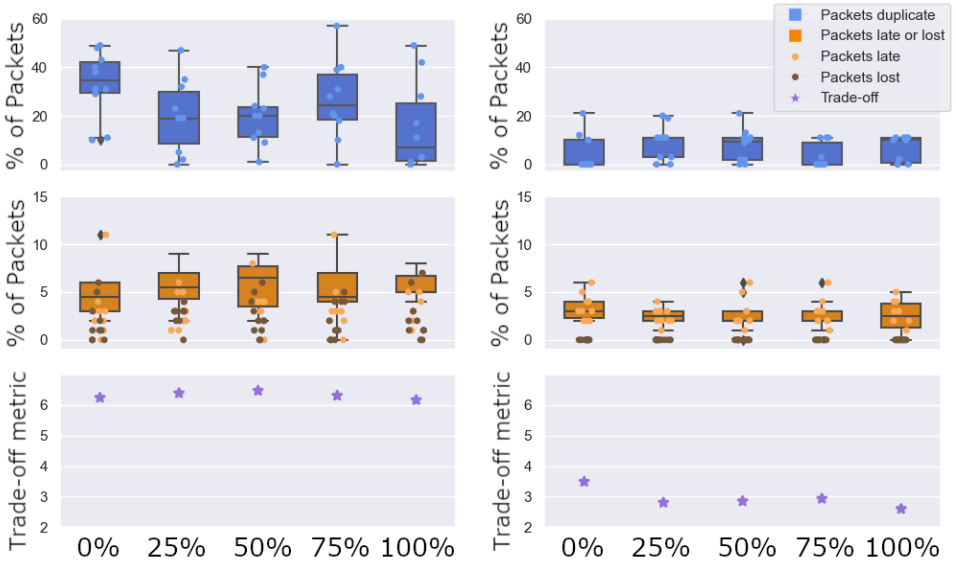
(a) Duplicate threshold (b) Loss-late threshold

Figure 5.17: Impact of the thresholds in the regular scenario



(a) Duplicate threshold (b) Loss-late threshold

Figure 5.18: Impact of the thresholds in the impaired scenario



(a) Impaired scenario (b) Regular scenario

Figure 5.19: Impact of the latency weight in the scenarios

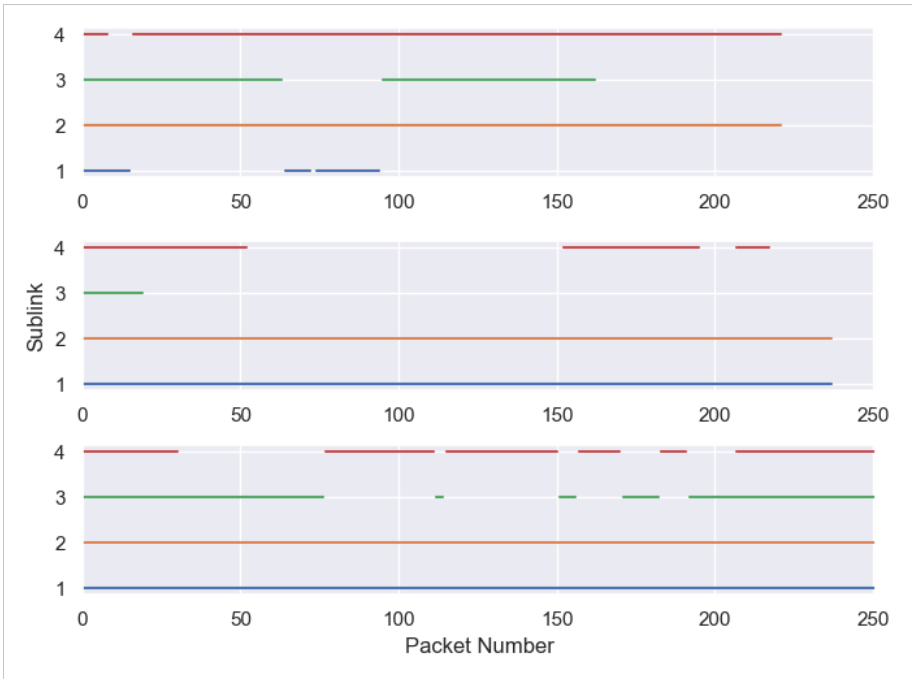


Figure 5.20: Example of sublinks used in the impaired scenario

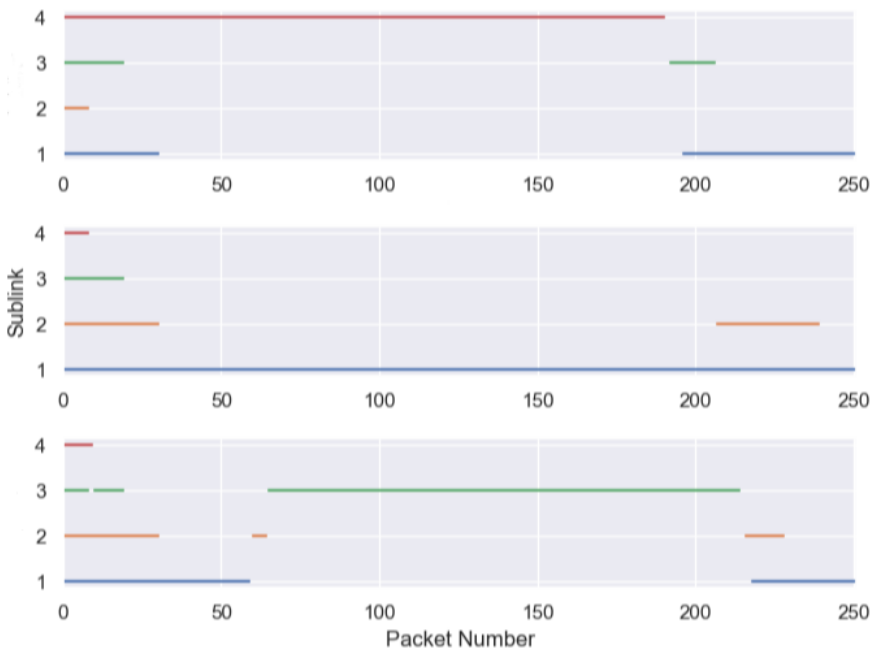
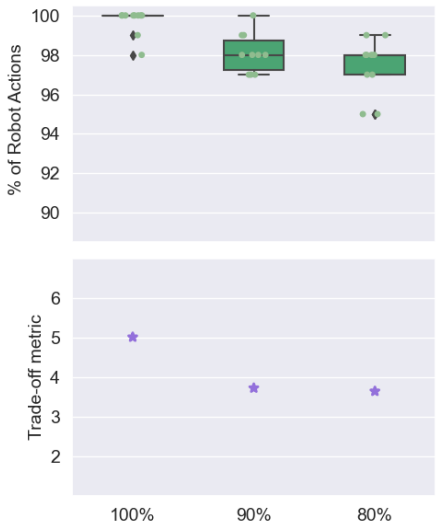


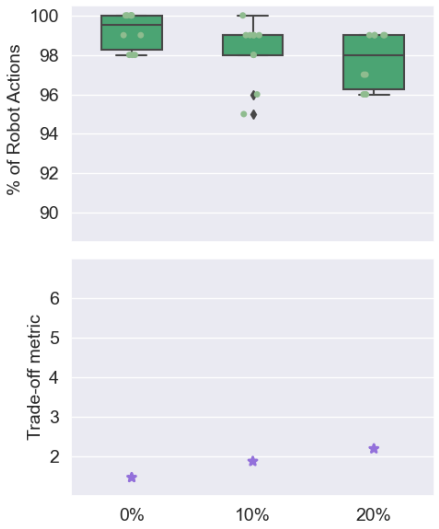
Figure 5.21: Example of sublinks used in the regular scenario

5.3.3.3 Application Layer Measurements

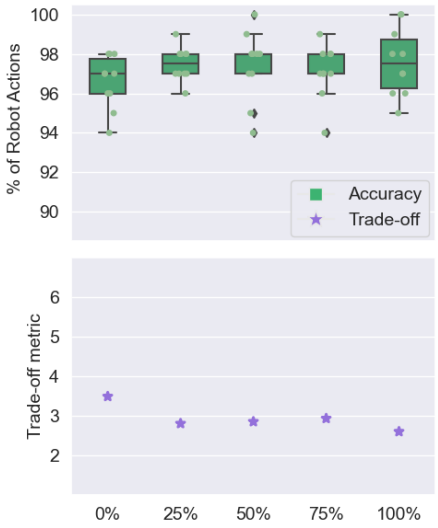
Figures 5.22 and 5.23 show the accuracy and the trade-off metric under the different configurations in the regular and impaired scenarios. In the regular scenario, we note that the most accurate case is the one of the duplicate threshold 100%, which receives 100% of the actions accurately in most of the iterations. However, the trade-off metric shows that it is the least efficient. The best balance between accuracy and efficiency was achieved with a loss-late threshold of 0%, with more than 99% of mean accuracy and a low trade-off metric. In the impaired scenario, the duplicate threshold of 100% was once again the most accurate. Nevertheless, the configuration with a duplicate threshold of 90% was the best in terms of efficiency while maintaining close to 99% of mean accuracy.



(a) Duplicate threshold

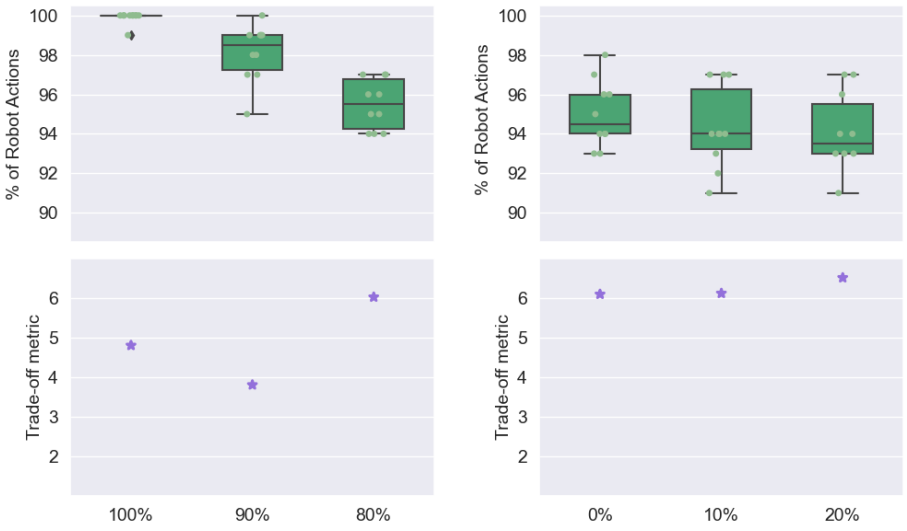


(b) Loss-late threshold

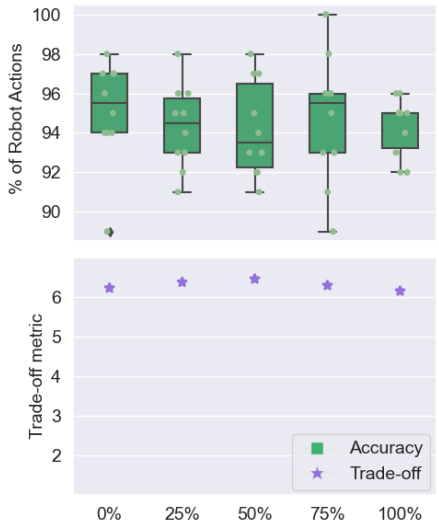


(c) Latency weight

Figure 5.22: Accuracy and trade-off metric in the regular scenario



(a) Duplicate threshold (b) Loss-late threshold



(c) Latency weight

Figure 5.23: Accuracy and trade-off metric in the impaired scenario

Chapter 6

Conclusion and Future Work

CONTENTS

Summary: This chapter provides an overview of the main conclusions drawn from this thesis, as well as potential areas for future research. Additionally, it includes the publications and projects related to this thesis.

6.1 Conclusion

In this thesis, we have presented the Multi-connection Tactile Internet Protocol. MTIP is a transport protocol for the remote control of Tactile Internet applications with stringent requirements such as industrial control and the remote control of drones and cars. The development of MTIP was driven by the need for a protocol that can effectively handle the unique challenges of these types of application, including low latency, high reliability, and exploiting current advances, such as the ability to operate over multiple networks. MTIP's ability to enhance communication is achieved through the use of multiple paths, namely sublinks. MTIP uses context-aware information to make decisions about the best sublinks for which to send redundant data. Specifically, it uses application preferences and measurements on the sublinks to send duplicate copies of the application commands simultaneously through multiple sublinks.

MTIP utilizes timestamps and sequence numbers to manage this operation, allowing it to identify both duplicate and undesirable late packets. In order to ensure the correctness and performance of the MTIP data exchange algorithms, several models were developed for verification purposes. To this end, two powerful modeling tools, SPIN and UPPAAL, were utilized. The first tool, SPIN, was used to perform a correctness analysis of the data exchange algorithms. Through this analysis, it was demonstrated that MTIP is capable of effectively managing duplicate packets, late packets, out-of-order packets, and packet losses. In addition to the one performed with the SPIN tool, a complementary performance analysis was conducted using the UPPAAL tool and Statistical Model Checking. The goal of this analysis was to evaluate the impact of using more sublinks on the percentage of correctly received packets, as well as the percentage of duplicate packets. In general, the use of the tools SPIN and UPPAAL provided valuable information on the performance and correctness of MTIP data exchange algorithms. As a result, we were able to proceed with the implementation and evaluation of MTIP.

We implemented a first version of the protocol using C/C++ and the Qt library in the Linux OS. We tested the implementation in a simulated environment to assess its expected behavior and analyse the impact of the different configurations of the protocol in the selection of sublinks and the percentage of packet losses. Following the simulated environment testing, we conducted an evaluation on a real 5G testbed using an industrial robot. Both evaluations provided valuable insights into the performance and effectiveness of MTIP. In particular, we observed that more restrictive thresholds tend to result in more duplicate packets but fewer losses, making this configuration more suitable for worse network environments. Conversely, less restrictive thresholds tend to reduce duplicates at the cost of some losses, a scenario that may be recommendable if any of the underlying sublinks is mostly stable and reliable through the connection.

In conclusion, the development of the Multi-connection Tactile Internet Protocol presents a significant contribution to the field of Tactile Internet applications. The protocol's utilization of multiple paths provides a more robust and reliable communication method, while the use of context-awareness enables the effective use of MTIP in a wide range of Tactile Internet applications. Overall, the development and evaluation of MTIP have demonstrated its potential for Tactile Internet applications, thereby laying a foundation for future advancements in this field.

6.2 Future Work

Regarding the future work, there are several open research topics that require further investigation.

Implementation and evaluation of MTIP in mobile devices At present, the evaluation of MTIP has been limited to the Linux OS, although it has also been successfully ported to Mac OS. An area of interest for further research would be exploring the adaptation of MTIP to mobile phones. Given that mobile phones are typically equipped with multiple network interfaces such as WiFi and cellular connections, this renders a promising area for the protocol. Following the research conducted on MPTCP [192, 193], there is potential to adapt MTIP for use on mobile phones and evaluate its performance in this context.

Evaluation of MTIP with Non-Terrestrial Networks In recent years, there has been a growing interest in Non-Terrestrial Networks (NTN) in the context of cellular networks [194, 195]. With the proliferation of new technologies and the increasing demand for connectivity and data transfer, NTN have become critical components of our global communication infrastructure. It is worth noting that the 3GPP has included these systems in the 5G and the new 6G architecture [196, 197]. Moreover, novel phones are going to support this capability as well [198, 199]. Multi-connectivity is an enabler for such communications, and we believe that MTIP could be used in this scenario. Particularly, in hazardous environments with bad access and connectivity and using low orbit satellites, which have the potential to maintain low latency levels. Therefore, exploring the application of MTIP in these contexts presents a promising area for future research.

Further evaluation and enhancements of MTIP Future work in the scope of automated verification will address performance aspects in different wireless scenarios, following the run-time verification approaches described in [200] and [201]. For this purpose, we will work with actual traces of the MTIP protocol generated in our experimental 5G network presented

in [202]. This approach will allow us to gain a deeper understanding of MTIP's performance in different wireless scenarios and will pave the way for further advancements in the field of Tactile Internet applications

Moreover, in our evaluation of MTIP's implementation, we assess the impact of the three main parameters of the algorithm to select the sublinks. However, MTIP implementation is tuned with additional parameters that we have yet to explore the impact, such as the alpha factor to consider previous measurements or the amount of packets used to measure the reliability of a sublink. Additionally, we need to assess the impact of using CHAR packets and including additional network measurements in MTIP algorithms. To optimize the performance with these additional parameters and measurements, it may be necessary to investigate the application of machine learning or artificial intelligence solutions that can learn from the network.

6.3 Publications and Projects

The following subsections present the publications and activities related to this thesis.

6.3.1 Journals and International Conferences

- **J1**: D. Rico and P. Merino, "A Survey of End-to-End Solutions for Reliable Low-Latency Communications in 5G Networks" in IEEE Access, vol. 8, pp. 192808-192834, 2020, doi: 10.1109/ACCESS.2020.3032726
- **J2**: D. Rico, K.-J. Grinemmo, A. Brunstrom and P. Merino, "Performance Analysis of The Multi-connection Tactile Internet Protocol over 5G" in Journal of Network and Systems Management, doi: 10.1007/s10922-023-09737-0
- **C1**: D. Rico, M.M. Gallardo, and P. Merino. "Modeling and verification of the Multi-connection Tactile Internet Protocol" in Proceedings of the 17th ACM Symposium on QoS and Security for Wireless and Mobile Networks (Q2SWinet '21). Association for Computing Machinery, New York, USA, 2021, 105–114, doi: 10.1145/3479242.3487328
- **C2**: A. Díaz Zayas, D. Rico, B. García, and P. Merino. "A Coordination Framework for Experimentation in 5G Testbeds: URLLC as Use Case" in Proceedings of the 17th ACM International Symposium on Mobility Management and Wireless Access (MobiWac '19). Association for Computing Machinery, New York, NY, USA, 71–79, doi: 10.1145/3345770.3356742
- **C3**: D. Rico, K.-J. Grinemmo, A. Brunstrom and P. Merino, "Implementation and evaluation of the Multi-connection Tactile Internet

Protocol and API" in 2022 IEEE/IFIP Network Operations and Management Symposium (NOMS '22), Budapest, Hungary, 2022, pp. 1-6, doi: 10.1109/NOMS54207.2022.9789842

6.3.2 Publications Under Review

- **P1**: D. Rico, M.M. Gallardo, and P. Merino. "Verification of a multi-connectivity protocol for Tactile Internet applications" (submitted, under review).

