Proximity Services (ProSe) Support for 5G NR Simulations

WNS3 2023 – June 2023

Aziza Ben Mosbah[1,2] and Samantha Gamboa[1,2]

[1] Associate, Wireless Networks Division (WND) - National Institute of Standards and Technology (NIST) - Gaithersburg, Maryland, USA

[2] Prometheus Computing LLC - Bethesda, Maryland, USA

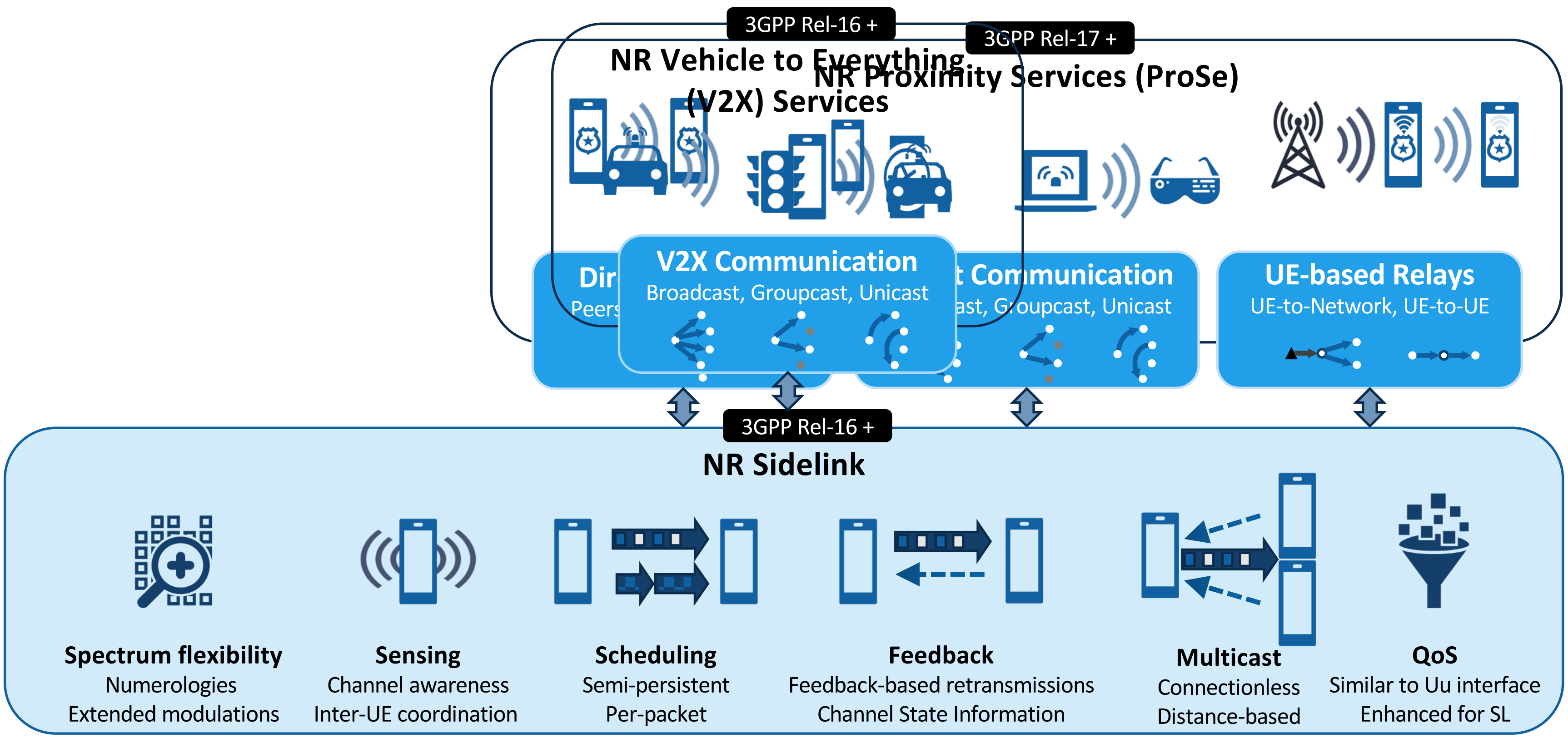
Disclaimer

Certain software is identified in this presentation in order to visualize simulation outputs adequately. Such identification is not intended to imply recommendation or endorsement of any product or service by NIST, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

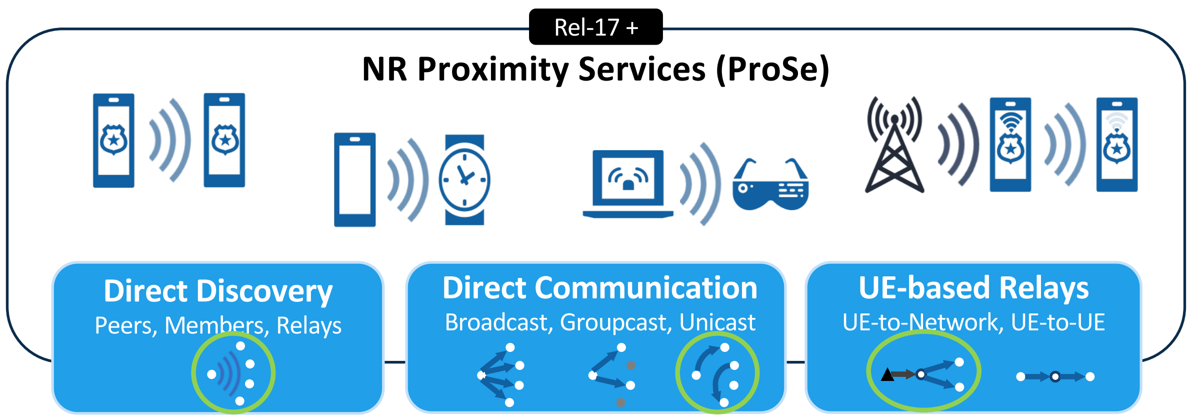
Outline

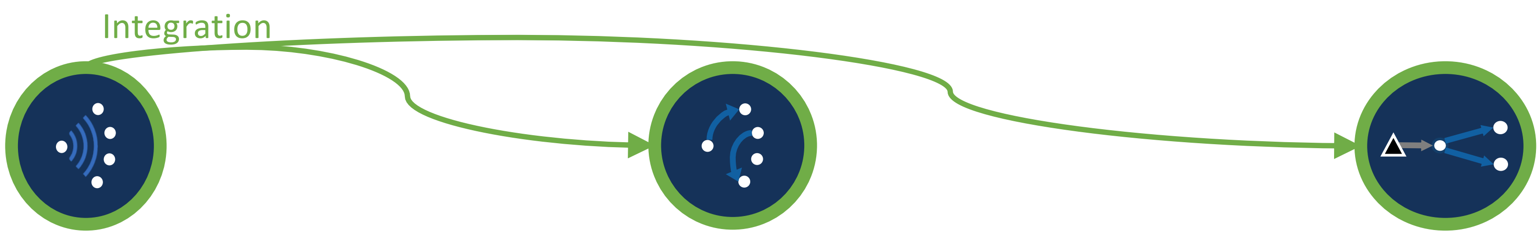
- Motivation
- Current functionalities
- Model overview
- Code availability and setup
- NR ProSe direct discovery
  - Overview
  - Example -> nr-prose-discovery.cc
  - Example -> nr-prose-discovery-l3-relay.cc
- NR ProSe unicast communication
  - Overview
  - Example -> nr-prose-unicast-multi-link.cc
- NR ProSe L3 UE-to-Network relay
  - Overview
  - Example -> nr-prose-l3-relay.cc
- NR ProSe L3 U2N Relay selection (Integration direct discovery with L3 U2N relay)
  - Example -> nr-prose-discovery-l3-relay-selection.cc

Motivation


3GPP Rel-16 +
NR Vehicle to Everything (V2X) Services
3GPP Rel-17 +
NR Proximity Services (ProSe)
V2X Communication
Broadcast, Groupcast, Unicast
Dir
Peers
t Communication
ast, Groupcast, Unicast
UE-based Relays
UE-to-Network, UE-to-UE
3GPP Rel-16 +
NR Sidelink
Spectrum flexibility
Numerologies
Extended modulations
Sensing
Channel awareness
Inter-UE coordination
Scheduling
Semi-persistent
Per-packet
Feedback
Feedback-based retransmissions
Channel State Information
Multicast
Connectionless
Distance-based
QoS
Similar to Uu interface
Enhanced for SL

Current functionalities





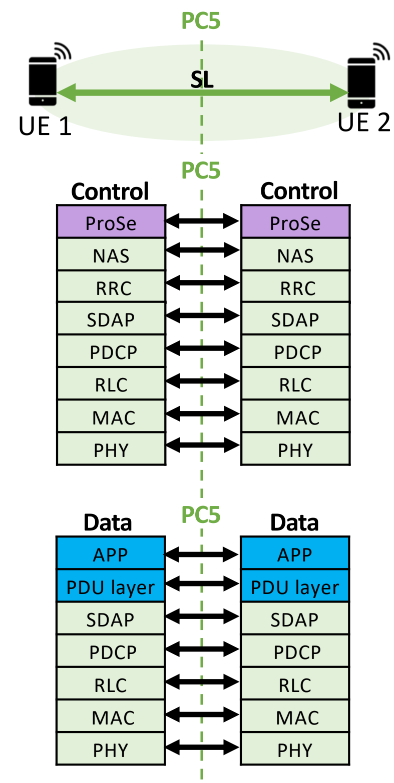
NR ProSe direct discovery
- Direct peer discovery
- UE-to-Network relay discovery
- Model A and Model B

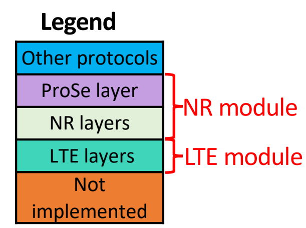
NR ProSe unicast communication
- Direct link establishment
- Traffic exchange between peer UEs
- Multiple direct links with different peers

NR ProSe L3 UE-to-Network relay
- Remote UE to Relay UE connection
- Layer 3 traffic relaying
- Remote UE network transparency

Model overview - ProSe over NR SL

NR ProSe Protocol Stacks

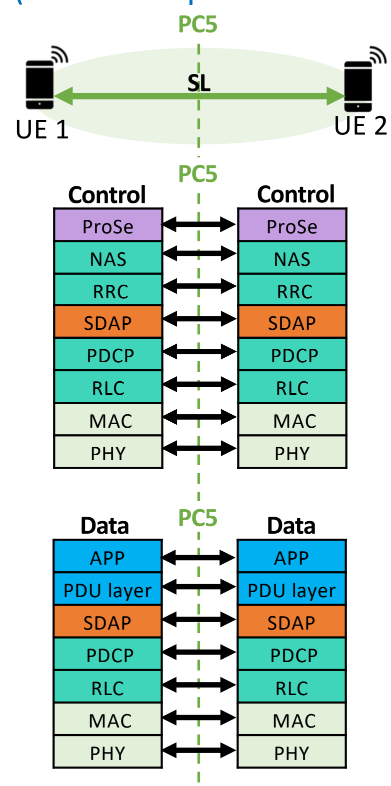




The ProSe model is based on
- CTTC's [NR V2X release v0.1]
and
- ns3 [ns-3.36]

We implemented the ProSe layer and modified other layers to support the ProSe functionalities

NR ProSe Protocol Stacks (current implementation)



- APP: Application
- MAC: Media Access Control
- NAS: Non-Access Stratum
- PDU: Protocol Data Unit
- PDCP: Packet Data Convergence Protocol
- PDU: Protocol Data Unit
- PHY: Physical Layer
- ProSe: Proximity Services
- RLC: Radio Link Control
- RRC: Radio Resource Control
- SL: Sidelink
- SDAP: Service Data Adaptation Protocol
- UE: User equipment

Model overview - UE Control Plane Architecture

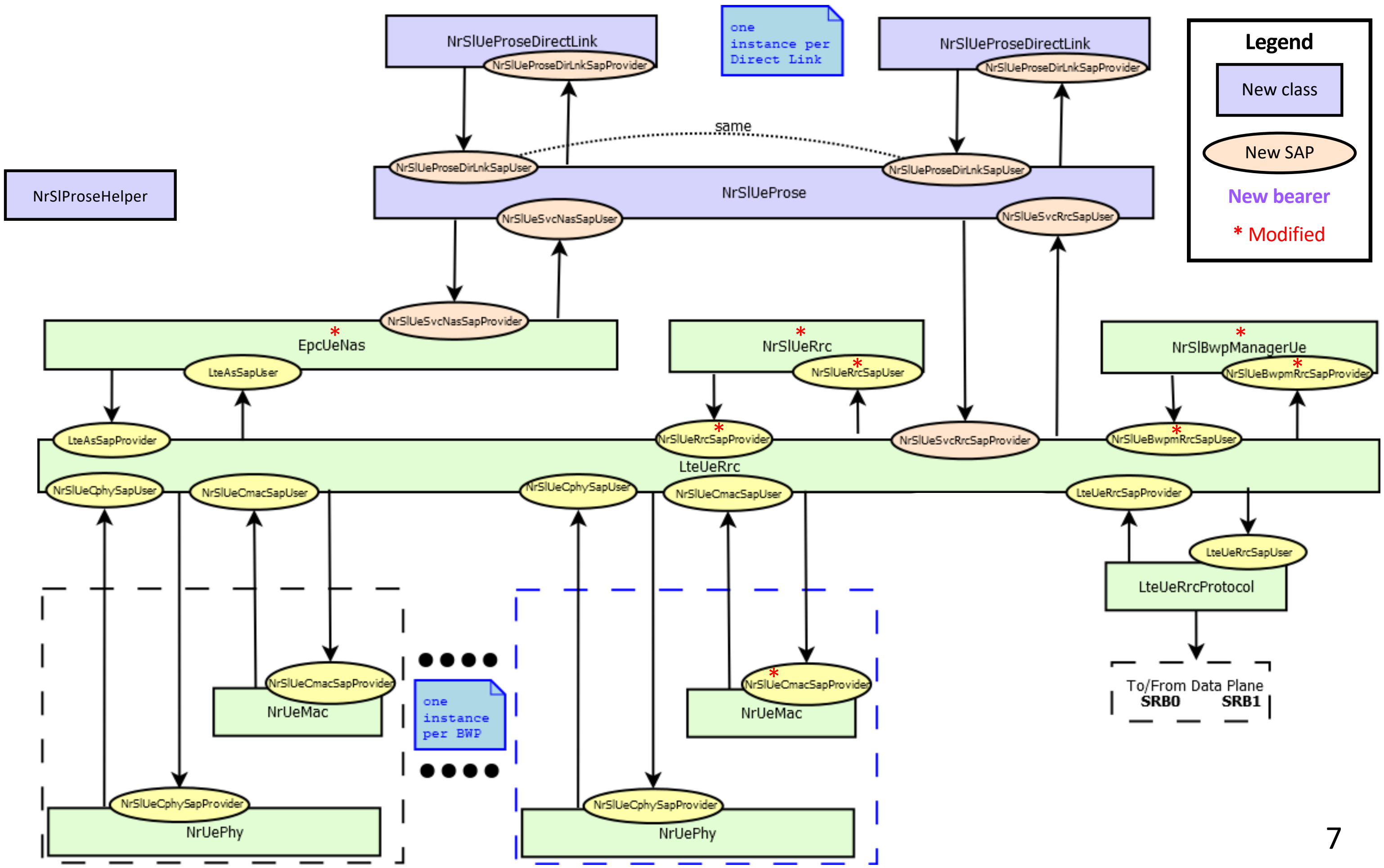
NR ProSe simulation typical flow:

<u>During scenario configuration:</u>

- Configure ProSe functionalities using ProSe helper
- Internally the helper installs ProSe layer and configures functionality

<u>During simulation:</u>

- ProSe layer controls functionalities
  - Context creation and management
  - SL bearer creation/configuration
  - Procedures execution



Code availability and setup

How to get started:

1. Get the code and move into psc-ns3 directory:

git clone "https://github.com/usnistgov/psc-ns3.git" -b wns3-2023-nr-prose-preview

cd psc-ns3

2. Setup ns3:

./ns3 configure --enable-examples

./ns3

3. Run examples:

- Running a ProSe example:

./ns3 run ‘exampleName’

e.g.,
```
-bash-4.2$ ./ns3 run 'nr-prose-discovery-l3-relay'
```

- Running a a ProSe example with command line parameters:

