AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ INFORMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI

KATEDRA TELEKOMUNIKACJI

Master of Science Thesis

A Wi-Fi and NR-U Coexistence Channel Access Simulator based on the Python SimPy Library

Implementacja symulatora dostępu do kanału radiowego dla rywalizujących sieci Wi-Fi i NR-U z użyciem biblioteki SimPy języka Python

Autor: *Jakub Cichoń*
Kierunek studiów: *Teleinformatyka*
Opiekun pracy: *dr hab. inż. Szymon Szott*

Kraków, 2022

Contents

List of Figures

List of Tables

1. Introduction

Nowadays people can not image a word without the Internet. A fast, reliable and widely accessible Internet connection has become the norm. In Cisco Annual Report [1] it is stated that the total amount of Internet users is projected to grow from 3.9 billion in 2018 to 5.3 billion by 2023. This means that by 2023, 66 percent of the global population will have access to the Internet. Moreover, mobile subscribers using cellular services are expected to grow from 5.1 billion in 2018 to a 5.7 billion by 2023. These vast numbers of network subscribers cause capacity problems for Long Term Evolution's (LTE) systems. Releasing the 6 GHz band for unlicensed access in USA, Brazil, Canada, Chile, Costa Rica, Korea, Saudi Arabia and several countries in Europe gives the opportunity to offload LTE's data traffic into unlicensed band.

To make this happen, the 3rd Generation Partnership Project (3GPP) created standards for 5th Generation systems (5G), which covers New Radio (NR), the radio access technology designed to operate in shared band. NR as a successor of LTE, which already attempted to use unlicensed spectrum via LTE Licensed-Assisted Access (LTE-LAA) must obey restrictions to successfully coexist with other technologies (such as Wi-Fi) in unlicensed bands. New Radio Unlicensed (NR-U) was introduced to fulfill these requirement and provide access to unlicensed spectrum for 5G systems.

The goal of this thesis is to implement and validate a simulator written in Python with the SimPy library. The main objective of the simulator is to implement channel access mechanism for IEEE 802.11 and NR-U. The simulator should support a wide variety of possible configurations of parameters and numbers of nodes for each technology. As a result, it is expected to obtain basic performance statistics such as channel occupancy, airtime and collision probability. The validation should be done by comparison obtained results with results from an alternative implementations.

The thesis consists of four main chapters. Chapter 2 covers the theoretical background on channel access for both IEEE 802.11 and NR-U, NR-U operation modes and the state of the art. Chapter 3 contains a description of the implementation of the simulator. It is focused on the technical perspective, with an explanation of implementation and examples. Chapter 4 is dedicated to validating the simulator. The validation was done for each technology separately with use of other Python SimPy implementations of 802.11 and NR-U, as well as for coexistence of 802.11 and NR-U using Matlab simulator. Chapter 5 contains a performance analysis of coexistence scenarios. In this chapter, possible solutions to achieve fair channel access for both technologies are investigated. The last chapter summarizes the thesis.

2. Background

Since LTE LAA entered the unlicensed band there has been significant growth in research regarding coexistence possibilities between cellular systems and Wi-Fi. Offloading LTE's data traffic into unlicensed bands is considered as a solution that can help with LTE's capacity problems.

However, there are significant topics that must be addressed before successful coexistence between both technologies is possible. Wi-Fi is based on random channel access, in which an access point (AP) (or station) can start transmission at any time, provided the channel is idle. On the contrary, LAA is based on scheduled channel access, also known as reservation-based access. That means LAA base stations can start their transmissions only at a discrete point in time, and this transmission is scheduled by the controller [2]. In addition, the successor of LAA, NR-U was introduced as a part of 5G for operation in shared bands. Taking this into consideration, some contention mechanism must me implemented in systems based on scheduled channel access. The ETSI specification EN 301 893 [3] specifies a Listen Before Talk (LBT) procedure that ensures all technologies operating in the 5 GHz band have fair access to the channel.

2.1. Listen Before Talk

To ensure fair channel access in unlicensed bands, an LAA, NR-U or Wi-Fi node must precede its transmission with an LBT procedure.When the channel is sensed idle, the node waits a Prioritization Period (PP), during which it checks if the state of the channel has not changed. PP consists of an initial value of 16 µs, and also m observation slots, each 9 µs long. The value of m is dependent on the channel access priority class. Example values for NR-U are presented in Table 2.1, and values for Wi-Fi are presented in Table 2.2 (only downlink is considered for both technologies). If the channel was idle while waiting PP, node can proceed into backoff procedure, but if the channel was sensed occupied during PP, the node must stop LBT, wait till the moment when the channel is idle again and restart LBT. Entering the backoff procedure, the node draws a random value of N observation slots from the range 0 to CW (contention window). The initial value of CW is set to CWmin. The node proceeds with further channel sensing, and every time the channel is idle for the time of an observation slot, the value of N is decremented. When the channel is sensed occupied, the process of decrementing N must be frozen, and can only be resumed after channel is again idle for at least PP. When N becomes equal to 0, the node is eligible to transmit. The transmission must not be longer than the value of Maximum Channel Occupation Time (MCOT) from Table 2.1 and Table 2.2. The difference in MCOT parameter between Wi-Fi and NR-U can be slightly even up by using frame aggregation for Wi-Fi[4]. After a successful transmission, the acknowledgment is sent over the licensed band for LAA and NR-U, and over the unlicensed band for Wi-Fi. The value of CW is reset to CWmin. However, in case of a failed transmission, which can be caused by a collision, the node retransmits. This involves doubling the current CW, taking into account that CW can not exceed CWmax, and drawing a new value of N observation slots to wait. The node also increases the value of the parameter R, which stores the amount of failed transmissions in a row. If R exceeds the retransmission limit, the data must be discarded. The LBT procedure flowchart is presented in Figure 2.1.

Channel access priority class	*m* for DL	CWmin	CWmax	MCOT [ms]
1	1	3	7	2
2	1	3	15	3 (4)
3	3	15	63	8 or 10
4	7	15	1023	8 or 10

Table 2.1. NR-U channel access parameters values for different priority classes [5]

Access category	*m* for DL	CWmin	CWmax	MCOT [ms]
VO	1	3	7	2.08
VI	1	3	15	4.096
BE	3	15	63	2.528
BK	7	15	1023	2.528

Table 2.2. Wi-Fi channel access parameters values for different priority classes [6]

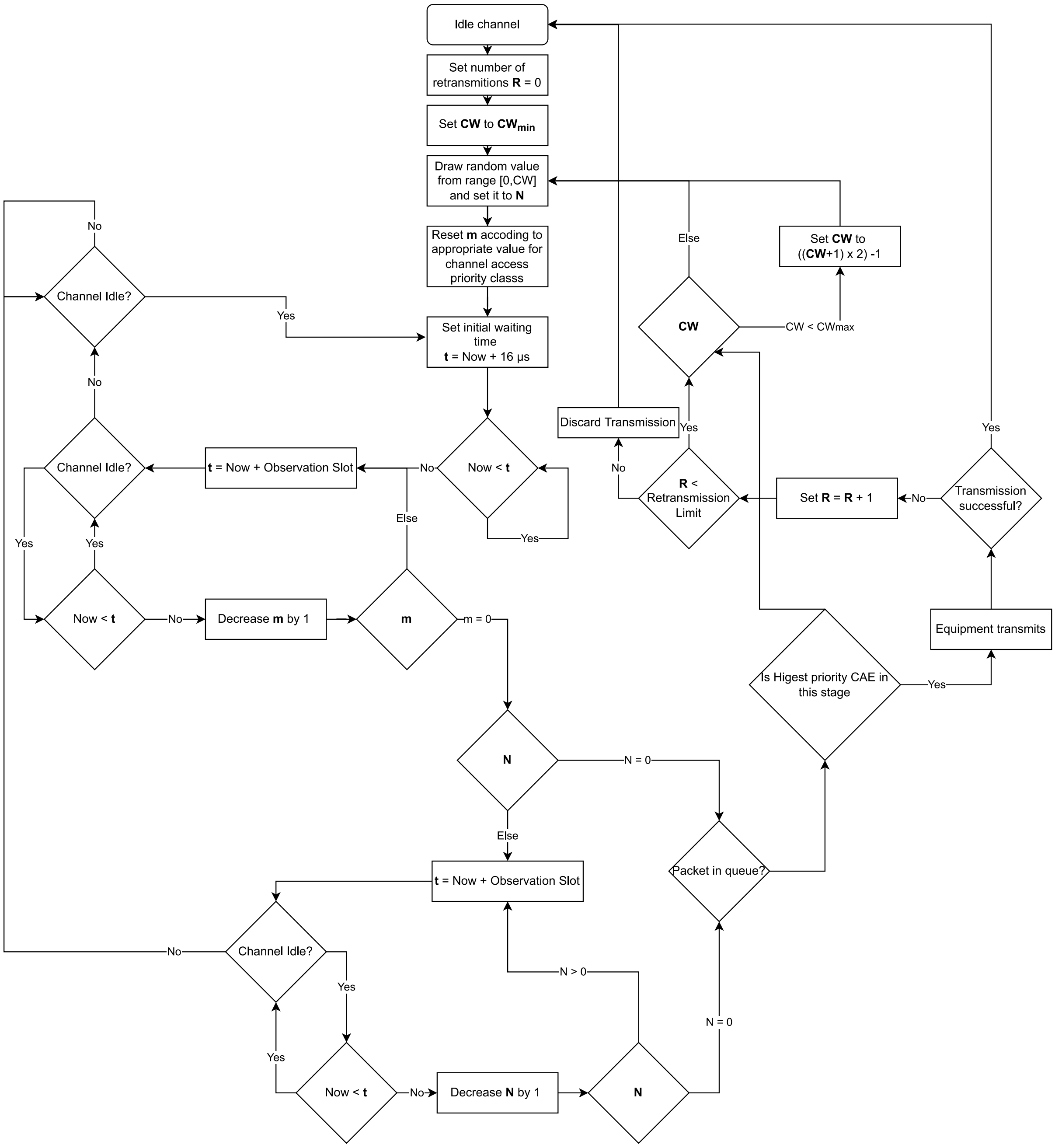


Figure 2.1. LBT procedure flowchart, based on [3] and [7]

2.2. Channel Access in IEEE 802.11

Wi-Fi as a technology with random channel access needs a mechanism that will coordinate access to the transmitting medium, which is the channel. APs use the Distributed Coordination Function (DCF) [8], which is based on carrier sense multiple access with collision avoidance (CSMA/CA) [9]. CSMA/CA is responsible for channel sensing, providing fair access to the channel for multiple APs operating in neighborhood and decreasing probability of collisions.

APs use the LBT procedure described in Chapter 2.1. In Wi-Fi, the period of 16 µs is called Short Interframe Space (SIFS) and the PP is known as DCF Interframe Space (DIFS), which consists of SIFS and *m* observation slots (each 9 µs long). In Wi-Fi, after a successful transmission, the ACK frame is sent (after a SIFS time) over the unlicensed band. This means that acknowledgments are sent via the same band as the data. DCF follows the flowchart presented in Figure 2.1, and an example of successful Wi-Fi transmission is shown in Figure 2.2.

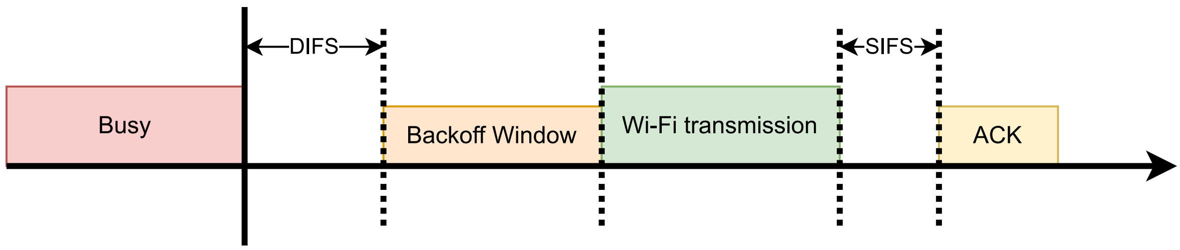


Figure 2.2. Example of Wi-Fi transmission

2.3. Channel Access in NR-U

The LBT procedure with its channel sensing and randomized backoff makes NR-U channel access similar to Wi-Fi channel access. However, LBT as a random channel access method may end in any point in time, not in the beginning of a synchronization slot boundary. Because a NR-U transmission can only start at the synchronization slot boundary, it can lead to creating significant gaps between the end of the LBT procedure and the start of the transmission, in which another node can start transmitting.

