AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ INFORMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI

Praca dyplomowa

Analiza statystyk warstwy łącza danych w sieciach IEEE 802.11
Analysis of Data Link-Layer Statistics in IEEE 802.11 Networks

Author: *Daria Dziunikowska*
Degree programme: *Teleinformatyka*
Supervisor: *prof. dr hab. inż. Szymon Szott*

Kraków, 2025

I would like to thank my supervisor, prof. dr hab. inż. Szymon Szott, for his guidance, support and valuable feedback throughout this thesis.

Table of Contents

Figures

Tables

Listings

1. Introduction

Wireless local area networks based on the IEEE 802.11 standard (Wi-Fi) are widely deployed and they often work in places where many devices try to use the same radio channel at the same time. When that happens, performance is mostly determined by what goes on at the PHY and MAC layers: contention for the medium, collisions, retransmissions, reception errors and how much airtime is actually consumed. Analyzing this performance is crucial in understanding how to properly configure Wi-Fi networks.

Performance analysis of IEEE 802.11 with analytical models can be helpful in building intuition, but they usually depend on strong simplifying assumptions. With modern Wi-Fi features and detailed timing effects, these models quickly become hard to use in practice. Meanwhile, experimental testbeds, while providing more precise results, are challenging to set up. Therefore, simulation-based analysis is commonly used in Wi-Fi performance studies. In this thesis, simulations are performed with the ns-3 network simulator, which is popular due to its open-source character, its detailed implementation of IEEE 802.11 PHY and MAC behavior, as well as support for higher-layer functionalities.

Any reliable simulation study needs to capture the right metrics. Measures such as end-to-end throughput or delay are easy to collect in ns-3, but they do not directly show the link-layer mechanisms that often dominate performance in heavily contended Wi-Fi networks. Until recently, obtaining PHY- and MAC-level statistics in ns-3 in a clean, reusable way was inconvenient – newer ns-3 versions include dedicated modules and helper classes that make this easier.

The goal of this thesis is to evaluate these new measurement mechanisms, develop example simulation code that demonstrates how to use them, verify the collected statistics in controlled validation scenarios and then apply the validated tools to analyse saturated IEEE 802.11 network performance. The work is fully simulation-based and focuses both on metric collection and on designing and running validation and performance scenarios. All the simulation code developed as part of this thesis is published to support reproducibility and further development [9].

The remainder of this thesis is organized as follows. Section 2 introduces the necessary background on IEEE 802.11 MAC operation, typical Wi-Fi topologies and existing approaches to Wi-Fi performance evaluation in ns-3. Section 3 describes the simulation model, including the implemented scenarios, configuration choices and link-layer measurement mechanisms. Section 4 presents the simulation results, starting with validation tests and followed by performance analysis under saturated conditions. Finally, Section 5 presents the main findings and outlines possible directions for future work.

2. Background

This chapter introduces the background needed to understand the results presented later in the thesis. It recalls the IEEE 802.11 MAC mechanisms that are most relevant under high load, focusing on how stations access the medium, how reliability is maintained and where airtime is consumed beyond payload transmission. The chapter also summarises how Wi-Fi performance is typically evaluated in ns-3 and explains why end-to-end metrics alone often fail to reveal the causes of throughput degradation or increasing delay. Finally, it highlights a practical gap observed in many ns-3-based Wi-Fi studies: limited and insufficiently validated analysis of PHY- and MAC-layer behaviour, which makes it difficult to relate high-level performance results to concrete link-layer mechanisms.

2.1. IEEE 802.11 MAC

Most IEEE 802.11 networks operate in an infrastructure-based topology, where multiple wireless stations communicate through a central Access Point (AP). The AP coordinates medium access, handles association procedures and typically acts as the receiver for uplink traffic generated by stations. Unless stated otherwise, this thesis assumes such an infrastructure topology with a single AP and multiple associated stations. The term AP is used consistently throughout the thesis.

In IEEE 802.11 (Wi-Fi), stations can share the same radio channel. Unlike in a wired network, transmissions are not physically separated and all stations operate on a common medium. As a result, medium access is a central part of the protocol rather than a minor implementation detail. Once the network becomes busy, contention, collisions and MAC-layer timing rules largely determine the observed performance.

In dense or heavily loaded deployments, MAC behaviour is often the main reason why throughput drops and why delay and jitter increase. From a performance perspective, three aspects are particularly important: how a station gains access to the channel, what happens when a transmission is not acknowledged and how much airtime is spent on protocol overhead instead of useful payload.

Medium access in IEEE 802.11 is governed by the Distributed Coordination Function (DCF), which is based on CSMA/CA. A station first senses the channel and defers transmission while the medium is busy. When the channel becomes idle, the station waits for a DIFS interval and then applies a random backoff. If the backoff counter happens to be zero, transmission starts immediately after DIFS. Otherwise, the station counts down a number of backoff slots, decrementing the counter only while the

channel remains idle. If another station starts transmitting, the countdown is frozen and resumed later. This procedure directly affects collision probability, access delay and how fairly airtime is shared among contending stations.

Reliability at the MAC layer is ensured through positive acknowledgement frames (ACKs). Unicast frames are confirmed by ACKs sent by the receiver. If the sender does not receive an ACK within the expected time, it treats the transmission as failed and retries after another backoff. Retransmissions continue until the frame is successfully delivered or a retry limit is reached, after which the MAC drops the frame. This distinction is important when analysing results, since packet losses observed at higher layers may originate from retry-limit drops rather than from PHY decoding failures.

Even when transmissions succeed, a significant portion of airtime is consumed by protocol overhead. Interframe spaces, backoff time, PHY headers and control frames—such as the RTS/CTS (Request to Send / Clear to Send) exchange, which is used for collision avoidance by reserving the medium prior to data transmission—all occupy the channel, and much of this cost is paid regardless of payload size. Under high load, this overhead becomes a limiting factor for efficiency and throughput. Newer IEEE 802.11 amendments mitigate part of this cost through frame aggregation and Block Acknowledgement, which reduce per-frame overhead but also change channel holding time and retransmission behaviour under contention.

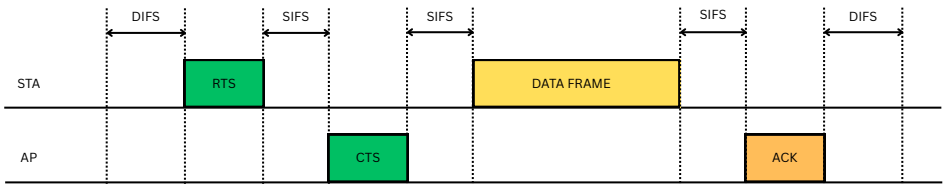

Figure 2.1. Basic IEEE 802.11 frame exchange sequence with RTS/CTS protection to illustrate all MAC-layer overhead in an idle channel.

Figure 2.1 illustrates a typical RTS/CTS-based exchange. The presented case is of a transmission in an idle channel when the backoff countdown is not needed (equivalently, when the backoff counter is zero), so the timeline highlights only the mandatory interframe spaces and the control/data frames. Even in this simplified view, a single payload transmission is surrounded by RTS/CTS, ACK, and timing gaps, all of which consume airtime. In a crowded network, the missing piece is the random backoff,

which adds extra waiting time and increases the chance of collisions and retransmissions. Together, these effects reduce throughput and push delay and jitter upward.

2.2. Literature Review

The ns-3 network simulator is widely used to analyse and evaluate communication networks. It is open-source, modular and provides detailed protocol models, which makes it a practical choice for reproducible simulation studies, including wireless research. A broader overview of ns-3 and its application areas is provided in [1]. In the context of Wi-Fi, an important feature of ns-3 is that it exposes many internal protocol events through trace sources and helper mechanisms, allowing researchers to observe behaviour inside the protocol stack rather than relying solely on end-to-end results.

In a large part of ns-3-based Wi-Fi literature, performance is still reported primarily using network-layer metrics such as throughput, delay and packet loss. Examples of recent research on Wi-Fi in ns-3 which reports only end-to-end metrics inlude [11]. In these ns-3 studies, end-to-end metrics are commonly collected with the `FlowMonitor` module [2], which provides per-flow, end-to-end statistics at the IP layer. While this approach is convenient for high-level comparisons, it leaves out the mechanisms that often dominate Wi-Fi performance under load. In particular, `FlowMonitor` does not indicate whether losses originate from collisions, repeated MAC retransmissions, or PHY-layer decoding failures and it does not capture medium access delays directly. When contention is high, this lack of context makes it difficult to explain *why* throughput degrades or delay increases, even if the end-to-end measurements themselves are clear.

This problem is especially visible in dense or saturated Wi-Fi scenarios, where understanding performance requires looking at PHY- and MAC-layer operation. For example, recent experimental work on Wi-Fi multi-link operation shows that latency and throughput are strongly influenced by link-level effects that are not visible from network-layer metrics alone [3]. Under saturation, collisions, backoff, retransmissions and protocol overhead overlap in time, so similar network-layer results can correspond to very different link-layer dynamics. Airtime-based measurements illustrate this well. Channel occupancy (CO), introduced in [4], measures the fraction of time the channel is sensed as busy regardless of whether payload is delivered successfully and has been used to help explain access delays and scheduling behaviour under contention. This suggests that link-layer measurements play a key role in explaining Wi-Fi behaviour under saturation and merit further attention in future studies.

2.3. Positioning of This Work

Existing ns-3 studies clearly show that the simulator already contains detailed IEEE 802.11 PHY and MAC models and exposes a rich set of internal trace points. In principle, this makes ns-3 well suited for analysing link-layer behaviour in Wi-Fi networks. In practice, however, many studies rely primarily on end-to-end metrics such as throughput, delay, or packet loss. When link-layer measurements are

included, they are often extracted through custom-added code, without any systematic validation. As a result, it is not always clear whether the collected PHY- and MAC-layer statistics accurately reflect expected protocol behaviour, nor how much confidence should be placed in their interpretation once the scenario becomes highly contended.

What is still lacking are well-documented and reproducible examples that demonstrate how to instrument ns-3 at the PHY and MAC layers, how to sanity-check the resulting measurements against simple reference cases and how to relate reception outcomes, retransmission counters and airtime usage to the trends observed in throughput and delay. The goal of this work is to build such a measurement workflow using existing ns-3 mechanisms, validate it in simple scenarios and then apply the same instrumentation to interpret performance degradation in saturated Wi-Fi networks.

3. Simulation Model

This chapter describes the simulation model used in the thesis and provides the technical background for the results presented later. It introduces the ns-3 simulator as the execution environment and explains how IEEE 802.11 scenarios are constructed. The chapter also outlines the main configuration choices and the measurement mechanisms used to observe PHY- and MAC-layer behaviour. By documenting the simulation setup in detail, this chapter serves as a reference for the performance results discussed in Chapter 4.

3.1. Overview of ns-3

All simulation studies presented in this thesis were carried out using the `ns-3` network simulator (version ns-3.44, released on 2025-03-09) [10]. Ns-3 is a discrete-event simulation platform widely used in research to model and evaluate communication networks. Its modular design makes it easy to configure protocol stacks and to control simulation parameters, timing and event scheduling in a repeatable way. Ns-3 is also used beyond classical packet-level studies, for example in real-time network emulation [5], in setups coupled with ray-tracing engines to improve channel modelling [6, 7] and in simulation frameworks targeting large-scale satellite networks [8]. In the following, we focus on ns-3's applicability as a simulator of Wi-Fi networks.

For IEEE 802.11 networks, ns-3 provides a dedicated Wi-Fi module with detailed PHY and MAC implementations. The PHY layer models transmission and reception timing together with channel propagation and frame decoding outcomes, while the MAC layer implements contention-based access, retransmission logic and common frame exchange procedures. These mechanisms directly determine performance once the medium becomes congested. The Wi-Fi module is actively developed and recent releases introduce incremental support for emerging IEEE 802.11 features, such as the ability to exchange IEEE 802.11be Multi-Link Probe Request frames, as documented in the ns-3.44 release notes [10].

What matters most for this thesis is the level of visibility that ns-3 provides at the link layer. Internal PHY and MAC events are typically exposed through trace sources and callbacks can be attached to observe reception outcomes, retransmissions and medium access activity during runtime. Such information is not easily accessible in real-world network- or application-layer tools, but it is needed to explain performance degradation in contention-dominated Wi-Fi scenarios.

In this work, ns-3 is used primarily for link-layer analysis rather than only for reporting end-to-end results (as it is commonly used). The simulation code is based on incremental extensions of a reference ns-3 Wi-Fi example (`ex1.cc`), which is provided in Appendix A.1. Traffic generation is typically and intentionally kept simple and is configured to create saturated uplink conditions, so that the measured behaviour is driven mainly by PHY and MAC mechanisms. The following sections describe how IEEE 802.11 simulations are built in ns-3 and explain how link-layer measurements are integrated into the simulation model.

3.1.1. Basics of Simulating IEEE 802.11 Networks

A typical IEEE 802.11 simulation in ns-3 is built from a set of modular blocks: network nodes, Wi-Fi network devices, a wireless channel model, PHY/MAC configuration and traffic-generating applications. Network nodes represent stations and access points. Each node hosts one or more Wi-Fi network devices (`WifiNetDevice`), which combine the PHY and MAC implementations. The Wi-Fi net device links the wireless medium with higher layers and also exposes PHY and MAC trace sources that can be used for instrumentation.

