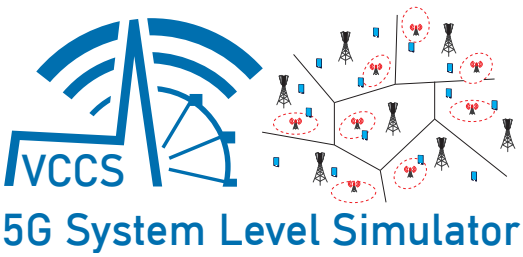


User Manual

The Vienna 5G System Level Simulator

Institute of Telecommunications,
TU Wien

Authors

Agnes Fastenbauer, Armand Nabavi, Alexander Bokor, Areen Shiyahin,
Thomas Lipovec, Jan Nausner, Christoph Buchner, Blanca Ramos Elbal,
Thomas Dittrich, Lukas Nagel, Martin Müller, Fjolla Ademaj,
Stefan Schwarz and Markus Rupp

Vienna, August 25, 2022

Institute of Telecommunications, TU Wien

Gußhausstraße 25/389
A-1040 Vienna
Austria

web: https://www.tuwien.at/etit/tc/en/

Contents

1 Introduction

In cellular communications, simulations are an inevitable tool for understanding the mutual interactions of all involved players in the network. Especially for gaining insight in the performance of a large-scale scenario, a real-world measurement approach becomes too costly and laborious. Therefore, system level simulators are developed along with the standardization process of the current mobile communications standard.

Our research group originally started off in 2009 with a freely-available Long Term Evolution (LTE) link-level simulator and supplemented it with an LTE system-level simulator. These simulators received quite some attention and kept growing over time, adding more features as research and standardization evolved further. Thanks to the open-source nature of our simulators and the vivid exchange between developers and active users via an online forum, the Vienna LTE simulators have been downloaded more than 50 000 times in total.

Following the evolution of LTE towards the 5th generation of mobile networks (5G) and the 6th generation of mobile networks (6G) we have introduced new 5G simulators to remain at the forefront of the latest developments. We again follow the approach to split this project in a link-level and a system-level simulator. The scope of this user manual is to give an overview of the capabilities and the general purpose of the Vienna 5G System Level (SL) Simulator, introduce its structure and describe the key features and their implementation details.

With the addition of the 5G System Level Simulator to the family of the Vienna Cellular Communications Simulators (VCCS), we tackle the need for simulating large scale networks, capturing the change in network layouts and physical transmission. Our simulator allows to create networks of arbitrary layout with several tiers of base stations and various user types in the same simulation.

The implementation is done in MATLAB and uses Object Oriented Programming (OOP). In general we made sure that the code structure is easy to expand, also with respect to the unknown prerequisites from the upcoming 6G standard. The usage of parallel computing is also supported by our simulator, since individual simulation chunks are defined after an initial pregeneration step.

Our simulator performs Monte-Carlo simulations in order to achieve an average network performance. Therefore, we average over many spatial constellations and channel realizations and thus obtain results for average throughput per user/base station, average Signal to Interference and Noise Ratio (SINR) performance and ratio of successful transmissions.

To determine the quality of each individual link, the instantaneous SINR is evaluated. For each transmission, the received power of all transmitters (desired and interfering) is calculated by combining distance dependent path loss, channel realization, antenna pattern and shadowing. It is possible to choose from several models and options for each of these individual propagation effects. Additionally, this is not a static choice that is set for the whole simulation, but is chosen dependent on the link conditions (e.g., Line-Of-Sight (LOS)/Non Line of Sight (NLOS)). These link-types can again be distinguished by different means. To stay with the LOS example, options are, e.g., to use probabilities of link obstruction or to find obstruction of the transmission link from explicitly placed blockages in the scenario (a more explicit description can be found in Section 7.4).

Since the complexity of a simulation increases rapidly with the considered network size and the resulting large number of individual network elements, we perform an abstraction step for the actual transmission. Therefore, performance curves from the 5G link level (LL) Simulator are used for an SINR to Bit Error Ratio (BER) mapping to assess the success of each individual transmission, without the need to simulate all aspects of the Physical (PHY) channel.

The current version of our 5G System Level Simulator supports heterogeneous networks with an arbitrary number of base station tiers and user types, including mobile users. Regarding the network geometry, not only base stations and users can be placed, but also 3D blockages, resembling walls and buildings. Consequently, randomly generated cities can be created, such as a Manhattan grid layout or randomly placed buildings with arbitrary orientation, but also building data from real cities can be chosen as network environment. The new transmission features of 5G, such as mmWave and massive Multiple-Input Multiple-Output (MIMO) are represented in our simulator by the corresponding channel model for the right frequency range.

2 Quick Start

This quick start guide explains how to run the Vienna 5G SL Simulator for the first time.

1. Switch to the simulator's root directory. Make sure the file `simulate.m` is present in your current working directory.

2. Open one of the scripts in `launcherFiles`. For illustration, let's consider one of the predefined launchers, `launcherFiles.launcherExample.m`

3. In this launcher file, the scenario to be used is specified:

```
result = simulate(@scenarios.basicScenario, parameters.
    setting.SimulationType.local);
```

4. Simulations are defined by their scenario files. In the folder `+scenarios` we find `basicScenario.m` which contains all the parameters that are needed to perform a simulation. To run your own simulation you can adapt this or one of the other scenarios to your needs or create your own scenario file. The scenario file `example.m` contains a list of all parameters that can be set and their respective default values. It can be used as a template for creating your own scenario. More information about scenarios can be found in Section 4.

5. In case you want to make use of multiple processor cores and parallelize the simulation, you can use `SimulationType.parallel` instead of `SimulationType.local` in step 3 (cf. Section 7.3).

6. Run the script `launcherFiles.launcherExample.m`. It runs the simulation and plots various results. The results can be found in the variable `result`.

```
% shows all available plots
result.showAllPlots;
```

3 System Requirements

The simulator is implemented using MATLAB. The current minimum required version is 2019b.

3.1 Matlab Toolboxes

It is possible to use the VCCS 5G System Level Simulator (SLS) without any toolboxes. However, the *Parallel Computing Toolbox* is required to use the parallel simulation mode of the simulator, the random user positioning relies on functions from the *Statistics and Machine Learning Toolbox*, and some traffic models rely on the *Statistics and Machine Learning Toolbox*. The majority of the implementation is done relying only on MATLAB built-in functions, supported without any toolbox. The functions employing methods from toolboxes are listed in Table 1. Note that the usage of those functions are optional, and therefore are not required in order to run the simulator.

function	**usage**	**toolbox**
`parfor`	`ParallelSimulation.m`	Parallel Computing Toolbox
`poissrnd`	`PoissonStreets.m`	Statistics And Machine Learning Toolbox
`poissrnd`	`NodeDistribution.m`	Statistics And Machine Learning Toolbox
`poissrnd`	`ClusteredDistribution.m`	Statistics And Machine Learning Toolbox
`binornd`	`UrbanMacro3D.m`	Statistics And Machine Learning Toolbox
`binornd`	`UrbanMacro5G.m`	Statistics And Machine Learning Toolbox
`makedist`	`FTP.m`	Statistics And Machine Learning Toolbox
`makedist`	`Gaming.m`	Statistics And Machine Learning Toolbox
`makedist`	`HTTP.m`	Statistics And Machine Learning Toolbox
`makedist`	`VideoStreaming.m`	Statistics And Machine Learning Toolbox
`random`	`FTP.m`	Statistics And Machine Learning Toolbox
`random`	`Gaming.m`	Statistics And Machine Learning Toolbox
`random`	`HTTP.m`	Statistics And Machine Learning Toolbox
`random`	`VideoStreaming.m`	Statistics And Machine Learning Toolbox

Table 1: Employed functions requiring MATLAB toolboxes.

4 Predefined Scenarios

In Section 2 we described how to run a simulation. To define a simulation we use scenario files. The desired parameters for a specific scenario are set in these files. Undefined parameters will be initialized with their default values. The default values are listed in the scenario file `+scenarios/example.m`. With the scenario files, all options are concentrated in one spot and are not distributed in hidden configuration files. A scenario file contains a function which expects one parameter of the type `parameters.Parameter` and returns the parameter object filled with settings required by the scenario. Scenarios are launched by passing their function handle to the `simulate` function. For example to start the basic scenario the function call would look like this:

```
result = simulate(@scenarios.basicScenario, parameters.
    setting.SimulationType.local);
```

The second argument of this function determines if the simulation is performed with or without parallelization.

One benefit of the configuration through scenario files is its flexibility. The only constraints are that the function has to take a `parameters.Parameter` object as input and returns this object with the properties set are required. Besides this, one can execute arbitrary MATLAB code in these functions. For example one can use calculations to find the right parameter values, call other functions or simply load data from configuration/data files. This flexibility allows to customize the scenario to a large degree.

4.1 Customization

For defining new scenarios we recommend to start from the existing scenario that is most similar to the intended simulation and modify a copy of it.

To change additional parameters that are not already set in the existing scenario, refer to `+scenarios.example`. In this file all parameters are set to the predefined default values. This serves as a reference for which parameters do not need to be set, because they are set to the desired value by default as well as an example on how to set each parameter, if it needs to be changed from the default value. For further explanations on the parameters, refer to the built-in documentation for each parameter class by using the following line in the command line:

```
        doc parameters.'subpackage'.'Class'.'property'
```

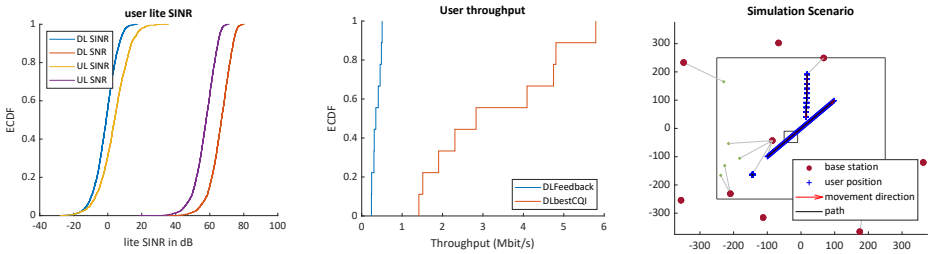
For example to understand the use of the setting `nElements` for Poisson distributed users type:

```
doc parameters.user.Poisson2D.nElements
```

in the command line. This opens a documentation window explaining what data types are expected for this setting and what is defined through this parameter.

In the package `+launcherFiles` several launchers for simulating predefined scenarios can be found. The corresponding scenario files are defined in `+scenarios`. In the following, the specifics of these scenarios are explained.

4.2 Example Launcher with Basic Scenario



(a) (b) (c)

Figure 1: Plots produced by the `launcherExample`.

In this launcher and scenario a basic network geometry is defined and simulated. To start, a building is placed in the simulation region, then base stations and users are distributed in the simulation region through a Poisson point process. An additional user is placed with manually defined positions to show how to set up predefined user positions. The simulated region of interested is extended by an interference region. The base station placement is extended to this interference region. The purpose of this additional region is to assure that users, that are placed close to the border of the original region of interest, experience network conditions comparable to the users at the center of the region of interest. Without the interference region, users close to the border would experience little to no interference, which would distort the simulation results.

The base stations are equipped with three-sector antennas, where each sector antenna has four antenna elements at each sector antenna. The users are equipped with two receive antenna elements and move at a very high speed to show the user movement when displaying the results. The users use vehicular Power Delay Profile (PDP) channel models to model the small scale fading that affects the channel between them and the base stations. The

trace length of the PDP models has to be increased to provide a sufficiently large number of channel realizations. More details on the channel trace can be found in Section 7.4.3.

The simulation time line is set up in several chunks to allow for parallelization of the simulation which reduces the simulation time.

After the simulation is run in the launcher file, some results are plotted. First, the *lite* Signal to Noise Ratio (SNR) and SINR is plotted for uplink and downlink transmission. The plot shows that the SNR is much higher than the SINR, showing that the interference from other cells has a high impact on the received signal quality. One realization of this plot is shown in Fig. 1a. Since the network elements are randomly positioned at each simulation run, the produced result varies each time the simulation is performed.

The second plot produced, see Fig. 1b, shows the user throughput empirical cumulative distribution function (ecdf) for the throughput achieved with the Channel Quality Indicator (CQI) values reported by the feedback and the maximum throughput achieved, if the highest CQI for which transmission is successful is used. A large gap between the the two throughput ecdfs can be observed. This is due to the high user speed and the feedback delay of three slots, which render the feedback unreliable. The high user speed leads to a channel that changes faster and the feedback delay leads to the reported feedback values being outdated. This shows that, for very large user speeds, only low throughput can be achieved if the transmission relies on channel feedback.

In a third plot the simulation scenario is plotted showing the border of the region of interest, the building placed in the simulation region the base stations and the users served by each base station, as well as the user movement. One network realization is shown in Fig. 1. It is recommended to run the launcher file and zoom in the plot to see the different movement patterns of the users. On type of users moves in a random walking pattern, changing direction in each slot. One type of users moves in a randomly chosen constant direction and one user moves in the predefined straight line from lower left to upper right through the origin of the simulation region.

4.2.1 Simulation of several chunks in parallel

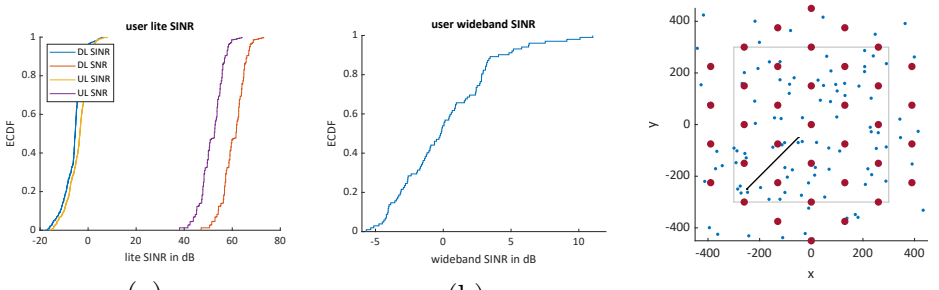
The launcher file `launcherFiles.launcherExample` can also be used with parallel simulation mode. For this, the simulation mode has to be changed from `local` to `parallel` in the `launcherFiles.launcherExample`.

```
1 result = simulate(@scenarios.basicScenario, parameters.
     setting.SimulationType.local);
2 % -> should be replaced with
```

```
result = simulate(@scenarios.basicScenario, parameters.
    setting.SimulationType.parallel);
```

It is recommended to run the local and the parallel simulation and observe the simulation time that is reduced in the parallel simulation case. More insights on parallelization gains that can be achieved can be found in Section 8.

4.3 Lite version



(a) (b) (c) Simulation Scenario

Figure 2: Plots produced by the `launcherExample`.

The launcher file `launcherFiles.launcherLiteSimulation.m` shows how to run a *lite* simulation. The corresponding scenario file for this launcher is `scenarios.basicLiteScenario.m`. As explained in Section 7.2, the *lite* mode can be used in all types of simulations. In this example in particular, users are deployed according to a Poisson Point Process (PPP) and base stations are placed in rings of a hexagonal grid. Additionally, a wall is placed in the simulation region to show how to set up manually defined walls in a simulation. All parameters can be changed, but in this type of simulation some parameters like the `scheduler` are irrelevant. The line of code that defines, whether a simulation is *lite* or not is:

```
params.postprocessor = simulation.postprocessing.
    LiteWithNetworkPP;
```

To select the *lite* mode, we can either use `LiteWithNetworkPP` or `LiteNoNetworkPP`. The difference between this lite postprocessors is explained in Section 7.14.

Once the simulation is done, the `lite` results are plotted. Exemplary realizations of one simulation run are shown in figure Fig. 2

4.4 Multi-tier and multi-user heterogeneous network

The launcher file `launcherFiles.launcherHetNet` provides an example of how to simulate a heterogeneous network consisting of different base station types and different user types. The launcher file starts by calling the corresponding scenario file located in `+scenarios`, in this case `scenarios.HetNet`. The scenario also shows how to set different path loss models for the different link types present in a multi-tier simulation. In addition, Non Orthogonal Multiple Access (NOMA) is used and to assure that femto cells have users associated to them, a cell association is set in favor of femto cells.

4.5 Manhattan city layout with QuaDRiGa channel model

The launcher file `launcherFiles.launcherManhattanQuadriga.m` shows how to use the QUAsi Deterministic RadIo channel GenerAtor (QuaDRiGa) channel model for a Manhattan city layout scenario. The QuaDRiGa channel model is selected with the following commands:

```
1 poissonUsers.channelModel   = parameters.setting.ChannelModel
    .Quadriga;
2 manhattanUsers.channelModel = parameters.setting.ChannelModel
    .Quadriga;
```

The scenario also shows how to set up a 2D antenna array, use Hybrid Automatic Repeat Request (HARQ) and the best CQI scheduler.