The first solution was to transmit a *reservation signal* (RS) (Figure 2.3) which blocks the medium until the start of the next synchronization slot boundary [10]. This approach has been criticized by IEEE [11] and by researchers [10], [12] and [13]. In the articles transmitting meaningless (from the data transmission point of view) jamming signal was highlighted as a main disadvantage of using RS. A jamming signal not only leads to wasting radio resources, which are already limited for NR-U by MCOT, but also results in channel resource sharing unfairness.

As an alternative, a second approach considering using gap period was proposed (Figure 2.4). Using the gap period result in initializing the channel access procedure (LBT) in a way, that if LBT succeeds it ends at the beginning of next synchronization slot boundary, and the Next Generation NodeB (gNB) can start transmitting. The gap can be placed before, after or even during the LBT procedure [14], [15]. The main advantage of using gap method is avoiding unnecessary channel occupation, but it can also cause a higher probability that during the gap period another node in the neighborhood starts transmitting and will prevent the gNB, that uses gap, from transmitting.

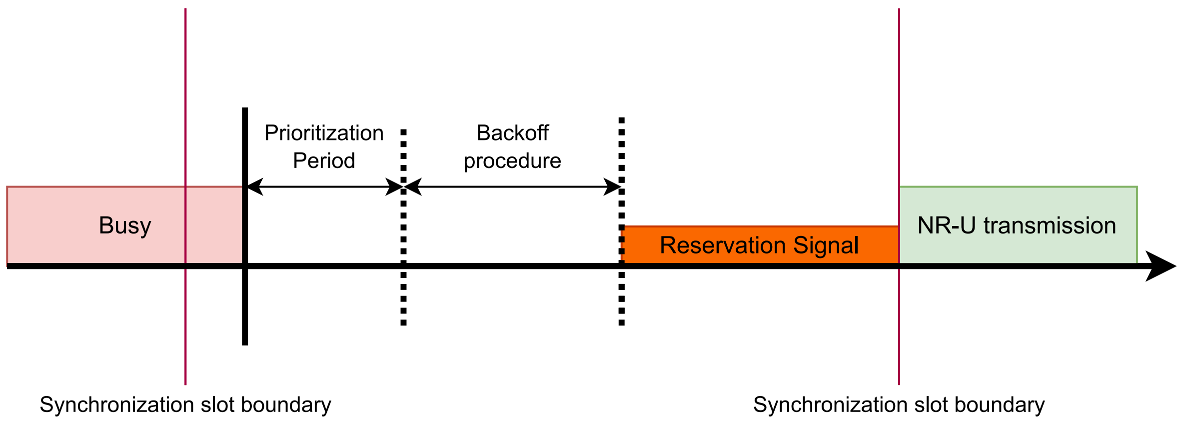


Figure 2.3. Example of NR-U transmission with RS

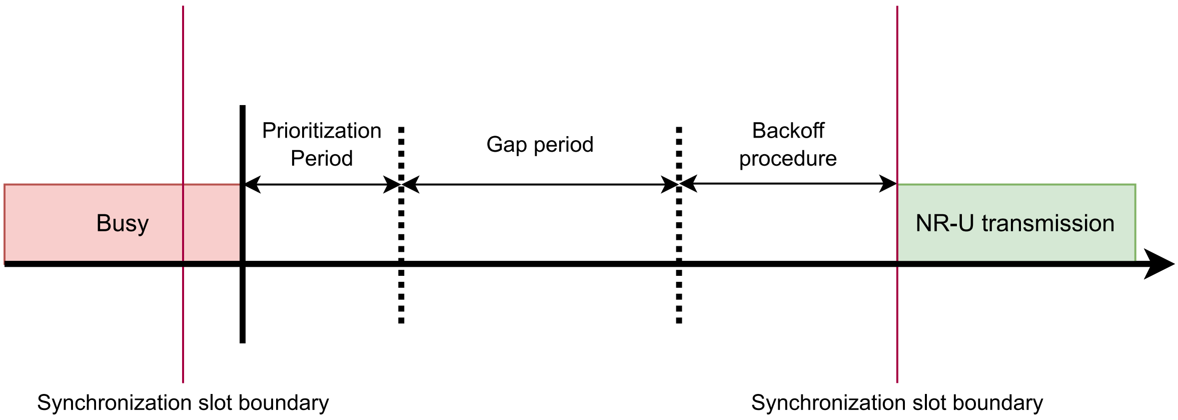


Figure 2.4. Example of NR-U transmission with gap period before backoff procedure

2.4. State of the Art

In recent years, there has been a constant growth in interest in research regarding coexistence in unlicensed bands. The most popular are the coexistence scenarios between LTE-LAA and Wi-Fi [13], in which coexistence scenarios are based on time division multiplexing (TDM). Multi-dimensional Markov chain models for network performance were presented in [16],[17], [18].

A second large group of interest is another category of coexistence scenarios, Frame Based Equipment. FBE is an option of LBT that allows devices operating in unlicensed bands to contend for channel access only at synchronized frame timestamps [19]. In [20], the authors describe FBE behaviour that aims to reduce the negative impact on Wi-Fi while keeping the performance of LTE under control. Further investigation into Wi-Fi and LTE performance, depending of data rates and packet sizes, was introduced using Markov chains and the ns-3 simulator. In [21], a theoretical analysis for FBE in coexistence scenarios was presented. Two variants of FBE introducing a backoff procedure before any channel access to reduce the greediness of FBE are proposed [22]. Other analytical models describing how the channel access is shared between FBE and Wi-Fi can be found in [23] and [24].

Among the latest research, there are papers regarding enhancing coexistence by exploiting multiple bandwidth parts (BWP) assignment [25]. BWP corresponds to resource allocation in the frequency domain, e.g., in the 5G-NR sub-band on which mobile user equipment operates. The new algorithm that exploits BWP and provides more opportunities for 5G operators to transmit was proposed, tested and compared with conventional LBT. More details about BWP can be found [26], and the analysis of the impact BWP switching has on 5G NR System Performance can be found in [27]. Another recent paper investigates how Cellular Vehicle to Everything (C-V2X) and Wi-Fi 6E in the 6 GHz band can cooperate [28]. Neutral networks are used to predict a set of strategies that ensures fair coexistence of LTE-U and Wi-Fi [29]. This shows that researchers are now diving into varied subjects and areas of coexistence in unlicensed bands.

Regarding existing tools used for research there is the ns-3 implementation of LAA/Wi-Fi coexistence [30] that was used for instance in [31]. Another existing simulator is the Matlab implementation of NR-U based on LAA [32], used in [33] and [34] as well as a simulator that implements NR-U operating with gap mode as well as RS mode [7]. The single simulator that can be used for coexistence of Wi-Fi and NR-U (that operates in both RS and gap modes) is the Matlab simulator presented in [14]. For this reason, creating the 5G-Coex-SimPy simulator presented in this thesis is essential for going forward with researching possible coexistence scenarios in unlicensed bands.

3. Implementation

This chapter describes the implementation of the main components of the 5G-Coex-SimPy simulator. These components consist of several functions that are used to simulate channel access procedures for IEEE 802.11 and NR-U networks. The simulator is written in Python, using the SimPy library.

3.1. SimPy

SimPy is a discrete-event simulation Python library that allows to queue, wait, and execute events in an ordered manner. Discrete-event simulation refers to systems that operate as a discrete sequence of events. Each of these events occurs at a specific time called *the timestamp* and changes the state in the system. There is no change in state of the system between events; therefore, the simulator can freely jump from one timestamp to the next.

SimPy is used for modeling the behaviour of active elements (such as stations, gNBs, transmissions, etc.) by processes. Processes are described as simple Python generators and live in a SimPy environment. They can interact with each other through events. Among many interactions we can list: waiting for the end of a process, and the commencement or interruption of a process. Processes during their lifetime can create and *yield* events. When an event is *yielded* the process is suspended till either the end of the event or the moment when the event occurred. This allows multiple processes to wait for the same event. A fundamental example of a SimPy event is *Timeout*. This type of event allows a process to remain inactive for a given amount of simulation time.

SimPy also provides various way of modeling shared resources that extends the possibilities of communication between processes. Namely, *Resources* can be used by a limited number of processes at a time; *Containers* model the production and consumption of either continuous or discrete assets, and *Stores* allow the production and consumption of Python objects.

The scenario of Wi-Fi stations coexisting with NR-U base stations can be presented in a discrete-event manner. Both technologies use specified procedures for sensing the state of the channel and waiting an appropriate amount of time to ensure the idleness of the channel. These procedures can be just presented as sets of events in time, changing the state of the channel. For these reasons the SimPy library is an appropriate tool to model channel access in such scenarios.

3.2. Channel Resources

In 5G-Coex-SimPy, the Channel class (Listing 3.1) is used to model the resource of radio channel. It uses two types of SimPy resources: simpy.Resource and simpy.PreemtiveResource. Simpy.Resource is used as tx_lock and is locked by the station, which is currently transmitting, and the simpy.PreemtiveResource (tx_queue) is locked by the station with the longest transmission time. The role of tx_queue is important in collision scenarios when the channel has to remain occupied by the longest transmission.

```
@dataclass()
class Channel:
    tx_queue: simpy.PreemptiveResource  # lock for the stations with the
    longest frame to transmit
    tx_lock: simpy.Resource  # channel lock (locked when there is ongoing
    transmission)
    n_of_stations: int  # number of transmitting stations in the channel
    n_of_gNB: int
    tx_list: List[Station] = field(default_factory=list)  # transmitting
    stations in the channel
    back_off_list: List[Station] = field(default_factory=list)  # stations
    in backoff phase
    tx_list_NR: List[Gnb] = field(default_factory=list)  # transmitting
    stations in the channel
    back_off_list_NR: List[Gnb] = field(default_factory=list)  # stations
    in backoff phase
    airtime_data: Dict[str, int]
    airtime_control: Dict[str, int]
    airtime_data_NR: Dict[str, int]
    airtime_control_NR: Dict[str, int]
    failed_transmissions: int = 0  # total failed transmissions
    succeeded_transmissions: int = 0  # total succeeded transmissions
    failed_transmissions_NR: int = 0  # total failed transmissions
```

```
    succeeded_transmissions_NR: int = 0  # total succeeded transmissions
```

Listing 3.1. Channel class representation

The channel class stores several input values determining the simulation conditions and data structures used in the simulation process. Among them are:

- n_of_stations – the number of Wi-Fi stations,
- n_of_gNB – the number of gNB base stations,
- back_off_list – list of Wi-Fi stations in the backoff procedure,
- tx_list – list of currently transmitting Wi-Fi stations,
- back_off_list_NR – list of gNBs in the backoff procedure,
- tx_list_NR – list of currently transmitting gNBs.

Not only is the Channel class used for initializing and processing, but also it enables storing obtained data for analyzing simulation results. The simulation outputs are:

- airtime_data – dictionary storing amount of time that each Wi-Fi station was transmitting data using the format {station name: value},
- airtime_control – dictionary storing amount of time of each Wi-Fi station was transmitting control data using the format {station name: value},
- airtime_data_NR – dictionary storing amount of time of each gNBs was transmitting data using the format {gNB name: value},
- airtime_control_NR – dictionary storing amount of time of each gNBs was transmitting jamming signal using the format {gNB name: value},
- failed_transmissions – stores number of failed Wi-Fi transmissions,
- succeeded_transmissions – stores number of succeeded Wi-Fi transmissions,
- failed_transmissions_NR – stores number of failed gNB transmissions ,
- succeeded_transmissions_NR – stores number of succeeded gNB transmissions.

3.3. Wi-Fi

Wi-Fi stations are represented in 5G-Coex-SimPy by Station class. This implementation comes from [35]. The Station class' vital part is one process, that is responsible for running backoff procedure and transmitting frames. It uses previously described Channel resource, and specific methods contained in Station class. To be more detailed, when instance of Station is created, it interacts with Channel to determine if radio channel is idle. If so, station proceeds into backoff mechanism, and after its finished reserve channel and performs frames transmission. Thanks to using the Channel resource, a station can be interrupted during backoff procedure if channel becomes occupied by other station, or this Station can send an interruption signal to other stations in waiting phase when it starts own transmission. There is also a method, that allows to detect collisions. It uses Channel resource to compare list of transmission and determine if the transmission was successful or there is need of re-transmission.