The PHY model determines transmission timing, reception processes and decoding outcomes. It defines whether a PSDU is decoded successfully, when reception fails and how long the medium is perceived as busy or idle. These events are the basis for reception-oriented metrics such as successful and failed PSDU receptions, collision effects and channel occupancy.

The MAC model implements medium access (including backoff), retransmission behaviour and frame exchange sequences such as RTS/CTS/DATA/ACK. Since these mechanisms directly control access delay and reliability under contention, the MAC layer is a central focus of this thesis.

Above the link layer, the IP stack is installed using standard ns-3 components and traffic is generated using transport-layer applications. In this thesis, traffic generation is kept simple to minimise higher-layer effects and keep the analysis centred on PHY and MAC behaviour.

3.1.2. FlowMonitor and Packet Capture Tools

Ns-3 includes several tools that help with monitoring and debugging. Two commonly used mechanisms are `FlowMonitor` and packet capture (PCAP) traces. The former is a built-in ns-3 module while the latter generates files, which can be analysed in external tools such as Wireshark.

`FlowMonitor` collects statistics at the network (IP) layer, including per-flow throughput, packet loss, delay and jitter [2]. This module is easy to setup and is able to automatically detect all network-layer flows during the course of the simulation. The provided metrics are useful for end-to-end performance analysis, but this does not expose IEEE 802.11 PHY/MAC behaviour. In particular, `FlowMonitor` does not show retransmissions, reception failures, collision effects, or medium access delays.

PCAP traces allow frame-level inspection and help verify traffic patterns and frame exchange sequences. They work well as a sanity check, but they are not sufficient for detailed link-layer statistics

when the analysis depends on internal decoding events, failure reasons, or precise timing relations between PHY and MAC.

For these reasons, `FlowMonitor` and PCAP are used mainly as auxiliary tools in this thesis. The core measurement approach relies on instrumentation placed directly at the PHY and MAC layers of the ns-3 Wi-Fi model.

3.2. Link-Layer Measurement Helpers

To analyse IEEE 802.11 behaviour at the link layer, this work relies on measurement components available in ns-3 version 3.44. This release extends the Wi-Fi module with additional trace sources and helper-based mechanisms that provide direct access to PHY- and MAC-layer events, including transmission outcomes, retransmission dynamics and channel occupancy. These mechanisms are documented in the official ns-3.44 release notes and changelog [10].

Rather than relying on a single source of statistics, complementary helper-based components are used to observe different aspects of link-layer behaviour. This allows reception outcomes at the PHY layer, transmission decisions at the MAC layer and airtime usage on the channel to be analysed in a consistent manner.

In the remainder of this chapter, the term *helper* refers both to native ns-3 helper classes and to measurement components that wrap trace callbacks and aggregation logic in a reusable form. Three helper-based components are used throughout this work:

- **`WifiPhyRxTraceHelper`** for PHY-layer reception statistics,
- **`WifiTxStatsHelper`** for MAC-layer transmission and retransmission statistics at the MPDU level,
- **`WifiCoTraceHelper`** for channel occupancy and airtime utilisation.

All these helpers are recent additions to ns-3 and each helper captures a different aspect of operation. PHY reception statistics describe decoding success and failure, MAC transmission statistics capture retransmission dynamics and retry-limit drops and channel occupancy shows how airtime is consumed regardless of payload delivery.

Together, these helpers form a consistent measurement framework that is used in validation tests and performance experiments. Correctness is checked in the validation tests in Chapter 4, while independent measurements are then used to interpret behaviour under saturated conditions.

3.2.1. Measurement Mechanisms and Collected Metrics

The measurement framework consists of the three helpers listed above, supplemented by a small set of custom trace callbacks used mainly for timing-related metrics. Each component adds a different

perspective on link-layer operation and the combined view is used in both validation and saturation scenarios.

3.2.1.1. `WifiPhyRxTraceHelper`: PHY Reception Statistics

`WifiPhyRxTraceHelper` is a helper provided by the ns-3 Wi-Fi module for collecting physical-layer reception events. It connects to PHY trace sources exposed by the `WifiPhy` model, most notably `PhyRxEnd` and `PhyRxDrop`, which correspond to successful PSDU reception and reception failure at the physical layer, respectively. Listing 3.1 shows how the helper is enabled and attached to all Wi-Fi PHY instances present in the simulation.

```
WifiPhyRxTraceHelper phyRxHelper;

phyRxHelper.Enable(
  "/NodeList/*/DeviceList/*/$ns3::WifiNetDevice/Phy"
);
```

Listing 3.1. Enabling PHY reception statistics using WifiPhyRxTraceHelper

The helper maintains counters for successfully received and failed PSDUs and provides access to detailed per-event information through its interfaces. In this thesis, the reception events collected by `WifiPhyRxTraceHelper` are further processed to identify individual IEEE 802.11 frame types (e.g., DATA, ACK, RTS, CTS). This is achieved by inspecting the associated `WifiMacHeader` of each received PSDU.

This classification enables PHY-layer reception outcomes to be directly correlated with specific MAC-layer frame exchanges. In the validation scenarios, these correlations are used to verify the correct sequence and timing of control and data frames against the expected IEEE 802.11 operation. In the performance evaluation scenarios, the same information is used to relate PHY-level reception failures to MAC-layer retransmissions and retry-limit drops observed by the corresponding MAC statistics helpers. As a result, `WifiPhyRxTraceHelper` provides a direct link between physical-layer decoding behaviour and higher-layer transmission outcomes, making it possible to interpret reliability and contention effects consistently across PHY and MAC layers.

3.2.1.2. `WifiTxStatsHelper`: MAC Transmission and Retransmission Statistics

While PHY-layer statistics describe reception outcomes, they do not capture how the medium access control layer reacts to transmission failures. MAC-layer behaviour is therefore analysed using `WifiTxStatsHelper`, which provides MPDU-level transmission statistics without the need for manual parsing of low-level trace events. Listing 3.2 shows how the helper is enabled for both the access point and the transmitting stations in the simulation.

```
WifiTxStatsHelper txStats;
txStats.Enable(apDevice);
txStats.Enable(staDevices);
```

Listing 3.2. Enabling MAC-layer transmission statistics using WifiTxStatsHelper

The helper records the number of MPDU transmission attempts, retransmissions triggered by missing acknowledgements, and frames dropped after exceeding the retry limit. These metrics provide direct insight into MAC-layer contention and recovery mechanisms.

In this thesis, the collected MPDU-level statistics are used to compute MPDU failure and retransmission rates as functions of the number of active stations. They are analysed together with PHY-layer reception results to explain how increasing contention leads to more frequent retransmissions, retry-limit drops and, consequently, throughput degradation and increased delay under saturated traffic conditions.

3.2.1.3. `WifiCoTraceHelper`: Channel Occupancy and Airtime Utilisation

To capture airtime usage directly, channel occupancy is measured using `WifiCoTraceHelper`. The helper records Channel Occupancy Time (COT) metrics that describe how long the wireless medium is sensed as busy, independently of whether payload data is successfully transmitted.

Listing 3.3 shows how the helper is enabled for all Wi-Fi devices participating in the simulation.

```
WifiCoTraceHelper coTrace;
coTrace.Enable(wifiDevices);
```

Listing 3.3. Enabling channel occupancy tracing using WifiCoTraceHelper

The helper provides several COT-related metrics describing different aspects of medium occupation. In this thesis, the most variable COT metric is selected for detailed analysis, as it is particularly sensitive to changes in offered load and contention intensity.

Channel occupancy results are analysed together with throughput and MAC-layer transmission statistics to assess how efficiently the available airtime is used under saturated conditions. This makes it possible to distinguish whether performance degradation is caused by a lack of transmission opportunities or by inefficient medium access due to contention.

3.2.1.4. Additional Callback-Based Instrumentation

Some metrics required in selected experiments are collected using custom trace callbacks that are not wrapped into dedicated helper classes. These callbacks are used primarily for timing-related measurements, such as one-way delay, jitter, transmission and reception timestamps, as well as data transfer duration.

Listing 3.4 shows how a PHY-layer trace callback is registered using the `MonitorSnifferRx` trace source, which is triggered for each successfully received MPDU.

```
Config::Connect(
  "/NodeList/*/DeviceList/*/$ns3::WifiNetDevice/Phy/MonitorSnifferRx",
  MakeCallback(&MyRxMonitorSnifferCallback)
);
```

Listing 3.4. Registration of a PHY-layer callback using the MonitorSnifferRx trace source

Callback-based instrumentation complements the helper-based statistics by enabling fine-grained observation of individual transmission events that cannot be easily derived from aggregated counters alone. In particular, `MonitorSnifferRx` provides direct access to reception timestamps at the moment of successful frame decoding, which allows precise temporal analysis during simulation runtime.

In this thesis, the registered callback is used to record the first and last reception times per device, as well as to analyse inter-arrival times of successfully received frames originating from the same transmitting station. These timestamps are later used to derive metrics such as data transfer duration and channel access delay.

By correlating PHY-layer reception timestamps obtained via `MonitorSnifferRx` with MAC-layer transmission statistics collected by the helper-based mechanisms, it becomes possible to interpret medium access behaviour and contention effects under saturated traffic conditions in a consistent manner.

3.3. Simulation Scenarios and Source Files

This section describes the simulation scenarios developed for this thesis together with the corresponding ns-3 source files. Each scenario represents a separate experimental setup and is used either to validate the correctness of the measurement framework or to analyse IEEE 802.11 performance under contention. The purpose of this section is to clarify the role of each scenario and to document its default configuration, so that the results presented in Chapter 4 can be interpreted in the correct experimental context. All file names are in reference to the project's repository [9].

3.3.1. Single-Station IEEE 802.11a Validation Scenario

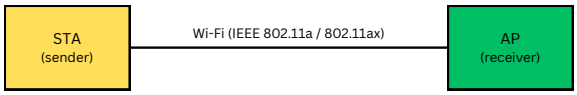

Figure 3.1. Single-station IEEE 802.11 simulation topology with one transmitting station (STA) and one receiving node (AP).

The most basic validation setup is implemented in `wifi-stats.cc`. It represents a simple IEEE 802.11a scenario with one transmitting station and one receiving node (Figure 3.1). With only a single transmitting station, the channel remains contention-free and the observed behaviour is determined solely by PHY- and MAC-layer mechanisms, as the nodes are placed in close proximity and the scenario does not involve signal attenuation effects. Although the logical topology follows an AP–STA structure, AP-specific functionality is disabled. Beacon transmissions and other management procedures are turned off and both nodes operate in ad-hoc mode, as configured in Listing 3.5. This prevents periodic management traffic from affecting timing, medium occupancy and PHY/MAC statistics. For clarity, the nodes are still referred to as STA (sender) and AP (receiver).

This scenario is used to verify basic IEEE 802.11 protocol properties, including the order of control and data frames, inter-frame spacing, acknowledgement handling and transmission timing. Two traffic variants are considered: one with a single transmitted DATA frame and one with a short sequence of consecutive DATA frames. Apart from the number of frames, the configuration remains unchanged, allowing timing relations and collected statistics to be compared directly. The PHY configuration follows legacy IEEE 802.11a operation in the 5 GHz band with a 20 MHz channel. A fixed modulation and coding scheme is used and rate adaptation is disabled to ensure deterministic behaviour. RTS/CTS is enabled explicitly so that MAC-layer overhead and control exchanges are clearly visible in the collected traces. This simplified configuration provides a clean baseline scenario against which more complex multi-station and contention-based experiments are later compared in Chapter 4.

```
% === AD-HOC MODE ===
mac.SetType("ns3::AdhocWifiMac");
NetDeviceContainer staDevices = wifi.Install(phy, mac, wifiStaNodes);
NetDeviceContainer apDevices  = wifi.Install(phy, mac, wifiApNode);
```

Listing 3.5. Configuration of IEEE 802.11 nodes operating in ad-hoc mode

3.3.2. Single-Station IEEE 802.11ax Validation Scenario

The single-station IEEE 802.11ax validation scenario is implemented in `wifi-stats_ax.cc`. The network topology is identical to the IEEE 802.11a validation case and consists of two nodes operating in ad-hoc mode (Figure 3.1). The STA (sender) and AP (receiver) naming convention is preserved for consistency throughout the thesis and beacon transmissions remain disabled.

The station transmits a short sequence of DATA frames using an IEEE 802.11ax PHY and MAC configuration. To maintain full control over the acknowledgement mechanism during validation, the Block ACK agreement can be configured statically using `WifiStaticSetupHelper`, as shown in Listing 3.6. PHY parameters such as channel width and guard interval are configured explicitly and rate adaptation is disabled to ensure reproducible timing behaviour. In addition to aggregate PHY- and MAC-layer statistics, this scenario includes explicit classification of successfully received frames at the PHY layer, which allows DATA, ACK, RTS, CTS and Block ACK frames to be distinguished during analysis and enables detailed assessment of protocol overhead and airtime usage. This scenario is used in Chapter 4 to validate PHY- and MAC-layer measurements under IEEE 802.11ax operation before moving to multi-station experiments.

```
WifiStaticSetupHelper::SetStaticBlockAck(apDev, staDevices, {0});
```

Listing 3.6. Enabling static Block ACK operation using WifiStaticSetupHelper

3.3.3. Collision Probability Scenario

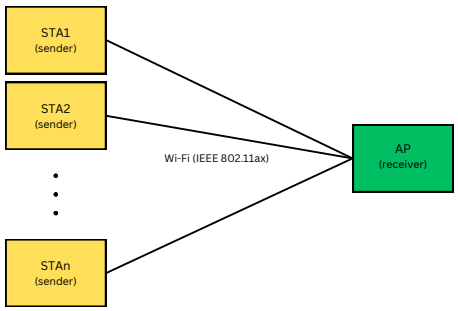

Figure 3.2. IEEE 802.11ax validation topology with multiple transmitting stations (STA) and a single access point (AP).

