Recent Updates to NR Sidelink Sensing, Scheduling, and HARQ Models

Tom Henderson and Samantha Gamboa

ns-3 Annual Meeting
June 26, 2023

UNIVERSITY *of* WASHINGTON

License

- The contents of this tutorial are copyright and licensed by the authors under Creative Commons CC-BY-SA 4.0 license
- Figures that have been copied from other sources are not covered by this CC-BY-SA 4.0 license, but are instead copyright by the various rights holders, and are reused herein under the fair use doctrine of U.S. copyright law

Schedule

- 9am-10:30: Introduction and recent improvements to NR V2X Mode 2 simulation models
  - Presenters: Tom Henderson and Samantha Gamboa
- *10:30-11: Break*
- 11-12:30: NR C-V2X Mode 2 Resource Allocation & ns-3 Implementation
  - Presenters: Liu Cao and Collin Brady
- *12:30-1:30: Lunch*
- 1:30-3:00: Proximity Services (ProSe) Support for 5G NR Simulations
  - Presenters: Samantha Gamboa and Aziza Ben-Mosbah
- *3:00-3:30: Break*
- 3:30-5:30: Panel on next steps for ns-3 sidelink and 3GPP Release 19 standardization

Outline of ns-3 Tutorials

- Introduction and recent improvements to NR V2X Mode 2 simulation models
  - NR V2X/sidelink models for ns-3
  - Example-driven tutorial on sidelink data-plane operation at the NR MAC sublayer (sensing, scheduling, and HARQ)
- NR C-V2X Mode 2 Resource Allocation & ns-3 Implementation
  - MAC-level performance analysis of Semi-Persistent Scheduling (SPS) resource allocation in a vehicular scenario
  - Validation of ns-3 NR sidelink MAC models
- Proximity Services (ProSe) Support for 5G NR Simulations
  - Discovery and Layer-3 UE-to-Network Relay models

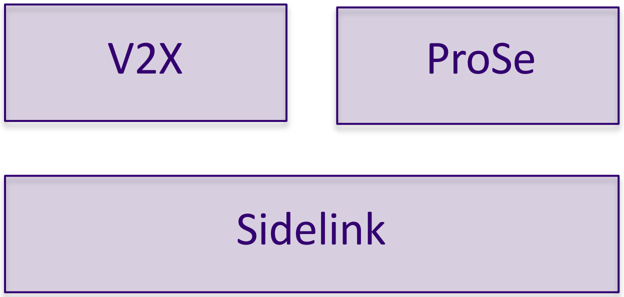
Terminology

> NR: New Radio

> V2X: Vehicle-to-Everything

> ProSe: Proximity Services

> SL: Sidelink



V2X

ProSe

Sidelink

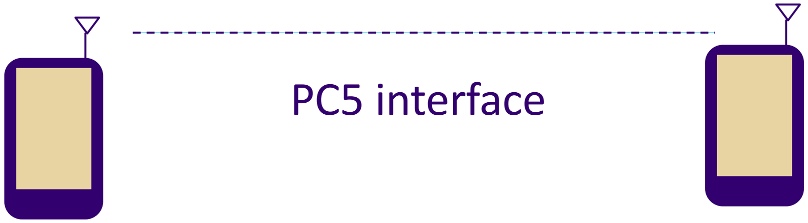
> Are V2X, ProSe, and SL interchangeable terms?