./ns3 run ‘exampleName --param1=param1Value --param2=param2Value’

e.g.,
```
bash-4.2$ ./ns3 run 'nr-prose-discovery-l3-relay --discInterval=4'
```

- Running a a ProSe example with command line parameters and specify (existent) output directory:

./ns3 run ‘exampleName --param1=param1Value --param2=param2Value’ --cwd=‘outputDirectory’

e.g.,
```
bash-4.2$ ./ns3 run 'nr-prose-discovery-l3-relay --discInterval=4' --cwd='output_nr-prose-discovery-l3-relay-4s'
```

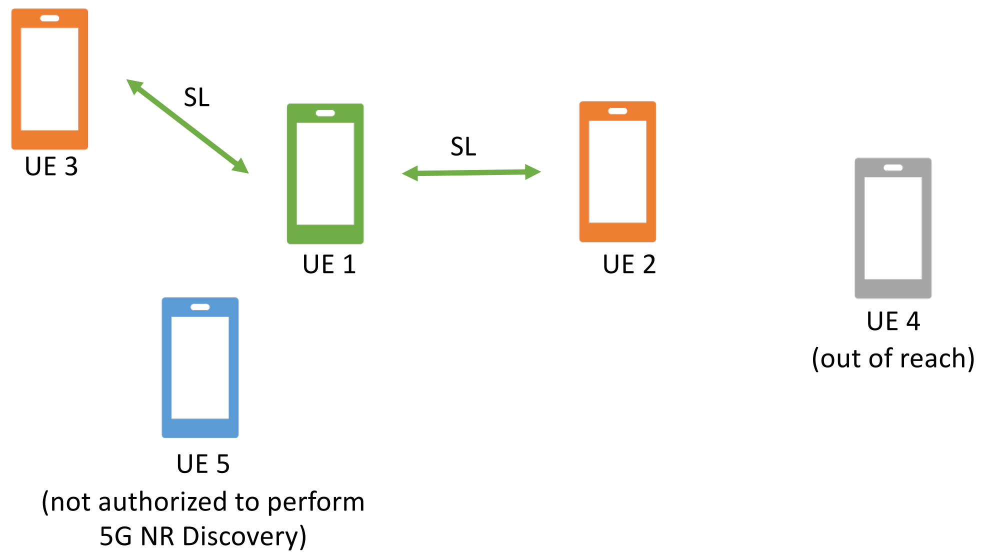
NR ProSe example’s source code location:

src/nr/examples/nr-prose-examples/

```
-bash-4.2$ ls -1 src/nr/examples/nr-prose-examples/
nr-prose-discovery.cc
nr-prose-discovery-l3-relay.cc
nr-prose-discovery-l3-relay-selection.cc
nr-prose-l3-relay.cc
nr-prose-l3-relay-on-off.cc
nr-prose-network-coex.cc
nr-prose-unicast-l3-relay.cc
nr-prose-unicast-multi-link.cc
nr-prose-unicast-single-link.cc
```

NR ProSe direct discovery

NR ProSe direct discovery: Overview



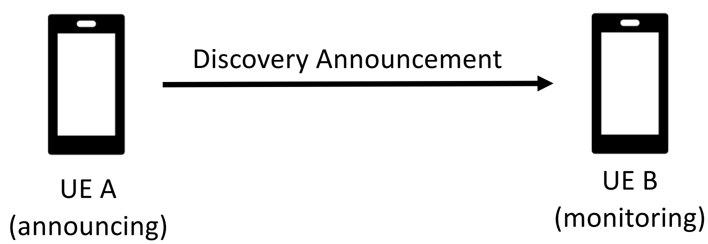
- 5G ProSe direct discovery allows 5G ProSe-enabled UEs discover other 5G ProSe-enabled UEs within their reach using direct NR radio transmissions. It can be performed independently from 5G ProSe Direct Communication or can be used to initiate one-to-one unicast communication.
- Direct discovery can be either open or restricted depending on whether an explicit permission from the 5G ProSe-enabled UE being discovered is needed.

NR ProSe direct discovery: Models

Model A

Model A is a discovery announcement using a **broadcast of a single discovery message** and can be either **open or restricted**.

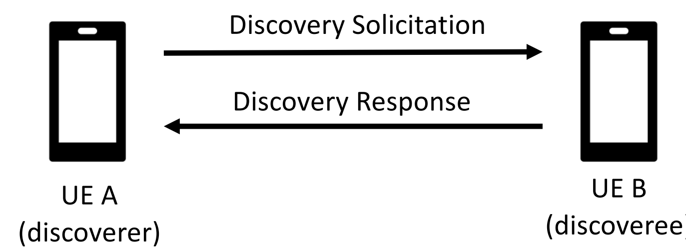
The UE sending the ProSe PC5 discovery message is called the "**announcing** UE" and the "**monitoring** UE" is the UE that triggers the lower layer to start listening for such message.



Model B

Model B employs a set of discovery messages based on a **Request/Response exchange** and can only be **restricted**.

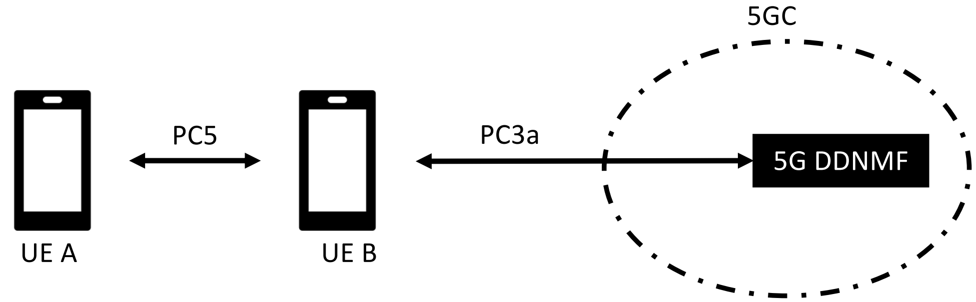
The UE sending the first discovery message is called the "**Discoverer** UE" and the UE receiving and responding to this message is called the "**Discoveree** UE".



NR ProSe direct discovery: Interfaces

PC3a:
The reference point between the UE and the 5G Direct Discovery Name Management Function (5G DDNMF), which is the logical function managing inter-PLMN 5G ProSe Direct Discovery operations.
It is used to authorize 5G ProSe Direct Discovery request and perform allocation of ProSe Application Codes / ProSe Restricted Codes corresponding to ProSe Application Identities used for 5G ProSe Direct Discovery.

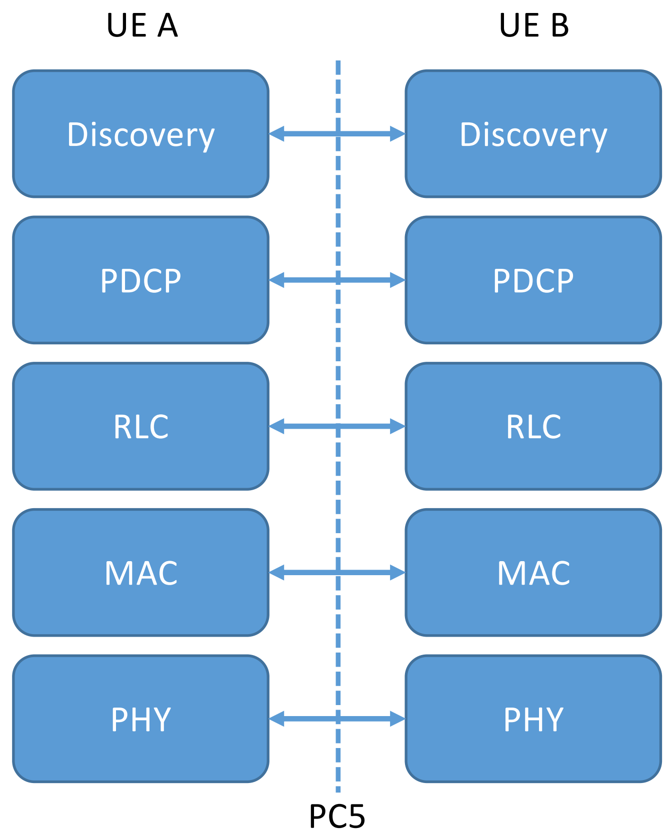
PC5:
The reference point between ProSe-enabled UEs used for control and user plane for 5G ProSe Direct Discovery, 5G ProSe Direct Communication and 5G ProSe UE-to-Network Relay.



NR ProSe direct discovery: PC5 Procedures

PC5 discovery supports the initiation and completion of the following PC5 procedures for both models A and B:

- **Direct or peer discovery** (for open and restricted modes)
    - to enable a ProSe-enabled UE to detect and identify another ProSe-enabled UE over PC5 interface.
- **Group member discovery** (only available for restricted mode) both public safety use and commercial services
    - to enable a ProSe-enabled UE to detect and identify another ProSe-enabled UE that belongs to the same application layer group (e.g. sharing the same application layer group ID) over PC5 interface.
- **UE-to-Network relay discovery** (only available for restricted mode)
    - to enable a ProSe-enabled UE to detect and identify another ProSe-enabled UE over PC5 interface for UE-to-Network relay communication between a UE and 5G Core (5GC).



NR ProSe direct discovery: Implementation

PC3a discovery is not supported

- UEs are considered pre-authorized to perform PC5 discovery.
- The required parameters (e.g., ProSe Application Code, Relay Service Code, filters, etc) are already provided in the scenario to use during the PC5 discovery process.

PC5 discovery is implemented

- Peer and U2N relay discovery are implemented with both discovery models (Model A and Model B).
- Group discovery is partially supported and be simulated using one Destination L2 ID for a group of UEs.
- A discovery transmission periodicity is added, i.e., the discovery message is sent periodically based on discovery interval that can be set in the scenario (default value is equal to 1 second).

NR ProSe direct discovery: Implementation

```
///< Information for application discovery
struct DiscoveryInfo
{
  DiscoveryModel model; ///< discovery model used
  DiscoveryRole role; ///< role in the discovery
  uint32_t appCode; ///< application code
  uint32_t dstL2Id; ///< destination L2 ID
};
```

Structure to store discovery information:
- Discovery model
- Discovery role
- The application code
- The Layer 2 ID of the target destination

```
///< The discovery models supported
enum DiscoveryModel
{
  ModelA = 0,     ///< announce
  ModelB          ///< request/response
};
```

Two discovery models:
- Model A for broadcast announcements
- Model B for request/response exchange

```
///< The types of discovery role
enum DiscoveryRole
{
  Monitoring = 0, ///< Model A: The UE receiving discovery messages
  Announcing,     ///< Model A: The UE sending discovery messages
  Discoveree,     ///< Model B: The UE responding to requests
  Discoverer,     ///< Model B: The UE sending requests
  RemoteUE,
  RelayUE
};
```

The UE can play multiple roles depending the discovery model used and on which side (transmitting or receiving) it is.

NR ProSe direct discovery: Peer Discovery

These functions from the *NrSlProseHelper* can be called in the scenario when performing peer discovery, allowing the start and the end of the discovery process and taking into consideration discovery parameters (e.g., ProSe Application Code, Destination L2 ID, and the role played by the UE).

```
/**
 * Starts discovery process for given applications depending on the interest (monitoring or announcing)
 * \param ueDevice the targeted device
 * \param appCodes application code to be added
 * \param dstL2Ids destination layer 2 IDs to be set for each appCode
 * \param role UE role (discovered or discoveree)
 */
void StartDiscovery (Ptr<NetDevice> ueDevice, std::list<uint32_t> appCodes, std::list<uint32_t> dstL2Ids, NrSlUeProse::DiscoveryRole role);

/**
 * Stops discovery process for given applications
 * \param ueDevice the targeted device
 * \param appCodes application codes to be removed
 * \param role UE role (discovered or discoveree)
 */
void StopDiscovery (Ptr<NetDevice> ueDevice, std::list<uint32_t> appCodes, NrSlUeProse::DiscoveryRole role);
```

NR ProSe direct discovery: Peer Discovery - Example

nr-prose-discovery.cc

Source code: src/nr/examples/nr-prose-examples/nr-prose-discovery.cc

```
// Application Codes to announce/monitor
std::map<Ptr<NetDevice>, std::list<uint32_t> > announcePayloads;
std::map<Ptr<NetDevice>, std::list<uint32_t> > monitorPayloads;

// Destination L2 IDs for each application code
std::map<Ptr<NetDevice>, std::list<uint32_t> > announceDstL2IdsMap;
std::map<Ptr<NetDevice>, std::list<uint32_t> > monitorDstL2IdsMap;

for (uint32_t i = 1; i <= ueVoiceNetDev.GetN (); ++i)
  {
    //For each UE, announce one appCode and monitor all the others appCode
    announcePayloads[ueVoiceNetDev.Get (i - 1)].push_back (i);
    announceDstL2IdsMap[ueVoiceNetDev.Get (i - 1)].push_back (100*i);

    for (uint32_t j = 1; j <= ueVoiceNetDev.GetN (); ++j)
      {
        if (i != j)
          {
            monitorPayloads[ueVoiceNetDev.Get (i - 1)].push_back (j);
            monitorDstL2IdsMap[ueVoiceNetDev.Get (i - 1)].push_back (100*j);
          }
      }
  }

for (uint32_t i = 0; i < ueVoiceNetDev.GetN (); ++i)
  {
    //Start announcing/monitoring
    Simulator::Schedule (startDiscTime, &NrSlProseHelper::StartDiscovery, nrSlProseHelper, ueVoiceNetDev.Get (i),
        announcePayloads[ueVoiceNetDev.Get (i)], announceDstL2IdsMap[ueVoiceNetDev.Get (i)], NrSlUeProse::Announcing);
    Simulator::Schedule (startDiscTime, &NrSlProseHelper::StartDiscovery, nrSlProseHelper, ueVoiceNetDev.Get (i),
        monitorPayloads[ueVoiceNetDev.Get (i)], monitorDstL2IdsMap[ueVoiceNetDev.Get (i)], NrSlUeProse::Monitoring);

    //Stop announcing
    Simulator::Schedule (stopDiscTime,&NrSlProseHelper::StopDiscovery, nrSlProseHelper, ueVoiceNetDev.Get (i),
                         announcePayloads[ueVoiceNetDev.Get (i)], NrSlUeProse::Announcing);
    //Stop monitoring
    Simulator::Schedule (stopDiscTime,&NrSlProseHelper::StopDiscovery, nrSlProseHelper, ueVoiceNetDev.Get (i),
                         monitorPayloads[ueVoiceNetDev.Get (i)], NrSlUeProse::Monitoring);
  }
```

Each UE is announcing its own service.
All UEs are monitoring other UEs' services.

Model A is used and the role each UE plays in the discovery process is specified in the scenario.
If instead of Announcing/Monitoring, Discoverer/Discoveree was used, then Model B is considered.

NR ProSe direct discovery: Peer Discovery - Example

nr-prose-discovery.cc

Running the scenario:

```
-bash-4.2$ mkdir output_nr-prose-discovery
-bash-4.2$ ./ns3 run 'nr-prose-discovery' --cwd='output_nr-prose-discovery'
```

Simulation output files:

```
-bash-4.2$ ls -1 output_nr-prose-discovery/
default-nr-prose-discovery.db
NrSlDiscoveryTrace.txt
```

The *NrSlDiscoveryTrace.txt* trace file is generated automatically when discovery is performed. As specified in the example, Model A is used, and the discovery is performed every 2 seconds (default value in the example) until the end of the simulation (10 seconds).

```
bash-4.2$ cat output_nr-prose-discovery/NrSlDiscoveryTrace.txt
Time (s)        TX/RX   senderL2Id      receiverL2Id    DiscType        DiscModel       ContentType     Content
2.0000000000    TX      1       100     Open    ModelA  Announcement    1
2.0000000000    TX      2       200     Open    ModelA  Announcement    2
2.0070821400    RX      1       2       Open    ModelA  Announcement    1
2.0075821400    RX      2       1       Open    ModelA  Announcement    2
4.0000000000    TX      1       100     Open    ModelA  Announcement    1
4.0000000000    TX      2       200     Open    ModelA  Announcement    2
4.0065821400    RX      2       1       Open    ModelA  Announcement    2
6.0000000000    TX      1       100     Open    ModelA  Announcement    1
6.0000000000    TX      2       200     Open    ModelA  Announcement    2
6.0020821400    RX      2       1       Open    ModelA  Announcement    2
6.1018321400    RX      1       2       Open    ModelA  Announcement    1
8.0000000000    TX      1       100     Open    ModelA  Announcement    1
8.0000000000    TX      2       200     Open    ModelA  Announcement    2
8.0025821400    RX      1       2       Open    ModelA  Announcement    1
```

Successful mutual discovery: Both UEs were able to discover each other.

