Institute of Parallel and Distributed Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Roaming with Deterministic Real-Time Guarantees in Wireless Time-Sensitive Networking

Lucas Haug

Course of Study:	Informatik
Examiner:	Prof. Dr. Christian Becker
Supervisor:	Dr. rer. nat. Frank Dürr
Commenced:	October 5, 2022
Completed:	April 5, 2023

Abstract

As Cyber-Physical Systems (CPSs) become increasingly popular in various domains such as Industrial Internet of Things (IIoT) and autonomous vehicles, the demand for deterministic real-time communication with a high reliability and bounded network delay and delay variance (jitter) has grown. In CPSs with networked sensors and actuators, these deterministic real-time bounds are often crucial in order to provide safety guarantees. Major standardization organizations like the Institute of Electrical and Electronics Engineers (IEEE) have acknowledged the necessity for deterministic networks, resulting in a set of standards known as Time-Sensitive Networking (TSN), which enable deterministic communication in wired Ethernet networks.

Many CPSs, however, necessitate mobility and therefore rely on wirelessly connected devices, such as a worker wearing an exoskeleton in order to carry heavy loads. To this end, the TSN standards will also be part of the upcoming Wi-Fi 7 standard (IEEE 802.11be). Although the increased flexibility offered by the mobility of devices in these scenarios is advantageous, it presents new challenges, such as controlling access to the shared wireless transmission medium and managing handovers such that deterministic real-time guarantees are maintained.

In this work, we investigate these challenges and provide novel approaches to improve the reliability, delay and jitter in wireless TSN. We first modify an already existing Integer Linear Program (ILP) to generate schedules for the Time-Aware Shaping (TAS) in a wireless network environment. The necessity for this arises because the wireless transmission medium utilizes a shared access method, whereas existing approaches are limited to point-to-point Ethernet connections. Furthermore, we provide a novel seamless handover approach for wireless TSN utilizing two wireless interfaces in a single device and extend it with a proactive handover approach in order to allow for smoother handovers with a greater reliability. In order to analyze our approaches, we extend the INET framework of the OMNeT++ simulator with an implementation of our approaches. Our evaluation shows that delay and jitter are mainly influenced by the random back-off algorithm of the channel access procedure in Wi-Fi indicating research topics for future work. Moreover, we were able to significantly improve the reliability for wireless TSN by employing our proactive handover approach in the simulation.

Kurzfassung

Da Cyber-Physical System (CPS) in verschiedenen Bereichen wie Industrial Internet of Things (IIoT) und autonomen Fahrzeugen immer beliebter werden, ist die Nachfrage nach deterministischer Echtzeitkommunikation mit hoher Zuverlässigkeit und begrenzter Netzwerkverzögerung und Verzögerungsvarianz (Jitter) gestiegen. In CPSs mit vernetzten Sensoren und Aktoren sind diese deterministischen Echtzeit-Grenzen oft entscheidend, um Sicherheitsgarantien zu bieten. Wichtige Standardisierungsorganisationen wie die Institute of Electrical and Electronics Engineers (IEEE) haben die Notwendigkeit deterministischer Netzwerke erkannt, was zu einer Reihe von Standards geführt hat, die als Time-Sensitive Networking (TSN) bekannt sind und eine deterministische Kommunikation in kabelgebundenen Ethernet-Netzwerken ermöglichen.

Viele CPSs erfordern jedoch die Mobilität von Geräten und sind daher auf drahtlos verbundene Geräte angewiesen, wie z. B. ein Arbeiter, der ein Exoskelett trägt, um schwere Lasten zu tragen. Zu diesem Zweck werden die TSN-Standards auch Teil der kommenden Wi-Fi 7-Norm (IEEE 802.11be) sein. Die erhöhte Flexibilität, die die Mobilität der Geräte in diesen Szenarien bietet, ist zwar vorteilhaft, bringt aber auch neue Herausforderungen mit sich, wie z. B. die Steuerung des Zugriffs auf das gemeinsam genutzte drahtlose Übertragungsmedium und die Verwaltung von Handovers in einer Form, die die Einhaltung von deterministischen Echtzeitgarantien gewährleistet.

In dieser Arbeit untersuchen wir diese Herausforderungen und bieten neuartige Ansätze zur Verbesserung der Zuverlässigkeit, der Verzögerung und des Jitters für drahtloses TSN an. Zunächst modifizieren wir ein bereits bestehendes Integer Linear Program (ILP), um Schedules für Time-Aware Shaping (TAS) in einer drahtlosen Netzwerkumgebung zu generieren. Die Notwendigkeit hierfür ergibt sich daraus, dass das drahtlose Übertragungsmedium ein Shared-Access-Verfahren verwendet, während bestehende Ansätze auf die Unterstützung von Punkt-zu-Punkt-Ethernet-Verbindungen beschränkt sind. Darüber hinaus stellen wir einen neuartigen Handover-Ansatz für drahtloses TSN vor, der zwei drahtlose Netzwerkschnittstellen in einem einzigen Gerät nutzt, und erweitern ihn mit einem proaktiven Handover-Ansatz, um einen reibungsloseren Handover mit höherer Zuverlässigkeit zu ermöglichen. Um unsere Ansätze zu analysieren, erweitern wir das INET-Framework des OMNeT++-Simulators mit einer Implementierung unserer Ansätze. Unsere Auswertung zeigt, dass Verzögerung und Jitter hauptsächlich durch den zufälligen Back-Off-Algorithmus des Kanalzugriffsverfahrens in Wi-Fi beeinflusst werden, was auf Forschungsthemen für zukünftige Arbeiten hinweist. Darüber hinaus konnten wir die Zuverlässigkeit für drahtloses TSN durch den Einsatz unseres proaktiven Handover-Ansatzes in der Simulation deutlich verbessern.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

AC Access Category. 27

ACK Acknowledgement. 26

AID Association Identifier. 29

AIFS Arbitration Interframe Space. 27

AP Access Point. 17, 22

BSS Basic Service Set. 22

BSSID Basic Service Set Identifier. 22

CBS Credit-Based Shaper. 31

CFP Contention Free Period. 26

CPS Cyber-Physical System. 3, 4, 17

CRC Cyclic redundancy check. 25

CSMA/CA Carrier Sense Multiple Access with Collision Avoidance. 25

CSP Constraint Satisfaction Problem. 17, 31

CW Contention Window. 26

DA Destination Address. 24

DCF Distributed Coordination Function. 25

DEI Drop Eligible Indicator. 20, 21

DFS Dynamic Frequency Selection. 39

DIFS DCF Interframe Space. 25

DS Distrubution System. 22

EAP Extensible Authentication Protocol. 28

EDCA Enhanced Distributed Channel Access. 27

ESS Extended Service Set. 22

ESSID Extended Service Set Identifier. 22

FCS Frame Check Sequence. 20

FDMA Frequency Division Multiple Access. 29

FIFO First in - First out. 54

FILS Fast Initial Link Setup. 28

GCL Gate Control List. 30

HC Hybrid Coordinator. 27

HCCA HCF Controlled Channel Access. 27

HCF Hybrid Coordination Function. 25, 27

HT High Throughput. 24

IEEE Institute of Electrical and Electronics Engineers. 3, 4, 17

IIoT Industrial Internet of Things. 3, 4

ILP Integer Linear Programming. 17, 19, 31, 40

IoT Internet of Things. 38

JRaS Joint Routing and Scheduling. 31, 37

JSSP Job Shop Scheduling Problem. 40

lhs left-hand side. 32

MAC Media Access Control. 25

MCTS Monte Carlo Tree Search. 40

MIMO Multiple Input Multiple Output. 22

MU-MIMO Multi-User MIMO. 30

NAV Network Allocation Vector. 26

OBSS Overlapping Basic Service Set. 38

PCF Point Coordination Function. 25, 26

PCP Priority Code Point. 20

PIFS PCF Interframe Space. 26

PTP Precision Time Protocol. 37

QoS Quality of Service. 25

RA Recieving Address. 24

rhs right-hand side. 32

RSSI Received Signal Strength Indicator. 39

SA Source Address. 24

SIFS Short Interframe Space. 23, 26

SMT Satisfiability Modulo Theory. 41

SNR Signal-to-noise Ratio. 39

STA Station. 17, 22

TA Transmitting Address. 24

TAS Time-Aware Shaping. 3, 4, 30, 31

TCI Tag Control Information. 20

TDMA Time Division Multiple Access. 31

TPID Tag Protocol Identifier. 20

TS Transmission Selection. 21, 30

TSA Transmission Selection Algorithm. 30

TSN Time-Sensitive Networking. 3, 4, 17, 19, 30

TSPEC Traffic Specification. 27

TWT Target Wake-up Time. 38

TXOP Transmission Opportunity. 27

VHT Very High Throughput. 24

VID VLAN Identifier. 20

VLAN Virtual Local Area Network. 20

WEP Wired Equivalent Privacy. 28

WM Wireless Medium. 24

WPA Wi-Fi Protected Access. 28

1 Introduction

In Cyber-Physical Systems (CPSs), deterministic real-time guarantees such as a high reliability, bounded delay and jitter are key-requirements for safety. These systems often employ distributed sensors and actuators exchanging data over a network. The actions of these actuators in a Constraint Satisfaction Problem (CSP) often have an effect on our physical world. Failing to meet specific real-time guarantees can lead to a high economical damage or the harm of humans in various scenarios. One possible example is a worker in a factory wearing an exoskeleton which receives data from distributed sensors. Failing to transmit specific sensor data in time may expose the worker to a dangerous situation.

The demand of such real-time networks has increased rapidly in the last years, which lead to a set of standards by the Institute of Electrical and Electronics Engineers (IEEE), namely Time-Sensitive Networking (TSN) for Ethernet networks. However, many CPS, like the example mentioned above, necessitate wireless communication due to the mobility requirements that allow for free movement. The IEEE recognized this need and aims to implement TSN functionality in the upcoming Wi-Fi 7 standard (IEEE 802.11be). Employing TSN in a wireless environment introduces new challenges. One of these challenges is the usage of a shared wireless transmission medium, as it necessitates for a controlled access among all involved devices while still preserving the real-time guarantees. More challenges are introduced by the mobility of devices, such as the worker wearing an exoskeleton moving around within the factory. If the worker only moves in a small, pre-defined area within the range of one Access Point (AP), the movement only results in a small change in the delay of transmitted and received packets. However, if the area covered by a worker is greater than the range of an AP, the wireless interface embedded in the exoskeleton needs to perform a handover to another AP. During this handover, there is no active connection to an AP leading to a loss of packets. Furthermore, after a handover, the route of packets from and to the Station (STA) (e.g. the exosceleton) changes. This results in a change of delay and thus a greater jitter.

One possibility to ensure deterministic real-time guarantees is to calculate so-called *Schedules* for streams of time-sensitive applications. By employing these schedules in the configuration of network devices, we can reserve bandwidth in order to provide deterministic real-time guarantees for the corresponding streams. One way to calculate these schedules is the usage of an Integer Linear Programming (ILP) model. Our first contribution of this work, is the extension of an already existing ILP model in order to support a wireless transmission medium and handovers. Furthermore, we propose a handover approach utilizing multiple wireless interfaces in a single STA. We improve upon this approach by proactively handing over devices based on their location and also use this location to experiment with a new approach which eagerly disassociated the second wireless interface after a handover. In order to analyze our approaches in regard to reliability, delay and jitter we extend the INET framework of the OMNeT++ network simulator with TSN functionality for wireless devices (i.e. STAs and APs) and implement our proposed handover approach.

We start this work by introducing the required technical background in Chapter 2. Thereafter, in Chapter 3, we delve into already existing research related to our proposed approaches. This includes work researching the challenges of employing TSN in wireless networks, work on other handover approaches and on the calculation of schedules. Chapter 4 contains a formal description of our network environment, our assumptions on the configuration of the network and its devices and the ILP model we build upon. Furthermore, we formalize the problem statement and provide the objectives for our approaches. In the main Chapter 5, we present our approaches. This includes our extension of the ILP model, our proposed handover approach and our method to reconfigure network devices to adapt for changing routes. The following Chapter 6 describes how we extend the INET framework to support TSN in wireless environments and how we implement our handover approach in OMNeT++. Following that, we analyze our proposed approaches in regard to reliability, delay and jitter in Chapter 7. First, by evaluating the results of our ILP model and thereafter by analyzing the results based on a simulated network environment in OMNeT++. Finally, we conclude our work in Chapter 8 and give an outlook on how to improve upon our approaches in future work.

2 Background

Before we are able to formalize our problem and explain our approaches, it is crucial to understand the underlying technologies. To this end, this chapter provides an introduction into the technologies relevant for our work. We first introduce the IEEE standards for about Ethernet and wireless networks. Thereafter, we provide an overview over the Time-Sensitive Networking (TSN) standards. We then explain the purpose and functionality of Integer Linear Programming (ILP). Thereafter, we provide an overview over the event simulator OMNeT++ and a framework for Ethernet and WLAN simulations in OMNeT++, called INET. The last section contains a description of graphs and graph algorithms, which allow us to formally model our network and perform calculations on them.

2.1 Ethernet

In order to understand TSN specific functionality, it is crucial to have a basic knowledge about the structure and transmission of Ethernet frames first. Therefore, the following two sections explain the structure of an Ethernet frame and the forwarding process in an Ethernet switch, respectively.

2.1.1 Ethernet Frame

In the following, we examine the structure of an Ethernet frame according to the [IEE22a] standard. We provide the description of the fields in an Ethernet header based on Figure 2.1 with a special focus on the IEEE 802.1Q header, as it contains necessary information for TSN such as the priority of a frame.

Source/Destination MAC The first two fields of the header contain the source and destination MAC addresses of the frame, respectively. While the source address is always an individual address, the *Destination MAC* field can contain an individual MAC address or a group MAC address (i.e. multicast or broadcast).

Length/Type The meaning of the *Length/Type* field depends on its numeric value:

- If the value is at most 1500 (`0x05DC`), it indicates the length of the following *Payload* field in Bytes.
- If the value is equal to or greater than 1536 (`0x600`) it serves as an *EtherType* field indicating the type of the frame. For example, `0x8100` indicates an optional IEEE 802.1Q tag as described below.

Payload The *Payload* field holds the payload from the initial sender. It has a maximum capacity of 1,500 B.

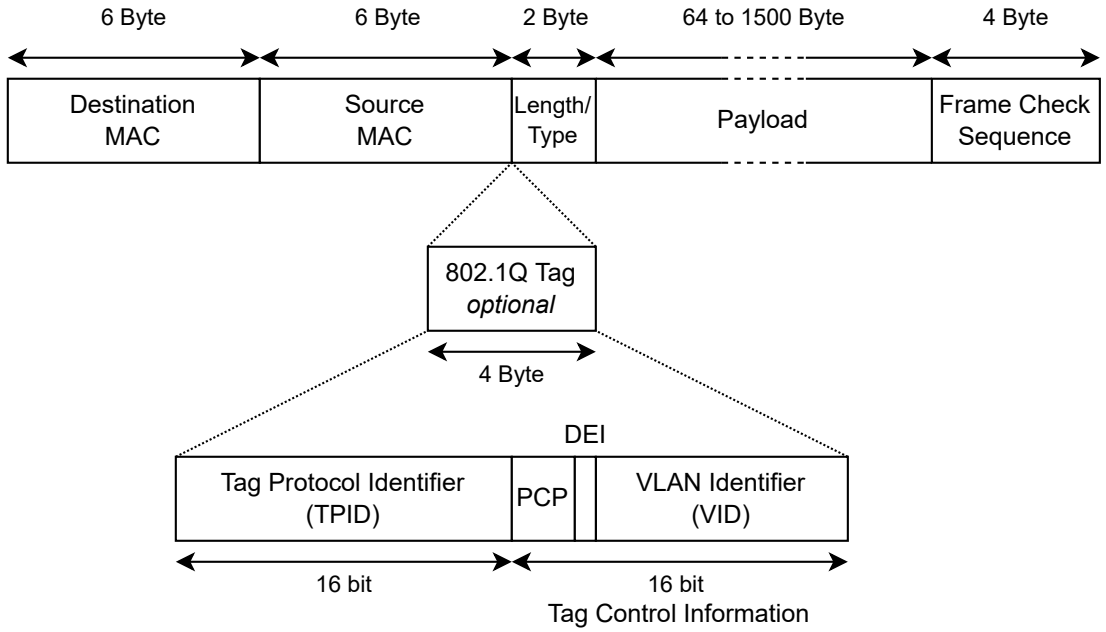


Figure 2.1: IEEE 802.3 Ethernet Frame including the optional IEEE 802.1Q VLAN tag [Hau20; IEE22a; IEE22b].

Frame Check Sequence (FCS) The final element of an Ethernet frame is a 4 B Frame Check Sequence (FCS) which includes a checksum to detect corrupted frames.

IEEE 802.1Q Tag

The IEEE 802.1Q tag, also known as the *Virtual Local Area Network (VLAN)* tag, is defined in the IEEE 802.1Q standard [IEE22b]. It provides functionality such as the support for VLANs and setting the priority for frames. The VLAN tag comprises two 16 bit parts. The initial part is the *Tag Protocol Identifier (TPID)*, while the second part is the *Tag Control Information (TCI)* which consists of three subfields (Figure 2.1):

Tag Protocol Identifier (TPID) The VLAN tag's first 16 bit are allocated for the *TPID*, which has a fixed HEX value of `0x8100`. In a tagged frame (i.e. a frame containing an IEEE 802.1Q tag), the *TPID* occupies the position of the EtherType field of an untagged frame and thus signifies the presence of a VLAN tag.

Priority Code Point (PCP) The *TCI* begins with a 3 bit *Priority Code Point (PCP)* field, enabling the sender to specify the frame's priority based on eight distinct priority classes. A higher PCP value usually implies a greater priority.

Drop Eligible Indicator (DEI) If the *DEI* bit is set to 1, it indicates that a switch is allowed to discard this frame in case of a high congestion.

VLAN Identifier (VID) The final 12 bit field is the *VLAN Identifier (VID)*. This field identifies the VLAN to which the frame belongs.

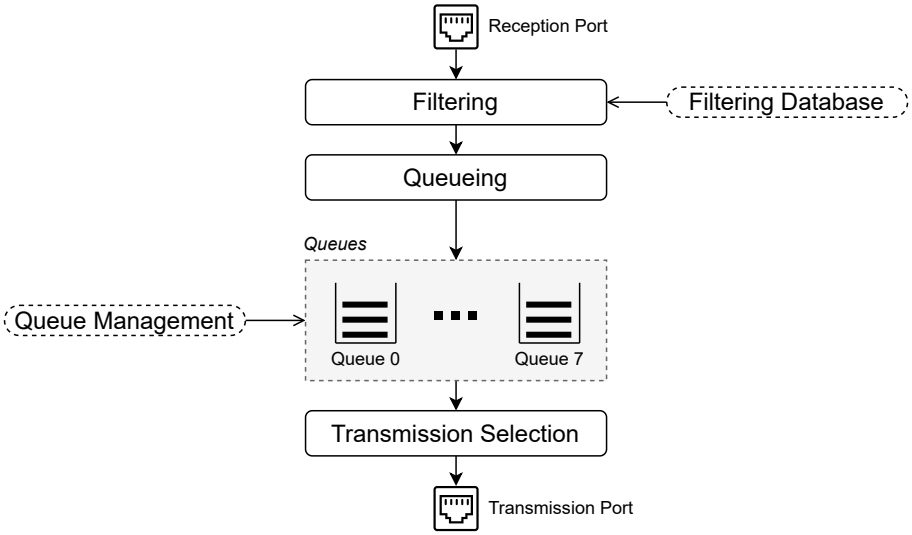


Figure 2.2: Forwarding process in a switch according to the IEEE 802.1Q standard [IEE22b].

2.1.2 Forwarding Process

In order to transmit frames to the correct receiver, Ethernet networks usually contain devices that forward data to the intended destinations (e.g. switches). Knowledge about this forwarding process is crucial to understand the mechanics of TSN in Ethernet networks. Therefore, we describe a simplified version of the forwarding process according to the IEEE 802.1Q standard [IEE22b]. Figure 2.2 illustrates this forwarding process.

The first step upon receiving a frame at the *Reception Port* is the *Filtering* step. In this step, the switch performs a forwarding decision which determines the transmission ports the frame is forwarded to. The forwarding decision is based on various parameters such as the VLAN-ID or the destination MAC address. These parameters can be modified in the *Filtering Database*. In this work, we solely rely on the destination MAC address for performing a forwarding decision. These are defined in a *MAC Forwarding Table*, which is part of the *Filtering Database*.

After performing the forwarding decision, the frame enters the *Queuing* step for every potential *Transmission Port*. This step puts the frame into a queue based on the frame's PCP value. Every egress port may have an arbitrary number of queues (also called traffic classes) between one and eight. The IEEE 802.1Q standard provides a recommended mapping of PCP values to traffic classes based on the number of available traffic classes.

Every *Queue* has its own *Queue Management* which is responsible for removing frames from the queue. This mainly happens after successful transmission of a frame. However, it may decide to also drop frames based on the queue length, the Drop Eligible Indicator (DEI) of frame or its queuing time.

The last step, namely *Transmission Selection (TS)*, selects a frame for transmission based on the priority of queues and their current state. We explain this in detail in Section 2.3 about TSN specific functionality.

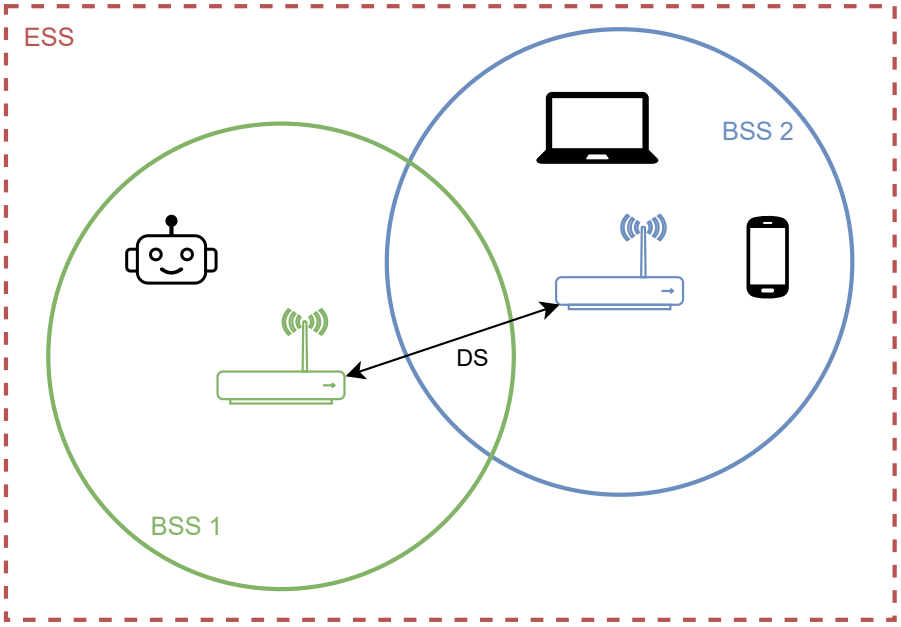


Figure 2.3: Example wireless network with multiple Access Points (APs) and Station (STA).

2.2 Wireless LAN

In this section, we discuss the basics of the IEEE 802.11 Wireless LAN standard [IEE21a]. This knowledge is crucial in order to understand the interaction between different devices in a shared wireless medium and the arising difficulties for time-sensitive applications.

The IEEE 802.11 standard allows for different operation modes. In our work, we only focus on the infrastructure mode. Figure 2.3 shows an example wireless network in infrastructure mode with two Access Points (APs) (green and blue) and different types of wireless devices called Stations (STAs). In the infrastructure mode, APs function as coordinators, to which STAs can connect. A set of connected devices is called a Basic Service Set (BSS), indicated by the colored circles around APs in Figure 2.3. Every BSS has a unique identifier, called the Basic Service Set Identifier (BSSID), which is typically the MAC address of the responsible AP. If multiple BSSs are connected via a Distrubution System (DS) (e.g. by an Ethernet network which connects the APs), they form an Extended Service Set (ESS) identified by an Extended Service Set Identifier (ESSID).