3.4. New Radio Unlicensed

Every base station in 5G-Coex-SimPy is represented as a Python object of a Gnb class (Listing 3.2) and has two main processes (lines 13-14). The first one starts at the moment of creation of gNB and is responsible for choosing random initial desynchronization time and after that calculating the next synchronization slot. The second process is used for base station activities. It creates new events including waiting backoff procedure, waiting for next synchronization slot, creating new transmissions (that are objects of Transmissions class in Listing 3.2) and executing transmissions. In more detail, the second process can be divided into two phases: executing Listen Before Talk procedure and, after it is completed, starting the transmission.

```
1  class Gnb:
2      def __init__(
3              self,
4              env: simpy.Environment,
5              name: str,
6              channel: dataclass,
7              config_nr: Config_NR = Config_NR(),
8      ):
9          self.config_nr = config_nr
10         self.name = name
11         self.env = env
```

```python
        self.channel = channel
        env.process(self.sync_slot_counter())
        env.process(self.start())
        self.succeeded_transmissions = 0
        self.failed_transmissions = 0
        self.failed_transmissions_in_row = 0
        self.cw_min = config_nr.cw_min
        self.next_sync_slot_boundry = 0
        self.cw_max = config_nr.cw_max
        self.back_off_time = 0
        self.time_to_next_sync_slot = 0
```

Listing 3.2. Gnb class representation

Each base station has a name, configuration file, and a reference to a SimPy environment and channel resource. Each station stores variables, which determine its operation and are used to gather obtained results. Some examples are: counters of failed transmissions, successful transmissions, the contention window size and time to next synchronization slot.

The transmission class is a data structure consisting of variables describing transmission parameters like transmission duration, station name, starting time, airtime, or time spent on sending reservation signal.

```python
@dataclass()
class Transmission_NR:
    transmission_time: int
    enb_name: str  # name of the owning it station
    col: str
    t_start: int  # generation time / transmision start (including RS)
    airtime: int  # time spent on sending data
    rs_time: int  # time spent on sending reservation signal before data
    number_of_retransmissions: int = 0
    t_end: int = None  # sent time / transsmision end = start + rs_time +
    airtime
    t_to_send: int = None
    collided: bool = False  # true if transmission colided with another one
```

Listing 3.3. Transmission class representation

3.4.1. Listen Before Talk

The implementation of LBT can differ depending on the current setting of the gap flag in 5G-Coex-SimPy. If gap is False, gNB uses RS method, and in opposite case it uses the gap based method. Both methods were implemented and are described in next sections. They use *generate_new_back_off_time()* to generate backoff time based on the current size of contention window (Listing 3.4).

```
def generate_new_back_off_time(self, failed_transmissions_in_row):
    upper_limit = (pow(2, failed_transmissions_in_row) * (
            self.cw_min + 1) - 1)  # define the upper limit basing on unsuccessful transmissions in a row
    upper_limit = (
        upper_limit if upper_limit <= self.cw_max else self.cw_max)  # set upper limit to CW Max if is bigger then this parameter
    back_off = random.randint(0, upper_limit)  # draw the back off value
    self.channel.backoffs[back_off][self.channel.n_of_stations] += 1  # store drawn value for future analyses
    return back_off * self.times.t_slot
```

Listing 3.4. Generating backoff time

3.4.1.1. RS-based Access

At the beginning, gNB draws random value of backoff between 0 and current value of contention window. Then it proceeds in waiting for the idle channel using request a of the channel resource. The next step is waiting the backoff time including the prioritization period. During this time another station can start transmitting which results in interrupting all waiting stations. In such cases an interrupted station decreases its backoff time by the already waited slots and continues waiting as soon as the channel is idle. When the backoff process ends successfully it sets its backoff value to -1 which causes it to leave the loop.

```
def wait_back_off(self):
    # Wait random number of slots N x OBSERVATION_SLOT_DURATION us
    self.back_off_time = self.generate_new_back_off_time(self.failed_transmissions_in_row)
    while self.back_off_time > -1:
            with self.channel.tx_lock.request() as req:  # waiting  for idle channel -- empty channel
```

```
                yield req
            self.first_interrupt = True
            prioritization_period_time = self.config_nr.deter_period + m *
    self.config_nr.observation_slot_duration
            self.back_off_time += prioritization_period_time  # add
    Priritization Period time to bacoff procedure
            self.channel.back_off_list_NR.append(self)  # join the list off
     stations which are waiting Back Offs
            yield self.env.timeout(self.back_off_time)  # join the
    environment action queue
            self.back_off_time = -1  # leave the loop
            self.channel.back_off_list_NR.remove(self)  # leave the waiting
     list as Backoff was waited successfully
```

Listing 3.5. RS backoff procedure

3.4.1.2. Gap-based Access

In the gap-based method, the gNB also starts by drawing a random value of backoff between 0 and current value of CW. Next, it calculates the gap period that enables the gNB to wait the backoff time right before the synchronization slot. A base station that waits a gap period remains inactive, and when it ends and the channel is still idle, the base station proceeds in waiting backoff. If there is an ongoing transmission at the end of the gap period, the station waits till the channel is idle and then calculates a new gap period. The backoff countdown can be interrupted in a similar way as in the RS-based method.

```
def wait_back_off_gap(self):
    self.back_off_time = self.generate_new_back_off_time(self.
    failed_transmissions_in_row)
    while self.back_off_time > -1:
            prioritization_period_time = self.config_nr.deter_period + m *
    self.config_nr.observation_slot_duration
            self.back_off_time += prioritization_period_time  # add
    Priritization Period time to bacoff procedure
            self.time_to_next_sync_slot = self.next_sync_slot_boundry -
    self.env.now
            while self.back_off_time > self.time_to_next_sync_slot:
                self.time_to_next_sync_slot += self.config_nr.
    synchronization_slot_duration
            gap_time = self.time_to_next_sync_slot - self.back_off_time
            yield self.env.timeout(gap_time)
```

```

            if self.channel.tx_lock.count > 0:
                log(self, 'Channel busy -- waiting to be free')
                with self.channel.tx_lock.request() as req:
                    yield req
                log(self, 'Finished waiting for free channel - restarting backoff procedure')
            else:
                log(self, 'Channel free')
                with self.channel.tx_lock.request() as req:
                    yield req
                log(self, f"Starting to wait backoff: ({self.back_off_time}) us...")
                self.channel.back_off_list_NR.append(self)  # join the list off stations which are waiting Back Offs
                yield self.env.timeout(self.back_off_time)  # join the environment action queue
                log(self, f"Backoff waited, sending frame...")
                self.back_off_time = -1  # leave the loop
                self.channel.back_off_list_NR.remove(self)  # leave the waiting list as Backoff was waited successfully
```

Listing 3.6. Gap backoff procedure

3.4.2. Transmissions and Collisions

When LBT ends successfully, a gNB creates a process responsible for executing transmissions and checking collisions. Both RS and Gap based methods use the same process (Listing 3.7). The maximum time of transmission is specified by the Maximum Channel Occupancy Time (MCOT). A gNB requests the channel resource and sends an interruption signal to all stations that are currently in backoff. In the RS method, where there is a need to sends reservation signal, MCOT is divided into two timers: rs_time and airtime. The former is the time that a station waits till the next synchronization slot where it can start data transmission and the latter is the remaining time. In the gap-based method, the backoff procedure ends at the beginning of the next synchronization slot so the whole MCOT can be used as airtime (rs_time is equal to 0).

```
def send_transmission(self):
    self.channel.tx_list_NR.append(self)  # add station to currently transmitting list
    self.transmission_to_send = self.gen_new_transmission()
```

```
res = self.channel.tx_queue.request(priority=(
        big_num - self.transmission_to_send.transmission_time))  # create request basing on this station frame length
try:
    result = yield res | self.env.timeout()  # try to hold transmitting lock(station with the longest frame will get this)
    if res not in result:  # check if this station got lock, if not just wait you frame time
        raise simpy.Interrupt("There is a longer frame...")
    with self.channel.tx_lock.request() as lock:  # this station has the longest frame so hold the lock
        yield lock
        for station in self.channel.back_off_list:  # stop all station which are waiting backoff as channel is not idle
            if station.process.is_alive:
                station.process.interrupt()
        for gnb in self.channel.back_off_list_NR:  # stop all station which are waiting backoff as channel is not idle
            if gnb.process.is_alive:
                gnb.process.interrupt()

        log(self, f'Transmission will be for: {self.transmission_to_send.transmission_time} time')
        yield self.env.timeout(self.transmission_to_send.transmission_time)

        self.channel.back_off_list_NR.clear()  # channel idle, clear backoff waiting list
        was_sent = self.check_collision()  # check if collision occurred

        if was_sent:  # transmission successful
            self.channel.airtime_control_NR[self.name] += self.transmission_to_send.rs_time
            log(self, f"adding rs time to control data: {self.transmission_to_send.rs_time}")
            self.channel.airtime_data_NR[self.name] += self.transmission_to_send.airtime
            log(self, f"adding data airtime to data: {self.transmission_to_send.airtime}")
            # yield self.env.timeout(self.times.get_ack_frame_time())  # wait ack
            self.channel.tx_list_NR.clear()  # clear transmitting list
```

```
                    self.channel.tx_queue.release(res)  # leave the transmitting queue
                    return True

            # there was collision
            self.channel.tx_list_NR.clear()  # clear transmitting list
            self.channel.tx_queue.release(res)  # leave the transmitting queue
            self.channel.tx_queue = simpy.PreemptiveResource(self.env, capacity=1)  # create new empty transmitting queue

            return False
    except simpy.Interrupt:  # this station does not have the longest frame, waiting frame time
            yield self.env.timeout(self.transmission_to_send.transmission_time)

    was_sent = self.check_collision()
    return was_sent
```

Listing 3.7. Sending transmission process

The transmission method includes checking if this gNB is the only one that started a transmission at this time, if not a collision has occurred and in other case the transmission is successful and channel parameters of time occupation are updated by adding airtime and rs timers to variables that store results. In case of a collision, the check_collisions() method updates gNB and channel parameters of unsuccessfully transmissions which results in higher Cw next drawing.

3.5. Running Simulations and Generating Outputs

To run the 5G-Coex-SimPy simulator, the client_coex.py script was implemented. There are two options of running simulation, the first one is running single run with specific parameters and the second one is to run multiple runs with changing parameters. Running a simulation requires providing input parameters:

– seed used by the pseudo random number generator.

– number of Wi-Fi stations.

– number of gNB stations.

– simulation time.

– Wi-Fi parameters: mcs value, Cw_Max and Cw_Min.

– gNB parameters: Cw_Max and Cw_Min.

After successful simulation run, the Channel class stores data of every station that was taking part in simulation. 5G-Coex-SimPy uses this data to calculate the total airtime, channel occupation time and collision probability for Wi-Fi, gNB and summarized for both technologies. These values are then saved in csv form and labeled with the used seed, number of Wi-Fi stations and gNBs used in this run.

4. Validation

This chapter describes the performance validation of the 5G-Coex-SimPy simulator. Three existing simulators were used: Pawel Topor's DCF-SimPy, which implements Wi-Fi channel access in 802.11a [35], Marek Zając's NRU-SimPy, which implements LBT for NR-U [36], and a Matlab implementation of both Wi-Fi and NR-U channel access, which can be used for both coexistence scenarios as well as testing the performance of each technology separately.

The comparison was based on three performance metrics: collision probability (*pcol*), channel occupancy (*cot*) and channel efficiency (*eff*). Collision probability is the probability that a collision occurs during transmission, channel occupancy is the normalized amount of time in which the channel was occupied (by a given technology), and channel efficiency refers to the normalized amount of time that was used for data transmission (i.e., without signalling overhead: ACK in the Wi-Fi case and RS in the NR-U case). Normalization is done with respect to the total simulation time. The 5G-Coex-SimPy can be assumed working correctly if its results are close to results obtained with other simulators.