UE 1 succeeded to discover UE 2, while UE 2 was not able to discover UE 1

NR ProSe direct discovery: Relay Discovery

These functions from the *NrSlProseHelper* can be called in the scenario when performing relay discovery, allowing the start and the end of the relay discovery process and taking into consideration discovery parameters (e.g., Relay Service Code, Destination L2 ID, discovery model, and the role played by the UE).

```
/**
 * Starts relay discovery process depending on the interest (relay or remote)
 * \param ueDevice the targeted device
 * \param relayCode relay code
 * \param dstL2Ids destination layer 2 ID
 * \param model UE model (A or B)
 * \param role UE role (relay or remote)
 */
void StartRelayDiscovery (Ptr<NetDevice> ueDevice, uint32_t relayCode, uint32_t dstL2Id, NrSlUeProse::DiscoveryModel model, NrSlUeProse::DiscoveryRole role);

/**
 * Stops relay discovery process for given code
 * \param ueDevice the targeted device
 * \param relayCode relay code to be removed
 * \param role UE role (relay or remote)
 */
void StopRelayDiscovery (Ptr<NetDevice> ueDevice, uint32_t relayCode, NrSlUeProse::DiscoveryRole role);
```

NR ProSe direct discovery: Relay Discovery - Example

nr-prose-discovery-l3-relay.cc

Source code: src/nr/examples/nr-prose-examples/nr-prose-discovery-l3-relay.cc

```
/*
 * Setup discovery applications
 */
NS_LOG_INFO ("Configuring discovery relay");

//Relay Discovery
uint32_t relayCode = 5;
uint32_t relayDstL2Id = 500;

Simulator::Schedule (startDiscTime, &NrSlProseHelper::StartRelayDiscovery, nrSlProseHelper,
ueVoiceNetDev.Get (0), relayCode, relayDstL2Id, NrSlUeProse::ModelB, NrSlUeProse::RelayUE);
Simulator::Schedule (startDiscTime, &NrSlProseHelper::StartRelayDiscovery, nrSlProseHelper,
ueVoiceNetDev.Get (1), relayCode, relayDstL2Id, NrSlUeProse::ModelB, NrSlUeProse::RemoteUE);
```

- 2 UEs are placed within reach from each other and performing relay discovery.
- To establish the relay discovery, we:
  - define the pre-requisite parameters: the Relay Service Code and the Layer-2 ID of the target destination.
  - schedule the start of the discovery process and specify the Relay Service Code, the destination L2 ID, the used discovery model (Model B here), and the role for each UE (relay/remote here).
- The Relay Service code, the destination L2 ID, and the discovery model should be all configured the same for both UEs. And the discovery roles should align with the discovery procedure considered.

NR ProSe direct discovery: Relay Discovery - Example

nr-prose-discovery-l3-relay.cc

Running the scenario:

```
-bash-4.2$ mkdir output_nr-prose-discovery-l3-relay
-bash-4.2$ ./ns3 run 'nr-prose-discovery-l3-relay' --cwd='output_nr-prose-discovery-l3-relay'
```

Simulation output files:

```
-bash-4.2$ ls -1 output_nr-prose-discovery-l3-relay
default-nr-prose-discovery-relay.db
NrSlDiscoveryTrace.txt
```

Model B is used, and the discovery is performed every 2 seconds (which is the default value in the example) until the end of the simulation (10 seconds).

```
bash-4.2$ cat output_nr-prose-discovery-l3-relay/NrSlDiscoveryTrace.txt
Time (s)        TX/RX   senderL2Id      receiverL2Id    DiscType        DiscModel       ContentType     Content
2.0000000000    TX      2       500     Restricted      ModelB  RelaySolicitation       5;2;1;2;0
2.0018321400    RX      2       1       Restricted      ModelB  RelaySolicitation       5;2;1;2;0
2.0018321400    TX      1       2       Restricted      ModelB  RelayResponse   5;1;1;1;0
2.0045821400    RX      1       2       Restricted      ModelB  RelayResponse   5;1;1;1;0
4.0000000000    TX      2       500     Restricted      ModelB  RelaySolicitation       5;2;1;2;0
4.0018321400    RX      2       1       Restricted      ModelB  RelaySolicitation       5;2;1;2;0
4.0018321400    TX      1       2       Restricted      ModelB  RelayResponse   5;1;1;1;0
4.0045821400    RX      1       2       Restricted      ModelB  RelayResponse   5;1;1;1;0
6.0000000000    TX      2       500     Restricted      ModelB  RelaySolicitation       5;2;1;2;0
6.0020821400    RX      2       1       Restricted      ModelB  RelaySolicitation       5;2;1;2;0
6.0020821400    TX      1       2       Restricted      ModelB  RelayResponse   5;1;1;1;0
6.0045821400    RX      1       2       Restricted      ModelB  RelayResponse   5;1;1;1;0
8.0000000000    TX      2       500     Restricted      ModelB  RelaySolicitation       5;2;1;2;0
8.0045821400    RX      2       1       Restricted      ModelB  RelaySolicitation       5;2;1;2;0
8.0045821400    TX      1       2       Restricted      ModelB  RelayResponse   5;1;1;1;0
```

Successful relay discovery

Failed relay discovery due to propagation loss

NR ProSe direct discovery: Relay Discovery - Example

nr-prose-discovery-l3-relay.cc

Running the scenario **with a different discovery interval**:

```
bash-4.2$ mkdir output_nr-prose-discovery-l3-relay-4s
bash-4.2$ ./ns3 run 'nr-prose-discovery-l3-relay --discInterval=4' --cwd='output_nr-prose-discovery-l3-relay-4s'
```

Simulation output files:

```
bash-4.2$ ls -1 output_nr-prose-discovery-l3-relay
default-nr-prose-discovery-relay.db
NrSlDiscoveryTrace.txt
```

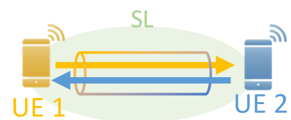
Changing the discovery periodicity to 4 seconds changes the output results.

```
bash-4.2$ cat output_nr-prose-discovery-l3-relay-4s/NrSlDiscoveryTrace.txt
Time (s)        TX/RX   senderL2Id      receiverL2Id    DiscType        DiscModel       ContentType     Content
2.0000000000    TX      2       500     Restricted      ModelB  RelaySolicitation       5;2;1;2;0
2.0018321400    RX      2       1       Restricted      ModelB  RelaySolicitation       5;2;1;2;0
2.0018321400    TX      1       2       Restricted      ModelB  RelayResponse   5;1;1;1;0
2.0045821400    RX      1       2       Restricted      ModelB  RelayResponse   5;1;1;1;0
6.0000000000    TX      2       500     Restricted      ModelB  RelaySolicitation       5;2;1;2;0
6.0018321400    RX      2       1       Restricted      ModelB  RelaySolicitation       5;2;1;2;0
6.0018321400    TX      1       2       Restricted      ModelB  RelayResponse   5;1;1;1;0
6.0045821400    RX      1       2       Restricted      ModelB  RelayResponse   5;1;1;1;0
```

NR ProSe unicast communication

NR ProSe unicast communication - Overview

- NR SL Unicast
  - Logical link between two UEs
  - Local control at different layers



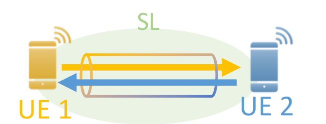
Summary from TS 38.300 NR; NR and NG-RAN Overall description; Stage-2- Section 16.9 Sidelink

Functionality	**Unicast**	**Groupcast**	**Broadcast**
Transmission and reception of user traffic over the Sidelink	Between two peer UEs	Between UEs belonging to a group	Between UEs
Transmission and reception of control information over the Sidelink	Between peer UEs (PC5-S, PC5-RRC)	-	-
Support of sidelink HARQ feedback	Yes	Yes	-
Support of sidelink transmit power control	Yes	-	-
Support of RLC AM	Yes	-	-
Support of one PC5-RRC connection	Between peer UEs	-	-
Detection of radio link failure for the PC5-RRC connection.	Yes	-	-

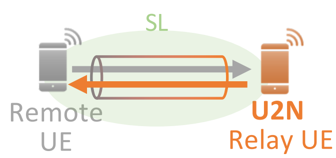
- How the link is established, used and controlled is defined by the service using it
  - NR V2X communication over PC-5, Unicast mode communication over PC5 reference point (TS 23.287 - 5.2.1.4)
  - **Unicast Mode 5G ProSe direct communications, One-to-one 5G ProSe direct communications (TS 24.554 – 7.2)**

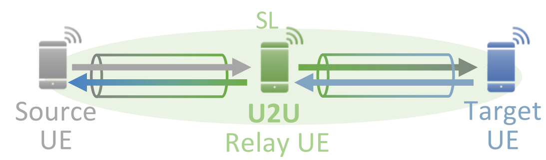
NR ProSe unicast communication - Overview

- Desired configurations
  - Manual unicast link association from scenario (pre-simulation)
    - Enables the evaluation of unicast communication protocols
      - Scheduling, HARQ, power control
      - Link control (PC5-S, PC5-RRC, RLF)



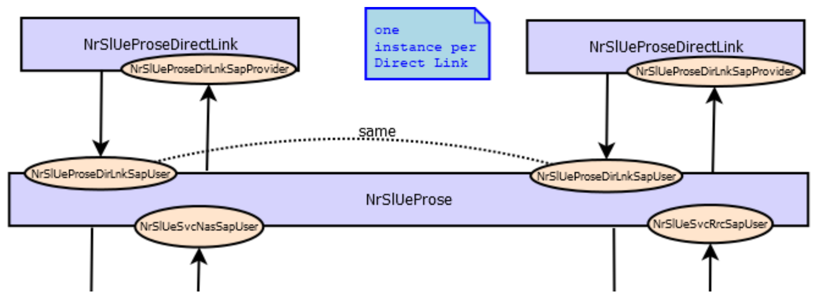
  - Dynamic unicast link association during simulation
    - Enables the use of unicast communication in other functionalities
      - E.g., U2N relay, U2U relay
        - Link establishment after relay discovery and selection



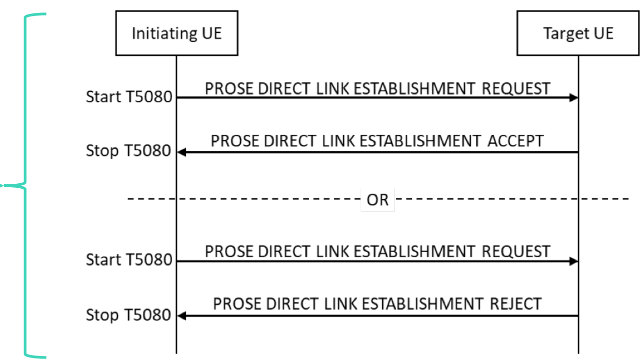


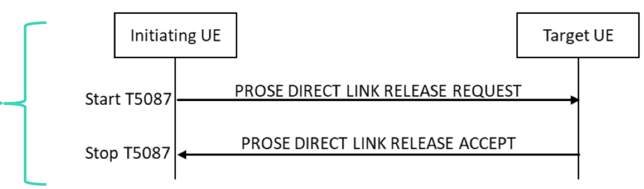
NR ProSe unicast communication - Implementation

- Architecture
  - One direct link per pair of UEs
  - A UE can have multiple direct links with different peer UEs
  - Each UE keeps a direct link instance associated to the peer L2ID in the link



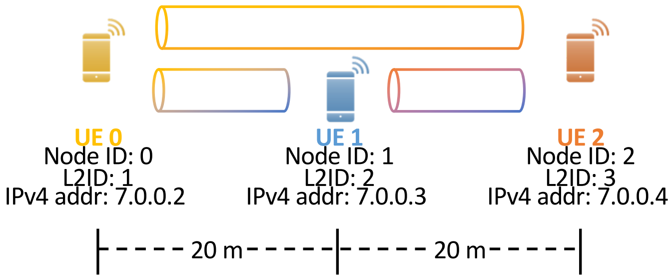
- Protocol (TS 24.554 section 7.2)
  - 5G ProSe direct link establishment procedure
  - 5G ProSe direct link modification procedure
  - 5G ProSe direct link identifier update procedure
  - 5G ProSe direct link keep-alive procedure
  - 5G ProSe direct link release procedure
  - PC5 QoS flow establishment over 5G ProSe direct link
  - PC5 QoS flow match over 5G ProSe direct link
  - Data transmission over 5G ProSe direct link
  - 5G ProSe direct link security mode control procedure
  - 5G ProSe direct link re-keying procedure
  - 5G ProSe direct link authentication procedure





NR ProSe unicast communication - Example

nr-prose-unicast-multi-link.cc



Three UEs deployed with a configurable inter-UE distance (default 20 m)

Each pair of UEs establish a unicast direct link with each other at a configurable simulation time (default 2 s)

The initiating UE is selected by increased node ID, i.e.,

Link	Initiating UE	Target UE
UE0 <--> UE1	UE0	UE1
UE0 <--> UE2	UE0	UE2
UE1 <--> UE2	UE1	UE2

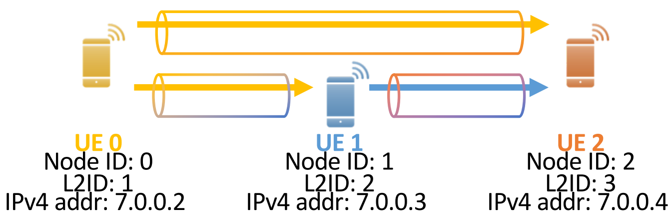
Source code: src/nr/examples/nr-prose-examples/nr-prose-unicast-multi-link.cc

Direct link configuration:

```
//Create ProSe helper
Ptr<NrSlProseHelper> nrSlProseHelper = CreateObject <NrSlProseHelper> ();
// Install ProSe layer and corresponding SAPs in the UEs
nrSlProseHelper->PrepareUesForProse (ueVoiceNetDev);
//Configure ProSe Unicast parameters. At the moment it only instruct the MAC
//layer (and PHY therefore) to monitor packets directed the UE's own Layer 2 ID
nrSlProseHelper->PrepareUesForUnicast (ueVoiceNetDev);
//Configure the value of timer Timer T5080 (Prose Direct Link Establishment Request Retransmissio
//to a lower value than the standard (8.0 s) to speed connection in shorter simulation time
Config::SetDefault ("ns3::NrSlUeProseDirectLink::T5080", TimeValue (Seconds (1.0)));
/*
 * Setup the start of the ProSe direct link establishment procedure
 * (with the 'Real' protocol, PC5-S messages are exchanged over the SL)
 * First UE on the function call will be the initiating UE (UE i),
 * which starts the procedure, and it is interested in establish a direct
 * link with the following j UEs, which will be the target UEs
 */
NS_LOG_INFO ("Configuring unicast direct links..." );

for (uint32_t i = 0; i < ueVoiceContainer.GetN () - 1; ++i)
  {
    for (uint32_t j = i + 1; j < ueVoiceContainer.GetN (); ++j)
      {
        nrSlProseHelper->EstablishRealDirectLink (startDirLinkTime,
                                                  ueVoiceNetDev.Get (i), ipv4AddressVector [i],
                                                  ueVoiceNetDev.Get (j), ipv4AddressVector [j]);
        NS_LOG_INFO ("Initiating UE nodeId " << i << " target UE nodeId " << j );

      }
  }
```

NR ProSe unicast communication - Example

nr-prose-unicast-multi-link.cc



Source code: src/nr/examples/nr-prose-examples/nr-prose-unicast-multi-link.cc

A CBR traffic flow from the initiating UE towards the target UE is configured by default

The packet size, data rate and general starting time of the flows can be configured. The default values:

Parameter	Default value
udpPacketSize	200 Bytes
dataRate	16 kb/s
startTrafficTime	3 s

Actual starting time is randomized within a 100 ms range to avoid simultaneous transmissions

