Proximity Services (ProSe) Support for 5G NR Simulations

WNS3 2023 – June 2023
Aziza Ben Mosbah\textsuperscript{1,2} and Samantha Gamboa\textsuperscript{1,2}

\textsuperscript{1} Associate, Wireless Networks Division (WND) - National Institute of Standards and Technology (NIST) - Gaithersburg, Maryland, USA
\textsuperscript{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

NR Proximity Services (ProSe)

Direct Communication
- Broadcast, Groupcast, Unicast
- UE-based Relays
- UE-to-Network, UE-to-UE

V2X Communication
- Broadcast, Groupcast, Unicast

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 Proximity Services (ProSe)

- Direct Discovery
  - Peers, Members, Relays

- Direct Communication
  - Broadcast, Groupcast, Unicast

- UE-based Relays
  - UE-to-Network, UE-to-UE

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
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 simulation typical flow:
During scenario configuration:
- Configure ProSe functionalities using ProSe helper
- Internally the helper installs ProSe layer and configures functionality
During simulation:
- ProSe layer controls functionalities
  - Context creation and management
  - SL bearer creation/configuration
  - Procedures execution
How to get started:
1. Get the code and move into psc-ns3 directory:
   ```bash
git clone "https://github.com/usnistgov/psc-ns3.git" -b wns3-2023-nr-prose-preview
cd psc-ns3
   ```

2. Setup ns3:
   ```bash
./ns3 configure --enable-examples
./ns3
   ```

3. Run examples:
   • Running a ProSe example:
     ```bash
     ./ns3 run 'exampleName'
     ```
     e.g.,
     ```bash
     ./ns3 run 'nr-prose-discovery-l3-relay'
     ```
   • Running a a ProSe example with command line parameters:
     ```bash
     ./ns3 run 'exampleName --param1=param1Value --param2=param2Value'
     ```
     e.g.,
     ```bash
     ./ns3 run 'nr-prose-discovery-l3-relay --discInterval=4'
     ```
   • Running a a ProSe example with command line parameters and specify (existent) output directory:
     ```bash
     ./ns3 run 'exampleName --param1=param1Value --param2=param2Value' --cwd='outputDirectory'
     ```
     e.g.,
     ```bash
     ./ns3 run 'nr-prose-discovery-l3-relay --discInterval=4' --cwd='output_nr-prose-discovery-l3-relay-4s'
     ```
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**".

![Diagram](image)
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.
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).
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

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

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

The UE can play multiple roles depending on the discovery model used and on which side (transmitting or receiving) it is.
NR ProSe direct discovery: Peer Discovery

These functions from the \textit{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).

```c
/**
 * 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

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

```cpp
// 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, &NrsIProseHelper::StartDiscovery, nrsIProseHelper, ueVoiceNetDev.Get (i),
                          announcePayloads[ueVoiceNetDev.Get (i)], announceDstL2IdsMap[ueVoiceNetDev.Get (i)], NrsIProse::Announcing);
    Simulator::Schedule (startDiscTime, &NrsIProseHelper::StartDiscovery, nrsIProseHelper, ueVoiceNetDev.Get (i),
                          monitorPayloads[ueVoiceNetDev.Get (i)], monitorDstL2IdsMap[ueVoiceNetDev.Get (i)], NrsIProse::Monitoring);

    // Stop announcing
    Simulator::Schedule (stopDiscTime, &NrsIProseHelper::StopDiscovery, nrsIProseHelper, ueVoiceNetDev.Get (i),
                         announcePayloads[ueVoiceNetDev.Get (i)], NrsIProse::Announcing);

    // Stop monitoring
    Simulator::Schedule (stopDiscTime, &NrsIProseHelper::StopDiscovery, nrsIProseHelper, ueVoiceNetDev.Get (i),
                         monitorPayloads[ueVoiceNetDev.Get (i)], NrsIProse::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

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 -I output_nr-prose-discovery/
default-nr-prose-discovery.db
Nrs1DiscoveryTrace.txt
```

The \textit{Nrs1DiscoveryTrace.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).

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).