– V2X refers to a use case enabled by an underlying SL air interface

– ProSe is another vertical sitting on top of SL

– SL refers to communications between UE that do not go through the network

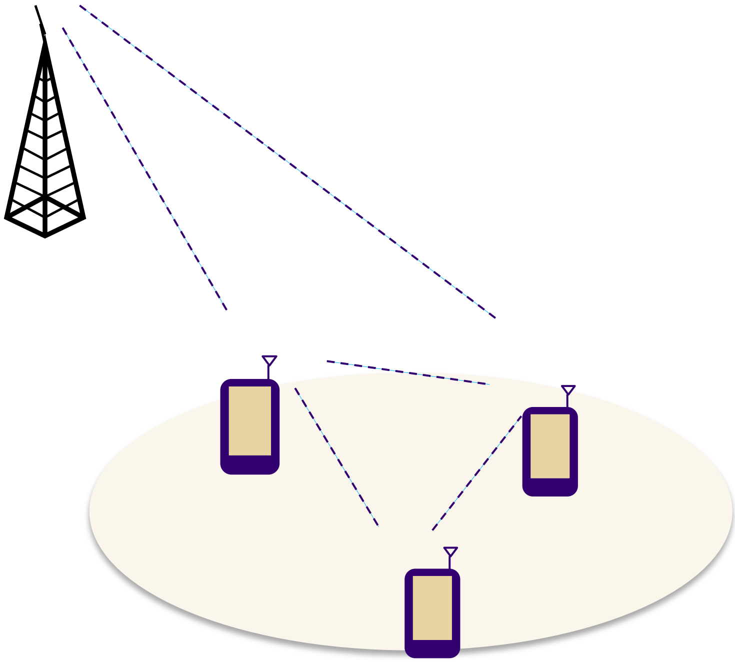
– Not all SL use cases involve vehicles (e.g., ProSe, public safety)