In the following, we delve into the relevant parts of IEEE 802.11 standard. We start with a description of the structure of an IEEE 802.11 frame similar to our description of an Ethernet frame. Secondly, we introduce different MAC protocols, which are responsible for controlling the access onto the shared wireless medium. The main focus of this section are the different coordination functions provided by the IEEE 802.11 standard. The next section contains a detailed description of the association process between a STA and AP. We then briefly provide some information about the physical layer of IEEE 802.11 such as frequency bands and the concept of channels in the 2.4 GHz and 5 GHz band. Lastly, we introduce a technology called Multiple Input Multiple Output (MIMO), which aims to improve the performance of wireless networks.

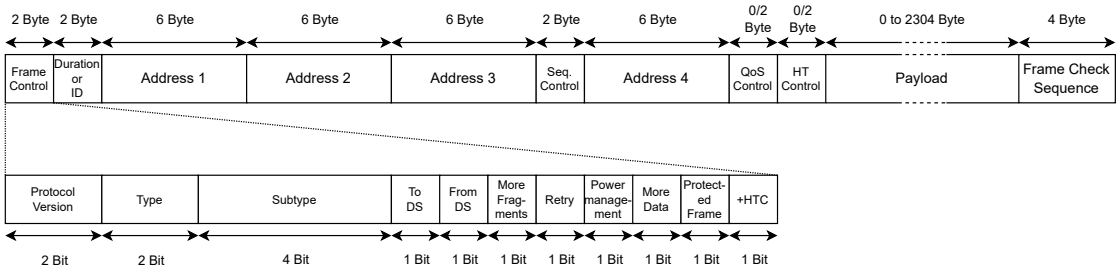


Figure 2.4: Wireless Frame according to the IEEE 802.11 standard [IEE21a].

2.2.1 WLAN Frame

In this section, we discuss the structure of a wireless frame according to the IEEE 802.11 standard [IEE21a] based on Figure 2.4:

Frame Control The *Frame Control* field contains general information about the frame, such as its type and forwarding information. It consists of the following sub-fields. The explanations below apply to *Data*-type frames:

- *Protocol Version* describes the version of the protocol and is always set to 0 in our case.
- The *Type* field distinguishes between four different frame categories: *Management*, *Control*, *Data* and *Extension* represented by 00, 01, 10 and 11 respectively. Combined with the *Subtype* field, this uniquely defines the type of the frame.
- The *From DS* and *To DS* bits are set to 1, iff the frame is sent from/to the DS respectively.
- The *More Fragments* bit is used, if a higher-level packet is divided into multiple frames. It is set to 1 on every but the last frame of this packet.
- The *Retry* bit is set, if this is the retransmission of a previously sent frame. It is used to eliminate duplicate frames.
- The *Power Management* bit is 1, iff a sending STA will be in power-saving mode.
- The *More Data* bit is set to 1 in a transmission to a STA in power-saving mode if the sending AP has more buffered data for the receiving STA.
- The *Protected Frame* bit is set iff the *Payload* is encapsulated using a cryptographic algorithm (e.g. WEP, WPA2)
- The *+HTC* bit indicates, whether the frame contains the *HT Control* field.

Duration/ID For data frames the duration field holds a duration depending on other fields of the frame. The value is:

- 0, if the address in the *Address 1* field is a group address (see below)
- the time in microseconds to transmit an ACK-frame plus Short Interframe Space (SIFS), if the the *more fragments* field is 0 and *Address 1* is an individual address
- the time in microseconds to transmit the next fragment plus 2 ACK-frames plus $3 \cdot$ SIFS, if the *more fragments* field is 1 and *Address 1* is an individual address

To DS	From DS	Address 1	Address 2	Address 3	Address 4
0	0	DA	SA	BSSID	-
0	1	DA	BSSID	SA	-
1	0	BSSID	SA	DA	-
1	1	RA	TA	DA	SA

Table 2.1: Values of the *Address 1* to *Address 4* fields based on the values in the *From DS* and *To DS* fields.

These values are calculated based on the data rate and are rounded up to the next integer microsecond.

Address 1 to 4 The values for the *Address 1* to *Address 4* fields can either be individual addresses assigned to a single STA or group addresses (e.g. for multicast or broadcast) and may be one of the following:

- *Basic Service Set Identifier (BSSID)*, which is the MAC-address of an AP in infrastructure mode
- *Destination Address (DA)*, which is the individual or group address of the final recipient(s) of the frame
- *Source Address (SA)*, which is the initiator of the frame and thus always an individual address
- *Recieving Address (RA)*, which is the individual or group address of the immediate recipient STA(s) on the Wireless Medium (WM)
- *Transmitting Address (TA)*, which is the individual address of a STA for data frames that is transmitting this frame onto the WM

The mapping of the addresses above to the *Address 1* to *Address 4* fields depend on the *From DS* and *To DS* subfields of the *Frame Control* header. Table 2.1 shows this mapping.

Sequence Control The *Sequence Control* field is a 2 B field used to manage frame reordering and deduplication. It consists of two parts: a 12-bit Sequence Number field and a 4-bit Fragment Number field. The Sequence Number field is incremented for each new frame transmitted by a sender, while the Fragment Number field is incremented for each fragment of a higher-level packet, allowing the receiver to correctly reassemble fragmented frames.

QoS Control The *QoS Control* field is a 2 B field present in frames when Quality of Service (QoS) is implemented. It manages traffic prioritization and resource allocation among different traffic classes. The QoS Control field consists of various subfields, including Traffic Identifier (TID), which indicates the priority level for the frame, and Ack Policy, which defines how the sender should receive acknowledgment for the frame transmission (see Section 2.2.2).

HT Control The *HT Control* field is a 4 B field present in frames with High Throughput (HT) or Very High Throughput (VHT) capabilities. It provides features specific to HT and VHT operations, such as support for channel bonding and MIMO (see Section 2.2.5).

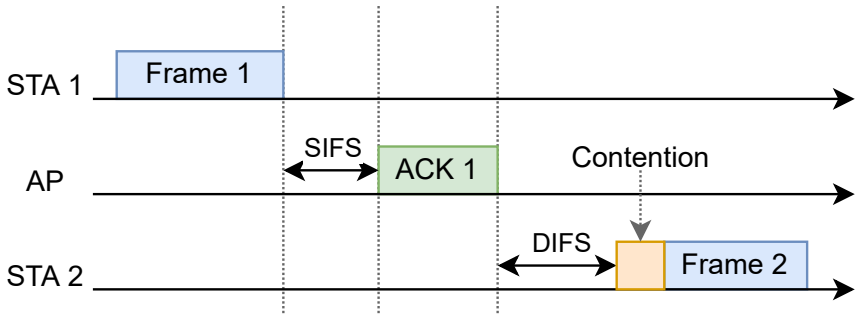


Figure 2.5: Transmission of two STAs and one AP on a shared wireless transmission medium [Rot02].

Payload The payload contains the main message by the sender. It typically has a maximum size of 1,500 B.

Frame Check Sequence The FCS is located at the end of a frame, because it uses Cyclic redundancy check (CRC), which enables the switch to calculate the checksum while receiving the frame and compare it to the received checksum in the end.

2.2.2 Media Access Control

In a wireless environment, multiple STAs and APs share the same communication medium. To prevent the interference of multiple transmissions at once, Media Access Control (MAC) protocols are required.

Figure 2.5 shows an exemplary transmission of *Frame 1* from *STA 1* to *AP*. We use this example in the following to describe the Media Access Control (MAC) protocols for IEEE 802.11 wireless networks We start with the default Distributed Coordination Function (DCF), controlling the access in a distributed manner. Thereafter, we explain the Point Coordination Function (PCF), which extends the previously described DCF. Finally, we present the Hybrid Coordination Function (HCF) which improves the two previous coordination functions by providing better a Quality of Service (QoS) support.

Distributed Coordination Function (DCF)

The Distributed Coordination Function (DCF) is the default MAC protocol used in IEEE 802.11 wireless networks. It is based on a protocol called Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA), and aims to ensure a fair sharing of the wireless medium among all STAs and APs. The DCF controls the medium access for data frames in the following way:

1. When a frame is ready for transmission, the STA checks if the transmission medium is free for a fixed duration, called the DCF Interframe Space (DIFS). Figure 2.5 shows this behavior before the transmission of *Frame 2*.
2. If the medium is free for at least DIFS, the STA initiates the transmission immediately.

3. If the medium is busy, the STA waits for the medium to become idle for DIFS and then selects a random back-off time from a Contention Window (CW). The back-off time is decremented as long as the medium remains idle (Orange box in Figure 2.5).
4. If the medium becomes busy before the back-off time expires, the STA freezes its back-off timer and resumes decrementing it once the medium is idle again for DIFS.
5. When the back-off time reaches zero, the STA initiates the transmission.

As DCF operates in a distributed manner, every STA is responsible for coordinating its own transmissions without the need for dedicated centralized control.

In the previous section, we already mentioned a difference in the *Duration* field based on the address type in the *Address 1* field. This is because transmission of broadcast and multicast frames experience a different handling than frames to an individual address. While transmissions to a group address do not require the Acknowledgement (ACK) of received frames, transmissions to an individual address require the receiving STA or AP to send an ACK to the sender of the frame. Figure 2.5 shows this behavior. After *STA 1* sends *Frame 1* to the *AP*, the *AP* waits for Short Interframe Space (SIFS) time and then responds with a specific management frame called the ACK frame. The SIFS time is shorter than the DIFS time to prioritize ACK frames over normal data frames. After *STA 1* finished sending its frame, it awaits the reception of the expected ACK frame after SIFS time. If no ACK is received, *STA 1* assumes a collision and initiates the binary exponential back-off algorithm to deal with contention. It doubles its CW (up to a maximum limit) and repeats steps 3-5 from above. The CW is reset to the initial value after a successful transmission.

Point Coordination Function (PCF)

With the DCF, the time until a STA can access the medium is hard to predict due to the probabilistic manner to deal with contention. The Point Coordination Function (PCF) is an optional extension to the DCF which provides a centralized control mechanism for coordinating the access to the wireless medium. This is achieved by utilizing a polling mechanism, in which an AP with point coordinator functionality periodically polls the associated STAs to grant them the opportunity to transmit data.

The operation of the PCF can be summarized as follows:

1. If the point coordinator within a AP wants to initiate a Contention Free Period (CFP) it has to wait a specific duration. This duration is called the PCF Interframe Space (PIFS) and lies in between the SIFS and DIFS time. Thus it does not interrupt ongoing frame sequences waiting for SIFS but gives priority over other STAs by preventing them to start a new transmission after DIFS. During the CFP only the PCF and polled STAs are allowed to access the medium.
2. The AP sends a beacon frame (*StartCF*) at the beginning of the CFP, which contains the CFP duration and a list of associated STAs to be polled. The duration is stored in the Network Allocation Vector (NAV) of every STAs, thus preventing it from sending during the CFP
3. After the beacon frame is sent, the AP starts polling the STAs in the list by sending a *Poll*-frame to each STA.

4. Upon receiving a *Poll*-frame, the polled STA can transmit its data immediately without the need for carrier sensing or back-off procedures. If a STA does not have any frames to transmit, it has to send a *null*-frame.
5. The AP acknowledges the successful reception of the data frame from the polled STA by sending an ACK frame. It may also include the next *Poll*-frame for the next STA in the same frame (piggybacking).
6. The CFP ends, when the specified duration elapses or when the point coordinator has polled all STAs and transmits a *EndCF*-frame. The medium then returns to the contention-based access mode, and the DCF resumes operation.

The PCF provides several advantages over the DCF, including deterministic access to the medium, reduced collisions, and improved QoS for time-sensitive applications. However, it also introduces additional overhead due to the polling and requires the implementation of a centralized coordinator, which may not always be desirable in certain network scenarios.

Hybrid Coordination Function (HCF)

The Hybrid Coordination Function (HCF) is an extension to the IEEE 802.11 MAC protocols, aiming to improve upon the previously explained DCF and PCF by providing better QoS support. HCF introduces two new coordination mechanisms called the Enhanced Distributed Channel Access (EDCA) and HCF Controlled Channel Access (HCCA).

Enhanced Distributed Channel Access (EDCA) is a prioritized version of the DCF, which allows differentiating between traffic types based on their QoS requirements. This differentiation works by classifying frames into different Access Categorys (ACs). Each AC is associated with a set of contention parameters, which influences its priority level in accessing the medium.

By default, EDCA utilizes four different ACs: *AC_VO*, *AC_VI*, *AC_BE* and *AC_BK* for Voice, Video, Best effort and Background traffic, respectively. Each AC has its own Arbitration Interframe Space (AIFS), with shorter AIFS values for higher priority traffic. Consequently, this results in reduced latency and jitter for time-sensitive applications. Additionally, EDCA employs AC-specific CW sizes and back-off timers. When a collision occurs, the CW is adjusted based on the AC, ensuring that high-priority traffic is less affected by collisions compared to lower-priority traffic.

The **HCF Controlled Channel Access (HCCA)** mechanism is based on the polling-based PCF and operates alongside the EDCA. In HCCA, a Hybrid Coordinator (HC) within an AP schedules and manages the transmission of data frames for each STA. The HCCA mechanism requires that stations submit Traffic Specification (TSPEC) requests to the HC, which contains information about their QoS requirements, such as data rate, delay, and jitter. Based on the TSPEC requests, the HC calculates the amount of resources required and grants or denies the requests. Once granted, the HC schedules a Transmission Opportunitys (TXOPs) for each station, ensuring that the QoS requirements are met.

2.2.3 Association Process

Before STAs and AP are able to exchange data, a STA first needs to associate with an AP. This process contains three phases, the *Discovery*, *Authentication* and *Association* phase respectively. In the following, we describe these three phases in detail.

Discovery Phase

The discovery process is used by STA to identify and select an appropriate AP to establish a connection. The discovery process in IEEE 802.11 networks allow for two different methods of AP discovery:

Passive Scanning In this method, the STA listens to the wireless medium for Beacon frames broadcasted periodically by APs. The *Beacon* frames contain information such as the BSSID of the AP, supported data rates, and other capabilities. By collecting and evaluating the received Beacon frames, the STA can identify nearby APs and select one to connect based on signal strength, supported features, or other metrics.

Active Scanning In active scanning, the STA actively participates in the discovery process by transmitting *Probe Request* frames. These frames may include the desired BSSID or be left blank to discover all available AP. Upon receiving a *Probe Request* frame, APs with a matching BSSID respond with a *Probe Response* frame, containing similar information to the Beacon frames. The STA evaluates the received *Probe Response* frames to select a suitable AP for connection.

Authentication Phase

After discovering and selecting an appropriate AP, the STA initiates the authentication process. Authentication establishes the identity of the STA and AP, and possibly ensuring secure communication. The IEEE 802.11 standard distinguish between *Open System Authentication* and *Shared Key Authentication*.

Open System Authentication authentication is solely based on the BSSID. The STA sends an *Authentication Request* frame to the AP, and if the BSSID matches, the AP responds with an *Authentication Response* frame, indicating success. This authentication method does not provide any further security measures, such as encryption.

Shared Key Authentication (mostly *Wired Equivalent Privacy (WEP)*) uses a shared secret key to authenticate the STA. The STA sends an authentication request with a challenge text to the AP. The AP encrypts the challenge text with the shared secret key and sends it back to the STA. The STA decrypts the message and sends it back to the STA. If the decrypted message matches the original challenge, the AP authenticates the STA.

Extensions to the IEEE 802.11 standards might introduce different authentication methods, such as *Wi-Fi Protected Access (WPA)* [IEE04], *Fast Initial Link Setup (FILS)* or *Extensible Authentication Protocol (EAP)*, but as this work does not focus on security, we do not further elaborate on these authentication methods.

Association Phase

Once authenticated, the STA needs to establish a connection with the AP through the association process. The association process involves the exchange of *Association Request* and *Association Response* frames between the STA and AP.

In the *Association Request* frame, the STA transmits information about its supported data rates, QoS capabilities, and other relevant features to the AP. Upon receiving the *Association Request* frame, the AP evaluates the AP's capabilities and decides whether to accept or deny the connection. If accepted, the AP assigns an Association Identifier (AID) to the STA and sends an *Association Response* frame, containing the AID, supported data rates, and other necessary information for the connection.

Once the association process is completed, the STA and AP are able to communicate using the established connection.

2.2.4 Frequency Bands and Channels

The IEEE 802.11 wireless technology operates in different frequency bands to facilitate communication between STAs and APs. The primary frequency bands used for IEEE 802.11 communications are the 2.4 GHz and 5 GHz bands, with newer standards also utilizing the 6 GHz band. The IEEE 802.11 standard employs Frequency Division Multiple Access (FDMA) and distributed the bands into multiple channels, which reduce the interference of neighboring channels. In this section, we discuss channel allocation and width for these frequency bands.

2.4 GHz Frequency Band

The 2.4 GHz frequency band is used by IEEE 802.11b, 802.11g, and 802.11n standards. This band has a range of 2.401 GHz to 2.495 GHz, divided into 14 overlapping channels, each with a width of 22 MHz and a spacing of 5 MHz. Channel 14 provides an exception, as it is 12 MHz instead of 5 MHz away from channel 13. As the use of channel 14 is forbidden in most countries, the 22 MHz channel width and 5 MHz channel spacing, only allow three non-overlapping channels (1, 6, and 11) which be used simultaneously without causing interference.

5 GHz Frequency Band

The 5 GHz frequency band is utilized by the 802.11a, 802.11n, 802.11ac, and 802.11ax standards. This band is divided into multiple non-overlapping channels, each having a width of 20 MHz and spacing of 10 MHz. The 5 GHz band offers more available channels than the 2.4 GHz band, with fewer chances of interference between neighboring channels. The exact amount is subject to regulatory restrictions in different countries and intended use-case (e.g. indoor/outdoor). The 5 GHz also provides support for larger channel widths (i.e. 40 MHz, 80 MHz and 160 MHz), which are used to achieve higher data rates. However, larger channel widths also increase the potential for interference and reduce the number of available non-overlapping channels.

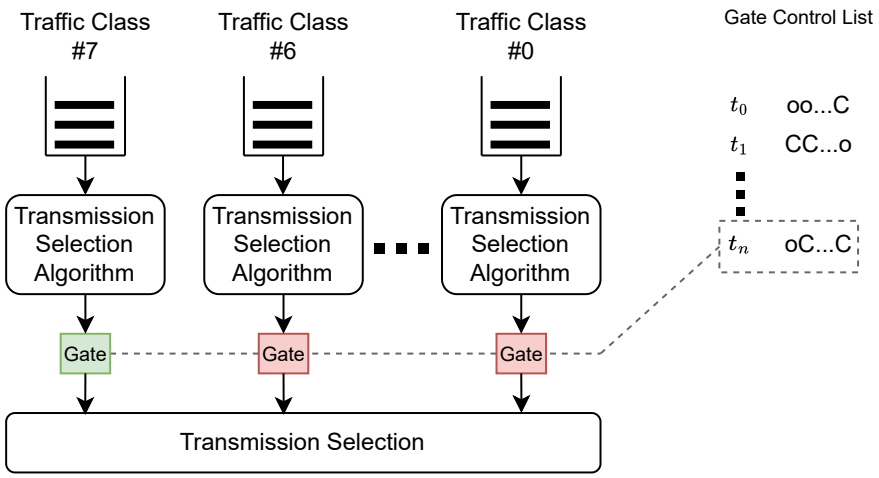


Figure 2.6: Time-Aware Shaping, Gates, Gate Control List and Transmission Selection.

2.2.5 Multiple Input Multiple Output (MIMO)

In order to increase the achievable datarates for wireless transmission, the IEEE 802.11n standard introduced a technology called Multiple Input Multiple Output (MIMO). MIMO employs multiple antennas at both the transmitter and receiver to improve communication performance. By leveraging spatial diversity and spatial multiplexing, MIMO systems can achieve higher data rates, increased coverage, and improved reliability compared to single-antenna systems.

In spatial diversity, multiple antennas are used to transmit the same data over different paths, providing redundancy and improving the likelihood of successful reception. Spatial multiplexing, on the other hand, allows for the transmission of multiple data streams simultaneously over the same frequency channel, effectively increasing the throughput.

Multi-User MIMO (MU-MIMO) is an extension of MIMO technology that enables multiple STAs to be served simultaneously by a single AP. This feature further improves the overall network capacity and efficiency, especially in high-density environments.

2.3 Time-Sensitive Networking (TSN)

The Time-Sensitive Networking (TSN) standards are a set of amendments extending already existing Ethernet standards as described in Section 2.1 with functionalities for deterministic real-time traffic. They were developed by the *TSN Task Group* [IEEb] which is part of *IEEE 802.1 working group* [IEEa] with the primary purpose of enabling TSN to deliver deterministic, low-latency, and highly reliable communication across Ethernet networks. The most important amendments for this work are the IEEE 802.1AS amendment [IEE20a] providing timing and time synchronization functionality and the IEEE 802.1Qbv amendment [IEE16a] with “Enhancements for Scheduled Traffic”. For our work, the most important TSN functionalities are the Transmission Selection Algorithm (TSA), gates, the Gate Control List (GCL) and the Transmission Selection (TS), which we describe in the following.

Figure 2.6 shows, how the TSAs, gates and the TS are connected to each other. They extend the approach of the traffic classes as described in Section 2.1.2.

The Transmission Selection Algorithm (TSA) is responsible for announcing, whether its respective queue has a frame eligible for transmission. A TSA is able to make the announcement decision based on various parameters, such as the number of frames in a queue or the maximum available bandwidth for its respective queue. The IEEE 802.1Qbv amendment allows for vendor-specific TSAs, but also already defines possible STAs, for example:

Strict Priority The *Strict Priority* TSA is the default TSA. When the queue is not empty, it always announces a frame ready for transmission.

Credit-Based Shaper (CBS) The CBS uses credits in order to announce if a frame is ready for transmission. Every transmission reduces the amount of available credits, while an idle queue accumulates new credits. This approach ensures a fair distribution of network resources.

Following the TSA of every queue, there is a gate, which can be controlled by a so-called Gate Control List (GCL). Every port has one GCL, which controls all gates of the corresponding port. This GCL contains entries, which specify the states of the respective gates and a time interval for how long this entry is active. Upon the expiration of one time interval, the next entry in the GCL becomes active. After a configurable cycle time, the entries in the GCL repeat. The state of a gate as defined by the GCL can be either *open* or *closed*. When the gate is in the *open* state, frames of the respective queue are eligible for transmission. In the *closed* state, however, no frames of the respective queue may be transmitted.

By utilizing a GCL, we can specify, when a queue is allowed to transmit frames based on the current time. This is a Time Division Multiple Access (TDMA) based approach and called Time-Aware Shaping (TAS). We later use this approach to employ calculated schedules for time-sensitive streams.

The last step before a transmission is the TS. It is responsible for selecting a frame for transmission at the transmission port based on the gate states and the announcement of the TAS. To this end, it only considers queue with an open gate state. Among these queues, it checks the TSA of every queue for an announced frame. Finally, it selects the queue of highest priority with an announced frame and an open gate state.

2.4 Integer Linear Programming (ILP)

In order to utilize GCLs and reserve bandwidth for time-sensitive streams, we need to pre-calculate schedules and routes. Calculating these routes and schedules for given streams and network is NP-hard [Ull75]. However, the scheduling and Joint Routing and Scheduling (JRaS) problem can be represented using formal descriptions such as Integer Linear Programs (ILPs) or Constraint Satisfaction Problems (CSPs), allowing optimized solvers to solve the problem.

In this work we rely on a well-researched method for calculating routes and schedules by utilizing *Integer Linear Programming (ILP)*. ILPs allow us to represent our network topology and streamset as variables and constraints. In detail, these are the components of an ILP [Sch86]:

Variables hold a numeric values and are the part the solver tries to find a valid value for. In an ILP, only integers are valid values.

Constraints consist of three parts, a left-hand side (lhs), a sense and a right-hand side (rhs). lhs and rhs are linear expressions (e.g. $2a$ or $3b - a$) and the sense is either $<$, $\leq$, $=$, $\geq$ or $>$. These three parts allow us to express relations between different variables, by providing a constraint, either as a linear equation (e.g. $2a = b$) or a linear inequality (e.g. $3b - a \leq 2c$).

Bounds allow us to restrict the value range of a variable, e.g. $a \in \{x \in \mathbb{Z} \mid -100 < x < 100\}$.

Objective An objective provides us the possibility to specify an optimization goal This allows the ILP-solver algorithm to determine the optimal solution with respect to that objective. For instance, the objective $\max a + b$ is employed to identify the solution featuring the largest valid value for $a + b$. Certain ILP-solvers provide the option to provide multiple objectives accompanied by their respective weights. Nonetheless, this essentially constitutes a linear combination of the objectives and is mathematically analogous to having a single, interpolated objective.