4.6 OpenStreetMap city layout

The launcher file `launcherFiles.launcherOpenStreetMap.m` provides an example of how to simulate buildings and streets arranged according to a real world city layout based on data from the OpenStreetMap database. This data is made available at [1] under the Open Database License (ODbL). The launcher file starts by calling the corresponding scenario file located in `+scenarios`, in this case `scenarios.openStreetMap`.

The scenario builds base stations with two technologies, LTE and 5G, that serve the respective users. Dynamic spectrum scheduling is used that allocates spectrum to each technology according to user traffic. Up to 1024-QAM is used for transmission.

5 Verification and Validation

5.1 Verification

More details on the verification of the Vienna 5G SL Simulator can be found in [3].

5.2 Comparison to LTE-A SL Simulator

The aim of this comparison is to show that results obtained with the Vienna Long Term Evolution-Advanced (LTE-A) SL Simulator can be reproduced with the Vienna 5G SL Simulator. For this purpose, the same parameter set is used in both simulators. The chosen parameters can be found in Table 2.

Parameter	Value
base stations	hexagonal layout, 1 ring, 7 Base Station (BS)s
users	50, uniform density
pathloss model	COST231 Urban Macro (UMa)
channel model	Pedestrian A PDP
TTIs/slots	100
feedback delay	3
user speed	30 km/h

Table 2: Simulation parameters used to compare the Vienna LTE-A SL Simulator and the Vienna 5G SL Simulator

The simulations were run 200 times to make sure that the results can be compared fairly. Several metrics are used for this comparison.

At first, it is important to show that the user placement works the same way in both simulators. This is demonstrated by the ecdf of the distances between the users and their assigned BS as depicted in Fig. 3.

Next, the ecdf of the macroscopic SINR as shown in Fig. 4 is examined. The close match of the curves indicates that the statistics of the macroscopic path loss computed by both simulators are very similar to each other.

Finally, the ecdf of the average user throughput obtained by both simulators is compared in Fig. 5.

Here, small discrepancies between the curves can be noticed. They are caused by minor implementation differences between the simulators. Nevertheless, these simulations still show that the channel models and the function of the Multiple Access Channel (MAC) layer behave similarly in both simulators, even with a non-zero feedback delay and a non-zero user speed.

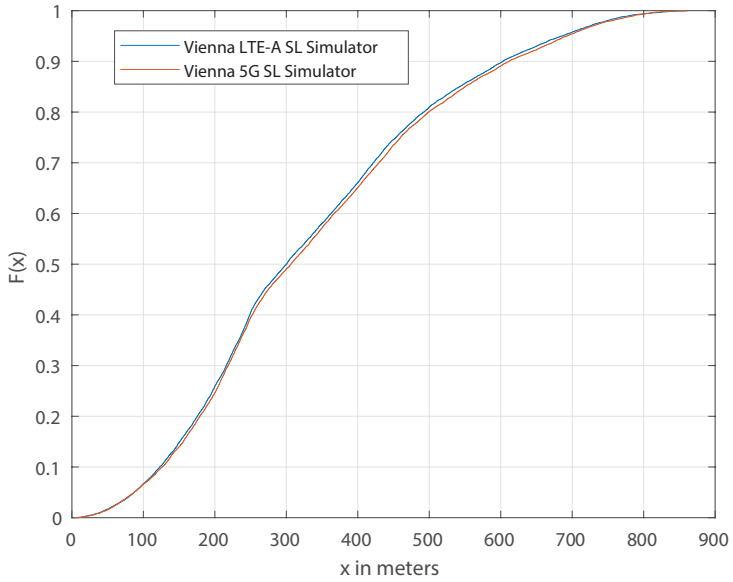


Figure 3: ecdf of the distances between users and their assigned BS.

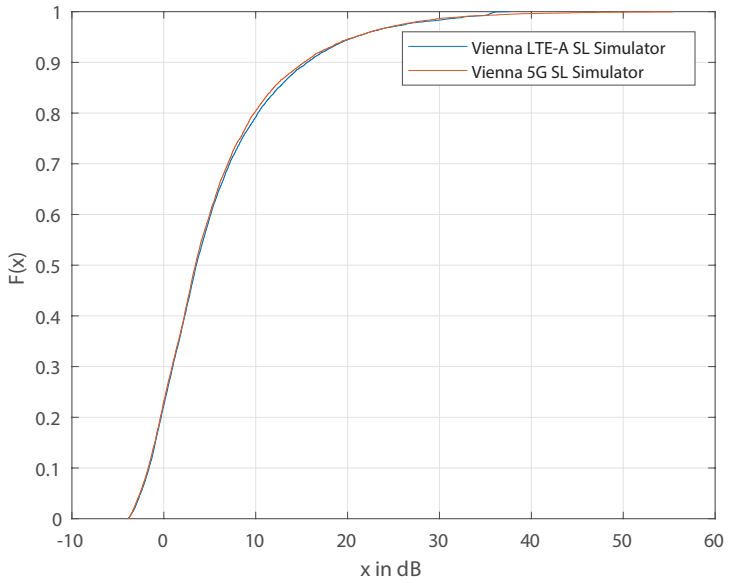


Figure 4: ecdf of the macroscopic SINR.

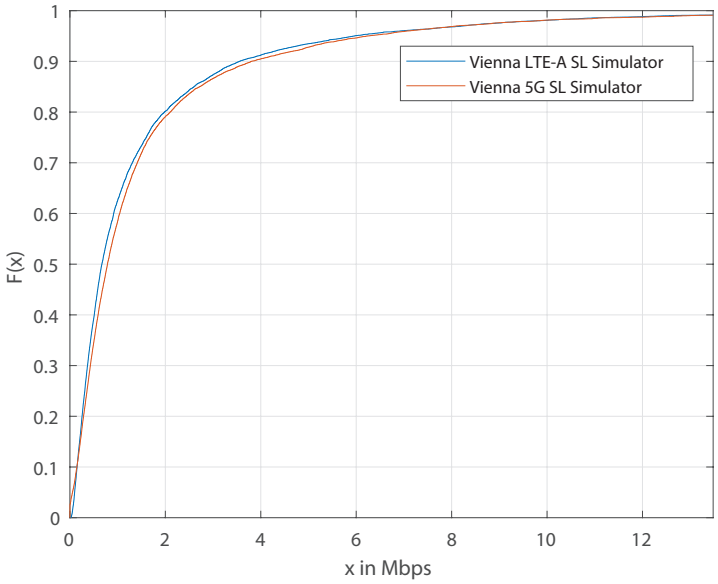


Figure 5: ecdf of the average user throughput.

6 Simulator Structure

6.1 A Typical Simulation

The Vienna 5G SL Simulator is written in MATLAB and is utilizing OOP. Individual parts of the simulator are separated into different packages and, e.g., network elements, such as BSs, are defined in classes. In general, the simulator is written in a modular fashion, such that new functions can be added easily, without the need to alter other parts of the code.

The simulator's structure is defined by four major parts, which are displayed in Fig. 6.

In the following, the most important steps of a simulation are explained and the corresponding functions to the four major parts of the simulator are introduced. [1]

Each simulation begins with running the desired simulation launcher file (cf. Section 4). This already fixes the parameter set which is going to be used for this particular simulation. In the file `simulate.m`, you find the line

```
%create simulation object
localSimulation = simulation.LocalSimulation(params);
```

where the simulation object is created and the predefined parameters are attached to this object.

Next, the simulation time line (cf. Section 6.2) is generated, as well as the network element objects (cf. Section 7.1). Additionally, the fast fading traces are pregenerated, if necessary. This is done in

```
%setup simulation and generate network elements
localSimulation.setup();
```

All of this is stored in `localSimulation.simulationSetup`.

Now that the pregeneration is done, the actual simulation is carried out. The following line contains the main simulation loop (over chunks and time slots (TSs) - cf. Section 6.3):

```
%main simulation loop
localSimulation.run();
```

Here, random samples in time (represented by channel realizations and scheduling decisions) and/or in space (represented by the geometry of the network elements) are simulated and the individual results for each TS are stored.

The postprocessing step then follows in

[1] Here we describe a local simulation - i.e., we set `SimulationType.local`. The explained steps are the same for a parallel simulation.

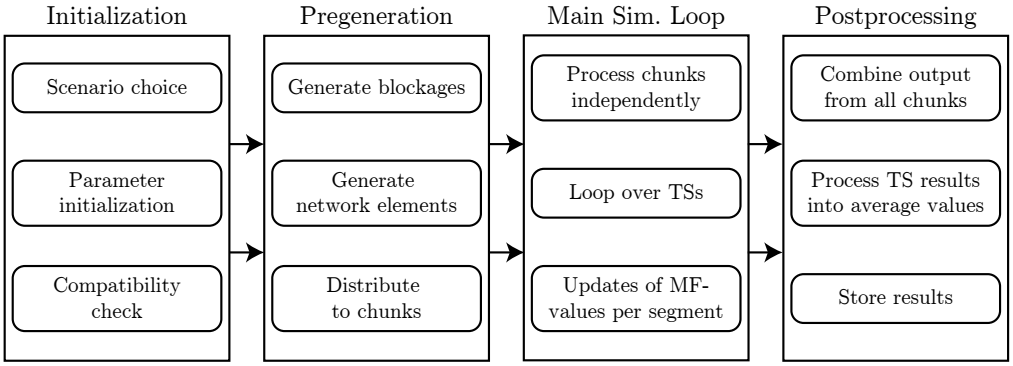


Figure 6: Overview of the main parts of the simulator.

```
% get results
simulationResults = localSimulation.simulationResult;
```

where the individual results from the main simulation loop are extracted and stored in the results object. Which results are actually calculated and stored also has to be specified in the simulation parameters. After processing and storing the results, selected results are plotted, which is integrated in the simulation launcher.

6.2 The Simulator Time Line

The time line of the simulator is divided into three different units, namely *time slots (TSs)*, *segments* and *chunks*. The TS is the shortest unit and also corresponds to the scheduling granularity. Thus it corresponds to one iteration of the inner simulation loop (cf. Section 6.3). It has a constant length, e.g., 1 ms to represent an LTE-A subframe, but can otherwise be specified freely. A segment consists of various time slots and corresponds to the time (and distance) in which the macroscopic fading (MF) is assumed to be constant (e.g., the user association, large scale path loss values, but not the channel realization). Thus, the MF values are only updated at the beginning of a segment, including the user association. The length of a segment depends on the user speed and trajectory, as well as the correlation distance of the MF values. This means that for a stationary scenario, only a single segment is created. A chunk consists of a fixed number of TSs and on one or more segments (depending on the maximum user speed). For creating the user trajectory, a consecutive generation is assumed, but also a considerably long distance between chunks, which leads to uncorrelated user positions among chunks. Even for stationary scenarios, channel coefficients are assumed to be

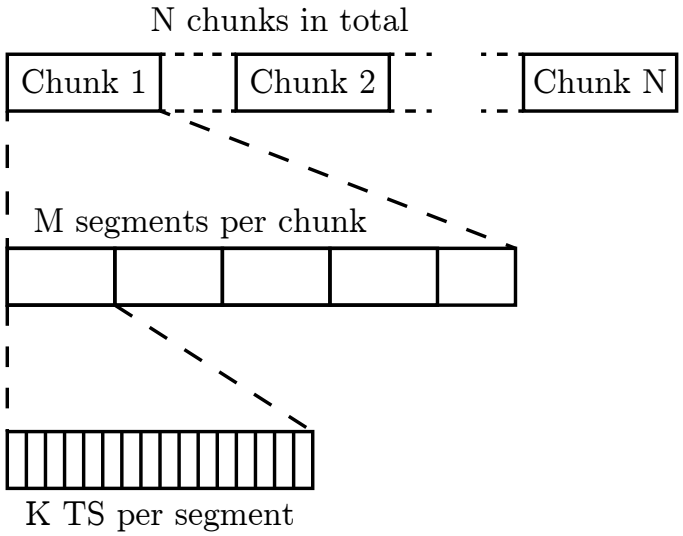


Figure 7: The different time units utilized in the simulator.

uncorrelated, due to the assumption of a non-constant scattering environment. The chunks are also the basis for parallelization, since all necessary data is independent for each chunk and the results can be calculated without knowledge of the simulation result of the previous chunks. A more detailed description of the parallelization can be found in Section 7.3.

The following parameters are utilized to set up the time line:

```
params.time.numberOfChunks = 2;
params.time.slotDuration = 1e-3; % duration in seconds
params.time.slotsPerChunk = 10; % each chunk consists of that many slots
params.time.timeBetweenChunksInSlots = 50; % timespan between two simulated chunks
```

In the function `simulation.ChunkSimulation.m`, the following function is contained:

```
function setNewSegmentIndicator(obj)
% creates a logical array that is true for all slots that are the first in a segment
% Marks all slots, for which a user has moved further than the
% maximum correlation distance. Large scale parameters are
% constant for a segment.
```

This means that based on the trajectory and correlation distance an indicator is set, where a new segment starts.

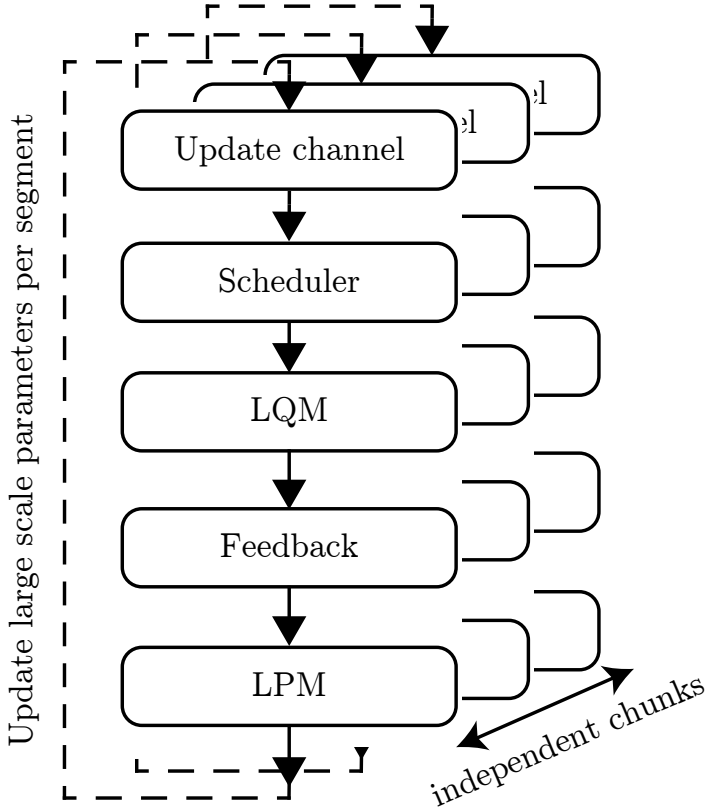


Figure 8: A representation of the main simulation loop.

6.3 The Main Simulation Loop

The main simulation loop contains a loop over chunks, whereas each chunk contains a loop over TSs (cf. Fig. 8). The loop over chunks is contained in the function `simulation.LocalSimulation.run` for local simulations and in the function `simulation.ParallelSimulation.run` for parallel simulations. For both options, all necessary data is distributed among chunks. This individual configuration per chunk is stored in `chunkSimulationList`.

Irrespective of local or parallel simulation, each chunk contains the inner loop over TSs. This loop can be found in the function `simulation.ChunkSimulation.runSimulati`

At the beginning of this loop, cell association and handovers are handled, as well as a segment update that sets the appropriate MF values for each user for the current segment. Then, in a loop over slots, the calculation of the *lite* SINR and the individual steps of a simulation depicted in Fig. 8 are performed. These include an update of the small scale fading channel, scheduling, call of the link quality model, generation of feedback and call of the link performance model.

6.4 Simulation Parameters

In order to be able to keep an overview of the simulation parameters, the simulation parameters are collected in the `+parameters` package. Within this package, the subpackage `+setting` collects list of possible parameter setting with short explanations of the setting's influence on the simulation. It is strongly recommended to use the built-in documentation of this package to set the simulation parameters. The built-in documentation can be accessed by typing `doc filename` or `help filename` in the MATLAB command line. For example, to access a list of available small scale fading channel models, type `doc parameters.setting.ChannelModel` in the command line. This opens a window with the available documentation.

As an additional help to set the parameters, the function `+scenarios.example` contains a list of all parameters and their default values. The file shows how to set parameters for different settings in the scenario file. It is encouraged to copy the parameter settings from the `example` file into the scenario file of a simulation to set up the desired scenario.

7 Overview of Key Functionalities

7.1 Generation of Network Elements and Geometry

System level simulations aim to evaluate the performance of a large network comprising a substantial number of BSs and users. In this regard, at its core the Vienna 5G SL Simulator simulates the communication between users and BSs. Due to its modular structure, the simulator allows the coexistence of different BS and user types. This, on one hand, allows to simulate multi-tier networks, and on the other hand, by also supporting different user types, more diverse and realistic scenarios are supported. Additionally to various propagation models that can be used, there is the option to distribute blockage objects and use the geometry to calculate different propagation parameters.