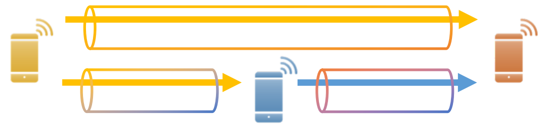
CBR: Constant Bit Rate

Traffic configuration:

```
// Random variable to randomize a bit start times of the client applications
//to avoid simulation artifacts of all the TX UEs transmitting at the same time.
Ptr<UniformRandomVariable> startTimeRnd = CreateObject<UniformRandomVariable> ();
startTimeRnd->SetAttribute ("Min", DoubleValue (0));
startTimeRnd->SetAttribute ("Max", DoubleValue (0.10));

std::string dataRateString  = std::to_string (dataRate) + "kb/s";
ApplicationContainer clientApps;
std::cout << "Traffic flows: " << std::endl;
for (uint32_t i = 0; i < ueVoiceContainer.GetN () - 1; ++i)
  {
    for (uint32_t j = i + 1; j < ueVoiceContainer.GetN (); ++j)
      {
        OnOffHelper sidelinkClient ("ns3::UdpSocketFactory",
                                    InetSocketAddress (ipv4AddressVector [j], port)); //Towards UE j
        sidelinkClient.SetAttribute ("EnableSeqTsSizeHeader", BooleanValue (true));
        sidelinkClient.SetConstantRate (DataRate (dataRateString), udpPacketSize);
        ApplicationContainer app = sidelinkClient.Install (ueVoiceContainer.Get (i)); // Installed in UE i
        Time appStart = startTrafficTime + Seconds (startTimeRnd->GetValue ());
        app.Start (appStart);
        clientApps.Add (app);
        NS_LOG_INFO ("OnOff application installed in UE nodeId " << i << " srcIp " << ipv4AddressVector [i] <<
                     " towards UE nodeId " << j << " dstIp " << ipv4AddressVector [j] );
        std::cout << ipv4AddressVector [i] << " -> " << ipv4AddressVector [j] <<
          " start time: " << appStart.GetSeconds () << " s, end time: " << simTime.GetSeconds () << " s" << std::endl;
```

NR ProSe unicast communication - Example

nr-prose-unicast-multi-link.cc



Running the scenario:

```
-bash-4.2$ mkdir output_nr-prose-unicast-multi-link
-bash-4.2$ ./ns3 run 'nr-prose-unicast-multi-link' --cwd='output_nr-prose-unicast-multi-link'
```

Simulation standard output:

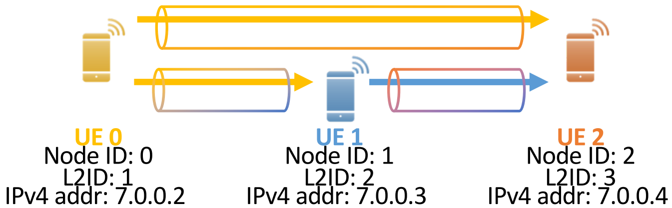
```
Traffic flows:
7.0.0.2 -> 7.0.0.3 start time: 3.05717 s, end time: 20 s
7.0.0.2 -> 7.0.0.4 start time: 3.03047 s, end time: 20 s
7.0.0.3 -> 7.0.0.4 start time: 3.06853 s, end time: 20 s
System total Tx packets = 507
System total Tx bits = 811200
System total Rx packets = 507
System total Rx bits = 811200
System average thput = 47.7176 kbps
Traffic flows statistics:
  Flow 1 (7.0.0.2 -> 7.0.0.4) UDP
    Tx Packets:    169
    Tx Bytes:      38532
    Tx Data rate:  18.132706 kbps
    Rx Packets:    169
    Rx Bytes:      38532
    Rx Data rate:  18.132706 kbps
    Mean delay:    4.716704 ms
  Flow 2 (7.0.0.2 -> 7.0.0.3) UDP
    Tx Packets:    169
    Tx Bytes:      38532
    Tx Data rate:  18.132706 kbps
    Rx Packets:    169
    Rx Bytes:      38532
    Rx Data rate:  18.132706 kbps
    Mean delay:    6.722074 ms
  Flow 3 (7.0.0.3 -> 7.0.0.4) UDP
    Tx Packets:    169
    Tx Bytes:      38532
    Tx Data rate:  18.132706 kbps
    Rx Packets:    169
    Rx Bytes:      38532
    Rx Data rate:  18.132706 kbps
    Mean delay:    5.293635 ms
```

Simulation output files:

```
-bash-4.2$ ls -1 output_nr-prose-unicast-multi-link/
default-nr-prose-unicast-multi-link.db
default-nr-prose-unicast-multi-link-flowMonitorOutput.txt
default-nr-prose-unicast-multi-link-NrSlPc5SignallingPacketTrace.txt
```

NR ProSe unicast communication - Example

nr-prose-unicast-multi-link.cc



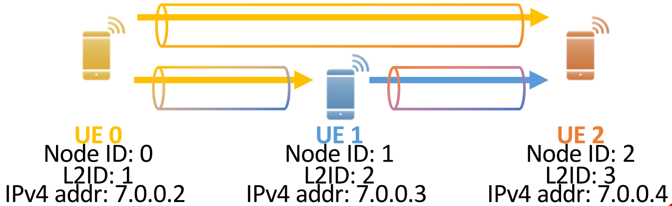
Simulation output files:

```
-bash-4.2$ ls -1 output_nr-prose-unicast-multi-link/
default-nr-prose-unicast-multi-link.db
default-nr-prose-unicast-multi-link-flowMonitorOutput.txt
default-nr-prose-unicast-multi-link-NrSlPc5SignallingPacketTrace.txt
```

```
-bash-4.2$ cat output_nr-prose-unicast-multi-link/default-nr-prose-unicast-multi-link-NrSlPc5SignallingPacketTrace.txt
time(s) nodeId  TX/RX   srcL2Id dstL2Id msgType
2       0       TX      1       2       PROSE DIRECT LINK ESTABLISHMENT REQUEST
2       0       TX      1       3       PROSE DIRECT LINK ESTABLISHMENT REQUEST
2       1       TX      2       3       PROSE DIRECT LINK ESTABLISHMENT REQUEST
2.00183 2       RX      1       3       PROSE DIRECT LINK ESTABLISHMENT REQUEST
2.00183 2       TX      3       1       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
2.00483 2       RX      2       3       PROSE DIRECT LINK ESTABLISHMENT REQUEST
2.00483 2       TX      3       2       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
2.00633 0       RX      3       1       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
2.00708 1       RX      1       2       PROSE DIRECT LINK ESTABLISHMENT REQUEST
2.00708 1       TX      2       1       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
2.00708 1       RX      3       2       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
2.01133 0       RX      2       1       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
```

NR ProSe unicast communication - Example

nr-prose-unicast-multi-link.cc



Simulation output files:

```
-bash-4.2$ ls -1 output_nr-prose-unicast-multi-link/
default-nr-prose-unicast-multi-link.db
default-nr-prose-unicast-multi-link-flowMonitorOutput.txt
default-nr-prose-unicast-multi-link-NrSlPc5SignallingPacketTrace.txt
```

```
-bash-4.2$ cat output_nr-prose-unicast-multi-link/default-nr-prose-unicast-multi-link-flowMonitorOutput.txt
  Flow 1 (7.0.0.2 -> 7.0.0.4) UDP
    Tx Packets:   169
    Tx Bytes:     38532
    Tx Data rate: 18.132706 kbps
    Rx Packets:   169
    Rx Bytes:     38532
    Rx Data rate: 18.132706 kbps
    Mean delay:   4.716704 ms
  Flow 2 (7.0.0.2 -> 7.0.0.3) UDP
    Tx Packets:   169
    Tx Bytes:     38532
    Tx Data rate: 18.132706 kbps
    Rx Packets:   169
    Rx Bytes:     38532
    Rx Data rate: 18.132706 kbps
    Mean delay:   6.722074 ms
  Flow 3 (7.0.0.3 -> 7.0.0.4) UDP
    Tx Packets:   169
    Tx Bytes:     38532
    Tx Data rate: 18.132706 kbps
    Rx Packets:   169
    Rx Bytes:     38532
    Rx Data rate: 18.132706 kbps
    Mean delay:   5.293635 ms
```

NR ProSe unicast communication - Example

nr-prose-unicast-multi-link.cc

Simulation output files:

```
-bash-4.2$ ls -1 output_nr-prose-unicast-multi-link/
default-nr-prose-unicast-multi-link.db
default-nr-prose-unicast-multi-link-flowMonitorOutput.txt
default-nr-prose-unicast-multi-link-NrSlPc5SignallingPacketTrace.txt
```

DB Browser for SQLite - C:\Users\smg\Desktop\wns32023SimFiles\default-nr-prose-unicast-multi-link.db

- Tables (3)
  - pktTxRx
  - pscchRxUePhy
  - psschRxUePhy

Table: pktTxRx

	timeSec	txRx	nodeId	imsi	pktSizeBytes	srcIp	srcPort	dstIp	dstPort	pktSeqNum	SEED	RUN
1	3.130471945	tx	0	1	200	7.0.0.2	49153	7.0.0.4	8000	0	1	1
2	3.13233214	rx	2	3	200	7.0.0.2	49153	7.0.0.4	8000	0	1	1
3	3.157167758	tx	0	1	200	7.0.0.2	49154	7.0.0.3	8000	0	1	1
4	3.15958214	rx	1	2	200	7.0.0.2	49154	7.0.0.3	8000	0	1	1
5	3.168534067	tx	1	2	200	7.0.0.3	49153	7.0.0.4	8000	0	1	1
6	3.17183214	rx	2	3	200	7.0.0.3	49153	7.0.0.4	8000	0	1	1
7	3.230471945	tx	0	1	200	7.0.0.2	49153	7.0.0.4	8000	1	1	1
8	3.23233214	rx	2	3	200	7.0.0.2	49153	7.0.0.4	8000	1	1	1

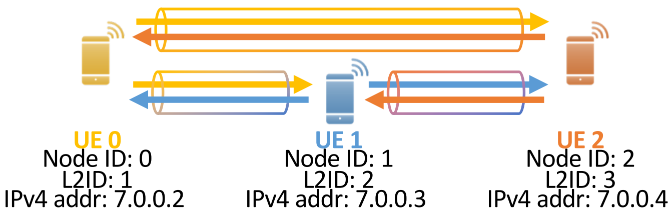
Table: psschRxUePhy

	timeMs	cellId	rnti	bwpId	frame	subFrame	slot	txRnti	srcL2Id	dstL2Id	psschRbStart	psschRbLen	psschSymStart	psschSymLen	psschMcs	ndi	rv	tbSizeBytes	avrgSinr	minSinr	psschTbler	psschCorrupt	sci2Tbler	sci2Corrupt	SEED	RUN
1	2001.73214	0	0	0	200	1	2	1	1	3	40	10	1	12	14	1	0	348	20292.80...	19423.50...	0	0	0	0	1	1
2	2002.23214	0	3	0	200	2	0	2	2	3	20	10	1	12	14	1	0	348	2....	1....	1	1	0	0	1	1
3	2004.73214	0	3	0	200	4	2	2	2	3	20	10	1	12	14	0	2	348	47368.64...	34993.22...	0	0	0	0	1	1
4	2004.98214	0	1	0	200	4	3	3	3	1	40	10	1	12	14	1	0	348	0....	0....	1	1	1	1	1	1
5	2006.23214	0	1	0	200	6	0	3	3	1	10	10	1	12	14	0	2	348	17679.59...	17533.70...	0.064	0	0	0	1	1
6	2006.98214	0	2	0	200	6	3	1	1	2	30	10	1	12	14	0	1	348	62162.42...	53919.68...	0	0	0	0	1	1
7	2006.98214	0	2	0	200	6	3	3	3	2	40	10	1	12	14	1	0	348	62774.05...	47004.94...	0	0	0	0	1	1
8	2011.23214	0	1	0	201	1	0	2	2	1	20	10	1	12	14	1	0	348	79025.23...	72943.03...	0	0	0	0	1	1
9	3132.23214	0	3	0	313	2	0	1	1	3	10	10	1	12	14	1	0	348	20849.09...	17135.52...	0	0	0	0	1	1
10	3159.48214	0	2	0	315	9	1	1	1	2	10	10	1	12	14	1	0	348	102850.2...	101569.9...	0	0	0	0	1	1

DB Browser for SQLite (Windows/Linux/MacOS) available in: https://sqlitebrowser.org/dl/

NR ProSe unicast communication - Example

nr-prose-unicast-multi-link.cc



When the parameter 'bidirectional' is set to true, additional CBR flows are added from the target UE towards the initiating UE of each link

Running the scenario with **bidirectional traffic**:

```
-bash-4.2$ mkdir output_nr-prose-unicast-multi-link-bidirectional
-bash-4.2$ ./ns3 run 'nr-prose-unicast-multi-link --bidirectional=true'
 --cwd='output_nr-prose-unicast-multi-link-bidirectional'
```

Simulation standard output

```
Traffic flows:
7.0.0.2 -> 7.0.0.3 start time: 3.05717 s, end time: 20 s
7.0.0.3 -> 7.0.0.2 start time: 3.03047 s, end time: 20 s
7.0.0.2 -> 7.0.0.4 start time: 3.06853 s, end time: 20 s
7.0.0.4 -> 7.0.0.2 start time: 3.05132 s, end time: 20 s
7.0.0.3 -> 7.0.0.4 start time: 3.06608 s, end time: 20 s
7.0.0.4 -> 7.0.0.3 start time: 3.03233 s, end time: 20 s
System total Tx packets = 1014
System total Tx bits = 1622400
System total Rx packets = 978
System total Rx bits = 1564800
System average thput = 92.0471 kbps
Traffic flows statistics:
  Flow 1 (7.0.0.3 -> 7.0.0.2) UDP
    Tx Packets:    169
    Tx Bytes:      38532
    Tx Data rate: 18.132706 kbps
    Rx Packets:    169
    Rx Bytes:      38532
    Rx Data rate: 18.132706 kbps
    Mean delay:    4.672325 ms
  Flow 2 (7.0.0.4 -> 7.0.0.3) UDP
    Tx Packets:    169
    Tx Bytes:      38532
    Tx Data rate: 18.132706 kbps
    Rx Packets:    144
    Rx Bytes:      32832
    Rx Data rate: 15.450353 kbps
    Mean delay:    10.387269 ms
  Flow 3 (7.0.0.4 -> 7.0.0.2) UDP
    Tx Packets:    169
    Tx Bytes:      38532
    Tx Data rate: 18.132706 kbps
    Rx Packets:    167
    Rx Bytes:      38076
    Rx Data rate: 17.918118 kbps
    Mean delay:    5.764554 ms
```