In some cases it is useful to use conditional constraints, which are either activated or deactivated dynamically depending on the value of a variable. One possibility to represent these as linear constraints, is the usage of so called *Big M* constraints [SSZ15]. These consist of the following parts:

- A binary decision variable $b \in \{x \in \mathbb{Z} \mid 0 \leq x \leq 1\}$, which is 0, when a constraint should be deactivated and 1, if it should be activated.
- A constant M bigger than any feasible value of other variables involved.

With these two parts a conditional constraint such as

$$\textbf{if } (b = 1) \textbf{ then } x < y$$

is represented as

$$x < y + (1 - b) \cdot M$$

With this representation, if b is 1 the constraint evaluates to $x < y$. But with b being 0, the constraint evaluates to $x < y + M$, allowing x to take any value due to the size of M and thus essentially disabling the constraint.

Due to the previously mentioned NP-hardness of ILPs, specialized tools are required to solve complex ILPs with a high number of variables and constraints in a reasonable amount of time. Often, these ILP-solvers provide additional functionality, such as providing hints for variable values or influencing the algorithm by tweaking parameters. As none of the approaches in this work require solver-specific functionality, our concepts work with all ILP-solvers following the aforementioned format of an ILP. In this work, we utilize the established and well-known ILP-solver *Gurobi* [Gur].

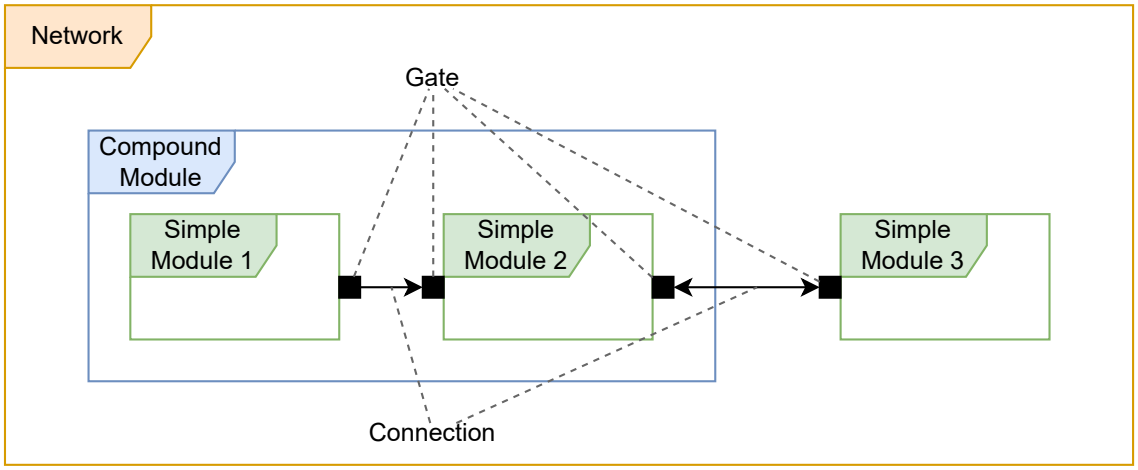


Figure 2.7: Model structure of the OMNeT++ simulation framework [VH08].

2.5 OMNeT++

To validate the effectiveness of our approaches and evaluate their performance, we either need a real-world testbed or a simulation tool that allows us to simulate a scenario with simulation models of real-world devices. In this work, we utilize the discrete event simulator OMNeT++ [Var01] and the INET framework [MVK19] extending OMNeT++ with different functionality to simulate wired and wireless networks. In the following, we describe both parts in detail.

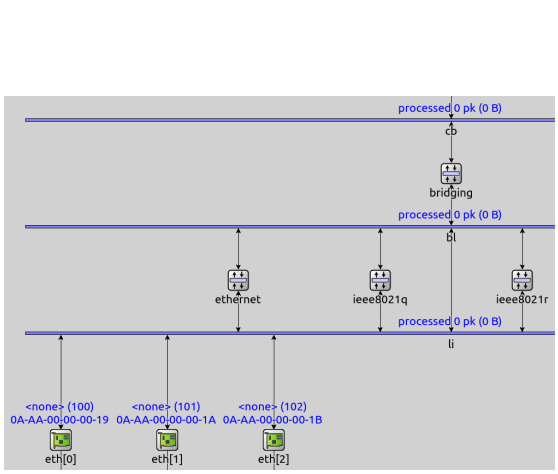
OMNeT++ is a generic simulator which supports any kind of networks with discrete events. This for example includes road-networks, distributed hardware systems, Ethernet and wireless network environments. It provides its own development environment based on the Eclipse IDE [Ecl] and allows simulation with a graphical user interface or headless simulations with output files for further usage and evaluation. The simulation framework follows a hierarchical and modular approach for modeling the desired networks, which allows for reusable components. All components of a simulation are described using the *NED language* [VH08]. Figure 2.7 shows an example simulation model with the different components of a simulation model:

Simple Module A simple module is the part of a simulation, where all active behavior is defined. It is linked to a C++ class, which describes the behavior of this module. Additionally, parameters (e.g. the IP address of a device) further specify the attributes of a module.

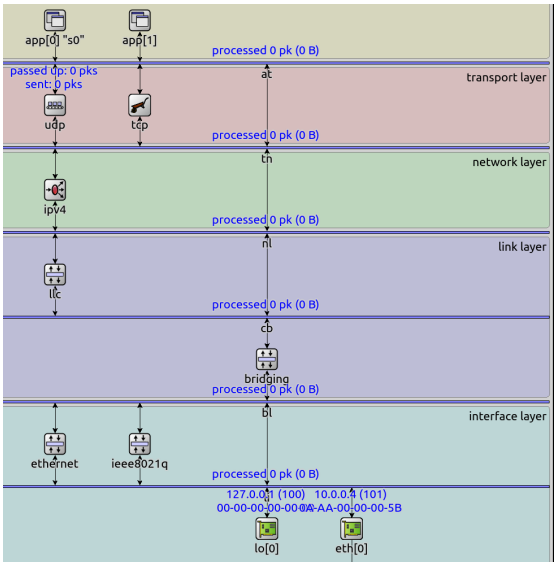
Compound Module A compound module does not define active behavior itself. Instead, a compound module consists of one or more simple and/or compound modules which interact together in order to specify the behavior of the compound module (e.g. an antenna and a sending application forming a wireless node).

Gate Every simple or compound module is able to define gates (e.g. an Ethernet port), these are endpoints to send and receive messages (e.g. an Ethernet frame) from other modules.

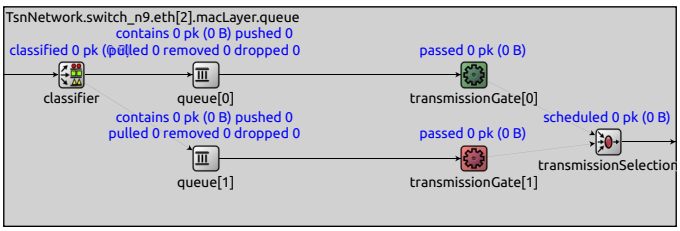
Connection A connection connects two different gates. These connections may be unidirectional or bidirectional and may have additional parameters (e.g. the bitrate of an Ethernet connection).



(a) Switch in INET implementing TSN functionality.



(b) Host in INET implementing TSN functionality.



(c) Implementation of a TAS with two queues.

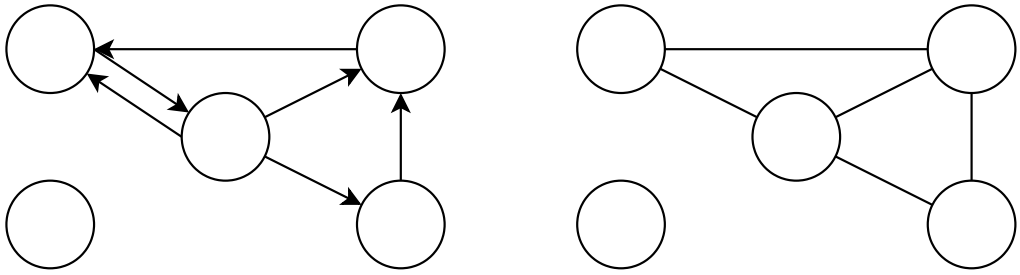
Figure 2.8: Different modules in INET implementing the TSN standards.

Network The network is the root of a simulation. It contains all modules and the connections between those.

The parameters of every component in the model is either defined in the *NED* files of the network and modules, or it is defined in a specific configuration file. This allows to run a simulation with the same devices and network structure, but with different parameters.

The INET framework [MVK19] builds upon OMNeT++ providing functionality for different communication networks. This includes models for the Internet stack (e.g. UDP, IPv4) and link layer protocols (such as Ethernet and IEEE 802.11 Wireless LAN). The implementation of modules in the INET framework follows the OSI layer architecture, allowing to specify the behavior, simulate and evaluate on different layers. Furthermore, INET provides models to simulate wireless environments close to real-world scenarios such as interference, the movement of devices and error rates. Additionally, it has predefined compound modules representing real-world network devices including switches, APs and wireless and wired hosts.

For Ethernet-devices it also has modules implementing the TSN standards as explained in Section 2.3. Figures 2.8a and 2.8b show the modules of a switch and host devices with TSN functionality respectively. Both devices implement a *StreamIdentifier* in the *bridging* module which allows to map different parameters (e.g. destination port and MAC-address) to an internal Stream-ID. The following *StreamEncoder* allows to define values such as the PCP or VLAN-ID based on



(a) Directed graph. **(b)** Undirected graph.

Figure 2.9: Comparison of a directed and undirected graph.

the Stream-ID of the previous step. After that, the *ieee8021q* module inserts the *802.1Q Tag* as described in Section 2.1.1. In our work, the most relevant module is the Time-Aware Shaping (TAS). This functionality is located in a *Ieee8021qTimeAwareShaper* within the *macLayer* in every Ethernet interface (e.g. *eth[0]* in Figure 2.8). Figure 2.8c shows INET's implementation of the TAS. It contains a configurable number of queues (one to eight). Its *classifier* checks the PCP of a frame and puts it into the respective queue. Following that, every TAS has a *transmissionGate*, which is configurable by a GCL. Finally, the *transmissionSelection* select a frame from one of the open queues. This is a *PriorityScheduler* by default which always selects the first non-empty queue based on their priority.

2.6 Graphs & Graph Algorithms

In order to model networks and the connection of network devices, we need a formal way to describe these environments. A graph is a common way to model a set of devices and connections. It also allows us to execute graph algorithms, such as calculating routes between devices. We start this section, by formally defining the different components of a graph, such as vertices and edges. Thereafter, we introduce different graph algorithms, which we utilize later in this work, namely the vertex coloring algorithm.

2.6.1 Graph Definition

A graph always consists of a set of vertices $\mathcal{V}$. One possibility to represent a vertex is to represent it as a node. Figure 2.9 shows an example with five vertices.

We model connections between vertices by a set of edges $\mathcal{E}$. Each edge is an ordered or unordered pair of vertices (e.g. (v_1, v_2) or $\{v_1, v_2\}$). There are two different basic types of graphs, directed and undirected graphs (Figure 2.9b). In a directed graph, each edge is an ordered pair of vertices, also known as 2-tuples, whereas in an undirected graph they are modeled as unordered pairs, also known as 2-sets. For example, in a directed graph $\mathcal{E} = \{(v_1, v_2)\}$ means v_1 is connected to v_2, but v_2 is not connected to v_1. In contrast, undirected graphs use unordered pairs, so $\{v_1, v_2\}$ is equivalent to $\{v_2, v_1\}$. This means that if v_1 is connected to v_2, v_2 is also connected to v_1. Directed edges are represented by arrows connecting nodes (Figure 2.9a), whereas undirected edges are represented as lines (Figure 2.9b).

Algorithm 2.1 Greedy Vertex Coloring Algorithm

```
 1: procedure VertexColoring(g: Graph)
 2:     colors: Dict[Vertex, Number] ← {}
 3:     for each current in g.vertices() do
 4:         neighboringColors: Set[Number] ← {}
 5:         for each neighbor in g.neighborsOf(current) do
 6:             if neighbor in colors then
 7:                 neighborColors.add(colors[neighbor])
 8:             end if
 9:         end for
10:         color ← 0
11:         while color in neighborColors do
12:             color ← color + 1
13:         end while
14:         colors[current] ← color
15:     end for
16:     return colors
17: end procedure
```

2.6.2 Vertex Coloring

The *Vertex Coloring* problem [Kub04] involves assigning colors to the vertices of a graph in such a way that no two adjacent vertices share the same color. Finding an optimal vertex coloring (i.e. one with the least amount of colors) is NP-hard [Kub04]. Thus, in scenarios which do not require optimal solutions, one can use non-optimal algorithms. Depending on the use-case one may use simple algorithms or more sophisticated, heuristic-based algorithms producing better results.

In Algorithm 2.1 we present a simple greedy graph coloring algorithms [Hus15]. This algorithm assigns colors based on numbers increasing from 0. It iterates over all vertices in the graph in arbitrary order. For each of these vertices, it assigns the lowest possible color (i.e. number) which is not yet used by any neighboring vertex. This ensures that no neighboring vertices have the same color.

3 Related Work

In this chapter, we provide an overview over relevant research for providing deterministic real-time guarantees in wireless networks with handovers. We first take a look at research that analyses the current state, challenges and possible solutions for adapting TSN for operation in wireless network environments. Following that, we outline different approaches aiming to provide a reliable and smooth handover in wireless networks in order to provide real-time guarantees during handovers. Finally, we present already existing solutions for the scheduling and Joint Routing and Scheduling (JRaS) problem in wired Ethernet networks, which we later use to build upon and extend for wireless environments.

3.1 Wireless TSN

In this work, one of our goals is to research challenges of time-critical traffic in wireless network environments and possible solutions. While time-sensitive networking was already implemented in the IEEE 802.1Q standard in 2018 [IEE22b], the research of time-critical traffic in wireless environments is under active research right now. In this section, we provide an insight in recent research in this field.

On of these researches is the current development of a new amendment IEEE 802.11be (i.e. Wi-Fi 7) to the IEEE 802.11 standard [IEE21a]. The amendment aims to improve upon the previous Wi-Fi 6 IEEE 802.11ax [IEE21b] in multiple ways [KLA20]. This includes a higher throughput of up to 40 Gbps by utilizing more spatial streams, allowing for higher bandwidth channels and the introduction of new coding schemes. Additionally, it aims to include the TSN standards as described in Section 2.3.

Adame et al. [ACB19] take a closer look at these key-features of IEEE 802.11be and the challenges and possible solutions on implementing TSN functionality within the new standard. The main points they focus on in their research regarding TSN are time synchronization, traffic prioritization, frame preemption, admission control and schedule operation. Firstly, they point out that time synchronization is a necessary requirement for providing TSN support. They suggest utilizing the existing IEEE 802.1AS standard [IEE20a] for timing and synchronization, as it already supports operation in IEEE 802.11 wireless networks. Haxhibeqiri et al. [HJA+21] elaborate further on time synchronization in wireless networks utilizing the Precision Time Protocol (PTP) synchronization protocol. In their research they analyze the influence of different parameters such as the beacon interval and network load. The absolute synchronization error in their research averages between 10 µs and 21 µs depending on these parameters.

The second point of Adame et al.'s research focuses on traffic prioritization, especially the four ACs provided by the EDCA as explained in Section 2.2.2. They argue that the existing ACs are insufficient for time-sensitive applications, as they do not provide hard jitter and latency bounds.

Their suggestion is to add another AC providing these bounds and the implementation of a TAS before the queues. Furthermore, congestion on the wireless-medium by an ongoing transmission of maximum size may take up to 5 ms and thus delay time-sensitive frames upon their latency bound. To mitigate this, they suggest implementing the IEEE 802.1Qbu [IEE16b] frame preemption mechanism, but also note that this imposes several changes to the current physical and link layer specifications of the IEEE 802.11 standard. Finally, Adame et al. mention that scheduled access is crucial for a low latency and discuss challenges such as Overlapping Basic Service Sets (OBSSs) and possibilities to enable scheduled access in a collision-free manner. For example, by utilizing a functionality called Target Wake-up Time (TWT) that wakes up STAs on a periodic basis to transmit and receive data or restricting the 6 GHz band to devices supporting the necessary features.

Another research by Cavalcanti et al. [CPR+19] analyzes current industrial state-of-the-art wireless environments besides IEEE 802.11 Wi-Fi, such as Bluetooth, cellular technologies and the well-established IEEE 802.15.4 standard [IEE20b] in Internet of Things (IoT) networks. In their analysis of challenges for establishing time-sensitive application in wireless environments, they compare the radio and channel models of different wireless standards regarding their modeling of path loss, as understanding the variations in a wireless transmission medium is crucial to enable TSN in these environments. Furthermore, they point out that the back-off procedure in wireless environments is the main source of delay and the randomness by these protocols make it hard to employ real-time applications with deterministic bounds to latency and jitter. In their work, they present two approaches to mitigate this problem. One approach is to utilize different multiple access techniques, such as TDMA or FDMA as explained in Chapter 2. Their second approach to remove the problem of uncontrolled delay due to the random back-off is a centralized coordination and scheduling. They note that this approach is only suitable with full control over the wireless environment such as in a factory. However, they also mention that unlicensed frequency bands may be subject to interference from other devices, such as phones utilizing Wi-Fi and Bluetooth, but also from other electronic devices such as microwave ovens. Due to these unforeseeable sources of interference, a constant monitoring of the wireless environment is indispensable. Finally, they provide multiple approaches aiming to improve the reliability of wireless links, e.g. Beamforming, a controlled transmit power for every transmission and redundant paths among others.

Fang et al. [FSC+21] analyze the aforementioned path redundancy in more detail. The main goal of using redundant paths is to mitigate the effect of unexpected interference and thus provide more reliability for time-sensitive applications. In their research, they utilize the IEEE 802.1CB standard [IEE17] for frame replication and elimination for reliability. Their approach is to connect a wireless device to two APs at once and send the time-critical frames over both paths redundantly. They evaluate this approach in a real-world environment with a wired sender, two APs and a wireless receiver for an environment with and without additionally generated interference on one channel. In the first scenario without additional interference, their findings show a difference in packet-loss by up to 4 %. In the second scenario however, the reliability increases from between 47 % and 75 % (depending on the latency bound) on the channel with generated interference to over 99 %.

Sudhakaran et al. [SMBC21] research the aforementioned challenges in an experimental setup using two wirelessly connected robots. These robots aim to collaborate on keeping a rod straight and unbent by exchanging data over the wireless network. One of these robots is the leader and sends relevant data about speed and direction as well as perception data in form of an image to the other robot. These two data transmission operate on different priorities in order to analyze the influence of streams of two different priorities. They analyze the results based on two different scenarios:

One with and one without a wireless TSN implementation. Their proposed wireless TSN solution includes time-synchronization and Time-Aware Shaping (TAS). In the scenario without the wireless TSN implementation, positional errors accumulate over time due to high delays introduced by the transmitted images and the robots fall out of sync and fail to keep the rod straight. When enabling the wireless TSN functionality, especially TAS, the robots are able to stay in sync and perform the task successfully.

3.2 Handover

In addition to the aforementioned challenges, covering bigger areas for mobile STAs requires multiple APs and handovers of the STAs devices between these APs. This section covers challenges and already existing approaches for smooth-handovers in order to provide the necessary real-time guarantees for TSN.

According to existing research, a normal handover in IEEE 802.11 Wi-Fi networks is a time-consuming process. The whole handover procedure takes around 2.8 s [FS17] and thus a drop in connection of this time period. Balažia et al. note that especially the scan phase as described in Section 2.2.3 is responsible for up to 90 % of the total handover time and when using authentication servers, the delay of the authentication phase can add up to 500 ms [AH08]. In order to mitigate the effects of a long handover, Feier et al. evaluate different approaches to provide smooth handovers. They first compare different possibilities or already existing research to reduce the time spent in the scan phase. For example by starting the scan on a specific channel shortly before an expected Beacon frame and by reducing the number of scanned channels, if the channels of neighboring APs are known. Additionally, they compare approaches eliminating the scan-phase partly or entirely by gathering information about other neighboring APs form their current AP or other STAs. They also mention approaches including a centralized roaming management, but these either lack compatibility with the IEEE 802.11 standard or require triggering of the handover by upper-layers of the STA. Finally, they propose a proactive handover concept based on the Received Signal Strength Indicator (RSSI) from neighboring APs. With this concept, they are able to reduce the association time from around 2.8 s for the first association to an average of 91 ms for subsequent handover associations.

According to Dutta et al., signal-based based metrics (e.g. RSSI, Signal-to-noise Ratio (SNR)) are unreliable metrics for proactive handovers, as they vary significantly because due to different fading phenomena [DCT+07]. To this end, they propose a location-based proactive handover method and compare it to a proactive handover approach utilizing SNR. Their findings show that the handover performance of the two approaches are similar. However, they note that in wireless environments with high-mobility SNR-based approaches may generate a ping-pong effects, which can be prevented by location-based handovers. Thus it is desirable to use a location-based or mixed handover approach in these environments.

Most existing handover approaches are either capable of providing a smooth handover or a high network performance [ZZW16]. A novel approach by Zubow et al. [ZZW16] aims to achieve by utilizing the Dynamic Frequency Selection (DFS) functionality of the IEEE 802.11 standard. DFS allows an AP to initiate a channel switch by sending a *Channel Switch Announcement* to its associated STAs. Normally, DFS is used to move to a different channel if radar signals are detected on the current channel, but Zubow et al. take advantage of it in order to enable smooth handovers. In their approach, they assign neighboring APs to different DFS-supporting channels.

In contrast to typical IEEE 802.11 environments, they assign the same BSSID (i.e. MAC-address) to all APs. Hence, a *Channel Switch Announcement* by an AP forces the STA to switch to that channel and continue the transmission with the AP on the new channel. From the STA view, only a channel switch occurred, as the BSSID of the connected AP did not change. In order for this to work, association information needs to be transferred from the old AP to the new AP. To this end, they introduce a centralized coordinator, which is also responsible for making handover decisions. Their results show that they are able to reduce the handover time from 4.26 s for a typical handover to 0.13 s with their novel approach. As the handover time in their approach is mainly limited by the channel switching time, which will decrease to 2 ms in the future according to manufacturers [ZZW16].

3.3 Scheduling

In order to provide deterministic real-time guarantees, it is crucial to reserve bandwidth and time-slots for time critical-traffic. Calculating these time-slots is a so-called *Scheduling* problem. Scheduling problems arise in a lot of different scenarios such as networks, multiprocessing environments and factory production lines. As all of these scheduling scenarios share a lot of similarities and can often be mapped to each other, research in one of these scenarios is also relevant for the others. Scheduling-only approaches calculated schedules along a predefined path. In network environments however, there are often multiple possible routes. *JRaS* approaches cover these scenarios, by calculating routes and schedules together. In this section, we cover different already existing JRaS and scheduling-only approaches.

One of the earliest research on scheduling problems is the Job Shop Scheduling Problem (JSSP) by Graham [Gra66]. A JSSP contains of a series of operations called jobs, and a number of machines, which are able to execute these operations. The goal of the JSSP is to calculate time-lots for these operations on machines in such a way that the finishing time of the jobs is minimized. Dürr et al. propose an approach, which utilizes the JSSP to calculate schedules for IEEE 802.1 Ethernet-based TSN networks [DN16]. To this end, they map the no-wait scheduling problem for Ethernet packets to a no-wait JSSP. In a second step, they provide an Integer Linear Programming (ILP) for the JSSP and modify it in order to support the packet scheduling problem for Ethernet networks. In order to increase the scheduling capabilities of their scheduler, they also provide a heuristic tabu-search approach, which is able to schedule around 1500 instead of 50 streams for the complete ILP-based approach with a negligible quality difference of the resulting schedules. To further optimize the number of gate-opening events, they post-process the resulting schedules and delay specific frames. This allows for a reduction of gate-opening events by about 24 %.

There are other heuristic-based approaches for TSN. Glavackij [Gla20] presents a tracing-based approach. To generate a schedule, he first sends packet of the time-critical streams in the network without pre-defined schedules and generates an initial schedule based on the packet traces. He then modifies the initial schedule using various heuristics, e.g. based on Monte Carlo Tree Search (MCTS).