7.1.1 Base Stations

The simulator distinguishes between the entity *BS* and *antenna*, the two objects are depicted in Fig. 9. Each BS can have one or more antennas attached, which can be seen as physical entities with a position $\{x, y, z\}$ in the 3-dimensional (3D) space. The antenna object represents an antenna or an antenna array with N_{Tx} transmit and N_{Rx} receive antennas. BSs do not have physical locations but instead, the physical positions are specified on the assigned antenna objects. This structure also enables the simulation of Distributed Antennas Systems (DASs) and Remote Radio Heads (RRHs) without additional extensions.

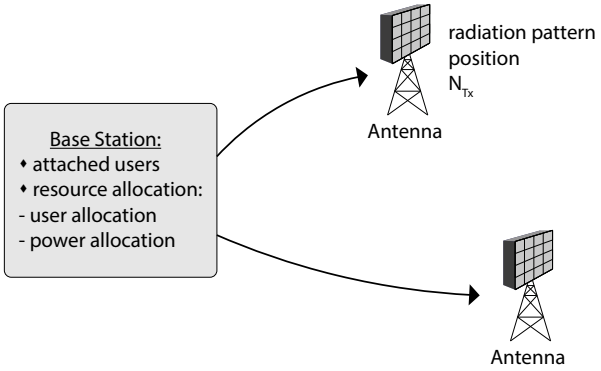


Figure 9: Abstract BS with two attached antennas with physical positions in the simulation region.

Base Station The BS object handles the physical resources available for transmission. It is defined in `+networkElements/+bs/BaseStation.m` and holds a list of all antennas that belong to this BS as well as a list of users currently attached to this BS. Each BS has a scheduler that allocates the available resources to the users attached to the BS. Each base station has an individual downlink and uplink baseband precoder.

Different placement methods that can be used to define placement of BSs are given in `+networkGeometry`. Which placement method is to be used, is specified in the scenario file by specifying the parameters for the desired placement method that can be found in `+parameters.basestation` (see 4.4 for more details). It is worth noting that in the initialization phase of the simulation, the BSs are defined through a positioning mechanism and in the main simulation loop, the BSs do not have a position in the Region Of Interest (ROI), only their attached antennas have a physical position. This is because, in the pregeneration phase, the positions generated for each BS type are moved to the antenna objects attached to the BSs.

Each BS type, e.g. macro or femto BS, is indicated with an enumeration, specified in the corresponding file `+parameters/+setting/BaseStationType.m`. If a new BS type is to be added in the simulator, then it has also to be indicated with a corresponding enumeration. Additionally a default transmit power and path loss models should be specified for new BS types.

Antenna The *antenna* object is defined in `+networkElements/+bs/Antenna.m`. The antennas have a position in the ROI and collect all parameters necessary to calculate the antenna gain for their antenna type. The scheduling information provided by the BS is stored at and accessible through the antenna object. Various antenna types are defined in `+networkElements/+bs/+antenna` and their respective parameter files for antenna creation can be found in `+parameters/+basestation/+antennas`. Two important parameters for antenna gain calculation are the antenna azimuth φ and antenna elevation θ. Fig. 10 clarifies the definition of both, where the origin would be the position of the antenna and the direction of the arrow where the antenna would have its maximum gain. The elevation is only used in the antenna gain calculation of antenna arrays. Note that $\theta = 0°$ would mean that the antenna would have its maximum gain in the direction of the zenith. The default values are $\phi = 0°$ and $\theta = 90°$.

Sectorized Base Stations The number of base station sectors is 1 by default. To simulate more than one sector, the number of BS sectors has to be specified with the BS parameter `nSectors`. The horizontal direction of

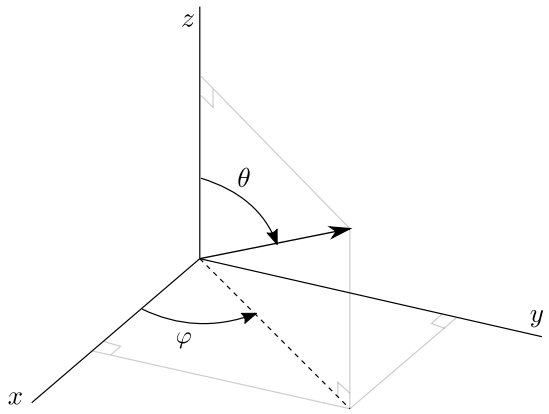


Figure 10: Illustration of antenna azimuth and elevation

the antenna, or the horizontal direction of the first sector antenna if there is more than one sector, can be specified with the antenna `azimuth` parameter. It is possible to specify up to 6 sectors. The following setting of parameters

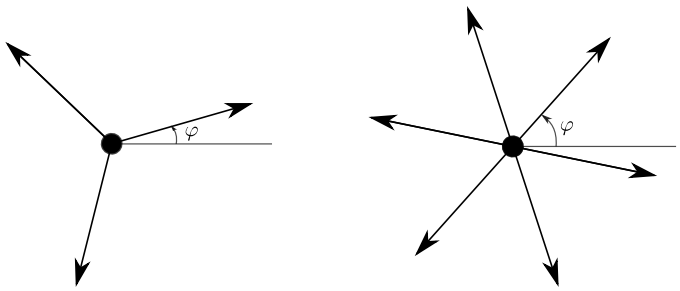


Figure 11: Base stations with multiple sectors and rotation φ

would result in a BS with three sectors like it is shown in the left part of Fig. 11. The antennas in the sectors are set to be three sector antennas and the rotation φ of the antenna in the first sector is 15 degrees. The angle between two antennas results in

$$360^\circ/3 \text{ sectors} = 120^\circ.$$

```
1 baseStation = parameters.basestation.PredefinedPositions();
2 baseStation.positions = positions;
3 baseStation.nSectors = 3;
4 antenna = parameters.basestation.antennas.ThreeSector;
5 antenna.azimuth = 15;
6 baseStation.antenna = antenna;
```

7.1.2 Users

The user object, as the other endpoint of the communication link, is defined in `+networkElements/+ue/User.m`. The user class, like the antenna, has a position in the ROI and collects parameters to define the channel between user and antenna, which include the channel model and the number of Radio Frequency (RF) transmit and receive chains.

Different placement methods that can be used to define placement of users are given in `+networkGeometry`. Which placement method is to be used, is specified in the scenario file through the parameter classes defined in `+parameters.user` (see 4.4 for more details how to set different user types within one scenario).

7.1.3 Blockages

On top of the the network element generation, the Vienna 5G SL simulator supports the generation of *blockages*. The super class of blockage elements is defined in `+blockages/Blockage.m`. The basic building block to represent signal blocking objects, is a wall with arbitrary dimensions and orientation in 3D. Buildings are then created by combining multiple walls. With these, city layouts can be generated, such as a Manhattan grid with streets and building blocks.

The buildings in the scenario are not restricted to a rectangular layout. To create an object of `+blockages/Building.m` the following properties need to be specified:

- `floorPlan` - array of x and y coordinates
- `height` - height of the building
- `loss` - loss through its walls in dB

An example of such an arbitrary building and its `floorPlan` is given in figure 12

Additionally there is the option to generate buildings together with streets, which is implemented in the `+blockages/City.m` class. This abstraction provides two mechanisms for reproducibility of simulation results, which can be configured by the following parameters in `+parameters/+city/Parameters.m`.

- `heightRandomSeed` - integer seed or `shuffle`
- `saveFile` - JSON file where the city should be stored
- `loadFile` - JSON file to load the city from

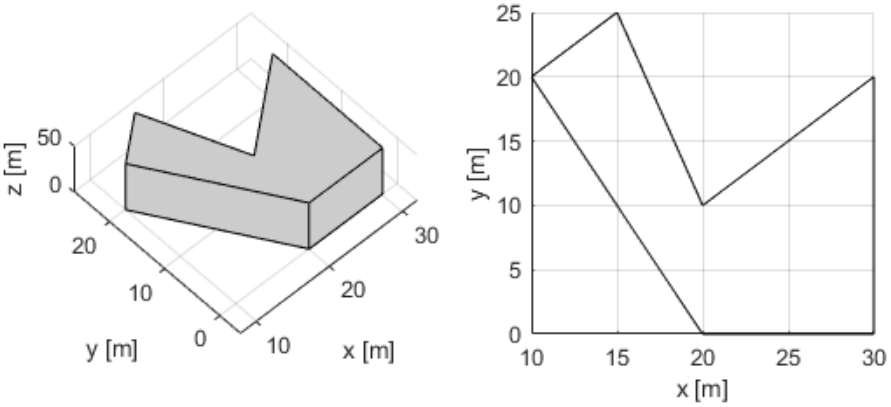


Figure 12: Building and `floorPlan`

Using the same `heightRandomSeed` will produce the same building heights on every simulation run, while using `'shuffle'` will lead to different heights on every run. If a city is stored to a JSON file, the building heights and many other parameters of buildings and streets can be adjusted manually by editing the file and then loading it again on subsequent simulation runs. The following listing shows a segment of an exemplary JSON file, where the parameters stored for every building in the city are visible.

```
1 {buildings: [{name: Karlsgasse 13, floorPlan: [[xCoords], [yCoords
    ]], height: 19.0, loss: 10}, ...], streetSystem...: { ... }}
```

There are two types of cities which can be found in `ManhattanCity.m`, where buildings and streets are generated according to a Manhattan grid layout or in `OpenStreetMapCity.m`, where streets and buildings are generated based on real-world data from OpenStreetMap [1]. Here the buildings and streets that are located in a certain area specified by coordinates are fetched via GET-request from [1]. This data is then processed and a `floorPlan` is derived for the buildings and a street layout is created. For an example of an OpenStreetMap scenario, refer to Section 4.6.

7.1.4 Mitigation of Border Effects

In SL simulations with finite simulation areas the network interference is inadequately represented for users that are close to the border of the simulated region. To mitigate these border effects, an interference region can be added to the simulation area in the Vienna 5G SL simulator. Alternatively a wraparound can be performed to simulate interference from outside of the simulation region. Which mitigation scheme is more reliable and efficient

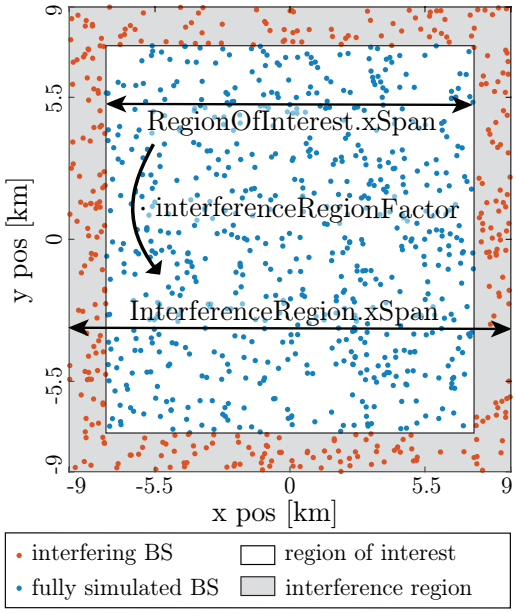


Figure 13: ROI (in white) with an additional interference region (in gray) with BSs creating additional interference for users at the border of the ROI.

depends on the size of the ROI and the path loss exponent of the used path loss model [4].

The different options for border effect mitigation schemes can be found in `parameters.setting.Interference` and the desired mitigation scheme can be set in `parameters.Parameters.regionOfInterest.interference`.

Interference Region The interference region is an addition to the ROI at its border, as depicted in figure 13.

The settings for size and user placement in the interference region are described in `parameters.regionOfInterest.RegionOfInterest`, the options for different user placements in the interference regions are listed in `parameters.setting.Interference`. The users in the interference region can either be placed automatically in the same manner as in the ROI or have to be placed manually in the interference region. The placement of the BSs and their antennas is an extension of the BS placement in the ROI with the exception of BSs with predefined positions. Therefore, if there are only BS with predefined positions in the ROI additional BS can be placed manually in the interference region with a given density specified by `parameters.basestation.InterferenceRegion`. The user placement strategy is chosen, when setting an interference region with the parameter `RegionOfInterest.interference`.

The size of the interference region is set with the parameter `RegionOfInterest.interferenceRegionFactor`. It indicates by which factor the total simulation

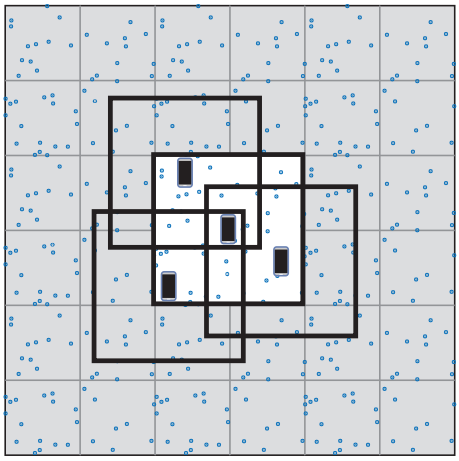


Figure 14: ROI (in white) with replicated BSs (in gray area). Each user chooses the closest BSs as virtual ROI. This is represented by black boxes surrounding each mobile station.

region is bigger than the ROI, i.e., the diameter of the ROI is multiplied by this factor to get the diameter of the interference region.

To save computational complexity the BSs in the interference region are not fully simulated unless a user of the ROI is attached to them. Instead, simplified random scheduling and feedback are used to simulate the interference region BSs.

Wraparound To perform a wraparound, the BS positions are replicated around the ROI and each user chooses the BS positions closest to it for the simulation as is indicated by the black boxes in Fig. 14. This virtually places each user at the center of its ROI, thus eliminating effects concerning the users at the border of the ROI.

7.2 *Lite* Version

For large simulation scenarios, the simulation run time can be prohibitively large, as is elaborated in Section 8. If only the instantaneous SINR and depending metrics are of interest, the *lite* simulation mode offers a faster way to obtain results that do not consider the influence of scheduling, precoding, MIMO-transmission, equalizing, or HARQ. In the *lite* mode, only geometric and depending macroscopic parameters are considered, as well as an instantaneous channel realization. Scheduling, the link quality model, feedback and the link performance model are skipped leaving only the calculation of the macroscopic parameters (path loss, antenna gain, shadowing, wall loss) and the generation of the small scale fading channel in the simulation.

The metric used to evaluate the network performance in the *lite* mode is

the *lite* SINR, which is calculated for all users in each slot, according to

$$\mathrm{SINR}_{lite} = \frac{h_d \cdot P_{\mathrm{rx},d}}{\sum_{i=1}^{N_{\mathrm{int}}} h_i \cdot P_{\mathrm{rx},i} + N}, \tag{7.1}$$

where N_{int} is the number of cells creating interference. The individual terms in (7.1) are explained for uplink and downlink separately in the following sections. The receive power P_{rx} consists of the transmit power P_{tx}, the antenna gain G, the path loss L_{path}, the shadowing L_{shadow}, and the wall loss L_{wall}:

$$P_{\mathrm{rx}} = P_{\mathrm{tx}} \cdot G \cdot L_{\mathrm{path}} \cdot L_{\mathrm{shadow}} \cdot L_{\mathrm{wall}} \tag{7.2}$$

An example of a simulation using only *lite* mode is given in Section 4.3. It should be noted that the *lite* SINR is calculated for all users in each time slot, which implies that all users are transmitting in each slot, which cannot be the case when more than one user is placed in a cell. Thus, the *lite* SINR has to be seen as a theoretical SINR, that is the SINR that could be achieved if the user was scheduled on all resources.

7.2.1 Uplink *lite* SINR

The uplink *lite* SINR is calculated by considering one user in all interfering cells as interferer. The interfering user is chosen randomly from all users associated with the interfering base station.
Thus, the received power P_{rx} is the power received by the desired base station from the transmission of the user- either the desired user, or the interfering user. The noise power N is the noise power at the base station accumulated over the whole transmission bandwidth. The small scale fading channel realization h represents the channel between the user and the desired base station. A Single-Input Single-Output (SISO) transmission is assumed, with h being the normalized channel coefficient, where normalized means that its mean power is one.

7.2.2 Downlink *lite* SINR

For the calculation of the downlink *lite* SINR, the base stations (and all of their antennas) are considered as interferers. Thus, the receive power P_{rx} is the power received by the user from a base station, the small scale fading channel realization h is the channel between a base station and the user for which the *lite* SINR is calculated. The noise N is the noise experienced at the user over the whole transmission bandwidth.

7.3 Parallelization

The two different simulation types, `local` and `parallel`, are defined in `parameters.setting.SimulationType`. In a parallel simulation the independent chunks are simulated in parallel in a `parfor` loop. This requires the *Parallel Computing Toolbox*. In case of a local simulation the independent chunks are simulated sequentially in a regular `for` loop and no additional toolbox is required.

The chunks, that are described in more detail in Section 6.2, are generated such that the simulation of each chunk can be executed independently from all other chunks. For this, copies of all network elements are created and distributed to the chunks. These copies only contain the network element position of the respective chunk. In each chunk, a new simulation is started with an empty feedback buffer and an empty transmit buffer. The first slots of each chunk are discarded from the results, since they are generated without valid feedback. The parameter `parameters.Time.feedbackDelay` indicates the number of slots for which the results are discarded. See Section 7.14 for more details on the post processing of results.

It should be noted that different types of user movements behave differently with respect to the chunk. For some user movements, the time between the chunks is considered for the movement of the users, for others, it is assumed that the users stay at the same position and start their movement again at the beginning of the new chunk.

In Section 8, an overview of the performance gains that can be achieved through parallelization is given. In general, the simulation time can be reduced substantially with parallelization. However, for very short, very small scenarios the cost of distribution to chunks can outweigh the benefits of parallelization.

7.4 Simulation of Propagation Effects

7.4.1 Path Loss Modeling and Situation dependent Model Choice

The aim, when implementing the path loss modeling was to assure modularity and flexibility on one hand but a simple, straight forward usage on the other.

As for the other parts of the simulator an object oriented approach was chosen. The superclass `PathlossModel` ensures that all path loss models can be used in the same way with the functions set in it. Like the other models, it can be found in the package `macroscopicPathlossModel`.

For a description of the necessary parameters to be set, an enumeration file `parameters.setting.PathlossModel` lists and describes all available path loss models. Another advantage of this enumeration file is that it offers auto

complete for the path loss model names, when choosing a model. Some models may require additional parameters to be set. Therefore the user is able to specify an additional parameter object for each model. A good example for that function is the `FreeSpace.m` model. The model defines a power damping factor, which is in our case called α. This factor has a default value of two but can be changed by the needs of the user. To achieve that, a parameter object of type `parameters.pathlossParameters.FreeSpace.m` has to be instantiated. The model enumerations are stored with theire associated parameters in an object of class `parameters.PathlossModelContainer.m`. The `parameters.PathlossModelContainer.m` supports different types of models to be set for each combination of BSs type, indoor/outdoor users and LOS/NLOS links.

The actual path loss values are calculated for each chunk separately in `simulation.ChunkSimulation.calculatePathloss`. This function is called in the initialization phase of each chunk, before the loop over TSs. The object `macroscopicPathlossModel.PathlossManagement.getPathloss` is called, which first calculates the current LOS/NLOS links, indoor/outdoor users. Based on the user parameters for losDecision and indoorDecision, the calculation can either be based on the geometry, static or based on a random decision. Additional it is possible to specify the model option for losDecision, where a specified path loss model defines the LOS/NLOS conditions. After determining those parameters, the appropriate model is used based on the models and parameters specified in `parameters.PathlossModelContainer.m`. The path loss is calculated without wall loss and antenna gain because those are already calculated in the functions `calulateAntennaGain` and `calculateWallLoss` of the `chunkSimulation`. An example of this can be found in the scenario file for the predefined scenario described in Section 4.4.

7.4.2 Modeling of Shadow Fading

Here should be added:

- explanations for wall loss calculations
- explanations of the use of shadow fading vs wall loss and a warning that there is no ray tracing

The modeling of the Shadow Fading (SF) is based on Shadow Fading maps (SFMs). Those SFMs are arrays of so called Shadow Fading Values (SFVs) which are spatially correlated Random Variables (RVs). Such an array of RVs can be generated by means of the following steps:

1. Generate an array of uncorrelated Gaussian RVs

2. Apply the FFT to the array
3. Multiply the transformed array with the Power Spectral Density (PSD) of the intended spatial correlation
4. Apply the inverse FFT

The advantage of this method is, that correlated shadow fading values for positions with minimal granularity can be generated without the need to resolve the whole ROI with the same precision. A more precise explanation on this topic can be found in [5].

7.4.3 Modeling of Small Scale Fading - Channel Models

PDP Channel Models: The used channel models are defined through a PDP. Descriptions of the available channel models can be found in the enumeration file `parameters.setting.ChannelModel`. This enumeration file also refers to the standards, in which the models are defined [6–10].

QuaDRiGa Channel Model: The QuaDRiGa channel model [11] is accessible through the Quadriga Interface provided as part of the `SmallScaleFading` package. The interface is contained in the `QuadrigaContainer` file. A detailed description of this channel model can be found in the QuaDRiGa documentation [12].

Before using QuaDRiGa for the first time, its source code has to be downloaded from the developer's website `https://quadriga-channel-model.de` and the `quadriga_src` folder has to be copied to the simulator's main directory. Currently, the simulator only supports QuaDRiGa v.2.4.0.

Please keep in mind that in order to use QuaDRiGa, you have to respect its license agreement (which is included in the download).

The QuaDRiGa channel model can be activated by setting `parameters.setting.ChannelModel.Quadriga` as the channel model in the scenario file. The interface passes all relevant simulation parameters to QuaDRiGa. Additional QuaDRiGa-specific parameters can be set in the file `parameters.channelModel.QuadrigaParameters`.

Trace Generation: For the PDP channel models, channel traces are generated before the main simulation loop. The generation of these channel traces is handled by the `PDPcontainer` in the `smallScaleFading` package. There, all possible combinations of number of receive antennas, number of transmit antennas, carrier frequency and channel model that can occur during the simulation are evaluated and channel traces for these scenarios are saved

in the folder `dataFiles/channelTraces`. If necessary, a different folder can be specified in the `SmallScaleParameters` class.

To get the channel realizations for a time slot the function `PDPcontainer.updatePDPcontainer` loads the needed channel traces into memory and the function `PDPcontainer.getChannelForAllAntennas` selects a random part of the trace to get a channel realization for one time slot. The channel traces should be much longer than the total simulation time to assure that the samples taken from the trace are independent from each other, but the trace should be as short as possible, since they require memory. The length of the channel traces has to be set in the `SmallScaleParameters` class.

The function `smallScaleFading.example` shows how to use the PDP container to generate channel traces for the link quality model and what parameters need to be set.

Assumptions: In general the channel is assumed to be constant in time for a time slot and a single value is calculated for the channel in the time domain. In case more different values for a TS are needed, a short block fading scenario is possible. For this the `SmallScaleParameters` class `symbolTimes` has to specify the time indices of the resource elements for which a channel realization should be calculated.

In the frequency domain the channel is assumed constant for a resource block. Values for all subcarriers are calculated but only a fraction of them is saved in the channel traces. In the `SmallScaleParameters` class this fraction can be specified, otherwise one value is saved for each resource block in frequency.

When the option correlated fading is chosen, the channel model is created according to [13] and a user speed has to be set to generate a channel trace. This user speed is set to the fastest user speed and used to calculate the Doppler frequency for the channel.

7.5 Cell Association and Wideband SINR

The cell association is performed according to the received signal quality, either according to the macroscopic receive power, or the wideband SINR. In both cases the cell with the highest considered signal quality metric is chosen as the serving (or desired) cell for each user.

Since the cell association only depends on macroscopic fading parameters, the cell association table is calculated for each segment and handovers from one base station to another are only performed at the beginning of a segment.

7.5.1 Wideband SINR

In the course of the cell association, the wideband SINR is calculated. The wideband SINR only depends on macroscopic fading parameters and is thus calculated per segment. The calculation considers all macroscopic fading parameters, which are the transmit power, antenna gain, path loss, wall loss, and shadowing and the wideband SINR is defined as

$$\mathrm{SINR}_{wideband} = \frac{P_{\mathrm{rx},d}}{\sum_{i=1}^{N_{\mathrm{int}}} P_{\mathrm{rx},i} + N}. \tag{7.3}$$

Here the same terms as in (7.1) are used, the indices d and i represent the desired and interfering cells and N represents the noise power experienced by the user aggregated over the whole transmission bandwidth. P_{rx} is defined in more detail in (7.2).

7.6 Link Quality and Link Performance Model

7.6.1 Link Abstraction

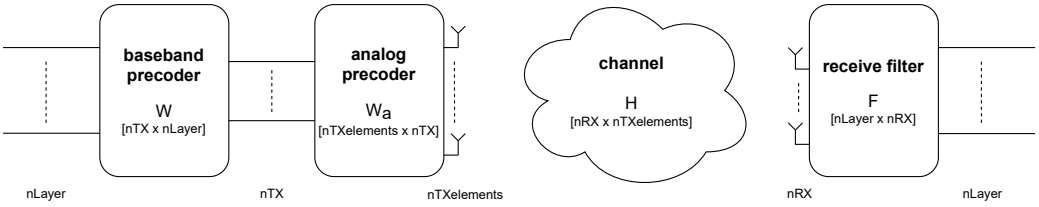


Figure 15: Abstraction of a link between a BS and a user. Each step is represented by a matrix in the simulation. The dimension of the matrix is indicated in squared brackets. `nLayer` represents the number of layers, `nTX` the number of transmit RF chains, `nTXelements` the number of antenna elements at the transmitter and `nRX` the number of receive antennas.

While the modulation and coding schemes are abstracted in the Link Performance Model (LPM) through the use of Block Error Ratio (BLER) curves for SINR to BLER mapping, the precoder, channel and receive filter are represented by matrices in the Link Quality Model (LQM). Details about the precoders are discussed in Section 7.8, details about the channel in Section 7.4.3. The receive filter is assumed to be a Zero Forcing (ZF) filter.

Figure 15 shows the processing chain modeled in the LQM. Each step in the processing chain is represented by a matrix of the dimension indicated in squared brackets in Fig. 15.

Terminology In the following, the terms used in Fig. 15 are related to commonly used terms and terms used in 3rd Generation Partnership Project (3GPP) standards. The layers that are input for the baseband precoder are also often referred to as streams or user streams. The `nTX` transmit RF chains are referred to as Transceiver Unit (TXRU) in 3GPP standards related to analog precoding, where analog precoding is referred to as TXRU virtualization. The concept of antenna ports and antenna port mapping is from LTE-A standardization is not implemented in the Vienna 5G SL Simulator - equivalently it can be seen as a one to one mapping - and the number of transmit RF chains `nTX` is always equal to the number of antenna ports `nAtPort`.

The number of transmit antenna elements `nTXelements` refers to the number of antenna elements, i.e. the antennas represented by crosses in Figs. 20 and 21. On the receiver side, `nRX` represents the number of receive antennas.

7.6.2 Link Quality Model

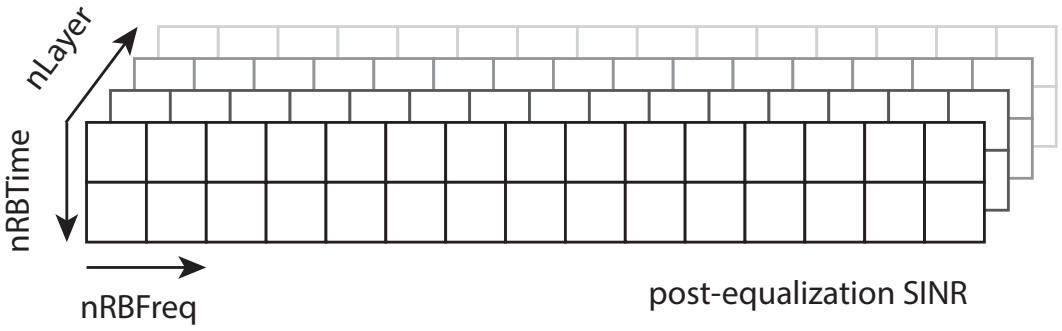


Figure 16: Output of the LQM for one time slot: the post-equalization SINR is calculated for each resource block and each layer for each slot.

The Link Quality Model (LQM) quantifies the quality of a link [14] in an SINR measure: the post equalization SINR, that is calculated for each layer of each resource block. The dimensions of the output of the LQM are depicted in Fig. 16.

The post equalization SINR takes into account the utilized receive filter and precoders and the inter layer interference, as well as the transmit power allocation.

To calculate these SINR values, the LQM needs constant parameters, like the receiver noise figure, that are set in the class constructor, large scale parameters, like path loss and antenna gain, that have to be updated

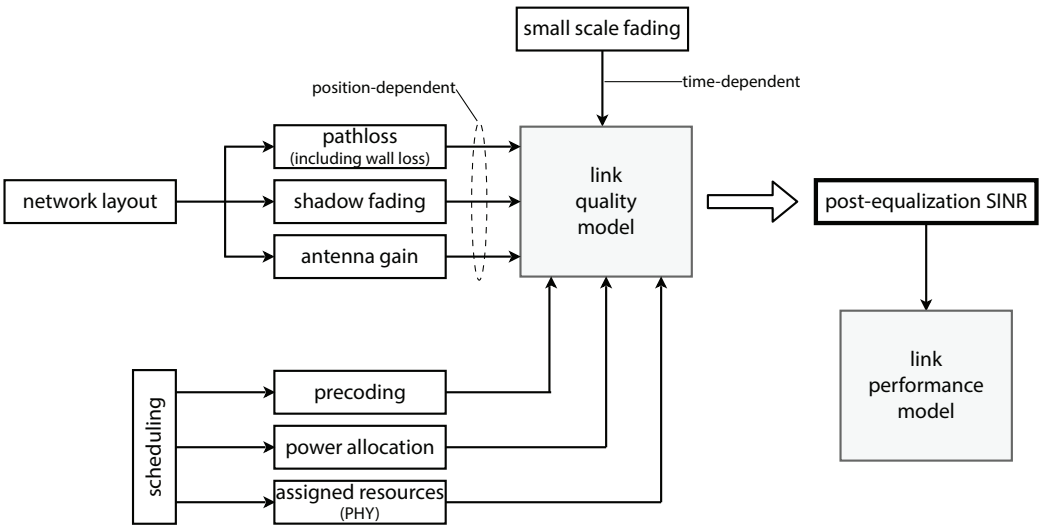


Figure 17: Schematic of the LQM.

for each segment with the function `updateMacroscopic` and small scale channel parameters that change for each time slot, like the channel matrix and transmit power allocation, that are updated in `updateSmallScale`. The example function shows how to use the LQM package. The general (simplified) functioning of the LQM is depicted in Fig. 17.

The implementation assumes that a Zero Forcing receive filter is used and that the small scale fading is constant for the duration of a time slot. It is also assumed that if no user is scheduled at an interfering BS, that this BS does not generate any interference, unless the `alwaysOn` feature is activated at the BS antenna. Further assumptions are that one receiver operates on a constant number of layers within a time slot and that the allocated transmit power is evenly distributed between those layers.

7.6.3 Link Performance Model

The Link Performance Model (LPM) takes the SINR values that were calculated by the LQM (see Fig. 17), the decision of the scheduler and the Channel Quality Indicator (CQI) of the feedback (see Section 7.7) to determine the throughput of a given user in terms of the number of bits. This is done in three steps as it is depicted in Fig. 18. The first step maps the time-frequency selective post-equalization SINR of a given user and transmission layer to an average SINR with equivalent Additive White Gaussian Noise (AWGN) performance for a modulation scheme selected by the feedback. This is based on a method called Mutual Information Effective Signal to Interference and Noise

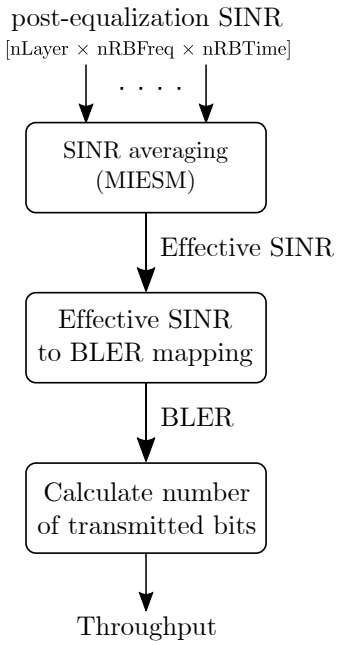


Figure 18: Schematic of the LPM.

Ratio Mapping (MIESM) described in [15] which uses calibrated capacity curves for the averaging process. The output is termed effective SINR. In the next step the effective SINR values is mapped to a BLER value. Each CQI value corresponds to a specific modulation scheme for which the BLER over SNR for a AWGN channel is saved in the simulator. Therefore the BLER is determined by a simple table-lookup. The last step is to calculate the number of transmitted bits. This is based on a Bernoulli experiment that determines if the current transmission was successful or not. In the case of a successful transmission the number of transmitted bits is set to the maximum size of the Transport Block (TB) for the scheduled Resource Blocks (RBs) and for a transmission failure it is set to zero. In the simulator it is possible to select if the LPM should perform the Bernoulli experiment in every slot or if it should give the expected value of the experiment as output.

On top of the actual throughput which is determined according to the CQI of the feedback the LPM calculates an upper bound on the throughput. This upper bound is the throughput that would result if the scheduler decides for the CQI that would result in the highest throughput. To get this value the LPM performs the above mentioned steps for all possible CQI values and takes the maximum over the resulting throughput.

An example for the usage of the Link Performance Model is given in the package `+linkPerformanceModel`. The example function calculates the

effective SINR for a random set of varying SINR values and determines the corresponding throughput based on the chosen type of CQI values. It also plots the capacity and AWGN curves that are used for the MIESM and BLER mapping as mentioned above.

7.7 Feedback

The `Feedback` class is used to provide the scheduler with information on channel conditions such as the Rank Indicator (RI), the Channel Quality Indicator (CQI) and the Precoding Matrix Indicator (PMI). Also, the feedback delivers ACKnowledged (ACK) or Non-ACKnowledged (NACK) infromation about user transmissions to the scheduler in order to hand in this to HARQ to decide whether retransmissions are needed or not. The scheduler needs this to determine optimal transmission parameters for future TSs.

All supported feedback types are implemented as subclasses of the `Feedback` class. Currently, these include

- `LTEDLFeedback`: This feedback type corresponds to the one used in the LTE Downlink. It is implemented according to [16].
- `MinimumFeedback`: This feedback type shows the minimum amount of feedback required by the scheduler.

Each feedback type implements the method `calculateFeedback`. All supported SINR to CQI mappings are implemented in subclasses of `parameters.transmissionParameters.CqiParameters`. They include

- `LteCqiParametersTS36213NonBLCEUE1`
  (based on QPSK, 16QAM, 64QAM)
- `LteCqiParametersTS36213NonBLCEUE2`
  (based on QPSK, 16QAM, 64QAM, 256QAM)
- `LteCqiParametersTS36213NonBLCEUE4`
  (based on QPSK, 16QAM, 64QAM, 256QAM and 1024QAM)

which are defined in 3GPP TS 36.213 [17]. The set of CQI parameters which is used for the simulation can be chosen with `parameters.setting.CqiParameterType` by setting the according transmission parameter. For example

```
1 % set CQI parameter type
2 params.transmissionParameters.DL.cqiParameterType        =
    parameters.setting.CqiParameterType.Cqi256QAM;
```

chooses a CQI reporting based on QPSK, 16QAM, 64QAM and 256QAM. It is also possible to turn off the feedback and only use the CQI values of the chosen set which result in the maximum possible throughput. This can be done by setting the variable `params.useFeedback` in scenario file to zero:

```
% use feedback
params.useFeedback = false;
```

7.8 Precoding

The precoders perform a mapping from the different layers to RF chains and from RF chains to antenna elements, as is depicted in Fig. 15. The precoders are collected in the `+precoders` package and used in the LQM, the feedback, and the scheduler. In the LQM, the choice of the precoder influences the calculated received powers for desired and interfering links and thus the calculated SINR. In the LTE feedback, each baseband precoder codebook is considered to calculate the expected SINR and then the feedback sets the PMI to maximize the SINR. In the scheduler this PMI is finally used to set a baseband precoder individually for each resource block.

7.8.1 Baseband Precoding

The baseband precoder is chosen individually for each resource block and maps the different layers to transmit RF chains. The layers are also often referred to as user streams. Baseband precoding is also referred to as digital precoding. The baseband precoder can utilize feedback values to choose a precoding matrix from a codebook. The documentation of each precoder refers to the feedback types that set the necessary feedback values. Each base station can have its own downlink and uplink precoder assigned. Some important implemented precoders compatible with the LTE downlink feedback are listed below. For more details please refer to the code documentation in `+parameters.precoders` and in the example file of the `+precoders` package. For base stations that have multiple antennas with different technologies consider using the `+parameters.precoders.Technology` precoder.

- `PrecoderLTEDL`: LTE precoder according to [2] for up to 4 transmit chains and four layers.
- `Precoder5G`: 5G precoder according to [18] for single panel antenna arrays with `codebookMode=1`. This precoder is suited for digital precoding without an analog precoder. Supports up to 6 transmit chains and 4 layers. Please take a look at the code documentation for the supported

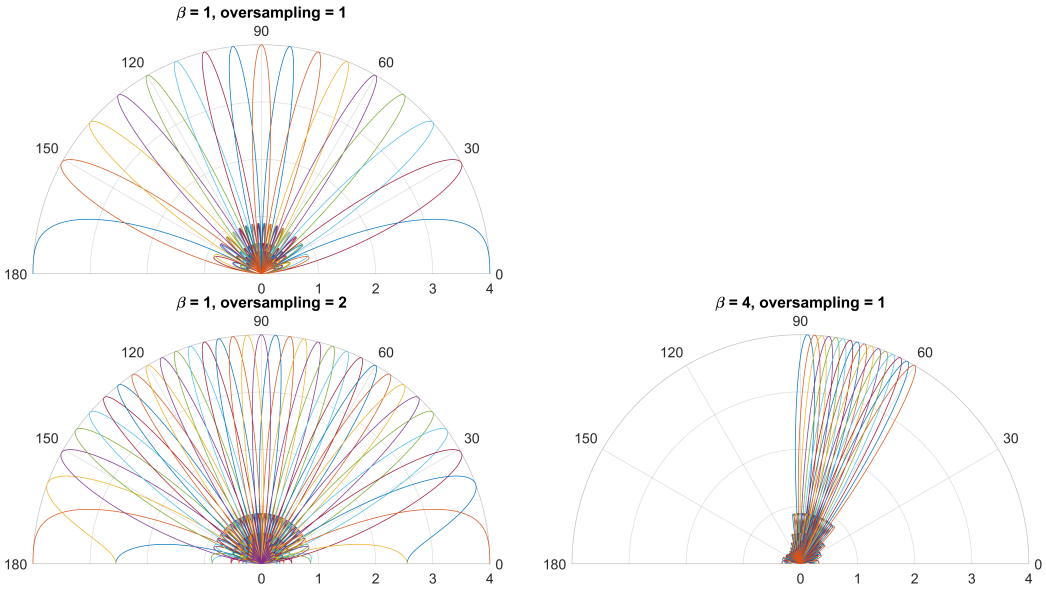


Figure 19: 2D Kronecker precoder beams with different oversampling factor and downtilt factor β for digital precoding.

antenna configurations. The standard specifies the precoders for dual polarized antenna arrays. Since the simulator uses single polarized antennas some modifications were made to fit them in the simulator. The modifications are documented in the code. For a more flexible and configurable precoder take a look at the Kronecker product based precoder.

- `PrecoderKronecker`: Kronecker product based precoder according to [19]. This precoder is designed for uniform planar arrays. Like the 5G precoder it is suited for digital precoding without an analog precoder. The codewords are created by taking the Kronecker product of two DFT codewords. It is based on the same principles as the 5G precoder but allows for greater flexibility since it is possible to adjust the oversampling factors of the DFT codewords and thus the codebooks size. Figure 19 shows the beams resulting from the precoders of the codebook in the 2D case. Increasing the oversampling factor increases the number of beams (and precoders) . Also they begin to overlap. By increasing the downtilt factor β the beams get restricted to a smaller area. For more details please refer to paper [19].

- `Technology`: This precoder is aware of the antenna technology. It can be configured to use different precoders for different technologies. For

example: an omnidirectional antenna for LTE and an antenna array for 5G can be connected to the same base station. The technology precoder can then be configured so that the base station uses the LTE precoder for the LTE antenna and the 5G precoder for the 5G antenna. For more details please refer to the technology and numerology section 7.12.

7.8.2 Analog Precoding

The analog precoder maps the RF chains to antenna elements. This allows to perform beamforming by adding a phase shift for each antenna element. In case no beamforming is performed, the analog precoder uses a one to one mapping from the RF chains to the antenna elements.

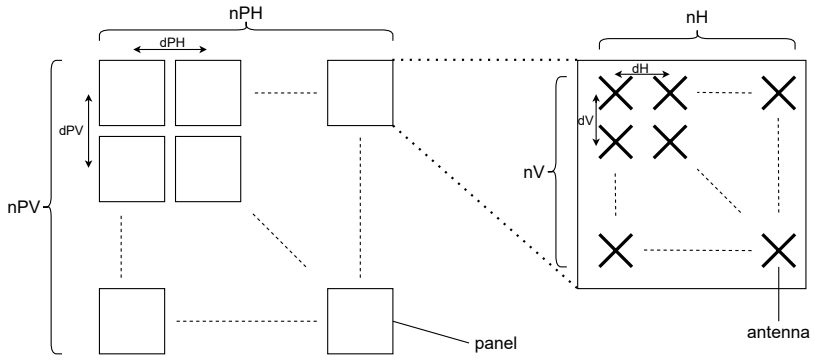


Figure 20: Antenna array consisting of panels with antenna elements placed on them.

MIMO Analog Precoder The MIMO analog precoder performs beamforming for antenna arrays. The antenna array consists of several panels with antenna elements placed on them as depicted in Fig. 20. The RF chains are mapped to a column of antenna elements as specified in [20]. This mapping is repeated for antenna columns as depicted in Fig. 21. The mapping is repeated on each panel.

7.9 Scheduling

The scheduling is performed by the schedulers located in the `+scheduler` package. The scheduling allocates physical resources available at a base station to the users associated to this base station. Physical resources here refers to the resource grid in time and frequency, that is divided in resource blocks that combine several Orthogonal Frequency Division Multiplexing (OFDM)

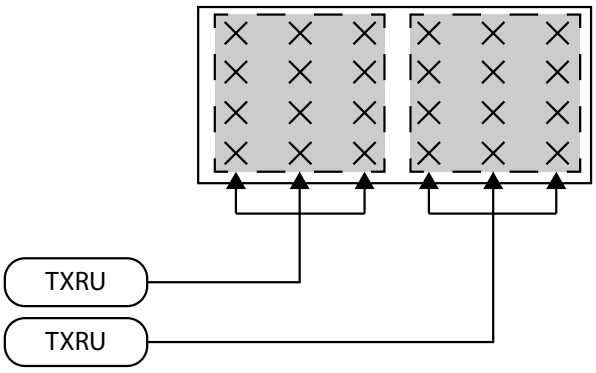


Figure 21: Mapping of RF chains to antenna columns. The mapping of the first column is repeated to the following columns if there are more antenna columns than RF chains.

symbols. For each resource block the scheduler decides which user is served on these resources and then sets the transmission characteristic accordingly. The transmission characteristic include the used precoder, transmit power, CQI, and number of codewords transmitted. This information is collected in the scheduler signaling, which can be found in the `+scheduler.signaling` package. Since the scheduling distributes physical resources that are attributed to a base station, the scheduling is performed individually per base station, that can here be seen as a cell with physical resources, i.e., a spectrum to use for data transmission to serve the users in the cell.

Two scheduling methods are implemented: the round robin scheduler and the best CQI scheduler. The round robin scheduler schedules the users one after the other in a row. When all users have been assigned resources, the first user is scheduled again and so on. In case of the best CQI scheduler, the user with the highest CQI value calculated bay the feedback is allocated to each resource block.

Additionally, the round robin scheduler can be expanded by a scheduling weight that can be set for each user group. The scheduling weight indicates the number of resource blocks assigned to a user when scheduled by the round robin scheduler. For example, a user with scheduling weight three, would be assigned three consecutive resource blocks and then the round robin scheduler would move on to the next user.

Other features that have impacts on the scheduling are spectrum sharing as described in Section 7.12.2, NOMA, see Section 7.13, and HARQ, as explained below in Section 7.11.

7.10 Traffic Models

Since 5G networks are heterogeneous with arbitrary user types, it is necessary to account for their traffic models in the simulation. The basic operation of the traffic models is defined within the last three parts in the simulator structure shown in Fig. 6. After the initialization part finishes, the setup of the traffic model is done for every user individually. During the main simulation loop, the necessity of new packet generation for each user is checked according to the model statistics and the packets buffer for each user is updated according to its throughput. After the post processing, ecdf of packets transmission latency is plotted.

The user data packets are discriminated with the following properties: generation time, successful transmission time, packet size, remaining of the packet size after transmission and identity number. Further details about the data packets can be found in the code documentation in `+trafficModels.DataPacket`.

The interaction between the simulator and the traffic model is built on top of `+trafficModels.PacketProcessing` class. It is a super class for all the traffic models and where packet generation and transmission functionalities typically reside. Fig. 22 shows the connection between any traffic model and other parts of the simulator. As illustrated, the schedulers in the `+scheduler` package inquire the traffic models input since the scheduling decision is being influenced accordingly. Hence, active users that have packets in the packets buffer are scheduled. The functions which are defined in the super class and have to be called by every model are listed below:

- `checkNewPacket`: It has to be called once for every simulation slot to check if a new data packet has to be generated, if necessary, a new data packet is being appended to the packets buffer.
- `get.isActive`: It checks whether a traffic model is active or not. The model is active if there is at least one bit in the packets buffer.
- `getBufferState`: It checks the number of packets in the buffer, their generation times and the number of remaining bits in each packet. This function has to be called by every scheduler since the scheduling decision is being affected by the packets buffer state.
- `updateAfterTransmit`: It checks if there are packets in the buffer and update the number of remaining data bits and the successful transmission time for these packets after each transmission.

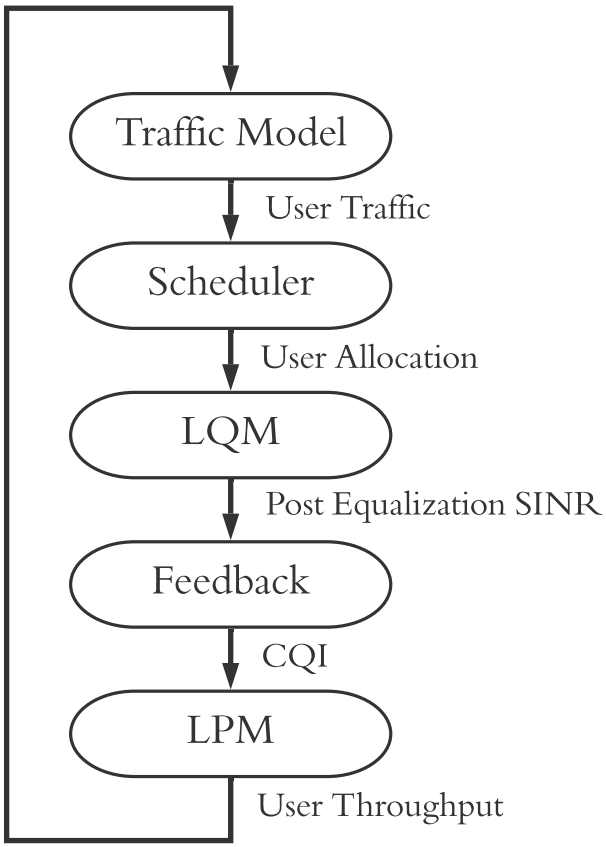


Figure 22: A block diagram illustrates the traffic model interaction with other parts of the simulator.

- `getTransmissionLatency`: It calculates the packets transmission latency.
- `clearBuffer`: It re-initializes the traffic model and has to be called at the end of each chunk simulation.

All necessary parameters to set up the traffic models can be found in this package `+parameters.user.trafficModel`. The available traffic models are implemented as sub classes of the super class `PacketProcessing`. The current models are listed below:

- *Constant Rate*: This model generates fixed size packets governed by a certain packet arrival rate as depicted in Fig. 23. The initial time for

generating the packets of different users is random to obtain realistic simulations.

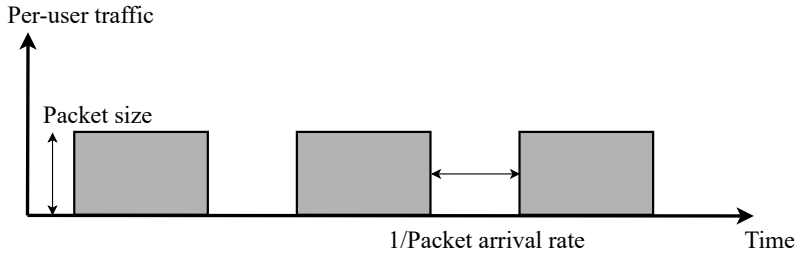


Figure 23: Constant Rate traffic model.

- *Full Buffer*: Users have infinite amount of data to be transmitted according to [21, Annex.2.1.3]. This model is implemented such that each user have a single packet of infinite size as depicted in Fig. 24. For this model no ecdf of the transmission latency is plotted since all packets are not fully transmitted at the end of the simulation.

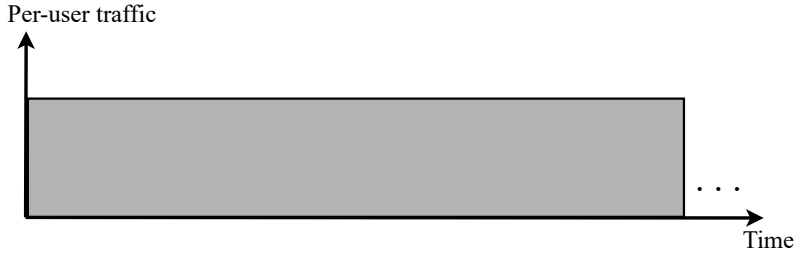


Figure 24: Full Buffer traffic model.

- *File Transfer Protocol*: An File Transfer Protocol (FTP) session is a sequence of file transfers separated by reading times according to [22, Appendix B]. The two main FTP session parameters are the size of a file to be transferred and the reading time, i.e., the time interval between end of download of previous file and the user request for the next file as shown in Fig. 25.

- *Hypertext Transfer Protocol*: Hypertext Transfer Protocol (HTTP) traffic characteristics are governed by the structure of the web-pages on the World Wide Web (WWW) and the nature of human interaction. The human interaction with the WWW causes the HTTP traffic to have a bursty profile. A web-page is usually composed of a main object and

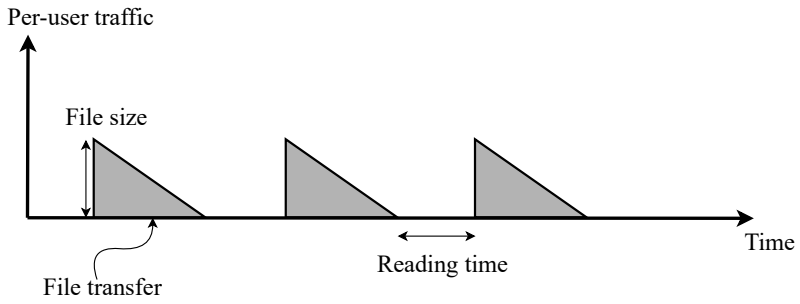


Figure 25: File Transfer Protocol traffic model.

several embedded objects. Hence, the main parameters to characterize web-browsing are: the size of a main object, the size of an embedded object, the number of embedded objects, reading time and parsing time for the main object as shown in Fig. 26. The reading time represents the time the user spends reading the web-page before transitioning to another page. All of these parameters are random values that follow statistical distributions according to [22, Appendix B].

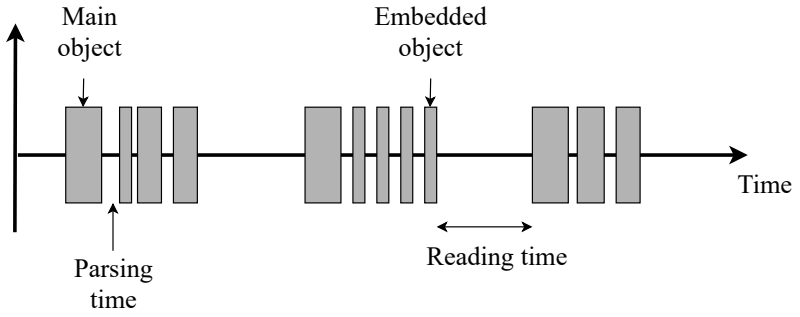


Figure 26: Hypertext Transfer Protocol traffic model.

- *Video Streaming*: Each frame of video data arrives at a regular time interval T. Each frame consists of a number of slices or packets which have random sizes. The video encoder introduces inter-arrival times between the packets of a frame known as encoding delay intervals. Therefore, the parameters of a video streaming session are: the inter-arrival time between the beginning of each frame, the number of packets in a frame, the packet size and the inter-arrival time between the packets in a frame as shown in Fig. 27. The latter two parameters follow statistical distributions according to [22, Appendix B]. The

distributions assume a source video rate of 64 kbps.

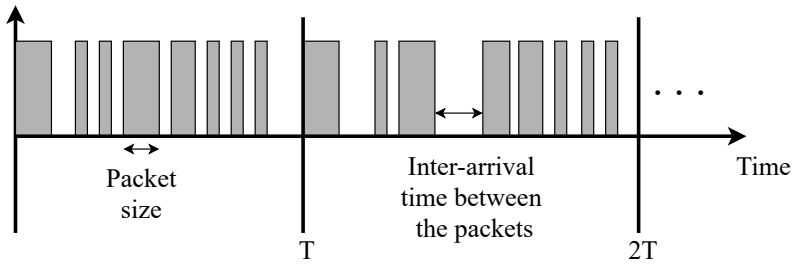


Figure 27: Video Streaming traffic model.

- *Gaming*: This traffic is generated by users engaged in interactive gaming of multiple users in different locations via the Internet. The starting time of a network gaming mobile is uniformly distributed to simulate the random timing relationship between client traffic packet arrival and uplink frame boundary. The following parameters also govern gaming traffic: the packet inter-arrival time, the packet size and deterministic size to be added to the packet size accounting for the User Datagram Protocol (UDP) header after header compression. These parameters are depicted in Fig. 28. The packet inter-arrival time and the packet size are random according to [22, Appendix B]. Gaming packets are relatively small in size and due to the interactive nature of gaming, packet delay must be short. In addition, a mobile network gaming user is in outage if the average packet delay is greater than 60 ms.

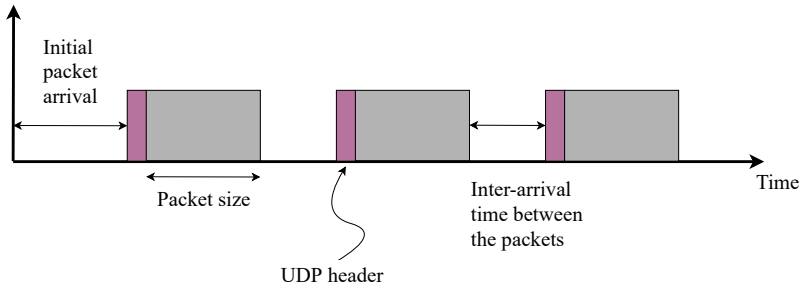


Figure 28: Gaming traffic model.

- *VoIP*: The voice activity model of Voice over IP (VoIP) is represented with simple two-state Markov model as shown in Fig. 29. According to the voice activity factor that is used in VoIP model, the probability

of transitioning from the inactive state to the active one is equal to c, while the probability of remaining in the active state is d. Adaptive Multi-rate (AMR) is the audio codec used in this model. It is an audio data compression scheme optimized for speech coding. The AMR speech codec consists of multi-rate speech codec with eight source rates range from 4.75 kbps to 12.2 kbps. The source rate used in VoIP model is 12.2 kbps. During voice activity periods, there is one VoIP packet (voice payload) generated every 20 ms. This inter-arrival time between VoIP packets is known as encoder frame length. Similarly, Silence Insertion Descriptor (SID) payload is generated every 160 ms during silence periods. There are some other relevant parameters for this traffic model such as: voice payload and SID payload. General functionality of the AMR codec is illustrated in Fig. 30. All parameters values are stated in [22, Appendix A]. A VoIP user is in outage if 98% radio interface tail latency of this user is greater than 50 ms. This assumes an end-to-end delay below 200 ms for mobile-to-mobile communications.

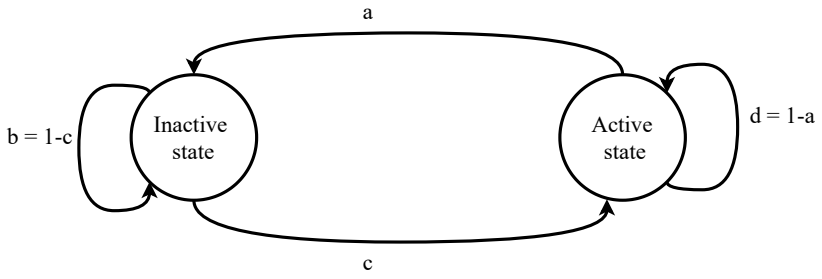


Figure 29: Two-state voice activity model.

Descriptions of the models can be found in the enumeration file `+parameters.setting.TrafficModelType`.

7.11 HARQ

HARQ is a combination of Forward Error Correction (FEC) and Automatic Repeat Request (ARQ) which is used to enhance the SNR and throughput performance by reducing the transmission errors. The operation of HARQ is illustrated in Fig. 31. Initially, the user decodes the packets and sends ACK for successful decoding and NACK for transmission failures on a codeword basis. This information is sent using the user feedback to the BS. For successful transmissions, the scheduler initiates a new transmission. For transmission

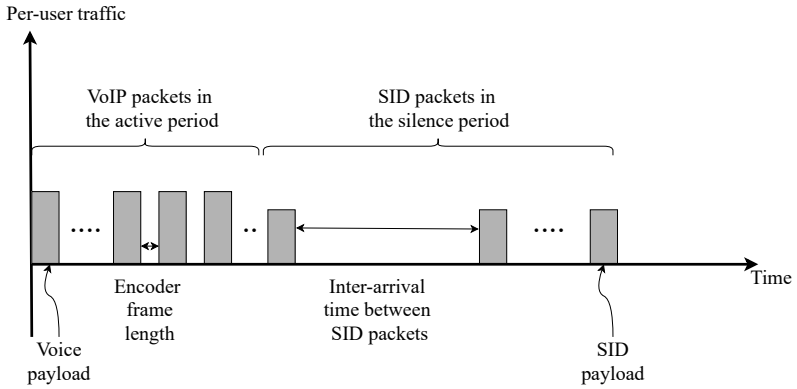


Figure 30: VoIP traffic model.

failures, the scheduler triggers retransmissions until the user decodes its packets correctly or the configured maximum number of retransmissions is reached. Up to three retransmisions are possible. Consequently, HARQ changes the redundancy version and hands it in to the LPM. The LPM uses the redundancy version for selecting the appropriate SNR-BLER mapping for the throughput calculations. According to the throughput results, the acknowledgment information is determined.
The redundancy version is the set of coded bits used in the transmission, in other words, it is the amount of redundancy added into the codeword while turbo encoding. The redundancy version parameter can be found in the class `parameters.transmissionParameters.TransmissionParameters` and has four values corresponding to the transmissions trials.
In LTE and 5G, adaptive asynchronous HARQ is used for the Downlink (DL) whereas non-adaptive synchronous HARQ is used in the SLS. Adaptive HARQ means that the number of coded bits transmitted could change from one retransmission to the next and these bits could be transmitted using different Modulation and Coding Scheme (MCS) and different number of resource blocks. On the other hand, non-adaptive HARQ imposes the same number of bits and resource blocks to be used in retransmissions. In synchronous HARQ, the retransmissions occur at a predefined time relative to the initial transmission. Whereas, in the asynchronous scheme, a retransmission can occur at any time relative to the initial transmission. Unlike the standard [23], HARQ is not performed with multiple parallel processes. These processes are transmission opportunities for the codewords such that a codeword process can start without waiting for ACK or NACK feedback for the previous one. Until now, only classic schedulers, namely Round Robin (RR), Best CQI

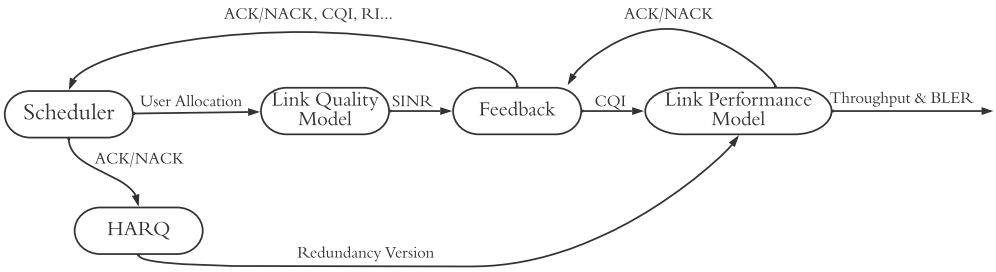


Figure 31: The schematic diagram illustrates HARQ operation with other parts of the simulator.

and Static Spectrum schedulers, include HARQ functionalities. Every classic scheduler has to call the following functions that are explained in Section 7.9 and reside in `+scheduler.Scheduler` in every simulation slot:

- `updateHARQMaskDL`
- `resetHARQMaskDL`

Retransmissions for users are signaled from the `+scheduler.signaling.HARQ` class. All the functions of this class that have to be called upon user feedback are listed below:

- `initiateRetransmission`: It has to be called from within the `updateHARQMaskDL`. It increases the redundancy version in case of a retransmission and resets it if the maximum number of retransmissions is reached or ACK is received as user feedback. The same resource blocks and CQI used in the original transmission have to be used for retranmissions. These information are saved for each user in `+scheduler.signaling.UserScheduling`.
- `resetRV`: It resets the redundancy version of one codeword and is being called from within `initiateRetransmission` function.
- `resetRVs`: It reinitialized the redundancy version of the two codewords. This function has to be called at the end of chunk simulation.
- `toStruct`: It writes HARQ signaling into a struct.

As stated in Section 7.6.3, SNR-BLER mapping for each CQI and redundancy version is saved into look-up table fashion in the simulator. These mappings are obtained from the LTE LL simulator. Using these different mappings, one can observe HARQ gain which includes SNR gain and throughput

reduction for every number of retransmissions. For HARQ results validation, a single user-BS scenario has been used with fixed CQI scheduling to avoid the user feedback impact. SNR sweeping is done achieving the desired number of retransmissions while the error free transmission case served as a baseline. Simulation parameters are summarized in Table 3.

Parameter	Value
base stations	1 BS
users	1, predefined position
pathloss model	Free space
channel model	AWGN
slots	100
feedback delay	1
user speed	0 km/h
traffic model	full buffer

Table 3: Simulation parameters used for HARQ validation in the 5G SLS using mappings from LTE LL simulator.

The throughput reduces by a factor of two when one retransmission is required for achieving error free transmission. Reduction by a factor of three and four correpsond to two and three retransmissions respectively as shown in Fig. 32. The effective SNR gain is illustrated in Fig. 33. It is worth mentioning that by deactivating the fading using the AWGN channel model, the effective SINR resulting from LPM and post-equalization SINR resulting from LQM are equal since the effective SINR is averaging of the post equalization SINR which is identical at all RBs.

7.12 Technologies and Numerologies

Users and antennas can have different technologies. The term technology refers to a more abstract concept of network elements, in which the direct interaction is only possible between elements of the same technology. Rather than the direct interaction the goal is to model the impact of shared resource constraints between these multiple technologies. To allow the analysis of more than one technology present in the simulator two additional elements are needed.

- Composite Base Station
- Spectrum Scheduler

These elements are further discussed in the following sections.

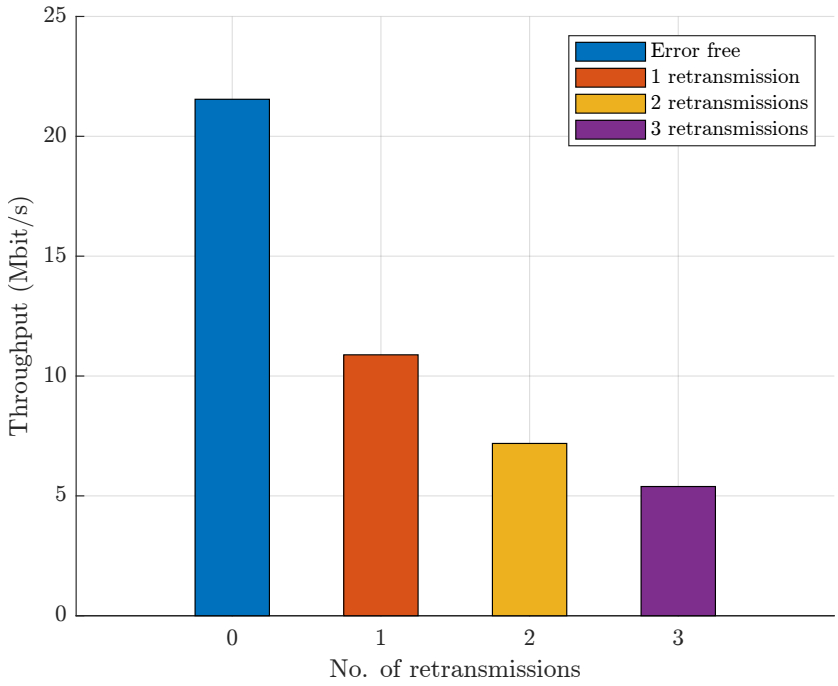


Figure 32: Throughput reduction due to HARQ retransmissions.

7.12.1 Composite Base Station

A composite base station is needed to organize several base station in a hierarchical, tree like structure. From an external point of view, it behaves as a single base station while internally, by allowing several base stations to work together, more advanced operations are possible.

One use case of such a composite base station is when simulating more than one technology. In that case the set up procedure will split the antennas with respect to their technologies into several base stations. These are then configured utilizing the same parameterization as for the parent base station. The property on which the splitting takes place can be changed by overwriting the `getNetworkElementSplitter` function in the `CompositeBasestation` class, so different types can be implemented.

7.12.2 Spectrum Scheduling

The spectrum scheduler is a special type of scheduler, which is always linked up to a composite base station, used for distributing resources between the several base stations inside the composite base station. This is accomplished according to some allocation rule, which is defined by the used spectrum scheduler type. The information of the spectrum scheduling is then forwarded

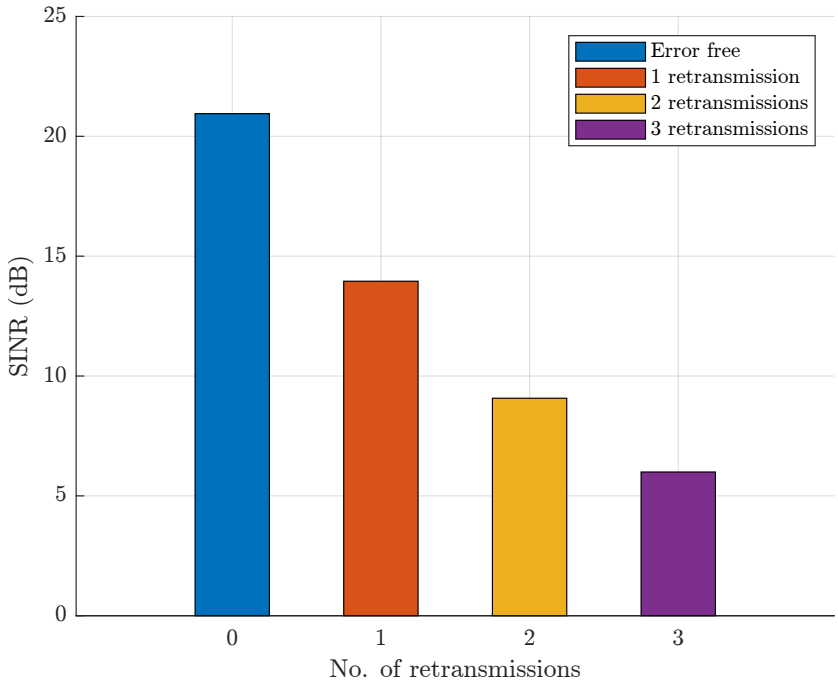


Figure 33: Effective SNR gain of HARQ.

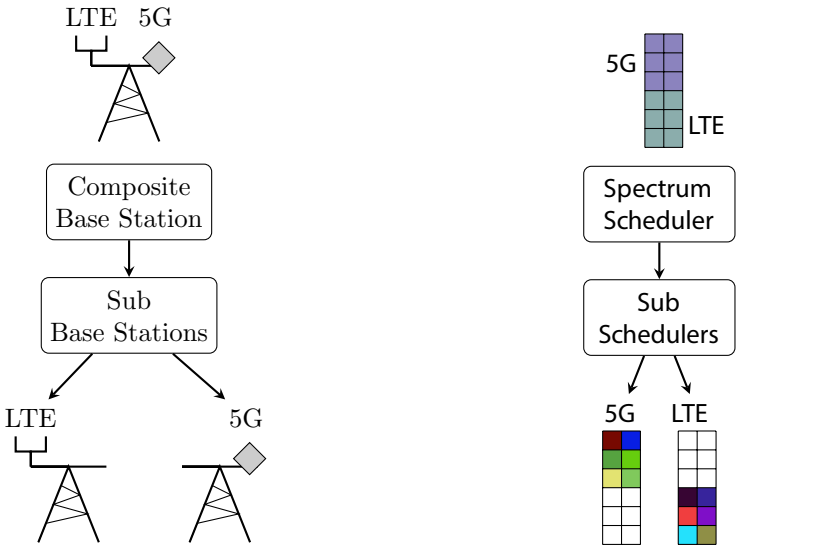
to the schedulers of the sub base stations building up the composite base stations.

Up until now three spectrum scheduler types are implemented:

- `static`
- `dynamicUser`
- `dynamicTraffic`

In each of them a different allocation rule is implemented. The `static` spectrum scheduler type allocates resources evenly between all available technologies. The `dynamicUser` spectrum scheduler type allocates based on the number of attached users in each technology and the `dynamicTraffic` utilizes the information of the traffic models by distributing resources based on the accumulated buffer size in each technology. Table 4 summarizes the previous described behavior.

Additional to the spectrum scheduler type an additional parameter can be used to manually bias the resource allocation for each technology. This parameter is specified in the spectrum scheduler and is called weighting. The default weight value of a technology is one. Utilizing the weighting parameter combined with a spectrum scheduler of type `static` enables therefore an



(a) Composite BS (b) Spectrum Scheduler

Figure 34: Mixed technology elements

Scheduler type	Technology	User count	Buffer size	Resource balance
dynamicTraffic	LTE	20	10 MBit	1/3
	5G	5	20 MBit	2/3
dynamicUser	LTE	20	10 MBit	4/5
	5G	5	20 MBit	1/5
static	LTE	20	10 MBit	1/2
	5G	5	20 MBit	1/2

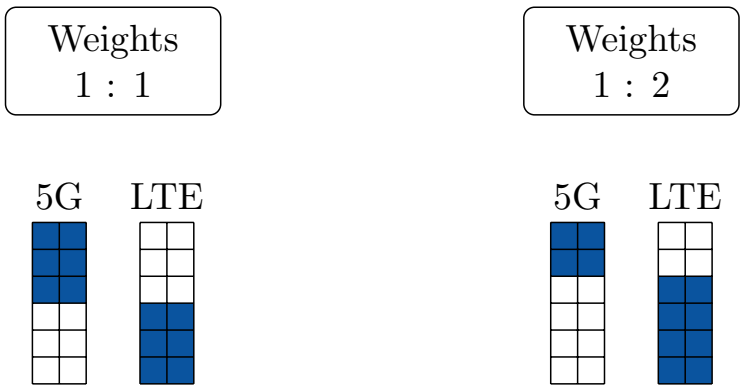
Table 4: Spectrum scheduling strategies

uneven resource distribution as can be seen in figure 35. Figure 35a displays the default resource distribution based on the static equal split without additional weighting. Figure 35b shows the biased resource allocation based on the additional weight for the LTE technology.

The concept of technologies allows to use an individual precoder for each technology. For this the `precoders.parameters.Technology` precoder type can be used (see also 7.8).

7.12.3 Numerology and mixed-numerology scenarios

The simulator supports mixed numerology simulations. A scenario can contain users and base stations of different numerologies. Meaning that the system operates with different subcarrier spacings. According to the 5G standard the



(a) Default (b) Biased

Figure 35: Weight dependent resource distribution.

system's subcarrier spacing is defined by the numerology parameter $\mu \in \mathbb{N}$. It is an integer number

$$\mu \in \{0, 1, 2, 3, 4, 5\}.$$

The subcarrier spacing f_Δ is connected to the numerology with the relation

$$f_\Delta = f_b 2^\mu,$$

where f_b is the base subcarrier spacing set to $f_b = 15\,\text{kHz}$ in 5G. With this the possible spacings are

$$f_\Delta \in \{15\,\text{kHz}, 30\,\text{kHz}, 60\,\text{kHz}, 120\,\text{kHz}, 240\,\text{kHz}, 480\,\text{kHz}\}.$$

Note that the parameter μ and the base subcarrier spacing f_b can also take on values not specified in the 5G standard. The base subcarrier spacing can be set in the resource grid parameters. Consult the built in documentation of `parameters.resourceGrid` for more information.

Network elements of different numerologies are always separated in different technologies. This ensures that users will never connect to base stations of other numerologies. The naming convention of the technology is:

```
[Technology]:[Numerology]
```

For example, consider a user of the technology LTE and the numerology 1. The resulting total technology label will be

```
LTE:1.
```

Therefore all features applying to the technology parameter can also be used together with numerologies. The numerology is set by defining the numerology parameter on each base station and user in the scenario file.

```
% define antenna with numerology 0
ant1 = parameters.basestation.antennas.Omnidirectional;
ant1.numerology = 0;
% define antenna with numerology 1
ant2 = parameters.basestation.antennas.Omnidirectional;
ant2.numerology = 1;

% define user with numerology 0
userNum1 = parameters.user.PredefinedPositions();
userNum1.numerology          = 0;
% define user with numerology 0
userNum1 = parameters.user.PredefinedPositions();
userNum1.numerology          = 1;
```

An example with more details is provided by the `numerologyScenario`.

7.12.4 Inter-numerology interference

The simulator considers interference created due to different numerologies in the LQM. The paper [24] provides equations for Inter Numerology Interference (INI) on a per subcarrier basis. It is assumed that the power in the resource blocks is distributed evenly on its carriers. The INI in an RB is calculated by summing up over the interference of its carriers. Figure 36 shows how the numerology can change in the RB due to spectrum sharing. Therefore it is necessary to calculate the INI in each slot, since it depends mainly on the distance between the resource blocks. The simulator scans in the setup phase for all numerologies and precaclulates the INI factors. During chunk simulation the LQM performs a table lookup and calculates the INI which is then treated as additional interference. The simulator also provides the option to omit inter-numerology interference. For that the following parameter option exists:

```
% disable ini calculation
params.calculateIni = false;
```

7.12.5 Resource Grid

The resource grid defines basic OFDM parameters like symbol duration, subcarrier spacing or the amount of resource elements per resource block. Some of these parameters are dependent on the numerology. Figure 37 sketches a resource grid shared by two systems with different numerologies. The size of the resource elements change in time and frequency. Since the LQM and other components depend on equally sized resource blocks, the

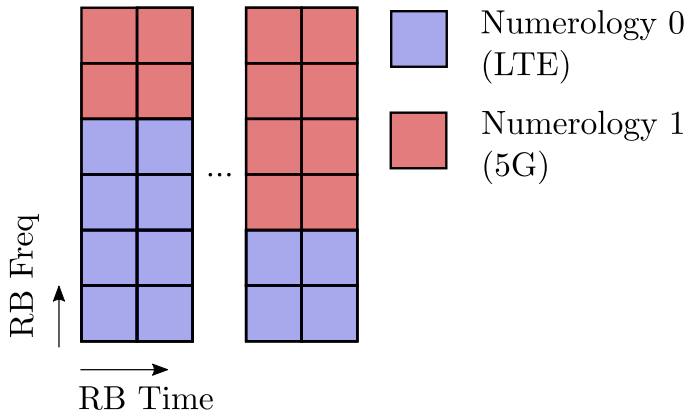


Figure 36: Changing numerologies in RB due to spectrum sharing

number of RE in time and frequency is adjusted so that all RBs have the same size. The base grid defines the size of the RB, and the resource grid for other numerologies are derived from the base grid, so that the RB size stays equal. The base grid defines this parameters for the smallest numerology.

```
% define base grid
params.transmissionParams.DL.resourceGridType = params.
    setting.ResourceGrid.LTE;
```

7.13 NOMA

7.13.1 A NOMA Transmission

The 5G VCCS SLS supports two user power domain NOMA transmission. To use NOMA transmissions in a simulation the NOMA parameters in `parameters.Noma` have to be set.

7.13.2 NOMA User Pairing

For a NOMA transmission to be efficient, the difference in channel quality between the near user and the far user needs to be large enough. The minimum difference for NOMA transmission to be used can be set in `parameters.Noma.deltaPairdB`. With this minimum difference Δ, the users are then paired starting with the strongest and weakest users in the cell. If the difference in their cell association metric is larger than Δ, they form a NOMA pair and will share their resources during the simulation. This procedure continues with the second strongest and second weakest user in the cell and goes on until a pair does not reach the threshold Δ, the remaining users will not share their

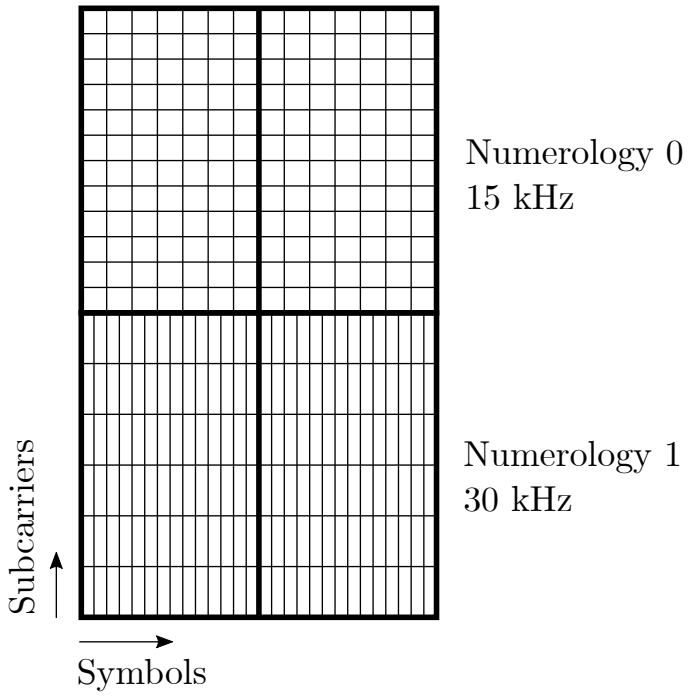


Figure 37: Resource Grid with two different numerologies. The thick lines indicate resource blocks.

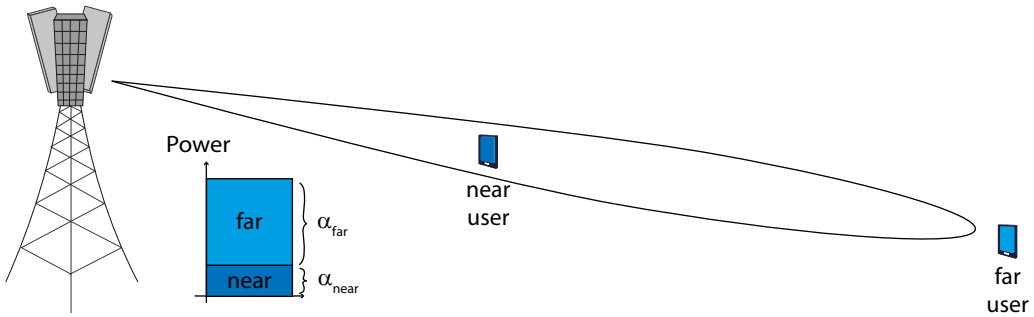


Figure 38: In a power domain NOMA transmissions two users share the same physical resource in the power domain. The near user performs Successive Interference Cancellation (SIC) to receive data.

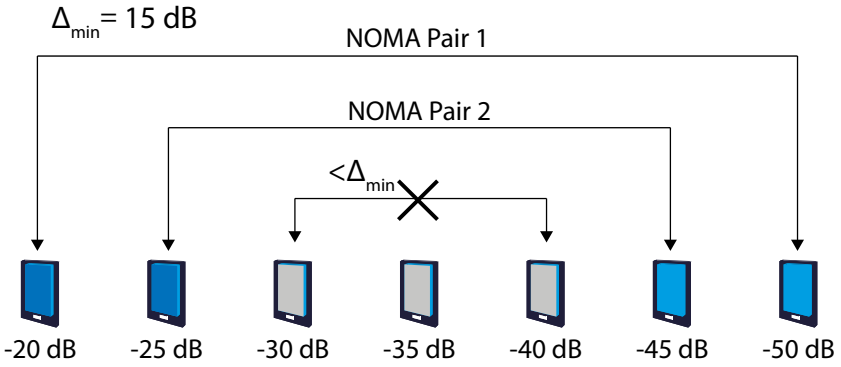


Figure 39: Users are paired for a NOMA transmission if the difference between their cell association metric is larger than `parameters.Noma.deltaPairdB`.

resources and will proceed with classical Orthogonal Multiple Access (OMA) transmissions. Figure 39 shows the user pairing strategy, NOMA near users are shown in dark blue, NOMA far users in light blue and OMA users in gray.

7.13.3 NOMA Scheduling

If two users are paired in the NOMA user pairing during cell association, then all resources assigned to the users in the NOMA pair are shared between the users and used for a NOMA transmission. Figure 40 shows an exemplary cell in which the users 1 and 3 form a NOMA pair and the users 7 and 5 are also paired for NOMA transmission. The remaining users 2, 4 and 6 perform classical OMA transmissions, their channel conditions are too similar for effective SIC transmissions. In the example in Fig. 40, a round robin scheduler, assigns all available resources to the users in the cell in a round

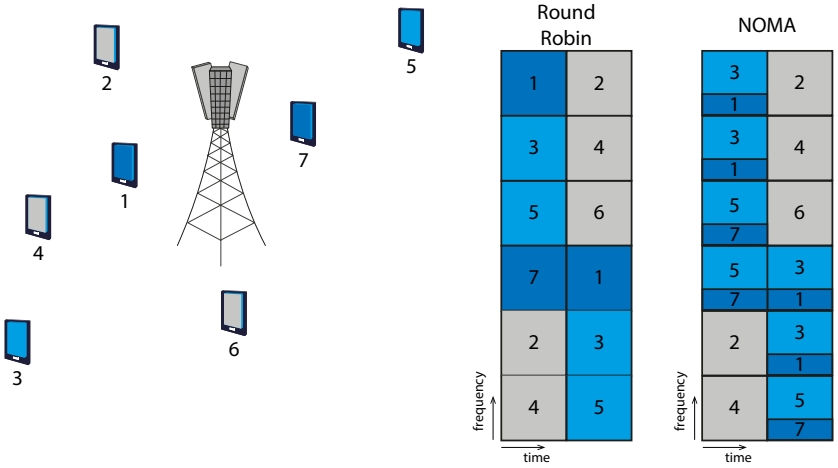


Figure 40: Resource sharing between NOMA users. The figure shows an exemplary cell with OMA users in gray, NOMA near users in dark blue and NOMA far users in light blue. The grid represents the RBs available for transmission. A round robin scheduler assigns an RB to each user in the cell. Afterwards the NOMA scheduling shares all resources assigned to a NOMA pair.

robin fashion. All resources assigned to a user in a NOMA pair are then shared and used for NOMA transmission for this NOMA user pair.

7.13.4 NOMA in the Link Quality Model

The SIC is modeled in the LQM. For more details about the LQM see Section 7.6.

SIC For the near NOMA user, SIC is performed. For this, the signal of the far NOMA user is received and decoded at the near user. The LQM calculates the post equalization SINR of the signal of the far user received at the near user $\gamma_{near}(far)$ as follows:

$$\gamma_{near}(far) = \frac{\alpha_{far} p_{near}(far)}{\alpha_{near} p_{near}(near) + I + n}, \tag{7.4}$$

where $p_{near}(far)$ is a very short notation for the power of the signal of the far user received at the near user. In the same manner $p_{near}(near)$ is a short notation of the power of the signal of the near user received at the near user - this is here considered as interference, since the signal of the far user is decoded in this step. I is a short notation for the interference from other cells

and n is the thermal noise experienced in the considered RB. α_{far} and α_{near} are the NOMA power share factors as depicted in Fig. 38.

This post equalization SINR $\gamma_{near}(far)$ is then handed over to the LPM and if the transmission of the signal of the far user is successful, the interference from the far user can be canceled out through the factor ϵ for the reception of the signal of the near user as follows:

$$\gamma_{near}(near) = \frac{\alpha_{near} p_{near}(near)}{\epsilon \alpha_{far} p_{near}(far) + I + n}. \tag{7.5}$$

If SIC cancels out the interference from the far user completely, then $\epsilon = 0$, otherwise ϵ can be set to a higher value up to 1.

If the signal of the far user can not be decoded successfully, then the transmission of the signal of the near user fails and the post equalization SINR is set to $-\infty$.

Additional Interference for the Far User For the far NOMA user, only additional interference is added to take into account the transmission of the near user. Since α_{near} is smaller than α_{far}, the additional interference is small. The post equalization SINR at the far user is calculated as follows:

$$\gamma_{far} = \frac{\alpha_{far} p_{far}(far)}{\alpha_{near} p_{far}(near) + I + n}. \tag{7.6}$$

The notation follows the notation used in the equations above.

7.14 Post Processing

After the main simulation loop is done, the results of the individual chunks are collected and combined to a final result. This is called the post processing phase and is performed by the so called postprocessor. The 5G SL simulator provides different types of postprocessors to be chosen from. The chosen postprocessor has two main impacts on the simulation. First it specifies if a *lite* or a full simulation is performed (see Section 7.2). Further it determines which results will be available at the end of the simulation and offers a set of plots to get a quick overview of the available simulation results.

The post processing starts by calling the `combineResults` function of the post processor. It returns a result object which contains the collected simulation results and is the final result of the simulation. It also defines methods to display them. All results contain at least the simulation parameters.

The 5G SL simulator is equipped with one postprocessor for full simulations and two postprocessors for *lite* simulations. A description of the available

postprocessors follows. For a detailed documentation of available properties and functions refer to the MATLAB code documentation in the package `simulator.results`.

- `FullPP`: By selecting this postprocessor the simulator performs a full simulation. The collected results are from type `ResultsFull` and include general results such as the preliminary SNR and SINR values, effective SINR, BLER and so on. Also the network elements are saved. In addition to the general results, additional results can be saved by selecting them with the `SaveObject` parameter. An example for an additional result would be the path loss table. All available additional results are listed in the MATLAB code documentation. The result object provides some basic functions to plot the SNR values, BLER and network geometry.

- `LiteWithNetworkPP`: By selecting this postprocessor the simulator performs a *lite* simulation. Since in the lite simulation the functionality is reduced, many parameters saved with the `FullPP` are not computed. The collected results are from type `ResultsLite` and include preliminary SNR and SINR values and the network geometry. The result object provides some basic functions to plot the SNR values and the network geometry.

- `LiteNoNetworkPP`: This postprocessor behaves like `LiteWithNetworkPP`. The difference is that network elements are not saved. Therefore also the network geometry is not available.

By default, the postprocessor `FullPP` is set. To select the *lite* simulation, the parameter `postprocessor` has to be set to a lite postprocessor in the scenario configuration file. For example:

```
1 params.postprocessor = simulation.postprocessing.
    LiteNoNetworkPP;
```

With any of the postprocessors, the results of the simulation are stored in a file inside the results folder. The filename is defined in `params.filename`. By default, the name contains the carrier frequency, the bandwidth, the time and whether the simulation is *lite* or not. The default filename can be changed by modifying the function `Parameters.setFilename`.

The result classes provide the `checkIntegrity()` method to audit if the simulation result sizes are matching the parameters. By default this check is executed by the postprocessor after every simulation and provide the user with a warning if an error is detected.

Results from multiple simulations can be combined with the `ResultsCombined` class from the `simulation.results` package as long the results are from the same type. The class provides methods to plot the combined data. An example:

```
1 res = ResultsCombined(result1, result2);
2 res.showAllPlots();
```

8 Performance Considerations

8.1 Simulation Time

Parameter	Values
simulation type	**local**, parallel
simulation extent	**full**, lite
number of users/base stations	**100/10**, 1000/100
number of chunks	**5**, 10
TTIs/slots (total)	**100**, 1000
bandwidth in MHz	**3**, 20, 100
transmission type	**SISO**, 4x4 MIMO

Table 5: Simulation parameters for simulation time evaluation. The bold parameters are the parameters set for the baseline scenario.

To give an idea on how the simulation time behaves depending on different parameters, this section lists the simulation time of different simulations on the same physical machine under comparable conditions. The absolute simulation time depends strongly on the hardware used to perform the simulations, the following numbers only can only showcase the behavior of the time consumption relative to the baseline scenario.

	baseline	**lite**	**large**	**large parallel**	**large lite**	**large lite parallel**
type	**local**			parallel		parallel
extent	**full**	lite			lite	lite
nUE/nBS	**100/10**		1000/100	1000/100	1000/100	1000/100
nChunk	**5**					
nSlot	**10**					
BW	**3**					
SISO	83	37	5780	2510	4636	2039
MIMO	286	37	7947	3476	4733	2086

Table 6: Simulation time for different simulation sizes. Simulation time is given in seconds for SISO and MIMO transmit mode in the last two lines.

	baseline	**long**	**long parallel**	**long chunks**	**long parallel chunks**
type	**local**		parallel		parallel
extent	**full**				
nUE/nBS	**100/10**				
nChunk	**5**			10	10
nSlot	**10**	200	200	100	100
BW	**3**				
SISO	83	764	338	767	241
MIMO	286	2801	1290	2816	986

Table 7: Simulation time for different simulation durations. Simulation time is given in seconds for SISO and MIMO transmit mode in the last two lines.

The simulations performed for this section are defined in the scenario file `scenarios.simulationTime` and the different parameters are defined in the launcher file `launcherFiles.launcherSimulationTime`.

Table 5 shows the parameters used for the simulation time scenarios. The parameters used for the baseline scenario are highlighted in a bold font.

	baseline	**20 MHz**	**100 MHz**
type	**local**		
extent	**full**		
nUE/nBS	**100/10**		
nChunk	**5**		
nSlot	**10**		
BW	**3**	20	100
SISO	83	143	457
MIMO	286	1417	7250

Table 8: Simulation time for different bandwidths. Simulation time is given in seconds for SISO and MIMO transmit mode in the last two lines.

In Table 6, simulations with different numbers of network elements are compared. The simulation time for *lite* simulations is short since *lite* simulations only calculate the large scale fading parameters. The simulation time

increases strongly, when the number of users and base stations is increased since the number of links to be calculated increases with both the increasing number of users and the increasing number of base stations. For MIMO simulations, the simulation time increases strongly because the LQM now deals with matrices instead of scalars. The table also shows that a significant parallelization gain can be achieved, even for very short simulations. Table 7 shows that a parallelization gain can be achieved for longer simulations, when the simulation duration is distributed to more chunks, which can be simulated in parallel.

Finally, Table 8 shows the increase in simulation time with the increase of the bandwidth.

9 Releases and Changelog

4. Vienna 5G SL Simulator release **1.3**
   Newly introduced features:

   - traffic models: VoIP, Gaming, Video Streaming, HTTP, FTP.
   - non-adaptive synchronous HARQ.
   - model-based LOS/NLOS decision
   - weighted round robin scheduling
   - technology aware precoders
   - 256-QAM and 1024-QAM
   - NOMA MIMO support
   - NOMA feedback extension
   - uplink *lite* SINR
   - QuaDRiGa MIMO support
   - small-cell cell-association bias

   Bug fixes and improvements:

   - maximum user speed is used for channel trace generation and calculating Doppler frequency for the small scale fading
   - signaling overhead toggle parameters (reference and synchronization signal can now be turned off)
   - refactored path loss calculation
   - INI power normalization bug fix
   - more optional additional results: total macroscopic fading, shadow fading, wall loss, antenna gain
   - uniform distribution of fixed number of Poisson street users and interference region base stations

3. Vienna 5G SL Simulator release **1.2**, released July 2021.
   Newly introduced features:

   - Kronecker product based precoder with adjustable number of layers and precoders
   - 5G Precoder according to 3GPP TS 38.214
   - OpenStreetMaps interface

- traffic models: Full Buffer and Constant Rate.
- path loss models: TR 38.901 RMa, UMi and UMa have been added
- mixed numerology simulations with dynamic spectrum sharing
- NOMA transmissions
- QuaDRiGa interface

Bug fixes and improvements:

- buildings can now have an arbitrary floorplan
- dummy scheduler now schedules all RBs instead of just one
- placement of clustered users with predefined cluster center
- number of sectors can be set arbitrarily for sectorized antennas (up to maximum number of sectors)
- compatibility issues with older MATLAB versions have been resolved: minimum version requirement is now R2018b

2. Vienna 5G SL Simulator **1.1**, released October 2020.
Newly introduced features:

- feedback: MIMO PMI and RI according to LTE release 8
- wraparound
- analog precoder for MIMO beamforming
- antenna array according to 3GPP TR 38.901

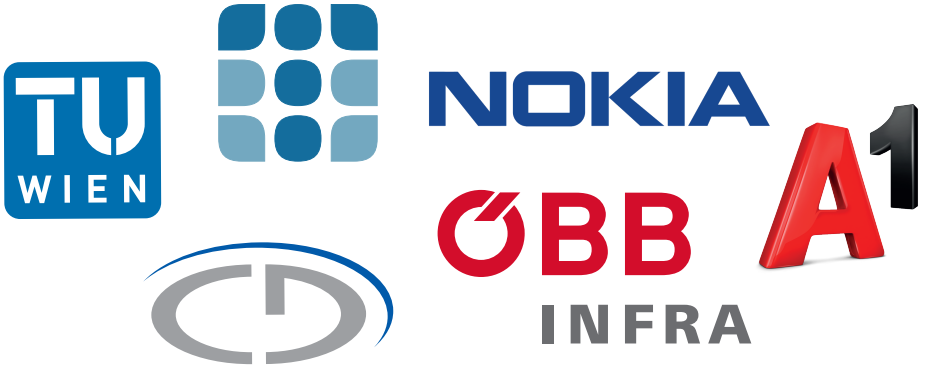
Bug fixes and improvements:

- +simulation.results and simulation.postprocessing packages are refactored
- bug fix for feedback read out at handover

1. Vienna 5G SL Simulator 1.0, released October 2018

Acknowledgements

The financial support by the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged. This work has been co-financed by A1 Telekom Austria AG, ÖBB Infrastruktur AG and Nokia Solutions and Networks.



References

[1] *OpenStreetMap*. https://www.openstreetmap.org/. Accessed: 2021-04-19.

[2] 3rd Generation Partnership Project (3GPP). *Evolved Universal Terrestrial Radio Access (E-UTRA) physical channels and modulation*. TS 36.211. 3rd Generation Partnership Project (3GPP), Jan. 2015.

[3] Stefan Pratschner, Martin Klaus Müller, Fjolla Ademaj, Armand Nabavi, Bashar Tahir, Stefan Schwarz, and Markus Rupp. "Verification of the Vienna 5G Link and System Level Simulators and Their Interaction". In: *Annual Consumer Communications Networking Conference (CCNC)*. Las Vegas, NV, USA, Jan. 2019, pp. 1–8. DOI: 10.1109/CCNC.2019.8651694.

[4] A. Fastenbauer, M. K. Muller, and M. Rupp. "Investigation of Wraparound Techniques for the Simulation of Wireless Cellular Networks". In: *WSA 2013; 23rd International ITG Workshop on Smart Antennas*. 2019.

[5] T. Dittrich, M. Rupp, and M. Taranetz. "An Efficient Method for Avoiding Shadow Fading Maps in System Level Simulations". In: *WSA 2017; 21st International ITG Workshop on Smart Antennas*. Mar. 2017, pp. 1–8.

[6] 3rd Generation Partnership Project (3GPP). *High Speed Downlink Packet Access: UE Radio Transmission and Reception*. Tech. rep. 3rd Generation Partnership Project (3GPP), 2002.

[7] TSGR1 010030. *Further Results on CPICH Interference Cancellation as A Means for Increasing DL Capacity*. Tech. rep. Intel Corporation, 2001.

[8] 3rd Generation Partnership Project (3GPP). *Radio transmission and reception, annex c.3 propagation models*. Tech. rep. 3rd Generation Partnership Project (3GPP), 2009.

[9] 3rd Generation Partnership Project (3GPP). *Universal Mobile Telecommunications System (UMTS) Deployment aspects*. TR 25.943. 3rd Generation Partnership Project (3GPP), Feb. 2010.

[10] T. B. Sorensen, P. E. Mogensen, and F. Frederiksen. "Extension of the ITU channel models for wideband (OFDM) systems". In: *VTC-2005-Fall. 2005 IEEE 62nd Vehicular Technology Conference, 2005.* IEEE, 2005. DOI: 10.1109/vetecf.2005.1557939.

[11] S. Jaeckel, L. Raschkowski, K. Boerner, and L. Thiele. "QuaDRiGa: A 3D Multi-Cell Channel Model With Time Evolution For Enabling Virtual Field Trials". In: *IEEE Transactions On Antennas and Propagation* 62 (2014), pp. 3242–3256.

[12] S. Jaeckel, L. Raschkowski, K. Boerner, L. Thiele, F. Burkhardt, and E. Eberlein. *QuaDRiGa - Quasi Deterministic Radio Channel Generator, User Manual and Documentation v.2.4.0*. Tech. rep. Fraunhofer Heinrich Hertz Institute, 2020.

[13] Y. R. Zheng and Chengshan Xiao. "Simulation models with correct statistical properties for rayleigh fading channels". In: *IEEE Transactions on Communications* 51.6 (June 2003), pp. 920–928. DOI: 10.1109/tcomm.2003.813259.

[14] J. Colom Ikuno. "System Level Modeling and Optimization of the LTE Downlink". PhD thesis. TU Wien, 2013.

[15] Lei Wan, Shiauhe Tsai, and M. Almgren. "A fading-insensitive performance metric for a unified link quality model". In: *IEEE Wireless Communications and Networking Conference, 2006. WCNC 2006.* 4 (2006), pp. 2110–2114.

[16] Stefan Schwarz, Christian Mehlfuhrer, and Markus Rupp. "Calculation of the spatial preprocessing and link adaption feedback for 3GPP UMTS/LTE". In: *2010 Wireless Advanced 2010*. IEEE, June 2010. DOI: 10.1109/wiad.2010.5544947.

[17] 3rd Generation Partnership Project (3GPP). *Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures*. TS 36.213. 3rd Generation Partnership Project (3GPP), Jan. 2015.

[18] 3rd Generation Partnership Project (3GPP). *5G;NR;Physical layer procedures for data*. TS 38.214. 3rd Generation Partnership Project (3GPP), Oct. 2018.

[19] Y. Xie, S. Jin, J. Wang, Y. Zhu, X. Gao, and Y. Huang. "A limited feedback scheme for 3D multiuser MIMO based on Kronecker product codebook". In: *2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. 2013, pp. 1130–1135. DOI: 10.1109/PIMRC.2013.6666308.

[20] 3rd Generation Partnership Project (3GPP). *Study on channel model for frequencies from 0.5 to 100 GHz (Release 16)*. Tech. rep. 3rd Generation Partnership Project (3GPP), 2020.

[21] 3rd Generation Partnership Project (3GPP). *Evolved Universal Terrestrial Radio Access (E-UTRA); Further advancements for E-UTRA physical layer aspects*. TR 36.814. 3rd Generation Partnership Project (3GPP), Mar. 2017.

[22] 3rd Generation Partnership Project (3GPP). *LTE physical layer framework for performance verification*. TSG RAN1-070674. 3rd Generation Partnership Project (3GPP), Feb. 2007.

[23] 3rd Generation Partnership Project (3GPP). *Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification*. TS 36.321. 3rd Generation Partnership Project (3GPP), June 2021.

[24] Abuu B. Kihero, Muhammad Sohaib J. Solaija, and Hüseyin Arslan. "Inter-Numerology Interference for Beyond 5G". In: *IEEE Access* 7 (2019), pp. 146512–146523. DOI: 10.1109/ACCESS.2019.2946084.