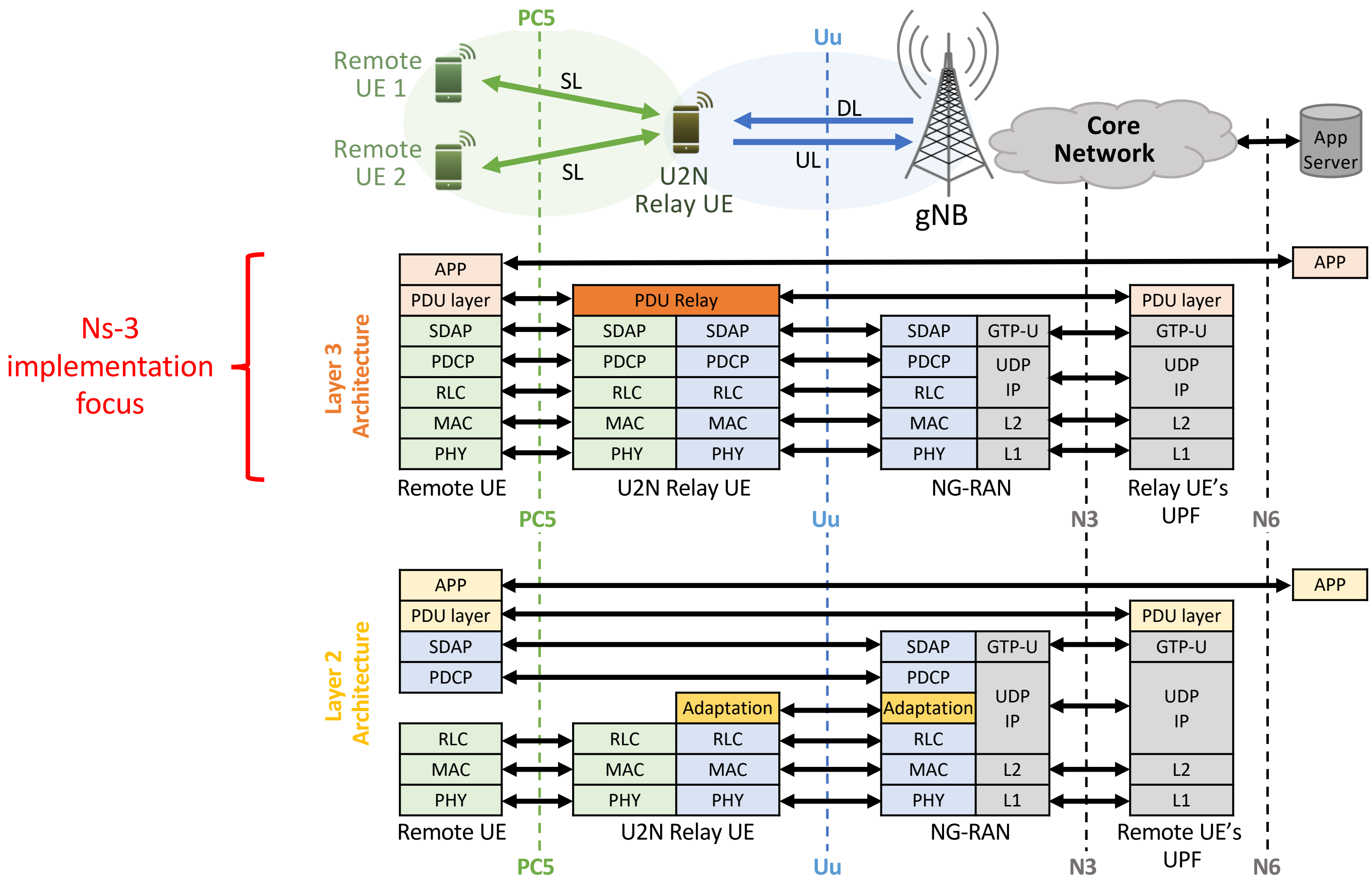
```
Flow 4 (7.0.0.2 -> 7.0.0.3) UDP
  Tx Packets:    169
  Tx Bytes:      38532
  Tx Data rate: 18.132706 kbps
  Rx Packets:    165
  Rx Bytes:      37620
  Rx Data rate: 17.703529 kbps
  Mean delay:    6.968927 ms
Flow 5 (7.0.0.3 -> 7.0.0.4) UDP
  Tx Packets:    169
  Tx Bytes:      38532
  Tx Data rate: 18.132706 kbps
  Rx Packets:    164
  Rx Bytes:      37392
  Rx Data rate: 17.596235 kbps
  Mean delay:    5.597453 ms
Flow 6 (7.0.0.2 -> 7.0.0.4) UDP
  Tx Packets:    169
  Tx Bytes:      38532
  Tx Data rate: 18.132706 kbps
  Rx Packets:    169
  Rx Bytes:      38532
  Rx Data rate: 18.132706 kbps
  Mean delay:    5.815824 ms
```

Simulation output files:

```
bash-4.2$ ls -1 output_nr-prose-unicast-multi-link-bidirectional/
default-nr-prose-unicast-multi-link.db
default-nr-prose-unicast-multi-link-flowMonitorOutput.txt
default-nr-prose-unicast-multi-link-NrSlPc5SignallingPacketTrace.txt
```

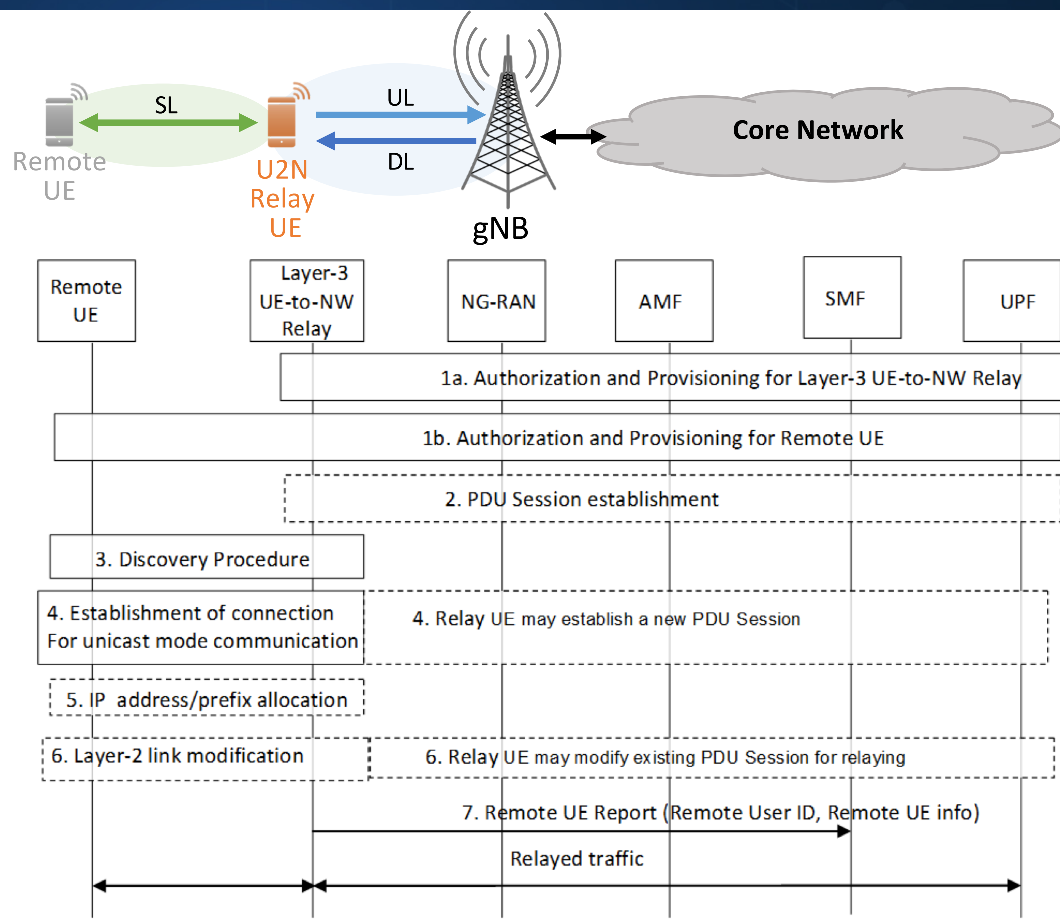
NR ProSe L3 UE-to-Network relay

NR ProSe L3 UE-to-Network relay - Overview



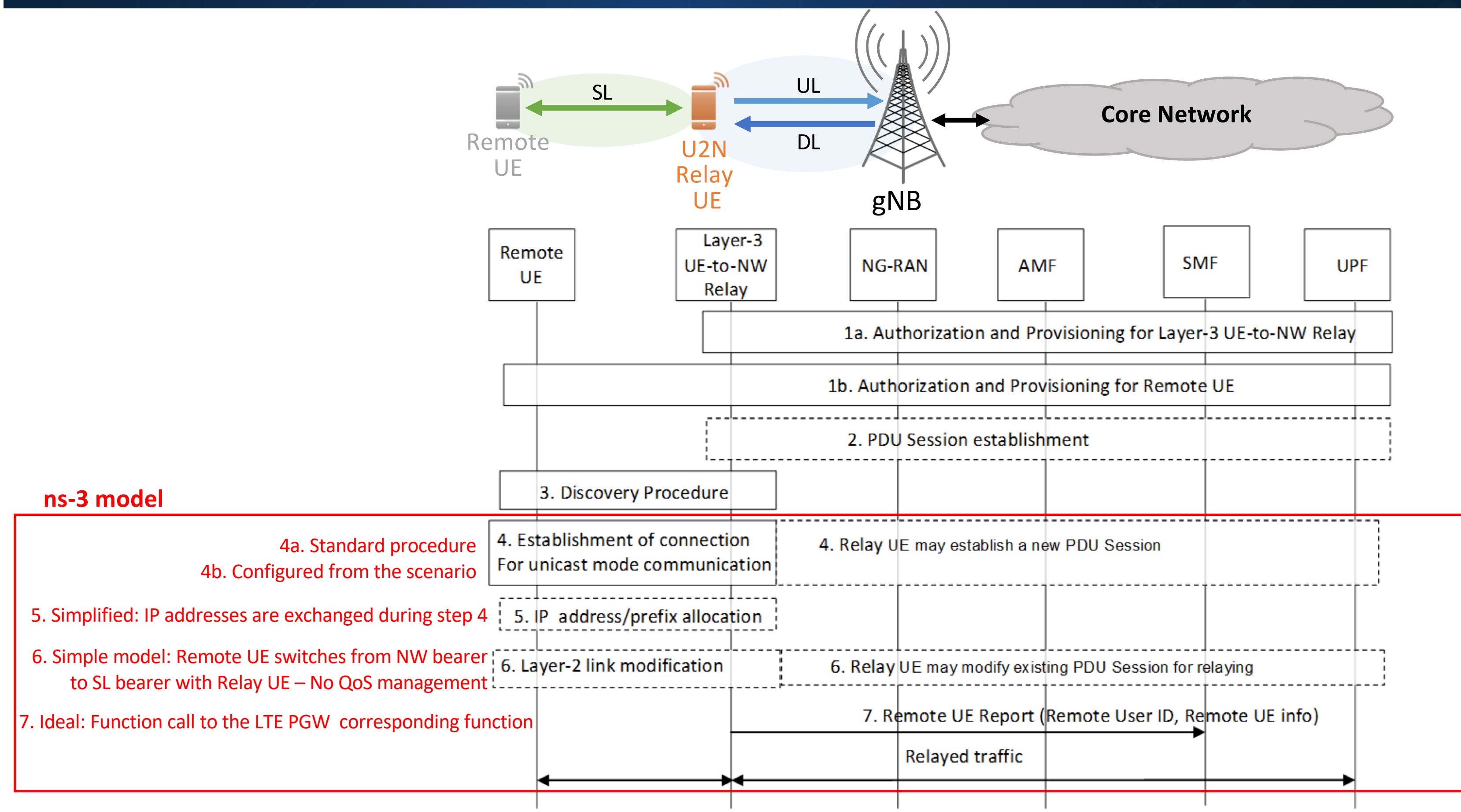
GTP-U: GPRS Tunneling Protocol
IP: Internet Protocol
L1: Layer 1
L2: Layer 2
MAC: Media Access Control
PDU: Protocol Data Unit
PDCP: Packet Data Convergence Protocol
PHY: Physical Layer
RLC: Radio Link Control
SDAP: Service Data Adaptation Protocol
UDP: User Datagram Protocol

NR ProSe L3 UE-to-Network relay - Overview



TS 23.304 **Figure 6.5.1.1-1: 5G ProSe Communication via 5G ProSe Layer-3 UE-to-Network Relay without N3IWF**

NR ProSe L3 UE-to-Network relay - Overview



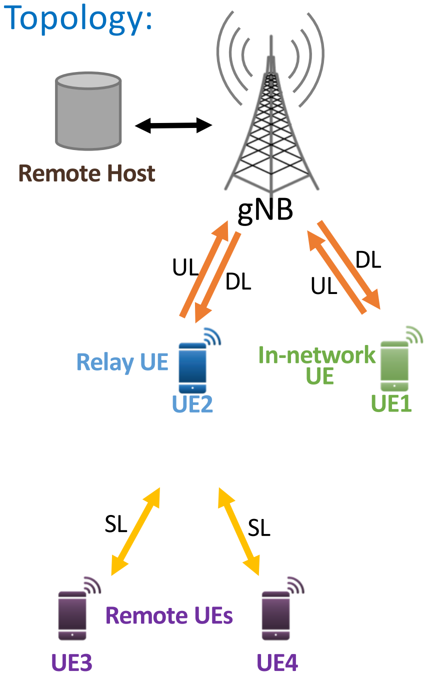
TS 23.304 **Figure 6.5.1.1-1: 5G ProSe Communication via 5G ProSe Layer-3 UE-to-Network Relay without N3IWF**

NR ProSe L3 UE-to-Network relay - Example

nr-prose-l3-relay.cc

Source code: src/nr/examples/nr-prose-examples/nr-prose-l3-relay.cc

Topology:



Spectrum division

The scenario uses one operational band, containing one component carrier, and two bandwidth parts (BWP):

- BWP0 is used for in-network communication, i.e., UL and DL between in-network UEs and gNBs
- BWP1 is used for direct communication, i.e., SL between the relay UE and the remote UEs

```
/*
 * The configured spectrum division is:
 * |------------- Band ------------|
 * |--------------CC0--------------|
 * |------BWP0------|------BWP1-----|
 */
```

```
NetDeviceContainer inNetUeNetDev = nrHelper->InstallUeDevice (inNetUeNodes, inNetBwp);
NetDeviceContainer enbNetDev = nrHelper->InstallGnbDevice (gNbNodes, inNetBwp);
```

```
//Install both BWPs on U2N relays
NetDeviceContainer relayUeNetDev = nrHelper->InstallUeDevice (relayUeNodes, allBwps);

//Install both BWPs on remote UEs
//This was needed to avoid errors with bwpId and vector indexes during device installation
NetDeviceContainer remoteUeNetDev = nrHelper->InstallUeDevice (remoteUeNodes, allBwps );
```

Relay UE uses both BWPs
Some special config is needed to do SL only in BWP1

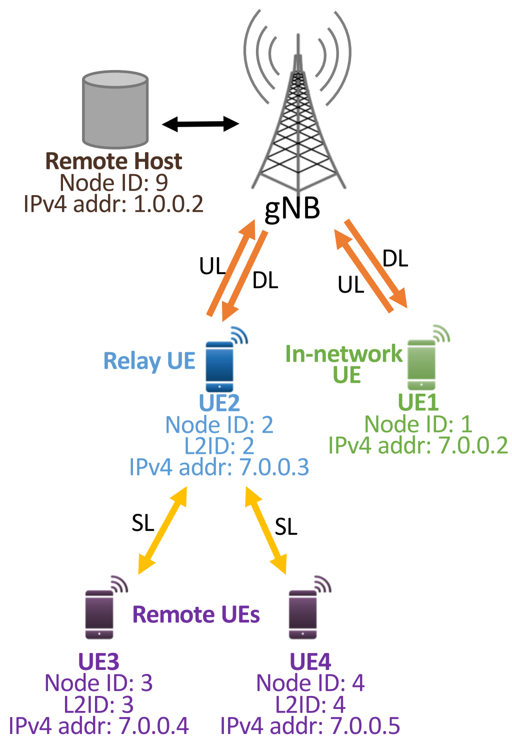
Remote UEs configure both BWPs but SL only uses BWP1

```
//For U2N relay UEs we need to modify some parameters to configure *only*
//BWP1 on the relay for SL and avoid MAC problems
LteRrcSap::SlFreqConfigCommonNr slFreConfigCommonNrRelay;
slFreConfigCommonNrRelay.slBwpList [bwpIdSl] = slBwpConfigCommonNr;

LteRrcSap::SidelinkPreconfigNr slPreConfigNrRelay;
slPreConfigNrRelay.slPreconfigGeneral = slPreconfigGeneralNr;
slPreConfigNrRelay.slUeSelectedPreConfig = slUeSelectedPreConfig;
slPreConfigNrRelay.slPreconfigFreqInfoList [0] = slFreConfigCommonNrRelay;

nrSlHelper->InstallNrSlPreConfiguration (relayUeNetDev, slPreConfigNrRelay);
```

NR ProSe L3 UE-to-Network relay - Example

nr-prose-l3-relay.cc

Source code: src/nr/examples/nr-prose-examples/nr-prose-l3-relay.cc



Network connections:

```
//Attach in-network UEs to the closest gNB
nrHelper->AttachToClosestEnb (inNetUeNetDev, enbNetDev);
```

```
//Attach U2N relay UEs to the closest gNB
nrHelper->AttachToClosestEnb (relayUeNetDev, enbNetDev);
```