Heuristic approaches are often incapable of finding feasible schedules in networks with high-utilization. However, in low to medium-utilized networks, they are superior regarding their runtime. An approach by Pop et al. [PRGS18] takes advantage of this, by providing an algorithm which is capable of adding streams on demand. Their heuristic algorithm first tries to fit the new stream into the existing schedule and only if this fails, it recalculates a new schedule.

In addition to the IEEE 802.1 TSN Ethernet protocol, there exist several other Ethernet-based real-time communication standards, such as TTEthernet [SAE16] and PROFINET [PRO14]. Steiner [Ste10] and Cracunias et al. [CO14] both research the scheduling problem for TTEthernet. In their research, they provide formal descriptions for this scheduling problem using Satisfiability Modulo Theory (SMT) and provide different optimization possibilities to reduce the runtime of their scheduling approach by introducing on-demand constraints. Cracunias et al. [CO15] also compare their SMT-based scheduling approach to a new ILP-based scheduling approach. Their results show that in most of the cases, their approach with on-demand SMT constraint performs better. In a later work by Steiner and Cracunias et al., they modify their previous SMT-based approaches for TTEthernet to support IEEE TSN schedules.

The approaches mentioned above are scheduling-only approaches scheduling along pre-defined or pre-calculated routes. In big networks with a high utilization, these approaches may be incapable of finding a feasible schedule. JRaS approaches are more suitable in these cases, as they provide a greater possible solution space. As the JRaS approach extends upon scheduling-only approaches, the same methods (e.g. ILP and SMT) can be utilized to solve a JRaS problem.

Schweissguth et al. present such an ILP-based approach for JRaS methods and compare it to an two scheduling-only approaches using *Shortest Path Routing* and *Load Balanced Routing* [SDT+17]. The shortest-path routing uses a round-robin scheme while load-balanced routing results in the lowest link utilization. Their results show that the JRaS approach and shortest-path approach produce results of the same quality, while the JRaS approach and the load-balanced approach result in a similar schedulability. So even though the JRaS approach has a greater schedulability than the shortest-path approach and produces lower end-to-end delays than the load-balanced routing, the scheduler runtime is generally higher. In a subsequent work, Schweissguth et al. modify their ILP to support multicast streams [STP+20] and research different opportunities to reduce the scheduler runtime, e.g. by providing additional bounds for scheduling variables.

A work by Hellmanns et al. focuses on minimizing the runtime of a ILP-based JRaS approach without compromising valid solutions [HDHK18]. They achieve this goal by reducing the ILP-complexity (i.e. number of variables and constraints). To this end, they preprocess the network topology in order to reduce the network topology on a per-stream basis, e.g. by removing unreachable vertices and edges. These changes directly affect the number of variables and constraints in the ILP and thus its complexity. With their optimization approach, they are able to reduce the scheduling runtime by a factor of 100. In a later work [Hau20], we research further optimization approaches based on this ILP. For example, we merge variables of subsequent edges, provide additional variable bounds and tweak parameters of the ILP-solver Gurobi. With these optimization approaches, we are able to increase the scheduling capabilities and reduce the runtime of the ILP solver by about 80 % on average. In Section 2.4 we provide a detailed description of this ILP, as we extend it with wireless capabilities in Section 5.1.

4 System Model & Problem Statement

Before we are able to introduce our approaches on how to bring time-critical applications including handovers to wireless environments, we first need to introduce a formalized description of our environment including assumptions on the configuration of its devices. We first formalize the representation of our scenario including the topology and streams. After that, we present our assumptions on the network environment and the configuration and behavior of all involved devices. Thereafter, we present a JRaS ILP-model by Hellmanns et al. [HDHK18] and our previous work [Hau20] which is based on a no-wait scheduling-only ILP model by Dürr et al. [DN16] and which we later extend to support wireless environments. Finally, we formalize the problem we aim to solve and define the objectives for this work.

4.1 System Model

In this section, we formally describe the representation of our scenario, which consists of all devices in a network and the data they exchange. We utilize this description later in order to build an ILP-model and perform algorithms on the network. Firstly, we give a description of the network topology as a graph and provide assumptions about the behavior and configuration of its devices. Thereafter, we describe our model of streams. Both parts also contain a description on simple functions, providing further information about attributes of components in our model.

4.1.1 Topology

Our network topology model is based on graphs, as explained in Section 2.6.1. In this model, every network device is modeled as a vertex. The connections/links between these network devices are modeled as edges.

Our network topology contains different types of network devices, depending on their functionality and their type of connection. Each category is an own subset of all vertices $\mathcal{V}$:

Wired Hosts are end devices, which may be the source and/or target of a stream and do not forward frames to other devices. They only support wired connections and are always connected to one *Switch*. We call the subset of all Wired Hosts $\mathcal{V}_{\text{wd}}$.

Wireless Hosts (STAs) are end devices that communicate through wireless connections and can be the source and/or target of a stream. They do not forward frames to other devices and are associated to none, one or multiple *Access Points* at a time. We call the subset of all Wireless Hosts $\mathcal{V}_{\text{wl}}$.

Switches are network devices responsible for forwarding frames between connected devices within the same network segment. They may not be the source or target of a stream. Switches may have an arbitrary number of wired connections to *Wired Hosts*, *Switches* and *Access Points*. In our network, we require switches to support at least 3 different traffic classes following the PCP mapping of the TSN standards. We call the subset of all Switches $\mathcal{V}_{\text{sw}}$.

Access Points are devices that provide wireless connectivity to wireless hosts. They act as an interface between the wireless hosts and the wired network infrastructure, forwarding data between the wireless and wired segments. An Access Point is always connected to one *Switch* and may have an arbitrary amount of *Wireless Hosts* associated with it. We require every AP to implement the EDCA of the HCF as introduced in Section 2.2.2. We call the subset of all Wireless Access Points $\mathcal{V}_{\text{ap}}$.

Every network device on our network is part of exactly one subset as described above. This means, all of these subsets together form the set of vertices ($\mathcal{V} = \mathcal{V}_{\text{wd}} \cup \mathcal{V}_{\text{wl}} \cup \mathcal{V}_{\text{sw}} \cup \mathcal{V}_{\text{ap}}$). We distinguish between wired and wireless edges, $\mathcal{E} = \mathcal{E}_{\text{wd}} \cup \mathcal{E}_{\text{wl}}$ respectively:

Wired Edges Our graph contains a wired edge $(v_{\text{src}}, v_{\text{dst}}) \in \mathcal{E}$, iff there is a wired connection between v_{src} and v_{dst} (i.e. $v_{\text{src}}, v_{\text{dst}} \in \mathcal{V} \setminus \mathcal{V}_{\text{wl}}$). As Ethernet connections are full-duplex, devices are either not connected or connected in both directions. (i.e. if there is an edge $v_{\text{src}}, v_{\text{dst}}$, there is also an edge $v_{\text{dst}}, v_{\text{src}}$). We assume all Ethernet link have a link speed of 1,000 Mbps.

Wireless Edges Our graph contains a wireless edge for each pair of a *Wireless Host* and an *Access Point* in both directions. We use this model, because a wireless host might associate with different APs over time and our goal is to have a static graph and only change the attributes of vertices and edges. We assume the link speed of a wireless link is always constant with 200 Mbps.

Wireless Environment

As devices in a shared wireless transmission medium behave different, we need more assumptions for our wireless environment. The only way to guarantee a reliable transmission in a wireless medium is by assuming that our environment is isolated from other, neighboring networks. This means, there is no interference originating from any device outside our network. Furthermore, we assume there are no unknown devices inside our network which might send unexpected data.

Regarding the usage of the wireless medium, we assume that our STAs and APs do not support MU-MIMO. This means that only one device is able to use the wireless transmission medium in one channel at a time.

Furthermore, we model the transmission range of APs and STAs using a *Unit Disk Radio Model*. In this model, the transmission and interference are defined by providing the maximum distance for a successful transmission. Figure 2.3 illustrates this with a circle around the APs, where the radius of the circle represents the transmission range. Furthermore, the quality of a link does not degrade with distance between devices (i.e. either the devices are within the transmission range and a successful communication is possible (given there is no interference) with a constant link speed or the devices are not within the transmission range and do not detect the transmissions at all).

Mobility Model

In order to model the movement of devices we introduce a new graph $\mathcal{G}_{\text{ap}} = (\mathcal{V}_{\text{ap}}, \mathcal{E}_{\text{range}})$. $\mathcal{E}_{\text{range}}$ contains an edge, if the transmission range of both APs overlap. Furthermore, we provide each STA with a set of APs, to which it is allowed to connect, leading to the following helper functions:

- in_range : $\mathcal{V}_{\text{ap}} \rightarrow \mathcal{P}(\mathcal{V}_{\text{ap}})$ gives all APs which have an overlapping range to the provided AP.
- ap_list : $\mathcal{V}_{\text{wl}} \rightarrow \mathcal{P}(\mathcal{V}_{\text{ap}})$ gives all APs to which the provided STA is allowed to connect.

To ensure a reliable connection, we assume that every STA always has at least one AP in range to associate with. In combination this means that a STA never moves to an area uncovered by at least one AP on its ap_list.

Attributes

As mentioned before, the different components of our network model also contain specific attributes. In order to access these attributes later, we provide a list of global functions:

- in_edges : $\mathcal{V} \rightarrow \mathcal{P}(\mathcal{E})$ gives a set of all incoming edges of a vertex v. In order to only access wired or wireless edges, we use the functions $\text{in\_edges}_{\text{wd}}$ or $\text{in\_edges}_{\text{wl}}$ respectively.
- out_edges : $\mathcal{V} \rightarrow \mathcal{P}(\mathcal{E})$ gives a set of all incoming edges of a vertex v. In order to only access wired or wireless edges, we use the functions $\text{out\_edges}_{\text{wd}}$ or $\text{out\_edges}_{\text{wl}}$ respectively.
- position : $\mathcal{V} \rightarrow \mathbb{R} \times \mathbb{R}$ gives a 2D position (p_1, p_2) of a device.
- processing_delay : $\mathcal{V}_{\text{sw}} \cup \mathcal{V}_{\text{ap}} \rightarrow \mathbb{N}$ gives the processing delay in ns.
- channel : $\mathcal{V}_{\text{ap}} \rightarrow \mathbb{N}$ gives the channel of an AP.
- ap : $\mathcal{E}_{\text{wl}} \rightarrow \mathcal{V}_{\text{ap}}$ gives the involved AP of this edge.
- propagation_delay : $\mathcal{E} \rightarrow \mathbb{N}$ gives the propagation delay of a link in ns. This is constant for wired links but might change for wireless links.
- link_speed : $\mathcal{E} \rightarrow \mathbb{N}$ gives the link speed of a link in bps.
- is_associated : $(v_{\text{wl}}, v_{\text{ap}}) \mapsto \begin{cases} 1 & \text{if } v_{\text{wl}} \in \mathcal{V}_{\text{wl}} \text{ is currently associated with } v_{\text{ap}} \in \mathcal{V}_{\text{ap}} \\ 0 & \text{otherwise} \end{cases}$
- $\text{dist}(v, w) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}$ gives the Euclidean distance between two network devices v, w based on their coordinates $(p_1, p_2) = \text{position}(v)$ and $(q_1, q_2) = \text{position}(w)$ respectively.

4.1.2 Traffic Model

In this work, we refer to a stream, as repeating transfer of data from one host to another. Our set of streams $\mathcal{S}$ contains one entry $(v_{\text{src}}, v_{\text{dst}}, \text{id})$ per stream. $v_{\text{src}}, v_{\text{dst}} \in (\mathcal{V}_{\text{wd}} \cup \mathcal{V}_{\text{wd}})$ are the source and destination of stream, whereas $\text{id} \in \mathbb{N}$ is a unique identifier belonging to at most one stream.

As mentioned above, the streams in our work have exactly one sender and exactly one receiver. However, each host may be the sender or receiver of multiple streams. This prevents us from routing streams based on the sender or receivers MAC-address. In order to distinguish between different streams, we assign a unique multicast address to each stream. Only the intended receiver of this stream is subscribed to this multicast address. This allows us to route each stream independently based on the multicast address.

Traffic Classes

In this work, we distinguish between three different traffic classes, namely *network control*, *time-critical* and *best-effort*. We map these traffic classes to the PCP values 7, 5 and 1 respectively. Our streams always use the *time-critical* traffic class with a PCP value of 5. In APs following the QoS standard, this means the *network control* traffic uses *AC_VO*, *time-critical* traffic uses *AC_VI* and *best-effort* traffic uses *AC_BE*.

Attributes

Streams have more attributes, which we access using global functions, similar to the network topology functions:

- stream_size : $\mathcal{S} \rightarrow \mathbb{N}$ gives a streams size in Bytes
- cycle_time : $\mathcal{S} \rightarrow \mathbb{N}$ gives the cycle time of a stream in ns.

Later in this work, we limit the vertices and edges when routing streams in different scenarios. The following functions provide a reduces set of vertices or edges given a stream:

- stream_vertices : $\mathcal{S} \rightarrow \mathcal{P}(\mathcal{V})$ gives all vertices which a stream might traverse when making a routing decision.
- stream_edges : $\mathcal{S} \rightarrow \mathcal{P}(\mathcal{E})$ gives all edges which a stream might utilize when making a routing decision. $\text{stream\_edges}_{\text{wd}}$ and $\text{stream\_edges}_{\text{wl}}$ are modified functions which only return wired and wireless edges respectively.
- $\text{transmission\_delay}(s, e) = \frac{8 \cdot \text{stream\_size}(s)}{\text{link\_speed}(e)} \cdot 10^9$ calculates the transmission delay of a stream s on edge e in ns.

4.2 ILP Model

There are different approaches on how to solve the *JRaS* problem. On of these approaches is the usage of an ILP. In this work we build upon an already existing ILP by Hellmanns et al. [HDHK18] and our previous work [Hau20] The following sections contain a description of variables and constraints responsible for routing and scheduling streams and managing conflicts between different streams. The provided ILP model is a no-wait ILP model only supporting full-duplex point-to-point connections. We present our changes to this ILP in Section 5.1.

4.2.1 Routing Constraints

The ILP model by Hellmanns et al. represents the routing decision for a stream by introducing a binary decision variable for each combination of a stream s and an edge e:

$$x_{s,e} = \begin{cases} 1 & \text{if } v_s \in \mathcal{S} \text{ uses edge } e \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

Using these variables, they are able to define all constraints relevant for the routing decision. As every stream has exactly on source and exactly one destination node, every stream may utilize exactly one out-edge of its source and one in-edge of its target. These constraints enforce that the stream starts and ends at its desired vertices:

$$\sum_{e \in \text{out\_edges}(v_{\text{src}})} x_{s,e} = 1 \tag{4.2}$$

$$\sum_{e \in \text{in\_edges}(v_{\text{dst}})} x_{s,e} = 1 \tag{4.3}$$

To ensure a continuous flow, without interruptions or other multiple start and end vertices Hellmanns et al. provide another constraint. This constraint ensures that for every vertex (excluding source and destination) the number of utilized in-edges has to be equal to the number of utilized out-edges:

$$\forall v \in \mathcal{V} \setminus \{v_{\text{src}}, v_{\text{dst}}\} : \sum_{e \in \text{in\_edges}(v)} x_{s,e} = \sum_{e \in \text{out\_edges}(v)} x_{s,e} \tag{4.4}$$

In our previous work [Hau20], we discovered that it is necessary to restrict a stream to visit every vertex at most once in order to calculate correct schedules with the following ILP. Additionally, this is required, because our network devices make routing decisions solely based on the destination of a stream and could not determine which edge to utilize next otherwise. We can achieve this behavior by limiting the number of utilized in-edges to one for every vertex:

$$\forall s \in \mathcal{S}, \forall v \in \mathcal{V} : \sum_{e \in \text{in\_edges}(v)} x_{s,e} \leq 1 \tag{4.5}$$

Even with this change, the ILP might still generate isolated loops. Due to our conversion of the ILP results to a GCL (see Section 5.1.5), this does not impose any problems on the resulting schedules.

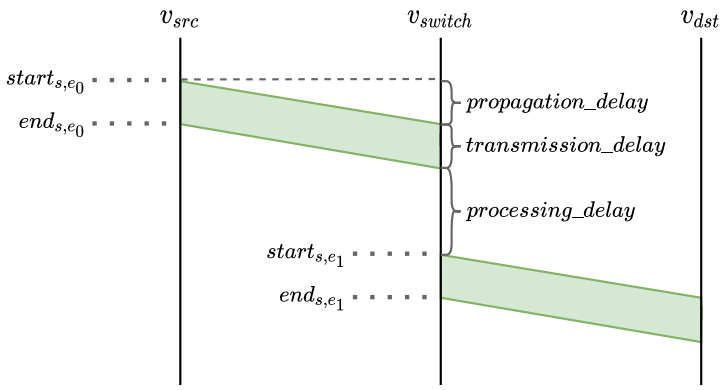


Figure 4.1: Variables and delays relevant for providing scheduling constraints.

4.2.2 Scheduling Constraints

In order to schedule streams along the route provided by the scheduling constraints, Hellmanns et al. define two more variables for every combination of a stream and an edge with the following meaning:

- $start_{s,e}$: Provides the time a network device starts transmitting the data for stream s on the transmission medium represented by edge e.
- $end_{s,e}$: Provides the time a network device finishes transmitting data for stream s on the transmission medium represented by edge e.

Figure 4.1 shows the relation between the delays and the start and end variables, we use this graphic to explain the scheduling constraints provided by Hellmanns et al. In case a stream s does not utilize this edge e, this variable needs to be zero. Also, we need to ensure that a scheduled stream does not exceed the cycle time. The following constraint achieves both requirements:

$$\forall s \in \mathcal{S}, \forall e \in \mathcal{E} : end_{s,e} \leq x_{s,e} \cdot \text{cycle\_time} \tag{4.6}$$

As Figure 4.1 shows us, the end variable of a stream and edge is dependent on the start variable of the same stream and the corresponding transmission delay. If a stream does not utilize an edge, the *start* and *end* variables are 0 and we cannot take the transmission delay into account. This leads to the follow constraint:

$$\forall s \in \mathcal{S}, \forall e \in \mathcal{E} : end_{s,e} = start_{s,e} + x_{s,e} \cdot \text{transmission\_delay}(s, e) \tag{4.7}$$

In a last step, Hellmanns et al. define a constraint, which sets the *start* variable of an edge in relation to the *end* variable of its previous edge for every vertex. As every intermediate vertex might have multiple in- and out-edges, we need to take all of these into account at once when setting the *start* and *end* variables into relation. Constraint (4.5) ensures that every stream might use at most one in- and out-edge of a vertex. In combination with Constraints (4.6) and (4.7) this ensures that at most one *start* and one *end* variable holds a non-zero value for every stream and intermediate

vertex. Therefore, the sum of all *start* variables of out-edges (lhs) and the sum of all *end* variables of in-edges (rhs) are the sum of the utilized out- and in-edge respectively. Together with the delays as shown in Figure 4.1, this leads to the following constraint:

$$\forall s = (v_{\text{src}}, v_{\text{dst}}, \_) \in \mathcal{S}, \forall v \in \mathcal{V} \setminus \{v_{\text{src}}, v_{\text{dst}}\} :$$

$$\sum_{e \in \text{out\_edges}(v)} start_{s,e} \tag{4.8}$$

$$= \sum_{e \in \text{in\_edges}(v)} end_{s,e} + x_{s,e} \cdot (\text{propagation\_delay}(e) + \text{processing\_delay}(v))$$

4.2.3 Conflict Constraints

Now, all constraints required to schedule a single stream are defined. The existence of multiple streams might lead to collisions between those streams, if they are scheduled on an edge at the same time. Hellmanns et al. define another set of constraints, which prohibits a schedule with overlapping usage of an edge. For these constraints another variable for every combination of two distinct streams $s_1, s_2 \in \mathcal{S}$ ($s_1 \neq s_2$) and edge e is required:

$$b_{s_1,s_2,e} = \begin{cases} 1 & \text{if } s_1 \text{ is scheduled before } s_2 \text{ on edge } e \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

This binary decision variable is used to define conditional constraints as *Big M* constraints, as described in Section 2.4. As the maximum value any variable in the ILP can take is the cycle time it is safe to set M to the value of the biggest cycle time among all streams, as bigger values for M might lead to numerical issues [Gur]. The following two conditional constraints ensure that the *end* variable of the first stream is smaller than then *start* variable of the next stream, depending on binary decision variable $b_{s_1,s_2,e}$:

$$\begin{aligned} &\forall (s_1, s_2, e) \in (\mathcal{S} \times \mathcal{S} \times \mathcal{E}) (\text{with } s_1 \neq s_2) : \\ &end_{s_1,e} \leq start_{s_2,e} \quad + (1 - b_{s_1,s_2,e}) \cdot M \\ &end_{s_2,e} \leq start_{s_1,e} \quad + b_{s_1,s_2,e} \cdot M \end{aligned} \tag{4.10}$$

4.3 Problem Statement

Reliable transmissions with deterministic bounds on latency and jitter are a key-requirement for time-sensitive applications. Achieving these deterministic real-time guarantees in wireless network environments imposes several additional challenges compared to wired-only networks. This includes the usage of a shared wireless transmission medium and the mobility of devices within the network. While scheduling streams for time-sensitive applications is already well-researched (see Chapter 3),

these approaches are not suitable for wireless network environments without modifications. In this chapter, we take a closer look on the challenges arising in wireless network environments and their impact on already existing approaches.

In a bridged wired network environment, network devices are typically connected by dedicated point-to-point connections (e.g. Ethernet cables) with a constant transmission speed. However, devices in a wireless environment share access onto the same transmission medium and the connection speed might change depending on several factors, such as obstacles and the distance between STAs and APs. Current scheduling approaches are incompatible or not suitable for these environments. Firstly, resolving conflicts between two streams per link as described in Section 4.2.3 does not take conflicts of other devices concurrently utilizing the same transmission medium into account. Additionally, existing approaches often implement no-wait schedules without an additional queuing time or buffering on intermediate devices. While these no-wait schedules are still possible in a network with wireless devices, they have two main disadvantages:

- A delay introduced due to the usage of the shared wireless medium in the beginning and/or end of a stream shifts the schedule of the whole stream forward or backward in time. This is especially problematic in cases where both, source and target, are wireless devices as it does not allow for buffering streams and thus removes freedom of scheduling streams onto the shared wireless medium.
- With no-wait schedules, the only way to influence the end-to-end delay for a packet is by using a longer path as necessary in order to increase the end-to-end delay. But if we want to reduce jitter, buffering packets en route is a useful mechanism to control the reception time of a packet and thus keeping the delay constant.

One main advantage of a wireless network is the opportunity to have mobile devices such as moving robots or workers in an exoskeleton. However, APs are only able to cover a limited area. Therefore, handovers between APs are inevitable when the mobility area of a device exceeds the transmission range of an AP. Current scheduling approaches are mostly static and do not provide the opportunity of dynamic routes (i.e. changing routes if a mobile STA changes its association with APs due to mobility). Extending current approaches to support handovers of devices and thus scheduling the same stream along different routes introduces further challenges, such as an increased jitter due to changing paths. Additionally, handovers itself are a time-consuming process [AH08; FS17], especially due to the scanning phase of the association procedure as described in Section 2.2.3. During this handover phase, there is no established connection between the wireless interface of a STA and AP and therefore no transmission of time-sensitive frames is possible. Furthermore, after the handover of a STA, the route of all streams including this STA changes.

To address these aforementioned challenges, our work has the following objectives:

1. Provide a scheduler implementation which prohibits the simultaneous usage of the shared wireless transmission medium by different devices.
2. Provide a scheduler implementation which allows buffering of packet en route to increase the scheduling flexibility.
3. Optimize schedules to minimize mobility-induced jitter due to dynamically changing routes because of handovers between APs.

4. Propose a handover approach which increases reliability by reducing the packet loss during a handover.
5. Define mechanisms for dynamic reconfiguration of routes to support handovers of STA with time-sensitive applications.

Our goal of this work is to provide possible solutions to the aforementioned challenges and evaluate their effect on the different requirements, namely packet loss, delay and jitter. In the next Chapter 5, we present our approaches aiming to reach the defined objectives.

5 Design

In this chapter, we present our approaches aiming to reach our objectives defined in the previous chapter. We start by providing multiple amendments to an already existing ILP in order to meet the requirements induced by the different nature of wireless network environments. This includes modifications to allow buffering for schedules, the adaption for a shared wireless transmission medium and the support of roaming STA. Furthermore, we provide a new objective to reduce mobility-induced jitter and explain how we convert the generated schedules to GCLs. Thereafter, we introduce our novel handover approach utilizing multiple wireless interfaces in one STA, followed by a description of our STA specific handover controller and its different configurations. In this description, we explain how our handover controller performs handover decisions and notifies our MAC forwarding table configurator about handovers in order to update the forwarding table of switches and APs in the network.