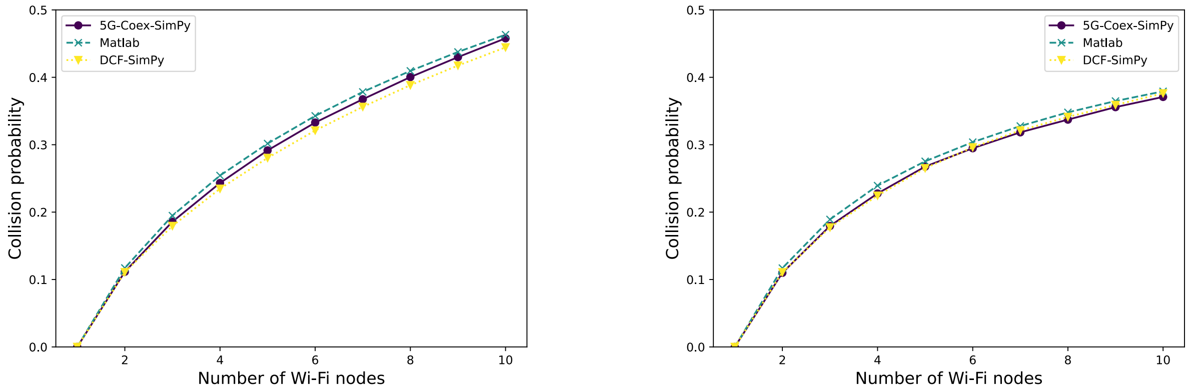
4.1. Wi-Fi

To validate the Wi-Fi implementation in 5G-Coex-SimPy, the DCF-SimPy and Matlab simulators were used. Tests were performed with various number of stations, ranging from 1 to 10. Every simulation lasts 100 seconds and was run 10 times. Validation was divided into two scenarios, which differ in used parameters (Table 4.1). Other used parameters are characteristic for the 802.11a standard.

Originally, 5G-Coex-SimPy used the DCF-SimPy implementation of Wi-Fi stations. In 5G-Coex-SimPy, there is significant upgrade in the process of counting down already waited slots during the backoff procedure. Due to this upgrade, the obtained results were even more similar to results from the Matlab simulator, which can be seen in Figure 4.1, Figure 4.2 and Figure 4.3.

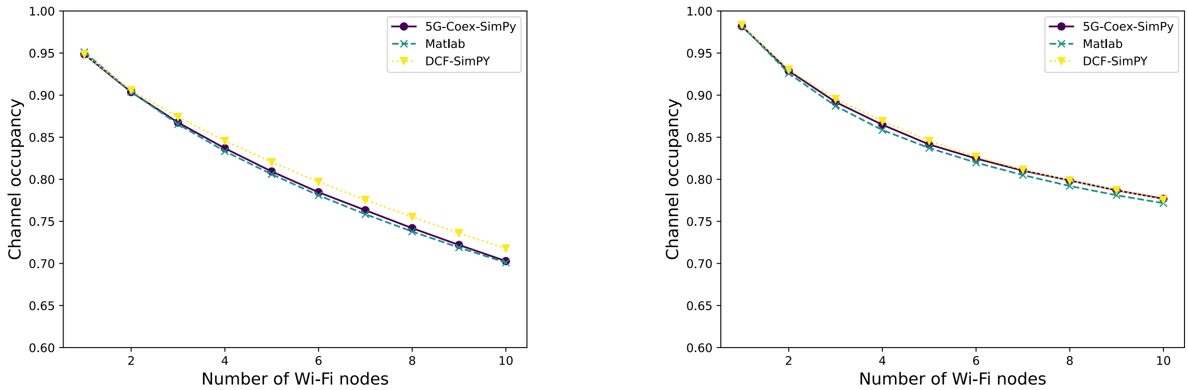
Parameters	Scenario I	Scenario II
Transmission time	2 ms	6 ms
CWmin	15	15
CWmax	63	1023
MCS	7	7
Retrasmission limit	3	7

Table 4.1. Wi-Fi simulation parameters



(a) Scenario I (b) Scenario II

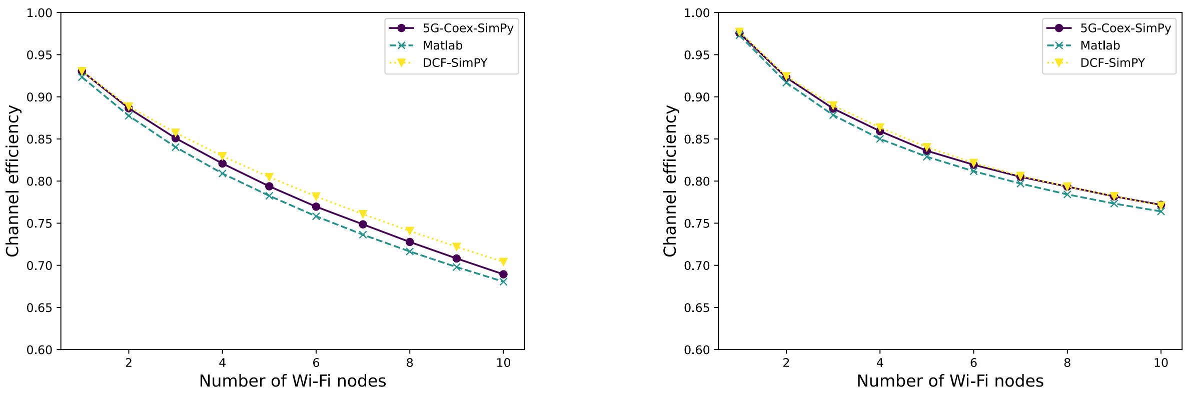
Figure 4.1. Collision probability for Wi-Fi



(a) Scenario I (b) Scenario II

Figure 4.2. Channel occupancy for Wi-Fi

5G-Coex-SimPy's results for both scenarios are similar to results obtained by the use of other simulators, thus 5G-Coex-SimPy can be considered working properly.



(a) Scenario I

(b) Scenario II

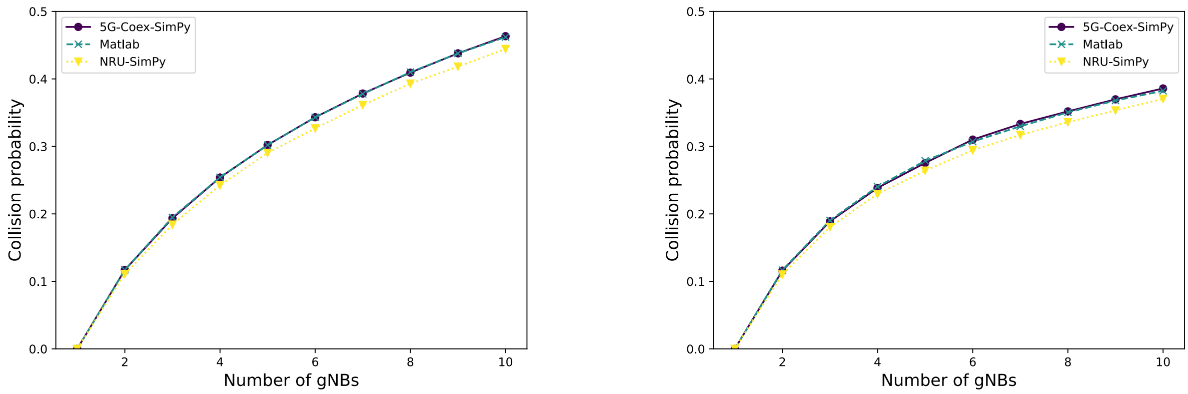
Figure 4.3. Channel efficiency for Wi-Fi

4.2. NR-U with Reservation Signal

5G-Coex-SimPy's NR-U RS based performance was validated using NRU-SimPy and Matlab. Tests were performed with various number of gNBs, ranging from 1 to 10. Every simulation lasts 100 seconds and was run 10 times. Validation was again divided into two scenarios representing different priority classes, which differ in used parameters (Table 4.2).

Parameters	Scenario I	Scenario II
Channel access priority class	3	4
CWmin	15	15
CWmax	63	1023
Number of observation slots	3	7
MCOT limit	6 ms	8 ms
Synchronization slot	1000 µs	1000 µs

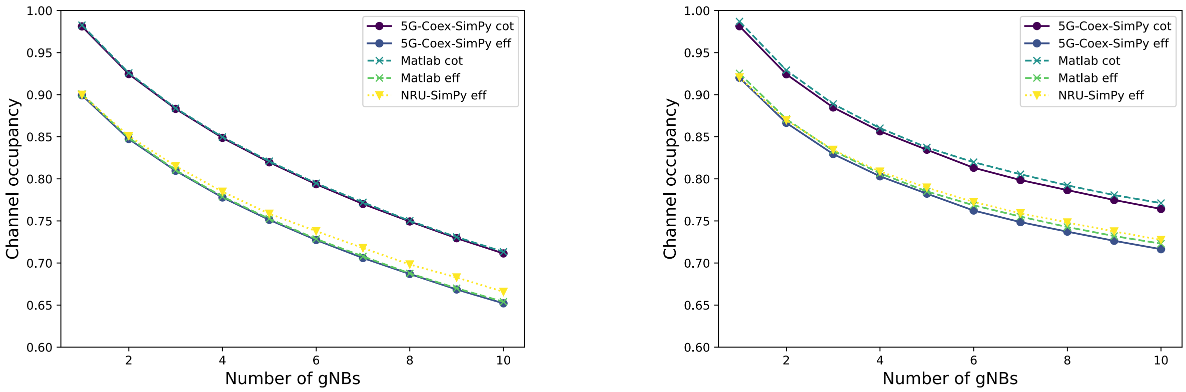
Table 4.2. NR-U simulation parameters



(a) Scenario I (b) Scenario II

Figure 4.4. Collision probability for NR-U with reservation signal

From Figure 4.4 and Figure 4.5, it is clear that Matlab's and 5G-Coex-SimPy's performance is similar. NRU-SimPy's results slightly differ from the rest due to different implementation and use of the SimPy library.



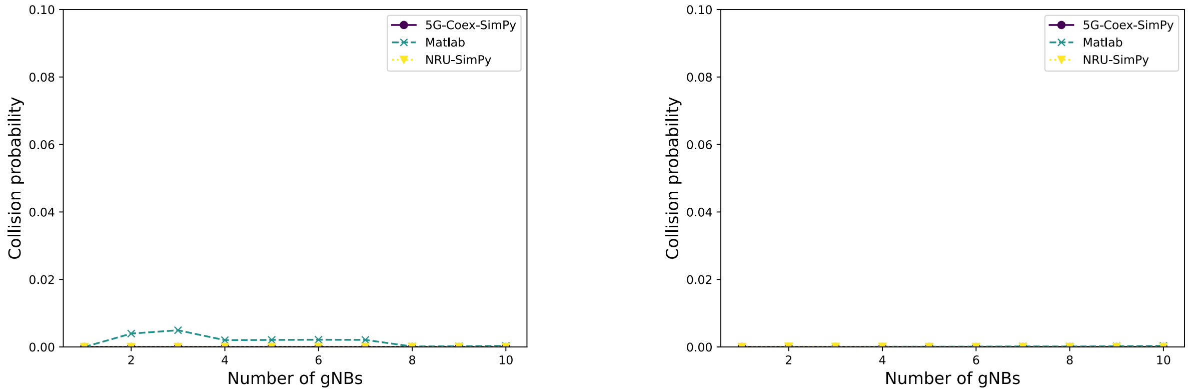
(a) Scenario I

(b) Scenario II

Figure 4.5. Channel occupancy and channel efficiency for NR-U with reservation signal

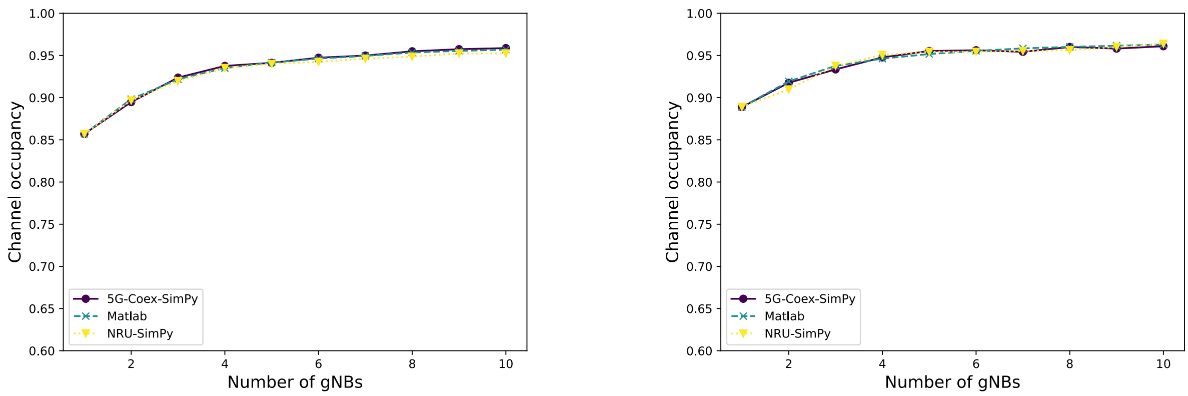
4.3. NR-U with Gap

As before, 5G-Coex-SimPy's NR-U gap based performance was validated using NRU-SimPy and Matlab simulators. Tests were performed with various number of gNBs, in range from 1 to 10. Every simulation lasts 100 seconds and was run 10 times. Validation was divided into similar two scenarios as previously, representing different priority classes (Table 4.2).