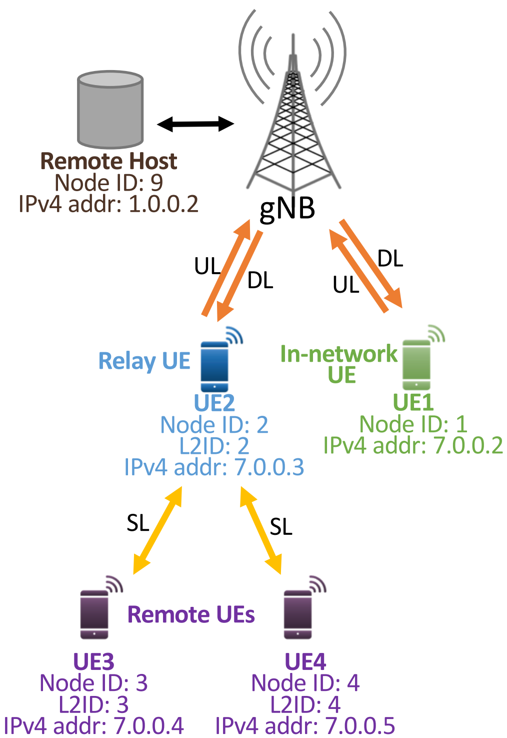
ProSe layer configuration:

```
/******** Configure ProSe layer in the UEs that will do SL  **********/
//Create ProSe helper
Ptr<NrSlProseHelper> nrSlProseHelper = CreateObject <NrSlProseHelper> ();
nrSlProseHelper->SetEpcHelper (epcHelper);

// Install ProSe layer and corresponding SAPs in the UEs
nrSlProseHelper->PrepareUesForProse (relayUeNetDev);
nrSlProseHelper->PrepareUesForProse (remoteUeNetDev);

//Configure ProSe Unicast parameters. At the moment it only instruct the MAC
//layer (and PHY therefore) to monitor packets directed the UE's own Layer 2 ID
nrSlProseHelper->PrepareUesForUnicast (relayUeNetDev);
nrSlProseHelper->PrepareUesForUnicast (remoteUeNetDev);

//Configure the value of timer Timer T5080 (Prose Direct Link Establishment Request Retransmission)
//to a lower value than the standard (8.0 s) to speed connection in shorter simulation time
Config::SetDefault ("ns3::NrSlUeProseDirectLink::T5080", TimeValue (Seconds (2.0)));
/******** END Configure ProSe layer in the UEs that will do SL  **********/
```

NR ProSe L3 UE-to-Network relay - Example

nr-prose-l3-relay.cc

Source code: src/nr/examples/nr-prose-examples/nr-prose-l3-relay.cc



L3 U2N Relay configuration (1/2)

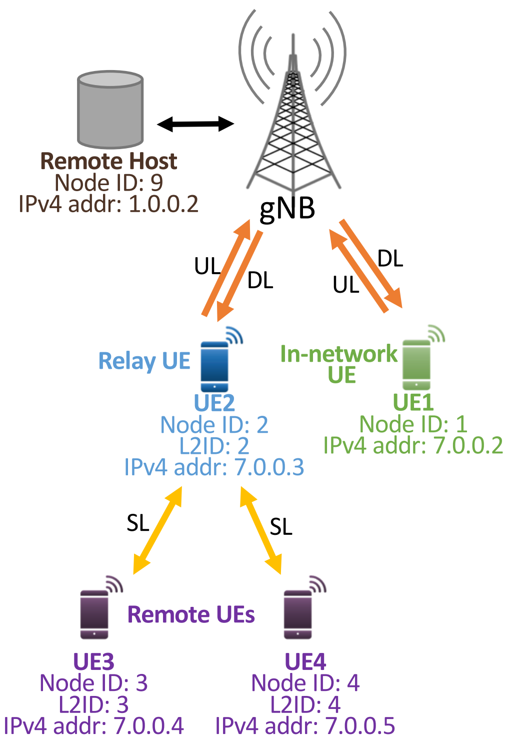
```
//-Configure relay service codes
// Only one relay service per relay UE is currently supported
uint32_t relayServiceCode = 5;
std::set<uint32_t> relaySCs;
relaySCs.insert (relayServiceCode);

//-Configure the UL data radio bearer that the relay UE will use for U2N relaying traffic
Ptr<EpcTft> tftRelay = Create<EpcTft> ();
EpcTft::PacketFilter pfRelay;
tftRelay->Add (pfRelay);
enum EpsBearer::Qci qciRelay;
qciRelay = EpsBearer::GBR_CONV_VOICE;
EpsBearer bearerRelay (qciRelay);

//Apply the configuration on the devices acting as relay UEs
nrSlProseHelper->ConfigureL3UeToNetworkRelay (relayUeNetDev, relaySCs, bearerRelay, tftRelay);
```

- Activates radio bearer to be used for relaying traffic
- Gets bearer Id and stores it in the corresponding context in the ProSe layer (used to direct relayed packets)
- Sets EpcHelper in the ProSe layer (used to configure data path in the network when a Remote UE connects)

NR ProSe L3 UE-to-Network relay - Example

nr-prose-l3-relay.cc

Source code: src/nr/examples/nr-prose-examples/nr-prose-l3-relay.cc



L3 U2N Relay configuration (2/2)

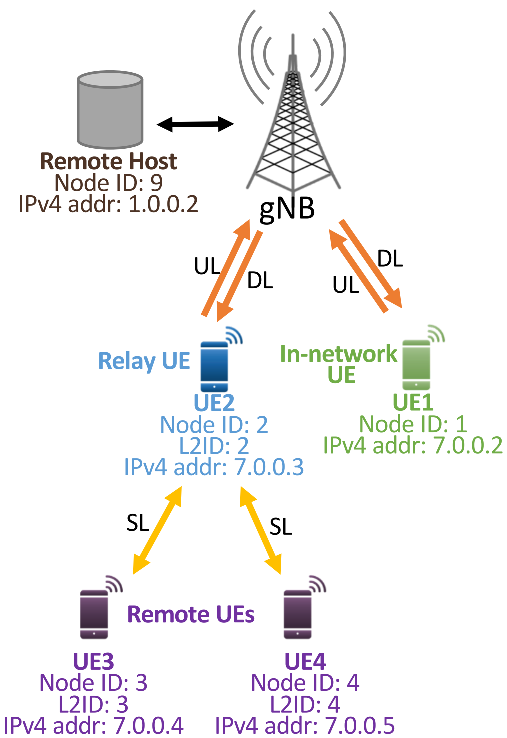
```
//Configure direct link connection between remote UEs and relay UEs
NS_LOG_INFO ("Configuring remote UE - relay UE connection..." );
uint32_t j = 0; //We have only one relay UE
for (uint32_t i = 0; i < remoteUeNodes.GetN (); ++i)
  {
    nrSlProseHelper->EstablishL3UeToNetworkRelayConnection (startRelayConnTime,
                                                            remoteUeNetDev.Get (i), slIpv4AddressVector [i], // Remote UE
                                                            relayUeNetDev.Get (j), relaysIpv4AddressVector [j], // Relay UE
                                                            relayServiceCode);

    NS_LOG_INFO ("Remote UE nodeId " << remoteUeNodes.Get (i)->GetId () <<
                 " Relay UE nodeId " << relayUeNodes.Get (j)->GetId ());
```

Assign IDs
Schedules the creation of the corresponding direct link instances in both UEs participating in the U2N relay connection (Remote UE is the initiating UE of the direct link and relay UE is the target UE)

NR ProSe L3 UE-to-Network relay - Example

nr-prose-l3-relay.cc

Source code: src/nr/examples/nr-prose-examples/nr-prose-l3-relay.cc



Traffic configuration

CBR traffic with configured packet size and data rate

One traffic flow from each UE to the Remote Host

One traffic flow from the Remote Host to each UE

Remote UEs code examples:

```
//Applications configuration
uint32_t packetSizeDlUl = 100; //bytes
uint32_t lambdaDl = 50; // packets per second
uint32_t lambdaUl = 50; // packets per second
double trafficStartTime = 5.0; //seconds
```

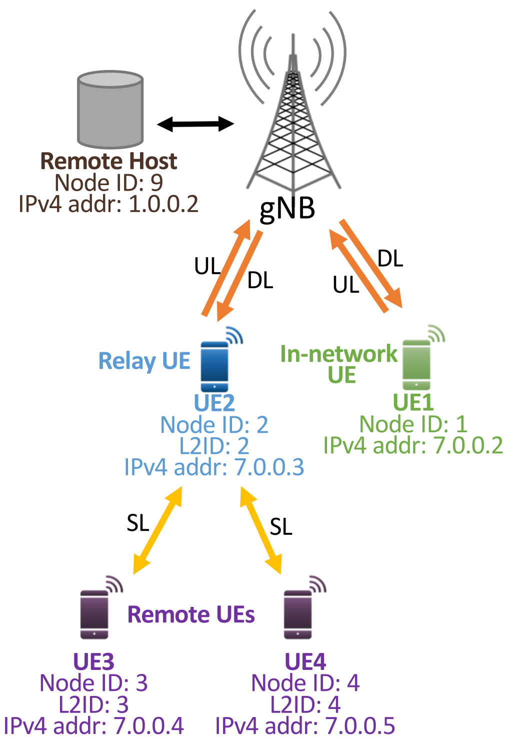
```
//DL traffic
PacketSinkHelper dlPacketSinkHelper ("ns3::UdpSocketFactory", InetSocketAddress (Ipv4Address::GetAny (), dlPort));
serverApps.Add (dlPacketSinkHelper.Install (remoteUeNodes.Get (u)));

UdpClientHelper dlClient (ueIpIfaceSl.GetAddress (u), dlPort);
dlClient.SetAttribute ("PacketSize", UintegerValue (packetSizeDlUl));
dlClient.SetAttribute ("Interval", TimeValue (Seconds (1.0 / lambdaDl)));
dlClient.SetAttribute ("MaxPackets", UintegerValue (0xFFFFFFFF));
ApplicationContainer dlApp = dlClient.Install (remoteHost);
appStartTime = Seconds (trafficStartTime + startTimeRnd->GetValue ());
dlApp.Start (appStartTime);
clientApps.Add (dlApp);
std::cout << " DL: " << remoteHostAddr << " -> " << ueIpIfaceSl.GetAddress (u) << ":" << dlPort <<
  " start time: " << appStartTime.GetSeconds ()  << " s, end time: " << simTime << " s" << std::endl;
```

```
// UL traffic
PacketSinkHelper ulPacketSinkHelper ("ns3::UdpSocketFactory", InetSocketAddress (Ipv4Address::GetAny (), ulPort));
serverApps.Add (ulPacketSinkHelper.Install (remoteHost));

UdpClientHelper ulClient (remoteHostAddr, ulPort);
ulClient.SetAttribute ("PacketSize", UintegerValue (packetSizeDlUl));
ulClient.SetAttribute ("Interval", TimeValue (Seconds (1.0 / lambdaUl)));
ulClient.SetAttribute ("MaxPackets", UintegerValue (0xFFFFFFFF));
ApplicationContainer ulApp = ulClient.Install (remoteUeNodes.Get (u));
appStartTime = Seconds (trafficStartTime + startTimeRnd->GetValue ());
ulApp.Start (appStartTime);
clientApps.Add (ulApp);
```

NR ProSe L3 UE-to-Network relay - Example

nr-prose-l3-relay.cc

Source code: src/nr/examples/nr-prose-examples/nr-prose-l3-relay.cc



Bearer activation

A dedicated EPS bearer is activated in each UE for each traffic flow direction

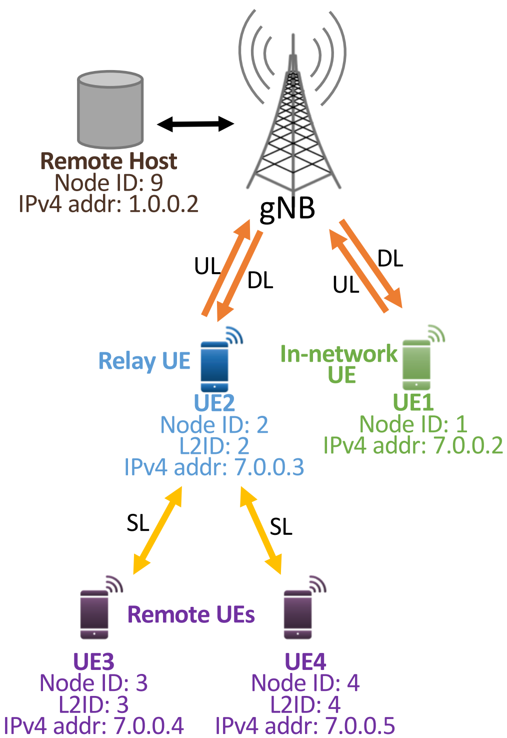
```
Ptr<EpcTft> tftDl = Create<EpcTft> ();
EpcTft::PacketFilter pfDl;
pfDl.localPortStart = dlPort;
pfDl.localPortEnd = dlPort;
++dlPort;
tftDl->Add (pfDl);

enum EpsBearer::Qci qDl;
qDl = EpsBearer::GBR_CONV_VOICE;

EpsBearer bearerDl (qDl);
nrHelper->ActivateDedicatedEpsBearer (remoteUeNetDev.Get (u), bearerDl, tftDl);
```

```
Ptr<EpcTft> tftUl = Create<EpcTft> ();
EpcTft::PacketFilter pfUl;
pfUl.remoteAddress = remoteHostAddr; //IMPORTANT!!!
pfUl.remotePortStart = ulPort;
pfUl.remotePortEnd = ulPort;
++ulPort;
tftUl->Add (pfUl);

enum EpsBearer::Qci qUl;

qUl = EpsBearer::GBR_CONV_VOICE;
EpsBearer bearerUl (qUl);
nrHelper->ActivateDedicatedEpsBearer (remoteUeNetDev.Get (u), bearerUl, tftUl);
```

For the remote UEs, these bearers won't be used, but the information on them is used to configure the traffic redirection on the NAS layer once a remote UEs connects to a relay UE.

NR ProSe L3 UE-to-Network relay - Example

nr-prose-l3-relay.cc



Running the scenario:

```
-bash-4.2$ mkdir output_nr-prose-l3-relay
-bash-4.2$ ./ns3 run 'nr-prose-l3-relay' --cwd='output_nr-prose-l3-relay'
```

Simulation standard output:

1

```
IP configuration:
 Remote Host: 1.0.0.2
 In-network only UE: 7.0.0.2
 Relay UE: 7.0.0.3
 Out-of-network UE: 7.0.0.4
 Out-of-network UE: 7.0.0.5
Traffic flows:
 DL: 1.0.0.2 -> 7.0.0.2:100 start time: 5.09149 s, end time: 15 s
 UL: 7.0.0.2 -> 1.0.0.2:200 start time: 5.03234 s, end time: 15 s
 DL: 1.0.0.2 -> 7.0.0.3:101 start time: 5.05273 s, end time: 15 s
 UL: 7.0.0.3 -> 1.0.0.2:201 start time: 5.00007 s, end time: 15 s
 DL: 1.0.0.2 -> 7.0.0.4:102 start time: 5.0144 s, end time: 15 s
 UL: 7.0.0.4 -> 1.0.0.2:202 start time: 5.0393 s, end time: 15 s
 DL: 1.0.0.2 -> 7.0.0.5:103 start time: 5.04749 s, end time: 15 s
 UL: 7.0.0.5 -> 1.0.0.2:203 start time: 5.02536 s, end time: 15 s
```

2

```
Simulation done!
Traffic flows statistics:
  Flow 1 (7.0.0.3 -> 1.0.0.2) UDP
    Tx Packets: 500
    Tx Bytes:   64000
    TxOffered:  0.051200 Mbps
    Rx Packets: 500
    Rx Bytes:   64000
    Throughput: 0.051200 Mbps
    Mean delay:  3.563026 ms
  Flow 2 (1.0.0.2 -> 7.0.0.4) UDP
    Tx Packets: 500
    Tx Bytes:   64000
    TxOffered:  0.051200 Mbps
    Rx Packets: 494
    Rx Bytes:   63232
    Throughput: 0.050586 Mbps
    Mean delay:  14.145069 ms
  Flow 3 (7.0.0.5 -> 1.0.0.2) UDP
    Tx Packets: 499
    Tx Bytes:   63872
    TxOffered:  0.051098 Mbps
    Rx Packets: 408
    Rx Bytes:   52224
    Throughput: 0.041779 Mbps
    Mean delay:  17.522055 ms
  Flow 4 (7.0.0.2 -> 1.0.0.2) UDP
    Tx Packets: 499
    Tx Bytes:   63872
    TxOffered:  0.051098 Mbps
    Rx Packets: 499
    Rx Bytes:   63872
    Throughput: 0.051098 Mbps
    Mean delay:  3.254544 ms
```