6.3.3 Other Dissemination Activities

- **D1 - Poster presentation**: A. Díaz Zayas, D. Rico, B. García, and P. Merino. "A Coordination Framework for Experimentation in 5G Testbeds: URLLC as Use Case", poster presentation in the 22nd ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM '19), Miami Beach, USA, Nov. 2019.

- **D2 - Oral presentation**: D. Rico, and P. Merino, "Ultra Reliable Low Latency Communications: A state of the art review", in Proceedings of the *XXVI Jornadas de Concurrencia y Sistemas Distribuidos* (JCSD '19) Zaragoza, Spain, Jun. 2019.

- **D3 - Oral presentation and publication**: A. Diaz-Zayas, D. Rico, B. García, and P. Merino, *"Marco de Trabajo para la Coordinacion de Testbeds 5G: URLLC como Caso de Uso"*, in Proceedings of the *XIV Jornadas de Ingeniería Telemática* (JITEL '19) Zaragoza, Spain, Oct. 2019. doi: 10.26754/uz.978-84-09-21112-8

- **D4 - Oral presentation**: D. Rico, A. Brunstrom, K-J. Grinemmo, and P. Merino, "Implementation and evaluation of the Multi-connection Tactile Internet Protocol and API", in Proceedings of the *XXVIII Jornadas de Concurrencia y Sistemas Distribuidos* (JCSD '22) Ávila, Spain, Jun. 2022.

6.3.4 Related Projects and Funding

- **P1**: **FPU**, Ministry of Science, Innovation and Universities of Spain under grant agreement FPU17/04292.

- **P2**: **5GENESIS**, European Union's Horizon 2020 research and innovation programme under grant agreement No 815178.

- **P3**: **EuWireless**, European Union's Horizon 2020 research and innovation programme under grant agreement No 777517.

- **P4**: **RFOG**, Ministry of Science, Innovation and Universities of Spain grant agreement RTI2018-099777-B-I00.
- **P5**: **EVOLVED5G**, European Union's Horizon 2020 research and innovation grant agreement No. 101016608.
- **P6**: **5G+TACTILE 1**, Ministry of Economic Affairs and Digital Transformation of Spain, UNICO I+D, under gran agreement TSI-063000-2021-11.
- **P7**: **6G-SANDBOX**, European Union's Horizon Smart Networks and Services Joint Undertaking grant agreement No. 101096328.

6.3.5 Research internship

This thesis involved a six-month internship in 2021 at Karlstad University, Sweden, which was made possible by the financial support of the Ministry of Education of Spain via grants EST19/00930 and EST21/00446.

Bibliography

[1] O. Holland, E. Steinbach, R. V. Prasad, Q. Liu, Z. Dawy, A. Aijaz, N. Pappas, K. Chandra, V. S. Rao, S. Oteafy, M. Eid, M. Luden, A. Bhardwaj, X. Liu, J. Sachs, and J. Araújo, "The IEEE 1918.1 "Tactile Internet" Standards Working Group and its Standards," *Proceedings of the IEEE*, vol. 107, no. 2, pp. 256–279, 2019.

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

[3] M. Simsek, A. Aijaz, M. Dohler, J. Sachs, and G. Fettweis, "5g-enabled tactile internet," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 460–473, 2016.

[4] Ericsson, "5g systems: Enabling the transformation of industry and society," White Paper UEN 284 23-3251 rev B, January 2017, https://www.ericsson.com/en/reports-and-papers/white-papers/5g-systems--enabling-the-transformation-of-industry-and-society.

[5] Cisco, "White paper: Time-sensitive networking: A technical introduction," Cisco Public, Tech. Rep. C11-738950-00, 2017. [Online]. Available: https://www.cisco.com/c/dam/en/us/solutions/collateral/industry-solutions/white-paper-c11-738950.pdf

[6] C. Cruces, R. Torrego, A. Arriola, and I. Val, "Deterministic hybrid architecture with time sensitive network and wireless capabilities," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, Sep. 2018, pp. 1119–1122.

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

[8] Z. MacHardy, A. Khan, K. Obana, and S. Iwashina, "V2X Access Technologies: Regulation, Research, and Remaining Challenges," *IEEE Communications Surveys Tutorials*, vol. 20, no. 3, pp. 1858–1877, thirdquarter 2018.

[9] A. Fotouhi, H. Qiang, M. Ding, M. Hassan, L. G. Giordano, A. Garcia-Rodriguez, and J. Yuan, "Survey on uav cellular communications: Practical aspects, standardization advancements, regulation, and security challenges," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2019.

[10] H. Chen, R. Abbas, P. Cheng, M. Shirvanimoghaddam, W. Hardjawana, W. Bao, Y. Li, and B. Vucetic, "Ultra-reliable low latency cellular networks: Use cases, challenges and approaches," *IEEE Communications Magazine*, vol. PP, 09 2017.

[11] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Service requirements for cyber-physical control applications in vertical domains; Stage 1 (Release 18)," 3rd Generation Partnership Project, Technical Specification (TS) 22.104, 03 2021, version 18.0.0.

[12] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Service requirements for the 5G system; Stage 1 (Release 18)," 3rd Generation Partnership Project, Tech. Rep. 22.261, 09 2020, version 18.0.0.

[13] K. S. Kim, D. K. Kim, C. Chae, S. Choi, Y. Ko, J. Kim, Y. Lim, M. Yang, S. Kim, B. Lim, K. Lee, and K. L. Ryu, "Ultrareliable and low-latency communication techniques for tactile internet services," *Proceedings of the IEEE*, vol. 107, no. 2, pp. 376–393, 2019.

[14] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Release 15 Description; Summary of Rel-15 Work Items (Release 15)," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 21.915, 09 2019, version 15.0.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3389

[15] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; System architecture for the 5G System (5GS); Stage 2 (Release 17)," 3rd Generation Partnership Project, Tech. Rep. 23.501, 12 2022, version 17.7.0.

[16] B. Blanco, J. O. Fajardo, I. Giannoulakis, E. Kafetzakis, S. Peng, J. Pérez-Romero, I. Trajkovska, P. S. Khodashenas, L. Goratti, M. Paolino, E. Sfakianakis, F. Liberal, and G. Xilouris, "Technology pillars in the architecture of future 5g mobile networks: Nfv, mec and sdn," *Computer Standards & Interfaces*, vol. 54, pp. 216–228, 2017, sI: Standardization SDN&NFV. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0920548916302446

[17] I. F. Akyildiz, S. Nie, S.-C. Lin, and M. Chandrasekaran, "5g roadmap: 10 key enabling technologies," *Computer Networks*, vol. 106, pp. 17–48, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128616301918

[18] Networld2020, "The strategic research and innovation agenda: Smart networks in the context of ngi," European Technology Platform, Tech. Rep., September 2020. [Online]. Available: https://www.networld2020.eu/3487-2/

[19] G. Papastergiou, G. Fairhurst, D. Ros, A. Brunstrom, K.-J. Grinnemo, P. Hurtig, N. Khademi, M. Tüxen, M. Welzl, D. Damjanovic, and S. Mangiante, "De-ossifying the internet transport layer: A survey and future perspectives," *IEEE Communications Surveys Tutorials*, vol. 19, no. 1, pp. 619–639, 2017.

[20] M. Scharf and A. Ford, "Multipath tcp (mptcp) application interface considerations," Internet Requests for Comments, RFC Editor, RFC 6897, March 2013.

[21] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar *et al.*, "The quic transport protocol: Design and internet-scale deployment," in *Proceedings of the conference of the ACM special interest group on data communication*, 2017, pp. 183–196.

[22] Y. Cao, Q. Liu, G. Luo, Y. Yi, and M. Huang, "Pr-mptcp+: Context-aware qoe-oriented multipath tcp partial reliability extension for real-time multimedia applications," in *2016 Visual Communications and Image Processing (VCIP)*, 2016, pp. 1–4.

[23] M. Li, A. Lukyanenko, and Y. Cui, "Network coding based multipath tcp," in *2012 Proceedings IEEE INFOCOM Workshops*, March 2012, pp. 25–30.

[24] Q. De Coninck and O. Bonaventure, "Multipath quic: Design and evaluation," in *Proceedings of the 13th international conference on emerging networking experiments and technologies*, 2017, pp. 160–166.

[25] R. Stewart, "Stream control transmission protocol," Internet Requests for Comments, RFC Editor, RFC 4960, September 2007. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4960.txt

[26] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "Rtp: A transport protocol for real-time applications," Internet Requests for Comments, RFC Editor, STD 64, July 2003. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3550.txt

[27] V. Singh, T. Karkkainen, J. Ott, S. Ahsan, and L. Eggert, "Multipath rtp (mprtp)," Working Draft, IETF Secretariat, Internet-Draft draft-singh-avtcore-mprtp, January 2017. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-avtcore-mprtp

[28] P. M. Vespa, C. Miller, X. Hu, V. Nenov, F. Buxey, and N. A. Martin, "Intensive care unit robotic telepresence facilitates rapid physician response to unstable patients and decreased cost in neurointensive care," *Surgical Neurology*, vol. 67, no. 4, pp. 331–337, 2007.

[29] S. Dadhich, U. Bodin, and U. Andersson, "Key challenges in automation of earth-moving machines," *Automation in Construction*, vol. 68, pp. 212–222, 2016.

[30] S. Dodeller, "Transport layer protocols for telehaptics update messages," in *Proc. 22nd Biennial Symposium on Communications, June 2004*, 2004.

[31] L. Ping, L. Wenjuan, and S. Zengqi, "Transport layer protocol reconfiguration for network-based robot control system," in *Proceedings. 2005 IEEE Networking, Sensing and Control, 2005.*, 2005, pp. 1049–1053.

[32] R. Wirz, M. Ferre, R. Marín, J. Barrio, J. M. Claver, and J. Ortego, "Efficient transport protocol for networked haptics applications," in *Haptics: Perception, Devices and Scenarios*, M. Ferre, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 3–12.

[33] G. Kokkonis, K. Psannis, M. Roumeliotis, and S. Kontogiannis, "A survey of transport protocols for haptic applications," 10 2012, pp. 192–197.

[34] M. Al Jaafreh, M. Alowaidi, H. Al Osman, and A. El Saddik, "Multimodal systems, experiences, and communications: A review toward the tactile internet vision," in *Recent Trends in Computer Applications*, J. M. Alja'am, A. El Saddik, and A. H. Sadka, Eds. Cham: Springer International Publishing, 2018, pp. 191–220.

[35] D. Rico and P. Merino, "A survey of end-to-end solutions for reliable low-latency communications in 5g networks," *IEEE Access*, vol. 8, pp. 192 808–192 834, 2020.

[36] N. Khademi, D. Ros, M. Welzl, Z. Bozakov, A. Brunstrom, G. Fairhurst, K. Grinnemo, D. Hayes, P. Hurtig, T. Jones, S. Mangiante, M. Tuxen, and F. Weinrank, "Neat: A platform- and protocol-independent internet transport api," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 46–54, 2017.

[37] T. Pauly et al., "An architecture for transport services," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-taps-arch-17, March 2023. [Online]. Available: https://www.ietf.org/archive/id/draft-ietf-taps-arch-17.txt

[38] Mininet Project, "Mininet," 2021. [Online]. Available: http://mininet.org/

[39] The Linux Fundation, "Netem," 2021. [Online]. Available: https://wiki.linuxfoundation.org/networking/netem

[40] M. R. Group, "Morse lab," http://www.morse.uma.es, 2021, online; accessed 28 May 2021.

[41] M. S. Elbamby, C. Perfecto, M. Bennis, and K. Doppler, "Toward low-latency and ultra-reliable virtual reality," *IEEE Network*, vol. 32, no. 2, pp. 78–84, March 2018.

[42] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5g: Ran, core network and caching solutions," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2018.

[43] J. Qadir, A. Ali, K. A. Yau, A. Sathiaseelan, and J. Crowcroft, "Exploiting the power of multiplicity: A holistic survey of network-layer multipath," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2176–2213, Fourthquarter 2015.

[44] K. Poularakis, G. Iosifidis, V. Sourlas, and L. Tassiulas, "Exploiting caching and multicast for 5g wireless networks," *IEEE Transactions on Wireless Communications*, vol. 15, no. 4, pp. 2995–3007, April 2016.

[45] G. Araniti, M. Condoluci, P. Scopelliti, A. Molinaro, and A. Iera, "Multicasting over emerging 5g networks: Challenges and perspectives," *IEEE Network*, vol. Vol. 31, pp. pp 80–89, 03 2017.

[46] O. Awobuluyi, J. Nightingale, Q. Wang, and J. M. Alcaraz-Calero, "Video quality in 5g networks: Context-aware qoe management in the sdn control plane," in *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, Oct 2015, pp. 1657–1662.

[47] D. Kutscher, S. Eum, K. Pentikousis, I. Psaras, D. Corujo, D. Saucez, T. Schmidt, and M. Waehlisch, "Information-centric networking (icn) research challenges," *Internet Research Task Force (IRTF)*, pp. 1–38, 2016.

[48] A. Al-Dulaimi, X. Wang, and C. I, *Network Softwarization View of 5G Networks.* IEEE, 2018, pp. 499–518. [Online]. Available: https://ieeexplore.ieee.org/document/8496391

[49] T. Jones, G. Fairhurst, and C. Perkins, "Raising the datagram api to support transport protocol evolution," in *2017 IFIP Networking Conference (IFIP Networking) and Workshops*, June 2017, pp. 1–6.

[50] B. Trammell, C. Perkins, and M. Kühlewind, "Post sockets: Towards an evolvable network transport interface," in *2017 IFIP Networking Conference (IFIP Networking) and Workshops*, June 2017, pp. 1–6.

[51] E. Atxutegi, "Moving toward the intra-protocol de-ossification of tcp in mobile networks: Start-up and mobility," Ph.D. dissertation, Jan 2018.

[52] B. Bangerter, S. Talwar, R. Arefi, and K. Stewart, "Networks and devices for the 5g era," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 90–96, February 2014.

[53] "Path Aware Networking RG (panrg) ," IETF https://datatracker.ietf.org/rg/panrg/about/, accessed: 2020-01-23.

[54] "Cisco intent-based networking (ibn)," https://www.cisco.com/c/en/us/solutions/intent-based-networking.html?dtid=osscdc000283, accessed: 2020-07-31.

[55] H. Koumaras, D. Tsolkas, G. Gardikis, P. M. Gomez, V. Frascolla, D. Triantafyllopoulou, M. Emmelmann, V. Koumaras, M. L. G. Osma, D. Munaretto, E. Atxutegi, J. S. de Puga, O. Alay, A. Brunstrom, and A. M. C. Bosneag, "5GENESIS: The Genesis of a flexible 5G Facility," in *23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD).* IEEE, November 2018.

[56] 5GENESIS. (2018) 5GENESIS Objectives. https://5genesis.eu/objectives/.

[57] A. Petlund, "Improving latency for interactive, thin-stream applications over reliable transport," *SIGMultimedia Rec.*, vol. 2, no. 1, pp. 17–18, Mar. 2010.

[58] Z. Yin, H. Alnuweiri, A. L. N. Reddy, H. Celebi, and K. Qaraqe, "Improving the performance of delay based protocol in delivering real time media via early retransmission," in *2011 18th International Conference on Telecommunications*, May 2011, pp. 511–516.

[59] J. W. Park, R. P. Karrer, and J. Kim, "Tcp-rome: A transport-layer parallel streaming protocol for real-time online multimedia environments," *Journal of Communications and Networks*, vol. 13, no. 3, pp. 277–285, June 2011.

[60] M. Massaro, C. E. Palazzi, and A. Bujari, "Exploiting tcp vegas' algorithm to improve real-time multimedia applications," in *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, Jan 2015, pp. 316–321.

[61] M. Polese, R. Jana, and M. Zorzi, "Tcp in 5g mmwave networks: Link level retransmissions and mp-tcp," in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, May 2017, pp. 343–348.

[62] I. Petrov and T. Janevski, "Advanced 5g-tcp: Transport protocol for 5g mobile networks," in *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2017, pp. 103–107.

[63] Google LLC, "Tcp bbr congestion control comes to gcp - your internet just got faster," Google Cloud Platform Blog, July 2017, https://cloudplatform.googleblog.com/2017/07/TCP-BBR-congestion-control-comes-to-GCP-your-Internet-just-got-faster.html.

[64] B. Gambhava and C. Bhensdadia, "Discrete tcp: Differentiating slow start and congestion avoidance," *International Journal of Intelligent Engineering and Systems*, vol. 11, no. 5, pp. 206–214, 2018.

[65] X. Zhu, R. Zheng, D. Yang, H. Liu, and J. Hou, "Radio-aware tcp optimization in mobile network," *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–5, 2017.

[66] J. Luo, J. Jin, and F. Shan, "Standardization of low-latency tcp with explicit congestion notification: A survey," *IEEE Internet Computing*, vol. 21, no. 1, pp. 48–55, Jan 2017.

[67] W. Zhou, Q. Li, M. Caesar, and P. B. Godfrey, "Asap: A low-latency transport layer," in *Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '11. New York, NY, USA: ACM, 2011, pp. 20:1–20:12.

[68] C. Park and H. Kim, "Short-term reliable protocol for low latency video transmission," in *2014 International Conference on Computational Science and Computational Intelligence*, vol. 2, March 2014, pp. 311–312.

[69] H. Cheng and K. C. Lin, "Source selection and content dissemination for preference-aware traffic offloading," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 11, pp. 3160–3174, Nov 2015.

[70] C. Park and H. Kim, "Low latency video transmission device," in *Information Science and Applications*. Springer, 2015, pp. 217–222.

[71] I. Johansson and Z. Sarker, "Self-clocked rate adaptation for multimedia," Internet Requests for Comments, RFC Editor, RFC 8298, December 2017.

[72] R. Mittal, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, D. Zats *et al.*, "Timely: Rtt-based congestion control for the datacenter," in *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4. ACM, 2015, pp. 537–550.

[73] B. Briscoe, K. Schepper, M. Bagnulo, and G. White, "Low latency, low loss, scalable throughput (l4s) internet service: Architecture," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-tsvwg-l4s-arch-06, March 2020, https://tools.ietf.org/html/draft-ietf-tsvwg-l4s-arch-06.

[74] M. Menth, A. Stockmayer, and M. Schmidt, "Lisp hybrid access," Working Draft, IETF Secretariat, Internet-Draft draft-menth-lisp-ha-00, July 2015, https://www.ietf.org/archive/id/draft-menth-lisp-ha-00.txt.

[75] K.-K. Yap, T.-Y. Huang, Y. Yiakoumis, S. Chinchali, N. McKeown, and S. Katti, "Scheduling packets over multiple interfaces while respecting user preferences," in *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '13. New York, NY, USA: ACM, 2013, pp. 109–120.

[76] S. Singh, S. Yeh, N. Himayat, and S. Talwar, "Optimal traffic aggregation in multi-rat heterogeneous wireless networks," in *2016 IEEE International Conference on Communications Workshops (ICC)*, May 2016, pp. 626–631.

[77] I. Gonzalez-Muriel, Á. Martín-Heredia, and P. Merino-Gómez, "Testbed to experiment with lte wifi aggregation," in *2019 European Conference on Networks and Communications (EuCNC)*, 2019, pp. 506–511.

[78] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "Tcp extensions for multipath operation with multiple addresses," Internet Requests for Comments, RFC Editor, RFC 8684, March 2020, http://www.rfc-editor.org/rfc/rfc8684.txt.

[79] Y. Cui, L. Wang, X. Wang, H. Wang, and Y. Wang, "Fmtcp: A fountain code-based multipath transmission control protocol," *IEEE/ACM Trans. Netw.*, vol. 23, no. 2, pp. 465–478, Apr. 2015.

[80] P. Hurtig, K. Grinnemo, A. Brunstrom, S. Ferlin, ´O. Alay, and N. Kuhn, "Low-latency scheduling in mptcp," *IEEE/ACM Transactions on Networking*, vol. 27, no. 1, pp. 302–315, Feb 2019.

[81] C. Diop, G. Dugue, C. Chassot, and E. Exposito, "Qos-oriented mptcp extensions for multimedia multi-homed systems," in *2012 26th International Conference on Advanced Information Networking and Applications Workshops*, March 2012, pp. 1119–1124.

[82] J. Wu, C. Yuen, B. Cheng, M. Wang, and J. Chen, "Streaming high-quality mobile video with multipath tcp in heterogeneous wireless networks," *IEEE Transactions on Mobile Computing*, vol. 15, no. 9, pp. 2345–2361, Sept 2016.

[83] Y. Cao, Q. Liu, G. Luo, Y. Yi, and M. Huang, "Pr-mptcp+: Context-aware qoe-oriented multipath tcp partial reliability extension for real-time multimedia applications," in *2016 Visual Communications and Image Processing (VCIP)*, Nov 2016, pp. 1–4.

[84] A. Nikravesh, Y. Guo, F. Qian, Z. M. Mao, and S. Sen, "An in-depth understanding of multipath tcp on mobile devices: Measurement and system design," in *Proceedings of the 22Nd Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '16. New York, NY, USA: ACM, 2016, pp. 189–201.

[85] C. Lee, S. Song, H. Cho, G. Lim, and J. Chung, "Optimal multipath tcp offloading over 5g nr and lte networks," *IEEE Wireless Communications Letters*, vol. 8, no. 1, pp. 293–296, Feb 2019.

[86] A. Singh and A. L. N. Reddy, "Multi path pert," in *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*, July 2013, pp. 1–9.

[87] Q. D. Coninck, "First experiments with multipath quic," IETF 99, July 2017, https://datatracker.ietf.org/meeting/99/materials/slides-99-quic-sessb-first-experiments-with-multipath-quic/.

[88] V. Singh, T. Karkkainen, J. Ott, S. Ahsan, and L. Eggert, "Multipath rtp (mprtp)," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-avtcore-mprtp-03, July 2016, https://tools.ietf.org/html/draft-ietf-avtcore-mprtp-03.

[89] O. C. Kwon, "Multipath transport protocols for video streaming over heterogeneous wireless networks," Ph.D. dissertation, December 2014.

[90] O. C. Kwon, Y. Go, Y. Park, and H. Song, "Mpmtp: Multipath multimedia transport protocol using systematic raptor codes over wireless networks," *IEEE Transactions on Mobile Computing*, vol. 14, no. 9, pp. 1903–1916, Sept 2015.

[91] Y. Hwang, B. O. Obele, and H. Lim, "Multipath transport protocol for heterogeneous multi-homing networks," in *Proceedings of the ACM CoNEXT Student Workshop*, ser. CoNEXT '10 Student Workshop. New York, NY, USA: ACM, 2010, pp. 5:1–5:2.

[92] W. Lei, S. Liu, and W. Zhang, "Multipath message transport protocol based on application-level relay (mpmtp-ar)," Working Draft, IETF Secretariat, Internet-Draft draft-leiwm-tsvwg-mpmtp-ar-09, February 2018, https://tools.ietf.org/html/draft-leiwm-tsvwg-mpmtp-ar-09.

[93] K. Kashwan and S. Karthik, "The modified mobile concurrent multipath transfer for joint resource management," *Procedia Engineering*, vol. 30, no. Supplement C, pp. 963 – 969, 2012, international Conference on Communication Technology and System Design 2011.

[94] L. P. Verma and M. Kumar, "An adaptive data chunk scheduling for concurrent multipath transfer," *Computer Standards & Interfaces*, vol. 52, pp. 97 – 104, 2017.

[95] T. Zhu, D. Feng, F. Wang, Y. Hua, Q. Shi, Y. Xie, and Y. Wan, "A congestion-aware and robust multicast protocol in sdn-based data center networks," *Journal of Network and Computer Applications*, vol. 95, pp. 105 – 117, 2017.

[96] E. Tsimbalo, A. Tassi, and R. J. Piechocki, "Reliability of multicast under random linear network coding," *IEEE Transactions on Communications*, vol. 66, no. 6, pp. 2547–2559, June 2018.

[97] X. Xiong and T. Chen, "Mtm: A reliable multiple trees multicast for data center network," in *2017 International Conference on Networking, Architecture, and Storage (NAS)*, Aug 2017, pp. 1–7.

[98] K. Chi, L. Huang, Y. Li, Y. Zhu, X. Tian, and M. Xia, "Efficient and reliable multicast using device-to-device communication and network coding for a 5g network," *IEEE Network*, vol. 31, no. 4, pp. 78–84, July 2017.

[99] S. Roger, D. Martín-Sacristán, D. Garcia-Roger, J. F. Monserrat, P. Spapis, A. Kousaridas, S. Ayaz, and A. Kaloxylos, "Low-latency layer-2-based multicast scheme for localized v2x communications," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–14, 2018.

[100] X. Zhang, M. Yang, L. Wang, and M. Sun, "An openflow-enabled elastic loss recovery solution for reliable multicast," *IEEE Systems Journal*, vol. 12, no. 2, pp. 1945–1956, June 2018.

[101] K. Mahajan, D. Sharma, and V. Mann, "Athena: Reliable multicast for group communication in sdn-based data centers," in *2017 9th International Conference on Communication Systems and Networks (COMSNETS)*, Jan 2017, pp. 174–181.

[102] C. A. García-Pérez and P. Merino, "Enabling low latency services on lte networks," in *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, Sept 2016, pp. 248–255.

[103] C. A. García-Pérez and P. Merino, "Experimental evaluation of fog computing techniques to reduce latency in LTE networks," *Trans. Emerging Telecommunications Technologies*, vol. 29, no. 4, 2018.

[104] K. Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Mobile edge computing and networking for green and low-latency internet of things," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 39–45, May 2018.

[105] M. Jalil Piran, S. M. R. Islam, and D. Y. Suh, "Cash: Content- and network-context-aware streaming over 5g hetnets," *IEEE Access*, vol. 6, pp. 46 167–46 178, 2018.

[106] S.-Q. Lee and J.-u. Kim, "Local breakout of mobile access network traffic by mobile edge computing," in *2016 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2016, pp. 741–743.

[107] G. Cattaneo, F. Giust, C. Meani, D. Munaretto, and P. Paglierani, "Deploying cpu-intensive applications on mec in nfv systems: The immersive video use case," *Computers*, vol. 7, no. 4, p. 55, 2018.

[108] A. Huang, N. Nikaein, T. Stenbock, A. Ksentini, and C. Bonnet, "Low latency mec framework for sdn-based lte/lte-a networks," in *2017 IEEE International Conference on Communications (ICC)*, May 2017, pp. 1–6.

[109] J. Heinonen, P. Korja, T. Partti, H. Flinck, and P. Pöyhönen, "Mobility management enhancements for 5g low latency services," in *2016 IEEE International Conference on Communications Workshops (ICC)*, May 2016, pp. 68–73.

[110] E. Schiller, N. Nikaein, E. Kalogeiton, M. Gasparyan, and T. Braun, "CDS-MEC: NFV/SDN-based application management for MEC in 5G

systems," *Computer Networks, 15 February 2018, ISSN 1389-1286*, 02 2018, http://www.eurecom.fr/publication/5461.

[111] B. Yang, W. K. Chai, Z. Xu, K. V. Katsaros, and G. Pavlou, "Cost-efficient nfv-enabled mobile edge-cloud for low latency mobile applications," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 475–488, March 2018.

[112] R. Cziva and D. P. Pezaros, "On the latency benefits of edge nfv," in *2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, May 2017, pp. 105–106.

[113] S. Nunna, A. Kousaridas, M. Ibrahim, M. Dillinger, C. Thuemmler, H. Feussner, and A. Schneider, "Enabling real-time context-aware collaboration through 5g and mobile edge computing," in *2015 12th International Conference on Information Technology - New Generations*, April 2015, pp. 601–605.

[114] S. Dutta, T. Taleb, P. A. Frangoudis, and A. Ksentini, "On-the-fly qoe-aware transcoding in the mobile edge," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec 2016, pp. 1–6.

[115] T. Taleb, S. Dutta, A. Ksentini, M. Iqbal, and H. Flinck, "Mobile edge computing potential in making cities smarter," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 38–43, March 2017.

[116] E. Li, Z. Zhou, and X. Chen, "Edge intelligence: On-demand deep learning model co-inference with device-edge synergy," in *Proceedings of the 2018 Workshop on Mobile Edge Communications*, ser. MECOMM'18. New York, NY, USA: Association for Computing Machinery, 2018, p. 31–36. [Online]. Available: https://doi.org/10.1145/3229556.3229562

[117] M. Maier and A. Ebrahimzadeh, "Towards immersive tactile internet experiences: Low-latency fiwi enhanced mobile networks with edge intelligence [invited]," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 11, no. 4, pp. B10–B25, April 2019.

[118] J. Liu and Q. Zhang, "Offloading schemes in mobile edge computing for ultra-reliable low latency communications," *IEEE Access*, vol. 6, pp. 12 825–12 837, 2018.

[119] J. Pagé and J. Dricot, "Software-defined networking for low-latency 5g core network," in *2016 International Conference on Military Communications and Information Systems (ICMCIS)*, May 2016, pp. 1–7.

[120] J. Costa-Requena, A. Poutanen, S. Vural, G. Kamel, C. Clark, and S. K. Roy, "Sdn-based upf for mobile backhaul network slicing," in *2018 European Conference on Networks and Communications (EuCNC)*, June 2018, pp. 48–53.

[121] J. Wang and D. Li, "Adaptive computing optimization in software-defined network-based industrial internet of things with fog computing," *Sensors*, vol. 18, no. 8, p. 2509, Aug 2018.

[122] E. Lakiotakis, C. Liaskos, and X. Dimitropoulos, "Application-network collaboration using sdn for ultra-low delay teleorchestras," in *2017 IEEE Symposium on Computers and Communications (ISCC)*, July 2017, pp. 70–75.

[123] S. Garg, K. Kaur, S. H. Ahmed, A. Bradai, G. Kaddoum, and M. Atiquzzaman, "Mobqos: Mobility-aware and qos-driven sdn framework for autonomous vehicles," *IEEE Wireless Communications*, vol. 26, no. 4, pp. 12–20, August 2019.

[124] L. Han, Z. Li, W. Liu, K. Dai, and W. Qu, "Minimum control latency of sdn controller placement," in *2016 IEEE Trustcom/BigDataSE/ISPA*, Aug 2016, pp. 2175–2180.

[125] G. Wang, Y. Zhao, J. Huang, and Y. Wu, "An effective approach to controller placement in software defined wide area networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 344–355, March 2018.

[126] K.-K. Yap, T.-Y. Huang, M. Kobayashi, Y. Yiakoumis, N. McKeown, S. Katti, and G. Parulkar, "Making use of all the networks around us: A case study in android," in *Proceedings of the 2012 ACM SIGCOMM Workshop on Cellular Networks: Operations, Challenges, and Future Design*, ser. CellNet '12. New York, NY, USA: ACM, 2012, pp. 19–24.

[127] T. Hu, P. Yi, J. Zhang, and J. Lan, "Reliable and load balance-aware multi-controller deployment in sdn," *China Communications*, vol. 15, no. 11, pp. 184–198, 2018.

[128] M. T. Raza and S. Lu, "Enabling low latency and high reliability for ims-nfv," in *2017 13th International Conference on Network and Service Management (CNSM)*, Nov 2017, pp. 1–9.

[129] M. T. Raza, S. Lu, M. Gerla, and X. Li, "Refactoring network functions modules to reduce latencies and improve fault tolerance in nfv," *IEEE Journal on Selected Areas in Communications*, pp. 1–1, 2018.

[130] L. Qu, C. Assi, K. Shaban, and M. J. Khabbaz, "A reliability-aware network service chain provisioning with delay guarantees in nfv-enabled enterprise datacenter networks," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 554–568, Sept 2017.

[131] P. Mekikis, K. Ramantas, A. Antonopoulos, E. Kartsakli, L. Sanabria-Russo, J. Serra, D. Pubill, and C. Verikoukis, "Nfv-enabled experimental platform for 5g tactile internet support in industrial environments," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 1895–1903, 2020.

[132] W. Ding, H. Yu, and S. Luo, "Enhancing the reliability of services in nfv with the cost-efficient redundancy scheme," in *2017 IEEE International Conference on Communications (ICC)*, May 2017, pp. 1–6.

[133] M. Nascimento, T. Primini, E. Baum, P. Martucci, F. Cabelo, and L. Mariote, "Acceleration mechanism for high throughput and low latency in nfv environments," in *2017 IEEE 18th International Conference on High Performance Switching and Routing (HPSR)*, June 2017, pp. 1–6.

[134] D. Cho, J. Taheri, A. Y. Zomaya, and P. Bouvry, "Real-time virtual network function (vnf) migration toward low network latency in cloud environments," in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, June 2017, pp. 798–801.

[135] C. Sun, J. Bi, Z. Zheng, H. Yu, and H. Hu, "Nfp: Enabling network function parallelism in nfv," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 43–56. [Online]. Available: https://doi.org/10.1145/3098822.3098826

[136] J. Fan, Z. Ye, C. Guan, X. Gao, K. Ren, and C. Qiao, "Grep: Guaranteeing reliability with enhanced protection in nfv," in *Proceedings of the 2015 ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, ser. HotMiddlebox '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 13–18. [Online]. Available: https://doi.org/10.1145/2785989.2786000

[137] O. Bekkouche, M. Bagaa, and T. Taleb, "Toward a utm-based service orchestration for uavs in mec-nfv environment," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[138] P. Valsamas, P. Papadimitriou, I. Sakellariou, S. Petridou, L. Mamatas, S. Clayman, F. Tusa, and A. Galis, "Multi-pop network slice

deployment: A feasibility study," in *2019 IEEE 8th International Conference on Cloud Networking (CloudNet)*, 2019, pp. 1–6.

[139] Y. Yao, Q. Cao, J. Chase, P. Ruth, I. Baldin, Y. Xin, and A. Mandal, "Slice-based network transit service: Inter-domain l2 networking on exogeni," in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2017, pp. 736–741.

[140] C. Liang, F. R. Yu, and X. Zhang, "Information-centric network function virtualization over 5g mobile wireless networks," *IEEE network*, vol. 29, no. 3, pp. 68–74, 2015.

[141] G. Carofiglio, L. Mekinda, and L. Muscariello, "Lac: Introducing latency-aware caching in information-centric networks," in *2015 IEEE 40th Conference on Local Computer Networks (LCN)*, Oct 2015, pp. 422–425.

[142] G. Carofiglio and L. Mekinda and L. Muscariello, "Focal: Forwarding and caching with latency awareness in information-centric networking," in *2015 IEEE Globecom Workshops (GC Wkshps)*, Dec 2015, pp. 1–7.

[143] G. Carofiglio, L. Mekinda, and L. Muscariello, "Joint forwarding and caching with latency awareness in information-centric networking," *Computer Networks*, vol. 110, pp. 133 – 153, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128616303176

[144] Z. Zhang, C.-H. Lung, I. Lambadaris, and M. St-Hilaire, "When 5g meets icn: An icn-based caching approach for mobile video in 5g networks," *Computer Communications*, vol. 118, pp. 81 – 92, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366417305066

[145] M. Sardara, L. Muscariello, and A. Compagno, "A transport layer and socket api for (h) icn: Design, implementation and performance analysis," in *In Proceedings of the 5th ACM Conference on Information-Centric Networking (ACM ICN'18)*, 2018.

[146] C. Dannewitz, D. Kutscher, B. Ohlman, S. Farrell, B. Ahlgren, and H. Karl, "Network of information (netinf) – an information-centric networking architecture," *Computer Communications*, vol. 36, no. 7, pp. 721 – 735, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366413000364

[147] Z. Wang, H. Luo, H. Zhou, and J. Li, "R2t: A rapid and reliable hop-by-hop transport mechanism for information-centric networking," *IEEE Access*, vol. 6, pp. 15 311–15 325, 2018.

[148] S. Vakilinia and H. Elbiaze, "Latency control of icn enabled 5g networks," *Journal of Network and Systems Management*, vol. 28, no. 1, pp. 81–107, 2020.

[149] M. Scharf and A. Ford, "Mptcp application interface considerations," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-mptcp-api-07, January 2013, http://www.ietf.org/internet-drafts/draft-ietf-mptcp-api-07.txt.

[150] B. Hesmans, G. Detal, S. Barre, R. Bauduin, and O. Bonaventure, "Smapp: Towards smart multipath tcp-enabled applications," in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '15. New York, NY, USA: ACM, 2015, pp. 28:1–28:7.

[151] B. Hesmans and O. Bonaventure, "An enhanced socket api for multipath tcp," in *Proceedings of the 2016 Applied Networking Research Workshop*, ser. ANRW '16. New York, NY, USA: ACM, 2016, pp. 1–6.

[152] K. Grinnemo, T. Jones, G. Fairhurst, D. Ros, A. Brunstrom, and P. Hurtig, "Towards a flexible internet transport layer architecture," in *2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, June 2016, pp. 1–7.

[153] N. Khademi, D. Ros, M. Welzl, Z. Bozakov, A. Brunstrom, G. Fairhurst, K. Grinnemo, D. Hayes, P. Hurtig, T. Jones, S. Mangiante, M. Tuxen, and F. Weinrank, "Neat: A platform- and protocol-independent internet transport api," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 46–54, 2017.

[154] T. Pauly, B. Trammell, A. Brunstrom, G. Fairhurst, C. Perkins, P. Tiesel, and C. Wood, "An architecture for transport services," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-taps-arch-08, July 2020, https://datatracker.ietf.org/doc/draft-ietf-taps-arch/.

[155] J. J. Nielsen, R. Liu, and P. Popovski, "Ultra-reliable low latency communication using interface diversity," *IEEE Transactions on Communications*, vol. 66, no. 3, pp. 1322–1334, March 2018.

[156] B. D. Higgins, A. Reda, T. Alperovich, J. Flinn, T. J. Giuli, B. Noble, and D. Watson, "Intentional networking: opportunistic exploitation of mobile network diversity," in *Proceedings of the sixteenth annual international conference on Mobile computing and networking*. ACM, 2010, pp. 73–84.

[157] R. Davoli and M. Goldweber, "Msocket: Multiple stack support for the berkeley socket api," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, 2012, pp. 588–593.

[158] P. S. Schmidt, T. Enghardt, R. Khalili, and A. Feldmann, "Socket intents: Leveraging application awareness for multi-access connectivity," in *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '13. New York, NY, USA: ACM, 2013, pp. 295–300.

[159] R. Kapoor, G. Porter, M. Tewari, G. M. Voelker, and A. Vahdat, "Chronos: Predictable low latency for data center applications," in *Proceedings of the Third ACM Symposium on Cloud Computing*. ACM, 2012, p. 9.

[160] A. Belay, G. Prekas, A. Klimovic, S. Grossman, C. Kozyrakis, and E. Bugnion, "IX: A protected dataplane operating system for high throughput and low latency," in *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*. Broomfield, CO: USENIX Association, 2014, pp. 49–65.

[161] A. Belay, G. Prekas, M. Primorac, A. Klimovic, S. Grossman, C. Kozyrakis, and E. Bugnion, "Corrigendum to 'the ix operating system: Combining low latency, high throughput and efficiency in a protected dataplane'," *ACM Trans. Comput. Syst.*, vol. 35, no. 3, pp. 10:1–10:1, Dec. 2017.

[162] A. A. Siddiqui and P. Mueller, "A requirement-based socket api for a transition to future internet architectures," in *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. IEEE, 2012, pp. 340–345.

[163] D. A. Chekired, M. A. Togou, L. Khoukhi, and A. Ksentini, "5g-slicing-enabled scalable sdn core network: Toward an ultra-low latency of autonomous driving service," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 8, pp. 1769–1782, 2019.

[164] E. Balevi and R. D. Gitlin, "Unsupervised machine learning in 5g networks for low latency communications," in *2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC)*. Los Alamitos, CA, USA: IEEE Computer Society, dec 2017, pp. 1–2. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/PCCC.2017.8280492

[165] M. E. Morocho-Cayamcela, H. Lee, and W. Lim, "Machine learning for 5g/b5g mobile and wireless communications: Potential, limitations, and future directions," *IEEE Access*, vol. 7, pp. 137 184–137 206, 2019.

[166] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTPv1-U) (Release 17)," 3rd Generation Partnership Project, Tech. Rep. 29.281, 03 2021, version 17.0.0.

[167] V. Shankarkumar, L. Montini, T. Frost, and G. Dowd, "Precision time protocol version 2 (ptpv2) management information base," Internet Requests for Comments, RFC Editor, RFC 8173, June 2017.

[168] M. Amend and D. Hugo, "Multipath sequence maintenance," Working Draft, IETF Secretariat, Internet-Draft draft-amend-iccrg-multipath-reordering-03, April 2022. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-amend-iccrg-multipath-reordering-03

[169] D. Rico, "MTIP: The Multi-connection Tactile Internet Protocol," accesible in Github, Jan. 2021. [Online]. Available: https://github.com/deliarico/MTIP

[170] G. J. Holzmann, "The model checker spin," *IEEE Trans. Softw. Eng.*, vol. 23, no. 5, p. 279–295, May 1997.

[171] M. Kamel and S. Leue, "Formalization and validation of the general inter-orb protocol (giop) using promela and spin," *International Journal on Software Tools for Technology Transfer*, vol. 2, no. 4, pp. 394–409, 2000.

[172] T. Nakatani and F. Works, "Verification of group address registration protocol using promela and spin," in *Proc. Third SPIN Workshop*. Twente Univ., The Netherlands, 1997.

[173] P. Merino and J. Troya, "Modeling and verification of the itu-t multipoint communication service with spin," *Proceedings of the 2nd International Wrokshop on SPIN Verification*, 1996.

[174] H. Zhu and G. Singh, "A communication protocol for a vehicle collision warning system," in *2010 IEEE/ACM Int'l Conference on Green Computing and Communications and Int'l Conference on Cyber, Physical and Social Computing*, 2010, pp. 636–644.

[175] Y. Cai and D. Qi, "Control protocols design for cyber-physical systems," in *2015 IEEE Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, 2015, pp. 668–671.

[176] S. Chouali, A. Boukerche, A. Mostefaoui, and M. A. Merzoug, "Formal verification and performance analysis of a new data exchange protocol

for connected vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 15 385–15 397, 2020.

[177] S. Saini and A. Fehnker, "Evaluating the stream control transmission protocol using uppaal," *Electronic Proceedings in Theoretical Computer Science*, vol. 244, pp. 1–13, mar 2017. [Online]. Available: https://doi.org/10.4204%2Feptcs.244.1

[178] A. Rashid, O. Hasan, and K. Saghar, "Formal analysis of a zigbee-based routing protocol for smart grids using uppaal," in *2015 12th International Conference on High-capacity Optical Networks and Enabling/Emerging Technologies (HONET)*, 2015, pp. 1–5.

[179] X. Wu, H. Ling, and Y. Dong, "On modeling and verifying of application protocols of ttcan in flight-control system with uppaal," in *2009 International Conference on Embedded Software and Systems*, 2009, pp. 572–577.

[180] W. Guo, Y. Huang, J. Shi, Z. Hou, and Y. Yang, "A formal method for evaluating the performance of tsn traffic shapers using uppaal," in *2021 IEEE 46th Conference on Local Computer Networks (LCN)*, 2021, pp. 241–248.

[181] H. G., *The SPIN Model Checker - primer and reference manual*. Addison-Wesley, 2003.

[182] UP4ALL International AB, "Uppaal," https://uppaal.org/, 2022, online; accessed 18 November 2022.

[183] T. Hérault, R. Lassaigne, F. Magniette, and S. Peyronnet, "Approximate probabilistic model checking," in *Verification, Model Checking, and Abstract Interpretation*, B. Steffen and G. Levi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 73–84.

[184] G. J. Holzmann, *The SPIN Model Checker - primer and reference manual*. Addison-Wesley, 2004.

[185] P. Wolper, *Expressing Interesting Properties of Programs in Propositional Temporal Logic*. New York, NY, USA: Association for Computing Machinery, 1986, p. 184–193. [Online]. Available: https://doi.org/10.1145/512644.512661

[186] A. David, K. G. Larsen, A. Legay, M. Mikučionis, and D. B. Poulsen, "Uppaal smc tutorial," *International Journal on Software Tools for Technology Transfer*, vol. 17, no. 4, pp. 397–415, 2015.

[187] V.N., "Victoria Network," https://www.victoria-network.eu, 2023, online; accessed 4 May 2023.

[188] The Qt Company, "Cross-platform software development for embedded & desktop," 2021. [Online]. Available: https://www.qt.io/

[189] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Radio Access Network; NR; Radio Link Control (RLC) protocol specification (Release 16)," 3rd Generation Partnership Project, Tech. Rep. 38.322, 12 2020, version 16.2.0.

[190] Clearpath, "Husky - unmanned ground vehicle," https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/, 2022, online; accessed 22 Oct 2022.

[191] Open robotics, "Ros," https://www.ros.org/, 2022, online; accessed 22 Oct 2022.

[192] Q. De Coninck, M. Baerts, B. Hesmans, and O. Bonaventure, "A first analysis of multipath tcp on smartphones," in *Passive and Active Measurement*, T. Karagiannis and X. Dimitropoulos, Eds. Cham: Springer International Publishing, 2016, pp. 57–69.

[193] Q. De Coninck and O. Bonaventure, "Multipathtester: Comparing mptcp and mpquic in mobile environments," in *2019 Network Traffic Measurement and Analysis Conference (TMA)*, 2019, pp. 221–226.

[194] M. Giordani and M. Zorzi, "Non-terrestrial networks in the 6g era: Challenges and opportunities," *IEEE Network*, vol. 35, no. 2, pp. 244–251, 2020.

[195] G. Araniti, A. Iera, S. Pizzi, and F. Rinaldi, "Toward 6g non-terrestrial networks," *IEEE Network*, vol. 36, no. 1, pp. 113–120, 2021.

[196] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Release 17 Description; Summary of Rel-17 Work Items (Release 17) ," 3rd Generation Partnership Project (3GPP), Technical Report (TR) 21.917, 01 2023, version 17.0.1. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3937

[197] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Service requirements for the 5G system; Stage 1 (Release 19) ," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 22.261, 03 2023, version 19.2.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3107

[198] Apple, "ios support of satellite communications," https://www.apple.com/uk/newsroom/2022/12/emergency-sos-via-

satellite-available-in-france-germany-ireland-and-the-uk/, 2022, online; accessed 11 Apr 2023.

[199] Samsung, "Samsung support of satellite communications," https://news.samsung.com/global/samsung-electronics-introduces-standardized-5g-ntn-modem-technology-to-power-smartphone-satellite-communication, 2023, online; accessed 11 Apr 2023.

[200] A. R. Espada, M. Gallardo, A. Salmerón, L. Panizo, and P. Merino, "A formal approach to automatically analyse extra-functional properties in mobile applications," *Softw. Test. Verification Reliab.*, vol. 29, no. 4-5, 2019.

[201] L. Panizo and M.-d.-M. Gallardo, "Stan: analysis of data traces using an event-driven interval temporal logic," *Automated Software Engineering*, vol. 30, 11 2022.

[202] A. Díaz-Zayas, G. Caso, Ö. Alay, P. Merino, A. Brunström, D. Tsolkas, and H. Koumaras, "A modular experimentation methodology for 5g deployments: The 5genesis approach," *Sensors*, vol. 20, no. 22, p. 6652, 2020.

Appendix A

Resumen en Español

A.1 Introducción

A.1.1 Motivación

Internet Táctil (IT) se refiere a la red que permite a los humanos interactuar a distancia con otros humanos o con máquinas mediante la transmisión en tiempo real de información háptica (tacto, movimiento, vibración, fuerzas, etc.). Algunos ejemplos de casos de uso de Internet Táctil son el control remoto de alta precisión de robots, coches o drones. Estos casos de uso y sus aplicaciones imponen requisitos estrictos en términos de indicadores de rendimiento (KPI) como la latencia o la fiabilidad. En la Tabla A.1 se muestran los casos de uso más relevantes y sus requisitos. La naturaleza móvil de gran parte de los dispositivos implicados en estas comunicaciones provoca que las comunicaciones inalámbricas sean una parte fundamental de la infraestructura necesaria. Por ello, las redes móviles y los protocolos extremo a extremo deben evolucionar para dar soporte a estos requisitos del IT.

Esfuerzos recientes en el diseño de redes 5G han adaptado estos requisitos con la definición de comunicaciones de baja latencia ultra fiables (URLLC). Sin embargo, la pila TCP/IP clásica sigue siendo la principal interfaz con la red 5G. Esta pila, constituida por protocolos de transporte como TCP y UDP, fue definida hace más de 40 años y, a pesar de las continuas mejoras, sus dominios de aplicación se encuentran muy lejos de los de estas aplicaciones críticas. Por ejemplo, TCP se diseñó para ofrecer una fiabilidad del 100% que puede provocar grandes latencias, mientras que UDP no incluye ningún mecanismo de calidad específico y, por tanto, no proporciona ninguna mejora de la fiabilidad. Variantes de estos protocolos clásicos que proporcionan contribuciones más cercanas a los requisitos del Internet Táctil son el protocolo de transporte en tiempo real (RTP) [26] y extensiones como el MPRTP [27]. Sin embargo, estos protocolos se centran en ofrecer servi-

Tabla A.1: Casos de uso del Internet Táctil, KPIs y consideraciones

Caso de uso del Internet Táctil	**Latencia**	**Fiabilidad**	**Otras consideraciones**[1]
Control remoto en la industria	1-10ms (altamente dinámico) 1-100ms (dinámico)	99.99%-99.9999%	TP: Pequeño (40-250 bytes) VD: Baja (1-10 Mbps) AS: Pequeña (cientos de metros) OAC: Jitter bajo (1-100us)
Conducción remota	0.5-2ms (vital) 5-10ms (dinámico)	99.9%-99.999%	TP: Medio (50-1000 bytes) VD: Media (1-25 Mbps) AS: Alta (Kilómetros) OAC: Enlaces inalámbricos
Control de drones	0.5-2ms (cinestésico) 10ms (táactil)	99.9%-99.999%	TP: Pequeño (40-250 bytes) VD: Baja (1-10 Mbps) Zona de servicio: Alta (Kilómetros) OAC: Enlaces inalámbricos
Teleoperaciones quirúrgicas	5-10ms	99.9999%	TP: Medio (50-1000 bytes) VD: Baja (∼1 Mbps) AS: Alta (Kilómetros) OAC: Alta disponibilidad (99,999%)
Realidad virtual	5-10ms	99.99%	TP: Alto (800-1500 bytes) VD: Alta (30-1000 Mbps) AS: Variable

[1] Tamaño de Paquete (TP), Velocidad de Datos (VD), Area de Servicio (AS) y Otros Aspectos Clave (OAC)

cios multimedia en tiempo real con grandes cantidades de datos, en lugar de paquetes de tamaño limitado con estrictos plazos de entrega como son habituales en el control remoto de aplicaciones del Internet Táctil.

Es por ello que se han desarrollado algunas extensiones de estos protocolos en tiempo real, como Smoothed SCTP [30], y nuevos protocolos para el control remoto, como IRTP [31] o ETP [32] (más ejemplos en [33, 34]). No obstante, la mayor parte de estos protocolos de transporte no aprovechan los avances de las nuevas redes móviles, especialmente una de las tendencias clave en los nuevos protocolos de transporte: el uso de la multiconectividad (más información sobre evolución de las redes en [18, 35]). Además, estos protocolos usualmente carecen de la flexibilidad necesaria para adaptarse a los

requisitos específicos de cada caso de uso, e incluyen prestaciones adicionales que no son necesarias para el IT e, incluso, podrían ser contraproducentes añadiendo complejidad y sobrecarga.

En conclusión, las aplicaciones del Internet Táctil necesitan un protocolo de transporte que pueda adaptarse a los requisitos específicos de cada caso de uso y, al mismo tiempo pueda aprovechar todo el potencial de las redes subyacentes. En este contexto, presentamos MTIP, el protocolo de multi-conectividad para el Internet Táctil (*The Multi-connection Tactile Internet Protocol*).

A.1.2 Preguntas de Investigación y Contribuciones

El objetivo principal de este trabajo de tesis es explorar la viabilidad de protocolos de transporte para comunicaciones fiables de baja latencia. Para ello, la tesis se basa en varias preguntas de investigación. En esta sección, se presentan las preguntas de investigación y las contribuciones que trabajan en ofrecer respuestas a las mismas.

- Pregunta 1 (P1): ¿Cuáles son los nuevos casos de uso de comunicaciones fiables de baja latencia y cuáles son sus requisitos en términos de KPIs?
- Pregunta 2 (P2): ¿Cuáles son las principales soluciones y tecnologías extremo a extremo en las nuevas redes celulares (como 5G) para soportar comunicaciones fiables de baja latencia?
- Pregunta 3 (P3): ¿Están preparados los protocolos de transporte tradicionales para a soportar las nuevas comunicaciones fiables de baja latencia?
- Pregunta 4 (P4): ¿Cómo pueden los nuevos avances mejorar un protocolo de transporte para ofrecer un mejor servicio a las comunicaciones fiables de baja latencia?
- Pregunta 5 (P5): ¿Qué rendimiento puede obtenerse con un protocolo de transporte para aplicaciones fiables de baja latencia?

Con el fin de abordar estas preguntas de investigación, el trabajo de tesis cuenta con diferentes contribuciones que se basan y amplían la literatura existente sobre el tema, dando lugar a un conjunto de trabajos publicados.

- Contribución 1 (C1): Para abordar las cuestiones P1, P2 y P3, realizamos un estudio exhaustivo del de las soluciones de extremo a extremo para comunicaciones fiables de baja latencia, identificando áreas de investigación relevantes y tecnologías habilitadoras como la multi-conectividad o el uso de información de contexto a través de interfaces de programación (API).

- Contribución 2 (C2): Para responder a la P4, tomamos nuestros conocimientos previos de la C1 y diseñamos un protocolo de transporte para comunicaciones fiables de baja latencia: el protocolo de multiconectividad para el Internet Táctil (MTIP). Este protocolo se basa en tecnologías habilitadoras estudiadas, como la multiconectividad y el uso de información de contexto, para transportar comandos de control remoto de aplicaciones de Internet Táctil.

- Contribución 3 (C3): Continuando el trabajo de la C2, identificamos las principales propiedades que definen el protocolo y creamos modelos formales para realizar una verificación exhaustiva de su funcionamiento.

- Contribución 4 (C4): Abordando la P5, desarrollamos una implementación del protocolo y el API para exponer las capacidades del protocolo de forma flexible a las aplicaciones.

- Contribución 5 (C5): Completando la C4, realizamos una evaluación del rendimiento del protocolo mediante simulaciones y experimentos en el mundo real. Para el entorno simulado, utilizamos las herramientas Mininet [38] y Netem [39] para crear una topología con múltiples rutas y estudiar el impacto de distintas configuraciones del protocolo. Para los experimentos en entorno real, utilizamos una plataforma de pruebas disponible en la Universidad de Málaga [40] que posee múltiples conexiones celulares (como 4G, 5G NSA y 5G SA) para profundizar en el impacto de las distintas configuraciones del protocolo.

Las contribuciones han resultado en seis trabajos de investigación, los cuales se relacionan con las preguntas y las contribuciones como se muestra en la Figura A.1.

A.2 Antecedentes: Soluciones Extremo a Extremo de Baja Latencia y Fiabilidad en Redes 5G

El primer trabajo derivado de esta tesis se incluye en el Capítulo 2. En él se presenta un estado del arte de las tecnologías y soluciones de extremo a extremo para mejorar las comunicaciones fiables y de baja latencia. En primer lugar, se distinguen 3 líneas de investigación para mejorar las comunicaciones: la mejora de los protocolos de extremo a extremo, los avances en las redes actuales y el uso de la información contextual para mejorar la comunicación.

Se evalúan soluciones novedosas dentro de cada área de investigación, analizando indicadores clave de rendimiento como la fiabilidad, la latencia y rendimiento, así como otros parámetros relevantes, como la fiabilidad parcial

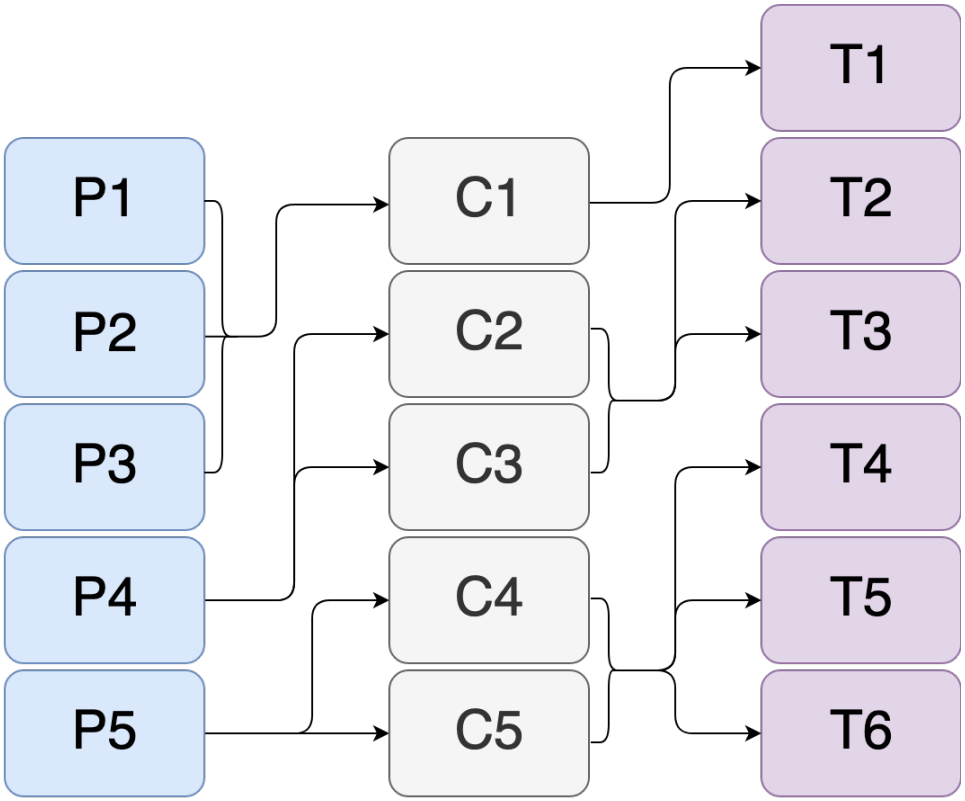


Figura A.1: Relación entre las preguntas de la investigación, las contribuciones y los trabajos resultantes

y el soporte de redes heterogéneas. A continuación, se analizan los métodos y técnicas utilizadas para mejorar el rendimiento de estas tecnologías habilitadoras. Finalmente, se identifican líneas de investigación relevantes para el desarrollo de nuevos protocolos de baja latencia y fiabilidad como son el uso de múltiples caminos o información de contexto a través de APIs.

A.3 *The Multi-connection Tactile Internet Protocol*

En el Capítulo 3 se el protocolo de multiconectividad para el Internet Táctil (MTIP). MTIP es un protocolo de transporte para el control remoto de aplicaciones de Internet Táctil. MTIP está diseñado para redes privadas grandes, entorno típico de casos de uso del IT como el industrial, que incluyen mecanismos de seguridad y sincronización. La base de MTIP es el uso de múltiples subenlaces o caminos independientes extremo a extremo (4G, 5G o 6G) para enviar datos de forma redundante, como se muestra en la Figura A.2. MTIP combina esta capacidad con el uso de información contextual para enviar datos de forma adaptándose a las necesidades de la aplicación.

Figura A.2: Uso de múltiples subenlaces en MTIP

Los algoritmos de envío y recepción de paquetes en MTIP están representados en las Figuras A.3 y A.4. En el envío, el protocolo selecciona qué subenlaces utilizar para duplicar paquetes y obtener un mejor rendimiento gracias a la redundancia. Esta decisión la realiza en dos fases. En una primera fase evalúa si en el receptor se están recibiendo demasiados paquetes duplicados o si se están perdiendo o retrasando demasiados paquetes. Los límites que regulan esta decisión están dados por los parámetros configurables `duplicate threshold` y `loss-late threshold`. En función de esta evaluación, MTIP reduce o aumenta el número de subenlaces en uso, respectivamente. A continuación, MTIP evalúa qué subenlaces específicos deben utilizarse para la transmisión. En esta operación, usa el parámetro `latency weight` para clasificar los subenlaces en términos de medidas de latencia (100% `latency weight`), fiabilidad (0% `latency weight`), o una media ponderada de los mismos. MTIP obtiene los parámetros gracias a exponer el API a la aplicación mientras que obtiene la caracterización de los subenlaces gracias a medidas de la red, utilizando información de contexto de tanto capas superiores como capas inferiores al protocolo. El API y su documentación está accesible en [169].

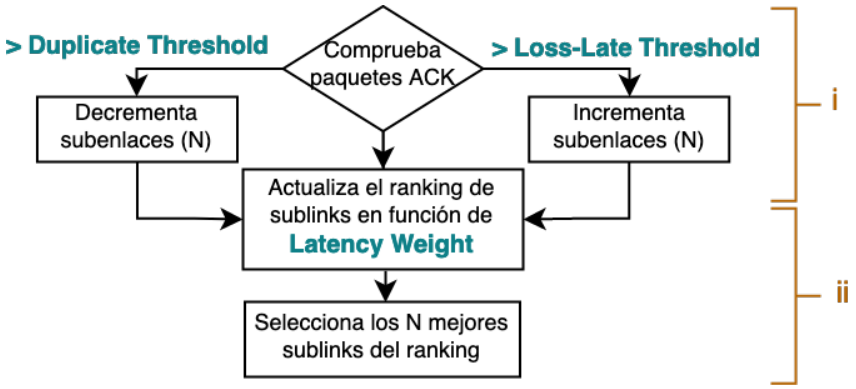


Figura A.3: Algoritmo de envío de MTIP

En la recepción, MTIP realiza tres comprobaciones antes de enviar los datos a la aplicación. En primer lugar, utiliza las marcas de tiempo de los paquetes para descartar paquetes que llegan con más retraso del que

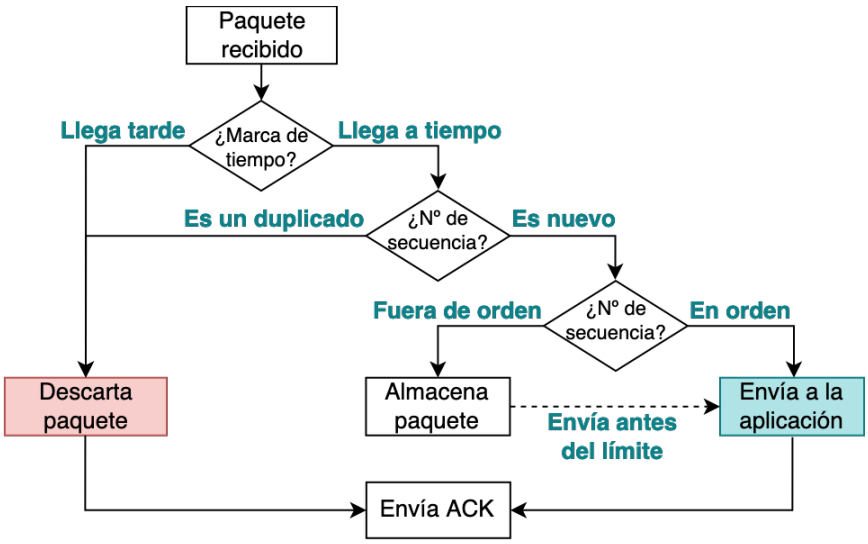


Figura A.4: Algoritmo de Recepción de MTIP

la aplicación define. En segundo lugar, utiliza los números de secuencia para comprobar que el paquete recibido no es un paquete duplicado, ya que los paquetes duplicados comparten número de secuencia. Finalmente, comprueba que el paquete recibido no está fuera de orden y puede enviarse a la aplicación. Si se detectase un reorden, el paquete quedaría en espera hasta que llegara el paquete perdido o se acercara el retraso límite que la aplicación ha estipulado.

A.4 Análisis Formal y Verificación de MTIP

El principal reto tras el diseño de MTIP es garantizar la corrección y el rendimiento de los algoritmos para enviar y recibir paquetes en MTIP. En el Capítulo 4, abordamos el análisis y la verificación automatizados de MTIP combinando las herramientas análisis y verificación SPIN y UPPAAL. SPIN es un verificador de modelos desarrollado por G. Holzmann [170] para la verificación lógica de software concurrente. Desde su origen, se ha utilizado en una serie de aplicaciones y protocolos [171, 172, 173], siendo hoy en día una poderosa herramienta utilizada para verificar sistemas modernos [174, 175, 176]. El potencial de SPIN radica en su capacidad para verificar la ausencia de bloqueos, bucles y propiedades complejas descritas mediante Lógica Temporal Lineal (LTL). Sin embargo, SPIN no soporta análisis de rendimiento ni modela el paso del tiempo. UPPAAL es también una herramienta de comprobación de modelos que se ha utilizado para analizar protocolos tanto tradicionales como novedosos [177, 178, 179, 180]. UPPAAL utiliza autómatas temporizados que permiten el análisis de propiedades rela-

cionadas con el rendimiento definidas mediante TCTL (Timed Computation Tree Logic).

Para realizar un análisis formal y una verificación exhaustiva, evaluamos MTIP desde dos perspectivas, la evaluación de la corrección con SPIN y la evaluación del rendimiento estadístico con UPPAAL.

A.4.1 Análisis Formal de Corrección

En cuanto a las propiedades de corrección, el funcionamiento de MTIP puede separarse en cuatro premisas que deben cumplirse en todo momento. Las cuatro propiedades de corrección de MTIP son las siguientes.

- **P1**: La aplicación nunca recibe mensajes duplicados.
- **P2**: La aplicación nunca recibe mensajes que hayan cumplido su plazo de entrega (paquetes que se considera que han llegado demasiado tarde).
- **P3**: La aplicación nunca recibe mensajes fuera de orden.
- **P4**: Todo paquete que no llegue duplicado o retrasado al protocolo, se enviará a la aplicación.

Para evaluar estas propiedades se han creado dos modelos en PROMELA, el lenguaje de modelado para SPIN, accesibles en [169]. La diferencia entre los modelos reside en la forma de modelar las marcas de tiempo, con un primer Modelo A, que implementa dos estados (paquete a tiempo y paquete tardío) y un Modelo B que implementa el tiempo de manera más gradual. Ambos modelos utilizan el test de Wolper, el cual demuestra que el uso de tres elementos de datos distintos (por ejemplo, rojo, azul y blanco) en una secuencia de un mensaje rojo y otro azul insertados aleatoriamente en una secuencia infinita de mensajes blancos, es suficiente para evaluar las propiedades de un modelo de un protocolo. De esta manera, traducimos las propiedades a las mostradas en A.1 a las siguientes. La propiedad **P1** comprueba la ausencia de mensajes duplicados en la aplicación, controlando que la aplicación nunca recibe más de un mensaje rojo o azul. **P2** comprueba que la aplicación nunca recibe *mensajes* que hayan cumplido su plazo, controlando que si todos los datos llegan tarde, no se envían a la aplicación. A continuación, **P3** comprueba que ningún dato rojo llega a la aplicación tras la recepción de un mensaje azul, utilizando el test de Wolper para demostrar la ausencia de fuera de orden. Por último, **P4** evalúa que si un paquete llega a tiempo y ningún mensaje de ese color ha llegado a la capa de aplicación, la aplicación recibirá finalmente ese mensaje.

```
ltl P1{ [] ( (nr<=1) && (nb<=1) ) }
#IFDEF LOSSES
ltl P2{ [] (nlr==2 -> [](nr==0)) && [] (nlb==2 ->[](nb==0))}
#ELSE
ltl P2{ [] (nlr>0 -> []!rr) && [] (nlb>0 -> []!rb) }
#ENDIF
ltl P3{ [] (rb -> []!rr) }
ltl P4{ [] ((receiver@ontime_red && nr==0) -> <>rr) &&
        [] ((receiver@ontime_blue && nb==0)-> <>rb)}
```

Código A.1: Propiedades de correción en LTL

Ejecutamos las propiedades en SPIN para comprobar la ausencia de errores y mostramos los resultados de las evaluaciones en la Tabla A.2. La primera columna (estados almacenados) muestra el número de estados únicos identificados por el proceso de verificación, la segunda columna (estados coincidentes) muestra el número de estados repetidos, o estados que ya se han encontrado y la última columna muestra la memoria total utilizada durante cada verificación.

Tabla A.2: Resultados de verificación en SPIN

Búsqueda en el espacio de estados		Estados almacenados		Estados coincidentes		Uso de memoria (MB)	
		No pérdidas	Pérdidas	No pérdidas	Pérdidas	No pérdidas	Pérdidas
Modelo B	No LTL	478333	6868279	453290	4224732	58.390	838.413
	P1	815041	10808408	692337	6483193	111.929	1484.309
	P2	2496867	34606828	3646870	29482333	342.893	4752.525
	P3	696317	8454045	628802	5280674	90.312	1096.487
	P4	1198662	15651815	1043621	13331607	164.611	2149.450
Modelo A	No LTL	12106416		10814529		1293.105	
	Asserts	18499903		11939937		2117.146	

A.4.2 Análisis Formal de Rendimiento

Las propiedades de rendimiento de MTIP definen el rendimiento del protocolo según las preferencias de la aplicación. Mientras que utilizar todos los subenlaces para enviar datos duplicados en todo momento suele ser lo más beneficioso para maximizar algunos KPIs, especialmente los relacionados con la fiabilidad, también incurriría en un uso excesivo de los recursos que podría afectar tanto a la red como a los dispositivos de Internet Táctil. Estas propiedades de rendimiento pueden resumirse en las siguientes:

- **P5**: El uso de más subenlaces tiende a resultar en más mensajes correctos recibidos.
- **P6**: El uso de más subenlaces tiende a resultar en más paquetes duplicados recibidos.

Para resolver esta disyuntiva entre el consumo de recursos y los KPI objetivo, MTIP expone una interfaz de programación de aplicaciones en la que

la aplicación puede indicar sus preferencias. A continuación, MTIP utiliza esta esta información, además de las mediciones de red, para seleccionar qué subenlaces utilizar para enviar datos. En esta sección, estudiamos el impacto de utilizar diferente número de subenlaces en este balance.

Para el análisis formal de rendimiento, hemos elaborado un modelo en UPPAAL accesible en [169]. El modelo es análogo al realizado en SPIN, con la principal diferencia que en UPPAAL los autómatas pueden definir relojes para controlar el paso del tiempo. En primer lugar, se realiza una comprobación de las propiedades de corrección de este modelo. Estas propiedades se muestran en el Código A.2 y son equivalentes a las de SPIN. Los resultados de la validación se presentan en la Tabla A.3.

```
tctl P1      { A[] not ((l_r == c_sblNum && a_r > 0)
                    || (l_b == c_sblNum && a_b > 0)) }
tctl P2      { A[] ((a_r <= 1) && (a_b <= 1)) }
tctl P3      { A[] not (r.r_r && a_b>0) }
tctl P4_rojo{ (o_r == 1 && a_r == 0) --> r.r_r }
tctl P4_azul{ (o_b == 1 && a_b == 0) --> r.r_b }
```

Código A.2: Propiedades de correción en TCTL

Tras la comprobación de correción, definimos dos conjuntos de propiedades para en análisis de rendimiento basadas en el test de Wolper. El primer conjunto de propiedades (**P5**) pretende comprobar la probabilidad de éxito (recibir paquetes rojos y azules), mientras que el segundo conjunto (**P6**), evalúa el desperdicio de recursos (recibir duplicados rojos y azules). Las propiedades se muestran en el Código A.3.

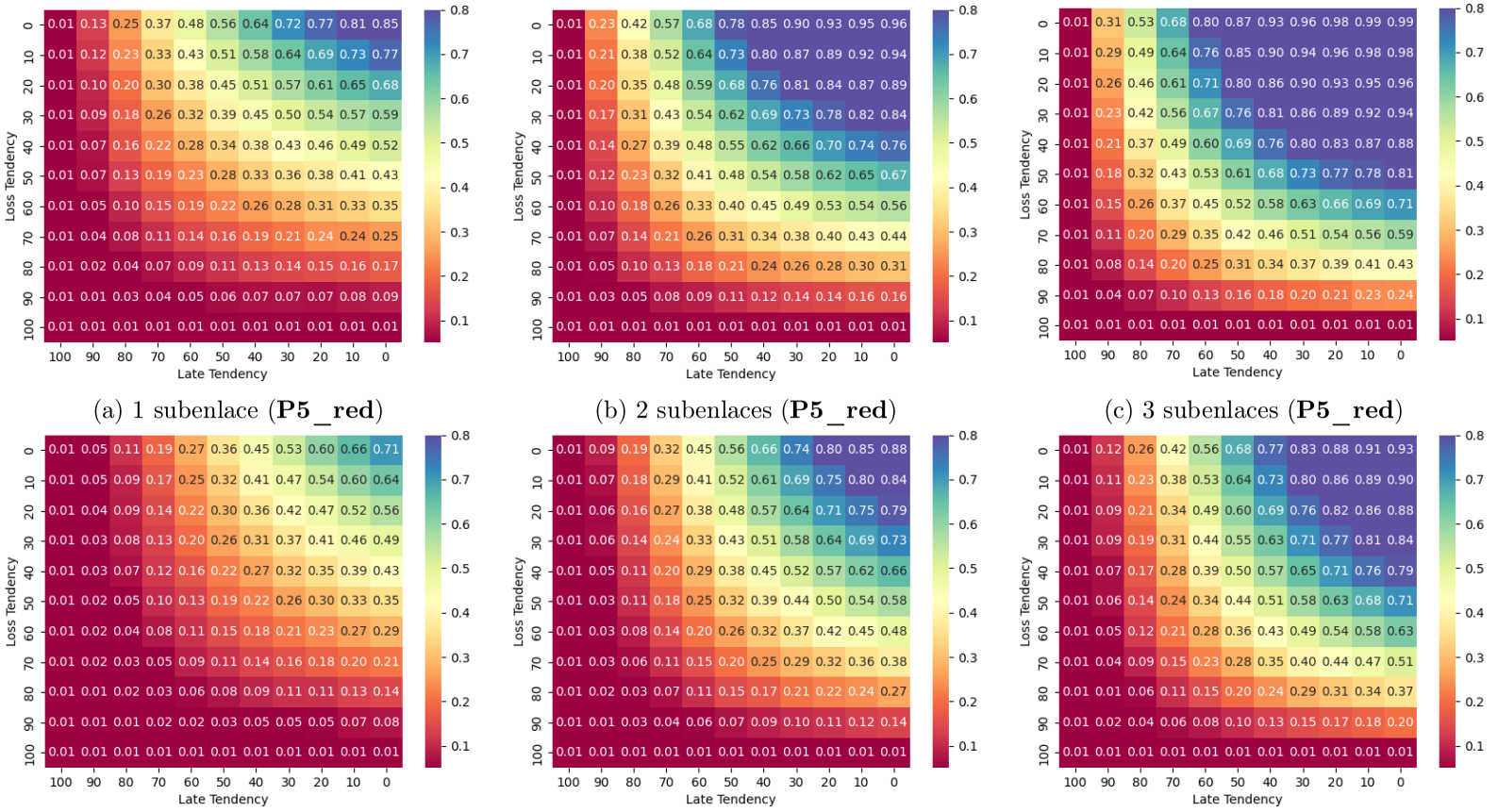
Tabla A.3: Verification results in UPPAAL

	Estados almacenados	**Estados coincidentes**	**Uso de memoria**
General	42105198	133566430	7581.232 MB
P1	42105198	133566430	7588.732 MB
P2	42105198	133566430	7588.732 MB
P3	42105198	133566430	7588.944 MB
P4_rojo	42105198	227928967	10735.344 MB
P4_azul	42105198	235998863	10933.564 MB

```
tctl P5_rojo{ Pr[<=100] (<> a_r > 0) }
tctl P5_azul{ Pr[<=100] (<> a_b > 0) }
tctl P6_rojo{ Pr[<=100] (<> d_r > 0) }
tctl P6_azul{ Pr[<=100] (<> d_b > 0) }
```

Código A.3: Propiedades de rendimiento en TCTL

Evaluamos el uso de 1, 2 y 3 subenlaces bajo configuraciones de 0% a 100% tendencia a pérdidas (*loss tendency*) y tendencia a retrasos (*late tendency*) en cambios de 10% en 10%. Los mapas de calor que muestran los resultados de la evaluación de **P5** se encuentran en la Figura A.5, y de **P6** en la Figura A.6.



(a) 1 subenlace (**P5_red**)

(b) 2 subenlaces (**P5_red**)

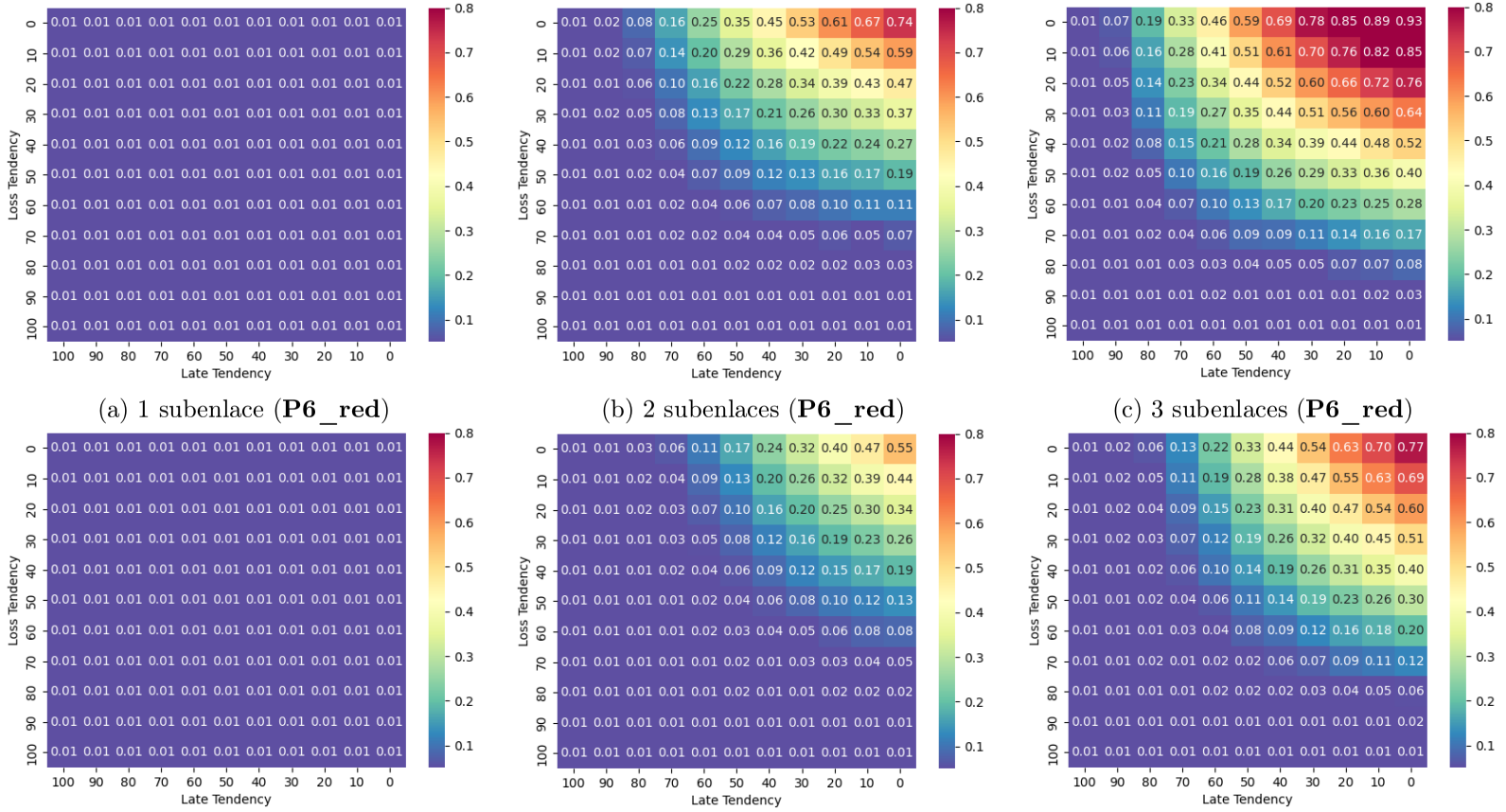
(c) 3 subenlaces (**P5_red**)

(d) 1 subenlace (**P5_blue**)

(e) 2 subenlaces (**P5_blue**)

(f) 3 subenlaces (**P5_blue**)

Figura A.5: Evaluación de la propiedad de rendimiento **P5**



(a) 1 subenlace (**P6_red**) (b) 2 subenlaces (**P6_red**) (c) 3 subenlaces (**P6_red**)

(d) 1 subenlace (**P6_blue**) (e) 2 subenlaces (**P6_blue**) (f) 3 subenlaces (**P6_blue**)

Figura A.6: Evaluación de la propiedad de rendimiento **P6**

Al evaluar las propiedades de rendimiento, podemos extraer las siguientes conclusiones. En primer lugar, en las propiedades de rendimiento **P5**, los colores de los gráficos sugieren que el uso de más subenlaces mejora la comunicación. Sin embargo, si el escenario es deficiente (véase tendencia de pérdidas 100% y tendencia de retrasos 100%), la comunicación no puede mejorarse mediante el uso de un mayor número de subenlaces. Las propiedades de rendimiento **P6** muestran un comportamiento análogo. En este caso, vemos que el uso de más subenlaces suele empeorar la propiedad. Esto se debe al hecho de que utilizar más subenlaces suele reflejarse en más duplicados y un mayor desperdicio de recursos. El balance de las propiedades de rendimiento de MTIP reside entonces en encontrar una configuración suficiente en términos tanto de fiabilidad como de desperdicio de recursos para los requisitos de la aplicación.

A.5 Implementación y Evaluación de MTIP

Tras el modelado de MTIP, se ha desarrollado en C/C++ una primera implementación de MTIP y del API, y se ha realizado una evaluación en un entorno de simulación y real. En el el Capítulo 5 se describen los detalles.

A.5.1 Evaluación en Entorno Simulado

En primer lugar, estudiamos el impacto de diferentes configuraciones del algoritmo de MTIP en un entorno de simulación. Este entorno está constituido por una topología virtual virtual generada por la herramienta Mininet [38] y diferentes escenarios con reglas Netem [39]. La topología consiste en dos extremos conectados por cuatro subenlaces. Los paquetes se envían a la típica velocidad de Internet Táctil de 1kHz con un límite de latencia de 10ms. Estudiamos el impacto de los principales parámetros del algoritmo MTIP, es decir, los umbrales `duplicate threshold`, `loss-late threshold` y el parámetro `latency weight`, en el principal compromiso del algoritmo: la fiabilidad, contando los paquetes perdidos y tardíos, frente al desperdicio de recursos, en términos de paquetes duplicados.

Los resultados se muestran en las Figuras A.7 y A.8. Los gráficos sugieren que un `loss-late threshold` bajo o un `duplicate threshold` alto provocan un mayor número de paquetes duplicados y menos pérdidas, ya que, en estos casos, es más probable que MTIP aumente el número de subenlaces, y viceversa en el caso opuesto. En cuestión de `latency weight`, observamos que valores altos del parámetro, los cuales utilizan medidas de latencia para seleccionar los subenlaces por los que enviar datos, suelen ser más beneficiosos que únicamente utilizar medidas de fiabilidad para los subenlaces.

Figura A.7: Efecto de los umbrales en diferentes escenarios

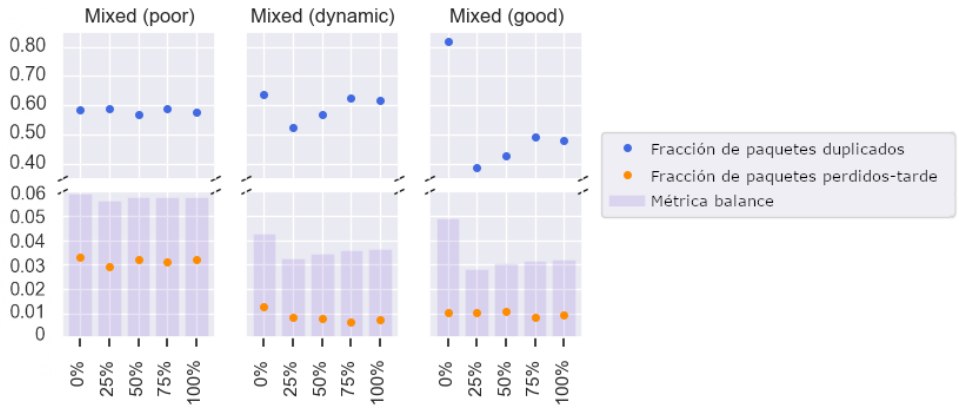


Figura A.8: Efecto de `latency weight` en los paquetes perdidos y duplicados

A.5.2 Evaluación en Entorno Real

Para la evaluación en un entorno real, hacemos uso del laboratorio Morse [40], ubicado en la Universidad de Málaga (UMA). El laboratorio ofrece una infraestructura que despliega una red totalmente privada, así como un marco de coordinación para la gestión de los elementos de red físicos y virtuales. La configuración consiste en un ordenador que actúa como controlador remoto y un segundo ordenador que actúa como dispositivo a bordo de un vehículo terrestre no tripulado (UGV). Cuatro redes diferentes los conectan: una red 4G, una red 5G NSA, una red 5G NSA con prioridad especial y una red 5G SA. El robot utilizado en las pruebas es el robot Husky Clearpath [190]. El robot Husky es un UGV de tamaño medio que ejecuta el Sistema Operativo para Robots (ROS) [191]. Para ejecutar aplicaciones ROS sobre MTIP, creamos una configuración proxy que encapsula los mensajes ROS sobre MTIP. La Figura A.9 presenta la configuración del entorno de comunicación.

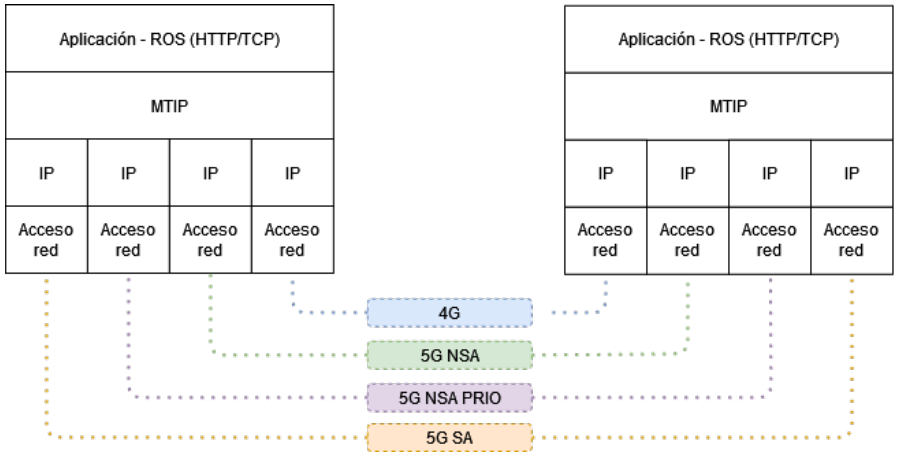
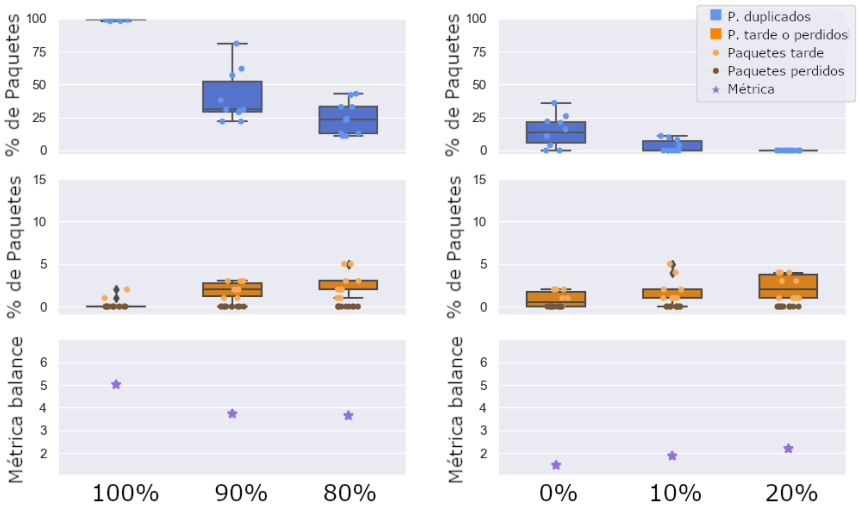


Figura A.9: Configuración de la comunicación entre el controlador y el robot

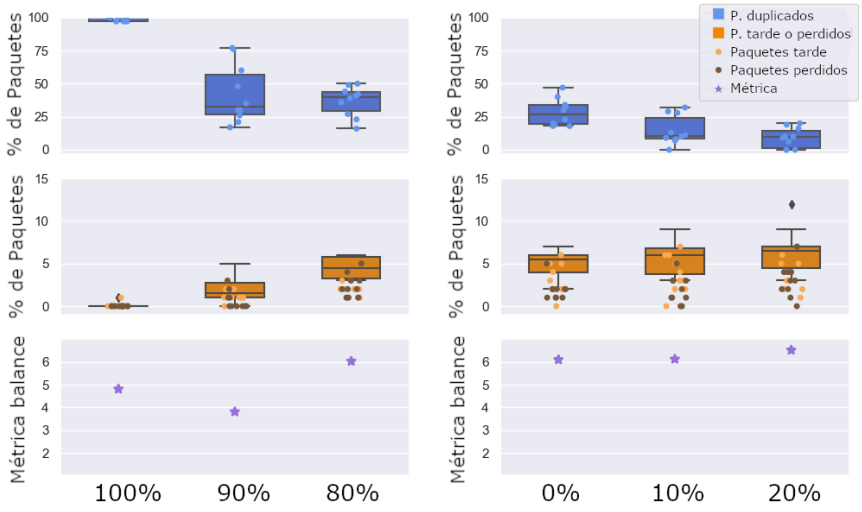
La evaluación es análoga a la realizada en el entorno simulado, estudiando el efecto de los parámetros en un entorno con pérdidas despreciables (escenario normal) y un entorno con pérdidas añadidas (medidas utilizando una cámara anecoica). Los resultados, que se muestran en las Figuras A.10 y A.11, complementan la evaluación previa, observándose que los umbrales más restrictivos tienden a dar lugar a más paquetes duplicados pero menos pérdidas, lo que hace que esta configuración más adecuada para entornos de red peores. Por el contrario, los umbrales menos restrictivos tienden a reducir duplicados a costa de algunas pérdidas, situación que puede ser recomendables en redes más estables.



(a) `Duplicate threshold` (b) `Loss-late threshold`

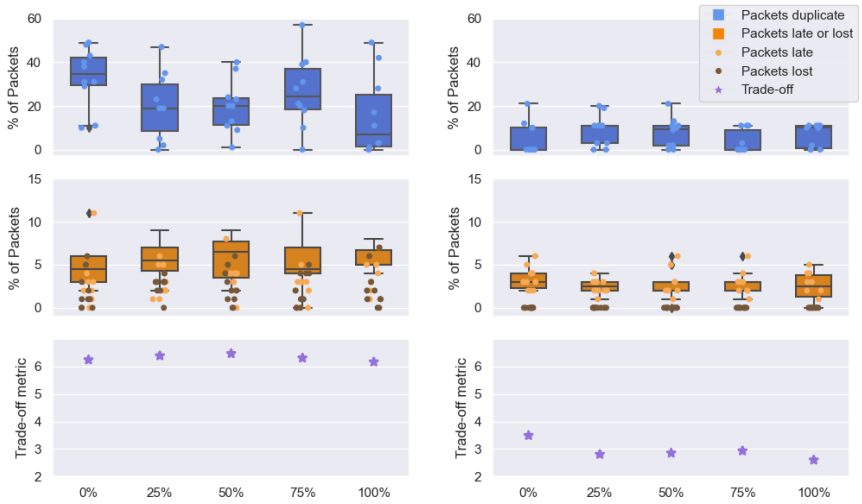
Figura A.10: Impacto de los umbrales el escenario normal

El resultado de la evaluación de la `latency weight` (Figura A.12) muestra que las distintas configuraciones del parámetro no afectan en gran medida al balance entre mensajes recibidos con éxito y paquetes duplicados. El escenario con problemas tiende a utilizar casi todos los subenlaces, por lo que la selección de los subenlaces en concreto no es crítica en este caso. En el escenario normal, la diferencia es algo más notable, sin embargo, dado que el escenario es dinámico y las pérdidas y paquetes tardíos no son numerosos, una vez que un subenlace es generalmente estable, la clasificación ya no es tan crítica. Estos comportamientos son observables en las Figuras A.13 y A.14.



(a) `Duplicate threshold` (b) `Loss-late threshold`

Figura A.11: Impacto de los umbrales el escenario con pérdidas



(a) Escenario con pérdidas (b) Escenario normal

Figura A.12: Impacto de `latency weight` en los escenarios

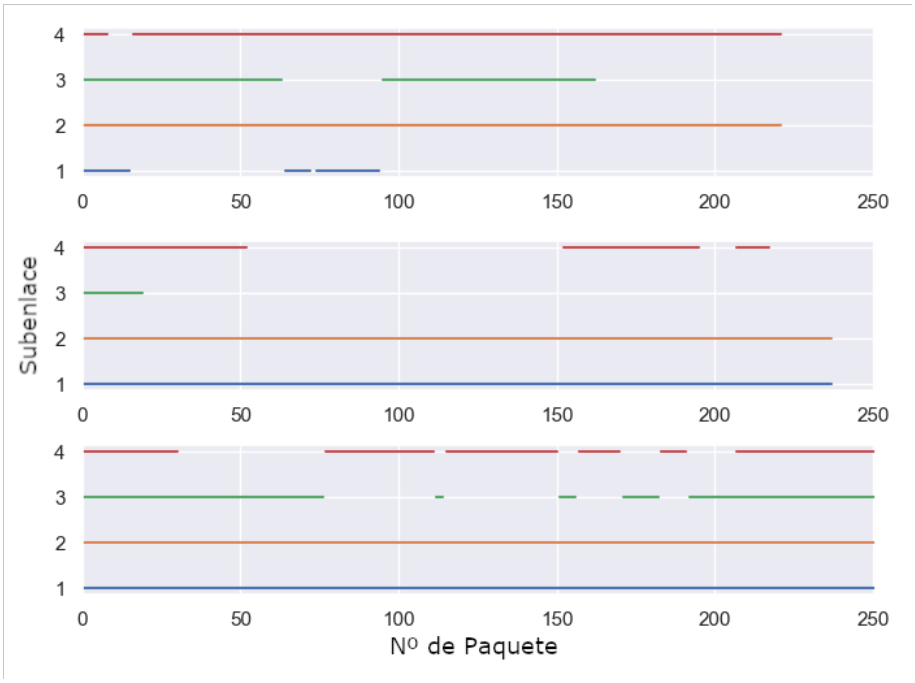


Figura A.13: Ejemplo de uso de subenlaces en el escenario con pérdidas

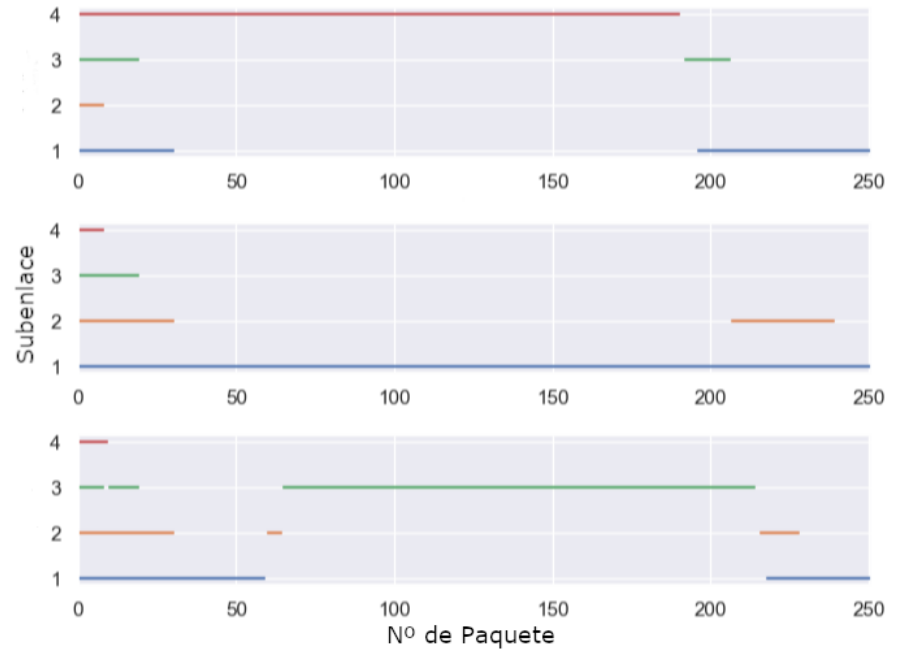
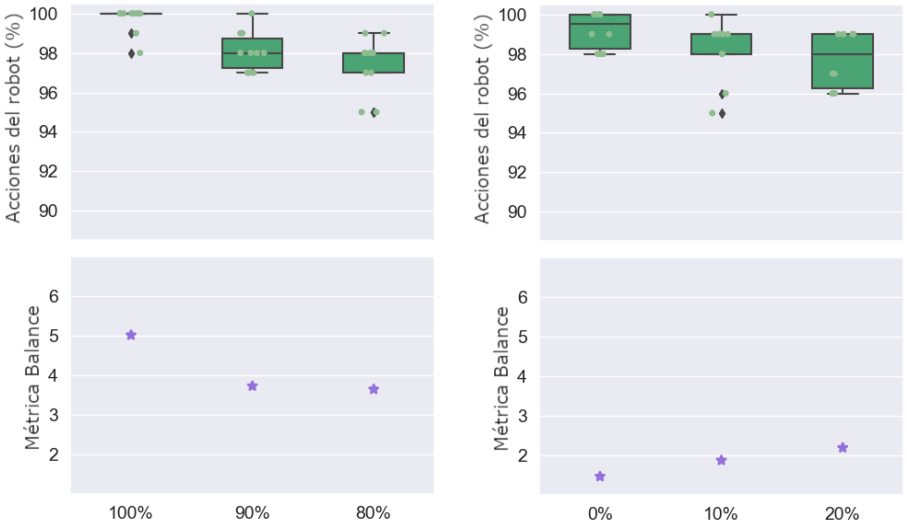
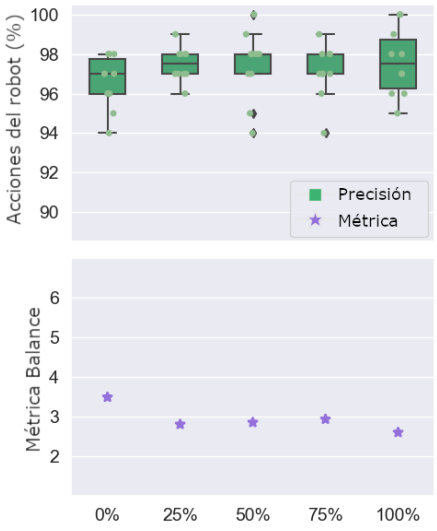


Figura A.14: Ejemplo de uso de subenlaces en el escenario normal

En el escenario real, se realiza una evaluación adicional para evaluar la precisión a nivel de aplicación. Las Figuras A.15 y A.16 muestran el porcentaje de órdenes recibidas correctamente a nivel de aplicación y la métrica que mide el balance con el desperdicio de recursos. En ambos casos, observamos que el caso más preciso no es siempre el mejor en términos del balance. Siendo interesante seleccionar configuraciones con menor precisión si se desea controlar el malgasto de recursos.

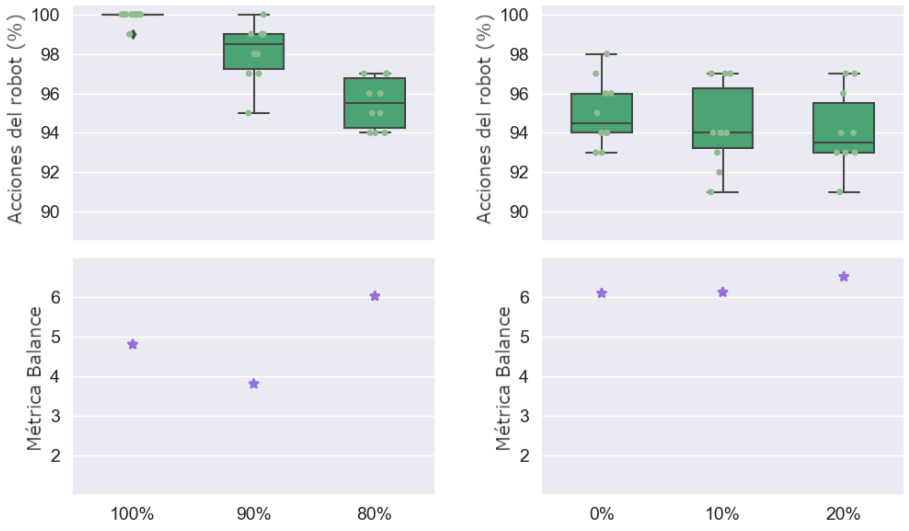


(a) `Duplicate threshold` (b) `Loss-late threshold`

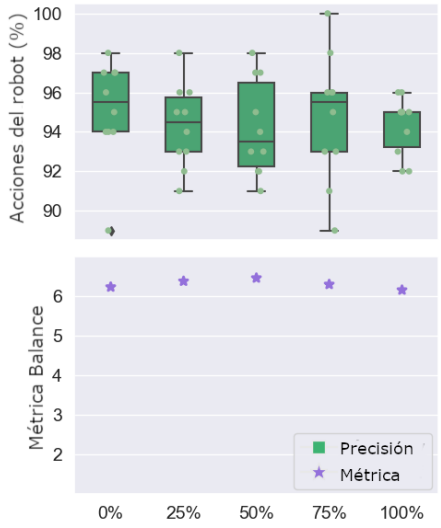


(c) `Latency weight`

Figura A.15: Precisión y balance en el escenario normal



(a) `Duplicate threshold` (b) `Loss-late threshold`



(c) `Latency weight`

Figura A.16: Precisión y balance en el escenario con pérdidas

A.6 Conclusión y Futuro Trabajo

En esta tesis, hemos presentado el Protocolo de Multiconectividad para el Internet Táctil (MTIP). MTIP es un protocolo de transporte para el control remoto de aplicaciones de Internet Táctil con requisitos estrictos como el control industrial industrial y el control remoto de drones y coches. La capacidad de MTIP para mejorar la comunicación se basa en el uso de múlti-

ples caminos (subenlaces) e información de contexto para tomar decisiones sobre el uso de estos subenlaces.

Para garantizar la corrección y el rendimiento de los algoritmos de intercambio de datos de MTIP, se desarrollaron varios modelos con fines de verificación. En primer lugar, la herramienta, SPIN se utiliza para realizar un análisis de la corrección de los algoritmos de intercambio de datos. Seguidamente, se realizó un análisis de rendimiento complementario utilizando la herramienta UPPAAL. El objetivo de este análisis es evaluar el impacto del uso de más subenlaces en el porcentaje de paquetes recibidos correctamente, así como en el malgasto de recursos por utilizar más subenlaces de los necesarios. Tras el modelado, se realizó una implementación del protocolo utilizando C/C++ y en el sistema operativo Linux. Esta implementación ha sido evaluada en un entorno simulado y en un entorno 5G utilizando un robot industrial. En las evaluaciones se estudian cómo afectan diferentes configuraciones del protocolo concretas en las decisiones del uso de los subenlaces, y se comprueba que configuraciones que utilizan una mayor redundancia son beneficiosas en entornos con mayores pérdidas, y viceversa.

En conclusión, el desarrollo del protocolo de multiconectividad para el Internet Táctil (MTIP) supone una contribución significativa al campo de las aplicaciones de Internet Táctil. La utilización de múltiples subenlaces por parte del protocolo proporciona un método de comunicación más robusto y fiable, mientras que el uso de la información de contexto permite el uso eficaz de MTIP en una amplia gama de aplicaciones de Internet Táctil.

Para el trabajo futuro, se han identificado diversas áreas de investigación relevantes. En primer lugar, se encuentra la implementación y evaluación de MTIP en dispositivos móviles. Estos dispositivos son especialmente importantes en el ámbito de la multiconectividad, ya que están equipados con múltiples interfaces de red (como wifi y celular). Por otro lado, la reciente proliferación de la combinación de redes no terrestres y celularess [194, 195] también se muestra como un área de interés para la evaluación del protocolo. Por último, se plantea la realización de una evaluación más exhaustiva del protocolo. Esto incluye aspectos de rendimiento en diferentes escenarios inalámbricos, como se plantea en [200, 201], así como la exploración del efecto de configuraciones adicionales del protocolo en la optimización del rendimiento. Para ello, puede ser necesario aplicar soluciones de aprendizaje automático o inteligencia artificial que puedan aprender de la red.

Appendix B

MTIP Implementation Details

This section presents MTIP implementation details with the API documentation and examples of remote control applications. This information is openly accessible in a in [169].

B.1 API

This section presents the API functions in a typical order of usage and the available options in current version.

B.1.1 Create a new socket

```
int mtip_socket(int options)
```

- **@param** `options`: `OPT_NOT_FULLMESH` (Disable connection fullmesh), `OPT_NOT_MONITOR` (Disable MTICP), `OPT_FULL_DEBUG` (Disable debug information).

B.1.2 Bind sublinks

```
int mtip_bind(int sd, char *addr, int port)
```

- **@param** `sd`: socket in use.
- **@param** `addr`: address to bind.
- **@param** `port`: port to bind.

B.1.3 Create a link

```
int mtip_link(int sd, uint8_t mode, char *addr, int port)
```

- **@param** `sd`: socket in use.

- **@param** `mode`: `CONTROLLER` (to connect), `DEVICE` (to listen).
- **@param** `addr`: address to link.
- **@param** `port`: port to link.

B.1.4 Set preferences

```
int mtip_preferences(int sd, char *prefs)
```

- **@param** `sd`: socket in use.
- **@param** `prefs`: preferences as described in Section B.2.2.

B.1.5 Set callback for receiving data

```
int mtip_receive(int sd, void (*callback)(char *, double))
```

- **@param** `sd`: socket in use.
- **@param** `callback`: function to callback when data arrives.
  - **@param** `char *`: message data.
  - **@param** `callback`: timestamp information.

B.1.6 Set a callback for when the connection finishes

```
int mtip_finished(int sd, void (*callback)())
```

- **@param** `sd`: socket in use.
- **@param** `callback`: function to callback when the connection finishes.

B.1.7 Send data

```
int mtip_send(int sd, char *message, int flag)
```

- **@param** `sd`: socket in use.
- **@param** `flag`: `FLAG_DATAPRIO` (data with priority), `0` (normal data).

B.1.8 Get feedback

```
int mtip_feedback(int sd, char *&feedback, int type)
```

- **@param** `sd`: socket in use.
- **@param** `feedback`: feedback information as described in Section B.2.1.
- **@param** `type`: `GENERAL` (characterization of each sublink).

B.1.9 Close Socket

```
int mtip_close(int sd)
```

- **@param** `sd`: socket in use.

B.2 Context information

Context information refers to the communication preferences that can be set in `mtip_preferences` and the feedback information that can be retrieved from `mtip_feedback`.

B.2.1 Feedback information

The application can obtain the feedback information presented in Table B.1.

Table B.1: Feedback information

Feedback	Description	Value
General	Network information, measured latency (ingress, egress) and measured reliability (ingress, egress) for each sublink.	Network information (sublink ID, IP and ports), latency (nanoseconds) and reliability (% of packets successfully forwarded to the application).
Sending	Status of each packet sent.	Packet on time, duplicate, late or ack not received.
Receiving	Status of each packet received.	Packet on time, duplicate, late or ack not received.

B.2.2 Communication preferences

The application can set the communication preferences presented in Table B.2.

Table B.2: Communication preferences

Communication preference	Description	Accepted values in current implementation	Default value in current implem.
Algorithm mode (AM)	It selects the algorithm that is going to be used for the selection of sublinks. The sublinks might be selected by the application (fixed selection) or dynamically by the protocol.	0: Use all sublinks (fixed selection) 1: Use one sublink (fixed selection) 2: Use best (N) sublinks (dynamic selection) 3: Use the MTIP Algorithm (dynamic selection)	3
Number of sublinks (N)	It selects the sublink that is in use (in the case of AM 1) or the number of sublinks that should be in use (in the case of AM 2).	0 to the maximum number of sublinks available	0
Maximum latency (deadline)	It defines the maximum end-to-end latency of the packets, namely the deadline.	0 to 1e9 nanoseconds (ns)	1e7 ns
Duplicate threshold (DT)	It defines the maximum percentage of duplicate packets that the MTIP algorithm considers reasonable (only used in AM 3).	0 to 100 %	50
Loss-late threshold (LT)	It defines the maximum percentage of loss or late packets that the MTIP algorithm considers reasonable(only used in AM 3).	0 to 100 %	10
Latency weight (LW)	It defines how the sublink ranking must be calculated, using just reliability measurements (LW 0), using only latency measurements (LW 100), or using a weighted mean of both measurements (LW from 0 to 100).	0 to 100 %	100

B.3 Application examples

To showcase an example of the use of MTIP API, Listing B.1 presents a remote control application while Listing B.2 shows the code for a controlled device application. The applications communicate through a link that consists on two differentiated sublinks. The code shows the use of MTIP extracting the API from a dynamic library and using the Qt library [188].

```
/**
* @file Device.cpp
* @author Delia Rico (deliarico@uma.es)
* @brief Example device using MTIP with two sublinks
* @version 1.0
* @date 2023-01-31
*
* @copyright Copyright (c) 2023
*
*/

#include <QCoreApplication>
#include <QDateTime>
#include <QHostAddress>
#include <QString>
#include <QNetworkInterface>
#include <iostream>
#include <thread>
#include <dlfcn.h>
#include <stdio.h>
#include <math.h>
#include <inttypes.h>
#include <sys/time.h>
#include "../MTIP/MTIP.h"

/** Buffer size **/
#define PREF_SIZE 260

/** Number of messages to be sent */
#define NUM_OF_MESSAGES 10000

/** IP and ports */
#define IP_1 "10.0.0.1"
#define IP_2 "11.0.0.1"
#define PORT 1111

/** Connection */
void reception_callback(char *order, double time);
void finished_callback();
void get_my_ip(char **theIP);
int g_socket_descriptor;

/** Message */
int g_signal_created[NUM_OF_MESSAGES];
double g_times[NUM_OF_MESSAGES];

using namespace std;

int main(int argc, char *argv[]) {

    QCoreApplication app(argc, argv);
    char *my_ip;

```

```
54      /* Extract API from a dynamic library */
55      void* handle = dlopen("libMTIP.so", RTLD_LAZY);
56
57      int (*mtip_socket)(int);
58      int (*mtip_bind)(int, char *, int);
59      int (*mtip_link)(int, uint8_t, char *, int);
60      int (*mtip_preferences)(int, char *);
61      int (*mtip_receive)(int, void (*)(char *, double));
62      int (*mtip_finished)(int, void (*)());
63      mtip_socket = (int (*)(int))
        ↪                   dlsym(handle, "mtip_socket");
64      mtip_bind = (int (*)(int, char *, int))
        ↪                   dlsym(handle, "mtip_bind");
65      mtip_link = (int (*)(int, uint8_t, char *, int))
        ↪                   dlsym(handle, "mtip_link");
66      mtip_preferences = (int (*)(int, char *))
        ↪                   dlsym(handle, "mtip_preferences");
67      mtip_receive = (int (*)(int, void (*)(char *, double)))
        ↪                   dlsym(handle, "mtip_receive");
68      mtip_finished = (int (*)(int, void (*)()))
        ↪                   dlsym(handle, "mtip_finished");
69
70      get_my_ip(&my_ip);
71      g_socket_descriptor = mtip_socket(OPT_NOT_FULLMESH |
        ↪ OPT_FULL_DEBUG);
72
73      if (g_socket_descriptor == -1) {
74          perror("Error opening socket");
75          exit(1);
76      }
77
78      if (mtip_bind(g_socket_descriptor,IP_1,PORT) == -1) {
79          perror("Error binding socket");
80          exit(1);
81      }
82
83      if (mtip_bind(g_socket_descriptor,IP_2,PORT) == -1) {
84          perror("Error binding socket");
85          exit(1);
86      }
87
88      free(my_ip);
89
90      char preferences_json[PREF_SIZE] = " {\n "
91                      " \" latency \": 1e+7 ,\n "
92                      " \" duplicateThreshold \": 20 ,\n "
93                      " \" losslateThreshold \": 5 ,\n "
94                      " \" latWeight \": 50"
95                      " } " ;
96
97      if (mtip_preferences(g_socket_descriptor,
        ↪ preferences_json) == -1) {
98          perror("Error adding preferences to the socket");
99          exit(1);
```

```
100     }
101
102     if (mtip_receive(g_socket_descriptor,
    ↪ reception_callback) == -1) {
103         perror("Error adding reception callback");
104         exit(1);
105     }
106
107     if (mtip_finished(g_socket_descriptor,
    ↪ finished_callback) == -1) {
108         perror("Error adding finished callback");
109         exit(1);
110     }
111
112     if (mtip_link(g_socket_descriptor, MTIP::Mode::DEVICE,
    ↪ NULL, 0) == -1) {
113         perror("Error opening link in listen mode");
114         exit(1);
115     }
116
117     return app.exec();
118 }
119
120 void reception_callback(char *order, double time) {
121
122     printf("-------------------------------\n");
123     printf("Received in APP: %s\n", order);
124     printf("-------------------------------\n");
125     g_times[atoi(order)] = time;
126     g_signal_created[atoi(order)] = atoi(order);
127
128 }
129
130 void finished_callback() {
131
132     printf("Time, Amplitude\n");
133     for (int i = 1; i < NUM_OF_MESSAGES; ++i) {
134         printf("%.0lf,%d\n", g_times[i], g_signal_created[i
    ↪ ]);
135     }
136     fflush(stdout);
137 }
138
139 void get_my_ip(char **the_ip) {
140     const QHostAddress &localhost = QHostAddress(
    ↪ QHostAddress::LocalHost);
141
142     for (const QHostAddress &address: QNetworkInterface::
    ↪ allAddresses()) {
143         if (address.protocol() == QAbstractSocket::
    ↪ IPv4Protocol && address != localhost) {
144             qDebug() << address.toString();
145             *the_ip = (char *) malloc((address.toString().
    ↪ size()) * sizeof(char));
```

```
146             strcpy(*the_ip, address.toString().toLocal8Bit
    ↪ ().data());
147         }
148     }
149 }
```

Listing B.1: Example device application using MTIP and Qt

```
1  /**
2  * @file Controller.cpp
3  * @author Delia Rico (deliarico@uma.es)
4  * @brief Simple controller using MTIP with two sublinks
5  * @version 1.0
6  * @date 2023-01-31
7  *
8  * @copyright Copyright (c) 2023
9  *
10 */
11
12 #include <QCoreApplication>
13 #include <QDateTime>
14 #include <QHostAddress>
15 #include <QString>
16 #include <QNetworkInterface>
17 #include <iostream>
18 #include <thread>
19 #include <dlfcn.h>
20 #include <unistd.h>
21 #include <stdio.h>
22 #include <sys/time.h>
23 #include <signal.h>
24 #include "../MTIP/MTIP.h"
25
26 /** Buffer size **/
27 #define PREF_SIZE 260
28
29 /** Number of messages to be sent */
30 #define NUM_OF_MESSAGES 10000
31 #define MAX_SIZE 15
32
33 /** Time interval for messages (ms) */
34 #define TIME_INTERVAL 10
35
36 /** IP and ports */
37 #define IP_1 "10.0.0.2"
38 #define IP_2 "11.0.0.2"
39 #define IP_REMOTE "10.0.0.1"
40 #define PORT 5555
41 #define PORT_REMOTE 1111
42
43 /** Connection */
44 void reception_callback(char *order, double time);
45 void send_message(int signum);
46 int (*mtip_send)(int, char *, int);
```

```
47 int (*mtip_feedback)(int, char *&, int);
48 void get_my_ip(char **the_ip);
49 int g_socket_descriptor;
50 bool g_is_sending;
51
52 /** Timer */
53 int set_ticker(long n_msecs);
54 int stop_ticker();
55
56 /** Message */
57 int g_signal_amplitude = 0;
58 int g_signal_created[NUM_OF_MESSAGES];
59 struct timespec g_times[NUM_OF_MESSAGES];
60 int get_amplitude();
61
62 using namespace std;
63
64 int main(int argc, char *argv[]) {
65
66     QCoreApplication app(argc, argv);
67     char *my_ip, *feedback;
68     int link_opened = -1;
69
70     /* Extract API from a dynamic library */
71     void* handle = dlopen("libMTIP.so", RTLD_LAZY);
72
73     int (*mtip_socket)(int);
74     int (*mtip_bind)(int, char *, int);
75     int (*mtip_link)(int, uint8_t, char *, int);
76     int (*mtip_preferences)(int, char *);
77     int (*mtip_receive)(int, void (*)(char*, double time));
78     int (*mtip_close)(int);
79     mtip_socket = (int (*)(int))
    ↪                   dlsym(handle, "mtip_socket");
80     mtip_bind = (int (*)(int, char *, int))
    ↪                   dlsym(handle, "mtip_bind");
81     mtip_link = (int (*)(int, uint8_t, char *, int))
    ↪                   dlsym(handle, "mtip_link");
82     mtip_send = (int (*)(int, char *, int))
    ↪                   dlsym(handle, "mtip_send");
83     mtip_preferences = (int (*)(int, char *))
    ↪                   dlsym(handle, "mtip_preferences");
84     mtip_feedback = (int (*)(int, char *&, int))
    ↪                   dlsym(handle, "mtip_feedback");
85     mtip_receive = (int (*)(int, void (*)(char *, double)))
    ↪                   dlsym(handle, "mtip_receive");
86     mtip_close = (int (*)(int))
    ↪                   dlsym(handle, "mtip_close");
87
88     get_my_ip(&my_ip);
89     g_socket_descriptor = mtip_socket(OPT_NOT_FULLMESH |
    ↪ OPT_FULL_DEBUG );
90
91     if (g_socket_descriptor == -1) {
```

```
92          perror("Error opening socket");
93          exit(1);
94      }
95
96      if (mtip_bind(g_socket_descriptor,IP_1,PORT) == -1) {
97          perror("Error binding socket");
98          exit(1);
99      }
100
101     if (mtip_bind(g_socket_descriptor,IP_2,PORT) == -1) {
102         perror("Error binding socket");
103         exit(1);
104     }
105
106     char preferences_json[PREF_SIZE] = " {\n "
107                     " \" latency \": 1e+7 ,\n "
108                     " \" duplicateThreshold \": 20 ,\n "
109                     " \" losslateThreshold \": 5 ,\n "
110                     " \" latWeight \": 50"
111                     " } " ;
112
113     if (mtip_preferences(g_socket_descriptor,
    ↪ preferences_json) == -1) {
114         perror("Error adding preferences to the socket");
115         exit(1);
116     }
117
118     if (mtip_receive(g_socket_descriptor,
    ↪ reception_callback) == -1) {
119         perror("Error adding reception callback");
120         exit(1);
121     }
122
123     while (link_opened < 0) {
124         link_opened = mtip_link(g_socket_descriptor,MTIP::
    ↪ Mode::CONTROLLER, IP_REMOTE, PORT_REMOTE);
125     }
126
127     free(my_ip);
128
129     signal(SIGALRM, send_message);
130     if (set_ticker(TIME_INTERVAL) == -1) {
131         perror("set_ticker");
132     } else {
133         g_is_sending = true;
134         while (g_is_sending) {
135             pause();
136         }
137     }
138
139     printf("--------------------------------\n");
140     printf("FINISHED.\n");
141     printf("--------------------------------\n");
142
```

```
143     sleep(3);
144
145     if (mtip_feedback(g_socket_descriptor, feedback, MTIP::
    ↪ GENERAL) == -1) {
146         perror("Error getting feedback");
147         exit(1);
148     }
149
150     fprintf(stdout, "Feedback: %s", feedback);
151     free(feedback);
152     sleep(1);
153
154     if (mtip_close(g_socket_descriptor) == -1) {
155         perror("Error closing");
156         exit(1);
157     }
158
159     return app.exec();
160 }
161
162 void reception_callback(char *order, double time) {
163
164     printf("--------------------------------\n");
165     printf("Received in APP: %s\n", order);
166     printf("--------------------------------\n");
167
168 }
169
170 void send_message(int signum) {
171
172     if (g_signal_amplitude >= (NUM_OF_MESSAGES - 1)) {
173         stop_ticker();
174
175         if (g_is_sending) {
176             g_is_sending = false;
177             for (int i = 1; i < NUM_OF_MESSAGES; ++i) {
178                 printf("%lld.%.9ld,%d\n", (long long)
    ↪ g_times[i].tv_sec, g_times[i].tv_nsec,
    ↪ g_signal_created[i]);
179                 fflush(stdout);
180             }
181         }
182     } else {
183         g_signal_amplitude++;
184         char *msg = (char *) malloc(sizeof(char) * MAX_SIZE
    ↪ ); /* deleted after sending */
185         sprintf(msg, "%d", g_signal_amplitude);
186         timespec_get(&g_times[g_signal_amplitude], TIME_UTC
    ↪ );
187         g_signal_created[g_signal_amplitude] =
    ↪ g_signal_amplitude;
188         mtip_send(g_socket_descriptor, msg, 0);
189     }
190 }
```


```
191
192 void get_my_ip(char **the_ip) {
193     const QHostAddress &localhost = QHostAddress(
    ↪ QHostAddress::LocalHost);
194
195     for (const QHostAddress &address: QNetworkInterface::
    ↪ allAddresses()) {
196
197         if (address.protocol() == QAbstractSocket::
    ↪ IPv4Protocol && address != localhost) {
198             qDebug() << address.toString();
199             *the_ip = (char *) malloc((address.toString().
    ↪ size()) * sizeof(char));
200             strcpy(*the_ip, address.toString().toLocal8Bit
    ↪ ().data());
201         }
202     }
203 }
204
205 /**
206  * Valid for less than 1 sec
207  */
208 int set_ticker(long n_msecs) {
209
210     struct itimerval new_timeset;
211     long n_sec, n_usecs;
212
213     n_sec = 0;
214     n_usecs = n_msecs * 1000L;
215
216     new_timeset.it_interval.tv_sec = n_sec;
217     new_timeset.it_interval.tv_usec = n_usecs;
218     new_timeset.it_value.tv_sec = n_sec;
219     new_timeset.it_value.tv_usec = n_usecs;
220
221     return setitimer(ITIMER_REAL, &new_timeset, NULL);
222 }
223
224 int stop_ticker() {
225
226     struct itimerval new_timeset;
227
228     new_timeset.it_interval.tv_sec = 0;
229     new_timeset.it_interval.tv_usec = 0;
230     new_timeset.it_value.tv_sec = 0;
231     new_timeset.it_value.tv_usec = 0;
232
233     return setitimer(ITIMER_REAL, &new_timeset, NULL);
234 }
```

Listing B.2: Example controller application using MTIP and Qt