(a) Scenario I (b) Scenario II

Figure 4.6. Collision probability for NR-U with gap period



(a) Scenario I (b) Scenario II

Figure 4.7. Channel occupancy for NR-U with gap period

From Figure 4.6 and Figure 4.7, it is clear that Matlab's and 5G-Coex-SimPy's performance is similar. In Figure 4.7, only *cot* was considered because NR-U with gap does not use airtime to transmit control data in the 5G band [2]. NRU-SimPy's results slightly differ from the rest due to different implementation and usage of SimPy library.

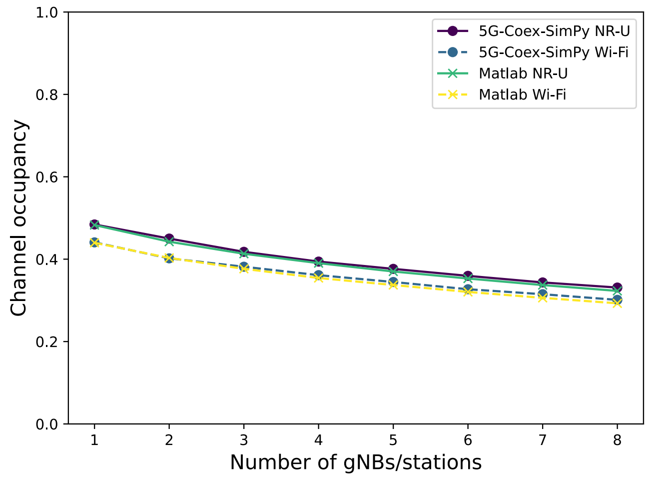
4.4. Coexistence

To validate 5G-Coex-SimPy's coexistence performance only the Matlab simulator could be used. Tests were performed with equal number of both Wi-Fi stations an gNBs at one time, in range from 1 to 8. Every simulation lasts 100 seconds and was run 10 times. Simulations were performed with parameters matching best effort access category (Table 4.3) and divided into two parts. In the first part gNBs use RS (Figure 4.8) and in the second gNBs use gap (Figure 4.9). In both cases, gNBs were desynchronized by random offset less or equal to 1000 ms.

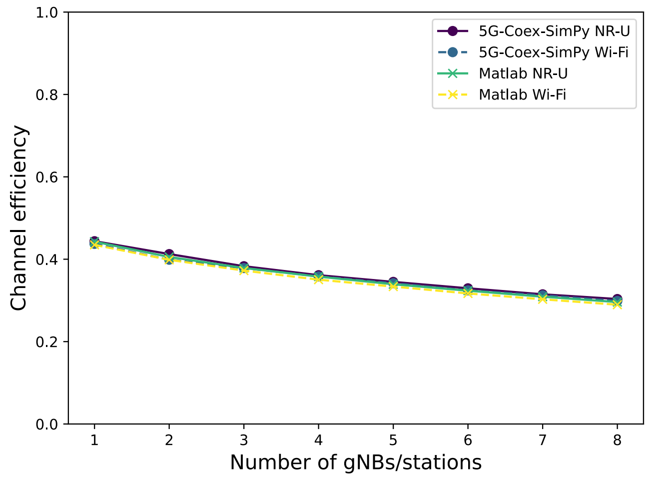
System	Number of observation slots	CWmin	CWmax	Transmission time [ms]
Wi-Fi	3	15	63	5.4
NR-U	3	15	63	6

Table 4.3. Coexistence simulation parameters

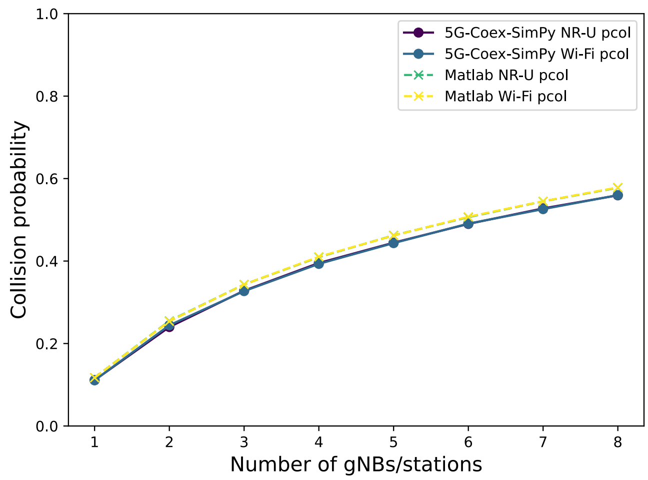
As can be seen in the figures presented in this Chapter 4.4. 5G-Coex-SimPy's results are again similar to Matlab's. The little discrepancy in Figure 4.9c in collision probability for NR-U comes from the difference in implementation gap method. In 5G-Coex-SimPy, gap is inserted before LBT, and in Matlab it is placed after LBT. Bearing in mind that in this scenario probability of performing transmission by NR-U is close to zero, even few attempts of transmission can result in differences in collision probability. The overall Channel occupancy for both Wi-Fi and NR-U in 5G-Coex-SimPy and Matlab are similar though. From this comparison it can be stated that 5G-Coex-SimPy works correctly.



(a) Channel occupancy for coexistence scenario with RS

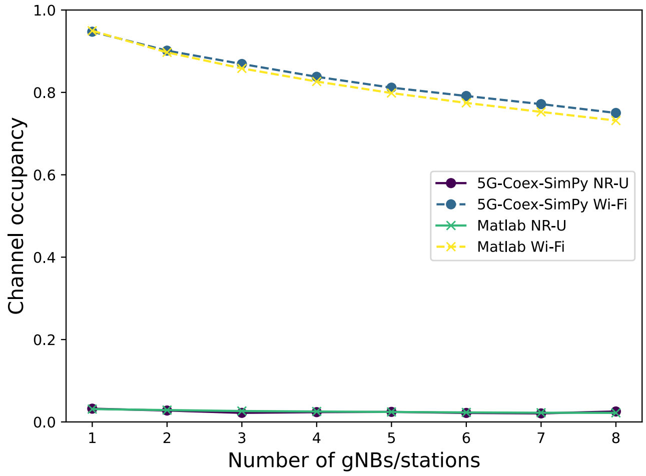


(b) Channel efficiency for coexistence scenario with RS

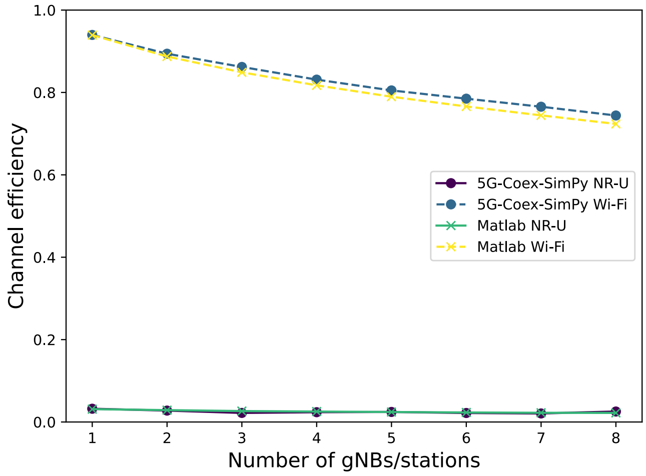


(c) Collision probability for coexistence scenario with RS

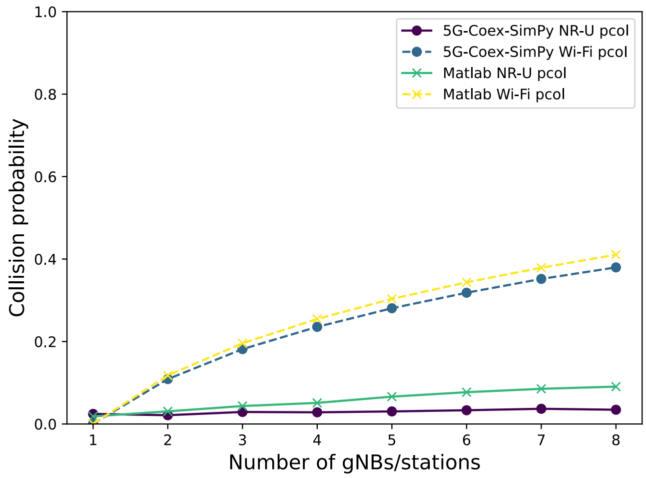
Figure 4.8. Coexistence between Wi-Fi and NR-U using RS



(a) Channel occupancy for coexistence scenario with gap



(b) Channel efficiency for coexistence scenario with gap



(c) Collision probability for coexistence scenario with gap

Figure 4.9. Coexistence between Wi-Fi and NR-U using gap

4.5. Credibility of Results

In order to estimate the credibility of the obtained results, their statistical dispersion of results was investigated. It was done for *cot* (for each technology separately) in simulation scenarios with 10 Wi-Fi stations or gNBs, and repeated 10 times. Results are presented in Table 4.4. Overall variance of this data sets tells us that data is not widely scattered and small confidence intervals ensures correct results.

Simulation	Standard deviation	Variance	Confidence interval
Wi-Fi scenario I	0.001	3.52e-06	0.003
Wi-Fi scenario II	0.0009	9.49e-07	0.002
NR-U RS scenario I	0.002	3.94e-06	0.001
NR-U RS scenario II	0.003	9.57e-06	0.002
NR-U gap scenario I	0.011	0.000129	0.005
NR-U gap scenario II	0.000	0.00e+00	0.000

Table 4.4. Statistical dispersion of each technology results

For coexistence scenarios, the credibility of results was checked by investigating *cot* in simulation scenarios with 8 Wi-Fi stations and 8 gNBs. Simulations were repeated 10 times. Results are presented in Table 4.5. Again, variance of this data tells that results are not widely scattered and small confidence intervals ensures correct results.

Coexistence version	System	Standard deviation	Variance	Confidence interval
RS based	Wi-Fi	0.003423	0.000012	0.002449
	NR-U	0.003	0.000	0.002
gap based	Wi-Fi	0.003	7.31e-06	0.002
	NR-U	0.006	3.55e-05	0.004

Table 4.5. Statistical dispersion of coexistence results

5. Performance Analysis

The main goal in coexistence scenarios is to provide comparable access to the channel for both Wi-Fi and NR-U technologies. NR-U can operate with two different modes: RS and gap based. The RS mode is effective in providing fair access to the channel as can be seen in Figure 4.8a. The downside of using RS is wasting radio resources while transmitting the jamming signal. For this reason, it has been criticised by IEEE and its not considered as a preferable approach [11]. The second, gap mode does not cause a waste of radio resources, but it often does not provide fair access to channel for both technologies. As Figure 4.9a shows, gap based NR-U can barely transmit in the vicinity of Wi-Fi stations. This chapter considers gap-based NR-U nodes and presents different approaches that allow fair channel sharing between NR-U and Wi-Fi. The conducted performance analysis repeats the work done in [14] but with the newly developed, SimPy-based simulator.

5.1. Synchronization in NR-U

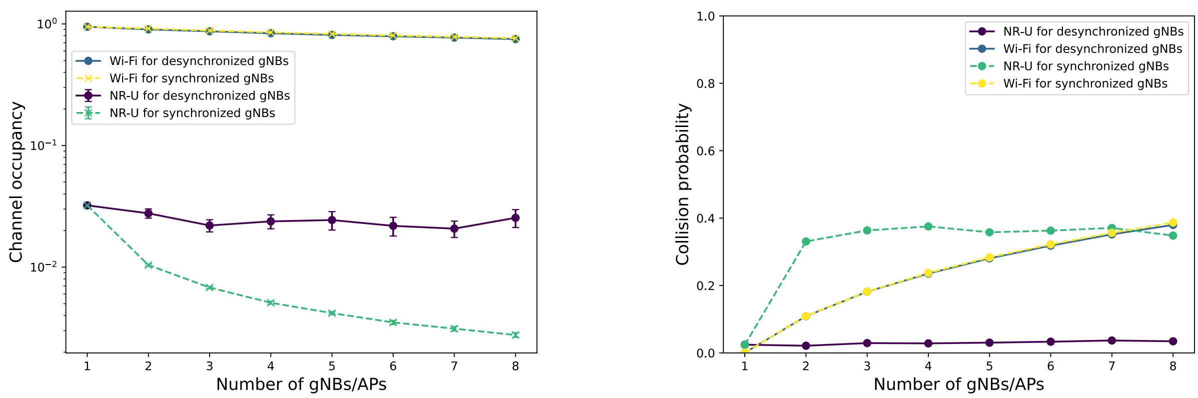
In contrary to Wi-Fi, in which channel access is random, NR-U channel access is scheduled. This means that gNB can only a start transmission at the beginning of a synchronization slot. In a real word scenario, all gNBs are synchronized and have the same synchronization slot starting times, which leads to a large amount of collisions between gNBs. In order to avoid such a situations, random desynchronization of each gNB is required (Listing 5.1). As a result, when a gNB is created, its synchronization slot will be delayed by a random offset, within a configured range from 0 to value of the synchronization slot.

```
def sync_slot_counter(self):
    # Process responsible for keeping the next sync slot boundry timestamp
    self.desync = random.randint(self.config_nr.min_sync_slot_desync, self.config_nr.max_sync_slot_desync)
    self.next_sync_slot_boundry = self.desync
```

```
    log(self, f"Selected random desync to {self.desync} us")
    yield self.env.timeout(self.desync)  # waiting randomly chosen desync
time
    while True:
        self.next_sync_slot_boundry += self.config_nr.
synchronization_slot_duration
        log(self, f"Next synch slot boundry is: {self.
next_sync_slot_boundry}")
        yield self.env.timeout(self.config_nr.synchronization_slot_duration
)
```

Listing 5.1. Synchronization slot function with initial gNB desynchronization

The coexistence of APs and gNBs is simulated in two scenarios: with synchronized gNBs and with desynchronized gNBs. In this analysis, the random desynchronization value is chosen in the range from 0 to 1000 µs, obtaining an offset value lower than synchronization slot length (1000 µs). In simulations, the number of APs is equal to the amount of gNBs, and is increased from one to eight nodes of each technology.



(a) Impact of gNBs synchronization on channel occupancy

(b) Impact of gNBs synchronization on collision probability

Figure 5.1. Comparison of NR-U with synchronized and desynchronized gNBs

Results in Figure 5.1a (*Channel occupancy* is in logarithmic scale) clearly indicate that desynchronizing gNBs increase channel occupancy for NR-U, especially in scenarios with larger amount of gNBs. The reason for this is the lower collision probability for desynchronized gNBs (Figure 5.1b). Although desynchronizing gNBs gives better results, it does not provide fair channel access for NR-U. The best results were still below 10% channel occupancy which is still not a satisfying result.

5.2. Disabling Backoff for NR-U

After adding desynchronization, every gNB can start its transmission at a different point in time, and waits a different amount of time as a gap period. For these two reasons, waiting the random backoff becomes unnecessary especially with large slot duration lengths, thus in the next step backoff in NR-U is disabled by setting for NR-U CWmin=CWmax=0, and for Wi-Fi CWmin=15, and CWmax=63.

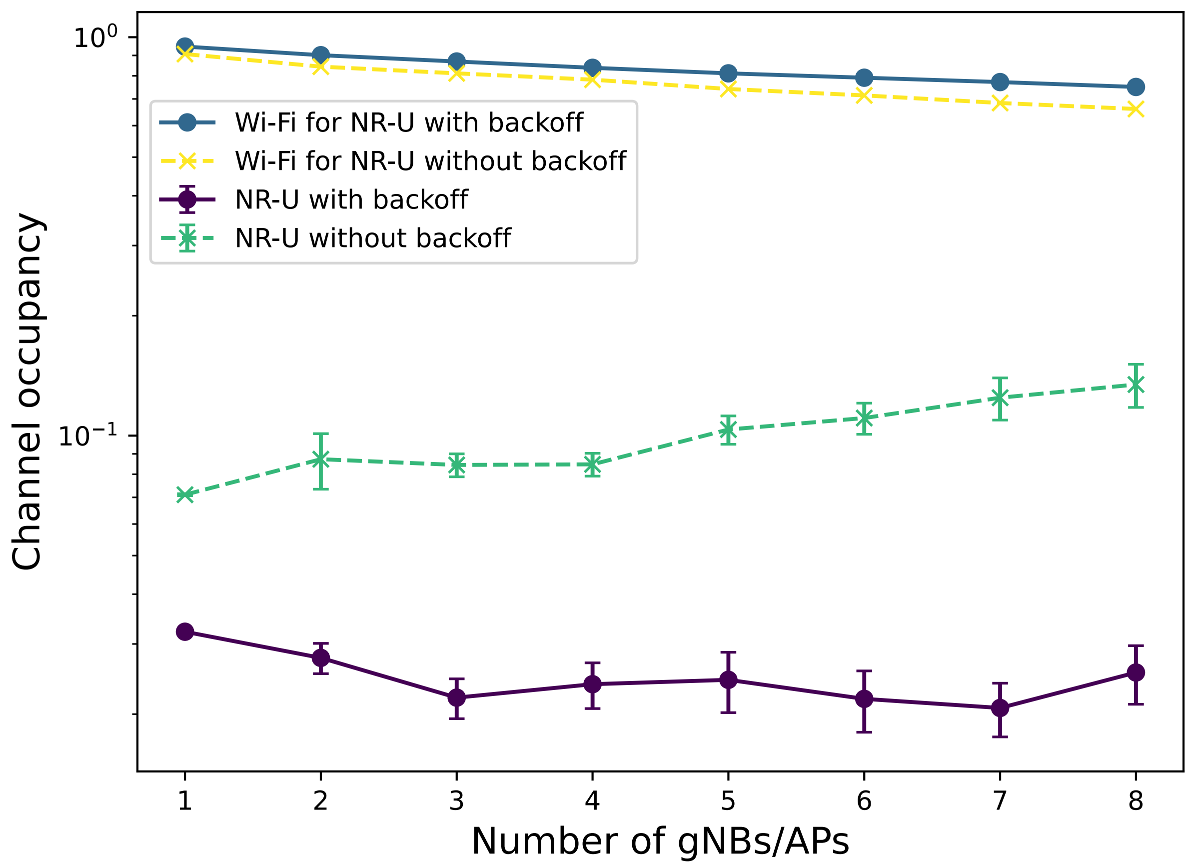


Figure 5.2. Result of disabling backoff procedure for NR-U

In Figure 5.2 (*Channel occupancy* is in logarithmic scale), it can be seen that disabling backoff for NR-U further increased NR-U performance. The gain grows visibly with the number of gNBs, which is a result of growth in the probability of finding free channel when there is more gNBs, with less random amount of time in the LBT procedure. Nonetheless, the NR-U airtimes remain strikingly low in comparison to Wi-Fi's.

5.3. Equalizing Per-Technology Airtime

The next step to provide fair access to radio channel for both technologies was adjusting Wi-Fi contention window (CWmin=CWmax) while NR-U was operating with no backoff (CWmin=CWmax=0). Simulations were done for different scenarios starting with Wi-Fi CW varying from 32 to 512 a with step of 48. In every scenario, there are two APs and two gNBs. Results of simulations are shown in Figure 5.3, and the best result, providing nearly equal division in channel occupancy is achieved for Wi-Fi CWmin=CWmax = 176 (while NR-U CWmin=CWmax=0).

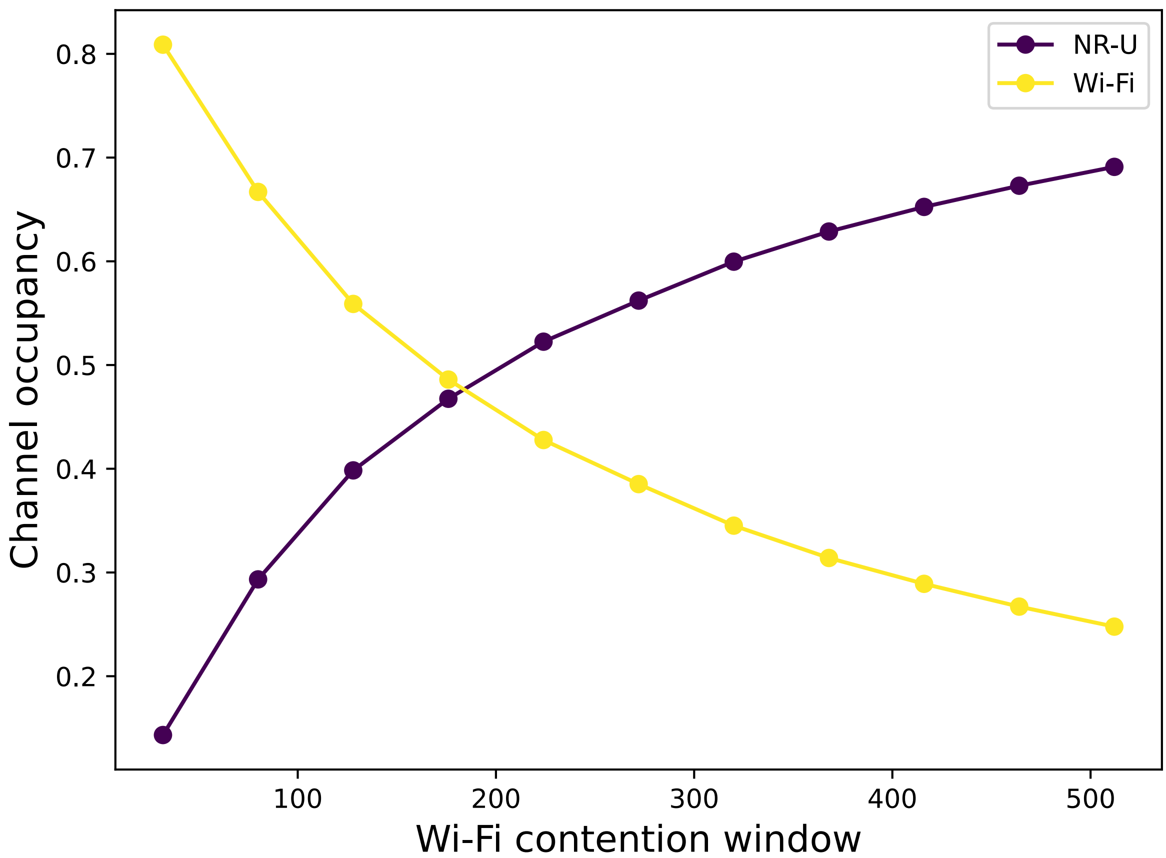


Figure 5.3. Determining Wi-Fi CW for equal channel airtime distribution between Wi-Fi and NR-U

In order to further investigate possible configurations that result in fair channel access more simulations were done. In the next simulations several values of NR-U contention window (CWmin=CWmax) with Wi-Fi contention window (CWmin=CWmax), varying from 100 to 400 with step of 25 were checked. Figure 5.4 presents Jain's fairness index for specific CW configurations, which increases for configuration that provides division in channel airtime that is most close to ideal.

Jain's fairness index (JFI) rates how fair a resource is allocated between users. In this case it is calculated in a per-technology manner, which means that I assume there are two users: Wi-Fi and NR-U. For calculations, the aggregated airtime was used for both technologies. The value of the fairness index can vary from $\frac{1}{n}$ (worst case) to 1 (best case) meaning equal share. Joint airtime-fairness factor, that is a product of aggregate normalized airtime and Jain's fairness index, is also considered. From Figure 5.4 we can determine best CWs for each technology, that ensures best fairness in channel occupancy (Table 5.1).

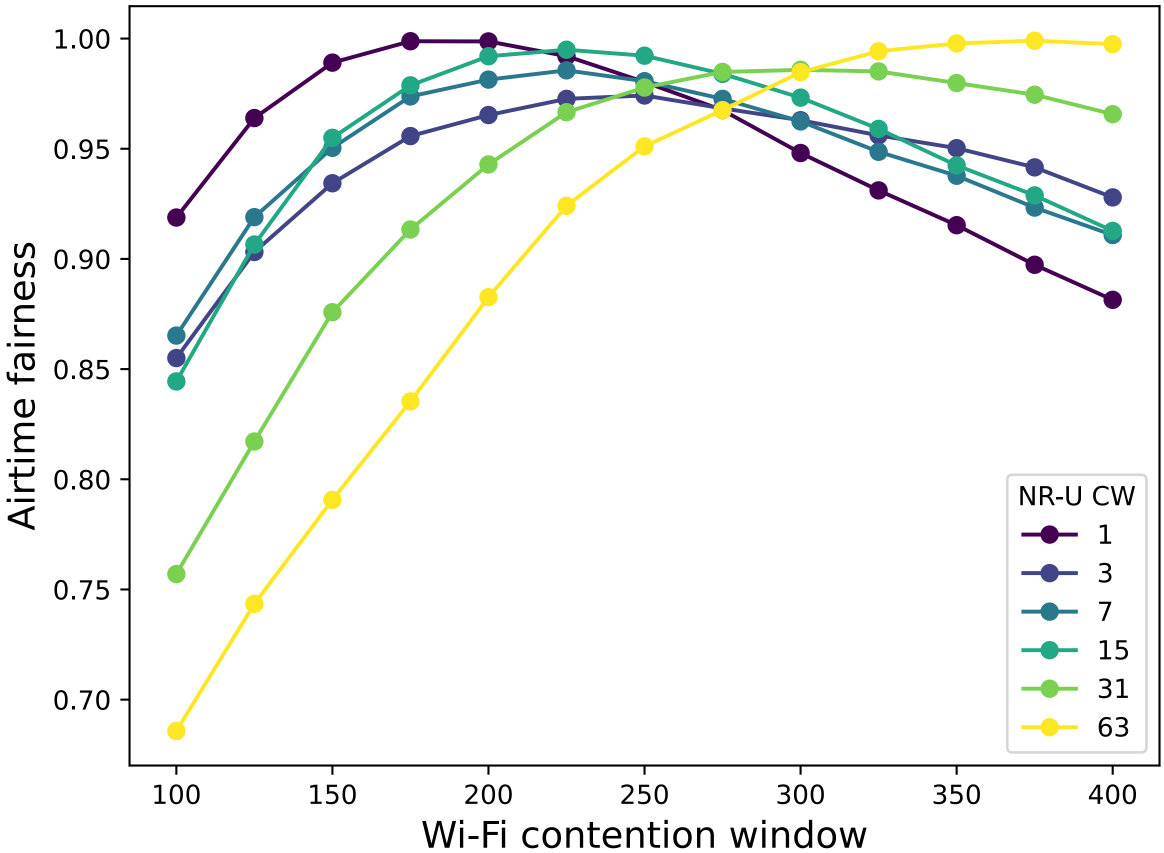


Figure 5.4. Jain's fairness index airtime for Wi-Fi and NR-U CWs configurations

NR-U CW	Wi-Fi CW	Jain's fairness index	Joint airtime-fairness
1	175	0.998	0.95
3	225	0.972	0.917
7	225	0.985	0.93
15	225	0.995	0.94
31	275	0.985	0.92
63	375	0.999	0.919

Table 5.1. Optimal configuration of contention windows with Jain's fairness index and Joint airtime-fairness

5.4. Comparison of Methods

Previously presented methods were designed to increase fair share in radio resource for coexistence scenarios. Four scenarios are now considered for four configurations of APs and gNBs with different CWs values configurations (Table 5.2).

Scenario	Wi-Fi CW	NR-U CW
1	standard	standard
2	standard	0
3	optimized	0
4	optimized	optimized

Table 5.2. Configuration of Wi-Fi's and NR-U's CWs

First scenario concerns the coexistence of APs with configuration from Table 4.1 (Scenario I) and gNBs using RS method with configuration from Table 4.2. This scenario is considered to compare fairness of RS method with next simulation scenarios and serves as a baseline.

Second scenario concerns the coexistence of APs with parameters as before and desynchronized gNBs using gap method, but with no backoff procedure. To achieve this, all gNB's CWmin and CWmax was set to 0. Scenario 2 already includes a method of increasing fairness (disabling backoff) due to fact that gNBs using gap method and with the default configuration (Table 4.2) do not achieve fair channel share, as stated before.

Third scenario is a further improvement of the second scenario. This time not only gNB's backoff procedure was disabled, but also AP's CW (CWmin=CWmax) was optimized in order to achieve best resource share. Wi-Fi CW was chosen from range 25 to 400, with a step of 25.

Fourth scenario is again an improvement of the previous one. For both technologies appropriate CWs were chosen so as to maximize JFI or joint-fairness index. Wi-Fi CW was chosen from range 25 to 400, with a step of 25, and NR-U CW could have value of 1, 3, 7, 15, 31 or 63.

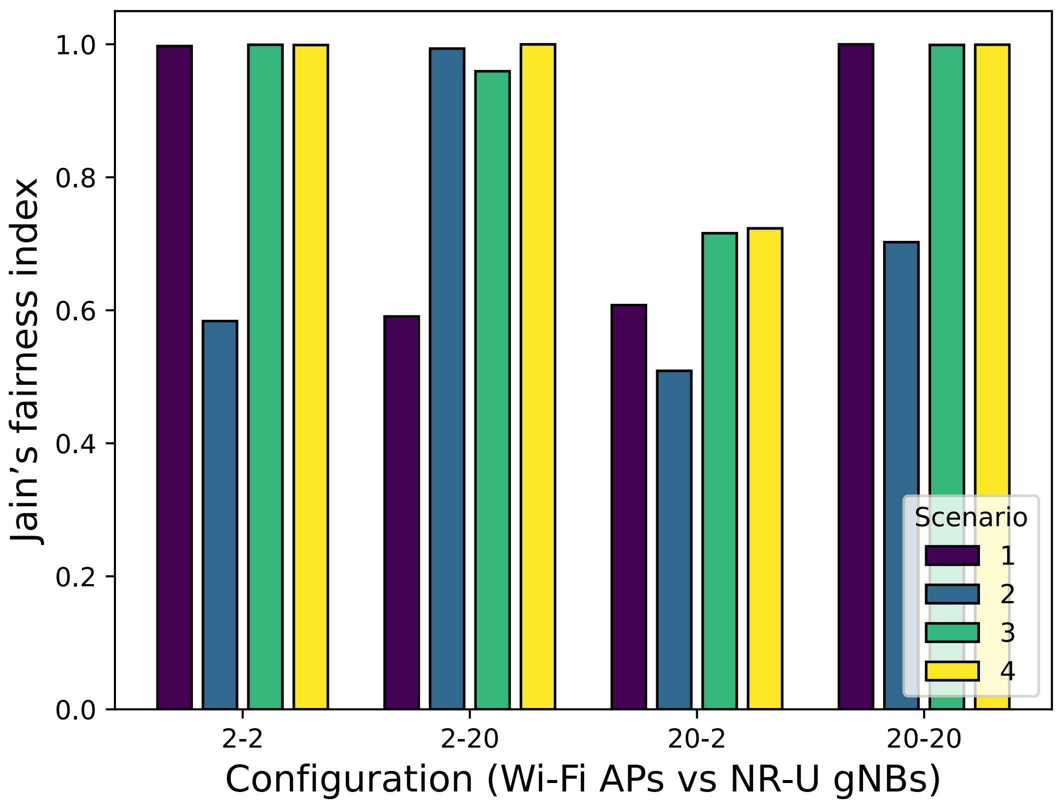


Figure 5.5. Simulation results comparing Jain's fairness index in airtime share

Figure 5.5 shows that using RS method provides fair share in scenarios where the number of stations for each technology is similar. But when there are significantly more stations of a given technology, using RS does not provide great results. Using gap with disabled backoff usually gives worse results than RS and is not enough to achieve fairness. However, when Wi-Fi CW is optimized and disable backoff for NR-U result in obtaining significant growth in Jain's fairness index value. For scenarios were amount of station for each technology were equal results were very close to best case, and the Wi-Fi CW was set to 175. In scenario that included more gNBs

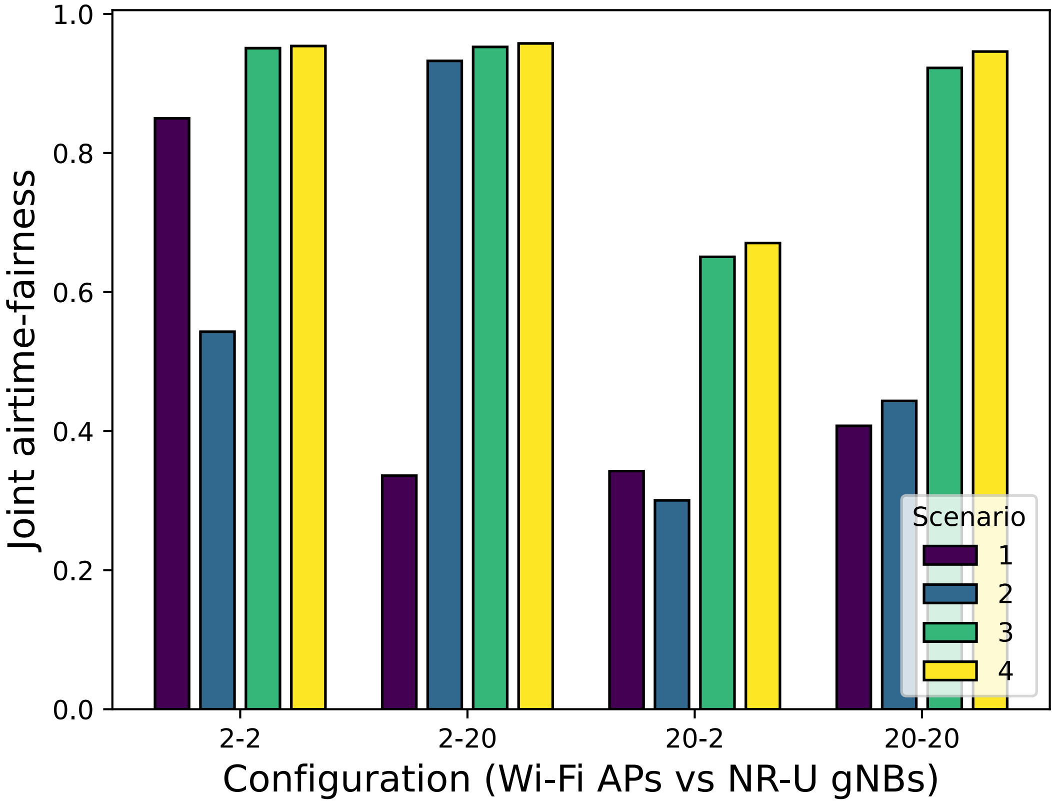


Figure 5.6. Simulation results comparing joint airtime-fairness

than AP, Wi-Fi CW was set to lowest possible value, which is 25. It allowed to ensure fair share of channel even with vast advantage of gNBs. Similar in opposite scenario, which consist of bigger number of APs, the biggest possible Wi-Fi CW was set and fair share was provided. The last scenario, in which CW for both technologies were optimized allowed to further enhance obtained fairness results. Despite the slight improvement, these differences were not significant.

Figure 5.6 shows, that when both normalized airtime and fairness is taking into consideration optimizing Wi-Fi CW results in much higher joint airtime-fairness values. In comes from the fact that, in these scenarios not only fairness in channel access is on a high level, but also optimizing Wi-Fi CW have an effect on achieving high aggregate normalized airtime. As in JFI, optimizing both Wi-Fi CW and NR-U CW gives better results, but these differences were not significant.

5.5. Conclusions

As it is presented in this chapter there are several ways to ensure fair coexistence for Wi-Fi and NR-U. Not only using NR-U with RS provides high fairness, but also using gap method with some improvements can lead to a high and fair channel share. Moreover, using gap with improvements like disabling backoff in NR-U and optimizing Wi-Fi CW can be used to provide fair channel share in scenarios with significant difference in number of stations for each technology.

6. Summary

The goal of this thesis was to implement and validate the simulator of IEEE 802.11 and NR-U Coexistence using SimPy library in Python. This goal was achieved and simulator for coexistence of IEEE 802.11 and NR-U was created. The simulator allows to perform simulations with different parameters and different number of nodes for each technology. Implementation was based on SimPy library, which allowed to create a discrete event based simulator easily. The source code is available in a GitHub repository[1].