5.1 Schedule Generation & Optimization

As already described in our Problem Statement in Section 4.3, calculating routes and schedules for networks including wireless devices imposes new challenges, such as the usage of a shared wireless transmission medium. In this section, we present our approaches to solve these challenges and reach the defined objectives for our scheduler.

Our scheduling approach is based on the ILP we introduced in Section 4.2. In the following, we first enable this ILP to support buffering of packets en route. Thereafter, we adapt it to meet the requirements of a shared wireless transmission medium. We then present an approach which reflects handover-induced route changes in the ILP. Following that, we provide different optimization objectives for the ILP, which aim to minimize the end-to-end delay and jitter respectively and a combination of these. Finally, we present how we generate GCLs from the ILP results.

5.1.1 Scheduling with Buffering

In our Problem Statement (Section 4.3), we described the disadvantages of utilizing no-wait schedules in a shared wireless network environment. This shows that buffering frames en route is desirable in order to reduce mobility-induced jitter. Thus, our first approach is to modify the existing ILP for no-wait packet scheduling in fixed networks to support schedules with buffering at intermediate device in the network.

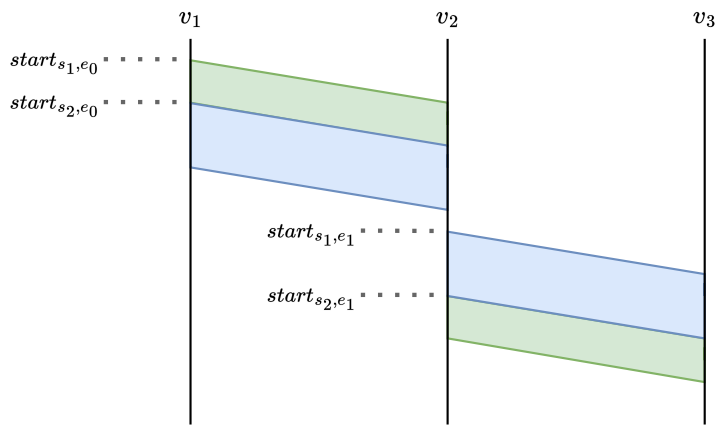


Figure 5.1: One stream overtaking another stream due to unrestricted buffering.

Currently, Constraint (4.8) is responsible for a continuous schedule of a stream between the in-edges and out-edges of every vertex in a no-wait fashion. In order to allow buffering at this vertex, we can relax this constraint by allowing the start time at the egress port to also be greater than time when the packet has been received from the ingress port. This leads to the following constraint:

$$\forall s = (v_{\text{src}}, v_{\text{dst}}, \_) \in \mathcal{S}, \forall v \in \mathcal{V} \setminus \{v_{\text{src}}, v_{\text{dst}}\} :$$

$$\sum_{e \in \text{out\_edges}(v)} start_{s,e} \tag{5.1}$$
$$\geq \sum_{e \in \text{in\_edges}(v)} end_{s,e} + x_{s,e} \cdot (\text{propagation\_delay}(e) + \text{processing\_delay}(v))$$

Even though this constraint now allows for buffering, it introduces a new problem, namely possible reordering of queued packets. Figure 5.1 demonstrates this problem. In the example streams s_1 (green) and s_2 (blue) both utilize the same path $v_1 \rightarrow v_2 \rightarrow v_3$ and as the streams do not overlap the conflict Constraint (4.10) is fulfilled on both edges. Due to the buffering of stream s_1 at vertex v_2, s_2 overtakes stream s_1 within this vertex. This is unwanted behavior, as real-world devices normally use *First in - First out (FIFO)* queues which do not allow for reordering of packets stored in the same egress queue of a switch. Therefore, we need further modifications to the ILP-model in order to ensure the FIFO order of packets.

There are two possible ways to represent the FIFO behavior in the ILP. On way is to introduce a new set of conflict constraints, which prohibits overtaking at every vertex for every combination of two distinct streams. Using this approach requires the introduction of a new binary decision variable for each of these combinations and two new constraints. This drastically increases the complexity of the ILP by up to $n_{\text{vertices}} \cdot n_{\text{streams}}^2$ new variables and $2 \cdot n_{\text{vertices}} \cdot n_{\text{streams}}^2$ new constraints. Another way is to modify the currently existing conflict constraint in such a way that it also enforces the FIFO property on all vertices. We achieve this by modifying the binary decision variable from Constraint (4.9) to decide whether a stream s_1 is scheduled before s_2 on a per-stream basis only. This leads to the new binary decision variable for resolving conflicts (namely conflict variable) as shown in Equation (5.2) and the new Constraint (5.3)

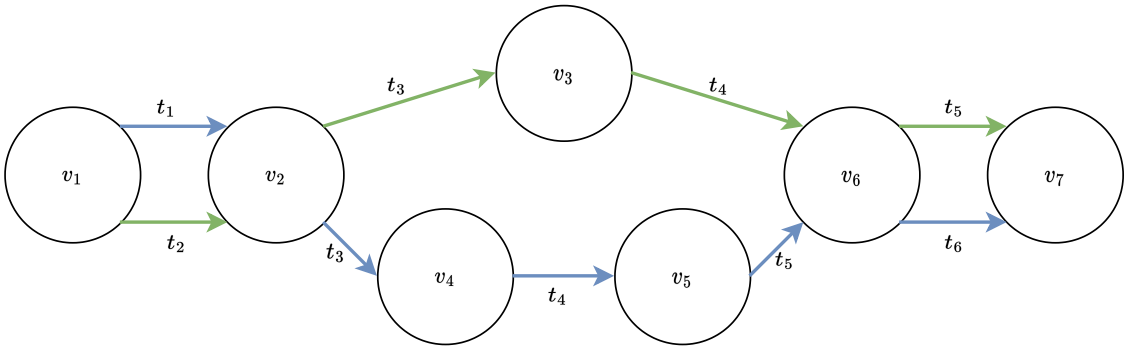


Figure 5.2: Example of one stream (green) overtaking another stream (blue) by using a different path. Timestamps (t_x) happen in ascending order.

$$\forall (s_1, s_2) \in (\mathcal{S} \times \mathcal{S}) (\text{with } s_1 \neq s_2) :$$
$$b_{s_1,s_2} = \begin{cases} 1 & \text{if } s_1 \text{ is scheduled before } s_2 \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

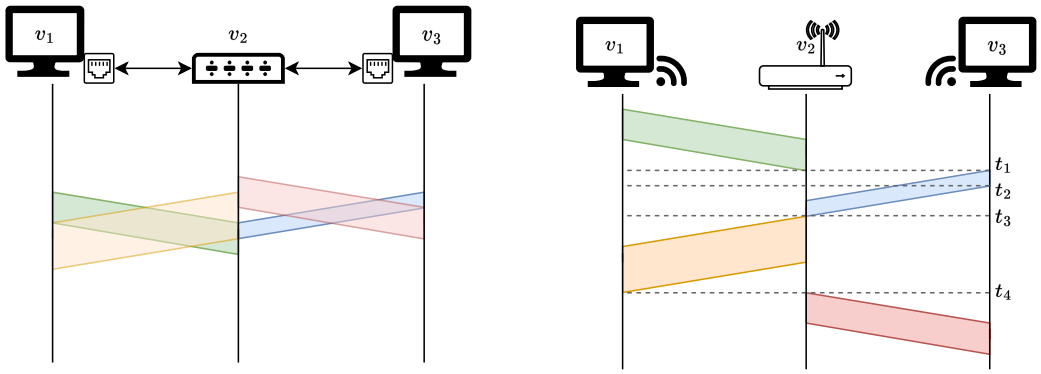
$$\begin{aligned} end_{s_1,e} &\leq start_{s_2,e} &&+ (1 - b_{s_1,s_2}) \cdot M \\ end_{s_2,e} &\leq start_{s_1,e} &&+ b_{s_1,s_2} \cdot M \end{aligned} \tag{5.3}$$

This ensures that stream s_1 is always scheduled before s_2 or vice versa independent of an edge. Therefore, it also enforces the FIFO property but reduces the solution space, as Figure 5.2 shows. In this example the green stream utilizes the edge (v_1, v_2) after the blue stream but then takes a shorter path via v_3, while the blue stream takes the longer path via v_4 and v_5. This allows the green stream to overtake and utilize the edge (v_6, v_7) before the blue stream.

Deciding among the ways to ensure the FIFO property is a trade-off between reducing the solution space and increasing the complexity of the ILP. As the amount of cases, where overtaking as in Figure 5.2 might be useful is small in many practical network topologies, our approach is to modify the existing constraints as presented above in Constraint (5.3) by accepting a reduction of the solution space in theory.

5.1.2 Shared Wireless Transmission Medium

As mentioned before, in bridged Ethernet networks, Ethernet devices are typically connected in a full-duplex fashion via point-to-point connections, which allow them to transmit data in both directions and to every connected device at the same time (Figure 5.3a) without collision and contention for the medium. Opposing to that, wireless devices share the same transmission medium. With shared wireless links, sending data between multiple devices on the same channel at the same time is not possible due to interference (assuming no MU-MIMO support). However, our new conflict Constraint (5.3) assumes full-duplex and point-to-point connections between neighboring devices and thus is not suitable for wireless environments.



(a) Two wired Ethernet devices sending and receiving data to/from a switch at the same time over different cables.

(b) Two WiFi devices transmitting and receiving data to/from an AP in a shared wireless environment.

Figure 5.3: Two devices sending and receiving data to/from one other device in two different environments.

To solve this problem, we propose a new constraint which fulfills the requirements of a shared medium that only one device transmits data at a time (Figure 5.3b). In this constraint, we also need to take propagation delay into account. Otherwise, the orange stream would be scheduled at t_2 instead of t_3 in Figure 5.3 which would result in an interference of the blue and orange stream at v_1. This new constraint as shown in Constraint (5.4) is similar to our conflict Constraint (5.3):

$$\begin{aligned} end_{s_1,e} + \text{propagation\_delay}(e) \leq start_{s_2,e} \quad &+ (1 - b_{s_1,s_2}) \cdot M \\ end_{s_2,e} + \text{propagation\_delay}(e) \leq start_{s_1,e} \quad &+ b_{s_1,s_2} \cdot M \end{aligned} \tag{5.4}$$

As the propagation delay in a wireless medium is not constant but instead depends on the distance between two devices, we calculate the maximum propagation delay based on the maximum transmission range of the involved devices.

Furthermore, we need to update the algorithm which generate the conflict constraints presented in Constraint (5.3) and Constraint (5.4), as conflicts on the wireless transmission medium do not necessarily share the same edge. Algorithm 5.1 shows our proposed algorithm. It iterates over all combinations of streams, while ensuring that we consider each pair of streams only once (line 3). For each of these combinations, we generate the conflict decision variable as in Equation (5.2). We then generate conflict constraints for wired edges for every shared edge between the two streams as in defined Equation (5.3). On wireless edges, a conflict can occur on any edge involving the same AP. To this end, we need to iterate over all combinations of the wireless edges between the two streams and add the wireless conflict constraint as defined in Constraint (5.4) only if both edges use the same AP.

Algorithm 5.1 Conflict Constraint Generation Algorithm

```
1:  procedure ConflictConstraintGenerator(streams: List[Stream], ilp: ILP)
2:      for each s1 in streams do
3:          for each s2 in streams do
4:              if s1.id < s2.id then
5:                  b ← ilp.addBinaryVariable(beforeVariable(s1,s2)) // beforeVariable as in (5.2)
6:                  GenerateWired(s1,s2,b,ilp)
7:                  GenerateWireless(s1,s2,b,ilp)
8:              end if
9:          end for
10:     end for
11:     return colors
12: end procedure
13: procedure GenerateWired(s1: Stream, s2:Stream, b: ILP.Var, ilp: ILP)
14:     for each edge in stream_edges_wd (s1) ∩ stream_edges_wd (s2) do
15:         ilp.add(wiredConstraint(b))                    // wiredConstraint as in (5.3)
16:     end for
17: end procedure
18: procedure GenerateWireless(s1: Stream, s2: Stream, b: ILP.Var, ilp: ILP)
19:     for each edge1 in stream_edges_wl (s1) do
20:         for each edge2 in stream_edges_wl (s2) do
21:             if ap(edge1) = ap(edge2) then
22:                 ilp.add(wirelessConstraint(b))         // wirelessConstraint as in (5.4)
23:             end if
24:         end for
25:     end for
26: end procedure
```

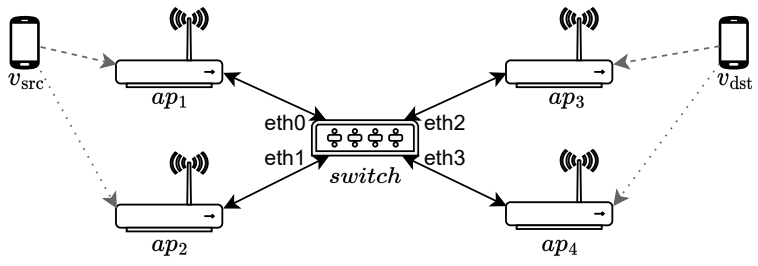


Figure 5.4: A scenario with two moving mobile devices.

5.1.3 Schedules and Routes for Mobile Stations

The ILP presented in the previous section so far supports network environments with wired and wireless devices and is able to calculate routes and schedules. However, so far, it only supports stationary wireless devices, which are always connected to the same AP. As we want to enable our ILP to schedule streams for STAs roaming between APs as well, we need to modify or ILP generation procedure.

Figure 5.4 shows an example with two STAs v_{src} and v_{dst}. v_{src} is currently associated with ap_1, but moves downwards towards ap_2 and v_2 is associated with ap_3 and moves downwards towards ap_4. In this scenario, both devices will associate with different AP over time. Therefore, a stream going from v_{src} to v_{dst} may utilize the four following paths: $(v_{\text{src}} \rightarrow ap_1 \rightarrow switch \rightarrow ap_3 \rightarrow v_{\text{dst}})$, $(v_{\text{src}} \rightarrow ap_1 \rightarrow switch \rightarrow ap_4 \rightarrow v_{\text{dst}})$, $(v_{\text{src}} \rightarrow ap_2 \rightarrow switch \rightarrow ap_3 \rightarrow v_{\text{dst}})$ and $(v_{\text{src}} \rightarrow ap_2 \rightarrow switch \rightarrow ap_4 \rightarrow v_{\text{dst}})$

Our approach on handling these scenarios treats every possible route of a stream for every possible STA/AP combination as a single stream. To achieve this behavior, we use the initially provided stream as a template called stream template. We then use these templates to generate a stream instance for every possible route in the following way:

1. For every wireless node v and every entry $u \in \text{ap\_list}(v)$ we generate a new vertex v_u and a wireless edge $e = (v, v_u)$.
2. For every stream template s with a wireless node as a source or target, we generate a stream instance s_i for every AP-association combination of source and target. We then set the source and target of the stream instance to the matching vertices generated in Step 1. In our example this leads to the following four streams instances:
   - s_{ap_1,ap_3} with source vertex v_{src,ap_1} and target vertex v_{dst,ap_3}
   - s_{ap_1,ap_4} with source vertex v_{src,ap_1} and target vertex v_{dst,ap_4}
   - s_{ap_2,ap_3} with source vertex v_{src,ap_2} and target vertex v_{dst,ap_3}
   - s_{ap_2,ap_4} with source vertex v_{src,ap_2} and target vertex v_{dst,ap_4}
3. For every stream instance s_i, we add the new source and target vertices to stream_vertices(s) and the corresponding edges to stream_edges(s)

With this modification our generated ILP provides independent, non-overlapping schedules for scenarios with mobile devices. However, it also contains conflict constraints between stream instances originating from the same stream template. This corresponds to conflict constraints being generated pairwise for the streams s_{ap_1,ap_4}, s_{ap_2,ap_3}, s_{ap_1,ap_4} and s_{ap_2,ap_4} in our example. More specifically, this prohibits utilizing an edge (e.g. $(ap_1, switch)$) at the same time as any other (copied) stream within the cycle. This unnecessarily reduces our solution space, as only one of the copied streams is active at a time depending on the association of the involved source and target vertices. We prohibit this behavior by adding another check to our Conflict Constraint Generation Algorithm 5.1 in line 4 and only continue, if the id of the originating stream of s_1 and s_2 are different. This allows streams originating from the same copy to utilize edges at the same time and resolved conflicts for all other streams as before. With these changes we have an ILP which is able to schedule streams in a mixed wired and wireless environment without sacrificing possible solutions from our solution space. However, the calculated schedules for stream instances originating from the

same stream template are completely independent, which might lead to mobility-induced jitter when switching from one stream instance (one AP) to another stream instance (another AP). This problem is considered next by introducing an optimization objective to minimize this jitter during handovers.

5.1.4 Optimization Goal

As mentioned in Section 2.4, ILPs also allow us to specify an optimization goal. For time-sensitive applications there are two main optimization goals, namely minimizing end-to-end delay and jitter.

In our ILP the end-to-end delay is constant for a stream instance, as the route and schedule of one stream never changes. Thus, for fixed networks with the same cycle time for all streams, we can only optimize for the end-to-end delay. One possible way is to provide a linear expression of the sums of all end-to-end delays and providing this as a minimization objective to the ILP solver:

$$\sum_{s=(v_{\text{src}},v_{\text{dst}},\_)\in\mathcal{S}} \left[\sum_{e\in\text{in\_edges}(v_{\text{dst}})} end_{s,e} - \sum_{e\in\text{out\_edges}(v_{\text{src}})} start_{s,e} \right] \tag{5.5}$$

If wireless devices are part of the network, the mobility-induced jitter is the difference of the end times of streams originating from the same stream (see previous Section 5.1.3). In order to provide a minimization objective for jitter, it is crucial to always treat jitter as a positive value. This means that for two streams s_1 and s_2 the jitter is $|\text{end}(s_1) - \text{end}(s_2)|$ and thus positive no matter which stream is scheduled first. ILPs do not support absolute values natively, thus we need additional variables and constraints to achieve this behavior. The first step is to introduce a new variable for every combination of streams originating from the same stream:

$$\begin{aligned} &\forall(s_1, s_2) \in (\mathcal{S}\times\mathcal{S})(\text{with } s_1 \neq s_2 \wedge \text{origin}(s_1) = \text{origin}(s_2)) : \\ &\quad jitter_{s_1,s_2} : \text{Upper limit of the jitter between } s_1 \text{ and } s_2 \end{aligned} \tag{5.6}$$

To ensure that variable $jitter_{s1,s2}$ defines an upper bound on the jitter when switching from stream instance s1 to instance s2, we need two additional constraints for every variable originating from the same stream template.

$$\begin{aligned} \forall(s_1 = (\_, v_{\text{dst}}, \_), s_1 = (\_, u_{\text{dst}}, \_)) \in (\mathcal{S}\times\mathcal{S})(\text{with } s_1 \neq s_2 \wedge \text{template}(s_1) = \text{template}(s_2)) : \\ jitter_{s_1,s_2} \geq \sum_{e\in\text{in\_edges}(v_{\text{dst}})} end_{s_1,e} - \sum_{e\in\text{in\_edges}(u_{\text{dst}})} end_{s_2,e} \\ jitter_{s_1,s_2} \geq \sum_{e\in\text{in\_edges}(u_{\text{dst}})} end_{s_2,e} - \sum_{e\in\text{in\_edges}(v_{\text{dst}})} end_{s_1,e} \end{aligned} \tag{5.7}$$

Note, that the rhs of one constraint is the negative of the other. Thus, the jitter variable is always lower bounded by one of the two constraints and indeed provides an upper bound for the jitter between the two streams. With these new variables and constraints, we are able to define our minimization objective to reduce jitter (jitter_vars is the set of all variables created in (5.6)):

$$\sum_{j\in\text{jitter\_vars}} j \tag{5.8}$$

With these objectives either the sum of end-to-end delays or the sum of jitters is minimized instead of minimizing the end-to-end delay and jitter for each stream individually. This can theoretically lead to a lot of streams having a low end-to-end delay/jitter and a few streams having a high end-to-end delay/jitter. If one wants to prioritize specific streams in the optimization it is possible to provide a weight before the delay or jitter of the specific stream. In the same way it is possible to optimize for end-to-end delay and jitter at the same time by multiplying the respective objectives with a desired weight w and adding them:

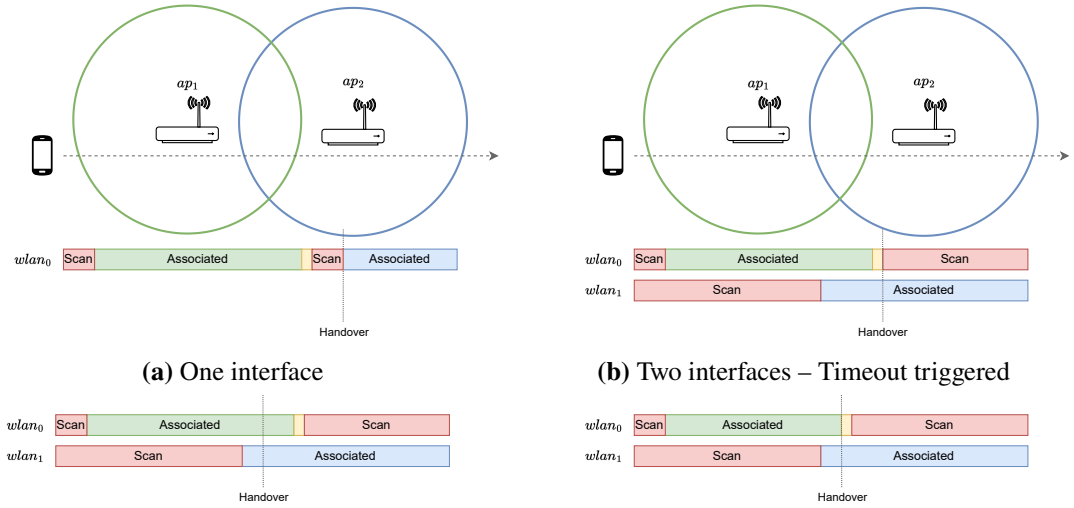
$$w_{\text{jitter}} \cdot \text{jitter\_objective} + w_{\text{e2e}} \cdot \text{e2e\_objective} \tag{5.9}$$

5.1.5 GCL Generation

The result of an ILP consists of values for all variables, so that they satisfy all provided constraints. This corresponds to the routing decisions ($x_{s,e}$) for each stream and the start and end times ($start_{s,e}$ and $end_{s,e}$)for all links along this route in our ILP. As the main purpose of calculating schedules is the configuration of GCLs, we use our ILP results to calculate GCL entries in the following way:

1. For each stream s, we follow the path from the source to the destination node according to the routing variables $x_{s,e}$. For every edge e along that path, we store a pair of the respective start and end variables ($start_{s,e}$, $end_{s,e}$) to a list l_e. This ensures that we do not consider any isolated loops generated by the ILP.
2. For each edge e, we sort the list l_e according to the $start$ variable of every pair. (This leads to the same results as sorting according to the end variables, as streams to not overlap or overtake each other)
3. For every pair in the list, we calculate the duration for the open gate state. As the back-off algorithm of the IEEE 802.11 MAC protocol introduces randomness, we double the time of the open gate state in order to mitigate the effect of delayed packets. This results in the following formula: $t = 2 \cdot (end - start)$.
4. For every edge e we generate the first entry of the GCL on the source node of e in the following way:
   a) If the start time of the first list element is 0, start with an open gate state of duration t
   b) If the start time of the first list element is greater than 0, start with a closed gate state of duration $start$ and add an open gate state for duration t afterwards.
5. For all consecutive list entries, we perform the following steps (note, that the last gate state is always an open gate state):
   a) If the start time is less or equal than the end time of the last GCL entry (end_{gcl}), extend the GCL entry by $t - (start - end_{\text{gcl}})$
   b) If the start time is greater than the end time of the last GCL entry, add a closed gate entry for duration $start - end_{\text{gcl}}$ and an open gate entry for duration t afterwards.
6. If the end time of the last GCL entry is smaller than the cycle time, add a closed gate entry for duration cycle_time $- end_{\text{gcl}}$

The result of this algorithm is one GCL for every node in the network.