3

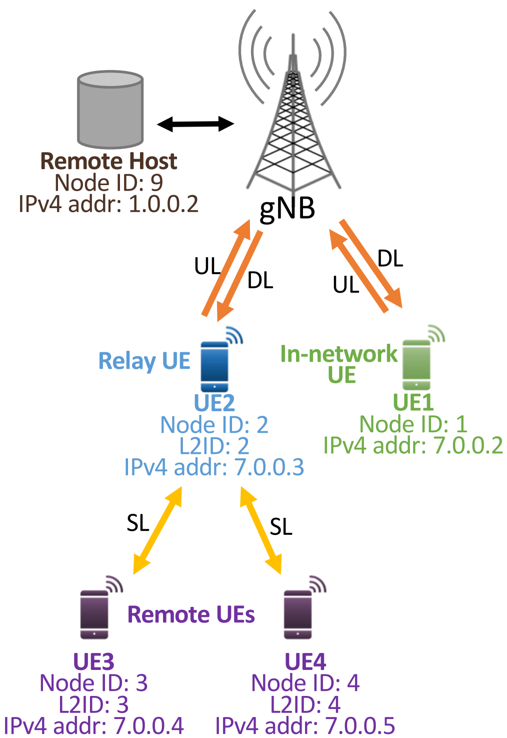
```
Flow 5 (7.0.0.4 -> 1.0.0.2) UDP
  Tx Packets: 499
  Tx Bytes:   63872
  TxOffered:  0.051098 Mbps
  Rx Packets: 322
  Rx Bytes:   41216
  Throughput: 0.032973 Mbps
  Mean delay:  15.643615 ms
Flow 6 (1.0.0.2 -> 7.0.0.5) UDP
  Tx Packets: 498
  Tx Bytes:   63744
  TxOffered:  0.050995 Mbps
  Rx Packets: 451
  Rx Bytes:   57728
  Throughput: 0.046182 Mbps
  Mean delay:  15.713995 ms
Flow 7 (1.0.0.2 -> 7.0.0.3) UDP
  Tx Packets: 498
  Tx Bytes:   63744
  TxOffered:  0.050995 Mbps
  Rx Packets: 498
  Rx Bytes:   63744
  Throughput: 0.050995 Mbps
  Mean delay:  1.146246 ms
Flow 8 (1.0.0.2 -> 7.0.0.2) UDP
  Tx Packets: 496
  Tx Bytes:   63488
  TxOffered:  0.050790 Mbps
  Rx Packets: 496
  Rx Bytes:   63488
  Throughput: 0.050790 Mbps
  Mean delay:  1.142618 ms
```

4

```
Number of packets relayed by the L3 UE-to-Network relays:
relayIp      srcIp->dstIp        srcLink->dstLink       nPackets
7.0.0.3      1.0.0.2->7.0.0.4       DL->SL              500
7.0.0.3      1.0.0.2->7.0.0.5       DL->SL              498
7.0.0.3      7.0.0.4->1.0.0.2       SL->UL              322
7.0.0.3      7.0.0.5->1.0.0.2       SL->UL              409
```

NR ProSe L3 UE-to-Network relay - Example

`nr-prose-l3-relay.cc`



Simulation output files (1/2):

```
-bash-4.2$ ls -1 output_nr-prose-l3-relay
default-nr-prose-l3-relay.db
default-nr-prose-l3-relay-flowMonitorOutput.txt
default-nr-prose-l3-relay-NrSlPc5SignallingPacketTrace.txt
default-nr-prose-l3-relay-NrSlRelayNasRxPacketTrace.txt
```

```
-bash-4.2$ cat  output_nr-prose-l3-relay/default-nr-prose-l3-relay-NrSlPc5SignallingPacketTrace.txt
time(s) TX/RX   srcL2Id dstL2Id msgType
2       TX      3       2       PROSE DIRECT LINK ESTABLISHMENT REQUEST
2       TX      4       2       PROSE DIRECT LINK ESTABLISHMENT REQUEST
2.00408 RX      4       2       PROSE DIRECT LINK ESTABLISHMENT REQUEST
2.00408 TX      2       4       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
2.00458 RX      3       2       PROSE DIRECT LINK ESTABLISHMENT REQUEST
2.00458 TX      2       3       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
2.00708 RX      2       4       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
2.01133 RX      2       3       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
```

```
-bash-4.2$ head -n 10 output_nr-prose-l3-relay/default-nr-prose-l3-relay-NrSlRelayNasRxPacketTrace.txt
time(s) nodeIp  srcIp   dstIp   srcLink dstLink
5.01637 7.0.0.3 1.0.0.2 7.0.0.4 DL      SL
5.02958 7.0.0.3 7.0.0.5 1.0.0.2 SL      UL
5.03513 7.0.0.3 1.0.0.2 7.0.0.4 DL      SL
5.04408 7.0.0.3 7.0.0.4 1.0.0.2 SL      UL
5.04788 7.0.0.3 1.0.0.2 7.0.0.5 DL      SL
5.04958 7.0.0.3 7.0.0.5 1.0.0.2 SL      UL
5.05513 7.0.0.3 1.0.0.2 7.0.0.4 DL      SL
5.06408 7.0.0.3 7.0.0.4 1.0.0.2 SL      UL
5.06788 7.0.0.3 1.0.0.2 7.0.0.5 DL      SL
```

NR ProSe L3 UE-to-Network relay - Example

nr-prose-l3-relay.cc

Simulation output files (2/2):

```
-bash-4.2$ ls -1 output_nr-prose-l3-relay
default-nr-prose-l3-relay.db
default-nr-prose-l3-relay-flowMonitorOutput.txt
default-nr-prose-l3-relay-NrSlPc5SignallingPacketTrace.txt
default-nr-prose-l3-relay-NrSlRelayNasRxPacketTrace.txt
```

DB Browser for SQLite - C:\Users\smg\Desktop\wns32023SimFiles\default-nr-prose-unicast-multi-link.db

- Tables (2)
  - pscchRxUePhy
  - psschRxUePhy

Table: psschRxUePhy

Filter in any column

	timeMs	cellId	rnti	bwpId	frame	subFrame	slot	txRnti	srcL2Id	dstL2Id	psschRbStart	psschRbLen	psschSymStart	psschSymLen	psschMcs	ndi	rv	tbSizeBytes	avrgSinr	minSinr	psschTbler	psschCorrupt	sci2Tbler	sci2Corrupt	SEED	RUN
1	2001.98214	0	0	1	200	1	3	4	4	2	0	10	1	12	14	1	0	348	2883.297...	1393.5346...	0	0	0	0	1	1
2	2002.23214	0	1	1	200	2	0	3	3	2	10	10	1	12	14	1	0	348	0....	0....	1	1	0.2635983	0	1	1
3	2004.73214	0	4	1	200	4	2	1	2	4	0	10	1	12	14	1	0	348	2883.297...	1393.5346...	0	0	0	0	1	1
4	2007.23214	0	1	1	200	7	0	3	3	2	0	10	1	12	14	0	3	348	2893.154...	2465.2901...	0.001	0	0	0	1	1
5	2011.98214	0	3	1	201	1	3	1	2	3	10	10	1	12	14	1	0	348	2294.743...	1988.7544...	0	0	0	0	1	1
6	5017.48214	0	3	1	501	7	1	1	2	3	0	20	1	12	14	1	0	704	1296.974...	994.37720...	0	0	0	0	1	1
7	5027.23214	0	1	1	502	7	0	4	4	2	0	20	1	12	14	1	0	704	1392.825...	696.76734...	0	0	0	0	1	1
8	5037.48214	0	3	1	503	7	1	1	2	3	0	20	1	12	14	1	0	704	1296.974...	994.37720...	0	0	0	0	1	1
9	5041.73214	0	1	1	504	1	2	3	3	2	0	20	1	12	14	1	0	704	1296.974...	994.37720...	0	0	0	0	1	1
10	5051.48214	0	1	1	505	1	1	4	4	2	0	20	1	12	14	0	1	704	1392.825...	696.76734...	0	0	0	0	1	1

L3 U2N Relay selection

(Integration direct discovery with L3 U2N relay)

NR ProSe relay selection: Overview

- The 5G ProSe remote UE triggers the relay selection procedure if is authorized to act as a remote UE towards a UE-to-network (U2N) relay UE and already obtained a list of UE-to-network relay UE candidate(s) fulfilling ProSe layer criteria and other lower layers criteria.

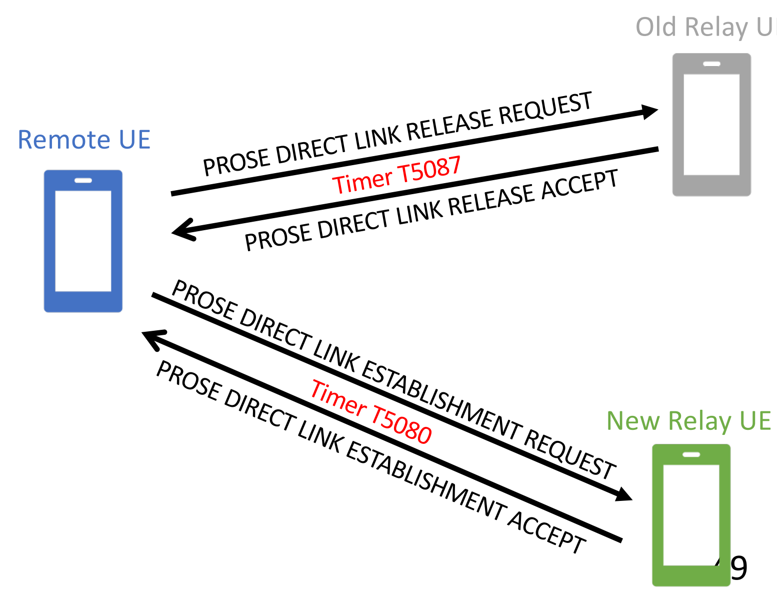
- A Reselection is triggered if:
  - the previously selected relay is no longer available; or
  - the relay configuration parameters have been updated and the relay no longer fulfil the selection requirements; or
  - the remote received a PROSE DIRECT LINK ESTABLISHMENT REJECT message or a PROSE DIRECT LINK RELEASE REQUEST from the relay; or
  - No response is obtained after the transmission of multiple PROSE DIRECT LINK ESTABLISHMENT REQUEST or PROSE DIRECT LINK KEEPALIVE REQUEST messages.

NR ProSe relay selection: Implementation

- Relay (re)selection is triggered if:
  - A new U2N relay UE is discovered
  - A new RSRP measurement is received

- Three relay selection algorithms are implemented:
  - *FirstAvailableRelay*: selecting the first available U2N relay UE in the list of discovered relays.
  - *RandomRelay*: selecting a U2N relay UE randomly from the list of discovered relays.
  - *MaxRsrpRelay* (3GPP-compliant): selecting the U2N relay UE that has the best RSRP measurement after fulfilling the threshold and hysteresis criteria.

- Relay (re)selection calls for:
  - The release of the direct link with the current U2N relay UE
  - The establishment of the direct link with the newly selected U2N relay UE
  - The bearers' reconfiguration by removing the old ones used for the released relay UE (this would be triggered by the RELEASE procedure) and adding new ones with the selected U2N relay (this would be triggered by the ESTABLISHMENT procedure).



NR ProSe relay selection: Example

`nr-prose-discovery-l3-relay-selection.cc`

Source code: src/nr/examples/nr-prose-examples/nr-prose-discovery-l3-relay-selection.cc

- Topology:
  - This example is composed of one gNB and 3 UEs: The first 2 UEs act as in-network L3 UE-to-Network relay UEs (which are attached to the gNB). The third UE acts as out-of-network remote UE.
  - All UEs are randomly deployed and follow a random walk mobility model.
  - The UEs will start performing NR discovery at random simulation times.
  - The relay selection algorithm can be selected in the scenario.

- Traffic:
  - Uplink and downlink traffic (of the same configuration) is flowing between the remote UE and the server.

- Trace outputs:
  - *NrSlPc5SignallingPacketTrace.txt*: log of the transmitted and received PC5 signaling messages used for the establishment and/or release of each ProSe unicast direct link.
  - *NrSlRelayNasRxPacketTrace.txt*: log of the packets received and routed by the NAS of the UE acting as L3 UE-to-Network UE.
  - *NrSlRelayDiscoveryTrace.txt*: to keep track of discovered relays.
  - *NrSlRelaySelectionTrace.txt*: to keep track of relay selection attempts.

NR ProSe relay selection: Example

nr-prose-discovery-l3-relay-selection.cc

Source code: src/nr/examples/nr-prose-examples/nr-prose-discovery-l3-relay-selection.cc

To be called in the scenario:

```
//Start discovery and relay selection
nrSlProseHelper->StartRemoteRelayConnection (remoteUeNetDev, startTimeRemote,
                                             relayUeNetDev, startTimeRelay,
                                             relayCodes, relayDestL2Ids,
                                             model, algorithm,
                                             tftRelay, bearerRelay);
```

```
/**
 * Start Relay discovery and link establishment betwwen relay and remote
 *
 * \param remoteDevices Net Devices of remote UEs
 * \param remoteTime when to start the discovery for remote UEs
 * \param relayDevices Net Devices of relay UEs
 * \param relayTime when to start the discovery for relay UEs
 * \param relayCodes relay codes to be announced
 * \param dstL2Ids destination layer 2 IDs to be associated with the relays
 * \param discoveryModel the discovery model considered: Model A or Model B
 * \param selectionAlgorithm the relay (re)selection algorithm considered
 * \param tft the traffic flow template to be used for relaying traffic
 * \param bearer EPS beraer to use for relaying traffic
 */
void StartRemoteRelayConnection (const NetDeviceContainer remoteDevices, const std::vector<Time> remoteTime,
                                 const NetDeviceContainer relayDevices, const std::vector<Time> relayTime,
                                 const std::vector<uint32_t> relayCodes, const std::vector<uint32_t> dstL2Ids,
                                 NrSlUeProse::DiscoveryModel discoveryModel, Ptr<NrSlUeProseRelaySelectionAlgorithm> selectionAlgorithm,
                                 Ptr<EpcTft> tft, EpsBearer bearer);
```

NR ProSe relay selection: Example

nr-prose-discovery-l3-relay-selection.cc

Running the scenario:

```
-bash-4.2$  mkdir output_nr-prose-discovery-l3-relay-selection-random
-bash-4.2$ ./ns3 run 'nr-prose-discovery-l3-relay-selection --relaySelectAlgorithm=RandomRelay'
 --cwd='output_nr-prose-discovery-l3-relay-selection-random'
```

Simulation standard output

```
UEs configuration:
 Number of Relay UEs = 2
 Number of Remote UEs = 1
IP configuration:
 Remote Host: 1.0.0.2
 In-network U2N relay UE: 7.0.0.2
 In-network U2N relay UE: 7.0.0.3
 Out-of-network remote UE: 7.0.0.4
Discovery configuration:
 UE 1: discovery start = 2.36525 s and discovery interval = 5 s
 UE 2: discovery start = 3.69174 s and discovery interval = 5 s
 UE 3: discovery start = 2.02281 s and discovery interval = 5 s
Remote traffic configuration:
 DL: 1.0.0.2 -> 7.0.0.4:1234 start time: 4 s, end time: 15 s
 UL: 7.0.0.4 -> 1.0.0.2:1236 start time: 4 s, end time: 15 s

/*********** Simulation done! ***********/

Number of packets relayed by the L3 UE-to-Network relays:
 relayIp       srcIp->dstIp       srcLink->dstLink            nPackets
 7.0.0.2       1.0.0.2->7.0.0.4       DL->SL           161
 7.0.0.2       7.0.0.4->1.0.0.2       SL->UL           155
 7.0.0.3       1.0.0.2->7.0.0.4       DL->SL           497
 7.0.0.3       7.0.0.4->1.0.0.2       SL->UL           465
```