MAC-layer contention is analysed using a multi-station IEEE 802.11ax scenario implemented in `wifi-stats-pcoll.cc`. The topology consists of one access point and a variable number of stations transmitting uplink traffic over a shared wireless channel (Figure 3.2), with all stations placed in close proximity to the access point so that packet losses caused by weak signal conditions are excluded and the observed failures can be attributed solely to contention effects. Traffic generation is configured to keep the medium saturated, so that stations continuously compete for access. As the number of active stations increases, collisions and retransmissions become more frequent. For each run, MPDU-level transmission statistics are collected using `WifiTxStatsHelper`, including the number of successful transmissions, retransmissions triggered by missing acknowledgements and frames dropped after exceeding the retry limit.

Collision probability is not computed directly during simulation. Instead, the scenario produces raw MAC-layer counters that are later processed to estimate collision probability and to compare the simulation results with an analytical contention model. The estimator treats MAC-layer retransmissions and retry-limit drops as indicators of failed transmission attempts and relates them to the total number of transmission attempts, providing an indirect but practical measure of collision probability under saturated conditions, as formalised in Listing 3.7. The estimated collision probability is later used in Chapter 4 to analyse how contention intensity evolves with the number of active stations and to compare simulation results with analytical contention models.

```
1 pCollision = static_cast<double>(failures_retry + mpduRetries)
2              / (successes + failures_retry + mpduRetries);
```

Listing 3.7. Estimation of collision probability based on MAC-layer transmission outcomes

3.3.4. Delay Measurement Scenario

Delay-related behaviour under saturation is studied using the scenario implemented in `wifi-stats-delay.cc`. The network consists of one AP and multiple stations generating continuous uplink traffic, ensuring persistent contention for channel access (Figure 3.2). The emphasis of this scenario is on timing and therefore custom MAC- and PHY-layer trace callbacks are used to record transmission and reception timestamps for DATA frames, which form the basis for delay-related measurements.

Based on the collected timestamps, two complementary delay metrics are derived. The first one is the one-way end-to-end delay, defined as the time difference between frame transmission at the sender and its successful reception at the receiver. This metric reflects the complete delivery time experienced by a packet and includes the effects of channel access contention, retransmissions, PHY transmission time and protocol overhead. The implementation of this measurement relies on matching transmission and reception timestamps using trace callbacks, as illustrated in Listing 3.8.

The second metric focuses specifically on medium access behaviour. Channel access delay is estimated from the time interval between consecutive successful receptions of DATA frames originating from the same transmitter. In this way, the influence of MAC-layer mechanisms such as backoff, contention and medium access arbitration can be isolated, showing how long a station must wait before gaining access to the shared wireless medium under saturated conditions. The corresponding measurement logic is presented in Listing 3.9.

The collected timing data are not processed during the simulation itself. Instead, delay samples from multiple runs are exported and analysed afterwards. This includes the construction of empirical delay distributions and cumulative distribution functions, which are later presented in Chapter 4. Separating data collection from statistical processing keeps the simulation logic simple and ensures consistent handling of delay samples across runs.

```
1  static std::map<uint64_t, Time> gTxTimestampPerSeq;
2  static std::vector<double> gDelays;
3
4  static void TxTrace(std::string context, Ptr<const Packet> packet)
5  {
6      gTxTimestampPerSeq[packet->GetUid()] = Simulator::Now();
7  }
8
9  static void RxTrace(std::string context, Ptr<const Packet> packet)
10 {
11     // ...
12     auto it = gTxTimestampPerSeq.find(packet->GetUid());
13     if (it != gTxTimestampPerSeq.end())
14     {
15         double delay = (Simulator::Now() - it->second).GetSeconds();
```

```
        gDelays.push_back(delay);
        gTxTimestampPerSeq.erase(it);
    }
}
```

Listing 3.8. End-to-end delay measurement based on packet transmission and reception timestamps

```
static std::map<Mac48Address, Time> gLastRxTime;
static std::vector<double> gChannelAccessDelays;

//...

Mac48Address sender = hdr.GetAddr2();
Time now = Simulator::Now();

if (gLastRxTime.find(sender) != gLastRxTime.end())
{
    double delta = (now - gLastRxTime[sender]).GetSeconds();
    if (delta > 0.0)
        gChannelAccessDelays.push_back(delta);
}

gLastRxTime[sender] = now;
```

Listing 3.9. Channel access delay estimation based on inter-arrival times of received frames

3.3.5. Saturated Performance Scenario

Overall performance under saturated conditions is analysed using the scenario implemented in `wifi-stats-perf.cc`. The topology again consists of one AP and multiple uplink stations sharing a common wireless channel (Figure 3.2). Simulation parameters are kept fixed across runs and only the number of active stations is varied, which allows isolating the impact of contention intensity on network performance.

In this scenario, several link-layer measurement helpers are used simultaneously. `WifiTxStatsHelper` captures MAC-level transmission and retransmission behaviour, while `WifiCoTraceHelper` records channel occupancy to quantify airtime usage independently of successful payload delivery. Together, these measurements provide a comprehensive view of efficiency, reliability and airtime consumption as contention increases. Traffic generation is configured to keep the wireless medium continuously saturated, so that each station always has data ready for transmission. This is achieved using a constant-rate `OnOff` application configuration, shown in Listing 3.10, which ensures uninterrupted packet generation throughout the simulation.

The resulting symmetric traffic pattern stresses the MAC and PHY layers under full-load conditions and forms the basis for the throughput, retransmission, collision probability, channel access delay and air-time utilisation results discussed in Chapter 4. By installing the same application instance on all stations, a fair and reproducible comparison between different network sizes is obtained.

```
OnOffHelper onOff("ns3::UdpSocketFactory", sinkSocket);

onOff.SetAttribute("OnTime",
    StringValue("ns3::ConstantRandomVariable[Constant=1]"));
onOff.SetAttribute("OffTime",
    StringValue("ns3::ConstantRandomVariable[Constant=0]"));

onOff.SetAttribute("DataRate",
    DataRateValue(DataRate(offeredLoad)));
onOff.SetAttribute("PacketSize",
    UintegerValue(payload));

for (uint32_t i = 0; i < nWifi; ++i)
{
    sourceApplications.Add(onOff.Install(wifiStaNodes.Get(i)));
}
```

Listing 3.10. Configuration of continuous traffic generation using the OnOff application

3.4. Data Flow

This section describes the complete data flow used in this thesis, including the execution of simulation scenarios, collection of raw results, statistical post-processing and generation of plots. The purpose of this section is to ensure transparency and reproducibility of the presented results by clearly documenting how simulation data are produced, processed and analysed.

3.4.1. Simulation Execution

All simulation scenarios were implemented using the `ns-3` network simulator and executed from the command line. Each scenario was compiled and run using the `ns3 run` interface, with key parameters passed as command-line arguments. An example command used to run a validation scenario is shown in Listing 3.11.

```
./ns3 run "scratch/wifi-stats --nWifi=1 --maxPackets=1 --interval=1
    --simulationTime=2"
```

Listing 3.11. Example command-line execution of an ns-3 validation scenario

In this command, `nWifi` specifies the number of transmitting stations, `maxPackets` defines the maximum number of packets generated by each station and `interval` sets the time interval between consecutive packet transmissions. The `simulationTime` parameter determines the total duration of the simulation in seconds, during which traffic generation and statistical measurements are performed. Using command-line parameters in this form enables controlled and repeatable experimentation while keeping the simulation source code unchanged.

During execution, simulation results are printed directly to the terminal, including aggregated throughput values as well as detailed PHY- and MAC-layer statistics. In addition, selected scenarios automatically export raw measurement data to CSV files during simulation runtime, enabling further offline processing.

3.4.2. Result Collection and Validation

Depending on the scenario, results are collected using three complementary mechanisms: textual output printed to the terminal, CSV files generated directly by the simulation code (e.g., delay samples or empirical CDF data) and PCAP files capturing PHY-layer traffic. CSV files are used as the primary input for numerical analysis and plot generation, while PCAP traces serve as an independent validation mechanism. By inspecting frame sequences, timing relations and control exchanges in PCAP files, the correctness of the measured statistics could be cross-checked against the expected IEEE 802.11 protocol behaviour.

3.4.3. Randomness and Repeated Simulations

To improve statistical reliability, selected scenarios were executed multiple times using different random number generator (RNG) seeds. In `ns-3`, RNG seeds control stochastic elements such as random backoff selection, ensuring that each simulation run represents an independent realisation of the same experimental setup.

For the collision probability and saturated performance scenarios, each configuration was simulated ten times using different RNG seeds (set using the `--RngRun` command line option). The resulting metrics were aggregated and used to compute 95% confidence intervals, providing a quantitative measure of variability and robustness of the observed results. For delay measurements, three independent simulation runs were performed for each configuration and delay samples from all runs were combined to increase the total number of observations. This was sufficient to obtain smooth empirical delay distributions.

3.4.4. Statistical Processing and Plot Generation

Raw data exported from simulations is processed offline. For collision probability and performance metrics, averages and confidence intervals are computed across repeated runs, assuming an approximately normal distribution of the measured values. For delay analysis, individual delay samples are

sorted to construct empirical cumulative distribution functions (CDFs), enabling a detailed characterisation of delay variability and tail behaviour under saturated conditions.

All plots presented in this thesis are generated using dedicated Python scripts. CSV files produced by the simulations are imported into Python, where numerical processing and visualisation are performed. This separation between simulation execution and post-processing ensures clarity of the simulation code, consistent handling of results across scenarios and reproducibility of the generated figures discussed in Chapter 4.

3.4.5. Implementation Challenges

The development of the simulation scenarios presented in this thesis required addressing a number of practical challenges related to both implementation and scenario configuration. These difficulties stemmed primarily from the complexity of the `ns-3` Wi-Fi model and the need to construct experiments that make lower-layer protocol behaviour clearly observable and interpretable.

One of the key challenges was preparing simulation setups that expose specific PHY- and MAC-layer mechanisms in a controlled manner. Early experiments based directly on IEEE 802.11ax showed that advanced features such as frame aggregation, Block Acknowledgement and internal optimisations often obscure individual protocol events. While these mechanisms are essential for realistic performance evaluation, they make it difficult to trace fundamental operations such as channel access, control frame exchanges, or precise transmission timing. For this reason, a deliberate step back to a legacy IEEE 802.11a scenario was taken during the validation phase. This simplified configuration made it possible to clearly observe RTS/CTS handshakes, ACK behaviour and inter-frame spacing before extending the analysis to more complex IEEE 802.11ax setups.

Another important challenge was the verification of measurement correctness. Relying solely on numerical counters printed by the simulator was not sufficient to ensure that collected statistics accurately reflected the intended protocol behaviour. As a result, extensive inspection of PCAP traces was required throughout the development process. Analysing captured frames and their timestamps allowed verification of frame ordering, transmission durations, retransmission events and control traffic and proved essential for confirming the validity of both PHY- and MAC-layer measurements as well as delay calculations.

From an implementation perspective, integrating multiple measurement mechanisms within a single scenario also required careful design. The use of built-in helpers such as `WifiTxStatsHelper`, combined with PHY trace callbacks and custom delay tracking logic, introduced the risk of overlapping or inconsistent statistics. To mitigate this, particular attention was paid to clearly separating data collection from post-processing. Simulation code was kept focused on event tracing and logging, while statistical analysis and result aggregation were performed offline.

Overall, resolving these challenges involved iterative refinement of scenario configurations, repeated validation using independent data sources and continuous alignment between simulation design and research objectives. This process ultimately resulted in a set of simulation scenarios that are both technically reliable and suitable for the detailed performance analysis presented in Chapter 4.

4. Simulation Results

All results presented in this chapter are obtained using the link-layer measurement framework introduced in Chapter 3. The first part is dedicated to validation tests, the second – to performance analysis. We consistently collect data as follows: PHY reception statistics and frame-type classifications are collected with `WifiPhyRxTraceHelper`, MAC transmission and retransmission behaviour is analysed using `WifiTxStatsHelper` and channel occupancy is measured with `WifiCoTraceHelper`. Selected timing-related metrics, such as transmission timestamps, are collected using custom MAC- and PHY-layer trace callbacks.

4.1. Validation Tests

The purpose of the validation tests is to confirm that the implemented measurement mechanisms correctly reflect the expected IEEE 802.11 protocol behaviour before they are used in more complex scenarios. The tests are organised from simple to slightly more involved cases, starting with contention-free single-station transmissions and gradually extending to packet bursts and multi-station setups. This progression makes it possible to verify timing relations, retransmission behaviour and collision-related effects in isolation and to ensure that the collected PHY- and MAC-layer statistics remain consistent with theoretical expectations.