(a) One interface

(b) Two interfaces – Timeout triggered

(c) Two interfaces – Location triggered

(d) Two interfaces – Scheduled Disassociations

Figure 5.5: Handovers with different number of Wi-Fi interfaces and different handover decisions. Simplified view excluding the authentication and association phase. – Red: Scan phase; Green/Blue: Associated with ap_1 or ap_2 respectively; Yellow: Association timing out (i.e. The STA is still associated to the AP but is too far away to transmit or receive data)

5.2 Handover

With the ILP approach of the previous chapter, we are able to generate schedules for streams with mobile STAs. As we mention in our problem statement in Section 4.3 we need smooth handovers in order to enable real-time applications on wireless devices. In this section we present our approach on a smooth handover. We start by presenting the idea of equipping STAs with multiple wireless interfaces. Following that, we introduce our procedure for making handover decisions. Finally, we describe how we update MAC forwarding tables after a successful handover.

5.2.1 Multiple Wireless Interfaces

As Figure 5.5a illustrates, a main problem during the handover is the time without an active connection to any AP. In this figure, we can see that after the STA moves out of the transmission range of ap_1 and is not able to transmit or receive data anymore. It takes a few missed beacons until the STA knows it lost connection to its associated AP (yellow phase) and initiates a new scan (red phase). During this time no data can be transferred between the STA and the rest of the network resulting in lost packets.

Our first approach to reduce the effects of the timeout and scan phase is by equipping the STAs with a second Wi-Fi interface. In order to use these two Wi-Fi interfaces in a way that allows us to reduce the effects of a handover, we introduce an additional *Handover Controller* in every STA. This *Handover Controller* has access to the status of every Wi-Fi interface of its STA and is able to control their association decisions. Our proposed handover controller works in the following way:

1. On startup, put every interface into the *Scan* phase.

2. As soon as an interface finishes its *Scan* phase, it notifies the *Handover Controller* about its scanning results (including all in-range APs).
3. The *Handover Controller* filters the received according to different *filter criteria* (see step 5 and Section 5.2.2) and sends back the filtered list to the interface.
4. If the filtered list contains any entries, the interface starts the Authentication & Association phases and notifies the *Handover Controller* about this decision.
5. Upon receiving this decision, the *Handover Controller* puts this AP on a *blocklist* used in step 3, so that no other interface can associate with the same AP.
6. When the connection between an interface and an AP times out, the interface notifies the *Handover Controller* about the loss of connection.
7. The *Handover Controller* removes this AP from the *blocklist*

This behavior ensures that the interfaces of one STA do not connect to the same AP multiple times simultaneously. Additionally, this approach provides us with a *Handover Controller* which allows us to apply additional filter criteria and always know the current state of all interfaces. The opportunity of providing additional filter criteria can be used in different ways, e.g. by excluding APs with too many associated STAs. As mentioned in our System Model in **??**, every STA has an ap_list(sta), which we use as an *allowlist* as a filter criterion in step 3.

With this approach, we are able to fulfill a handover directly to the other interface after a timeout between the STA and its current AP. In Figure 5.5b, we can see that the time without a connection now is only the timeout-phase (yellow), as we are able to skip the scan phase before performing a handover. Theoretically, our approach above allows the utilization of an arbitrary number of Wi-Fi interface, but for the rest of this work, we only consider STAs with one or two interfaces.

5.2.2 Handover Decision

Our previous approach is still a reactive handover approach, meaning it only preforms the handover procedure after a timeout of the currently associated STA. Packets transmitted during that timeout period (yellow in Figure 5.5b) are still lost. In order to further improve upon our previous approach, we extend it by providing a proactive handover decision.

Location-based Handover

A simple idea to proactively handover to the next AP is to immediately perform a handover after association. This however is problematic in the following scenarios:

- The STA only barely passes the transmission range of another AP, associates and times out shortly after.
- The STA stays in the transmission range of the current AP.

Algorithm 5.2 Location-based Proactive Handover Algorithm

```
 1: procedure CheckHandover(this: HandoverController)
 2:     bestInterface ← this.reponsibleInterface
 3:     bestDist ← dist(this, bestInterface.getAssociatedAp())
 4:     for each otherInterface in (this.interfaces \ this.responsibleInterface) do
 5:         if otherInterface.isAssociated() then
 6:             otherDist ← dist(this, otherInterface.getAssociatedAp())
 7:             if otherDist < bestDist then
 8:                 bestInterface, bestDist ← otherInterface, otherDist
 9:             end if
10:         end if
11:         if bestInterface ≠ this.responsibleInterface then
12:             InitiateHandover(bestInterface)
13:         end if
14:     end for
15: end procedure
```

In both scenarios, we perform an unnecessary handover possibly resulting in lost packets and a higher jitter. Additionally, in more sophisticated radio models, where the link quality degrades upon distance, a too early handover reflects to a handover on a worse link. To this end, we propose a proactive handover based on the signal strength/location of the involved APs. As mentioned in Section 4.1.1, we assume a unit disk radio model without any additional obstacles. With this model, the distance between a STA and AP directly correlates to the received signal strength (i.e. the closer the STA to an AP the stronger the received signal and vice versa), thus we refer to our approach as a location-based proactive handover approach in the following.

For this proactive handover approach we utilize the knowledge of our *Handover Controller* and extend it with additional functionality. In order to allow for a location-based handover our *Handover Controller* needs to stay updated about its current STA's location. To this end, every interface notifies the *Handover Controller* upon the reception of a *Beacon* frame including its signal strength. Based on these beacon frames, it calculates its position and executes the location-based Proactive Handover Algorithm as in Algorithm 5.2. This algorithm iterates over all other interface of its STA and calculates the distance to all associated APs. Among these, it selects the interface associated with the closest AP and initiates a handover, if it differs from the currently responsible interface (i.e. it only performs a handover if there is an interface with a better connection).

With this approach, we are able to hand over a STA to another AP before the association times out. As Figure 5.5c illustrates, this approach ideally allows us to perform a handover without a drop in connection and thus without the loss of frames.

Scheduled Disassociation

In Figure 5.5c, we can also see that the $wlan_0$ interface stays connected after a handover until it regularly times out. Normally, this should not lead to any problems as our location based handover algorithm always connects a STA to its closest AP. In rare scenarios, where a STA needs to perform two handovers shortly after each other (e.g. when passing an AP close to the transmission range

Destination	Source Association	Destination Association	Interface
01-00-5E-00-00-02	ap_1	ap_3	eth2
01-00-5E-00-00-02	ap_2	ap_3	eth2
01-00-5E-00-00-02	ap_1	ap_4	eth3
01-00-5E-00-00-02	ap_2	ap_4	eth3

Table 5.1: MAC forwarding table configuration for *switch* in Figure 5.4.

boundary) it might be beneficial to disconnect interfaces early in order to have a free interface for scanning. We want to experiment if this leads to an even better handover performance, hence we propose a new algorithm to schedule early disassociations. This new algorithm works similar to our *CheckHandover* procedure in Algorithm 5.2, but instead of searching for the best interface and initiating a handover it searches for the worst interface (e.g. the one furthest away from its associated AP) and instructs it to disconnect. This procedure is only performed if all interfaces of an STA are associated (i.e. we aim to have exactly one unassociated interface). Upon receiving the disconnect instruction, the interface ignores incoming beacons from its associated AP. We do not immediately disconnect an interface, as some frames might still utilize the old route, until all MAC forwarding tables are updated. Instead, this approach simulates a timeout by ignoring the incoming beacon frames (i.e. not restarting the beacon timeout). Figure 5.5d shows this behavior, with a timeout directly after the handover and the start of a new scan phase afterwards.

By utilizing only this approach, interfaces would immediately start re-associating when the AP is still in range after the simulated timeout. To this end, we extend our algorithm with an additional functionality. Instead of ignoring beacon frames completely, we still use them to calculate the distance between STAs and APs. For interfaces with a scheduled disconnect, we keep track of the last distance before the association times out. When an interface tries to associate with the same AP again, we use the stored distance as a filter criterion in step 3 in our *Handover Controller* and only allow an association if the STA is closer to the AP than before the timeout. To further optimize this approach, we also recalculate the distance after each received beacon frame and cancel a scheduled disconnect as soon as the distance reduces again.

5.2.3 MAC Forwarding Table Configuration

Our previous approaches allow us to perform a smooth handover between APs. In order to deliver time-sensitive data, it is necessary to configure the forwarding tables along the new route properly. An entry in a MAC forwarding table usually consists of two parts: The destination MAC-address and the outgoing network interface (e.g. *eth0*). Our ILP provides us with all possible routes that a stream can utilize, hence we do not need a fully dynamic configuration approach. Instead, we can store the possible routes in a different format on the network devices and derive MAC table entries from it. Table 5.1 shows an example MAC forwarding table for the scenario in Figure 5.4. In this example, the outgoing interface only depends on the current association of the destination STA. But as our ILP treats these scenarios as separate streams it might schedule them along different paths, hence we need to take both, the association of the source and target STA into account.

In a real-world scenario, deriving actual MAC table forwarding entries from our example configuration can be achieved in different ways, for example:

- Broadcasting a new association to the network and deriving the MAC table entries based on these broadcast messages.
- Having a centralized controller that instructs network devices to update their MAC table using unicast messages.
- Notifying network devices about new associations along the new path (either forward or backward, depending on which device associated with a new AP) and updating the MAC table accordingly.

As our work solely utilizes a simulation framework, we are able to use a centralized controller that updates MAC forwarding tables immediately after a handover. This approach is not suitable in real-world networks, but it helps us to analyze the effect of our handover approaches without additional influence by MAC forwarding table updates.

6 Implementation

In this chapter, we explain how we implement our approaches from the previous Chapter 5 in OMNeT++. As mentioned before, the INET framework for OMNeT++ already provides devices implementing the Ethernet and IEEE 802.11 WLAN standard. In this section, we present how we extend the functionality of these devices to support our approaches from the previous chapter. We start by explaining our modifications of wireless devices in INET in order to support the TSN standards. Following that, we describe the implementation of our handover controller in OMNeT++ and finally we illustrate how we configure and update the MAC forwarding tables of switches and APs.

6.1 Wireless TSN

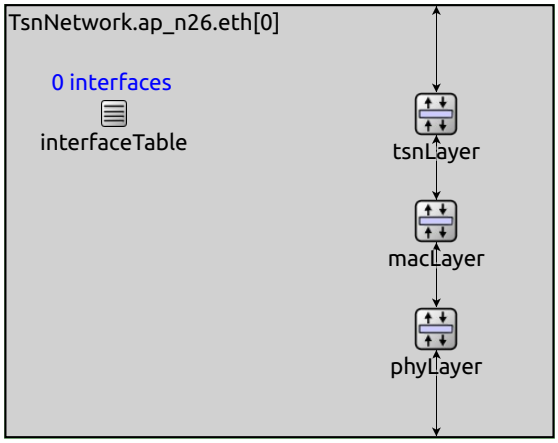
As described in Section 2.5, the INET framework already supports TSN functionality in Ethernet devices. However, these implementations are incompatible to the implementation of APs and STAs. In this section, we explain how we establish compatibility among the different devices and how we add TSN functionality to APs and STAs respectively.

The main problem prohibiting a simultaneous use of TSN Ethernet devices and non-TSN devices in INET is that INET's TSN devices implement Ethernet and TSN functionality in a more modular way than their non-TSN counterparts. These different implementations are incompatible among each other and as there are no wireless interfaces following the modular implementation approach of the wired TSN devices, we cannot simply equip APs and STA with the modules from TSN switches and hosts. To this end, we first modified the implementation of the *ethernet* module (*EthernetEncapsulation*) in APs and STAs to behave in the same way as the modular counterpart (*EthernetLayer*) in wired devices does.

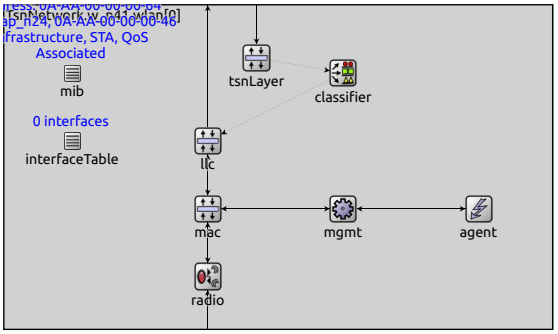
Our modified *EthernetEncapsulation* does not work when embedding the *StreamIdentifier* and *StreamCoder* modules introduced in Section 2.5 directly into STAs and APs. Therefore, we copy the module description of INET's Ethernet interface (*LayeredEthernetInerface*) and wireless interface (*Ieee80211Interface*) and extend it with a new submodule implementing TSN specific functionality. We call these submodules *EthTsnLayer* (Figure 6.1c) and *WlanTsnLayer* (Figure 6.1d), respectively.

Figure 6.1a shows our modified Ethernet interface. As explained in Section 2.5, its *macLayer* already implements the TAS. Thus, we only need to embed the *StreamIdentifier* and *StreamCoder* in our *EthTsnLayer*.

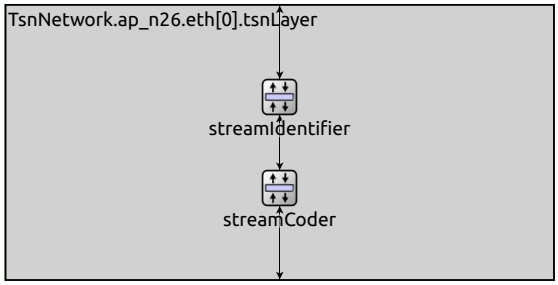
Wireless interfaces in INET, however, do not provide a TAS natively. To this end, our *WlanTsnLayer* additionally has a submodule *queue* of the *Ieee8021qTimeAwareShaper* type as described in Section 2.5. In order to pull and forward frames from the queue to the next submodules, we



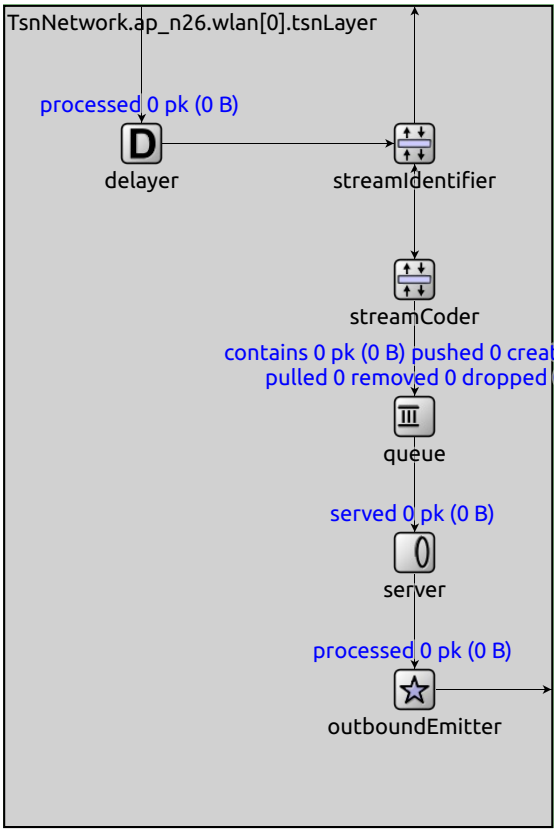
(a) Ethernet interface with TSN functionality.



(b) Wireless interface with TSN functionality.



(c) TSN layer for Ethernet interfaces in APs.



(d) TSN layer for wireless interfaces in APs and STAs.

Figure 6.1: Ethernet and wireless interfaces with TSN functionality

embed two more submodules, namely *server* and *outboundEmitter*. Figure 6.1b shows, how the *WlanTsnLayer* is embedded within our modified wireless interface. The *classifier* following our *WlanTsnLayer* is responsible for mapping PCP values to the ACs (see Section 2.2.2). INET implements the different coordination functions including the HCF with different ACs in the *mac* submodule.

6.2 Handover Controller

With the previous implementations, we enabled TSN functionality for APs and STAs in INET. In this section, we explain how we enable the usage of multiple wireless interfaces in STA and present our OMNeT++ implementation of the handover controller proposed in Section 5.2.

The implementations of STAs in INET natively support employing multiple wireless interfaces at once. However, they do not directly interact with each other, which means they might associate with the same AP simultaneously and by default upper layers only instruct one wireless interface to transmit frames. To this end, we implement one *Handover Controller* in every STA. This *Handover Controller* interacts with the *mgmt* and *agent* submodules of all wireless interfaces belonging to the same STA.

Before we are able to implement our handover controller, we first need to modify INET's implementation of the *mgmt* and *agent*. One of these changes is fixing a bug in INET's implementation, where the *agent* of one interface reacts on the *beaconLost* signal of the *mgmt* and vice-versa. Furthermore, we add a dynamically configurable *allowlist* and *blocklist* into the *agent* of every wireless interface.

As our handover controller always needs to be informed about the state of the interfaces, we extended the *agent* and *mgmt* with additional signals to which our handover controller subscribes:

disassociatedSignal is emitted when the interface disassociated from an AP, for example caused by a timeout.

associatedSignal is emitted when the interface finishes the association with an AP.

associationStartedSignal is emitted when the interface starts the association procedure with an AP.

associationFailedSignal is emitted when the interface tried to associate with an AP but the association procedure failed (e.g. caused by a timeout)

Upon receiving any of these signals, our basic handover controller (without any further improvements) performs the following duties:

- When receiving an *associatedSignal* or *associationStartedSignal*, the handover controller adds the respective AP to the *blocklist* of all other interfaces, so that no other interface can connect to the same AP.
- When receiving a *disassociatedSignal* or *associationFailedSignal*, it removes the respective AP from the *blocklist* of all other interfaces.

Device	Stream	Source Association	Destination Association	Next Hop
ap_1	s_0	ap_1	ap_3	*switch*
ap_1	s_0	ap_1	ap_4	*switch*
ap_2	s_0	ap_2	ap_3	*switch*
ap_2	s_0	ap_2	ap_4	*switch*
switch	s_0	ap_1	ap_3	ap_3
switch	s_0	ap_2	ap_3	ap_3
switch	s_0	ap_1	ap_4	ap_4
switch	s_0	ap_2	ap_4	ap_4
ap_3	s_0	ap_1	ap_3	v_{dst}
ap_3	s_0	ap_2	ap_3	v_{dst}
ap_4	s_0	ap_3	ap_4	v_{dst}
ap_4	s_0	ap_4	ap_4	v_{dst}

Table 6.1: Configuration of our MAC table forwarding table configurator for Figure 5.4.

- When receiving a *associatedSignal* or *disassociatedSignal*, it initiates a handover if necessary (i.e. the newly associated interfaces is the only interface with an active association or the disassociated interface was the responsible interface.)

With our proactive handover approach, the handover controller additionally performs a regular handover check based on the distances to all associated APs (based on their last beacon frame). This handover check compares the distances of all associated APs and initiates handover if the closest AP is not the AP associated with the currently responsible interface. After that and if scheduled disassociations are enabled, our handover controller searches for the interface with the furthest AP and schedules a disconnect. We implement the different functionality of scheduled disassociations as described in Section 5.2.2 into the *mgmt* and *agent* modules directly.

When our handover controller decides to initiate a handover, it performs the following steps:

1. Instruct the *ipv4* module of its corresponding STA to forward future frames to the newly selected responsible wireless interface.
2. Emit a *handoverSignal*. We explain the usage of this signal in the next section about the implementation of our MAC forwarding table configurator.

6.3 Forwarding Table Configurator

In the last section, we explained our handover controller implementation in OMNeT++ and mentioned that it emits a *handoverSignal* upon performing a handover. As with every handover of a STA the routes of all streams originating or ending at this STA ends, we need to update the MAC forwarding tables of all switches and APs along this route in our network.

As mentioned in Section 5.2.3, we use a simplified approach in our work which updates all MAC forwarding tables immediately. In order to achieve this behavior, we introduce a network-wide module, namely *MAC Forwarding Table Configurator*. We provide this module with the MAC table

configuration for each module based on the current associations from source and destination nodes of the streams. Table 6.1 shows the configuration for the example topology introduced earlier in Figure 5.4. Upon startup, our MAC table forwarding configurator subscribes to the *handoverSignal* of all STAs and performs the following steps when receiving one of these signals:

- Stores the association of the STA that initiated the handover in an internal dictionary.
- For all streams originating or ending at the respective STA, find the matching entries in the configuration table (based on *Source Association* and/or *Destination Association*).
- If there are multiple possible entries for one device find the correct entry based on the association of the other STA of the stream by looking up its association in the dictionary. This only applies if both, source and destination, are STA, as there is only one entry for wired hosts.
- Get the destination MAC address $addr$ based on the stream id.
- For every device d with a matching entry, perform the following action based on the device type and next hop:
  - If d is a switch, find the ID id of the Ethernet interface that connects d with the next hop and add/replace the following entry in the MAC forwarding table of d: $(addr, eth[id])$
  - If d is an AP and next hop is a switch, add/replace the following entry in the MAC forwarding table of d: $(addr, eth[0])$
  - If d is an AP and next hop is a STA (i.e. the destination), add/replace the following entry in the MAC forwarding table of d: $(addr, wlan[0])$

With this implementation, we ensure that all MAC forwarding tables along the new route are updated after every handover. As we only override conflicting entries but do not delete outdated entries on the old route, packets which are still in transmission on the old route can still reach their desired destination.

7 Evaluation

In Chapter 5 we proposed different approaches in order to employ TSN in wireless environments. It is left to show if and to which extent these approaches improve our desired metrics, namely reliability, delay and jitter. We start this chapter with an introduction to our evaluation setup including the abbreviations we use for the evaluation and the generation of our simulations. Following that, we analyze the results of our ILP model regarding schedulability and the influence of our objectives onto delay and jitter. We then continue with an analysis of the OMNeT++ simulation. First, we analyze the influence of our device configuration (i.e. number of wireless interfaces and properties of the handover controller) handovers itself onto packet loss. Finally, we analyze how these device configurations and the ILP objective effect the delay and jitter in the simulation.

7.1 Evaluation Setup

In this section we explain our evaluation setup in detail. We start with an introduction to the abbreviations for parameters of our simulation in order to use them in figures. Following that, we explain how we generate our topologies and streamsets. We then explain, how we solve the ILP models deriving from these scenarios. Finally, we introduce our different configuration properties for simulating the scenarios in the OMNeT++ simulator.

7.1.1 Abbreviations

In order to distinguish between our different simulation scenarios in our analysis, we introduce an abbreviation scheme in Table 7.1. This scheme allows us to uniquely identify between different topology, streamset or simulation configurations.

We can use this naming scheme to uniquely identify a specific simulation, for example top_s30_a1-str_h1.5_c25-sim_1 describes a simulation in a topology with approximately 30 switches in the network, an average of one STA per AP on a streamset with an average of 1.5 streams per host, a cycle time of 25 ms and one Wi-Fi interface per wireless host.

7.1.2 Topology Generation

Regarding the topology, there are two main parameters that are interesting to research. The first one is the influence of the total topology size and the second one is the average amount of wireless nodes connected to an AP. To this end, we generate multiple random topologies varying in these parameters.

Abbreviation	Description
top	This prefix indicates that the following parameters describe our topology.
_sX	X is the approximate number of switches in the respective topology.
_aY	Y is the average number of STAs per AP in the respective topology.
	Example: top_s20_a0.5 describes a topology with 20 switches and 0.5 STAs per AP
str	This prefix indicates that the following parameters describe our streamset.
_hX	X is the average number of outgoing streams per host.
_cY	Y is the cycle time of the respective streamset in ms.
	Example: str_2_5 describes a streamset with an average of 2 outgoing streams per host and a cycle time of 5 ms.
sim	This prefix indicates that the following parameters describe our simulation configuration.
_1	This simulation has one wireless interface per STA.
_2	This simulation has two wireless interfaces per STA, if not specific otherwise by any following parameters, it does not utilize the proactive handover approach.
_a	Utilizes the proactive handover approach without any further optimizations.
_b	Utilizes all optimizations above, additionally schedules disassociations.
_c	Utilizes all optimizations above, additionally checks the distance before allowing reconnections to the same AP.
_d	Utilizes all optimizations above, additionally cancels scheduled disassociations, if the STA moves closer to the AP again.
	Example: sim_2_c describes a simulation with 2 wireless interfaces per STA utilizing the proactive handover approach, with scheduled disassociations and checking distance before reconnecting to the same AP.
obj	This prefix indicates that the following parameters describe our ILP objective
_e2e	Our ILP minimizes the sum of end to end delays as In Equation (5.5)
_e2e+jitter	Our ILP minimizes the sum of end to end delays and sum of jitters (with $w_{\text{jitter}} = 3$ and $w_{\text{e2e}} = 1$)

Table 7.1: Abbreviation for optimization combinations.