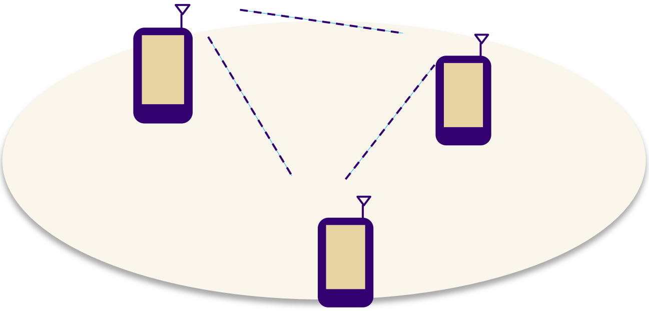
PC5 interface

NR V2X Modes

- **Mode 1**: Resource allocation over the sidelink channel is managed by the network
- Analogous to LTE C-V2X Mode 2



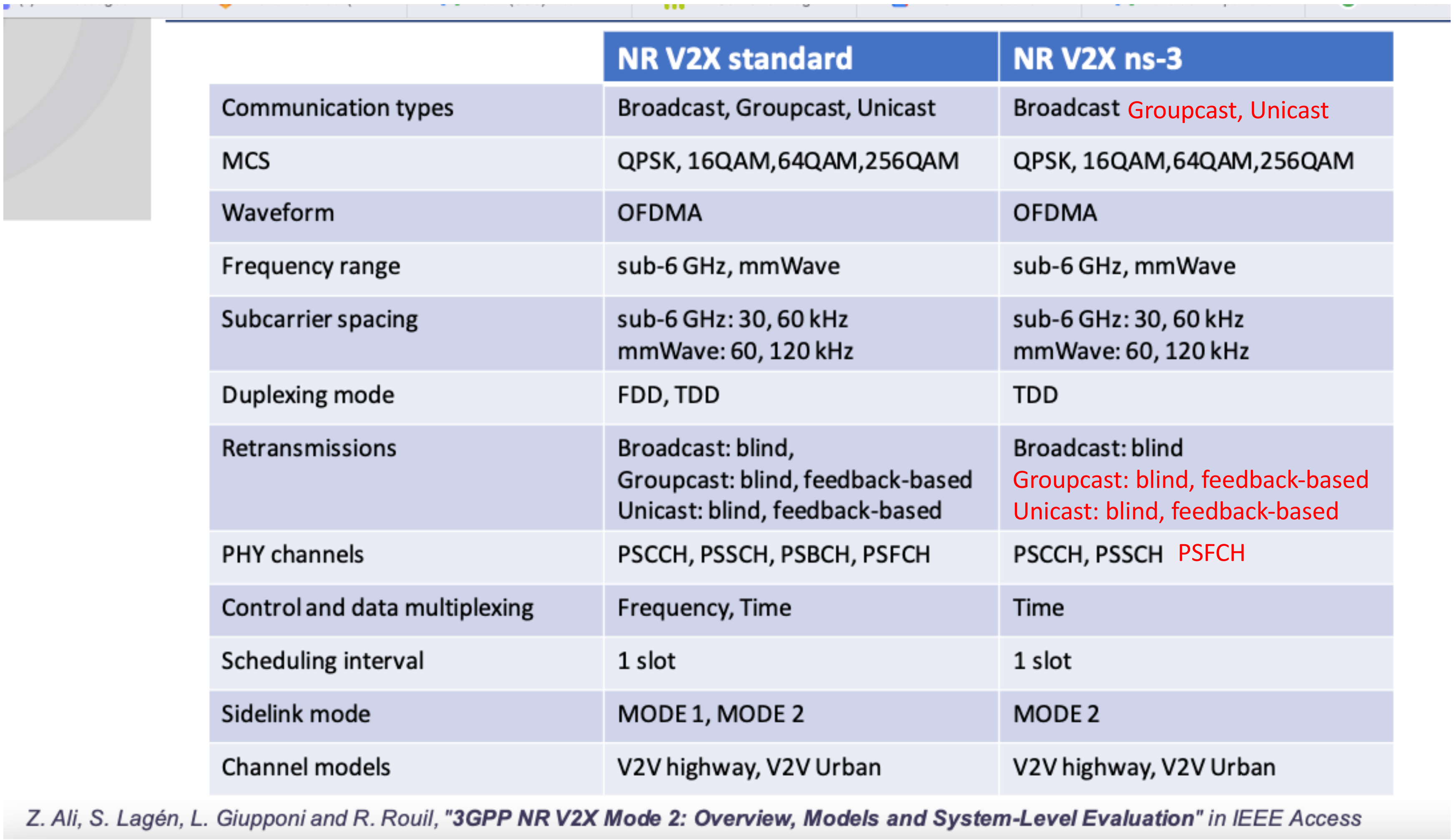
- **Mode 2**: Resource allocation performed without network assistance
- Analogous to LTE C-V2X Mode 4