4.1.1. Single 802.11a Transmission

This first test is intentionally simple. Its purpose is to verify that the basic PHY and MAC mechanisms in ns-3 behave as expected before moving to more complex scenarios. The setup uses a single transmitter and the legacy IEEE 802.11a standard. This avoids modern features such as aggregation and removes contention and collisions, which makes it easier to spot issues in either the configuration or the measurement code.

Although the logical topology follows the AP–STA structure, AP functionality is disabled for this test. Beaconing and other AP-specific behaviour are turned off and both nodes are configured in ad-hoc mode. This prevents periodic management traffic (e.g., beacons) from affecting timing, medium occupancy and PHY/MAC statistics. For clarity, the nodes are still referred to as STA (sender) and AP (receiver) in the discussion. The main simulation parameters are listed in Table 4.1.

Parameter	Value
Number of transmitted packets	1
Simulation time	2 s
Wi-Fi standard	IEEE 802.11a
Channel width	20 MHz
Modulation and coding scheme (MCS)	7
Frequency band	5 GHz
Guard interval	800 ns

Table 4.1. Simulation parameters for the single 802.11a transmission test

4.1.1.1. PHY Layer Time Share

PHY State	STA	AP
IDLE [s]	1.000090	1.000090
CCA_BUSY [s]	0.000032	0.000032
TX [s]	0.002124	0.000088
RX [s]	0.000048	0.002084
OTHER [s]	0	0

Table 4.2. PHY layer time share for the single 802.11a transmission test

The time-share values in Table 4.2 come from `WifiPhyStateHelper`, which records how long each node stays in the IDLE, CCA_BUSY, TX and RX states. In this scenario, the numbers should match what is expected for a single RTS/CTS/DATA/ACK exchange.

Both nodes remain in IDLE for roughly one second, which covers the active part of the run from the first relevant event to the end of the last exchange. Since only one data packet is sent, PHY activity occupies only a small fraction of that interval. The CCA_BUSY state lasts only a few tens of microseconds, reflecting short periods where each node senses the channel as busy while the other one is transmitting.

It is important to note that the time between the last recorded PHY or MAC event and the end of the simulation is not included in the reported state durations. In ns-3, the simulation stop time does not interrupt an ongoing event and periods of inactivity after the final transmission are therefore not accounted for in the IDLE statistics. This behaviour is specific to the current ns-3 implementation and may change in future versions.

The TX/RX durations differ between STA and AP, as expected. The STA transmits RTS and DATA, so its TX time (2124 µs) is much larger than the AP's TX time (88 µs), since the AP sends only the short CTS and ACK frames. The RX times show the opposite: the STA receives CTS and ACK (48 µs in total), while the AP receives RTS and the much longer DATA frame, resulting in about 2084 µs spent in RX.

Finally, the difference between TX time on one node and RX time on the other is 40 µs (e.g., AP TX is 88 µs, while STA RX is 48 µs). This can be explained by the preamble and PHY headers, which last 20 µs per frame. Each node receives two frames, so 40 µs is not counted as RX time in these statistics.

4.1.1.2. PHY PSDU Statistics

Metric	Value
PHY header failures	0
RX while decoding preamble	0
RX aborted by TX	0
PSDU successes	4
PSDU failures	0
Total RX events (PSDU-based)	4
Total PHY error/abort events	0

Table 4.3. PHY PSDU decoding statistics for the single 802.11a transmission test

Table 4.3 shows PSDU-level reception results collected with `WifiPhyRxTraceHelper`. No PHY header failures are reported, which means all PHY headers were decoded correctly. There are also no events indicating overlapping reception during preamble decoding and no cases where reception was aborted because the receiver started transmitting.

Most importantly, the test reports four PSDU successes and zero PSDU failures. This matches the expected RTS/CTS/DATA/ACK exchange: four frames are received and decoded correctly and no PHY error/abort events occur during the run.

4.1.1.3. PHY RX/TX Frame Classification

Metric	Value
Transmitted DATA frames	1
Transmitted ACK frames	1
Transmitted RTS frames	1
Transmitted CTS frames	1
Transmitted BlockAck frames	0
Received DATA frames	1
Received ACK frames	1
Received RTS frames	1
Received CTS frames	1
Received BlockAck frames	0
Dropped DATA frames	0
Dropped ACK frames	0
Dropped RTS frames	0
Dropped CTS frames	0
Dropped BlockAck frames	0

Table 4.4. PHY RX/TX frame classification for the single 802.11a transmission test

Table 4.4 lists the frames observed at the PHY layer, obtained from `WifiPhyRxTraceHelper` together with frame-type classification. In this test, exactly one RTS, one CTS, one DATA and one ACK are transmitted in total and each of them is also received exactly once. This is the expected RTS/CTS handshake followed by DATA transmission and a single ACK frame.

The BlockAck counters remain zero, which is consistent with the lack of aggregation in this 802.11a scenario. Finally, all drop counters are zero, indicating that no frames were discarded at the PHY layer.

4.1.1.4. Advanced PHY Metrics

Metric	STA	AP
Channel utilization [%]	0.07	0.07
PSDU success per receiver	2	2
PHY error & abort events (per MAC)	0	0

Table 4.5. Advanced PHY metrics for the single 802.11a transmission test

Table 4.5 summarises a few additional PHY-level indicators collected with `WifiPhyStateHelper` (PHY state durations) and `WifiCoTraceHelper` (channel occupancy / COT). Channel utilisation is only about 0.07% for both nodes, which is exactly what would be

expected here: the scenario contains just a single RTS/CTS/DATA/ACK exchange and the medium is idle for almost the whole run. The “PSDU success per receiver” value is equal to two for both nodes. This means the STA successfully decodes CTS and ACK, while the AP successfully decodes RTS and DATA. This matches the PSDU success counters shown earlier. Finally, there are no PHY error or abort events. In other words, nothing unusual happens at the PHY layer: no decoding failures and no cases where a reception had to be interrupted.

4.1.1.5. MAC Layer MPDU Statistics

Metric	Value
MPDU successes	1
MPDU failures	0
MPDU retransmissions	0
TX attempts (aggregate)	0
TX retries (aggregate)	0
TX failures (aggregate)	0
Success rate	100%
Failure rate	0%

Table 4.6. MAC-layer MPDU statistics for the single 802.11a transmission test

Table 4.6 shows MAC-layer MPDU statistics reported by `WifiTxStatsHelper` for the single data packet. The outcome is as clean as it gets: one MPDU is delivered successfully, with no retransmissions and no failures. The “aggregate” counters remain zero in this test. This is expected, because the scenario does not use aggregation mechanisms; the relevant activity is captured by the MPDU-level counters above. Overall, from the MAC point of view, the packet goes through on the first try and nothing gets dropped.

4.1.1.6. Per-Sender / Per-Receiver Statistics

Metric	STA	AP
Packets sent	1	0
Packets received	0	1
First RX timestamp [s]	1.000146	1.000086
Last RX timestamp [s]	1.002294	1.002234
Data transfer duration [s]	0.002148	0.002148
TX Throughput [Mbit/s]	0.005888	0.000000
RX Throughput [Mbit/s]	0.000000	0.006032

Table 4.7. Per-sender and per-receiver statistics for the single 802.11a transmission test

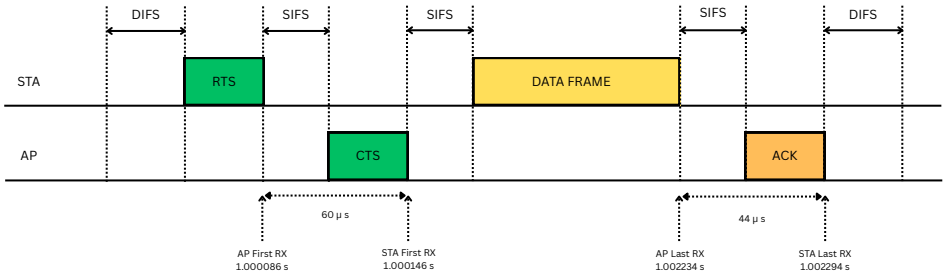

Figure 4.1. Timing of events in the single 802.11a transmission test.

Table 4.7 and Figure 4.1 show the basic sender/receiver timeline obtained from custom MAC-layer trace callbacks. The STA sends one application packet and the AP receives it; the traffic is unidirectional, so the STA does not receive any application data. The first RX timestamps differ by about 60 µs. This offset comes from the SIFS interval plus the CTS transmission. The same ∼60 µs offset appears for the last RX timestamps and corresponds to SIFS plus the ACK transmission. The reported data transfer duration of 0.002148 s represents the transmission time of the DATA frame itself and is identical for both nodes. The transmitted packet size is 1508 bytes, consisting of 1472 bytes payload, 8 bytes UDP header, 20 bytes IP header and 8 bytes LLC/SNAP header. This packet size directly determines the throughput values shown in Table 4.7, which are calculated from the successful transmission of a single application packet over the measured data transfer duration.

4.1.1.7. Delay and Jitter Results

Metric	Value
Average delay	2234.093 µs
Jitter	0

Table 4.8. Delay and jitter metrics for the single 802.11a transmission test

The delay values in Table 4.8 are computed from timestamps collected with custom MAC- and PHY-layer trace callbacks. In this test there is only one application packet, so we can report its end-to-end delay, but there is no meaningful jitter sample (jitter needs at least two packets).

The measured delay (2234 µs) corresponds to the full RTS/CTS/DATA/ACK exchange in IEEE 802.11a. Most of it comes from the DATA transmission itself; the rest is overhead introduced by RTS/CTS/ACK, SIFS intervals and PHY processing.

4.1.1.8. Summary

This first validation test behaves exactly as expected for a single, contention-free 802.11a RTS/CTS/DATA/ACK exchange. There are no PHY drops, no MAC retransmissions and the frame sequence and timing are consistent across PHY and MAC measurements. This provides a clean baseline for the more complex scenarios that follow.

4.1.2. 802.11a Packet Burst

This test repeats the single-station IEEE 802.11a setup, but instead of sending one packet the transmitter sends a short burst of ten packets in the same run. The idea is simple: if the configuration and measurements are correct, the same RTS/CTS/DATA/ACK pattern should just repeat multiple times, without introducing any timing artefacts.

Compared to the single-packet case, the run now contains a sequence of consecutive RTS/CTS/-DATA/ACK exchanges generated by the same station. Even though there are more exchanges, the structure of each individual cycle should remain unchanged and follow the standard 802.11a access procedure.

The *First RX* and *Last RX* timestamps show that packets are sent one after another with the expected SIFS and DIFS spacing between access attempts. No unexpected gaps or irregular timing behaviour appears in the traces. From the PHY point of view, all PSDUs are received correctly and at the MAC layer no retransmissions are triggered, which is exactly what is expected in a contention-free single-transmitter scenario.

Overall, the burst case mainly extends the single-packet test in time. It does not change the per-packet behaviour, which confirms that the PHY and MAC measurement mechanisms remain stable also for multiple consecutive transmissions.

4.1.3. Single 802.11ax Transmission

This test applies the same baseline check to IEEE 802.11ax. A single station transmits one DATA frame and the goal is to verify that the timing and the measurement mechanisms behave correctly for 802.11ax under contention-free conditions.

The observed exchange follows the expected access procedure (medium sensing, inter-frame spaces, DATA and ACK). The *First RX* and *Last RX* timestamps at the sender and receiver line up as expected, with no irregular gaps.

Standard	Node	Data transfer duration [s]
IEEE 802.11a	STA	0.002148
IEEE 802.11a	AP	0.002148
IEEE 802.11ax	STA	0.000741
IEEE 802.11ax	AP	0.000757

Table 4.9. Comparison of data transfer duration for a single DATA transmission in IEEE 802.11a and IEEE 802.11ax

Table 4.9 compares the data transfer duration measured for a single DATA transmission in IEEE 802.11a and IEEE 802.11ax. Although the handshake structure is identical in both standards, the measured data transfer durations are significantly shorter in the IEEE 802.11ax case. This reduction results from different PHY timing parameters and higher effective data rates used by IEEE 802.11ax. A small difference between the durations observed at the STA and the AP in the IEEE 802.11ax case can be attributed to PHY-layer reception timing, in particular the preamble duration and receiver-side processing, which slightly shift the effective RX timestamp. No PHY failures or MAC retransmissions are observed in either case, confirming fully reliable operation under contention-free conditions.

4.1.4. 802.11ax Packet Burst

Finally, the single-station IEEE 802.11ax scenario is repeated with a ten-packet burst. The run contains a sequence of repeated DATA/ACK exchanges and is used to verify whether the measurement mechanisms remain consistent when the same transmission cycle is executed multiple times under contention-free conditions.

Metric	STA	AP
First RX timestamp [s]	1.000118	1.000058
Last RX timestamp [s]	10.000204	10.000160

Table 4.10. First and last RX timestamps for the 802.11ax packet burst scenario (10 packets)

Table 4.10 presents the first and last reception timestamps observed at the STA and AP. The timestamps show that packets are transmitted sequentially over the entire duration of the burst, with no unexpected gaps between consecutive transmissions. The small offset between STA and AP timestamps is consistent with the expected SIFS intervals and control frame transmissions and confirms correct temporal alignment of PHY- and MAC-layer measurements.