The validation was performed in three steps. The first step was to validate only IEEE 802.11 performance by the use of previous SimPy implementation of IEEE 802.11 and Matlab simulator. The second step was validation of NR-U performance by use of another SimPy and Matlab simulator. Final step was to validate coexistence scenarios by the use of Matlab simulator. Results of created simulator overlap with results from alternative implementations. Therefore, the simulator can be considered working properly and can be used in research of coexistence in shared bands.

From a performance analysis, conducted in Chapter 5, there are several conclusions that lead to providing fair channel access in coexistence scenarios. The research was mainly focused on coexistence scenarios between Wi-Fi and NR-U using gap mode, for the reason that RS mode is not recommended by IEEE and criticized by researchers. The main findings are as follows (each presented optimization method covers the previous ones):

1. Implementing initial desynchronization between NR-U nodes results in notable improvement in the performance for NR-U. It is caused by the fact that with different slot boundaries gNBs have lower probability of collisions between each others.

2. Disabling backoff for NR-U significantly increase performance of NR-U in Wi-Fi neighborhood. Although the results obtained by disabling backoff are better than results with

[1] https://github.com/CichonJakub/5G-Coexistence-SimPy

only desynchronized gNBs, disabling backoff does not provide fair channel access for NR-U.

3. Adjusting Wi-Fi's contention window leads to equalizing per-technology airtime. This method finally allows to provide fair or close to fair channel access for each technology. The difference between results from this method and the previous one is the most remarkable.

4. Adjusting both Wi-Fi and NR-U contention windows leads to further increase in fair channel share. The difference between this method and adjusting just Wi-Fi contention window is that large.

These findings confirms the findings of [14], but are obtained by discrete event simulation instead of the Monte Carlo method.

In the process of creating the simulator some simplifications were made. The simulator presents a high level of abstraction of data transmission for both Wi-Fi and NR-U. It does not implement MAC or PHY layer, and does not take into consideration the condition of the channel, impact of noise or signal deterioration. As future work, these topics can be developed. The simulator assumes transmissions only in downlink, and the opportunity to extend it by adding possibility of uplink transmissions is viable as well. There are also several mechanisms that would add more details in NR-U transmissions (e.g., partial subframes) or in Wi-Fi transmissions (RTS/CTS mechanism or frame fragmentation).

Bibliography

[1] "Cisco Annual Internet Report (2018–2023) White Paper". In: (2022). Accessed: 2022-06-21. URL: *https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html*.

[2] J. Wszołek et al. "Revisiting LTE LAA: Channel Access, QoS, and Coexistence with WiFi". In: *IEEE Communications Magazine* 59.2 (2021), pp. 91–97. DOI: *10.1109/MCOM.001.2000595*.

[3] 5 GHz RLAN. "Harmonised Standard covering the essential requirements of article 3.2 of Directive 2014/53/EU". In: (2017). Accessed: 2022-06-21. URL: *https://ec.europa.eu/growth/single-market/european-standards/harmonised-standards/radio-equipment_en*.

[4] Katarzyna Kosek-Szott and Norbert Rapacz. "Tuning Wi-Fi Traffic Differentiation by Combining Frame Aggregation with TXOP Limits". In: *IEEE Communications Letters* 24.3 (2020), pp. 700–703. DOI: *10.1109/LCOMM.2019.2958625*.

[5] 3GPP. *Physical layer procedures for shared spectrum channel access (Release 16)*. Technical Specification (TS) 37.213. Version 16.4.0. 3rd Generation Partnership Project (3GPP), 2021. URL: *https://www.etsi.org/deliver/etsi_ts/137200_137299/137213/16.08.00_60/ts_137213v160800p.pdf*.

[6] "IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications". In: *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)* (2016), pp. 1–3534. DOI: *10.1109/IEEESTD.2016.7786995*.

[7] Marek Zając and Szymon Szott. "Resolving 5G NR-U Contention for Gap-Based Channel Access in Shared Sub-7 GHz Bands". In: *IEEE Access* 10 (2022), pp. 4031–4047. DOI: *10.1109/ACCESS.2022.3141193*.

[8] Xinghua Sun and Lin Dai. “Backoff Design for IEEE 802.11 DCF Networks: Fundamental Tradeoff and Design Criterion”. In: *IEEE/ACM Transactions on Networking* 23.1 (2015), pp. 300–316. DOI: *10.1109/TNET.2013.2295242*.

[9] “IEEE Standards for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks-Specific requirements-Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications”. In: *IEEE Std 802.3, 1998 Edition* (1998), pp. 1–1262. DOI: *10.1109/IEEESTD.1998.88276*.

[10] Polina Kutsevol et al. “New Collision Detection Method for Fair LTE-LAA and Wi-Fi Coexistence”. In: *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. 2019, pp. 1–6. DOI: *10.1109/PIMRC.2019.8904230*.

[11] P. Nikolich. “Liaison Statement to 3GPP from IEEE 802 LMSC”. In: IEEE 802.19-16/0037r9 (2016). Accessed: 2022-06-21. URL: *https://mentor.ieee.org/802.19/dcn/16/19-16-0037-09-0000-laa-comments.pdf*.

[12] Michal Cierny et al. “Fairness vs. Performance in Rel-13 LTE Licensed Assisted Access”. In: *IEEE Communications Magazine* 55.12 (2017), pp. 133–139. DOI: *10.1109/MCOM.2017.1700032*.

[13] Loginov V.A., Lyakhov A.I., and Khorov E.M. “Coexistence of Wi-Fi and LTE-LAA Networks: Open Issues”. In: *J. Commun. Technol. Electron* 63 (2018). Accessed: 2022-06-21, 1530–1537. DOI: *https://doi.org/10.1134/S1064226918120148*.

[14] Ilenia Tinnirello et al. “No Reservations Required: Achieving Fairness between Wi-Fi and NR-U with Self-Deferral Only”. In: MSWiM ’21. Accessed: 2022-06-21. Alicante, Spain: Association for Computing Machinery, 2021, pp. 115–124. ISBN: 9781450390774. DOI: *10.1145/3479239.3485680*. URL: *https://doi.org/10.1145/3479239.3485680*.

[15] Katarzyna Kosek-Szott et al. “Downlink channel access performance of NR-U: Impact of numerology and mini-slots on coexistence with Wi-Fi in the 5 GHz band”. In: *Computer Networks* 195 (2021). Accessed: 2022-06-21, p. 108188. ISSN: 1389-1286. DOI: *https://doi.org/10.1016/j.comnet.2021.108188*. URL: *https://www.sciencedirect.com/science/article/pii/S1389128621002437*.

[16] Salman Saadat et al. "Enhanced Network Sensitive Access Control Scheme for LTE–LAA/WiFi Coexistence: Modeling and Performance Analysis". In: *Comput. Commun.* 172.C (2021). Accessed: 2022-06-21, 45–53. ISSN: 0140-3664. DOI: *10.1016/j.comcom.2021.03.003*. URL: *https://doi.org/10.1016/j.comcom.2021.03.003*.

[17] Jie Xiao et al. "Performance Modeling of LAA LBT with Random Backoff and a Variable Contention Window". In: *2018 10th International Conference on Wireless Communications and Signal Processing (WCSP)*. 2018, pp. 1–7. DOI: *10.1109/WCSP.2018.8555559*.

[18] Mohammed Hirzallah, Marwan Krunz, and Yong Xiao. "Harmonious Cross-Technology Coexistence With Heterogeneous Traffic in Unlicensed Bands: Analysis and Approximations". In: *IEEE Transactions on Cognitive Communications and Networking* 5.3 (2019), pp. 690–701. DOI: *10.1109/TCCN.2019.2934108*.

[19] Gaurang Naik et al. "Next Generation Wi-Fi and 5G NR-U in the 6 GHz Bands: Opportunities and Challenges". In: *IEEE Access* 8 (2020), pp. 153027–153056. DOI: *10.1109/ACCESS.2020.3016036*.

[20] Amr Abdelfattah, Naceur Malouch, and Jonathan Ling. "Analytical Evaluation and Potentials of Frame Based Equipment for LTE-LAA/Wi-Fi Coexistence". In: *2019 IEEE Symposium on Computers and Communications (ISCC)*. 2019, pp. 1–7. DOI: *10.1109/ISCC47284.2019.8969743*.

[21] Zhiqiang Wang et al. "An LTE-U coexistence scheme based on cognitive channel switching and adaptive muting strategy". In: *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. 2017, pp. 1–6. DOI: *10.1109/PIMRC.2017.8292568*.

[22] Geeth P. Wijesiri and Frank Y. Li. "Frame Based Equipment Medium Access in LTE-U: Mechanism Enhancements and DTMC Modeling". In: *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*. 2017, pp. 1–6. DOI: *10.1109/GLOCOM.2017.8254612*.

[23] Mihaela Beluri et al. "Mechanisms for LTE coexistence in TV white space". In: *2012 IEEE International Symposium on Dynamic Spectrum Access Networks*. 2012, pp. 317–326. DOI: *10.1109/DYSPAN.2012.6478155*.

[24] Ran Zhang et al. "Modeling and Analysis of MAC Protocol for LTE-U Co-Existing with Wi-Fi". In: *2015 IEEE Global Communications Conference (GLOBECOM)*. 2015, pp. 1–6. DOI: *10.1109/GLOCOM.2015.7417440*.

[25] Mehdi Haghshenas and Maurizio Magarini. "NR-U and Wi-Fi Coexistence Enhancement Exploiting Multiple Bandwidth Parts Assignment". In: *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*. 2022, pp. 260–263. DOI: *10.1109/CCNC49033.2022.9700517*.

[26] Frederic Launay. "5G-NR Radio Interface – Bandwidth Part". In: *NG-RAN and 5G-NR: 5G Radio Access Network and Radio Interface*. 2021, pp. 191–202. DOI: *10.1002/9781119851288.ch7*.

[27] Fuad Abinader et al. "Impact of Bandwidth Part (BWP) Switching on 5G NR System Performance". In: *2019 IEEE 2nd 5G World Forum (5GWF)*. 2019, pp. 161–166. DOI: *10.1109/5GWF.2019.8911626*.

[28] Han Sol Kim et al. "Interference analysis study for coexistence between C-V2X and Wi-Fi 6E in the 6GHz band". In: *2022 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. 2022, pp. 101–103. DOI: *10.1109/ICAIIC54071.2022.9722698*.

[29] Hager Hafaiedh et al. "WiFi/LTE learning based Qos-aware Coexistence". In: *2020 9th IFIP International Conference on Performance Evaluation and Modeling in Wireless Networks (PEMWN)*. 2020, pp. 1–6. DOI: *10.23919/PEMWN50727.2020.9293087*.

[30] L. Giupponi et al. "Simulating LTE and Wi-Fi Coexistence in Unlicensed Spectrum with ns-3". In: (Apr. 2016).

[31] Vyacheslav Loginov, Andrey Lyakhov, and Vyacheslav Zhdanovskiy. "Performance Analysis of LBT with Limited Reservation Signal Duration for Fair LTE-LAA/Wi-Fi Coexistence". In: *2019 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*. 2019, pp. 1–5. DOI: *10.1109/BlackSeaCom.2019.8812810*.

[32] H. Zarrinkoub. "Understanding LTE with MATLAB: From Mathematical Modeling to Simulation and Prototyping". In: 2014.

[33] Baoan Jia and Meixia Tao. "A channel sensing based design for LTE in unlicensed bands". In: *2015 IEEE International Conference on Communication Workshop (ICCW)*. 2015, pp. 2332–2337. DOI: *10.1109/ICCW.2015.7247529*.

[34] Narongchon A-mapat and Phichet Moungnoul. "Mitigation technique for LTE-LAA and LTE-LWA coexistence". In: *2020 6th International Conference on Engineering, Applied Sciences and Technology (ICEAST)*. 2020, pp. 1–3. DOI: *10.1109/ICEAST50382.2020.9165524*.

[35] Paweł Topór. “An IEEE 802.11 Channel Access Simulator based on the Python SimPy Library”. In: (2021). Accessed: 2022-06-21. URL: *https://github.com/ToporPawel/DCF-Simpy*.

[36] Marek Zając. “An NR-U Channel Access Simulator based on the Python SimPy Library”. In: (2021). Accessed: 2022-06-21. URL: *https://github.com/marekzajac97/nru-channel-access*.