```c
/**
 * 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

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

- 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 -l output_nr-prose-discovery-l3-relay
default_nr-prose-discovery-relay.db
NrsI1DiscoveryTrace.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).

```
Time (s)  TX/RX  SenderL2Id  receiverL2Id  DiscType  DiscModel  ContentType  Content
0.000000000  TX   2        500         Restricted  ModelB  RelaySolicitation  5:2;1:2;0
0.0008321400  RX   1        2                ModelB  RelaySolicitation  5:2;1:2;0
2.0016321400  TX   2        1                ModelB  RelayResponse   5:1:1:1:0
2.0045821400  RX   1        2                ModelB  RelayResponse   5:1:1:1:0
4.0000000000  TX   2        500         Restricted  ModelB  RelaySolicitation  5:2;1:2;0
4.0016321400  RX   1        2                ModelB  RelaySolicitation  5:2;1:2;0
4.0045821400  RX   1        2                ModelB  RelayResponse   5:1:1:1:0
```

Successful relay discovery

Failed relay discovery due to propagation loss
NR ProSe direct discovery: Relay Discovery - Example

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 -l 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
```

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>TX/RX</th>
<th>SenderL2ID</th>
<th>ReceiverL2ID</th>
<th>DiscType</th>
<th>DiscModel</th>
<th>ContentType</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0000000000</td>
<td>TX</td>
<td>2</td>
<td>500</td>
<td>Restricted</td>
<td>ModelB</td>
<td>RelaySolicitation</td>
<td>5:2:1:2:0</td>
</tr>
<tr>
<td>2.001321400</td>
<td>TX</td>
<td>1</td>
<td>2</td>
<td>Restricted</td>
<td>ModelB</td>
<td>RelaySolicitation</td>
<td>5:2:1:2:0</td>
</tr>
<tr>
<td>2.001321400</td>
<td>RX</td>
<td>1</td>
<td>2</td>
<td>Restricted</td>
<td>ModelB</td>
<td>RelayResponse</td>
<td>5:1:1:1:0</td>
</tr>
<tr>
<td>2.0045821400</td>
<td>RX</td>
<td>1</td>
<td>2</td>
<td>Restricted</td>
<td>ModelB</td>
<td>RelayResponse</td>
<td>5:1:1:1:0</td>
</tr>
<tr>
<td>6.0000000000</td>
<td>TX</td>
<td>2</td>
<td>500</td>
<td>Restricted</td>
<td>ModelB</td>
<td>RelaySolicitation</td>
<td>5:2:1:2:0</td>
</tr>
<tr>
<td>6.001321400</td>
<td>RX</td>
<td>1</td>
<td>2</td>
<td>Restricted</td>
<td>ModelB</td>
<td>RelaySolicitation</td>
<td>5:2:1:2:0</td>
</tr>
<tr>
<td>6.0045821400</td>
<td>RX</td>
<td>1</td>
<td>2</td>
<td>Restricted</td>
<td>ModelB</td>
<td>RelayResponse</td>
<td>5:1:1:1:0</td>
</tr>
</tbody>
</table>
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

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

• 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)
• 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

Direct link configuration:

<table>
<thead>
<tr>
<th>Link</th>
<th>Initiating UE</th>
<th>Target UE</th>
</tr>
</thead>
<tbody>
<tr>
<td>UE0 &lt;-&gt; UE1</td>
<td>UE0</td>
<td>UE1</td>
</tr>
<tr>
<td>UE0 &lt;-&gt; UE2</td>
<td>UE0</td>
<td>UE2</td>
</tr>
<tr>
<td>UE1 &lt;-&gt; UE2</td>
<td>UE1</td>
<td>UE2</td>
</tr>
</tbody>
</table>

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.,
NR ProSe unicast communication - Example

Traffic configuration:

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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>udpPacketSize</td>
<td>200 Bytes</td>
</tr>
<tr>
<td>dataRate</td>
<td>16 kb/s</td>
</tr>
<tr>
<td>startTrafficTime</td>
<td>3 s</td>
</tr>
</tbody>
</table>

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

CBR: Constant Bit Rate
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: 3.00853 s, end time: 20 s
System total Tx packets = 507
System total Rx packets = 507
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 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
```

Node ID: 0
L2ID: 1
IPv4 addr: 7.0.0.2

Node ID: 1
L2ID: 2
IPv4 addr: 7.0.0.3

Node ID: 2
L2ID: 3
IPv4 addr: 7.0.0.4
NR ProSe unicast communication - Example

**Simulation output files:**

```
-bash:4.2$ ls 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
```

<table>
<thead>
<tr>
<th>time(s)</th>
<th>nodeId</th>
<th>TX/RX</th>
<th>srcL2Id</th>
<th>dstL2Id</th>
<th>msgType</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>TX</td>
<td>1</td>
<td>2</td>
<td>PROSE DIRECT LINK ESTABLISHMENT REQUEST</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>TX</td>
<td>1</td>
<td>3</td>
<td>PROSE DIRECT LINK ESTABLISHMENT REQUEST</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>TX</td>
<td>2</td>
<td>3</td>
<td>PROSE DIRECT LINK ESTABLISHMENT REQUEST</td>
</tr>
<tr>
<td>2.00183</td>
<td>2</td>
<td>RX</td>
<td>1</td>
<td>3</td>
<td>PROSE DIRECT LINK ESTABLISHMENT REQUEST</td>
</tr>
<tr>
<td>2.00183</td>
<td>2</td>
<td>RX</td>
<td>1</td>
<td>3</td>
<td>PROSE DIRECT LINK ESTABLISHMENT REQUEST</td>
</tr>
<tr>
<td>2.00183</td>
<td>2</td>
<td>RX</td>
<td>3</td>
<td>1</td>
<td>PROSE DIRECT LINK ESTABLISHMENT ACCEPT</td>
</tr>
<tr>
<td>2.00483</td>
<td>2</td>
<td>RX</td>
<td>2</td>
<td>3</td>
<td>PROSE DIRECT LINK ESTABLISHMENT REQUEST</td>
</tr>
<tr>
<td>2.00483</td>
<td>2</td>
<td>RX</td>
<td>2</td>
<td>3</td>
<td>PROSE DIRECT LINK ESTABLISHMENT REQUEST</td>
</tr>
<tr>
<td>2.00633</td>
<td>0</td>
<td>RX</td>
<td>3</td>
<td>1</td>
<td>PROSE DIRECT LINK ESTABLISHMENT ACCEPT</td>
</tr>
<tr>
<td>2.00708</td>
<td>1</td>
<td>RX</td>
<td>1</td>
<td>1</td>
<td>PROSE DIRECT LINK ESTABLISHMENT ACCEPT</td>
</tr>
<tr>
<td>2.00708</td>
<td>1</td>
<td>RX</td>
<td>2</td>
<td>1</td>
<td>PROSE DIRECT LINK ESTABLISHMENT ACCEPT</td>
</tr>
<tr>
<td>2.01133</td>
<td>0</td>
<td>RX</td>
<td>2</td>
<td>1</td>
<td>PROSE DIRECT LINK ESTABLISHMENT ACCEPT</td>
</tr>
</tbody>
</table>
```
NR ProSe unicast communication - Example

Node ID: 0
L2ID: 1
IPv4 addr: 7.0.0.2

Node ID: 1
L2ID: 2
IPv4 addr: 7.0.0.3

Node ID: 2
L2ID: 3
IPv4 addr: 7.0.0.4

Simulation output files:

```
bash-4.2$ ls -l 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-NrslPcSSignallingPacketTrace.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
```
Simulation output files:

```bash
-bash-4.2$ ls
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-NrSlPCSSignallingPacketTrace.txt
```

DB Browser for SQLite (Windows/Linux/MacOS) available in: [https://sqlitebrowser.org/dl/](https://sqlitebrowser.org/dl/)
NR ProSe unicast communication - Example

nr-prose-unicast-multi-link.cc

Running the scenario with **bidirectional traffic**:

![Simulation output files]

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

Simulation output files:
- default-nr-prose-unicast-multi-link.bidirectional.db
- default-nr-prose-unicast-multi-link-flowMonitorOutput.txt
- default-nr-prose-unicast-multi-link-No3PsSignallingPacketTrace.txt

Simulation standard output

Traffic flows:
- Flow 1 (7.0.0.3 -> 7.0.0.2) UDP
  - Tx Packets: 169
  - Tx Bytes: 38532
  - Rx Packets: 169
  - Rx Bytes: 38532
  - Rx Data rate: 18.132706 kbps
  - Mean delay: 10.387269 ms

- Flow 2 (7.0.0.4 -> 7.0.0.3) UDP
  - Tx Packets: 169
  - Tx Bytes: 38532
  - Rx Packets: 169
  - Rx Bytes: 38532
  - Rx Data rate: 18.132706 kbps
  - Mean delay: 10.387269 ms

- Flow 3 (7.0.0.4 -> 7.0.0.2) UDP
  - Tx Packets: 169
  - Tx Bytes: 38532
  - Rx Packets: 169
  - Rx Bytes: 38532
  - Rx Data rate: 18.132706 kbps
  - Mean delay: 10.387269 ms

System total Tx packets = 1014
System total Rx packets = 978
System total Tx bits = 1622400
System total Rx bits = 1564800
System average thpt = 92.0471 kbps

Traffic flows statistics:
- Flow 1 (7.0.0.3 -> 7.0.0.2) UDP
  - Tx Packets: 169
  - Tx Bytes: 38532
  - Rx Packets: 169
  - Rx Bytes: 38532
  - Rx Data rate: 18.132706 kbps
  - Mean delay: 10.387269 ms

- Flow 2 (7.0.0.4 -> 7.0.0.3) UDP
  - Tx Packets: 169
  - Tx Bytes: 38532
  - Rx Packets: 169
  - Rx Bytes: 38532
  - Rx Data rate: 18.132706 kbps
  - Mean delay: 10.387269 ms

- Flow 3 (7.0.0.4 -> 7.0.0.2) UDP
  - Tx Packets: 169
  - Tx Bytes: 38532
  - Rx Packets: 169
  - Rx Bytes: 38532
  - Rx Data rate: 18.132706 kbps
  - Mean delay: 10.387269 ms

Simulation output files:
- default-nr-prose-unicast-multi-link.bidirectional.db
- default-nr-prose-unicast-multi-link-flowMonitorOutput.txt
- default-nr-prose-unicast-multi-link-No3PsSignallingPacketTrace.txt
NR ProSe L3 UE-to-Network relay
NR ProSe L3 UE-to-Network relay - Overview

**Layer 3 Architecture**
- **APP**: Application
- **PDU layer**
- **SDAP**: Service Data Adaptation Protocol
- **PDCP**: Packet Data Convergence Protocol
- **RLC**: Radio Link Control
- **MAC**: Media Access Control
- **PHY**: Physical Layer

**Layer 2 Architecture**
- **APP**: Application
- **PDU layer**
- **SDAP**: Service Data Adaptation Protocol
- **PDCP**: Packet Data Convergence Protocol
- **RLC**: Radio Link Control
- **MAC**: Media Access Control
- **PHY**: Physical Layer
- **Adaptation**: Adaptation Layer

**Remote UE**
- **U2N Relay UE**
- **Core Network**
- **gNB**
- **App Server**

**Ns-3 implementation focus**
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

ns-3 model

4a. Standard procedure
4b. Configured from the scenario
5. Simplified: IP addresses are exchanged during step 4
6. Simple model: Remote UE switches from NW bearer to SL bearer with Relay UE – No QoS management
7. Ideal: Function call to the LTE PGW corresponding function

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

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

**Topology:**

Remote Host --- gNB --- Relay UE

UL  DL  UL  DL

In-network UE

Remote UEs

SL

UE1

Remote UE uses both BWPs

Some special config is needed to do SL only in BWP1

**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

Remote UEs configure both BWPs but SL only uses BWP1

Source code:
```c
/* The configured spectrum division is: */
/* ------------------------------------ */
/* ----------- Band ------------------- */
/* ----------- CC0 ------------------- */
/* ----------- BWP0 ------------------- */
/* ----------- BWP1 ------------------- */

// Install both BWPs on U2N relays
NetDeviceContainer inNetUeNetDev = nrHelper->InstallUeDevice (inNetUeNodes, inNetBwp);
NetDeviceContainer enbNetDev = nrHelper->InstallNbDevice (enbNodes, inNetBwp);

// Install both BWPs on remote UEs
NetDeviceContainer remoteUeNetDev = nrHelper->InstallUeDevice (remoteUeNodes, allBwp);

// For U2N relay UEs we need to modify some parameters to configure *only*
// /BWP0 on the relay for SL and avoid MAC problems
// 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
```
NR ProSe L3 UE-to-Network relay - Example

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

Network connections:

```cpp
634  //Attach in-network UEs to the closest gNB
635  nrHelper->AttachToClosestEnb (inNetUeNetDev, enbNetDev);
655  //Attach U2N relay UEs to the closest gNB
656  nrHelper->AttachToClosestEnb (relayUeNetDev, enbNetDev);
```

ProSe layer configuration:

```cpp
677  //********** Configure ProSe layer in the UEs that will do SL **********/
678  //Create ProSe helper
679  Ptr<NrSlProseHelper> nrSlProsehelper = CreateObject <NrSlProseHelper> () ;
680  nrSlProseHelper->SetEpcHelper (epcHelper);
682  //Install ProSe layer and corresponding SAPs in the UEs
683  nrSlProseHelper->PrepareUsForProse (relayUeNetDev);
684  nrSlProseHelper->PrepareUsForProse (remoteUeNetDev);
688  //Configure ProSe Unicast parameters. At the moment it only instruct the MAC
689  //layer (and PHY therefore) to monitor packets directed the UE's own Layer 2 ID
690  nrSlProseHelper->PrepareUsForUnicast (relayUeNetDev);
691  nrSlProseHelper->PrepareUsForUnicast (remoteUeNetDev);
693  //Configure the value of timer T5000 (Prose Direct Link Establishment Request Retransmission)
694  //to a lower value than the standard (8.0 s) to speed connection in shorter simulation time
695  Config::SetDefault ("ns3::NrSlUeProseDirectLink::T5000", TimeValue (Seconds (2.0)));
697  //********** END Configure ProSe layer in the UEs that will do SL **********/
```
NR ProSe L3 UE-to-Network relay - Example

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

### L3 U2N Relay configuration (1/2)

```cpp
// 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;
quciRelay = EpsBearer::GBR_CONV_VOICE;
EpsBearer bearerRelay (qciRelay);

// Apply the configuration on the devices acting as relay UEs
nsIProseHelper->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

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

L3 U2N Relay configuration (2/2)

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

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:

```cpp
// Applications configuration
uint32_t packetSizeDLU1 = 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 (), dIPort));
serverApps.Add (dlPacketSinkHelper.Install (remoteUeNodes::Get (u)));

UdpClientHelper dlClient (ue1::GetAddress (u), dIPort);
dlClient.SetAttribute ("PacketSize", Uint32Value (packetSizeDLU1));
dlClient.SetAttribute ("Interval", TimeValue (Seconds (1.0 / lambdaDL)));
dlClient.SetAttribute ("MaxPackets", Uint32Value (0xffffffff));
ApplicationContainer dlApp = dlClient.Install (remoteHost);
appStartTime = Seconds (trafficStartTime + startTimeRnd - GetValue ());
dlApp.Start (appStartTime);
clientApps.Add (dlApp);

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

UdpClientHelper ulClient (remoteHostAddr, uIPort);
ulClient.SetAttribute ("PacketSize", Uint32Value (packetSizeUL));
ulClient.SetAttribute ("Interval", TimeValue (Seconds (1.0 / lambdaUL)));
ulClient.SetAttribute ("MaxPackets", Uint32Value (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

**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.

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

Running the scenario:

```
-bash-4.2$ mkdir output.nr-prose-l3-relay
-bash-4.2$ export HOME="/home/" nr-prose-l3-relay --cwd="output.nr-prose-l3-relay"
```

Simulation standard output:

```
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: 7.0.0.3 -> 7.0.0.3:101 start time: 5.05273 s, end time: 15 s
UL: 7.0.0.3 -> 7.0.0.3:201 start time: 5.00007 s, end time: 15 s
DL: 1.0.0.2 -> 7.0.0.4:102 start time: 5.04144 s, end time: 15 s
UL: 7.0.0.4 -> 1.0.0.4:203 start time: 5.0393 s, end time: 15 s
UL: 7.0.0.4 -> 1.0.0.4:203 start time: 5.02536 s, end time: 15 s

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->7.0.0.2 0 0 0
7.0.0.3 1.0.0.2->7.0.0.2 DL->SL 500
7.0.0.3 7.0.0.4->1.0.0.2 SL->UL 322
7.0.0.3 7.0.0.4->1.0.0.2 DL->SL 400
7.0.0.3 7.0.0.4->1.0.0.2 SL->UL 409
```
NR ProSe L3 UE-to-Network relay - Example

Simulation output files (1/2):

- bash-4.2$ ls 2 output_nr-prose-l3-relay
  default_nr-prose-l3-relay.db
  default_nr-prose-l3-relay-flowMonitorOutput.txt
- bash-4.2$ cat output_nr-prose-l3-relay/default_nr-prose-l3-relay-NrSlPc5SignallingPacketTrace.txt
  time(s) nodeID srcIP dstIP srcL2ID dstL2ID msgType
  0.01617 7.0.0.3 1.0.0.2 7.0.0.4 1.0.0.4 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
- bash-4.2$ head -n 10 output_nr-prose-l3-relay/default_nr-prose-l3-relay-NrSlRelayNasRxPacketTrace.txt
  time(s) nodeID srcIP dstIP srcL2ID dstL2ID msgType
  0.01617 7.0.0.3 1.0.0.2 7.0.0.4 1.0.0.4 DL SL
NR ProSe L3 UE-to-Network relay - Example

Simulation output files (2/2):

```
bash-4.2$ ls \*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\32023SimFiles\default-nr-prose-unicast-multi-link.db
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

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:

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

/*** Start Relay discovery and link establishment between 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 bearer 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

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'
```

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:

<table>
<thead>
<tr>
<th>srcIp</th>
<th>dstIp</th>
<th>relType</th>
<th>npackets</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.0.0.2</td>
<td>1.0.0.2 - 7.0.0.4</td>
<td>DL-&gt;SL</td>
<td>161</td>
</tr>
<tr>
<td>7.0.0.2</td>
<td>7.0.0.0 - 1.0.0.2</td>
<td>SL-&gt;UL</td>
<td>155</td>
</tr>
<tr>
<td>7.0.0.3</td>
<td>1.0.0.2 - 7.0.0.4</td>
<td>DL-&gt;SL</td>
<td>497</td>
</tr>
<tr>
<td>7.0.0.3</td>
<td>7.0.0.4 - 1.0.0.2</td>
<td>SL-&gt;UL</td>
<td>465</td>
</tr>
</tbody>
</table>
```

Simulation output files:

```
-bash-4.2$ ls output.nr-prose-discovery-l3-relay-selection-random
Nrs1DiscoveryTopology.txt
Nrs1DiscoveryTrace.txt
Nrs1PcSsignalingPacketTrace.txt
Nrs1RelayDiscoveryTrace.txt
Nrs1RelayNasRxPacketTrace.txt
Nrs1RelayRsrpTrace.txt
Nrs1RelaySelectionTrace.txt
```

Traffic is relayed through relay UE 1 to remote UE 3
Traffic is relayed through relay UE 2 to remote UE 3
NR ProSe relay selection: Example

Relevant simulation output files:

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.

Already connected to the same relay
NR ProSe relay selection: Example

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'
```

Simulation standard output:

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
```

- Ns3DiscoveryTopolgy.txt
- Ns3DiscoveryTrace.txt
- NrSlPcsSignallingPacketTrace.txt
- NrSlRelayDiscoveryTrace.txt
- NrSlRelayAs RxPacketTrace.txt
- NrSlRelayRsrpTrace.txt
- NrSlRelaySelectionTrace.txt
NR ProSe relay selection: Example

Relevant simulation output files:

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