Metric	**802.11a**	**802.11ax**
Average delay [μs]	556.4	228.4
Jitter [μs]	621.3	218.3

Table 4.11. Comparison of burst transmission results (10 packets) for IEEE 802.11a and IEEE 802.11ax

In addition to timing consistency, the burst transmission results allow a direct comparison of delay-related metrics between IEEE 802.11a and IEEE 802.11ax. Table 4.11 summarises the average end-to-end delay and jitter measured for ten-packet bursts in both standards. The results show that IEEE 802.11ax achieves significantly lower average delay and jitter compared to IEEE 802.11a. The reduction in both delay and jitter in the IEEE 802.11ax case is a direct consequence of shorter PHY transmission times and more efficient timing parameters compared to IEEE 802.11a. Since the scenario is contention-free and no retransmissions occur, the observed differences are dominated by PHY-layer characteristics rather than MAC-layer effects.

Overall, the burst experiment extends the single-packet validation by demonstrating that PHY- and MAC-layer measurements remain stable and consistent over repeated transmissions. The absence of retransmissions, packet losses and timing anomalies confirms the correctness of the measurement framework and provides a reliable basis for more complex multi-station and saturated performance scenarios analysed later in this thesis.

4.1.5. Collision Probability

This validation step examines collision probability in a multi-station IEEE 802.11ax scenario. Collision probability is a key MAC-layer indicator of contention intensity: as the number of transmitting stations increases, simultaneous transmission attempts become more likely, leading to collisions, retransmissions and reduced efficiency.

While it is well known that ns-3 reproduces the expected collision behaviour of 802.11, the purpose of this experiment is to verify whether the proposed measurements provide values consistent with a reference analytical contention model. The same simulation setup is therefore executed for different numbers of transmitting stations and the resulting collision probability is compared with the analytical prediction.

Ns-3 does not expose collisions as explicit MAC-layer events. For this reason, collision probability is estimated indirectly using MPDU-level transmission statistics collected with `WifiTxStatsHelper`.

For each simulation run, three counters are recorded: the number of successfully transmitted MPDUs (N_{succ}), the number of retransmissions (N_{retry}) and the number of MPDUs dropped but only on account of exceeding the retry limit ($N_{\text{fail, retry}}$)[1]. Based on these values, collision probability is computed as:

$$p_{\text{coll}} = \frac{N_{\text{retry}} + N_{\text{fail, retry}}}{N_{\text{succ}} + N_{\text{retry}} + N_{\text{fail, retry}}}. \tag{4.1}$$

This definition treats retransmissions and retry-limit drops as indications of failed initial transmission attempts, which in saturated multi-station scenarios are dominated by collisions. For each network size, ten independent simulation runs with different random seeds are performed and the reported collision probability is obtained as the average across these runs.

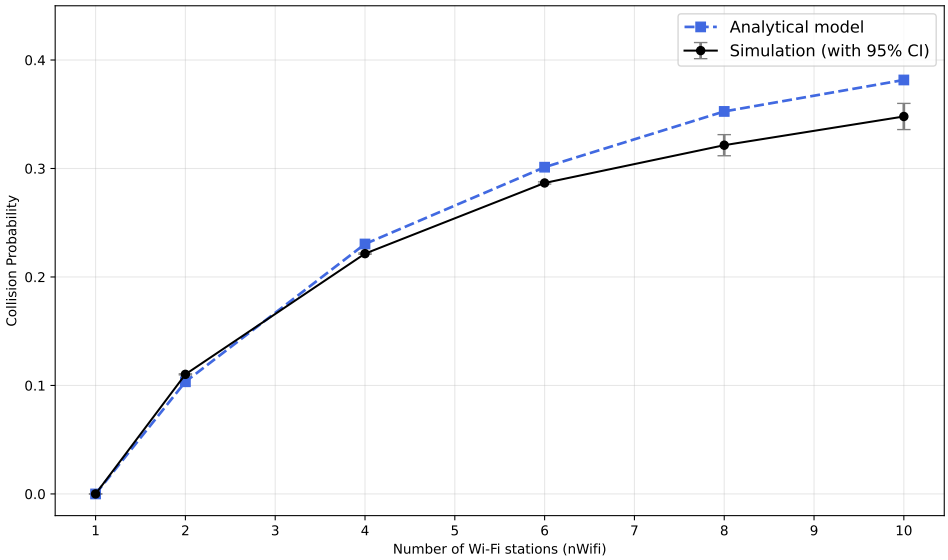

Figure 4.2. Collision probability as a function of the number of Wi-Fi stations: ns-3 results compared with the analytical model.

Figure 4.2 shows collision probability as a function of the number of active stations, together with the analytical reference model. As expected, collision probability is zero for a single station and increases monotonically as more stations contend for access to the channel.

Across all considered network sizes, the simulation results closely follow the analytical curve. Small deviations are expected due to the random backoff process and the finite simulation duration. Overall, the agreement confirms that contention and collision behaviour in ns-3 is consistent with the analytical model, supporting the correctness of the measurement framework used in this thesis.

[1]Frames can be dropped for other reasons as well, such as due to queue blocking, but this is not relevant to calculating the collision probability.

4.2. Performance Analysis

This section highlights the applicability of the measurement framework in typical 802.11 performance analyses. All results presented in this section were obtained using the same measurement framework introduced in Chapter 3. This ensures full consistency between the validation tests and the saturation performance analysis, allowing all observed trends to be directly linked to well-defined PHY- and MAC-layer mechanisms.

4.2.1. Channel Access Delay

This section investigates the statistical distribution of packet delay in an IEEE 802.11ax network operating under saturated traffic conditions. Instead of focusing only on average delay values, the cumulative distribution function (CDF) is used to obtain a more detailed view of how individual packet delays are spread over time. This allows identifying not only typical delays, but also worst-case performance. In this experiment, all transmitting stations generate traffic at a constant rate such that the wireless medium remains continuously saturated. As a result, packets always compete for channel access and the observed delays are dominated by MAC-layer mechanisms such as random backoff, contention between stations, retransmissions, and queueing at the transmitter.

Two types of delay are analyzed. The *end-to-end delay* represents the total time from packet generation at the transmitter to successful reception at the receiver. The *channel-access delay* reflects only the time a packet spends waiting for access to the wireless medium before it can be transmitted.

For each simulation run, individual delay samples were collected using dedicated MAC- and PHY-layer trace callbacks. The end-to-end delay was computed from timestamps recorded at the `MacTx` and `MacRx` trace points as the time difference between packet transmission and successful reception. Channel-access delay was derived from PHY-layer reception events captured via the `MonitorSnifferRx` trace and computed as the time interval between two consecutive successfully received DATA frames originating from the same transmitting station. Jitter was calculated as the variation between successive end-to-end delay samples. Based on the recorded delay values, empirical cumulative distribution functions (CDFs) were constructed by sorting the samples and assigning each one its cumulative probability.

To improve the statistical reliability of the results, the experiment was repeated using three independent random number generator seeds. The final CDF curves presented in the following figure were obtained by aggregating the delay samples from all three simulation runs.

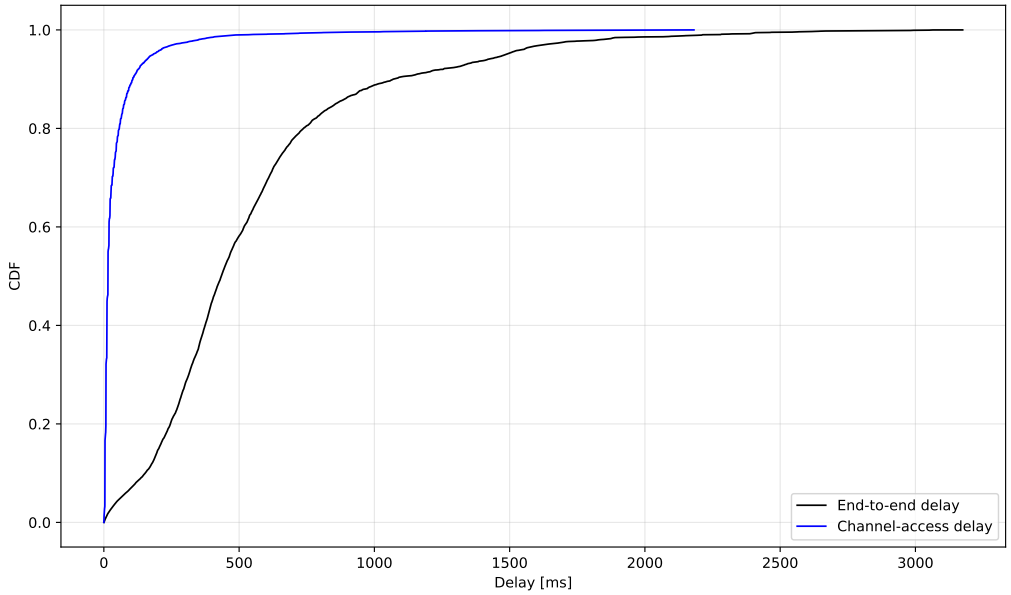

Figure 4.3. CDF comparison of end-to-end delay and channel-access delay under saturation.

Figure 4.3 presents the empirical cumulative distribution functions of the channel-access and end-to-end delays measured under saturated traffic conditions. The horizontal axis represents packet delay in milliseconds, while the vertical axis shows the cumulative probability.

The channel-access delay curve rises very rapidly and reaches high probability values for very small delays. This indicates that most packets obtain access to the wireless medium shortly after becoming ready for transmission. Only a small fraction of packets experience noticeably longer access delays, which are mainly caused by random backoff and contention between stations.

In contrast, the end-to-end delay CDF increases much more gradually and extends to significantly larger delay values. Unlike the channel-access delay, this metric includes not only the waiting time for medium access, but also queueing effects, processing at the receiver and possible retransmissions. As a result, its distribution exhibits a much longer tail.

The clear separation between the two curves shows that the longest packet delays observed in the system are not caused by medium access alone. Instead, they are dominated by queueing and repeated contention under saturation. This highlights the strong impact of MAC-layer dynamics on the end-to-end performance of a heavily loaded IEEE 802.11ax network.

4.2.2. Performance under Saturation

This experiment investigates how the performance of an IEEE 802.11ax network changes with the number of active transmitting stations under saturated traffic conditions. All stations generate traffic at a constant rate so that the wireless medium remains fully saturated throughout the entire simulation. All scenarios are based on the IEEE 802.11ax standard operating at 5 GHz with a channel bandwidth of

20 MHz. The total offered traffic load is fixed at 150 Mbit/s and shared among the active stations, which guarantees permanent saturation independently of the network size.

The simulations were carried out for five network sizes: 1, 2, 4, 8 and 16 stations. For each configuration, ten independent simulation runs with different random number generator seeds were performed in order to reduce the influence of the stochastic MAC backoff process. Each run lasted 100 s, which was sufficient to reach steady-state operating conditions.

All performance metrics presented in the following figures are shown as mean values computed over all simulation runs. In addition, 95% confidence intervals are included in the plots in order to visualise the statistical uncertainty of the results.

To obtain a comprehensive view of network performance, different measurement mechanisms available in the simulator were deliberately used. PHY-layer reliability is assessed using the percentage of failed PSDUs obtained from the `WifiPhyRxTraceHelper`, which captures PHY-layer reception failures via trace callbacks. MAC-layer behaviour is evaluated through the percentage of MPDU retransmissions and failures reported by the `WifiTxStatsHelper`. Finally, channel occupancy dynamics are analysed using the Channel Occupancy Time (COT) metric extracted with the `WifiCoTraceHelper`.

4.2.2.1. PSDU Failure Rate

The PSDU failure rate is defined as the fraction of physical layer protocol data units (PSDUs) that are not successfully received at the PHY layer. In practice, this metric reflects the impact of collisions and reception errors and provides an indication of PHY-layer reliability under contention.

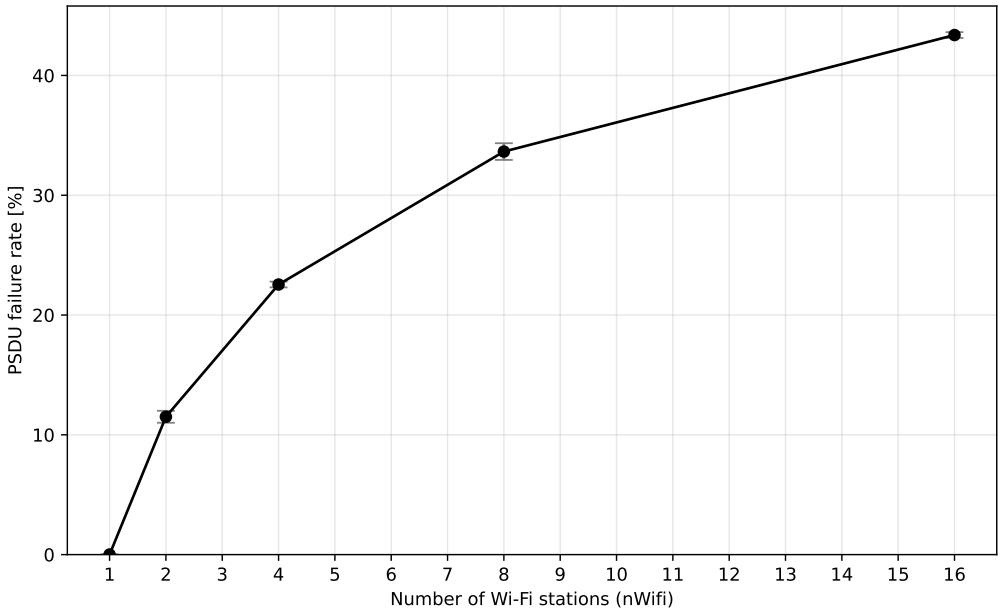

Figure 4.4. PSDU failure rate as a function of the number of stations under saturation.