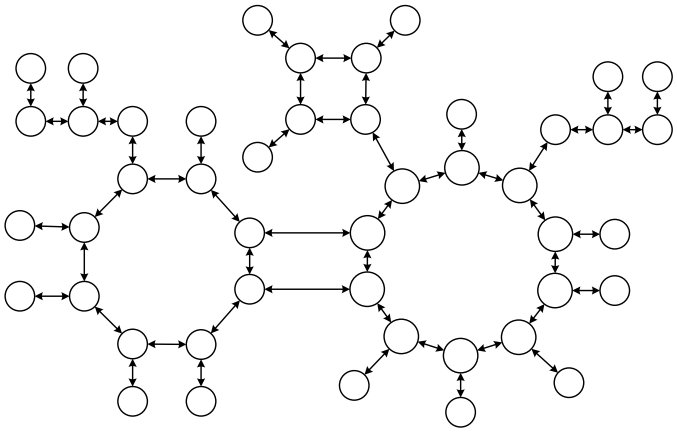


Figure 7.1: Factory Automation Topology [HDHK18].

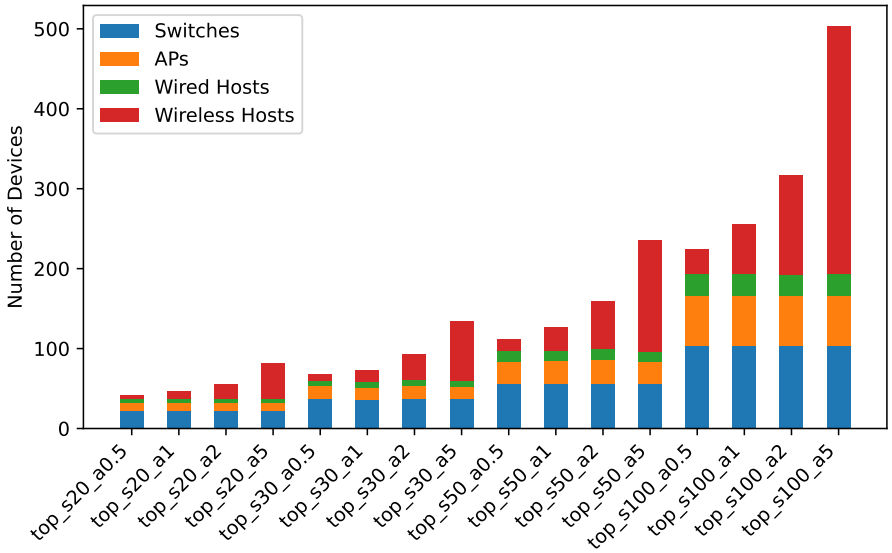


Figure 7.2: Number of devices per topology.

Our topology generation algorithm generates topologies based on factory automation networks as in Figure 7.1. These topologies resemble real-world factory networks having a tree-like structure with ring topologies close to the root and ring or line topologies closer to the leafs. Our topology generation procedure works as follows:

1. Generate a random factory automation network with approximately n switches. ($n \in \{20, 30, 50, 100\}$)
2. For every switch with two or less outgoing edges, attach an AP with a probability of 70 % and a wired host with a probability of 30 %. We decided to use this partitioning as a trade-off between a high enough AP density and a sufficient number of wired hosts.
3. Generate m wireless hosts, with $m = \text{num\_aps} \cdot \text{hosts\_per\_ap}$ ($\text{hosts\_per\_ap} \in \{0.5, 1, 2, 5\}$)

OS	Ubuntu 22.04.2 LTS
Kernel	GNU/Linux 5.15.0-60-generic x86_64
CPU	4x Intel(R) Xeon(R) CPU E7-4850 v4 @ 2.10 GHz (4x16 Cores/Threads)
Memory	1 TB

Table 7.2: Hardware for ILP execution.

4. Calculate coordinates for all devices using a spring-based layout.
5. Calculate the transmission ranges of APs in such a way that every AP has at least one AP with an overlapping transmission range.
6. For every AP generate a list of APs with an overlapping transmission range.
7. Connect every wireless STA to a random AP. Calculate a list of at most 5 AP utilizing the in_range(ap) list from the previous step, which the STA traverses later (Note: The detailed moving path is not determined yet).
8. Calculate wireless channels for all APs according to the Vertex Coloring Algorithm in Algorithm 2.1.

This generation approach provides us with a total of 16 different topologies with different sizes for the backbone topology and different numbers of STAs per AP. Figure 7.2 shows the amount of devices per generated topology.

7.1.3 Streamset Generation

For each of the topologies generated in the previous step, we then generate a number of different streamsets. The two main points we aim to research are the number of streams and the cycle time of these. Hence, we generate streamsets varying these parameters.

Our streamset generation selects two random hosts in the network (either wireless or wired) and generates with a random size between 16 B and 512 B with one of these nodes being the source and one being the destination. We generate streamsets with 0.25, 0.5, 1, 1.5, 2 and 5 outgoing streams per host. The cycle time is either 1 ms, 5 ms, 25 ms or 125 ms This provides us with 24 steamsets per topology from the previous step.

7.1.4 ILP Execution

For each combination of topology and streamsets from the steps above, we generate our ILP model as described in Section 5.1 and execute it with two different objectives. Once minimizing the end-to-end delay with Objective (5.5) and once minimizing a the combination of jitter and end-to-end delay with Objective (5.9). For the combined objective we use the weights $w_{\text{jitter}} = 3$ and $w_{\text{e2e}} = 1$ as we want to prioritize jitter but prevent unnecessary long buffering at intermediate switches and APs.

We execute the ILP model on the hardware as described in Table 7.2. Additionally, we provide a timeout of 1,800 s, after which we stop the ILP execution without a result. We limit the number of concurrent Gurobi instances to 16, so that each Gurobi instance can utilize at least 4 cores simultaneously.

7.1.5 Simulation Configurations

In the final step, we generate different OMNeT++ simulation configurations for the scenarios generated above. Our main goal is to analyze the effect of our different handover optimization approaches. To this end, we generate the following OMNeT++ simulation configurations:

- Every STA has one wireless interface.
- Every STA has two wireless interfaces with reactive handover.
- Every STA has two wireless interfaces with proactive handover.
- Every STA has two wireless interfaces with proactive handover. Scheduled disassociations are enabled.
- Every STA has two wireless interfaces with proactive handover. Scheduled disassociations and distance checking before reconnections are enabled.
- Every STA has two wireless interfaces with proactive handover. Scheduled disassociations, distance checking before reconnections and the possibility of canceling scheduled disassociations are enabled.

Only in this step, we generate a random movement path along the APs a STA traverses. The movement path is generated in the following way:

1. Start at a random position in range of the first AP.
2. Stay there for a random time between 1 s and 5 s.
3. Calculate a new random position within the range of the next AP.
4. Calculate a constant movement speed, so that moving to the new destination takes a random time between 1 s and 5 s.
5. If there are is an AP left in the traversal list, continue at step 2.

We set the simulation time to 120 s. During that time every STA moves back and forth on its random path.

Overall, this provides us with 6 simulation configurations per ILP model.

7.2 ILP Results

In the first step of our evaluation, we analyze the output of our ILP model. We start with an analysis of the schedulability results in regard to topology size and streamset. Thereafter, we evaluate the influence of our different objective functions on jitter and delay in an optimal network.

7.2.1 Schedulability

In this section we analyze the influence of the topology size and streamset parameters on the schedulability of our streamsets. Figure 7.3 shows the result status of solving our ILP model in Gurobi. Once grouped by the underlying topology and once grouped by the parameters from the streamset generation.

As the complexity of an ILP grows with the number of variables and constraints, we expect a reduced schedulability for bigger topologies. Figure 7.3a confirms this assumption, as the percentage of solved ILP models reduces with a growing amount of devices in the network. The figure also shows that the number of wireless devices in a network has a greater effect on schedulability than the total number of devices. For example, our topology *top_s20_a2* has less devices than *text_s50_a0.5* in total, but a greater number of wireless devices (compare Figure 7.2). In regard to schedulability however, topology *top_s20_a2* has less successful ILP executions and more ILP models proven to be infeasible compared to *text_s50_a0.5*. This is based on the following to results:

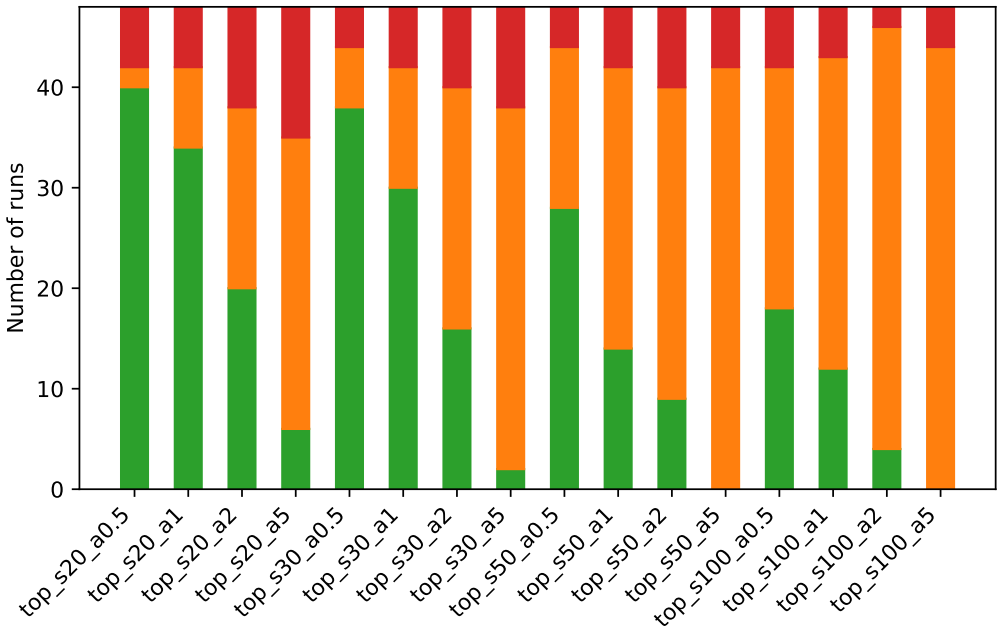
1. With a higher ratio of wireless devices to wired devices, a higher percentage of streams use wireless edges. Wireless edges, however, make scheduling streams more difficult in two ways. Firstly, wireless edges in our topologies have a lower link speed and thus a higher transmission delay and secondly, solving conflicts on the wireless transmission medium involves more constraints (see Constraint (5.4)) than solving conflicts for wired connections (see Constraint (5.3)).
2. Wireless devices can roam between different APs. As our approach treats every possible AP association as a separate stream, calculating a schedule for one wireless device actually means calculating multiple schedules for every possible association.

As Figure 7.3b shows, the schedulability also depends on the number of streams and their cycle time. Generally, more streams and a small cycle time reduce the schedulability. Especially streamsets with a cycle time of 1 ms are often infeasible. Even for streamsets with only one outgoing stream per host, approximately 50 % of the streamsets are proven infeasible when using a cycle time of 1 ms (*str_h1_c1*). This shows that our current approach lacks the ability of scheduling streams with low cycle times of around 1 ms in a wireless environment. We assume the reason for this is the over-reservation of bandwidth, as we reserve bandwidth for every possible AP association even though a STA is only connected to one AP at a time.

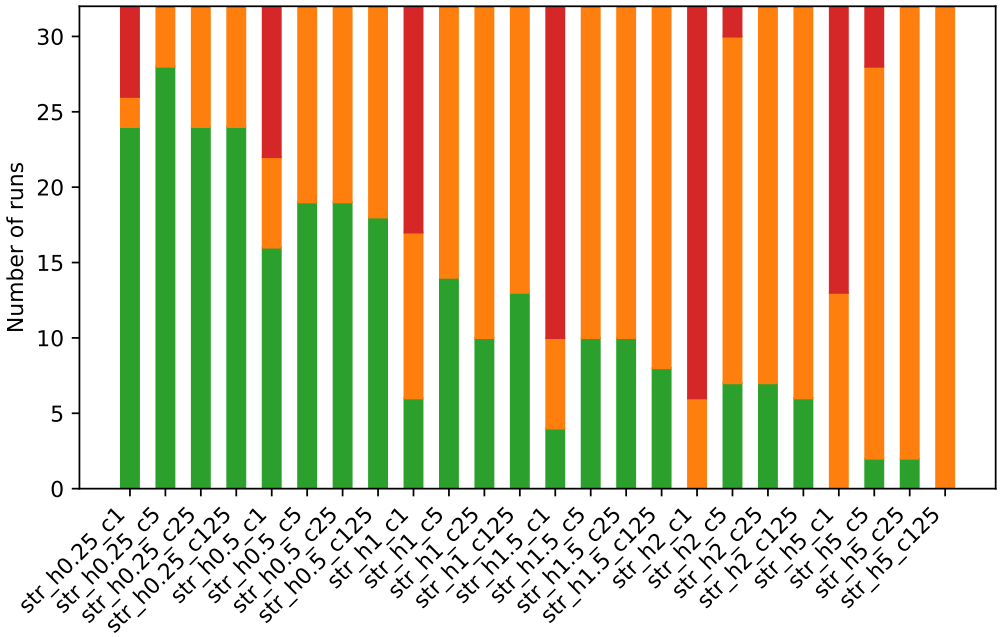
Furthermore, Figure 7.3b shows that high cycle times of 125 ms lead to timeouts of Gurobi more often than streamsets with the same number of streams but lower cycle times. This most likely comes from the fact that a greater cycle time leads to greater bounds for the scheduling variables and thus increases the search-space of the ILP-solver.

7.2.2 Objectives

As mentioned before, we execute every ILP model with two different objectives. In this section we analyze the influence of our two different objectives on the delay and jitter of the calculated schedule.



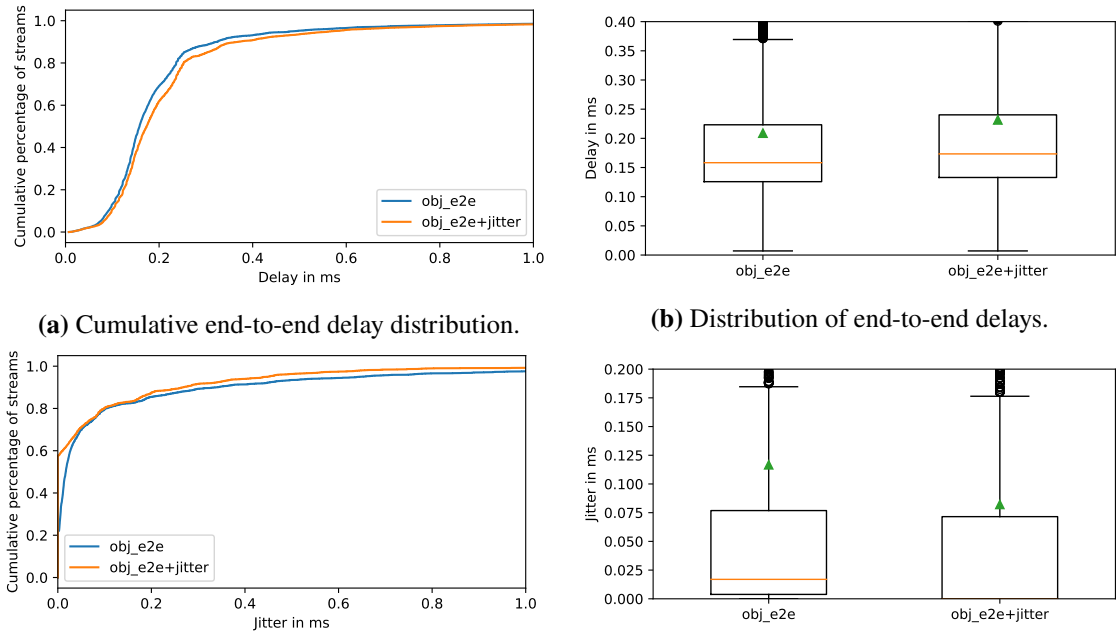
(a) Schedulability of different topologies.



(b) Schedulability of different streamsets.

(c) Graph legend.

Figure 7.3: Schedulability by topology size and streamset parameters



(a) Cumulative end-to-end delay distribution.

(b) Distribution of end-to-end delays.

(c) Cumulative jitter distribution.

(d) Distribution of jitter values.

Figure 7.4: Comparison of jitter and end-to-end delay from solutions with different objectives.

As Figure 7.4a shows, the end-to-end delay of streams only varies a little between both objectives. The biggest difference occurs at around 0.2 ms, as the objective solely optimizing the end-to-end delay achieves to schedule approximately 70 % within and end-to-end delay of 0.2 ms while the combined objective achieves around 60 %. Both objectives, however, are able to schedule almost all streams within a cycle time of 0.6 ms without a significant difference between them. Figure 7.4b shows us that the medians of end-to-end delays is almost equal for both objectives, but the mean for the combined objective is slightly higher. This indicates that both objectives schedule most of the streams with a low end-to-end delay, but with outliers of greater values for the combined objective (i.e. in both cases only a few streams have a high end-to-end delay, but these are higher in the case of the combined objective).

The comparison of the jitter shows a greater difference between both constraints. In Figure 7.4c, we can see that the combined objective is schedules almost 60 % of streams without any jitter while the end-to-end objective achieves this for approximately 20 % of the streams. The objectives perform similar for a jitter of around 0.05 % and above, with the combined objective performing slightly better continuously. The median jitter value for the combined objective is significantly lower as the median for the end-to-end objective, as Figure 7.4d shows, which indicates that the percentage of streams with a lower jitter is significantly higher for the combined objective. Additionally, the boxplot for the end-to-end objective extends further up with a significantly higher mean, indicating higher jitter values in the upper part of the distribution.

Result Status	
Successful	1136 (20 excluded)
Failure: Port Binding	188
Failure: Access Granted	152
Failure: Others	55
Crash	95

Table 7.3: Status of OMNeT++ runs.

Overall, the comparison of end-to-end delays and jitters shows that the combined objective produces schedules with a generally lower while only slightly sacrificing the end-to-end delay. However, it is important to note that these are only theoretical result for an optimal network without lost packets or randomness (e.g. by the back-off algorithm) and evaluation on a simulated or real network may result in different findings. We analyze this in the upcoming Section 7.3 about simulation results.

7.3 Simulation Results

In this section, we analyze our OMNeT++ simulation results. As Table 7.3 shows, not all executed simulations were successful, due to bugs in our and INET's implementation:

Successful Most our simulations executed successfully. However, a bug in the implementation of our handover controller allowed two wireless interfaces of one STA to be connected the same AP twice. This leads to a double reception of packets. To this end, we exclude these 20 runs in our further evaluation.

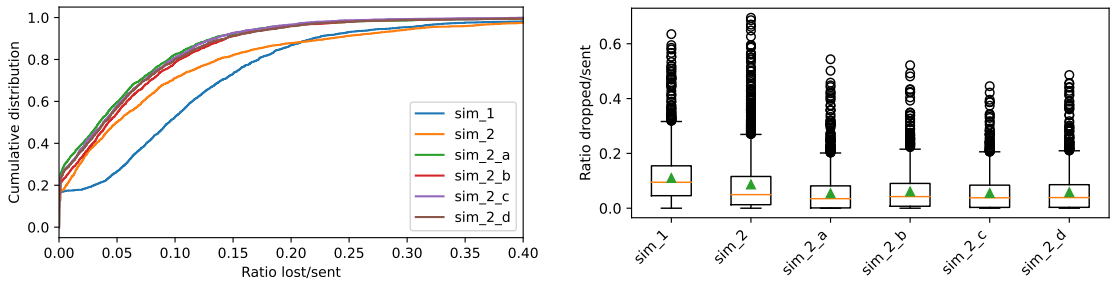
Port Binding The most common bug occurs during startup of the simulation when apps try to bind their port in INET's UDP module. When apps in INET subscribe to a multicast group, INET selects an unbound port and bind it to the application. After that, applications send a binding message for specific ports. If one of these ports is already bound in the previous step, the simulation fails with an error.

Access Granted In rare cases, the HCF module in INET grant access to the wireless channel, while another frame sequence is running. This occurs, when an AP tries to send a data frame from the *AC_VI* queue, while it is waiting for an *ACK* frame of the authentication procedure with another STA (in the *AC_VO* queue).

Others Failed simulations with other errors

Crash In some cases, the OMNeT++ simulation failed without any error message. We did not further analyze these cases.

The further evaluation is structured as follows: First, we analyze the influence of our different simulation configurations (i.e. the number of wireless interfaces and configuration of the handover controller) and handovers on lost packets. After that, we present how the simulation configurations influence the delay and jitter packets. Finally, we evaluate how our different ILP objectives perform in the OMNeT++ simulation.



(a) Cumulative distribution of lost packets.

(b) Distribution of lost packets.

Figure 7.5: Percentage of lost packets in relation to simulation configurations.

7.3.1 Lost Packets

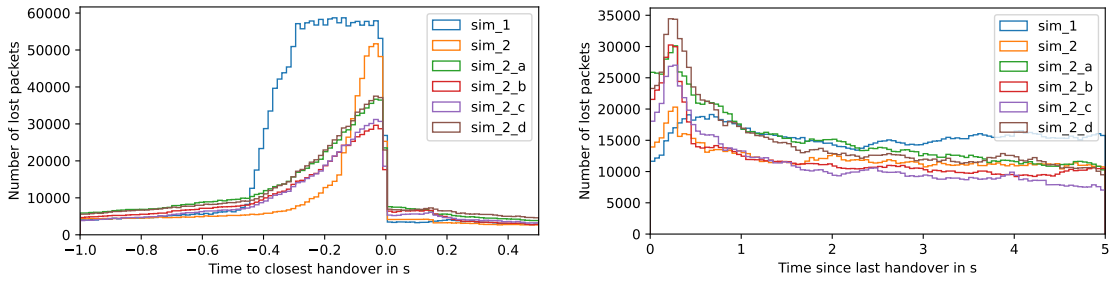
As reliability is an important metric in TSN, it is crucial to analyze the cause of lost packets in our network. In the following, we first analyze how the percentage of lost packets relate to our different simulation configurations. After that, we take a look at the influence of handovers on packet loss.

Simulation Configuration

In this section, we analyze the benefits or equipping STA with two wireless interfaces and performing proactive handovers. The main metric we are interested in is the percentage of lost packets per stream.

Figure 7.5a shows that, regardless of the configuration, approximately 20 % of streams experience almost no packet loss. However, in the field with a small amount of lost packets, STAs with two wireless interfaces perform significantly better. For example, only around 20 % of streams on STAs with one wireless interface have a packet loss of less than 5 %. With two wireless interfaces however, this already applies to 50 % to 60 % of streams. We can also see that the proactive handover approach outperforms the reactive handover approach on STAs with two wireless interfaces. This is the most visible at a packet loss of around 20 %. Here, the reactive handover approach on STAs with one and two wireless devices performs similar with around 90 % of streams. With the proactive handover approach however, almost all streams have a packet loss of 20 % or less.

With Figure 7.5b, we can confirm our previous findings. It clearly shows that two wireless interfaces on a STA leads to a lower packet loss on average. Also, as both quartiles are lower for two interfaces, we can see that two wireless interfaces leads to more streams with a low packet loss. When comparing the proactive handover and reactive handover approaches on STAs with two wireless interfaces, we can also see that the proactive handover leads to a lower average of lost packets. The lower quartiles of both approaches is approximately the same. However, the upper quartile of the proactive handover approach is lower compared to reactive handover approach. This indicates that the proactive handover approach leads to fewer streams with a high packet loss. Both figures do not show significant improvements for scheduled disassociation.



(a) Lost packets in relation to the closest handover. **(b)** Lost packets in relation to the previous handover.

Figure 7.6: Lost packets in relation to the handover times of the source and destination STA of a stream.

Handover

To further analyze the reason for the results above, this section shows how exactly handovers influence the packet loss with the different configurations. To this end, we calculate how close to a handover the packet losses occur. Once, by calculating the difference to the closest handover (before or after a packet loss) and once only calculating the time difference of the handover preceding the lost packet (Figure 7.6a and Figure 7.6b respectively).