Traffic is relayed through relay UE 1 to remote UE 3

Traffic is relayed through relay UE 2 to remote UE 3

Simulation output files:

```
-bash-4.2$ ls -1 output_nr-prose-discovery-l3-relay-selection-random
NrSlDiscoveryTopology.txt
NrSlDiscoveryTrace.txt
NrSlPc5SignallingPacketTrace.txt
NrSlRelayDiscoveryTrace.txt
NrSlRelayNasRxPacketTrace.txt
NrSlRelayRsrpTrace.txt
NrSlRelaySelectionTrace.txt
```

NR ProSe relay selection: Example

nr-prose-discovery-l3-relay-selection.cc

Relevant simulation output files:

```
-bash-4.2$ cat output_nr-prose-discovery-l3-relay-selection-random/NrSlRelayDiscoveryTrace.txt
Time (s)        RemoteL2ID       DiscoveredRelayL2ID      RelayCode        RSRP
2.3825642810    3       1        101     -inf
3.7090642810    3       2        102     -inf
7.3825642810    3       1        101     -87.0253988703
8.6950642810    3       2        102     -86.4161028690
12.3830642810   3       1        101     -85.0695562716
13.7040642810   3       2        102     -86.6393106766
```

```
-bash-4.2$ cat output_nr-prose-discovery-l3-relay-selection-random/NrSlPc5SignallingPacketTrace.txt
Time (s)        TX/RX   srcL2Id dstL2Id msgType
2.38256 TX      3       1       PROSE DIRECT LINK ESTABLISHMENT REQUEST
2.38856 RX      3       1       PROSE DIRECT LINK ESTABLISHMENT REQUEST
2.38856 TX      1       3       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
2.39406 RX      1       3       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
3.8     TX      3       1       PROSE DIRECT LINK RELEASE REQUEST
3.8     TX      3       2       PROSE DIRECT LINK ESTABLISHMENT REQUEST
3.81306 RX      3       2       PROSE DIRECT LINK ESTABLISHMENT REQUEST
3.81306 TX      2       3       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
3.81356 RX      3       1       PROSE DIRECT LINK RELEASE REQUEST
3.81356 TX      1       3       PROSE DIRECT LINK RELEASE ACCEPT
3.81956 RX      1       3       PROSE DIRECT LINK RELEASE ACCEPT
3.82406 RX      2       3       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
8.69506 TX      3       2       PROSE DIRECT LINK RELEASE REQUEST
8.69506 TX      3       1       PROSE DIRECT LINK ESTABLISHMENT REQUEST
8.70456 RX      3       2       PROSE DIRECT LINK RELEASE REQUEST
8.70456 TX      2       3       PROSE DIRECT LINK RELEASE ACCEPT
8.71506 RX      3       1       PROSE DIRECT LINK ESTABLISHMENT REQUEST
8.71506 TX      1       3       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
8.72006 RX      2       3       PROSE DIRECT LINK RELEASE ACCEPT
8.73406 RX      1       3       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
8.8     TX      3       1       PROSE DIRECT LINK RELEASE REQUEST
8.8     TX      3       2       PROSE DIRECT LINK ESTABLISHMENT REQUEST
8.80456 RX      3       2       PROSE DIRECT LINK ESTABLISHMENT REQUEST
8.80456 TX      2       3       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
8.81856 RX      3       1       PROSE DIRECT LINK RELEASE REQUEST
8.81856 TX      1       3       PROSE DIRECT LINK RELEASE ACCEPT
8.82006 RX      2       3       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
8.82506 RX      1       3       PROSE DIRECT LINK RELEASE ACCEPT
12.3831 TX      3       2       PROSE DIRECT LINK RELEASE REQUEST
12.3831 TX      3       1       PROSE DIRECT LINK ESTABLISHMENT REQUEST
12.3886 RX      3       2       PROSE DIRECT LINK RELEASE REQUEST
12.3886 TX      2       3       PROSE DIRECT LINK RELEASE ACCEPT
12.3941 RX      2       3       PROSE DIRECT LINK RELEASE ACCEPT
12.4001 RX      3       1       PROSE DIRECT LINK ESTABLISHMENT REQUEST
12.4001 TX      1       3       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
12.4146 RX      1       3       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
```

```
-bash-4.2$ cat output_nr-prose-discovery-l3-relay-selection-random/NrSlRelaySelectionTrace.txt
Time (s)        RemoteL2ID      CurrentRelayL2ID        NewRelayL2ID    NewRelayCode    NewRSRP
2.3825642810    3       0       1       101     -inf
2.4000000000    3       1       1       101     -87.0253988703
3.7090642810    3       1       1       101     -87.0253988703
3.8000000000    3       1       2       102     -86.4161028690
7.3825642810    3       2       2       102     -86.4161028690
7.4000000000    3       2       2       102     -86.4161028690
8.6950642810    3       2       1       101     -85.0695562716
8.8000000000    3       1       2       102     -86.6393106766
12.3830642810   3       2       1       101     -85.0695562716
13.7040642810   3       1       1       101     -86.7368959789
13.8000000000   3       1       1       101     -86.7368959789
```

Already connected to the same relay

First time selecting a relay UE after a successful discovery

A relay selection may be triggered by the discovery of an eligible relay or the reception of an RSRP measurement corresponding to a discovery procedure.

Once a new relay UE is **randomly** selected, a release procedure is triggered with the previously selected relay and an establishment procedure is started for the newly selected relay.

NR ProSe relay selection: Example

nr-prose-discovery-l3-relay-selection.cc

Running the scenario with **a different relay selection algorithm**:

```
-bash-4.2$ mkdir output_nr-prose-discovery-l3-relay-selection-maxRsrp
-bash-4.2$ ./ns3 run 'nr-prose-discovery-l3-relay-selection --relaySelectAlgorithm=MaxRsrpRelay'
 --cwd='output_nr-prose-discovery-l3-relay-selection-maxRsrp'
```

Simulation standard output:

```
UEs configuration:
 Number of Relay UEs = 2
 Number of Remote UEs = 1
IP configuration:
 Remote Host: 1.0.0.2
 In-network U2N relay UE: 7.0.0.2
 In-network U2N relay UE: 7.0.0.3
 Out-of-network remote UE: 7.0.0.4
Discovery configuration:
 UE 1: discovery start = 2.36525 s and discovery interval = 5 s
 UE 2: discovery start = 3.69174 s and discovery interval = 5 s
 UE 3: discovery start = 2.02281 s and discovery interval = 5 s
Remote traffic configuration:
 DL: 1.0.0.2 -> 7.0.0.4:1234 start time: 4 s, end time: 15 s
 UL: 7.0.0.4 -> 1.0.0.2:1236 start time: 4 s, end time: 15 s

/*********** Simulation done! ***********/

Number of packets relayed by the L3 UE-to-Network relays:
 relayIp       srcIp->dstIp       srcLink->dstLink          nPackets
 7.0.0.2       1.0.0.2->7.0.0.4       DL->SL          383
 7.0.0.2       7.0.0.4->1.0.0.2       SL->UL          332
 7.0.0.3       1.0.0.2->7.0.0.4       DL->SL          276
 7.0.0.3       7.0.0.4->1.0.0.2       SL->UL          274
```

Traffic is relayed through relay UE 1 (7.0.0.2)

Traffic is relayed through relay UE 2 (7.0.0.3)

Simulation output files:

```
-bash-4.2$ ls -1 output_nr-prose-discovery-l3-relay-selection-maxRsrp
NrSlDiscoveryTopology.txt
NrSlDiscoveryTrace.txt
NrSlPc5SignallingPacketTrace.txt
NrSlRelayDiscoveryTrace.txt
NrSlRelayNasRxPacketTrace.txt
NrSlRelayRsrpTrace.txt
NrSlRelaySelectionTrace.txt
```

NR ProSe relay selection: Example

nr-prose-discovery-l3-relay-selection.cc

Relevant simulation output files:

```
bash-4.2$ cat output_nr-prose-discovery-l3-relay-selection-maxRsrp/NrSlRelayDiscoveryTrace.txt
Time (s)          RemoteL2ID      DiscoveredRelayL2ID     RelayCode       RSRP
2.3825642810    3       1       101     -inf
3.7090642810    3       2       102     -inf
7.3825642810    3       1       101     -87.0253988703
8.7050642810    3       2       102     -86.3516533493
12.3840642810   3       1       101     -85.0695562716
13.6940642810   3       2       102     -86.4677220751
```

* Already connected to the best relay

```
bash-4.2$ cat output_nr-prose-discovery-l3-relay-selection-maxRsrp/NrSlRelaySelectionTrace.txt
Time (s)        RemoteL2ID      CurrentRelayL2ID        NewRelayL2ID    NewRelayCode    NewRSRP
2.4000000000    3       0       1       101     -87.0253988703
3.7090642810    3       1       1       101     -87.0253988703 } *
3.8000000000    3       1       2       102     -86.3516533493
7.3825642810    3       2       2       102     -86.3516533493 } *
7.4000000000    3       2       1       101     -85.0695562716
8.7050642810    3       1       1       101     -85.0695562716 ]
8.8000000000    3       1       1       101     -85.0695562716 |
12.3840642810   3       1       1       101     -85.0695562716 | *
12.4000000000   3       1       1       101     -86.2341358897 |
13.6940642810   3       1       1       101     -86.2341358897 ]
13.8000000000   3       1       2       102     -85.9122474679
```

```
bash-4.2$ cat output_nr-prose-discovery-l3-relay-selection-maxRsrp/NrSlPc5SignallingPacketTrace.txt
Time (s)        TX/RX   srcL2Id dstL2Id msgType
2.4     TX      3       1       PROSE DIRECT LINK ESTABLISHMENT REQUEST
2.40456 RX      3       1       PROSE DIRECT LINK ESTABLISHMENT REQUEST
2.40456 TX      1       3       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
2.41306 RX      1       3       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
3.8     TX      3       1       PROSE DIRECT LINK RELEASE REQUEST
3.8     TX      3       2       PROSE DIRECT LINK ESTABLISHMENT REQUEST
3.81306 RX      3       2       PROSE DIRECT LINK ESTABLISHMENT REQUEST
3.81306 TX      2       3       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
3.81306 RX      3       1       PROSE DIRECT LINK RELEASE REQUEST
3.81356 TX      1       3       PROSE DIRECT LINK RELEASE ACCEPT
3.81956 RX      1       3       PROSE DIRECT LINK RELEASE ACCEPT
3.82406 RX      2       3       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
7.4     TX      3       2       PROSE DIRECT LINK RELEASE REQUEST
7.4     TX      3       1       PROSE DIRECT LINK ESTABLISHMENT REQUEST
7.41856 RX      3       1       PROSE DIRECT LINK ESTABLISHMENT REQUEST
7.41856 TX      1       3       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
7.41906 RX      3       2       PROSE DIRECT LINK RELEASE REQUEST
7.41906 TX      2       3       PROSE DIRECT LINK RELEASE ACCEPT
7.42456 RX      2       3       PROSE DIRECT LINK RELEASE ACCEPT
7.43406 RX      1       3       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
13.8    TX      3       1       PROSE DIRECT LINK RELEASE REQUEST
13.8    TX      3       2       PROSE DIRECT LINK ESTABLISHMENT REQUEST
13.8041 RX      3       2       PROSE DIRECT LINK ESTABLISHMENT REQUEST
13.8041 TX      2       3       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
13.8091 RX      2       3       PROSE DIRECT LINK ESTABLISHMENT ACCEPT
13.8191 RX      3       1       PROSE DIRECT LINK RELEASE REQUEST
13.8191 TX      1       3       PROSE DIRECT LINK RELEASE ACCEPT
13.8236 RX      1       3       PROSE DIRECT LINK RELEASE ACCEPT
```

First time selecting a relay UE (Relay L2ID 1) after a successful discovery

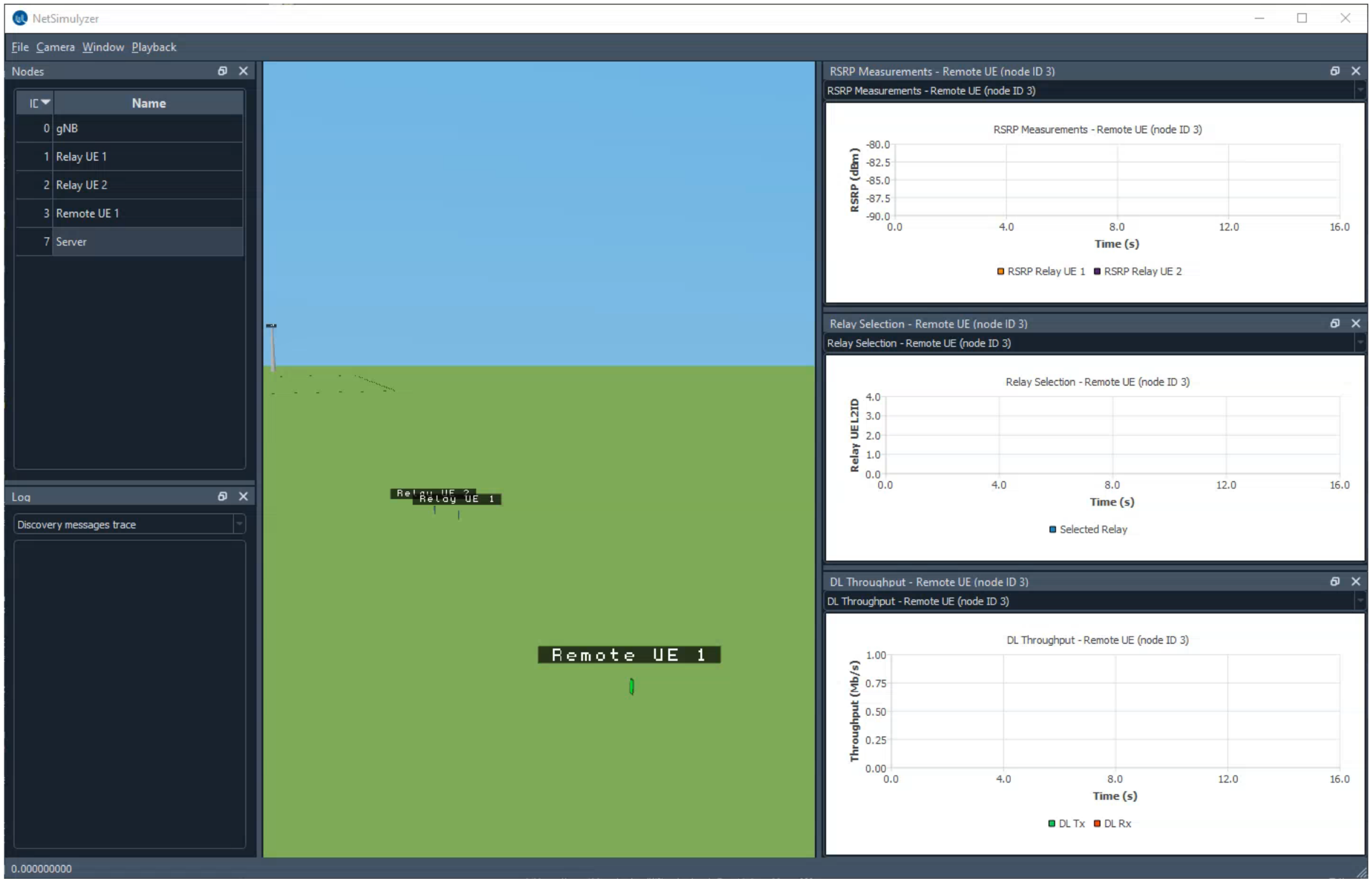
Relay L2ID 2 shows a **better RSRP measurement, relay reselection is triggered:** release of the connection with Relay L2ID 1 and establishment of the connection with the Relay L2ID 2

Relay L2ID 1 has a better RSRP measurement again, relay reselection is triggered

Relay L2ID 2 has a better RSRP measurement again, relay reselection is triggered

NR ProSe relay selection: Example

Visualization with NetSimulyzer



Future work

- Near term:
  - Integrate the recent updates made to the NR Sidelink Sensing, Scheduling, and HARQ feedback models to the NR ProSe module.
  - Complete the release of the NR ProSe code.

- Long term:
  - Implement the 3GPP Rel-18 one-hop NR ProSe UE-to-UE (U2U) relay functionality.
  - Investigate and prototype multi-hop NR ProSe U2U relay functionality and how to incorporate it into the 5G NR infrastructure.

STAY IN TOUCH

CONTACT US

aziza.benmosbah@nist.gov
samantha.gamboaquintiliani@nist.gov

https://www.nist.gov/programs-projects/public-safety-communications