Figure 4.4 illustrates how the PSDU failure rate changes with the number of active transmitting stations under saturated traffic conditions. For a single station, no PSDU failures are observed, which is

fully expected in the absence of contention, as all stations are located in close proximity and the scenario does not involve signal attenuation effects. Once a second station is introduced, the failure rate increases visibly, indicating the appearance of collisions caused by competing access to the wireless medium.

As the number of stations grows further to 4, 8, and 16, the PSDU failure rate increases in a clear monotonic manner. This trend directly reflects the increasing level of contention: a larger number of simultaneous transmitters leads to more frequent overlapping transmissions and, consequently, a higher probability of unsuccessful PSDU receptions. The observed behaviour confirms the strong impact of network density on PHY-layer reliability. Under saturated conditions, collisions rapidly become the dominant factor limiting successful frame reception in dense IEEE 802.11ax networks.

4.2.2.2. MPDU Failure Rate

The MPDU failure rate describes the fraction of MAC protocol data units that are not successfully delivered due to exceeding the retry limit at the MAC layer. This metric reflects the reliability of MAC-level transmission under contention and complements PHY-layer failure statistics.

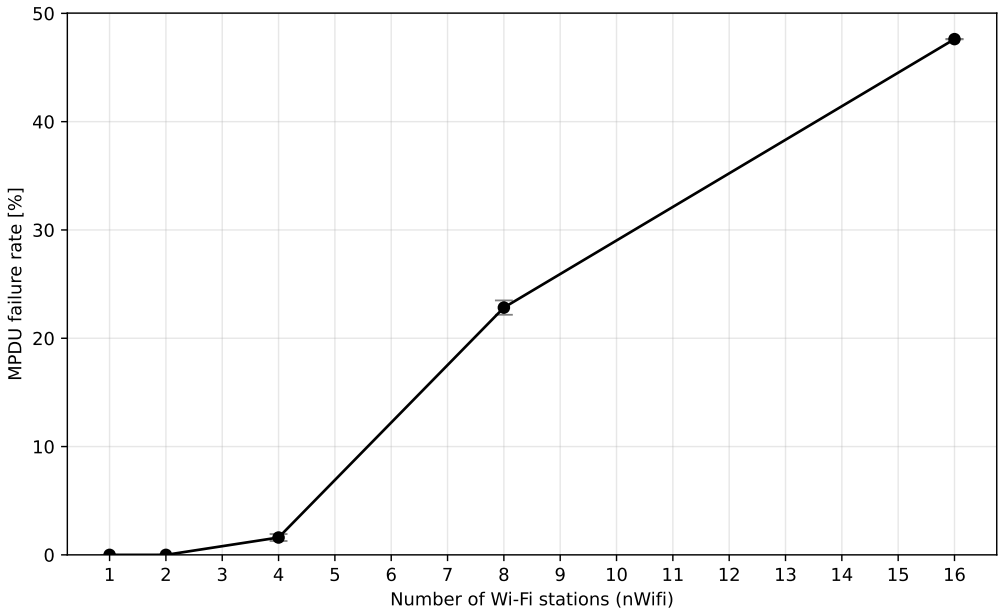

Figure 4.5. MPDU failure rate as a function of the number of stations under saturation.

Figure 4.5 presents the MPDU failure rate as a function of the number of transmitting stations under saturated conditions. For a single station, no MPDU failures are observed, which is consistent with the PSDU results and follows directly from the lack of contention. A small increase appears for two stations, while a rapid growth of the failure rate is visible already for four active nodes.

For larger network sizes, the MPDU failure rate increases sharply and reaches very high values. This confirms that, under saturation, a significant fraction of MAC-layer transmissions fails due to repeated collisions and exceeded retry limits. The observed trend is fully consistent with the PSDU failure behaviour discussed earlier: as the number of stations grows, collisions propagate from the PHY layer

to the MAC layer, leading not only to reception failures but also to an increasing number of dropped MPDUs after unsuccessful retransmission attempts.

4.2.2.3. MPDU Retransmission Rate

The MPDU retransmission rate indicates how frequently MPDUs are retransmitted when acknowledgements are missing. This metric reflects MAC-layer behaviour under contention and highlights how often transmission attempts have to be repeated before success or failure.

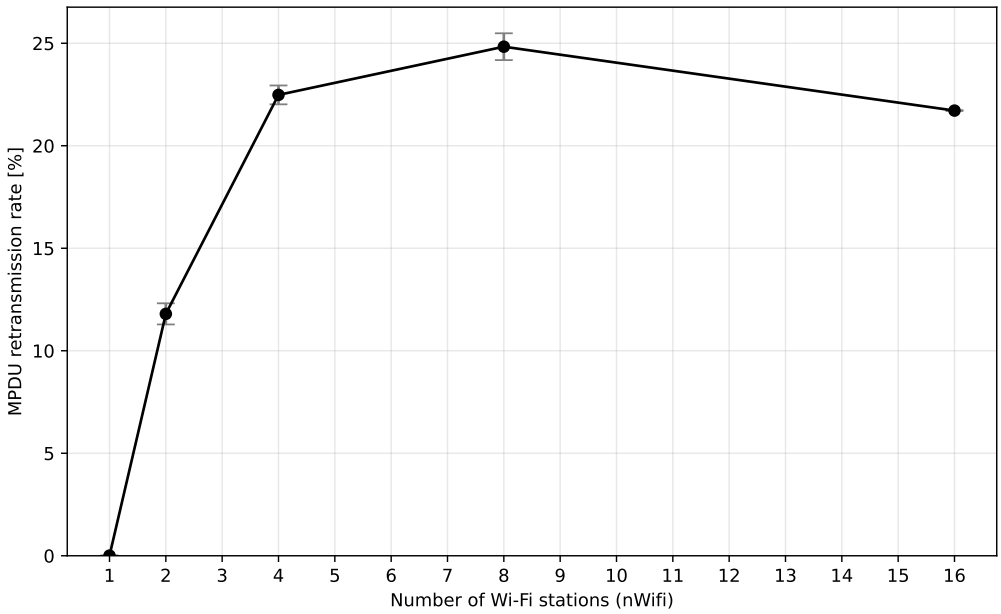

Figure 4.6. MPDU retransmission rate as a function of the number of stations under saturation.

Figure 4.6 presents the MPDU retransmission rate under saturated conditions for different numbers of transmitting stations. As expected, the retransmission rate grows rapidly when the number of stations increases from one to four, reflecting the higher likelihood of collisions caused by intensified contention in the channel. The rate reaches its maximum at eight stations, where collisions are frequent yet the MAC layer is still able to perform several retransmission attempts before reaching the retry limit.

For a high number of stations, the retransmission rate does not continue to increase and may even slightly decrease. This behaviour does not indicate improved channel conditions. Instead, it is a consequence of extreme network congestion. In very dense scenarios, MPDUs may experience long waiting times in the MAC transmission queue and can be dropped due to queue management limits before reaching the retry limit. As a result, such frames are classified as MPDU failures rather than retransmissions, which reduces the measured retransmission rate while simultaneously increasing the MPDU failure rate.

Overall, the results illustrate the expected behaviour of the IEEE 802.11ax MAC under saturation: retransmissions dominate in moderately loaded networks, but in extremely dense scenarios they are gradually replaced by outright transmission failures caused by MAC queue management policies.

4.2.2.4. Aggregate Throughput

Aggregate throughput is defined as the sum of the throughputs achieved by all active stations. It provides a direct measure of the total data rate sustained by the network.

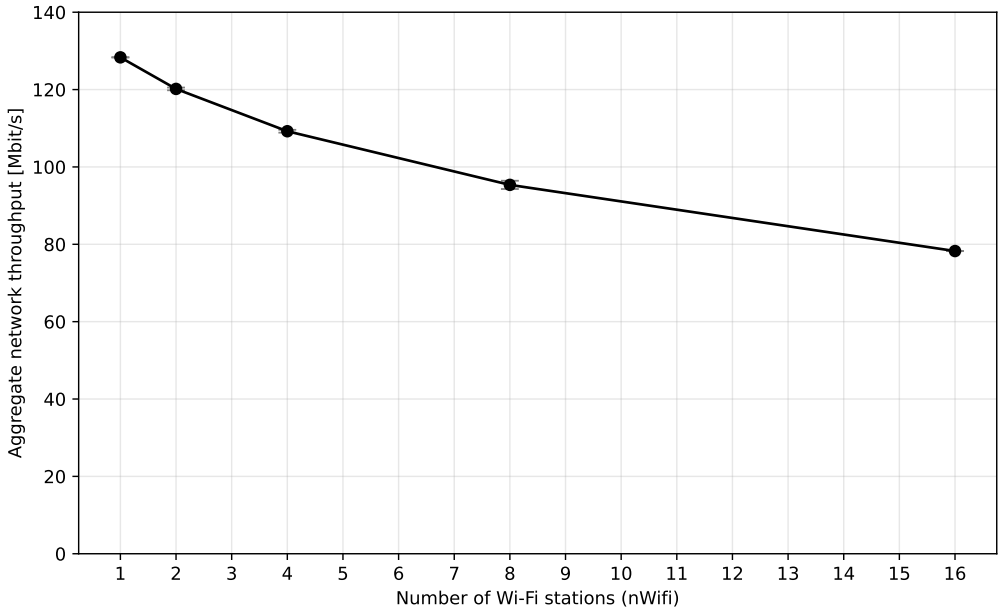

Figure 4.7. Aggregate network throughput as a function of the number of stations under saturation.

Figure 4.7 presents the aggregate network throughput measured under saturated traffic conditions as the number of transmitting stations increases. The offered load in each scenario is fixed at 150 Mbit/s, divided equally among the active stations. However, the achieved throughput is naturally lower than the offered load due to protocol overheads such as MAC/PHY headers, interframe spaces (SIFS, DIFS), acknowledgementsand backoff periods.

For a single station, the aggregate throughput reaches approximately 130 Mbit/s. This value is consistent with the PHY data rate used in the simulation (approximately 143 Mbit/s for IEEE 802.11ax with a 20 MHz channel and a single spatial stream). The remaining difference is explained by inevitable MAC and PHY overheads, including frame headers, inter-frame spaces and acknowledgement transmissions, which consume airtime even in the absence of contention.

As additional stations are introduced, the throughput gradually decreases. With two and four stations, the shared channel must handle an increasing number of medium-access attempts. Since RTS/CTS exchange is disabled in this scenario, access conflicts are resolved solely through contention and backoff, which leads to longer waiting times and more frequent access deferrals. These contention-related delays directly reduce the amount of time available for successful data transmission, resulting in a noticeable throughput drop.

At eight and sixteen stations, the decline becomes more pronounced. In these dense scenarios, frequent collisions lead to repeated doubling of contention windows and extended backoff periods, significantly limiting channel efficiency. Although all stations continuously attempt to transmit, a large fraction of the channel time is spent resolving access conflicts rather than carrying useful payload data.

Overall, the decreasing trend in throughput aligns with the expected behaviour of IEEE 802.11ax under saturation. As the number of active nodes increases, contention overhead becomes the dominant limiting factor, causing the aggregate throughput to degrade despite the constant offered load.

4.2.2.5. Transmission Time

Transmission time quantifies how much time the wireless medium is occupied by frame transmissions. This metric provides insight into channel utilisation and how airtime is shared among competing stations.

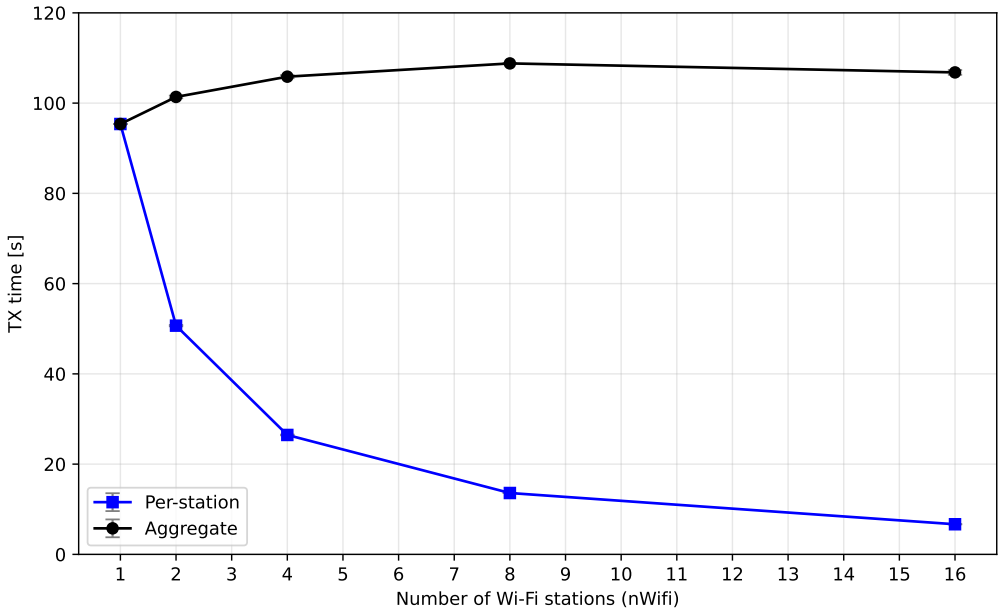

Figure 4.8. Aggregate transmission time as a function of the number of stations under saturation.

Figure 4.8 shows the transmission time as a function of the number of transmitting stations under saturated traffic conditions. Two curves are presented: the aggregate transmission time of the entire network and the average per-station transmission time.

The aggregate transmission time remains relatively stable as the number of stations increases and stays close to the total simulation time. This indicates that the wireless channel is continuously busy regardless of the network size, which confirms that saturation conditions are maintained in all scenarios. Even for larger numbers of stations, the channel is fully utilized, although the efficiency of successful transmissions decreases.

In contrast, the per-station transmission time decreases rapidly with the growing number of stations. For a single station, the transmission time is close to the total simulation duration, as the node has almost exclusive access to the medium. When additional stations are introduced, each node must share the channel with others, which significantly reduces the fraction of time it can occupy the medium.

For eight and sixteen stations, the per-station transmission time becomes very small. This effect is a direct consequence of increased contention, frequent backoff deferrals and collisions, which limit individual access opportunities to the channel. As a result, although the channel remains busy in the aggregate sense, each single transmitter obtains only a small share of the available transmission time.

Overall, this figure clearly illustrates the difference between global channel utilization and individual station access under saturation. While the network as a whole maintains continuous activity, the transmission opportunities available to a single station shrink rapidly as the number of contending nodes increases.

4.2.2.6. Average Delay

Average end-to-end delay is defined as the time elapsed between packet transmission at the sender and its successful reception at the receiver. This metric captures the combined impact of medium access contention, retransmissions, queueing effects and PHY transmission time on packet delivery.

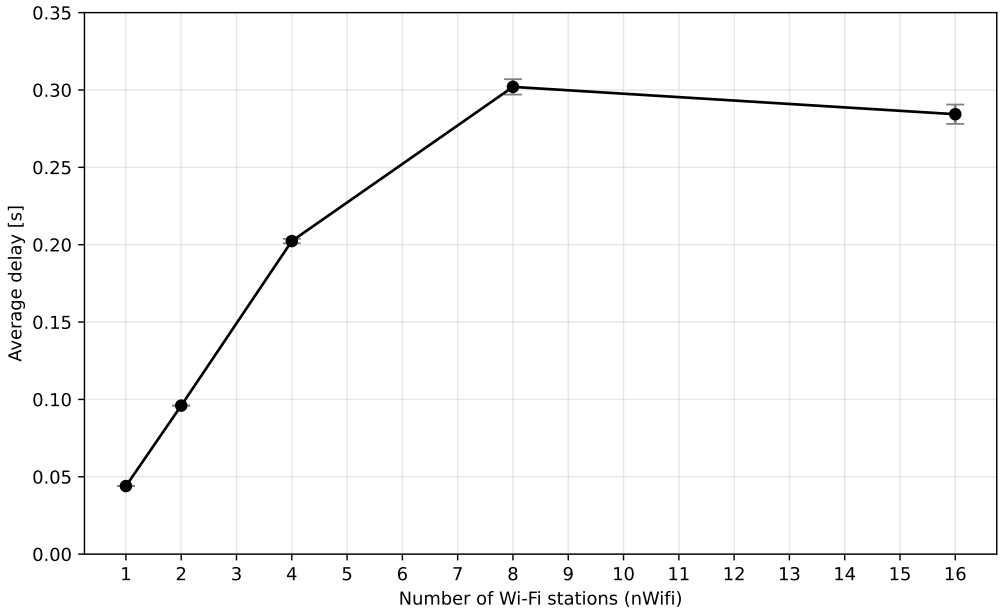

Figure 4.9. Average end-to-end delay as a function of the number of stations under saturation.

Figure 4.9 presents the average end-to-end packet delay as a function of the number of transmitting stations under saturated traffic conditions. For a single station, the average delay is very low, as packets rarely experience contention or queueing. As the number of stations increases to two and four, the delay rises rapidly due to growing contention for medium access and longer backoff times. The highest average

delay is observed for eight stations, where both contention and queueing effects strongly impact packet delivery time.

When the number of stations is increased further to 16, a slight decrease in the average delay is observed. This effect does not indicate improved transmission conditions. Instead, it results from extreme congestion at the MAC layer. Under very dense conditions, many packets experience long waiting times in the MAC transmission queue and are dropped before they can be successfully transmitted, for example due to queueing limits being exceeded. Since only successfully delivered packets contribute to the delay statistics, packets dropped in the queue are not accounted for in the measured delay. As a result, the average delay calculated over the remaining successfully received packets is reduced. Overall, the results confirm that under saturation the average end-to-end delay initially increases with network density, while in extremely congested scenarios it may decrease due to packet drops caused by MAC queue management.

4.2.2.7. Jitter

Jitter describes the variability of packet delay over time, reflecting how much the end-to-end delay fluctuates between consecutive packets. It is an important metric for assessing the regularity of packet delivery, especially under contention and retransmission effects.

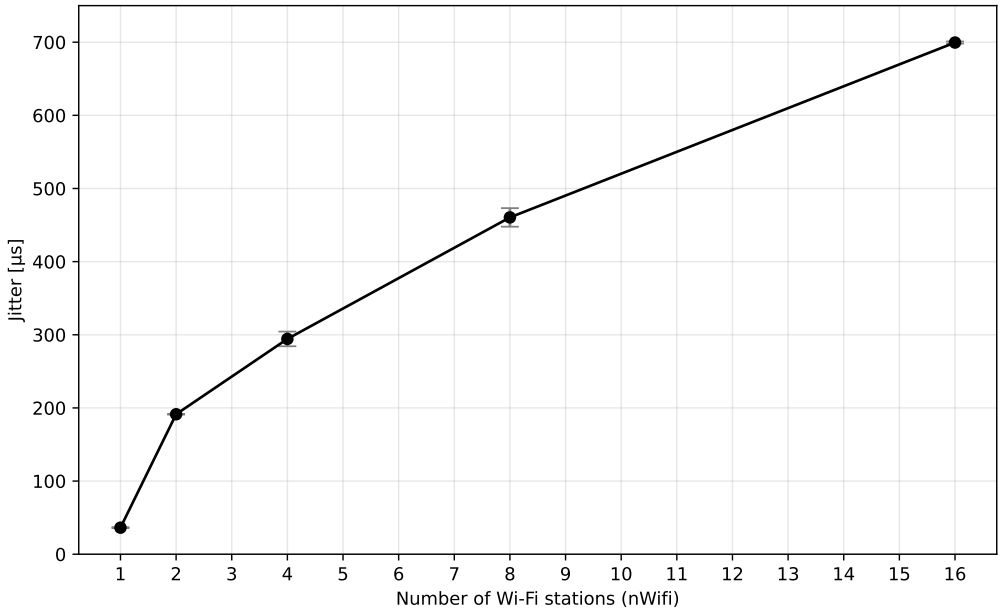

Figure 4.10. Jitter as a function of the number of stations under saturation.

Figure 4.10 presents the jitter measured under saturated IEEE 802.11ax operation for different numbers of transmitting stations. With only one station in the network, jitter remains very low, as packets experience almost no contention and are transmitted at regular intervals. When the number of stations increases to two and four, jitter grows noticeably. This increase is caused by varying backoff durations and occasional retransmissions, which introduce fluctuations in packet inter-arrival times.

For denser networks (8 and 16 stations), the jitter continues to rise. In these scenarios, strong contention and frequent collisions lead to highly irregular access opportunities for each station. As a result, the time between consecutive successful transmissions becomes increasingly variable, producing large jitter values.

4.2.2.8. Discussion

The results presented in this section provide a consistent picture of how an IEEE 802.11ax network behaves under saturated traffic conditions as the number of active stations increases. Across all evaluated metrics, contention emerges as the dominant factor shaping performance.

The PSDU and MPDU failure rates grow steadily with the number of stations, reflecting the increasing likelihood of collisions in dense deployments. The MPDU retransmission rate follows a similar trend, peaking for medium-sized networks before slightly decreasing for the largest configuration. This behaviour is explained by the fact that, with many stations sharing the channel, each individual node transmits less frequently, which naturally limits the number of retransmission attempts per station.

Throughput results confirm the expected saturation behaviour: the aggregate throughput decreases gradually as more stations compete for the medium. Since the offered load is constant, this degradation is driven entirely by contention overhead, collisions and repeated backoff cycles. The transmission time analysis further illustrates this effect: while the aggregate transmission time remains close to the full duration of the simulation, the per-station transmission time drops sharply, showing how airtime is increasingly divided among competing nodes.

Delay and jitter exhibit particularly strong sensitivity to network load. Average packet delay rises quickly as the number of stations increases and queueing effects become more pronounced. Jitter grows even more dramatically, indicating that transmission opportunities become highly irregular in heavily loaded scenarios. This makes jitter one of the most affected performance indicators under saturation.

From a measurement perspective, the results validate the framework developed in this work. The combined use of helper-based statistics (`WifiPhyRxTraceHelper`, `WifiTxStatsHelper`, `WifiCoTraceHelper`) and custom trace callbacks enables detailed, reproducible analysis of contention effects at both the PHY and MAC layers. This confirms that the proposed measurement approach can be reliably used for systematic studies of saturated Wi-Fi networks and for linking high-level performance trends with concrete protocol-level events.

Overall, the combined analysis of PHY-layer reliability, MAC-layer retransmissions, throughput, transmission time, delay and jitter clearly demonstrates how network performance degrades with increasing contention. The results reflect the fundamental trade-off in IEEE 802.11ax networks: while multiple users can share the channel fairly, heavy saturation inevitably leads to reduced reliability, higher latency and increased variability in packet delivery.

5. Summary

This chapter summarises the main contributions and key results of the thesis and discusses their relevance for the analysis of IEEE 802.11 networks. It also points to possible directions for future work based on the proposed measurement framework.

5.1. Contribution

The main contribution of this thesis is a measurement-oriented simulation framework for IEEE 802.11 in ns-3, together with a set of validation tests that confirm the correctness of the collected link-layer metrics. While many ns-3 studies focus mainly on end-to-end results at the network layer, this work puts the emphasis on what happens at the PHY and MAC layers and on how those mechanisms translate into throughput, delay and reliability. The specific contributions are:

- integration of complementary link-layer measurement mechanisms in ns-3, including PHY-layer trace callbacks, MAC-layer transmission statistics and channel occupancy measurements,
- validation of the measurement setup in simple, contention-free scenarios, with explicit checks of PHY/MAC timing relations and frame exchange behaviour,
- performance analysis of IEEE 802.11ax under saturated uplink traffic, covering reliability, retransmissions, throughput, airtime usage, delay and jitter,
- a consistent interpretation approach that links observed performance degradation to contention-driven PHY- and MAC-layer phenomena instead of relying only on network- layer indicators.

All simulation scenarios and the code developed in this thesis are made publicly available as an open-source repository [9].

5.2. Main Conclusions

The results of this thesis lead to three main conclusions. First, the validation tests show that the PHY- and MAC-layer measurement mechanisms used in this work behave as expected. In contention-free scenarios, the observed frame exchanges, timing relations and reliability outcomes match the theoretical

operation of IEEE 802.11 for both a legacy configuration (802.11a) and a modern one (802.11ax). This supports the correctness of the simulation setup and the implemented measurement framework. Importantly, the helper-based measurement components were found to provide results consistent with independent verification methods, including PCAP trace analysis and theoretical timing relations defined in the IEEE 802.11 standard.

Second, under saturated conditions the dominant limitation is MAC-layer contention. As the number of transmitting stations increases, collision probability rises and retransmissions become more frequent. MPDU failures and retry-limit drops increase as well, which together reduce throughput and cause delay and jitter to grow rapidly. These effects cannot be explained convincingly using network-layer metrics alone; they require direct link-layer visibility.

Third, airtime-based measurements are essential for understanding behaviour in dense Wi-Fi scenarios. Channel occupancy and transmission-time indicators show that even when the medium is almost continuously busy, the useful transmission opportunities available to a single station shrink as contention grows. This explains both the drop in per-station throughput and the increasing irregularity of packet delivery.

From a practical perspective, the evaluated helper-based measurement mechanisms proved to be straightforward to integrate into ns-3 simulation scenarios and significantly reduced the implementation effort compared to manual trace processing. The unified interfaces for collecting PHY reception statistics, MAC transmission outcomes and channel occupancy enabled rapid extension of scenarios and facilitated systematic analysis of advanced IEEE 802.11 features, such as those introduced in IEEE 802.11ax. This confirms that the validated module is not only correct, but also well suited for experimental studies of modern and emerging Wi-Fi functionalities.

5.3. Future Work

The work presented in this thesis suggests several natural directions for further study. A first extension would be to apply the same measurement framework to features that were outside the main scope of this thesis, such as different aggregation strategies, BlockAck behaviour under varying loss conditions, and rate adaptation algorithms. Studying these mechanisms with link-layer visibility could clarify how they interact with contention and retransmissions.

A second direction is multi-link and multi-band operation in newer standards, in particular IEEE 802.11be. The channel occupancy metrics used here are well suited for analysing scheduling decisions and access delays in such scenarios.

Further work could also move beyond the ideal channel assumptions used in this thesis by including propagation loss, interference and mobility. Combining link-layer metrics with more realistic channel models would make it possible to study the interplay between PHY impairments and MAC contention. Finally, the framework could support validation of analytical IEEE 802.11 models or be used to prototype and evaluate new MAC-layer mechanisms directly in ns-3.

Bibliography

[1] L. Campanile, M. Gribaudo, M. Iacono, F. Marulli and M. Mastroianni, *Computer Network Simulation with ns-3: A Systematic Literature Review*, Electronics, vol. 9, no. 2, Article 272, 2020. Available online: https://www.mdpi.com/2079-9292/9/2/272 Accessed: 14 December 2024.

[2] G. Carneiro, P. Fortuna and M. Ricardo, *FlowMonitor: a network monitoring framework for the network simulator 3 (ns-3)*, Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS 2009), ICST, 2009. Available online: https://dl.acm.org/doi/abs/10.4108/icst.valuetools2009.7493 Accessed: 14 December 2024.

[3] M. Carrascosa-Zamacois, G. Geraci, E. Knightly and B. Bellalta, *Wi-Fi Multi-Link Operation: An Experimental Study of Latency and Throughput*, IEEE/ACM Transactions on Networking, vol. 32, no. 1, pp. 308–322, 2024. Available online: https://ieeexplore.ieee.org/abstract/document/10149044 Accessed: 14.12.2025.

[4] P. Kumar, J. Kulshrestha, M. Maity and S. Roy, *Use of Channel Occupancy for Multi Link WiFi 7 Scheduler Design in ns-3*, Proceedings of the 17th International Conference on Communication Systems & Networks (COMSNETS), Bengaluru, India, 2025. Available online: https://ieeexplore.ieee.org/document/10885613 Accessed: 14.12.2025.

[5] M. Ottens, J. Deutschmann, K.-S. Hielscher and R. German, *Performance Evaluation of ns-3 Real-Time Emulation*, IEEE Access, vol. 13, 2025. Available online: https://ieeexplore.ieee.org/abstract/document/10943112 Accessed: 14 December 2024.

[6] R. Pegurri, F. Linsalata, E. Moro, J. Hoydis and U. Spagnolini, *Toward Digital Network Twins: Integrating Sionna RT in ns-3 for 6G Multi-Rat Networks Simulations*, Proceedings of IEEE INFOCOM 2025, IEEE Conference on Computer Communications, 2025. Available online: https://ieeexplore.ieee.org/abstract/document/11152946 Accessed: 14.12.2025.

[7] A. Zubow, Y. Pilz, S. Rösler and F. Dressler, *Ns3 meets Sionna: Using Realistic Channels in Network Simulation*, arXiv preprint arXiv:2412.20524 [cs.NI], 2024. Available online: https://arxiv.org/abs/2412.20524 Accessed: 14.12.2025.

[8] P. Manzanares-Lopez, J. P. Muñoz-Gea and J. Malgosa-Sanahuja, *A Comprehensive Review of ns-3-Based Simulation Frameworks for LEO Satellite Constellations: Capabilities and Limitations*,

Software: Practice and Experience, vol. 55, no. 10, pp. 1657–1675, 2025. DOI: 10.1002/spe.70001. Available online: https://onlinelibrary.wiley.com/doi/full/10.1002/spe.70001 Accessed: 14.12.2025.

[9] D. Dziunikowska, *ns3-wifi-stats: Link-layer measurement framework for IEEE 802.11 simulations*, Source code repository, GitHub, 2025. Available online: https://github.com/lotycal/ns3-wifi-stats Accessed: 29.12.2025.

[10] The ns-3 Consortium, *ns-3.44 Release*, Official ns-3 release documentation, March 9, 2025. Available online: https://www.nsnam.org/releases/ns-3-44/ Accessed: 14.12.2025.

[11] S. K. Venkateswaran, C.-L. Tai, R. Garnayak, Y. Ben-Yeheskel and Y. Alpert, *IEEE 802.11ax Target Wake Time: Design and Performance Analysis in ns-3*, Proceedings of the 2024 Workshop on ns-3 (WNS3 '24), pp. 10–18, 2024. Available online: https://dl.acm.org/doi/abs/10.1145/3659111.3659115 Accessed: 20.12.2025.

A. Simulation Code Listings

A.1. Base Simulation Scenario (`ex1.cc`)

```
/* -*-  Mode: C++; c-file-style: "gnu"; indent-tabs-mode:nil; -*- */
/*
 * Copyright (c) 2016 SEBASTIEN DERONNE
 * Copyright (c) 2024 AGH University of Krakow
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation;
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
   USA
 *
 * Author: Szymon Szott <szott@agh.edu.pl>
 * Based on he-wifi-network.cc by S. Deronne <sebastien.deronne@gmail.com>
 * Last update: 2024-01-30 12:29
 */

#include "ns3/command-line.h"
#include "ns3/config.h"
#include "ns3/uinteger.h"
#include "ns3/boolean.h"
#include "ns3/double.h"
#include "ns3/string.h"
#include "ns3/log.h"
```

```
31 #include "ns3/yans-wifi-helper.h"
32 #include "ns3/ssid.h"
33 #include "ns3/mobility-helper.h"
34 #include "ns3/internet-stack-helper.h"
35 #include "ns3/ipv4-address-helper.h"
36 #include "ns3/udp-client-server-helper.h"
37 #include "ns3/packet-sink-helper.h"
38 #include "ns3/on-off-helper.h"
39 #include "ns3/ipv4-global-routing-helper.h"
40 #include "ns3/packet-sink.h"
41 #include "ns3/yans-wifi-channel.h"
42 #include <chrono> // For high resolution clock
43
44
45 // Exercise: 1
46 //
47 // This is a simple scenario to measure the performance of an IEEE
       802.11ax Wi-Fi network.
48 //
49 // Under default settings, the simulation assumes a single station in an
       infrastructure network:
50 //
51 //  STA     AP
52 //    *     *
53 //    |     |
54 //   n0     n1
55 //
56 // The user can specify the number of transmitting stations and the MCS
       value (0-11).
57 // The scenario assumes a perfect channel and all nodes are placed in the
       same location.
58 // All stations generate constant traffic so as to saturate the channel.
59 // The simulation output is the aggregate network throughput.
60
61 using namespace ns3;
62
63 NS_LOG_COMPONENT_DEFINE ("ex1");
64
65 int main(int argc, char *argv[])
66 {
67
68   // Initialize default simulation parameters
69   uint32_t nWifi = 1;         // Number of transmitting stations
```

```
70  double simulationTime = 1; // Default simulation time [s]
71  int mcs = 11;              // Default MCS is set to highest value
72  int channelWidth = 20;     // Default channel width [MHz]
73  int gi = 800;              // Default guard interval [ns]
74
75  // Parse command line arguments
76  CommandLine cmd;
77  cmd.AddValue("simulationTime", "Simulation time in seconds",
     simulationTime);
78  cmd.AddValue("mcs", "use a specific MCS (0-11)", mcs);
79  cmd.AddValue("nWifi", "number of stations", nWifi);
80  cmd.Parse(argc, argv);
81
82  // Print simulation settings to screen
83  std::cout << std::endl
84            << "Simulating an IEEE 802.11ax network with the following
     settings:" << std::endl;
85  std::cout << "- number of transmitting stations: " << nWifi << std::endl;
86  std::cout << "- frequency band: 5 GHz" << std::endl;
87  std::cout << "- modulation and coding scheme (MCS): " << mcs <<
     std::endl;
88  std::cout << "- channel width: " << channelWidth << " MHz" << std::endl;
89  std::cout << "- guard interval: " << gi << " ns" << std::endl;
90
91  // Create stations and an AP
92  NodeContainer wifiStaNodes;
93  wifiStaNodes.Create(nWifi);
94  NodeContainer wifiApNode;
95  wifiApNode.Create(1);
96
97  // Create a default wireless channel and PHY
98  YansWifiChannelHelper channel = YansWifiChannelHelper::Default();
99  YansWifiPhyHelper phy;
100 phy.SetChannel(channel.Create());
101
102
103 // Create and configure Wi-Fi network
104 WifiMacHelper mac;
105 WifiHelper wifi;
106 wifi.SetStandard(WIFI_STANDARD_80211ax);
107
108 // Set channel width for given PHY
```

```
109  std::string channelStr("{0, " + std::to_string(channelWidth) + ",
      BAND_5GHZ, 0}");
110  phy.Set("ChannelSettings", StringValue(channelStr));
111
112  std::ostringstream oss;
113  oss << "HeMcs" << mcs;
114  wifi.SetRemoteStationManager("ns3::ConstantRateWifiManager", "DataMode",
      StringValue(oss.str()),
115                               "ControlMode", StringValue(oss.str())); //
      Set MCS
116
117  Ssid ssid = Ssid("ns3-80211ax"); // Set SSID
118
119  mac.SetType("ns3::StaWifiMac",
120              "Ssid", SsidValue(ssid));
121
122  // Create and configure Wi-Fi interfaces
123  NetDeviceContainer staDevice;
124  staDevice = wifi.Install(phy, mac, wifiStaNodes);
125
126  mac.SetType("ns3::ApWifiMac",
127              "Ssid", SsidValue(ssid));
128
129  NetDeviceContainer apDevice;
130  apDevice = wifi.Install(phy, mac, wifiApNode);
131
132  // Set guard interval on all interfaces of all nodes
133
      Config::Set("/NodeList/*/DeviceList/*/$ns3::WifiNetDevice/HeConfiguration/GuardInt
      TimeValue(NanoSeconds(gi)));
134
135  // Configure mobility
136  MobilityHelper mobility;
137  mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
138  mobility.Install(wifiApNode);
139  mobility.Install(wifiStaNodes);
140
141  // Install an Internet stack
142  InternetStackHelper stack;
143  stack.Install(wifiApNode);
144  stack.Install(wifiStaNodes);
145
146  // Configure IP addressing
```

```
147  Ipv4AddressHelper address;
148  address.SetBase("192.168.1.0", "255.255.255.0");
149  Ipv4InterfaceContainer staNodeInterface;
150  Ipv4InterfaceContainer apNodeInterface;
151
152  staNodeInterface = address.Assign(staDevice);
153  apNodeInterface = address.Assign(apDevice);
154
155  // Install applications (traffic generators)
156  ApplicationContainer sourceApplications, sinkApplications;
157  uint32_t portNumber = 9;
158  for (uint32_t index = 0; index < nWifi; ++index) // Loop over all
      stations (which transmit to the AP)
159  {
160    auto ipv4 = wifiApNode.Get(0)->GetObject<Ipv4>();
      // Get destination's IP interface
161    const auto address = ipv4->GetAddress(1, 0).GetLocal();
      // Get destination's IP address
162    InetSocketAddress sinkSocket(address, portNumber++);
      // Configure destination socket
163    OnOffHelper onOffHelper("ns3::UdpSocketFactory", sinkSocket);
      // Configure traffic generator: UDP, destination socket
164    onOffHelper.SetConstantRate(DataRate(150e6 / nWifi), 1472);
     // Set data rate (150 Mb/s divided by no. of transmitting stations) and
     packet size [B]
165    sourceApplications.Add(onOffHelper.Install(wifiStaNodes.Get(index)));
       // Install traffic generator on station
166    PacketSinkHelper packetSinkHelper("ns3::UdpSocketFactory",
     sinkSocket); // Configure traffic sink
167    sinkApplications.Add(packetSinkHelper.Install(wifiApNode.Get(0)));
      // Install traffic sink
168  }
169
170  // Configure application start/stop times
171  // Note:
172  // - source starts transmission at 1.0 s
173  // - source stops at simulationTime+1
174  // - simulationTime reflects the time when data is sent
175  sinkApplications.Start(Seconds(0.0));
176  sinkApplications.Stop(Seconds(simulationTime + 1));
177  sourceApplications.Start(Seconds(1.0));
178  sourceApplications.Stop(Seconds(simulationTime + 1));
179
```

```
180   // Define simulation stop time
181   Simulator::Stop(Seconds(simulationTime + 1));
182
183   // Print information that the simulation will be executed
184   std::clog << std::endl
185             << "Starting simulation... ";
186   // Record start time
187   auto start = std::chrono::high_resolution_clock::now();
188
189   // Run the simulation!
190   Simulator::Run();
191
192   // Record stop time and count duration
193   auto finish = std::chrono::high_resolution_clock::now();
194   std::clog << ("done!") << std::endl;
195   std::chrono::duration<double> elapsed = finish - start;
196   std::cout << "Elapsed time: " << elapsed.count() << " s\n\n";
197
198   // Calculate throughput
199   double throughput = 0;
200   for (uint32_t index = 0; index < sinkApplications.GetN(); ++index) //
       Loop over all traffic sinks
201   {
202     uint64_t totalBytesThrough =
      DynamicCast<PacketSink>(sinkApplications.Get(index))->GetTotalRx(); //
      Get amount of bytes received
203     throughput += ((totalBytesThrough * 8) / (simulationTime *
      1000000.0));                              // Mbit/s
204   }
205
206   // Print results
207   std::cout << "Results: " << std::endl;
208   std::cout << "- aggregate throughput: " << throughput << " Mbit/s" <<
       std::endl;
209
210   // Clean-up
211   Simulator::Destroy();
212
213   return 0;
214 }
```