As Figure 7.6a shows, the greatest packet loss occurs shortly before handovers. This effect is the greatest for STAs with one wireless interface This is to be expected, as a handover with one wireless interface only occurs after a timeout and all packets from the start of the timeout until a new association are lost. For two wireless interfaces, the reactive handover approach still suffers from the timeout. However, if the association procedure on the second interface is already finished, the handover can be performed immediately. This explains the spike shortly before a handover due to the timeout phase. The spike is lower for the proactive handover approach, as the timeout phase may also be skipped. Overall, this confirms our previous findings that two wireless interfaces and a proactive handover approach reduce the effect of the handover onto lost packets.

In Figure 7.6b, we can see that the amount of lost packets is higher directly after a handover. There are two main reasons that explain this effect:

1. For our calculation, we take the handover times from the source and target STA into account. A lost packet on the source STA only comes into effect at the expected arrival time at the destination, which is shortly after a handover.

2. Even though we update the MAC forwarding tables immediately, frames might still be in transmission on the old route and therefore don't reach the destination. We assume that this has a greater effect in a realistic scenario without an instant MAC forwarding table update.

Furthermore, Figure 7.6b also shows us that the spike is greater for the proactive handover approach. A similar behavior occurs in Figure 7.6a, where the lost packet count starts to grow faster before a handover than the reactive handover approach. To understand this behavior, it is important to note that our distribution only shows the total number of lost packets without normalizing them by the amount of handovers. Figure 7.7 illustrates that the number of handovers for the reactive handover approach with one or two wireless interfaces has a similar amount of handovers. When employing the proactive handover approach however, the number of handovers increases, as the

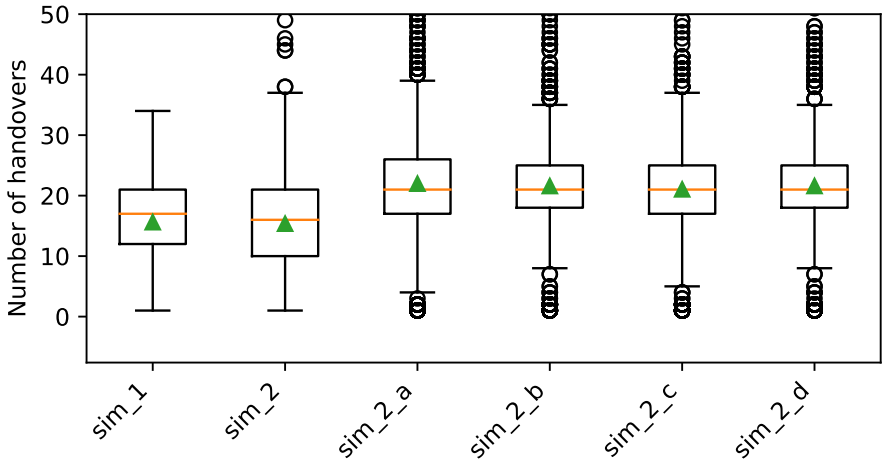


Figure 7.7: Number of handovers by simulation configurations.

average number of handovers as well as the lower quartile are significantly higher. This explains the behavior above, as a greater number of handovers in the same time frame (i.e. 120 s simulation time) leads to a greater absolute number of lost packets around handovers.

Summarized, Figure 7.6 shows that the effect of handovers on lost packets shrinks when employing a second wireless interface and using the proactive handover approach. This confirms our findings form the previous section.

7.3.2 Delay & Jitter

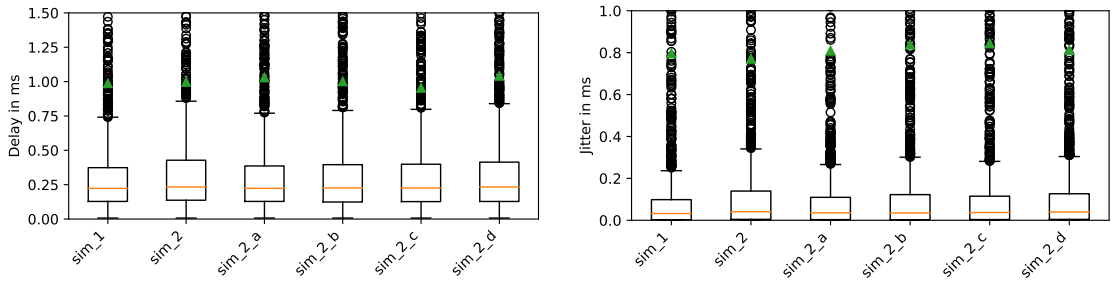
Delay and jitter are two other important metrics besides the reliability. In this section we research the influence how these metrics depend on various factors. We first analyze, if our simulation configurations from above have an influence on the delay and jitter. Following that, we evaluate if the theoretical findings of our ILP objectives also apply to the simulation.

Simulation Configuration

In the section above, we showed that our different configurations of wireless devices significantly influence the reliability of our network. It is left to analyze if the same applies for delay and jitter.

Figure 7.8 shows that all simulation configurations perform similar with only slight differences for both, delay and jitter We can also see that the mean (green triangle) is significantly above the median and the upper quartiles. This indicates that while most packets have a low delay a few outliers with a significantly higher delay pull up the mean.

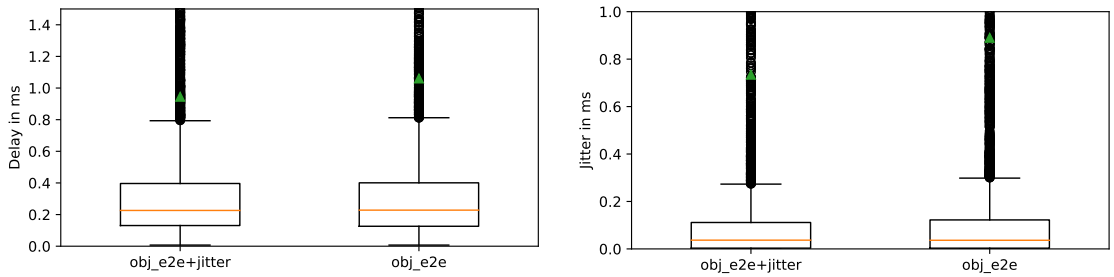
For STAs with two wireless interfaces and a reactive handover approach we can see a slight increase of the average and upper quartile of the jitter Figure 7.8b. Even though this effect is small, it most likely comes from the fact that the second wireless interfaces performs more associations in the background (which does not necessarily lead to a handover). These associations lead to a higher congestion of the wireless medium and thus more back-offs with a random time for time-sensitive



(a) Distribution of average delay.

(b) Distribution of average jitter.

Figure 7.8: Average delay and jitter per stream by different simulation configurations (without outliers).



(a) Distribution of average delay.

(b) Distribution of average jitter.

Figure 7.9: Average delay and jitter per stream by different ILP objectives (without outliers).

data frames, which results in a greater jitter. However, this effect is negligible, especially as the second wireless interface provides a significantly greater reliability as shown in the previous section.

In Figure 7.8a we can see that the medians and quartiles for the delay are similar among the different combinations. The average however, increases when employing a second wireless interface due to the same reason as described above. When employing scheduled disassociations with checks on the distance before re-associating, we can see a significant drop in the average delay. As checking the distance to an AP before an association attempt highly reduces the number of association in the network, this leads to a lower congestion of the wireless medium and thus fewer delays induced by back-offs. The fact that the median and quartiles are almost the same, but the average is significantly lower, suggest that more associations produce outliers with a greater delay.

Overall, Figure 7.8 shows that the effect of our different configurations is less significant than their effect on reliability. However, it shows that the number of ongoing associations in the network influences the delays.

Objective

Our results from Section 7.2.2 show that employing the combined objective for jitter and end-to-end delay provides slight improvements in an optimal network environment without randomness and additional frames required for the association procedure. In this section, we analyze if this objective also improves the results in our simulated network environment.

Figure 7.9a shows that there is almost no difference in delay, as the median and quartiles are approximately the same. The average delay for the combined objective is slightly lower, which might seem counter-intuitive at first. Our assumption is that optimizing for jitter employs more buffering at intermediate switches and thus distributes the streams more stretched-out within the cycle time. This leads to fewer streams with high delays induced by the back-off algorithm because of other streams.

The jitter comparison in Figure 7.9b shows a slightly reduced upper quartile for the combined objective. This suggests that streams scheduled by the combined objective generally have a slightly lower jitter. As the average is approximately the same, we assume outliers with high values introduced by handovers.

In summary, Figure 7.9 shows that the combined objective has a positive influence on delay and jitter in the simulation. However, the effect is smaller than our previous theoretical results from the ILP evaluation suggest.

8 Conclusion & Outlook

In this work, we aimed to research challenges regarding time-critical applications arising in wireless networks and to provide possible solutions for these challenges with a main focus on the handover procedure for mobile STAs. To this end, we determined relevant metrics for time-critical applications, namely reliability (i.e. packet loss), delay and jitter. We provided a modifier ILP model to support wireless links and modified the ILP generation in order to support handovers. Furthermore, we extended the INET framework for OMNeT++ with several approaches to provide wireless network devices with TSN functionality and allow for smooth handovers.

To adapt the ILP model for wireless environments, we modified the scheduling and conflict constraints in order to allow buffering at intermediate nodes, relaxing the previous no-wait fashion of the ILP. We also introduced a new set of conflict constraints, as the access to a shared wireless transmission medium requires different handling than a point-to-point Ethernet connection. In order to support mobile STAs, we employed a new ILP generation algorithm which treats multiple paths as different streams. Finally, we proposed a new objective which allows optimizing the theoretical jitter in a network.

Our main contribution for smooth handovers is a STA-specific handover controller allowing the employment of multiple wireless interfaces and a proactive location-based handover. In order to evaluate our approaches in a simulation environment, we extended the implementation of STAs in INET with our handover controller and TSN specific functionality, such as a TAS.

The evaluation of our approaches show that the mobility of STAs increases the complexity of the ILP model significantly, as it reserves bandwidth for multiple possible paths, even though only one path is used at a time. However, the results of our ILP objectives show that we are able to theoretically improve the jitter in an optimal network environment. Furthermore, our results show an increase in reliability when introducing a second wireless interface and using a proactive handover, with a negligible influence on delay and jitter. The results also indicate a slight improvement when utilizing an objective aiming to reduce the jitter. However, effects are small due to the induced randomness of the back-off algorithm.

Outlook

While our approaches above already show significant improvements in some fields, they also indicate possibilities to improve upon them in future work.

Firstly, while our ILP is able to model the basic requirements of a wireless environment it suffers from a low schedulability and high runtimes due to over-reserving channels. Future work might improve upon this, for example by making more assumptions regarding the maximum amount of connected devices to an AP and not reserving the wireless channel for every stream solely.

Furthermore, our ILP model does not take other frames on the wireless transmission medium, especially beacon frames and frames for the association procedure, into account. Scheduling time-slots for these frames might reduce the randomness introduced by the back-off algorithm.

Another approach to reduce the influence of the back-off algorithm is to utilize a different coordination function, such as the Hybrid Coordination Function (HCF) with HCF Controlled Channel Access (HCCA). Future work might extend INET with a HCCA implementation and analyze the influence on the relevant metrics. Finally, the unit disk radio model we used in our work is a highly simplified model compared to real-world environments. In order to get results similar to a real-world environment, future work might evaluate our approaches using a more realistic radio model.

Bibliography

[ACB19] T. Adame, M. Carrascosa, B. Bellalta. "Time-Sensitive Networking in IEEE 802.11be: On the Way to Low-latency WiFi 7". In: *CoRR* abs/1912.06086 (2019). arXiv: 1912.06086. URL: http://arxiv.org/abs/1912.06086 (cit. on p. 37).

[AH08] H. Ahmed, H. Hassanein. "A performance study of roaming in wireless local area networks based on IEEE 802.11r". In: *2008 24th Biennial Symposium on Communications*. 2008, pp. 253–257. DOI: 10.1109/BSC.2008.4563250 (cit. on pp. 39, 50).

[CO14] S. S. Craciunas, R. S. Oliver. "SMT-Based Task- and Network-Level Static Schedule Generation for Time-Triggered Networked Systems". In: *Proceedings of the 22nd International Conference on Real-Time Networks and Systems*. RTNS '14. Versaille, France: Association for Computing Machinery, 2014, pp. 45–54. ISBN: 9781450327275. DOI: 10.1145/2659787.2659812. URL: https://doi.org/10.1145/2659787.2659812 (cit. on p. 41).

[CO15] S. S. Craciunas, R. S. Oliver. "Combined task- and network-level scheduling for distributed time-triggered systems". In: *Real-Time Systems* 52.2 (Oct. 2015), pp. 161–200. DOI: 10.1007/s11241-015-9244-x (cit. on p. 41).

[CPR+19] D. Cavalcanti, J. Perez-Ramirez, M. M. Rashid, J. Fang, M. Galeev, K. B. Stanton. "Extending Accurate Time Distribution and Timeliness Capabilities Over the Air to Enable Future Wireless Industrial Automation Systems". In: *Proceedings of the IEEE* 107.6 (2019), pp. 1132–1152. DOI: 10.1109/JPROC.2019.2903414 (cit. on p. 38).

[DCT+07] A. Dutta, S. Chakravarty, K. Taniuchi, V. Fajardo, Y. Ohba, D. Famolari, H. Schulzrinne. "An Experimental Study of Location Assisted Proactive Handover". In: *IEEE GLOBECOM 2007 - IEEE Global Telecommunications Conference*. 2007, pp. 2037–2042. DOI: 10.1109/GLOCOM.2007.390 (cit. on p. 39).

[DN16] F. Dürr, N. G. Nayak. "No-Wait Packet Scheduling for IEEE Time-Sensitive Networks (TSN)". In: *Proceedings of the 24th International Conference on Real-Time Networks and Systems*. RTNS '16. Brest, France: Association for Computing Machinery, 2016, pp. 203–212. ISBN: 9781450347877. DOI: 10.1145/2997465.2997494. URL: https://doi.org/10.1145/2997465.2997494 (cit. on pp. 40, 43).

[Ecl] Eclipse Fountation. *Eclipse IDE*. URL: https://www.eclipse.org/ide/ (cit. on p. 33).

[FS17] S. Feirer, T. Sauter. "Seamless handover in industrial WLAN using IEEE 802.11k". In: *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*. 2017, pp. 1234–1239. DOI: 10.1109/ISIE.2017.8001421 (cit. on pp. 39, 50).

[FSC+21] J. Fang, S. Sudhakaran, D. Cavalcanti, C. Cordeiro, C. Chen. "Wireless TSN with Multi-Radio Wi-Fi". In: *2021 IEEE Conference on Standards for Communications and Networking (CSCN)*. 2021, pp. 105–110. DOI: 10.1109/CSCN53733.2021.9686180 (cit. on p. 38).

[Gla20] A. Glavackij. “Tracing-based Scheduling of Isochronous Traffic in Time-Sensitive Networks”. Bachelor’s Thesis. Aug. 19, 2020 (cit. on p. 40).

[Gra66] R. L. Graham. “Bounds for certain multiprocessing anomalies”. In: *The Bell System Technical Journal* 45.9 (June 11, 1966), pp. 1563–1581 (cit. on p. 40).

[Gur] Gurobi. *Gurobi Optimizer*. Gurobi Optimization. URL: https://www.gurobi.com/ (visited on 06/09/2020) (cit. on pp. 32, 49).

[Hau20] L. Haug. “Optimizing ILP-based joint scheduling and routing for time-aware shaping in factory automation networks”. Bachelor’s Thesis. Universität Stuttgart, 2020. DOI: 10.18419/OPUS-11199. URL: http://elib.uni-stuttgart.de/handle/11682/11216 (cit. on pp. 20, 41, 43, 47).

[HDHK18] D. Hellmanns, F. Dürr, R. Hummen, S. Kehrer. In: *Reducing Runtime of Schedule and Route Synthesis in TSN without Sacrificing Valid Solutions*. 2018. DOI: 10.4230/LIPIcs.ECRTS.2018.YY (cit. on pp. 41, 43, 47, 75).

[HJA+21] J. Haxhibeqiri, X. Jiao, M. Aslam, I. Moerman, J. Hoebeke. “Enabling TSN over IEEE 802.11: Low-overhead Time Synchronization for Wi-Fi Clients”. In: *2021 22nd IEEE International Conference on Industrial Technology (ICIT)*. Vol. 1. 2021, pp. 1068–1073. DOI: 10.1109/ICIT46573.2021.9453686 (cit. on p. 37).

[Hus15] T. Husfeldt. *Graph colouring algorithms*. 2015. DOI: 10.48550/ARXIV.1505.05825. URL: https://arxiv.org/abs/1505.05825 (cit. on p. 36).

[IEEa] IEEE. *IEEE 802.1 Working Group*. URL: https://1.ieee802.org/ (visited on 08/25/2020) (cit. on p. 30).

[IEEb] IEEE. *Time-Sensitive Networking (TSN) Task Group*. URL: https://1.ieee802.org/tsn/ (visited on 08/25/2020) (cit. on p. 30).

[IEE04] IEEE. “IEEE Standard for information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment 6: Medium Access Control (MAC) Security Enhancements”. In: *IEEE Std 802.11i-2004* (2004), pp. 1–190. DOI: 10.1109/IEEESTD.2004.94585 (cit. on p. 28).

[IEE16a] IEEE. “IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic”. In: *IEEE Std 802.1Qbv-2015 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, and IEEE Std 802.1Q-2014/Cor 1-2015)* (2016), pp. 1–57 (cit. on p. 30).

[IEE16b] IEEEE. “IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks – Amendment 26: Frame Preemption”. In: *IEEE Std 802.1Qbu-2016 (Amendment to IEEE Std 802.1Q-2014)* (2016), pp. 1–52. DOI: 10.1109/IEEESTD.2016.7553415 (cit. on p. 38).

[IEE17] IEEEE. “IEEE Standard for Local and metropolitan area networks–Frame Replication and Elimination for Reliability”. In: *IEEE Std 802.1CB-2017* (2017), pp. 1–102. DOI: 10.1109/IEEESTD.2017.8091139 (cit. on p. 38).

[IEE20a] IEEE. “IEEE Standard for Local and Metropolitan Area Networks–Timing and Synchronization for Time-Sensitive Applications”. In: *IEEE Std 802.1AS-2020 (Revision of IEEE Std 802.1AS-2011)* (2020), pp. 1–421. DOI: 10.1109/IEEESTD.2020.9121845 (cit. on pp. 30, 37).

[IEE20b] IEEEE. “IEEE Standard for Low-Rate Wireless Networks”. In: *IEEE Std 802.15.4-2020 (Revision of IEEE Std 802.15.4-2015)* (2020), pp. 1–800. DOI: 10.1109/IEEESTD.2020.9144691 (cit. on p. 38).

[IEE21a] IEEE. “IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks–Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications”. In: *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016)* (2021), pp. 1–4379. DOI: 10.1109/IEEESTD.2021.9363693 (cit. on pp. 22, 23, 37).

[IEE21b] IEEE. “IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks–Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 1: Enhancements for High-Efficiency WLAN”. In: *IEEE Std 802.11ax-2021 (Amendment to IEEE Std 802.11-2020)* (2021), pp. 1–767. DOI: 10.1109/IEEESTD.2021.9442429 (cit. on p. 37).

[IEE22a] IEEE. “IEEE Standard for Ethernet”. In: *IEEE Std 802.3-2022 (Revision of IEEE Std 802.3-2018)* (2022), pp. 1–7025. DOI: 10.1109/IEEESTD.2022.9844436 (cit. on pp. 19, 20).

[IEE22b] IEEE. “IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks”. In: *IEEE Std 802.1Q-2022 (Revision of IEEE Std 802.1Q-2018)* (2022). DOI: 10.1109/IEEESTD.2022.10004498 (cit. on pp. 20, 21, 37).

[KLA20] E. Khorov, I. Levitsky, I. F. Akyildiz. “Current Status and Directions of IEEE 802.11be, the Future Wi-Fi 7”. In: *IEEE Access* 8 (2020), pp. 88664–88688. DOI: 10.1109/ACCESS.2020.2993448 (cit. on p. 37).

[Kub04] M. Kubale. *Graph Colorings*. en. Contemporary mathematics. Providence, RI: American Mathematical Society, June 2004 (cit. on p. 36).

[MVK19] L. Mészáros, A. Varga, M. Kirsche. “INET Framework”. In: *Recent Advances in Network Simulation: The OMNeT++ Environment and its Ecosystem*. Ed. by A. Virdis, M. Kirsche. Cham: Springer International Publishing, 2019, pp. 55–106. ISBN: 978-3-030-12842-5. DOI: 10.1007/978-3-030-12842-5_2. URL: https://doi.org/10.1007/978-3-030-12842-5_2 (cit. on pp. 33, 34).

[PRGS18] P. Pop, M. L. Raagaard, M. Gutierrez, W. Steiner. “Enabling Fog Computing for Industrial Automation Through Time-Sensitive Networking (TSN)”. In: *IEEE Communications Standards Magazine* 2.2 (2018), pp. 55–61 (cit. on p. 41).

[PRO14] PROFIBUS Nutzerorganisation e. V. “PROFINET System Description”. In: (2014) (cit. on p. 41).

[Rot02] J. Roth. *Mobile computing: Grundlagen, Technik, Konzepte*. 1. Aufl. Heidelberg: dpunkt-Verl., 2002. ISBN: 3898641651. URL: http://www.gbv.de/dms/hebis-darmstadt/toc/104390921.pdf (cit. on p. 25).

[SAE16] SAE International. *Time-Triggered Ethernet*. Nov. 9, 2016. DOI: 10.4271/as6802 (cit. on p. 41).

[Sch86] A. Schrijver. *Theory of Linear and Integer Programming*. USA: John Wiley Sons, Inc., 1986. ISBN: 0471908541 (cit. on p. 31).

[SDT+17] E. Schweissguth, P. Danielis, D. Timmermann, H. Parzyjegla, G. Mühl. "ILP-Based Joint Routing and Scheduling for Time-Triggered Networks". In: *Proceedings of the 25th International Conference on Real-Time Networks and Systems*. RTNS '17. Grenoble, France: Association for Computing Machinery, 2017, pp. 8–17. ISBN: 9781450352864. DOI: 10.1145/3139258.3139289. URL: https://doi.org/10.1145/3139258.3139289 (cit. on p. 41).

[SMBC21] S. Sudhakaran, V. Mageshkumar, A. Baxi, D. Cavalcanti. "Enabling QoS for Collaborative Robotics Applications with Wireless TSN". In: *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*. 2021, pp. 1–6. DOI: 10.1109/ICCWorkshops50388.2021.9473897 (cit. on p. 38).

[SSZ15] G. Sierksma, G. Sierksma, Y. Zwols. *Linear and Integer Optimization*. Chapman and Hall/CRC, May 2015. DOI: 10.1201/b18378. URL: https://doi.org/10.1201/b18378 (cit. on p. 32).

[Ste10] W. Steiner. "An Evaluation of SMT-Based Schedule Synthesis for Time-Triggered Multi-hop Networks". In: *2010 31st IEEE Real-Time Systems Symposium*. 2010, pp. 375–384 (cit. on p. 41).

[STP+20] E. Schweissguth, D. Timmermann, H. Parzyjegla, P. Danielis, G. Mühl. "ILP-Based Routing and Scheduling of Multicast Realtime Traffic in Time-Sensitive Networks". In: *2020 IEEE 26th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. 2020, pp. 1–11 (cit. on p. 41).

[Ull75] J. D. Ullman. "NP-Complete Scheduling Problems". In: *J. Comput. Syst. Sci.* 10.3 (June 1975), pp. 384–393. ISSN: 0022-0000. DOI: 10.1016/S0022-0000(75)80008-0. URL: https://doi.org/10.1016/S0022-0000(75)80008-0 (cit. on p. 31).

[Var01] A. Varga. "The OMNET++ discrete event simulation system". In: *Proc. ESM'2001* 9 (Jan. 2001). URL: https://omnetpp.org/ (cit. on p. 33).

[VH08] A. Varga, R. Hornig. "An Overview of the OMNeT++ Simulation Environment". In: *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*. Simutools '08. Marseille, France: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008. ISBN: 9789639799202 (cit. on p. 33).

[ZZW16] A. Zubow, S. Zehl, A. Wolisz. "BIGAP — Seamless handover in high performance enterprise IEEE 802.11 networks". In: *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*. 2016, pp. 445–453. DOI: 10.1109/NOMS.2016.7502842 (cit. on pp. 39, 40).

All links were last followed on April 4, 2023.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature