User Manual

5G New Radio Sidelink Link-Level Simulator v1.0

Chunmei Liu
Fernando J. Cintrón
Richard Rouil
National Institute of Standards and Technology

Peng Liu
Prometheus Computing LLC, Sylva, North Carolina
National Institute of Standards and Technology

Chen Shen
Georgetown University, Washington, D.C.
National Institute of Standards and Technology
Now at Google

Lyutianyang Zhang
Liu Cao
Sumit Roy
University of Washington[1]

October, 2022

[1]This work was performed under the financial assistance award 70NANB20H179 from U.S. Department of Commerce, National Institute of Standards and Technology.

Table of Contents

List of Tables

List of Figures

Abbreviations

3GPP the Third Generation Partnership Project

4G Fourth Generation

5G Fifth Generation

ACK Acknowledgement

AGC Automatic Gain Control

AWGN Additive White Gaussian Noise

BLER Block Error Rate

BWP Bandwidth Part

CP-OFDM Cyclic Prefix–Orthogonal Frequency Division Multiplexing

CQI Channel Quality Indication

CRC Cyclic Redundancy Check

CSI Channel State Information

CSI-RS Channel-State Information Reference Signal

D2D Device-to-Device

DM-RS Demodulation Reference Signal

FDD Frequency Division Duplex

FFT Fast Fourier Transform

FR1 Frequency Range 1

FR2 Frequency Range 2

HARQ Hybrid Automatic Repeat Request

I2I Indoor-to-Indoor

IE Information Element

LDPC Low Density Parity Check

LLR Log-Likelihood Ratio

LLS Link-Level Simulator

LOS Line-of-Sight

LTE Long Term Evolution

MCS Modulation and Coding Scheme

MIMO Multiple-Input and Multiple-Output

NACK Negative Acknowledgement

NIST National Institute of Standards and Technology

NLOS Non-Line-of-Sight

NR New Radio

O2I Outdoor-to-Indoor

O2O Outdoor-to-Outdoor

OFDM Orthogonal Frequency-Division Multiplexing

PDP Power Delay Profile

PHY Physical Layer

POI Point of Interest

PRB Physical Resource Block

ProSe Proximity Services

PSBCH Physical Sidelink Broadcast Channel

PSCCH Physical Sidelink Control Channel

PSFCH Physical Sidelink Feedback Channel

PSSCH Physical Sidelink Shared Channel

PT-RS Phase-Tracking Reference Signal

QAM Quadrature Amplitude Modulation

QPSK Quadrature Phase Shift Keying

RB Resource Block

RE Resource Element

RRC Radio Resource Control

S-SSB Sidelink Synchronization Signal Block

SCI Sidelink Control Information

SCI1 1st-stage Sidelink Control Information

SCI2 2nd-stage Sidelink Control Information

SL-SCH Sidelink Shared Channel

SNR Signal-to-Noise Ratio

TB Transport Block

TBS Transport Block Size

TDD Time Division Duplex

UE User Equipment

V2X Vehicle-to-Everything

1. Introduction

In cellular networks, using sidelink, two or more devices can communicate with each other without signal relay through base stations. That is, communications among devices become possible not only when they are in network coverage, but also when they are out of coverage. Sidelink also has the potential to improve performance, such as a lower latency due to fewer hops along the communication path. These properties lead to increasing research activities in sidelink communications since it was first introduced in the Third Generation Partnership Project (3GPP) for Fourth Generation (4G) Long Term Evolution (LTE). Accordingly, its associated services have been extended from its original Proximity Services (ProSe) to Vehicle-to-Everything (V2X), and potentially other services that may benefit from it, such as smart city.

With 3GPP advancing from 4G LTE to Fifth Generation (5G) New Radio (NR), a new sidelink design was introduced, that includes two-stage Sidelink Control Informations (SCIs) and feedback-based Hybrid Automatic Repeat Request (HARQ), among other capabilities. While MathWorks[2] provides 5G toolbox with a rich set of 5G NR Physical Layer (PHY) modules [1], and Vienna simulator suite[2] includes 5G downlink and uplink Link-Level Simulators (LLSs) [2], to the best of our knowledge, there is currently no publicly accessible NR sidelink LLS yet.

To meet simulation needs and to fill the gap, we developed this 5G NR Sidelink LLS. To ensure adequate functionalities, flexibility, and correctness, the Sidelink LLS follows closely 3GPP Release 16 specifications on 5G NR sidelink, and has a highly flexible structure that is easy to adapt to various NR sidelink link-level simulations of interest. The Sidelink LLS also keeps the branches of Vienna 5G downlink and uplink LLS, so that it supports all downlink, uplink, and sidelink link-level simulations.

The 5G NR Sidelink LLS is MATLAB based. The current release of the Sidelink LLS focuses on Frequency Division Duplex (FDD) unicast point-to-point simulations and data channel, with Time Division Duplex (TDD) and study of other channels and signals to be added in future releases. Instead of developing an NR sidelink LLS from scratch, we made use of the rich set of PHY modules from Vienna downlink and uplink LLS and MathWorks 5G toolbox, and focused on the development of NR sidelink specific features. The modules reused include general LLS software structure and some basic functionalities from Vienna 5G LLS, and channel coding from MathWorks 5G toolbox. The NR sidelink features developed include, but not limited to, sidelink slots, sidelink Bandwidth Part (BWP) and resource pool, sidelink slot format, data and control multiplexing and scrambling, sidelink layer mapping and precoding, and blind- and feedback-based HARQ. To facilitate simulations, we also developed additional features such as constant management, range conversion, and adaptive sweeping, which will be described in detail later in this manual.

[2]Certain commercial equipment, instruments, or materials are identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

Although the focus of the current release is on data channel, except Channel State Information (CSI) and Phase-Tracking Reference Signal (PT-RS) which are left for future releases, all other control channels and physical signals are implemented in sidelink slots and assumed error-free, including: Physical Sidelink Broadcast Channel (PSBCH), Physical Sidelink Control Channel (PSCCH), Physical Sidelink Feedback Channel (PSFCH), and reference signal Demodulation Reference Signal (DM-RS). That is, there is no loss on broadcast, 1st-stage SCI1, Acknowledgements (ACKs), and Negative Acknowledgements (NACKs). Accordingly, dummy symbols with unit power are used to fill the Resource Elements (REs) for PSCCH and DM-RS. Meanwhile, Automatic Gain Control (AGC) and guard symbols are implemented per the 3GPP TS 38.211 [3]. Additionally, instead of implementing each field, dummy bits are used for SCI2 in Physical Sidelink Shared Channel (PSSCH) and both perfect and error-prone SCI2 can be simulated by simulation configurations.

In this user manual, we will describe the general idea and scope of the 5G NR Sidelink LLS, together with its implementation details and usage. To facilitate the usage of the Sidelink LLS, we will also showcase a couple of simulation examples.

2. Quick Start

The NR Sidelink LLS can be used in two ways. One way is to use a pre-defined sidelink simulation scenario, as discussed in Sec. 5.2. Another way is to implement sidelink-only scripts with customized features and configurations, as discussed in Sec. 5.1. This quick start guide describes the use of the first way, using a pre-defined sidelink scenario.

1. Open the script `main.m` in the simulator's root directory.
2. Make sure the `SidelinkScenario` is selected

```
1. simulationScenario = 'SidelinkScenario';
```

3. All the sidelink-specific parameters are defined in `SidelinkScenario`, which can be modified. A customized scenario can also be created under `/Scenarios`.
4. As HARQ is implemented in the Sidelink LLS, the slot indexing becomes critical. Therefore the parallel mode using `parfor` is strongly discouraged

```
1. % loop over sweep parameter

2. for iSweep = 1:length(simParams.simulation.sweepValue) %
   this may be 'for' or 'parfor' for UL/DL, but can only be
   'for' for SL
```

5. Run the `main.m` script. Block Error Rate (BLER) plots for data and 2nd-stage Sidelink Control Information (SCI2) will be shown when simulation is finished, and the results for sidelink are stored in the cell `sl_result` as well as in the directory `/results/sidelink/`.

3. System Requirements

The 5G NR Sidelink LLS is built upon the MATLAB-based Vienna 5G NR Link Level Simulator [2, 4]. It requires the Vienna 5G NR Link Level Simulator Release 1.2 and MATLAB version 2021b with the following toolboxes: communication toolbox, 5G toolbox, and signal blocks.

4. Installation Guide

The Sidelink LLS is released as a patch file to the Vienna 5G LLS v1.2. Once the original Vienna LLS code is downloaded, the patch file can be installed using any common patch application method. For convenience, one common method that uses `patch` is described below.

1. Launch a git client (e.g., git bash) or terminal;
2. Change the working directory to that holding the original Vienna LLS;
3. Deploy the patch file using `patch` command:

```
1. patch --binary -s -p1 < path-to-patch/5G_NR_Sidelink_Link-
   Level_Simulator.patch
```

where `path-to-patch` is to be replaced with the actual directory that holds the patch file.

The patch should be applied to the Vienna 5G LLS v1.2 without an error. However, if rejected files are created, pay attention to the differences indicated in them and make appropriate changes as needed.

5. Simulation Examples and Scenarios

In this section, we provide several simulation examples that utilize the pre-defined `SidelinkScenario`. Each example focuses on evaluating the impact of a sidelink-specific configuration or functionality, in a specified propagation environment. We then describe the pre-defined `SidelinkScenario` in detail.

5.1 Simulation Examples

This section introduces the simulation examples provided in the `/Examples/Sidelink/` directory that focus specifically on link-level simulations for NR sidelink. Each example utilizes the pre-defined `SidelinkScenario` as introduced in Sec. 2, and modifies the parameters for a specific research purpose. In addition, the `Adaptive Sweeping Value Selection` feature, as introduced in Sec. 6.3, is implemented in each of the examples.

Note, the project root directory and all its sub-directories must be added to the MATLAB search path, and the MATLAB `Current Folder` must be set to the project root directory before running the simulation examples.

Sidelink Link-Level Simulation — Baseline
The example script `Sidelink_baseline_with_adaptive_sweep.m` performs a link-level simulation to study sidelink communication range, with a baseline configuration on NR sidelink. Some key configurations and parameters include: $\mu = 0$, Quadrature Phase Shift Keying (QPSK) modulation, code rate $= 78/1024$, unicast, 1×1 antenna, PSFCH disabled with one transmission, and Additive White Gaussian Noise (AWGN) channel. In addition, error-free SCI2 is configured for decoding of Transport Blocks (TBs).

At the end of a script execution, the plots for data and SCI2 BLERs vs. Signal-to-Noise Ratio (SNR) are generated. Note that in this scenario, although erroneous SCI2 does not impact data decoding, they are still recorded for analysis purpose (refer to Sec. 12 for details). In addition, the parameters related to communication performance (range, throughput, and latency) are calculated corresponding to a target loss rate of 0.5 %, noise figure of 9 dB, and channel model of Outdoor-to-Outdoor (O2O) Line-of-Sight (LOS) (refer to Sec. 10.1.1). These numbers are configurable to fit simulation needs.

Sidelink Link-Level Simulation — Error-Prone SCI2
The example script `Sidelink_error_prone_SCI2_with_adaptive_sweep.m` extends the baseline configuration and performs a link-level simulation that considers the decoding of both TB and its corresponding SCI2. The error-prone SCI2 feature is enabled by

```
1. simParams.sidelink.SCI2ErrorProne = true;
```

The same as the baseline script, this script generates plots for both data and SCI2 BLERs vs. SNR, and the parameters of range, throughput and latency as those in baseline are calculated.

Sidelink Link-Level Simulation — Feedback-Based HARQ

The example script `Sidelink_Feedback_HARQ_with_adaptive_sweep.m` extends the baseline configuration and performs a link-level simulation with feedback-based HARQ enabled.

```
simParams.sidelink.PSFCH_enabled = true;
```

The default value of the maximum number of transmissions is 4. The link-level performances with different maximum number of transmissions can be evaluated by setting the parameter `simParams.sidelink.harq.maxTrans` to a number between 1 and 32, which is consistent with 3GPP:

```
simParams.sidelink.harq.maxTrans = 4;
```

This script also generates plots for both data and SCI2 BLERs vs. SNR, as well as the parameters of range, throughput and latency, with the same configurations as baseline scenario.

5.2 Simulation Scenarios

In addition to the pre-defined scenarios from the original Vienna LLS [2, 4], such as `genericScenario`, `flexibleNumerology`, etc., the `SidelinkScenario` is specifically pre-defined for sidelink simulation.

In `SidelinkScenario`, sidelink simulation is enabled by

```
scStr.simulation.simulateSidelink = true;
```

whereas downlink and uplink are disabled by

```
scStr.simulation.simulateDownlink = false; % Downlink
scStr.simulation.simulateUplink = false; % Uplink
```

The parameters related to simulation configuration include:

```
scStr.simulation.sweepValue = linspace(150, 110, 10);
```

which defines the path loss values, and

```
scStr.simulation.nFrames = 1000;
```

which defines the number of sidelink slots to simulate.

The parameters related to Multiple-Input and Multiple-Output (MIMO) include:

```
scStr.simulation.nAntennasUser = [1, 1]; % per UE; number of
antennas at the user.  For 1 link and 2-layer case, assign [2,
2]
scStr.modulation.nStreams = [1]; % per Link; number of active
spatial streams.  For 1 link and 2-layer case, assign [2]
scStr.modulation.precodingMatrix1 = 1; % Link 1 employed precoding
matrix, For 1 link and 2-layer case, assign 1/sqrt(2)*eye(2)
```

The parameters for sidelink configuration are provided in the `General and Sidelink` section, specifically `scStr.general` and `scStr.sidelink`, where parameters for BWP,

resource pool, and error-prone/error-free SCI2, are initialized.

6. Simulator Structure

The Sidelink LLS inherits the flexible structure from Vienna LLS [2, 4] with modifications to support sidelink, and bases the simulator structure on `Link` object. Additionally, sidelink-specific information and configurations are added accordingly to the structure and objects. These changes are discussed in this section.

In addition, to facilitate scalability and usability, new features were developed, including constant management, adaptive sweeping value selection, and communication range evaluation. These features are discussed in this section as well.

6.1 General Structure

The 5G NR sidelink LLS inherits the structure from Vienna 5G LLS together with its general features [2, 4], with modifications to support sidelink. The basic simulator structure is illustrated in Fig. 1.

When a simulation is initiated, the simulator first loads a scenario profile that contains the setup of the two User Equipments (UEs) and initializes the link object that represents the connection between the two UEs. The scenario profile also specifies the transmission parameters, which include but not limited to antenna configurations, precoding matrix, channel coding scheme, waveform, channel model, and equalizer. Next, in the case that sidelink simulation is enabled, the `SL_BWP` object is created according to the configurations in the scenario profile. The slots where Sidelink Synchronization Signal Blocks (S-SSBs) are transmitted are also calculated. Then, a slot-by-slot simulation follows in the form of a nested for-loop, and each iteration of the outer loop is associated with a sweep parameter. Finally, the results of interest are recorded and post-processed after the completion of a specified number of slots.

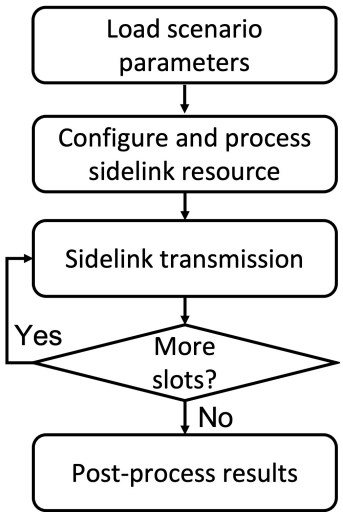

Fig. 1. Basic Simulator Structure.

The current release of the Sidelink LLS focuses on data channel PSSCH, so does the box "Sidelink transmission" in Fig. 1. Fig. 2 illustrates at high level the sidelink transmitter-Channel-receiver processing chain. The development of the Sidelink LLS follows this processing chain, which will be described in detail in later sections with focus on sidelink specific implementations.

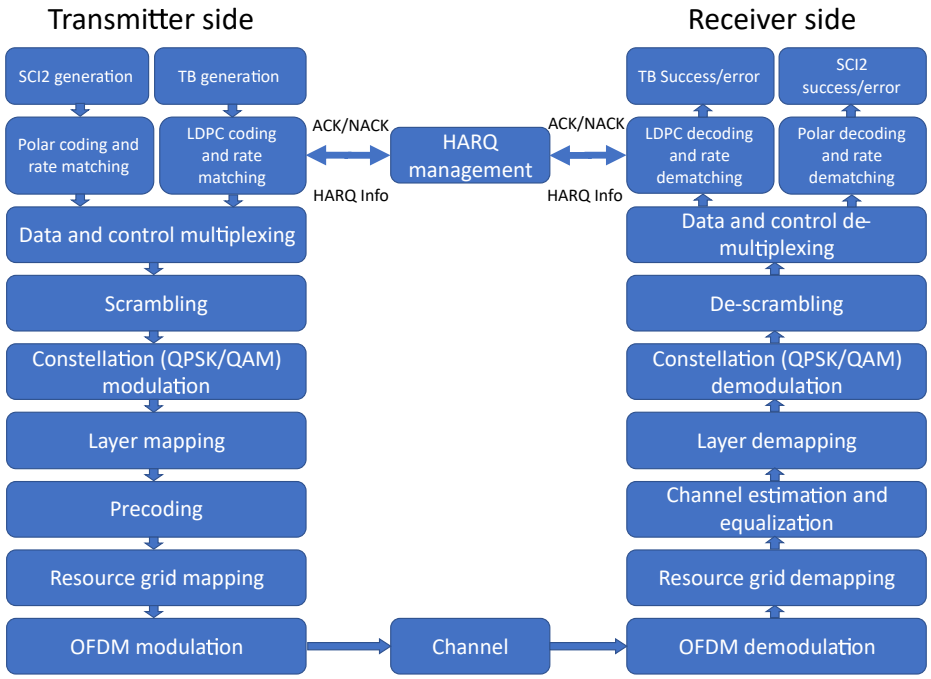

Fig. 2. Sidelink Transmitter-Channel-Receiver Processing Chain.

Similar to the original Vienna LLS [2], the Sidelink LLS uses the `Link` object to carry all the information and functionalities of the connection between nodes [4]. In the initial implementation of one single-directional unicast transmission between a transmitter and a receiver, the following configurations specific to sidelink are implemented.

6.1.1 Topology Configuration

The topology information related to `Link` object generation is specified in the `Sidelink Scenario`:

```
1. scStr.topology.nodes = ['UE1,UE2'];

2. scStr.topology.primaryLinks = ['UE1:UE2'];

3. scStr.topology.interferenceGeneration = 'Manual';
```

where `scStr.topology.nodes` specifies the UEs for sidelink, and `scStr.topology.primaryLinks` specifies the connected link(s). The first entry of each pair of the `primary Links` represents the transmitter and the second represents the receiver. `scStr.topology.`

`interferenceGeneration` is set to `Manual` to ensure a controlled configuration of attenuation and interference links.

6.1.2 Link Object

Besides the connection properties, signals, and signal processing functionalities, as introduced in the original Vienna LLS user manual [4], the Sidelink LLS carries the additional sidelink-specific parameters, signals and functions, including but not limited to:

For SCI2:

- `SCI2Bits`: SCI2 bits before channel coding,
- `NrSCI2Bits`: number of SCI2 bits before channel coding,
- `SCI2ChannelCoder`: structure containing SCI2 channel coder parameters,
- `TransmitSCI2Symbols`: SCI2 modulation symbols to be transmitted,
- `NrSCI2Symbols`: number of SCI2 modulation symbols,
- `DecodedSCI2Bits`: SCI2 bits after channel decoding at the receiver.

For data:

- `NrDataBitsAfterRateMatching`: number of data bits after rate matching,
- `encodeDLSCH`: channel coder object for data that enables HARQ,
- `TransmitDataSymbols`: data modulation symbols to be transmitted,
- `decodeDLSCH`: channel decoder object for data that enables HARQ,
- `decodedDataBits`: data bits after channel decoding at the receiver, etc.

For other channel or signal related parameters:

- `PSFCH_overhead_indication`: boolean that indicates if this slot contains PSFCH,
- `MultiplexedBits`: bit sequence after data and SCI2 multiplexing,
- `PSSCHScramblingSequence`: scrambling sequence for PSSCH scrambling and descrambling,
- `sl_slot_struct_tx`: sidelink slot format object at the transmitter,
- `sl_slot_struct_rx`: sidelink slot format object at the receiver, etc.

For radio resource control and other system parameters:

- `sl_bwp_obj`: `SL_BWP` object,
- `harqEntity`: object for HARQ Entity that manages processes and redundancy versions for HARQ transmissions,
- `TB_info`: a property that keeps record of transmission status for each TB, and is used in result analysis,
- `iFrame`: iteration index related to slot index, starting from 1,
- `i_SL`: sidelink slot index, starting from 0 until $10240 \times 2^{\mu} - 1$, where μ is numerology, and
- `SCI2ErrorProne`: a boolean that indicates whether error-prone SCI2 is enabled or not.

Besides the sidelink-specific properties and functionalities, the majority of the ones in the original Vienna LLS [2, 4] are reused in the Sidelink LLS for the same original purpose.

6.1.3 General Simulation Parameters

The general simulation parameters for uplink and downlink were described in the original Vienna 5G LLS user manual [4]. In this section, we discuss the parameters that are used in sidelink link-level simulations, as well as those that are used in uplink/downlink simulations but not in the sidelink scenario.

To enable sidelink simulation, the following parameter needs to be configured:

```
1. scStr.simulation.simulateDownlink = false;

2. scStr.simulation.simulateUplink = false;

3. scStr.simulation.simulateSidelink = true;
```

As the sidelink simulation is currently developed for sidelink communication only, uplink and downlink are not enabled at the same time when sidelink is in use. The corresponding parameters are set to `false` in this case. In addition, the parameter `scStr.simulation.simulateSidelink` is adopted instead of `scStr.simulation.simulateD2D`.

The same as the original Vienna LLS [2, 4], the sidelink simulation uses path loss as its sweep parameter, whereas the SNR is calculated at the receiver for each sweep value. The corresponding configuration is as follows:

```
1. scStr.simulation.sweepParam = 'simulation.pathloss';

2. scStr.simulation.sweepValue = linspace(150, 110, 10);

3. scStr.simulation.nFrames = 1000;
```

where `scStr.simulation.sweepValue` defines the sweep values to be used in the simulation, and `scStr.simulation.nFrames` defines the number of slots for the simulation of each sweep value.

It is important to note that all the sweep values defined by `scStr.simulation.sweepValue` will be applied in the simulations when the Adaptive Sweeping Value Selection function is not applied, as introduced in Sec. 6.3. In case when the Adaptive Sweeping Value Selection is enabled, only the range defined in `scStr.simulation.sweepValue` are utilized, together with the granularity configured in the `adaptiveSweep` object (Sec. 6.3).

Moreover, as the sidelink link-level simulation results currently focus on BLER, a separate method for data collecting and processing is implemented. Accordingly, some parameters in the original Vienna LLS [2, 4] related to data recording and plot displaying are no longer needed. These parameters include

- `scStr.simulation.plotResultsFor`,
- `scStr.simulation.plotOverSNR`,
- `scStr.simulation.plotPAPR`, and
- `scStr.simulation.saveData`.

6.2 Constant Management

In 3GPP standards, parameters are often related to each other in the form of constant. Such constants are often presented using tables, and may be subject to update following the 3GPP meetings. To help researchers keep track of the constant updates without roaming through the code to make modifications, the `Constant3GPP` class was developed.

Each of the 3GPP tables that are utilized in the Sidelink LLS is created as a constant table in the code under a unique function. The name of the function is added to the cell array of `dataList` under the `Constant3GPP` class. By running the `getData` function, `dataList` accesses the table of interest versatilely in the form of anonymous function. The `Constant3GPP` class and the constant tables are defined in the `/Constant3GPP/` directory. The users can also create and manage their own constant tables and use `getData` and `dataList` to access the data in them.

6.3 Adaptive Sweeping Value Selection

A common problem that researchers often encounter is that, to generate desired outputs, the range of an input simulation parameter is unknown. It often requires manual trial and error, which is typically time consuming, especially when the simulation for a single data point takes a long time. It also makes it unrealistic to run batch executions.

The adaptive sweep function focuses on solving this problem and making executions automatic by

1. setting a wide range for the input parameter that includes values to generate desired outputs, as well as the granularities for both the input and output based on acceptable accuracies. This is realized by the constructor function `adaptiveSweep` of the `adaptiveSweep` class.

2. receiving the output values for the end points of the input range, and adaptively preparing the parameters needed for searching for the next input range. This is realized by the `receive_sweep_output` function. And

3. using a "divide and conquer" strategy to determine the next input value. This is realized by the `get_next_sweep_to_calculate` function.

4. continuing the above steps until the configured granularities for the input and output are reached, which ensures the desired accuracies.

In addition, the adaptive sweep function also allows for designating a Point of Interest (POI) and its granularity, where a series of input values around the POI can be generated with a finer granularity, which leads to a higher accuracy around the POI.

The `adaptiveSweep` class is contained in the `/Utils/` directory, and is used in the `main_adaptive_SL_batch.m` script and the sidelink simulation examples in the `/Examples/Sidelink/` directory.

6.4 Communication Range Evaluation

Since the introduction of sidelink, how far it can reach has been of special interest. To facilitate researches on this topic, a feature that retrieves communication range and corresponding performance metrics at the range edge was developed in the Sidelink LLS. In this context, communication range refers to the maximum distance between a transmitter and a receiver such that the transport block transmitted could be received with an error probability not exceeding a configured target loss rate.

In the Sidelink LLS, this feature is realized by the function `getRangeParameters`:

```
1. function [targetRange,targetPathloss,targetSNR,targetThroughput,...
2.          targetLatency] = getRangeParameters(targetLossRate, ...
3.          bler_data,pathloss,NF,SNR,throughput,latency,Channel_model)
```

The outputs of this function are the sidelink communication range and performance at the maximum range, including throughput and latency, together with the corresponding SNR and path loss. The inputs include the configured target loss rate and BLER data points. In addition, considering that a receiver always introduces noise, a noise figure (`NF`) is also included. The communication range is calculated using the function `getDistanceFromPathloss`, which requires a propogation path loss model. The built-in path loss models in the Sidelink LLS come from 3GPP [5] (refer to Sec.10.1.1) and include `O2O_LOS`, `O2O_NLOS`, `O2I_LOS`, `O2I_NLOS`, or `I2I_LOS`, which can be easily extended to other path loss models as needed.

In the Sidelink LLS, `getRangeParameters` is contained in the /SL_Utils/ directory, and is used at the end of each sidelink example as discussed in Sec. 5.1.

7. Sidelink Resource

A sidelink slot can contain multiple physical channels and reference signals. Depending on different configurations and slot number, the physical channels and reference signals in a sidelink slot may vary. In this section, we first discuss the possible content of a sidelink slot, and then introduce how the corresponding sidelink configurations are organized in the `SL_BWP` and `SidelinkResourcePool` classes. Then, we introduce the `SL_SlotStructure` class, which implements the sidelink slot format utilizing the resource configurations in the `SL_BWP` and `SidelinkResourcePool` objects.

7.1 Sidelink Slot Review

An NR sidelink slot is composed of 12 or 14 Orthogonal Frequency-Division Multiplexing (OFDM) symbols in the time domain, depending on the cyclic prefix and numerology, and 10 to 275 Physical Resource Blocks (PRBs) in the frequency domain, depending on the bandwidth. A sidelink slot can contain the following physical channels, reference signals, and OFDM symbols:

- PSCCH, including 1st-stage Sidelink Control Information (SCI1) and its associated DM-RS,
- PSSCH, including data, SCI2, and their associated DM-RS,
- PSFCH,
- AGC, which may include AGC for PSCCH and PSSCH, and AGC for PSFCH, and
- guard symbol(s), which may include guard symbol for PSCCH and PSSCH, and guard symbol for PSFCH.

In addition, PSSCH may also contain Channel-State Information Reference Signal (CSI-RS), and in case of Frequency Range 2 (FR2), PT-RS. These two reference signals are not within the scope of the current release of the Sidelink LLS.

When feedback-based HARQ is not enabled, a sidelink slot does not contain PSFCH. Fig. 3 shows an example of a sidelink slot without PSFCH. Within the allocated OFDM symbols, the first symbol is the AGC symbol, which is a duplicate of the symbol that follows it. The last symbol is an empty symbol, which is the guard symbol. Starting from the second allocated symbol, two or three symbols can contain PSCCH. Depending on the number of allocated symbols and the number of symbols that contains PSCCH, two, three, or four symbols can contain PSSCH DM-RS, and SCI2 occupies the REs starting from the lowest index of the available REs of the first PSSCH DM-RS symbol. The remaining REs are assigned to the data symbols in PSSCH.

When feedback-based HARQ is enabled, a sidelink slot may contain PSFCH. Fig. 4 shows a sidelink slot that contains PSFCH, where the last three of the allocated OFDM symbols are occupied by PSFCH AGC, PSFCH, and its guard symbol.

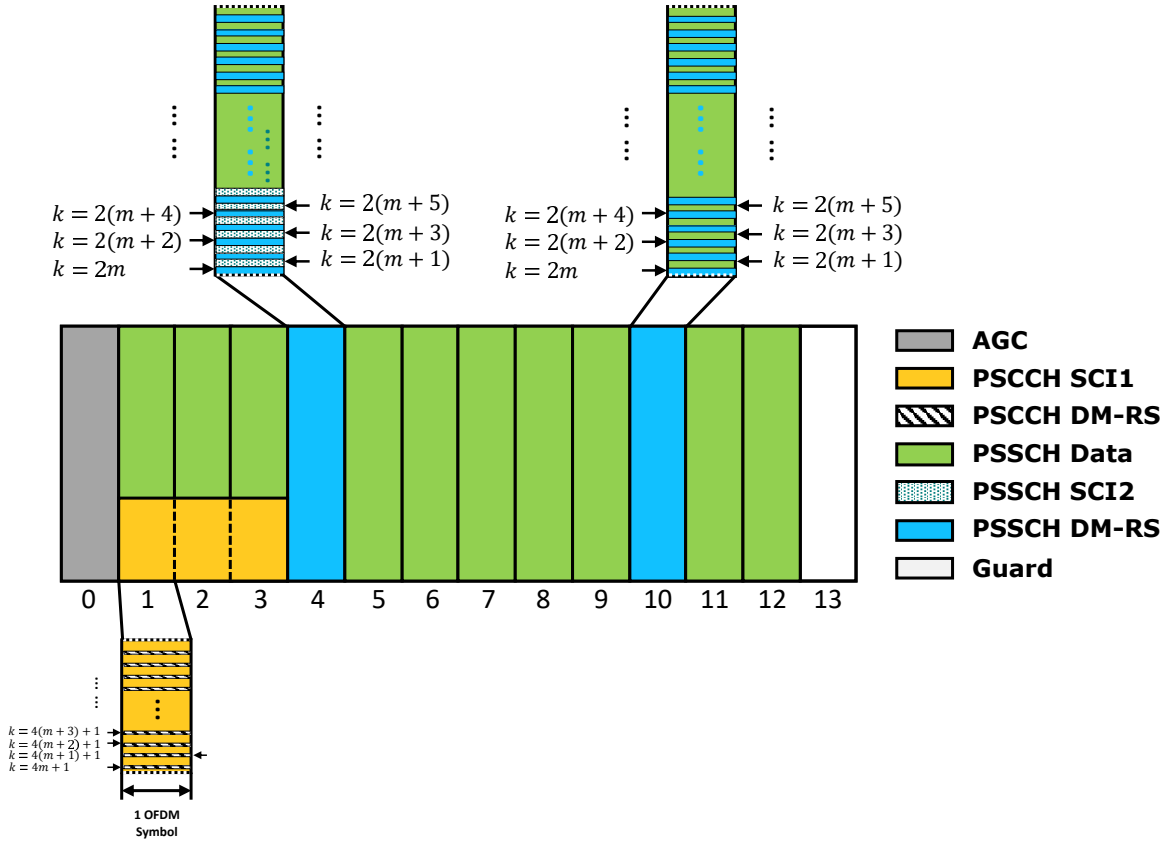

Fig. 3. An NR sidelink slot without PSFCH.

The configurations for the NR sidelink slots are contained in the sidelink BWP and its corresponding sidelink resource pool. In the following sections, we introduce how sidelink BWP and resource pool, as well as the sidelink slot format itself, are implemented in the Sidelink LLS.

7.2 Sidelink Bandwidth Part and Resource Pool

7.2.1 Sidelink Bandwidth Part

Complying with 3GPP [6], `SL_BWP` class is developed for NR sidelink BWP configuration in Radio Resource Control (RRC). It provides two sets of properties:

1) Generic sidelink BWP configuration: `sl_BWP_Generic`, which contains generic sidelink BWP information in both frequency domain and time domain. The frequency domain configuration includes the start index of PRB, the number of allocated PRBs, subcarrier spacing, and cyclic prefix, whereas the time domain configuration includes index of start OFDM symbol, and the number of OFDM symbols for a sidelink slot. This property set is configured via the constructor method.

2) Sidelink resource pool configuration: `sl_BWP_PoolConfig`. This property set is

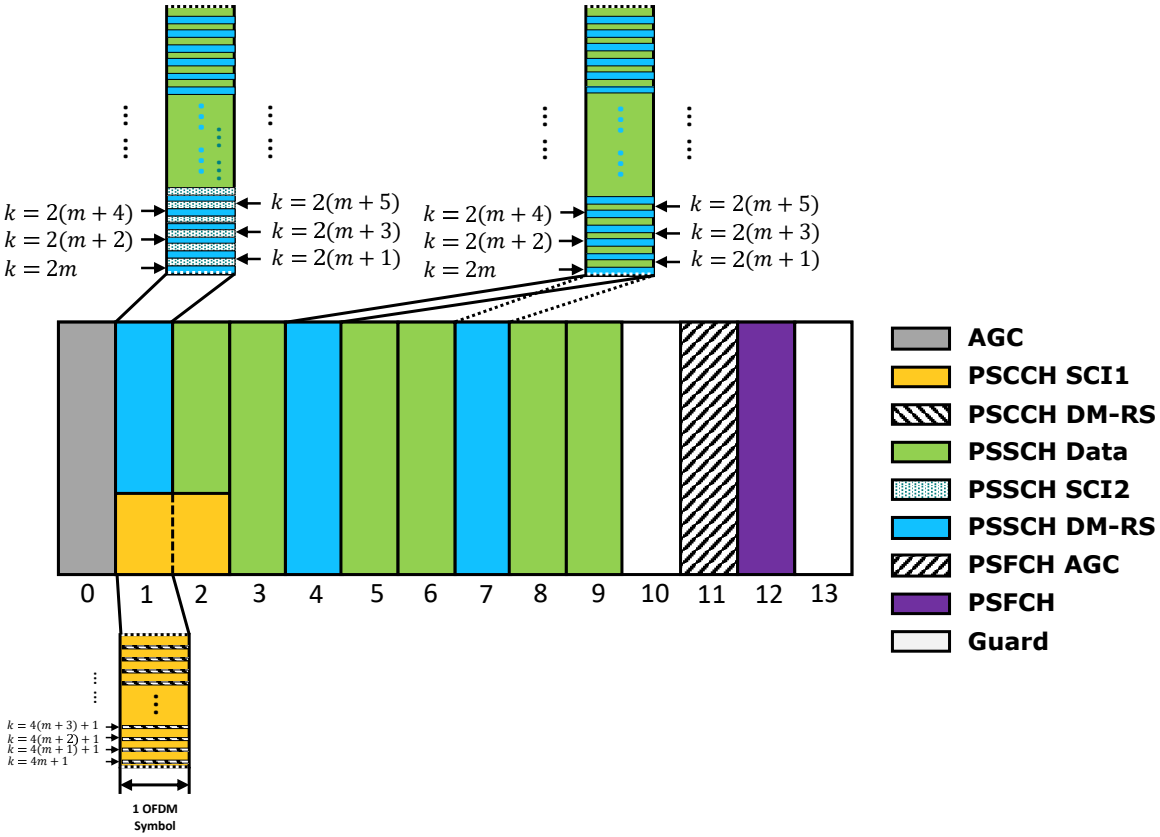

Fig. 4. An NR sidelink slot with PSFCH.

configured using `SidelinkResourcePool` class, and its constructor method is called in the `SL_BWP` constructor method.

The constructor method is defined as:

```
function obj = SL_BWP(type, BW, RB_start, nPRB, mu, cyclicPrefix,
sl_LengthSymbols, sl_StartSymbol, FR, sl_TimeResourcePSCCH,
sl_FreqResourcePSCCH, sl_PSSCH_DMRS_TimePatternList,
sl_BetaOffsets2ndSCI, sl_Scaling, sl_RB_Number, sl_X_Overhead,
PSFCH_enabled, varargin)
```

where

- `type`: `'UL'`, `'DL'`, or `'SL'`. It indicates the communication direction. The object is created only when `type = 'SL'`,
- `BW`: BWP bandwidth in unit of MHz. The bandwidth options are specified in Table 5.3.2-1 of both [7] and [8],
- `RB_start`: start index of PRB,

- `nPRB`: number of PRBs in the sidelink BWP. `RB_start` + `nPRB` cannot exceed the maximum number of PRBs associated with `BW`,
- `mu`: numerology, which can be 0, 1, or 2 in Frequency Range 1 (FR1), and 2 or 3 in FR2.
- `cyclicPrefix`: cyclic prefix. Options can be `'normal'` or `'extended'`,
- `sl_LengthSymbols`: number of OFDM symbols for a sidelink slot. Options can be 7, 8, 9, 10, 11, 23, 13, or 14,
- `sl_StartSymbol`: index of start OFDM symbol. Options can be 0, 1, 2, 3, 4, 5, 6, or 7,
- `FR`: frequency range option, which can be `'FR1'` for FR1, or `'FR2'` for FR2.

As the remaining inputs are used in `SidelinkResourcePool` class for sidelink resource pool configuration, we will introduce them in the corresponding section.

7.2.2 Sidelink Resource Pool

`SidelinkResourcePool` class complies with the Information Element (IE) *SL-Resource Pool* in 3GPP [6], and its properties are tailored according to the need of the Sidelink LLS. It comprises of three major configuration categories:

1) PSCCH configuration: `SL_PSCCH_Config`, which defines time and frequency resources for PSCCH;

2) PSSCH configuration: `SL_PSSCH_Config`, which defines configurations of DM-RS time pattern, beta offset, and scaling factor for SCI2 resource, and

3) PSFCH configuration: `SL_PSFCH_Config`, which defines PSFCH transmission period, minimum gap of slots between the reception of a slot and the transmission of its PSFCH, and the Resource Block (RB) bitstring of the PSFCH OFDM symbol.

In addition, the enabling of PSFCH, the number of RBs in the resource pool, and the overhead from CSI-RS and PT-RS, are also defined, where whether PSFCH is configured is dependent on the enabling of PSFCH.

The constructor method is defined as:

```
1. function obj = SidelinkResourcePool(sl_bwp_obj,
   type, sl_TimeResourcePSCCH, sl_FreqResourcePSCCH,
   sl_PSSCH_DMRS_TimePatternList, sl_BetaOffsets2ndSCI, sl_Scaling,
   sl_RB_Number, sl_X_Overhead, PSFCH_enabled, varargin)
```

where

- `sl_bwp_obj`: `SL_BWP` object,
- `type`: `'UL'`, `'DL'`, or `'SL'`. It indicates the communication direction. The object is created only when `type = 'SL'`,

- `sl_TimeResourcePSCCH`: number of OFDM symbols for PSCCH. It can be 2 or 3,
- `sl_FreqResourcePSCCH`: number of PRBs for PSCCH. It can be 10, 12, 15, 20, or 25,
- `sl_PSSCH_DMRS_TimePatternList`: number of OFDM symbols containing PSSCH DM-RS. It can be 2, 3, or 4. It is important to note that in 3GPP [6], up to three time patterns can be defined in each resource pool, whereas in the Sidelink LLS, only ONE time pattern is used in the resource pool, which is configured in `SidelinkScenario` by

```
1. scStr.sidelink.sl_PSSCH_DMRS_TimePatternList = 2; % it can be
   2, 3, or 4
```

- `sl_BetaOffsets2ndSCI`: Beta offset value. It is important to note that it differs from 3GPP [6] and [9] in that the actual Beta_offset value in Table 9.3-2 of [9] is assigned to this property, whereas in the standards, the indexes of the Beta_offset values are defined [6], which map to the Beta_offset values in Table 9.3-2 of [9]. Besides, only one Beta_offset value is used in the Sidelink LLS, which is configurable, whereas in the standards [6], four indexes are specified.
- `sl_Scaling`: a scaling factor to limit the number of resource elements assigned to SCI2 on PSSCH, according to Sec. 8.4.4 of [10] and IE *SL-ResourcePool* of [6]. Its values can be 0.5, 0.65, 0.8, or 1,
- `sl_RB_Number`: the number of RBs assigned to the resource pool. It can be no larger than the `nPRB` in `sl_bwp_obj`,
- `sl_X_Overhead`: the number of overhead RBs from CSI-RS, PT-RS. The value can be 0, 3, 6, or 9.
- `PSFCH_enabled`: a boolean indicating whether PSFCH is enabled in the resource pool. It is important to note that `PSFCH_enabled` is not the enabling of PSFCH in a specific sidelink slot but rather in the resource pool.
- `varargin`: PSFCH configuration, which contains

  `sl_PSFCH_Period = varargin{1}`: PSFCH period in unit of slot, which can be 0, 1, 2, or 4,

  `sl_PSFCH_RB_Set = varargin{2}`: bitstring of RBs for PSFCH. Its size can be between 10 and 275. And

  `sl_MinTimeGapPSFCH = varargin{3}`: the minimum gap, in unit of slot, between reception of a TB and transmission of its feedback. Its value can be 2 or 3.

It is important to mention that as subchannel is not a focus to our link-level simulations, the subchannel-related configuration is not realized in the current release of the Sidelink LLS. The instantiation of the subchannel-dependent channels, such as PSCCH, PSSCH and PSFCH, are thus not constrained by the subchannel size or number. They are instead constrained by `sl_RB_Number`.

7.3 Sidelink Slot Format

The numbers of OFDM symbols and PRBs in each NR sidelink slot are reviewed in Sec. 7.1. In addition, each transmission can have up to two spatial-multiplexing layers. This section discusses the `SL_SlotStructure` class that implements the sidelink slot format, which takes into account symbols, PRBs, and layers.

The `SL_SlotStructure` class provides the following functions:

1) Initialize parameters related to sidelink slot configuration;

2) Specify the locations of each physical channel and reference signal at RE-level;

3) Display slot format for each layer; and

4) Map and demap the physical channels and reference signals.

The constructor method is defined as:

```
1. function obj = SL_SlotStructure(sl_bwp_obj, nLayer,
   PSFCH_overhead_indication)
```

where `sl_bwp_obj` is the object of the `SL_BWP` class that contains majority of the slot configuration related parameters, `nLayer` is the number of layers, and `PSFCH_overhead_indication` indicates if a sidelink slot contains PSFCH.

The `SL_SlotStructure` class uses a three dimensional matrix `sl_SlotStructure_Matrix` to store the locations of the physical channels and reference signals, the rows, columns, and pages of which represent the subcarriers, OFDM symbols, and layers, respectively. Each element of the matrix represents an RE and contains an integer labeling which physical channel or reference signal it is associated with. The mapping of the integers to physical channels/reference signals are specified in `elementDict`.

To map these integers, the following method is used:

```
1. function obj = update_SlotStructMatrix(obj, L_SCI2_mod_symbol)
```

where `L_SCI2_mod_symbol` is the number of REs for SCI2 in each layer.

In addition, `sl_SlotStructure_Matrix` is also used in the function `plot_SlotStructMatrix` to display each layer's slot format.

The mapping functions are listed as follows:

```
function obj = map_PSCCH_SCI1(obj, symbol_SCI1)

function obj = map_PSCCH_DMRS(obj, symbol_PSCCH_DMRS)

function obj = map_PSSCH(obj, symbol_PSSCH)    % symbol_PSSCH is the
modulation symbol sequence after data and control multiplexing

function obj = map_PSSCH_DMRS(obj, symbol_PSSCH_DMRS)

function obj = map_PHY_Channels(obj, symbol_SCI1, symbol_PSCCH_DMRS,
symbol_PSSCH, symbol_PSSCH_DMRS, varargin)

function obj = map_PSFCH(obj, symbol_PSFCH)
```

and the demapping functions are

```
function SCI1_symbols = demap_PSCCH_SCI1(obj, symbolMatrix)

function PSCCH_DMRS_symbols = demap_PSCCH_DMRS(obj, symbolMatrix)

function PSSCH_DMRS_symbols = demap_PSSCH_DMRS(obj, symbolMatrix)

function PSFCH_symbols = demap_PSFCH(obj, symbolMatrix)

function PSSCH_symbols = demap_PSSCH(obj, symbolMatrix)

function [SCI1_symbols, PSCCH_DMRS_symbols, PSSCH_symbols,
PSSCH_DMRS_symbols, PSFCH_symbols] = demap_PHY_Channels(obj,
symbolMatrix)
```

It is worth noting that the supported physical channels and reference signals can either be individually mapped/demapped, or they can be mapped/demapped all together using `map_PHY_Channels` for mapping or `demap_PHY_Channels` for demapping.

The `SL_SlotStructure` objects for TX and RX are initialized and updated in the `updateLink` function under `Link` class, and the mapping functions are implemented in the `sl_generateTransmitSignal` function under `User` class.

8. Sidelink Multiplexing and Channel Coding

In NR sidelink, the bit sequence of a TB or SCI2 is first channel coded and rate matched to the designated sequence length. After then, the data and SCI2 bit sequences are multiplexed into a single sequence, the procedures of which is so called data and control multiplexing. In this section, we first introduce the function we developed to determine the TB size, followed by channel coding and rate matching of data and SCI2. In the end, we introduce how data and control multiplexing and demultiplexing are implemented in the Sidelink LLS.

8.1 Sidelink TBS Generation

A sidelink TB should be generated before the Data and Control Multiplexing. Its Transport Block Size (TBS) is mainly determined by the number of REs allocated to the data in PSSCH (excluding the DM-RS, overhead, SCI1 and SCI2). The TBS in this LLS is calculated through the function `sl_TBGeneration`:

```
1. function [nCodedBits, TBS, N_RE_SCI2] = sl_TBGeneration(sl_bwp_obj,
   layerMapping, PSFCH_overhead_indication, R_SCI2, R_data, Q_data)
```

where `TBS` as an output is determined under a specific setup, `nCodedBits` is the number of data bits after channel coding and rate matching, and `N_RE_SCI2` is the number of allocated REs for SCI2.

Inside the function `sl_TBGeneration`, the size of each component (such as the number of REs for DM-RS, overhead, SCI1 and SCI2, etc) related to the TBS calculation is outputted by calling `sl_TBS_parameters.m`:

```
1. function [N_SC_RB, N_symb_sh, N_symb_PSFCH, N_oh_PRB, N_RE_DMRS,
   n_PRB, N_RE_SCI1, N_RE_SCI2] = sl_TBS_parameters(sl_bwp_obj,
   PSFCH_overhead_indication, R_SCI2)
```

where a value of 1 and 0 for `PSFCH_overhead_indication` indicates that 3 and 0 symbols are assigned to PSFCH, respectively. It should be noted that `beta_offset_SCI2` is used to scale the number of REs to balance reliability, which is specified by 3GPP in Table 9.3-2 of [9]).

Afterwards, given the number of REs assigned to the data after the above calculations, modulation order (for data), coding rate (for data) and layer mapping, `TBS` is obtained by following the procedures in TS38.214 5.1.3.2 [11].

8.2 Coding and Rate Matching

In NR sidelink, the bit sequence of data is first attached with 24-bit Cyclic Redundancy Check (CRC). Next, code block segmentation may be implemented, depending on the length of the bit sequence. Then, the bit sequence for a given code block is channel coded using Low Density Parity Check (LDPC). After that is rate matching, where the bit sequence input is matched to a certain length. Finally, different redundancy versions may be

implemented if HARQ is enabled.

To implement the LDPC coding/decoding and rate matching/recovery that support HARQ transmissions, the `nrDLSCH` and `nrDLSCHDecoder` functions from MATLAB 5G Toolbox is implemented. These functions are first initialized in `initializeLinks` under the `Link` class, with the object names of `encodeDLSCH` and `decodeDLSCH`, respectively. `encodeDLSCH` is implemented in `sl_generateTransmitSignal` for transmission and retransmissions, and `decodeDLSCH` is used in `sl_decodeProcess` for LDPC rate recovery and channel decoding.

The SCI2 bit sequence is also channel coded and rate matched before being multiplexed with the coded data sequence. It is first attached with a 24-bit CRC sequence, and then channel coded with Polar Coding. The length of the bit sequence after rate matching is determined by combining 8.4.4 of [10] with QPSK modulation for SCI2. In the Sidelink LLS, the parameters for SCI2 channel coder are configured in `SCI2ChannelCoder` by `initializeLinks`. The following functions from MATLAB 5G Toolbox are used:

- `nrCRCEncode` for CRC attachment,
- `nrPolarEncode` for Polar encoding,
- `nrRateMatchPolar` for rate matching,
- `nrRateRecoverPolar` for rate recovery,
- `nrPolarDecode` for Polar decoding, and
- `nrCRCDecode` for CRC checking.

`nrCRCEncode`, `nrPolarEncode`, and `nrRateMatchPolar` are implemented in `sl_generateTransmitSignal` at the transmitter, whereas `nrRateRecoverPolar`, `nrPolarDecode`, and `nrCRCDecode` are implemented in `sl_decodeProcess` at the receiver.

8.3 Data and Control Multiplexing

This section introduces the functions for data and control multiplexing and demultiplexing at the transmitter and receiver, respectively.

8.3.1 Data and Control Multiplexing

Data and control multiplexing is performed following the channel coding and rate matching of Sidelink Shared Channel (SL-SCH) and SCI2, and before the multiplexed sequence is scrambled. It multiplexes the coded data bits of SL-SCH and coded SCI2 bits to PSSCH, which comprises of a single sequence of bits. The format of the multiplexed sequence is determined by the number of layers for MIMO.

Following 3GPP [10], in this Sidelink LLS, data and control multiplexing is realized by the function `sl_multiplex`:

```
1. function multiplexedBits = sl_multiplex(codedDataBits,
   codedSCI2Bits, LayerMapping)
```

The coded SCI2 bits and coded data bits, `codedSCI2Bits` and `codedDataBits`, respectively, are multiplexed according to the number of layers for MIMO, `LayerMapping`. If

`codedSCI2Bits`: $b_0 b_1 ... b_{m-1}$, and

`codedDataBits`: $d_0 d_1 ... d_{n-1}$,

the multiplexed sequence, `multiplexedBits`, is as below:

when `LayerMapping` $= 1$,

`multiplexedBits`: $b_0 b_1 ... b_{m-1} d_0 d_1 ... d_{n-1}$,

and when `LayerMapping` $= 2$,

`multiplexedBits`: $b_0 b_1 -1 -1\, b_2 b_3 -1 -1 ... b_{m-2} b_{m-1} -1 -1\, d_0 d_1 ... d_{n-1}$,

where -1 replaces x in [10] as a placeholder bit.

8.3.2 Data and Control demultiplexing

As data and control demultiplexing is not specified in 3GPP standards, the process of demultiplexing at the receiver is realized by reverting the multiplexing process. This process follows the demodulation and descrambling. It takes the descrambled PSSCH bit sequence as input, and returns the bit sequences of coded SL-SCH data bits and coded SCI2 bits. This function `sl_demultiplex` is defined as:

```
1. function [demuxedSCI2Bits, demuxedDataBits] =
   sl_demultiplex(descrambledBits, L_sci2, LayerMapping)
```

The coded SCI2 bits from all layers (`demuxedSCI2Bits`) are demultiplexed and regrouped from the first `L_sci2` bits of the descrambled bit sequence (`descrambledBits`), according to the value of `LayerMapping`. `L_sci2` is the length of SCI2 bits from all layers and `LayerMapping` is the number of layers for MIMO. The coded data bits (`demuxedDataBits`) are the remaining bits of `descrambledBits` after the first `L_sci2` are excluded.

It is important to note that, as each layer contains one SCI2 bit sequence, `demuxedSCI2Bits` contains two coded SCI2 bit sequences in case `LayerMapping` $= 2$, one from each layer. These two sequences are expected to be identical when bit error is absent. To improve the performance of SCI2 decoding, in the Sidelink LLS, we deliver both sequences to the upper layer for decoding. Therefore, given

`descrambledBits`: $b_0 b_1 ... b_{L_{SCI2}-1} d_0 d_1 ... d_{n-1}$,

when `LayerMapping` $= 1$,

`demuxedSCI2Bits`: $b_0 b_1 ... b_{L_{SCI2}-1}$, and

`demuxedDataBits`: $d_0 d_1 ... d_{n-1}$;

when `LayerMapping` $= 2$,

`demuxedSCI2Bits`: $[b_0 b_1 b_4 b_5 ... b_{L_{SCI2}-4} b_{L_{SCI2}-3}$;
$b_2 b_3 b_6 b_7 ... b_{L_{SCI2}-2} b_{L_{SCI2}-1}]$, and

`demuxedDataBits`: $d_0 d_1 ... d_{n-1}$.

9. Physical Channels and Modulation

In this section, we introduce the processing chain at the transmitter after data and control multiplexing and before OFDM modulation. The procedures include data and SCI2 scrambling on the multiplexed bit sequence, data and SCI2 modulation that modulates the scrambled bit sequence into symbols, layer mapping and precoding for MIMO, and OFDM modulation and power control that creates and scales the baseband waveform. Accordingly, some of their counterparts at the receiver are discussed as well.

9.1 Data and SCI2 Scrambling

After data and control multiplexing, the multiplexed data and SCI2 sequence is scrambled with the scrambling sequence. As introduced in Sec. 8.3.1, the format of the multiplexed sequence can be different for different configured number of layers. Nonetheless, when scrambling is performed, each bit in b and d is scrambled with the corresponding bit in the scrambling sequence c:

$$\begin{aligned} \widetilde{b}_n &= (b_n + c_n) \bmod 2, \\ \widetilde{d}_n &= (d_n + c_n) \bmod 2. \end{aligned} \tag{1}$$

In a one-layer sequence, these scrambled sequences are concatenated in the same format as the multiplexed sequence before multiplexing:

`b_out:` $\widetilde{b}_0\widetilde{b}_1 \ldots \widetilde{b}_{m-1}\widetilde{d}_0\widetilde{d}_1 \ldots \widetilde{d}_{n-1}$.

In a two-layer sequence, however, each $[-1,-1]$ group in the multiplexed sequence is replaced by the group of two scrambled SCI2 bits before it:

`b_out:` $\widetilde{b}_0\widetilde{b}_1\widetilde{b}_0\widetilde{b}_1\widetilde{b}_2\widetilde{b}_3\widetilde{b}_2\widetilde{b}_3 \ldots \widetilde{b}_{m-2}\widetilde{b}_{m-1}\widetilde{b}_{m-2}\widetilde{b}_{m-1}\widetilde{d}_0\widetilde{d}_1 \ldots \widetilde{d}_{n-1}$.

In the Sidelink LLS, the above data and SCI2 scrambling procedure is realized via the `sl_scrambling` function:

```
1. function [b_out, c] = sl_scrambling(codedBits, L_sci2, L_data,
   p_CRC, LayerMapping)
```

where

- `b_out`: scrambled data and SCI2 sequence,
- `c`: scrambling sequence,
- `codedBits`: multiplexed sequence,
- `L_sci2`: total number of coded SCI2 bits of all layers,
- `L_data`: total number of coded data bits of all layers,
- `p_CRC`: 24-bit parity bits for the corresponding PSCCH,
- `LayerMapping`: number of layers.

The sl_scrambling function is implemented in the sl_generateTransmitSignal function under User class.

The scrambling sequence c is generated using the sl_scrambleSequence following Sec. 8.3.1.1 and 5.2.1 of [3]:

```
1. function c = sl_scrambleSequence(p_CRC, L_seq)
```

where L_seq is the length of the scrambling sequence. By implementing this function in the sl_scrambling function, the scrambling sequence is generated and saved in the Link class, which is later used by the sl_descrambling function.

At the receiver, the descrambling is performed after the received symbols are demodulated. It reverses the procedures for scrambling and descrambles the demodulated bits using the same scrambling sequence as the transmitter used:

```
1. function descrambledBits = sl_descrambling(demodulatedBits, L_sci2,
   L_data, scramblingSequence, LayerMapping)
```

where for each transmission, scramblingSequence has to be the same as the c in the sl_scrambling function.

For the one-layer case, the descrambled sequence is in the same format as the multiplexed Bits in Sec. 8.3.1. For two-layer case, however, the descrambled SCI2 bits for layer 2 are also saved for decoding, instead of being replaced by $[-1,-1]$ groups. It is important to note that, as the demodulated sequence is in the form of Log-Likelihood Ratio (LLR), the descrambling is also performed by converting the corresponding scrambling sequence to the LLR form.

The sl_descrambling function is implemented in the sl_processReceiveSignal function under User class.

9.2 Data and SCI2 Modulation

In addition to the modulation and demodulation of the coded data bits, the Sidelink LLS also supports those on the coded SCI2 bits.

The modulation of the coded data and SCI2 bits follows the scrambling operation at the transmitter, and their demodulation follows the layer demapping at the receiver.

In this Sidelink LLS, we use the modulation and demodulation modules provided by the Vienna LLS [2, 4]. To modulate the coded data bits, the SignalConstellation property is initialized when the Modulator object is initialized in the initializeLinks function, with the supported modulation scheme to be QPSK, 16 Quadrature Amplitude Modulation (QAM), 64QAM, or 256QAM [3]. The Modulation and Coding Scheme (MCS) is defined in SidelinkScenario:

```
1. % CQI selection

2. scStr.modulation.mcs = [15]; % 1 to 15, per link
```

the value of which indicates the Channel Quality Indication (CQI) index in the selected table of Table 5.2.2.1-2, 5.2.2.1-3, or 5.2.2.1-4 in [11].

In `sl_generateTransmitSignal`, the coded data bits, which are the $\widetilde{d}_0\widetilde{d}_1...\widetilde{d}_{n-1}$ of `b_out` in Sec. 9.1, are modulated using the `Bit2Symbol` function and the configuration in `SignalConstellation`.

To modulate the coded SCI2 bits, the `SCI2SignalConstellation` property is initialized in `sl_generateTransmitSignal` with 4QAM, which is equivalent to QPSK as defined by [3]. $\widetilde{b}_0\widetilde{b}_1...\widetilde{b}_{m-1}$ or $\widetilde{b}_0\widetilde{b}_1\widetilde{b}_0\widetilde{b}_1\widetilde{b}_2\widetilde{b}_3\widetilde{b}_2\widetilde{b}_3...\widetilde{b}_{m-2}\widetilde{b}_{m-1}\widetilde{b}_{m-2}\widetilde{b}_{m-1}$ in Sec. 9.1 is modulated using `Bit2Symbol` and the configuration in `SCI2SignalConstellation`. The resulting symbol sequence is therefore:
For one-layer case:

`mod_symbol:` $x^b_0x^b_1...x^b_{m/2-1}x^d_0...x^d_{n/Q-1}$,

where x^b and x^d are the modulation symbols for coded SCI2 and data bits, respectively, and Q is the modulation order for the coded data bits.
For two-layer case:

`mod_symbol:` $x^b_0x^b_0x^b_1x^b_1...x^b_{m/2-1}x^b_{m/2-1}x^d_0...x^d_{n/Q-1}$,

The demodulation of data and SCI2 symbols are implemented using the `LLR_AWGN` function under `SignalConstellation` and `SCI2SignalConstellation`, respectively, in `sl_detect` under `Link` class.

9.3 Sidelink Layer Mapping

NR sidelink supports spatial-multiplexing transmission. It follows the modulation of coded data and SCI2 bits and works on the modulation symbols. Per Sec. 8.3.1.3 of [3], sidelink supports layer mapping up to two layers. The layer mapped sequence x is:
For one-layer case,

x: $\left[x^b_0 \;\; x^b_1 \;\; ... \;\; x^b_{m/2-1} \;\; x^d_0 \;\; ... \;\; x^d_{n/Q-1}]\right]$

and for two-layer case:

$$x\text{:}\quad \begin{bmatrix} x^b_0 & x^b_1 & ... & x^b_{m/2-1} & x^d_0 & x^d_2 & ... & x^d_{n/Q-2} \\ x^b_0 & x^b_1 & ... & x^b_{m/2-1} & x^d_1 & x^d_3 & ... & x^d_{n/Q-1} \end{bmatrix}$$

The layer mapping is implemented in `sl_generateTransmitSignal` at the transmitter.

At the receiver, after channel equalization, the `sl_demapLayers` function under the `MIMO` class demaps the data and SCI2 symbols separately. This function is implemented in the `sl_getLLRs` function under the `Modulator` class.

9.4 Sidelink Precoding, OFDM Modulation, and Power Control

As specified in 8.3.1.4 of [3], the precoding matrix W is an identity matrix, where for one-layer transmissions,

$$W = 1, \tag{2}$$

and for two-layer transmissions,

$$W = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{3}$$

In the Sidelink LLS, the precoding matrix is set via parameter `scStr.modulation.precodingMatrix` in `/Scenarios/SidelinkScenario.m`.

```
1. % Per link, precoder selection.  Precoding matrix needs to be of
   size [nAntennas x nStreams]

2. scStr.modulation.precodingMatrix{1} = 1/sqrt(2)*eye(2); % Link 1
   employed precoding matrix.  For 1 link and 1-layer case, apply 1,
   and for 2-layer case, assign 1/sqrt(2)*eye(2)
```

where `1/sqrt(2)` is to guarantee the precoding matrix has unit power so that the signal power does not change.

According to power control requirement in 3GPP standard, the transmission power can be set as 23 dBm for power class 3 or 31 dBm for power class 1 with parameter `scStr.simulation.txPowerUser` in `/Scenarios/SidelinkScenario.m`.

```
1. scStr.simulation.txPowerUser = [23]; % per UE; UE total transmit
   power in dBm
```

Cyclic Prefix–Orthogonal Frequency Division Multiplexing (CP-OFDM) is the most prominent multicarrier scheme and is used as the access technology for 5G NR. The OFDM modulation in this Sidelink LLS is realized through the function `OFDM.m`. The `ofdm` object can be initialized by

```
1. OFDMobject = Modulation.OFDM(...
   L, ...  % number of subcarriers
   K, ...  % number of OFDM symbols in time
   F, ...  % subcarrier spacing (Hz)
   fs, ...  % sampling rate (sampless)
   fI, ...  % intermediate frequency of the 1st subcarrier (Hz)
   false, ...  % Transmit real valued signal, true/ false
   TCP,...  % Length of the cyclic prefix (s)
   TZG,...  % Length of the zero guard time (s),(frame)
   );
```

where we consider a block transmission of L subcarriers and K OFDM symbols. The intermediate frequency f_I corresponds to a circular shift of the Fast Fourier Transform (FFT).

A given symbol vector $\mathbf{x} \in \mathbf{C}^{L \times K}$, for example chosen from a QPSK signal constellation, can be then modulated by

```
1. s = OFDMobject.Modulation(x);
```

where `s` represents the transmitted signal in time. In the Sidelink LLS, the OFDM modulation is performed in `sl_generateTransmitSignal`

```
1. transmitSignalUnitPower(:,iAntenna) = primaryMod.WaveformObject.Mo
   dulation(modulationMat(:, :, iAntenna));
```

The demodulation at the receiver can be performed by applying the following method on the received time signal `r`

```
1. y = OFDMobject.Demodulation(r);
```

This function is called by the `demodulateTimeSignal` function under the `Modulator` class, and `demodulateTimeSignal` is implemented in the `sl_processReceiveSignal` function:

```
1. currentLink.sl_slot_struct_rx.sl_SlotStructure_Matrix_symbols =
   currentLink.Modulator.demodulateTimeSignal(totalSignal);
```

10. Channel Models, Channel Estimation and Equalization

This section discusses channel-related configurations and functions. We first introduce the propagation models developed and the corresponding path-loss-to-distance conversion function, which can be applied later in the data post-processing stage to calculate sidelink communication range. The small scale channel models are then discussed, followed by the channel estimation and equalization methods.

10.1 Channel Models

In this section, we introduce the propagation path loss models and the path-loss-to-distance conversion function. We also discuss the small scale channel models that are defined through the Power Delay Profile (PDP) files.

10.1.1 Propagation Path Loss Models

The channel models developed are the Device-to-Device (D2D) channel models defined in the 3GPP study for ProSe in LTE [5, Annex A.2.1.2]. The channel models consider three scenario settings, Outdoor-to-Outdoor (O2O), Outdoor-to-Indoor (O2I), and Indoor-to-Indoor (I2I).

We compute the theoretical path loss against distance. For O2O and O2I, we consider LOS and Non-Line-of-Sight (NLOS). For I2I, we consider both cases, within the same building and from different buildings. The path loss for each scenario are depicted in Fig. 5.

A set of utility functions were implemented in `/Pathloss_Utils/` to compute the expected path loss vs. distance for each channel model. General parameters are defined in `/Pathloss_Utils/generateChModelTables.m`. The default configurations are as below. Since sidelink communications is considered, the heights defined in LTE ProSe channel model [5, Annex A.2.1.2] for both transmitter and receiver are configured to 1.5 m. The center frequency is configured to be LTE Band 14 Uplink center frequency 793 MHz, i.e., from the Uplink frequency range [788, 798] MHz. In addition, a penetration loss of 40 dB is used for the I2I from different buildings. These default configurations could be modified per user needs.

Using the configurations, `generateChModelTables.m` generates path loss lookup tables from input [min., max.] distance range, at a desired distance increment step in meters, e.g., the range [10, 2500] m and step 0.5 m. Then, the `getDistanceFromPathloss` function finds the distance in meters from input parameters, the path loss in dB, and a channel model indicated from a string of characters, e.g., 'O2O_LOS' for O2O with LOS or 'O2O_NLOS' when NLOS. Liner interpolation is employed to compute the distance from the channel model path loss lookup tables. The results are used later to calculate and analyze the sidelink communication range.

The channel models developed and the configurations mentioned above are implemented in a modular way. They can be easily extended to other channel models and con-

figurations per user needs.

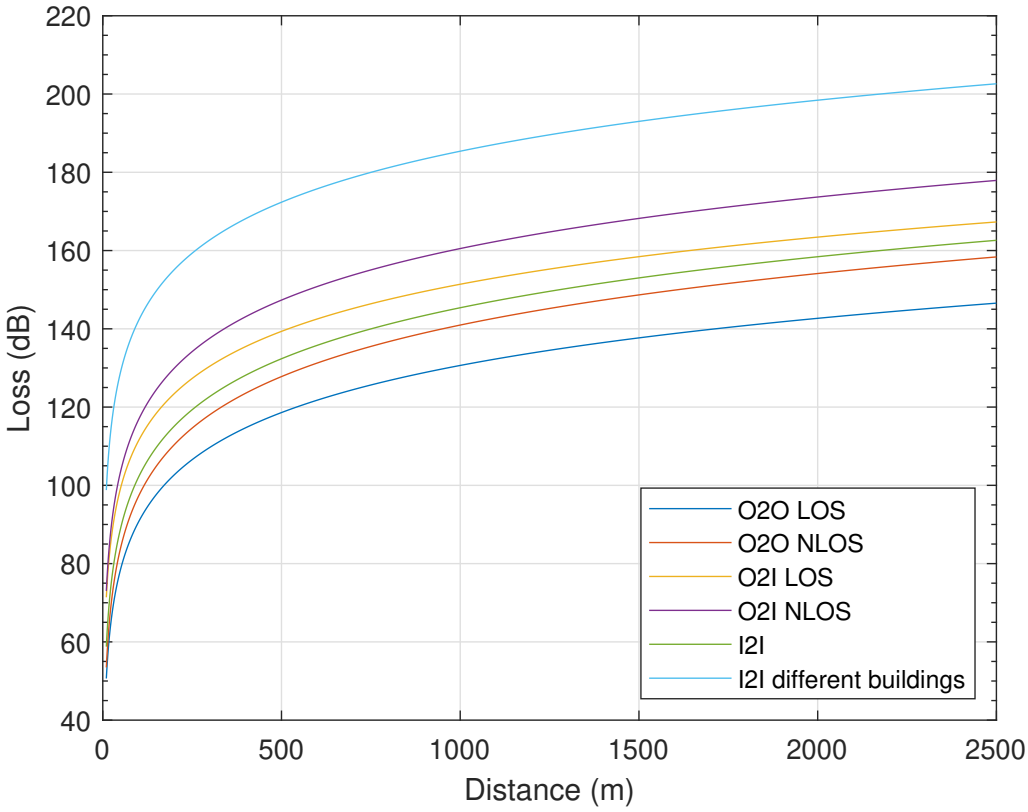

Fig. 5. Path loss propagation for D2D channel model considering different environments.

10.1.2 Small Scale Channel Models

The used small scale channel models are defined through PDPs. Descriptions of the available channel models can be found in the `SidelinkScenario` script, shown as the comments for the parameter `scStr.channel.powerDelayProfile`. For each iteration of `iFrame` in `main.m`, the `Newrealization` function under `FastFading` is called to update the link channel model as the result of small scale fading.

```
1. function NewRealization(obj, frame, frameStructure, direction)
```

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.

10.2 Channel Estimation and Equalization

The channel estimation method is called in `main.m` by `UE{iRx}.sl_processReceive Signal(UETotalSignal, Links, simParams)`. In the `User` class, the `sl_processRec`

eiveSignal function calls the channel estimation function by

```
1. [scheduledSCI2Channel, scheduledSCI2PseudoSymbols,
   scheduledDataChannel, scheduledDataPseudoSymbols] =
   currentLink.sl_estimateScheduledChannel;
```

Then the channel interpolation is used to estimate the channel model, either with Approximate-Perfect or PilotAided channel interpolation method in the sl_estimateScheduledChannel function under the Links class. In the original Vienna LLS [2], the pilot symbols in the PilotAided mode can be specified by users [4], whereas in NR sidelink, the channel estimation is implemented via DM-RS symbols. The Sidelink LLS channel estimation is developed by assuming perfect channel knowledge, and channel estimation using DM-RS will be developed in later releases.

Channel equalization is also called in main.m by UE{iRx}.sl_processReceiveSignal(UETotalSignal, Links, simParams). It calls the function sl_detect, where the channel equalization method is finally implemented in sl_getLLRs. Channel equalization methods such as zero-forcing (ZF), maximum-ratio-transmission (MRT), minimum mean square error (MMSE), and maximum-ratio-combining (MRC) are the options in the original Vienna downlink and uplink LLS [2, 4], and can be chosen in sidelink simulations to equalize the channel and recover the signal. The functions for ZF and MMSE are modified to fit the Sidelink LLS structure, named as sl_LLR_MIMO_ZF and sl_LLR_MIMO_MMSE, respectively. The channel equalization method can be set in /Scenarios/SidelinkScenario.m by:

```
1. scStr.simulation.receiverTypeMIMO = 'MMSE';
```

11. Blind-Based and Feedback-Based HARQ

Unlike 4G LTE sidelink which mandates blind-based HARQ with four transmissions, for unicast, 5G NR sidelink supports both blind-based HARQ and feedback-based HARQ, and the maximum number of transmissions is configurable and up to 32 per 3GPP [6]. Our Sidelink LLS supports both blind-based and feedback-based HARQ as well as a configurable number of transmissions, which can be configured in `SidelinkScenario` and are used later in the resource pool object:

```
1. scStr.sidelink.PSFCH_enabled = true;

2. scStr.sidelink.sl_PSFCH_Period = 4;

3. scStr.sidelink.sl_PSFCH_RB_Set = ones(1,scStr.sidelink.sl_RB_Number);

4. scStr.sidelink.sl_MinTimeGapPSFCH = 2;
```

Since the focus of the Sidelink LLS is one point-to-point link, HARQ feedback is not enabled/disabled at the TB level, but rather at the resource pool level. Accordingly, PSFCH configuration is realized in the resource pool object, including PSFCH enabled indicator `PSFCH_enabled`, PSFCH period `sl-PSFCH-Period` (1, 2, or 4 slots), and the time gap between the reception of a TB and the transmission of its PSFCH `sl-MinTimeGapPSFCH` (2 or 3 slots) .

In addition, given the nature of link-level simulation where scheduling algorithm is not the focus, the Sidelink LLS implements a simple resource allocation mechanism. For blind-based HARQ, all transmissions of one TB are transmitted in consecutive slots. For feedback-based HARQ, since the number of symbols available to PSSCH differs between a slot with PSFCH and a slot without PSFCH (Figure 3 and 4), the sizes of the TBs held in these two types of slots are different. In the Sidelink LLS, if the initial transmission of one TB is on a slot with PSFCH, all its retransmissions are scheduled in slots with PSFCH, and vice versa.

Furthermore, a sidelink HARQ entity `sidelinkHARQEntity` is developed to manage HARQ processes and redundancy versions per 3GPP [10] and [11], together with the simplified resource allocation mechanism described above. It can be found under the `/SL_HARQ/` directory. Figure 6 illustrates its work flow. As shown in the figure, the active HARQ process is selected based on whether its associated slot type (with or without PSFCH) matches the current slot and whether the minimum time gap between its associated PSSCH and PSFCH is satisfied. If both requirements are met, the process that waited longer in the buffer is selected. After the active HARQ process is selected, the next redundancy version of the associated TB is transmitted.

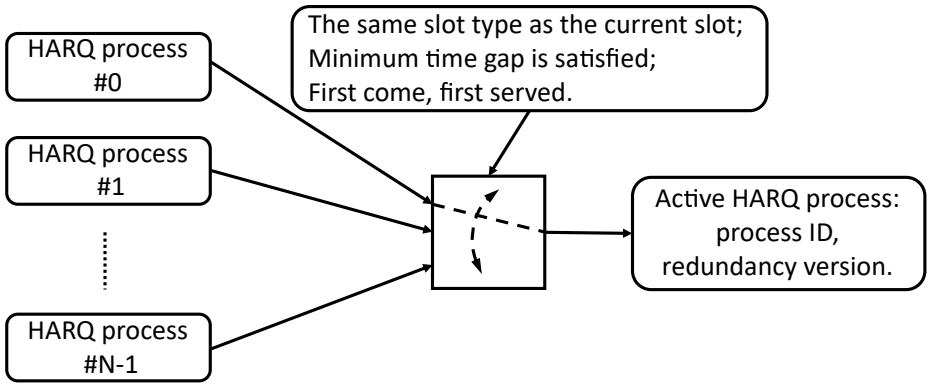

Fig. 6. HARQ Management.

12. Error-Free and Error-Prone SCI2

To provide flexible options for researchers, the Sidelink LLS provides 1) error-free SCI2, which separates the data collection and processing of SCI2 from decoding of a TB, and 2) error-prone SCI2, where a TB cannot be decoded properly if the associated SCI2 is received with error.

In the error-free SCI2 scenario, the decoding of a data transmission is executed no matter if the corresponding SCI2 decoding reports an error. The data transmission in each slot is decoded, and the data BLER is calculated by summarizing the decoding results of each TB transmission(s). The decoding of SCI2 transmission is recorded separately, and its BLER is calculated by summarizing the decoding result of SCI2 in each slot. To enable error-free SCI2, the following parameter in the `SidelinkScenario` script needs to be set to `false`:

```
1. %% SCI2 error prone?

2. scStr.sidelink.SCI2ErrorProne = false;
```

By setting the above parameter to `true`, the error-prone SCI2 scenario is enabled. In this scenario, if the decoding of SCI2 in a slot returns an error, the decoding of the corresponding TB transmission is no longer executed and the decoding result is automatically recorded as erroneous. In this scenario, the decoding error of SCI2 transmissions is still recorded for each slot, whereas the decoding error of TB transmission(s) reflects the error from both SCI2 and TB transmissions. The above operation is implemented in the `sl_decodeProcess` under the `Link` class.

13. Release and Changelog

1. 5G New Radio Sidelink Link-Level Simulator 1.0, released October 2022.

Acknowledgments

The authors thank Tom Henderson (University of Washington) for insightful technical discussions and guidance throughout the project, and Jean-Aicard Fabien (National Institute of Standards and Technology) for providing timely support on 3GPP specifications and their developments.

References

[1] The MathWorks Inc (2022) 5G Toolbox User's Guide. Accessed: 2022-06-18 Available at https://www.mathworks.com/help/pdf_doc/5g/5g_ug.pdf.

[2] Pratschner S, Tahir B, Marijanovic L, Mussbah M, Kirev K, Nissel R, Schwarz S, Rupp M (2018) Versatile mobile communications simulation: the Vienna 5G Link Level Simulator. *EURASIP Journal on Wireless Communications and Networking* 2018(1):226. https://doi.org/10.1186/s13638-018-1239-6

[3] 3GPP (2021) NR; Physical channels and modulation (Release 16) (3rd Generation Partnership Project (3GPP)), Technical Specification (TS) 38.211. Version 16.6.0.

[4] Pratschner S, Tahir B, Nissel R, Marijanovic L, Mussbah M, Kirev K, Schwarz S, Rupp M (2020) *The Vienna 5G Link Level Simulator v1.2* Institute of Telecommunications, TU Wien, . Available at https://www.tuwien.at/etit/tc/wp-content/uploads/2022/06/5gll-usermanual.pdf.

[5] 3GPP (2014) Study on LTE device to device proximity services; Radio aspects (Release 12) (3rd Generation Partnership Project (3GPP)), Technical Report (TR) 36.843. Version 12.0.1 Available at https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2544.

[6] 3GPP TS38331 (2021) 3rd Generation Partnership Project;Technical Specification Group Radio Access Network;NR;Radio Resource Control (RRC) protocol specification (3GPP), Standard.

[7] 3GPP TS38101-1 (2021) 3rd Generation Partnership Project;Technical Specification Group Radio Access Network;NR;User Equipment (UE) radio transmission and reception; Part 1: Range 1 Standalone (3GPP), Standard.

[8] 3GPP TS38101-2 (2021) 3rd Generation Partnership Project;Technical Specification Group Radio Access Network;NR;User Equipment (UE) radio transmission and reception; Part 2: Range 2 Standalone (3GPP), Standard.

[9] 3GPP TS38213 (2021) 3rd Generation Partnership Project;Technical Specification Group Radio Access Network;NR;Physical layer procedures for control (3GPP), Standard.

[10] 3GPP TS38212 (2021) 3rd Generation Partnership Project;Technical Specification Group Radio Access Network;NR;Multiplexing and channel coding (3GPP), Standard.

[11] 3GPP (2020) NR; Physical layer procedures for data (Release 16) (3rd Generation Partnership Project (3GPP)), Technical Specification (TS) 38.214.

Version 16.3.0 Available at https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3216.