Acknowledgments

- This tutorial extends last year's NR-V2X tutorial by Zoraze Ali (formerly of CTTC)
- CTTC and Zoraze Ali are the primary authors of ns-3 NR V2X/SL models
- Tom Henderson and Collin Brady (University of Washington), and Samantha Gamboa and Aziza Ben-Mozbah (Prometheus Computing/NIST) have been improving and extending the V2X, ProSe, and SL models
- CTTC's work and the University of Washington's work were funded by the National Institute of Standards and Technology (NIST), led by Richard Rouil and Wesley Garey
- Thanks are due to both CTTC and NIST for open sourcing their NR V2X, sidelink, and ProSe models

What’s new?



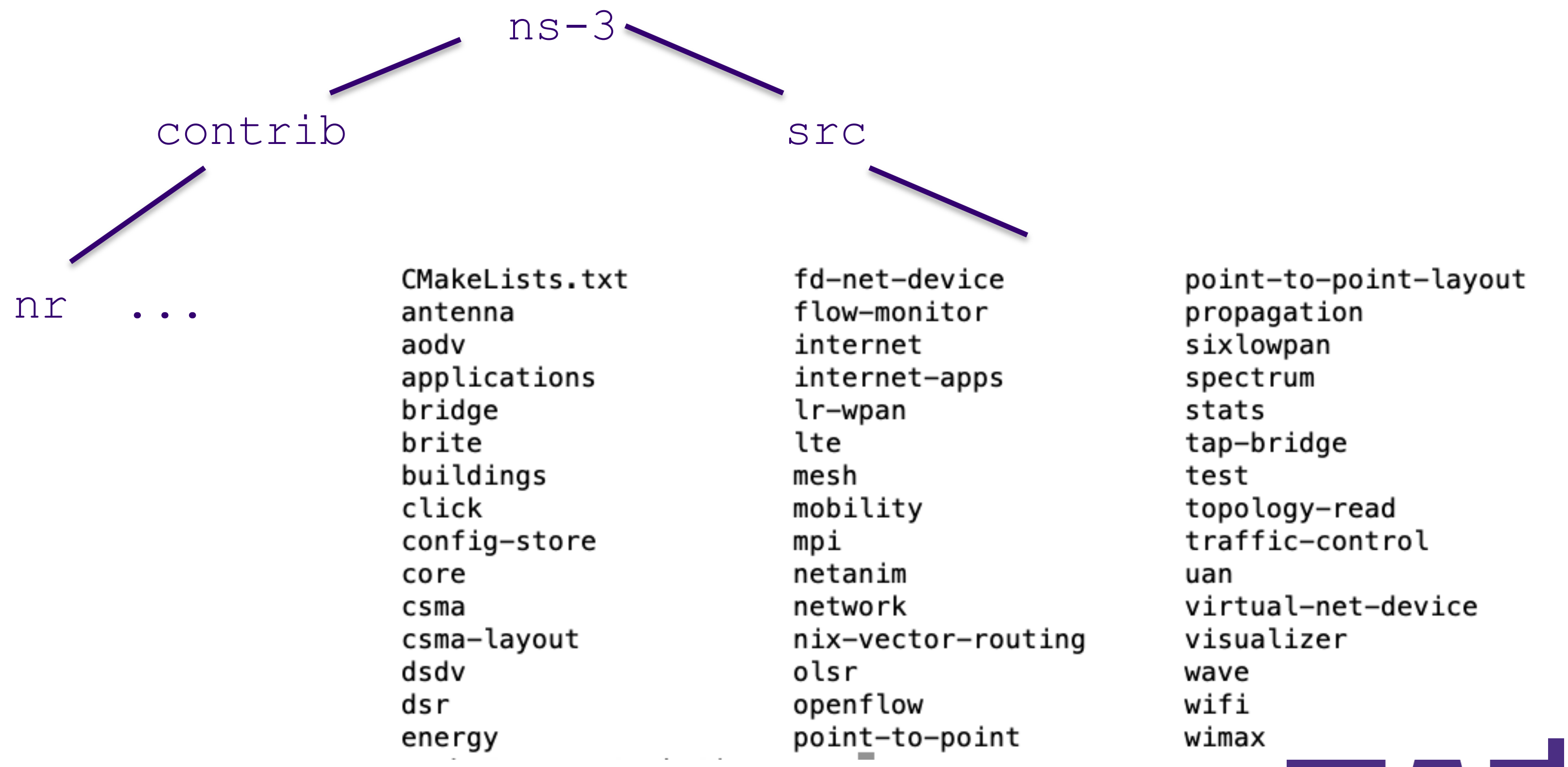
	NR V2X standard	NR V2X ns-3
Communication types	Broadcast, Groupcast, Unicast	Broadcast Groupcast, Unicast
MCS	QPSK, 16QAM,64QAM,256QAM	QPSK, 16QAM,64QAM,256QAM
Waveform	OFDMA	OFDMA
Frequency range	sub-6 GHz, mmWave	sub-6 GHz, mmWave
Subcarrier spacing	sub-6 GHz: 30, 60 kHz mmWave: 60, 120 kHz	sub-6 GHz: 30, 60 kHz mmWave: 60, 120 kHz
Duplexing mode	FDD, TDD	TDD
Retransmissions	Broadcast: blind, Groupcast: blind, feedback-based Unicast: blind, feedback-based	Broadcast: blind Groupcast: blind, feedback-based Unicast: blind, feedback-based
PHY channels	PSCCH, PSSCH, PSBCH, PSFCH	PSCCH, PSSCH PSFCH
Control and data multiplexing	Frequency, Time	Time
Scheduling interval	1 slot	1 slot
Sidelink mode	MODE 1, MODE 2	MODE 2
Channel models	V2V highway, V2V Urban	V2V highway, V2V Urban

*Z. Ali, S. Lagén, L. Giupponi and R. Rouil, "**3GPP NR V2X Mode 2: Overview, Models and System-Level Evaluation**" in IEEE Access*

Other: Sensing: Support for multiple subchannels
Scheduling: Dynamic grants, multiple logical channels

Roadmap to ns-3 NR V2X/SL software

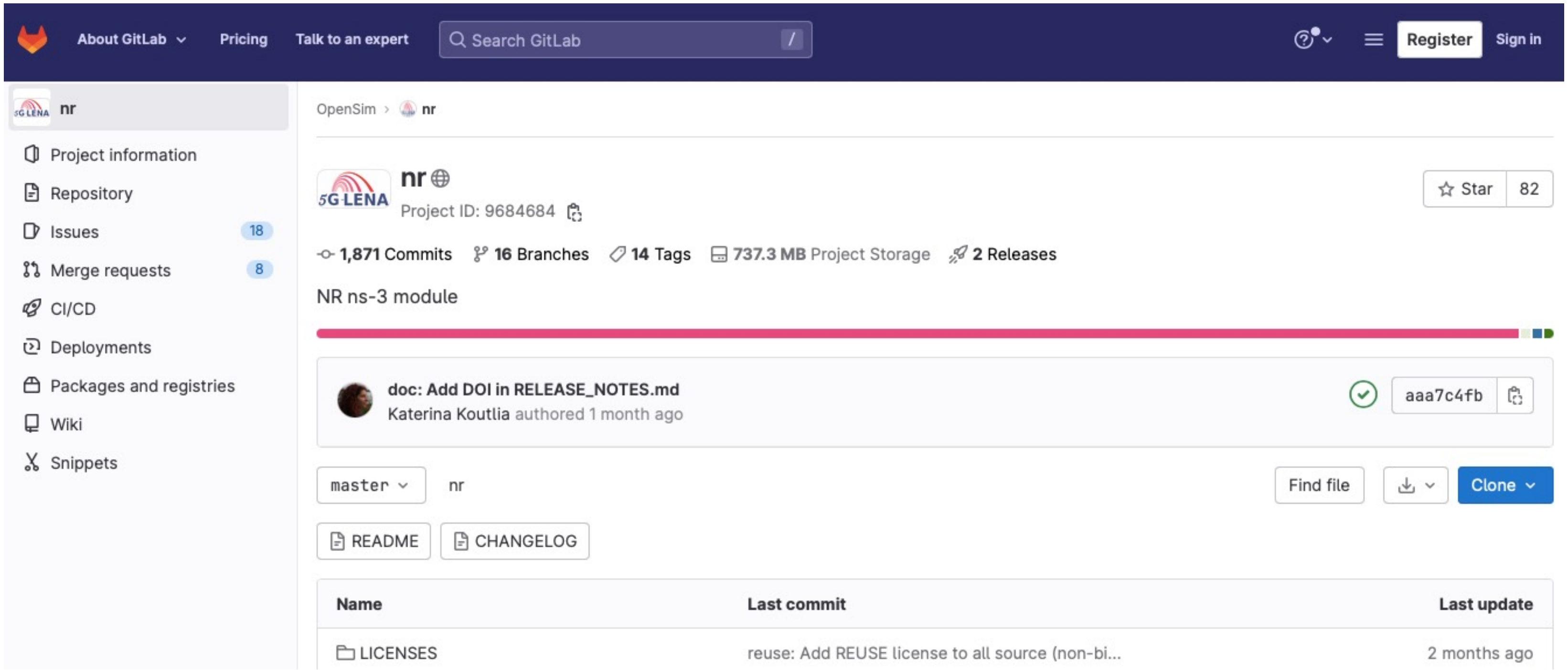
> ns-3 has moved to a modular codebase, with optional models supported through the `contrib` directory



```
CMakeLists.txt      fd-net-device         point-to-point-layout
antenna             flow-monitor          propagation
aodv                internet              sixlowpan
applications        internet-apps         spectrum
bridge              lr-wpan               stats
brite               lte                   tap-bridge
buildings           mesh                  test
click               mobility              topology-read
config-store        mpi                   traffic-control
core                netanim               uan
csma                network               virtual-net-device
csma-layout         nix-vector-routing    visualizer
dsdv                olsr                  wave
dsr                 openflow              wifi
energy              point-to-point        wimax
```

nr module from CTTC’s OpenSim

> The public nr module is available at on GitLab.com:
  – https://gitlab.com/cttc-lena/nr.git



nr

Project ID: 9684684

1,871 Commits · 16 Branches · 14 Tags · 737.3 MB Project Storage · 2 Releases

NR ns-3 module

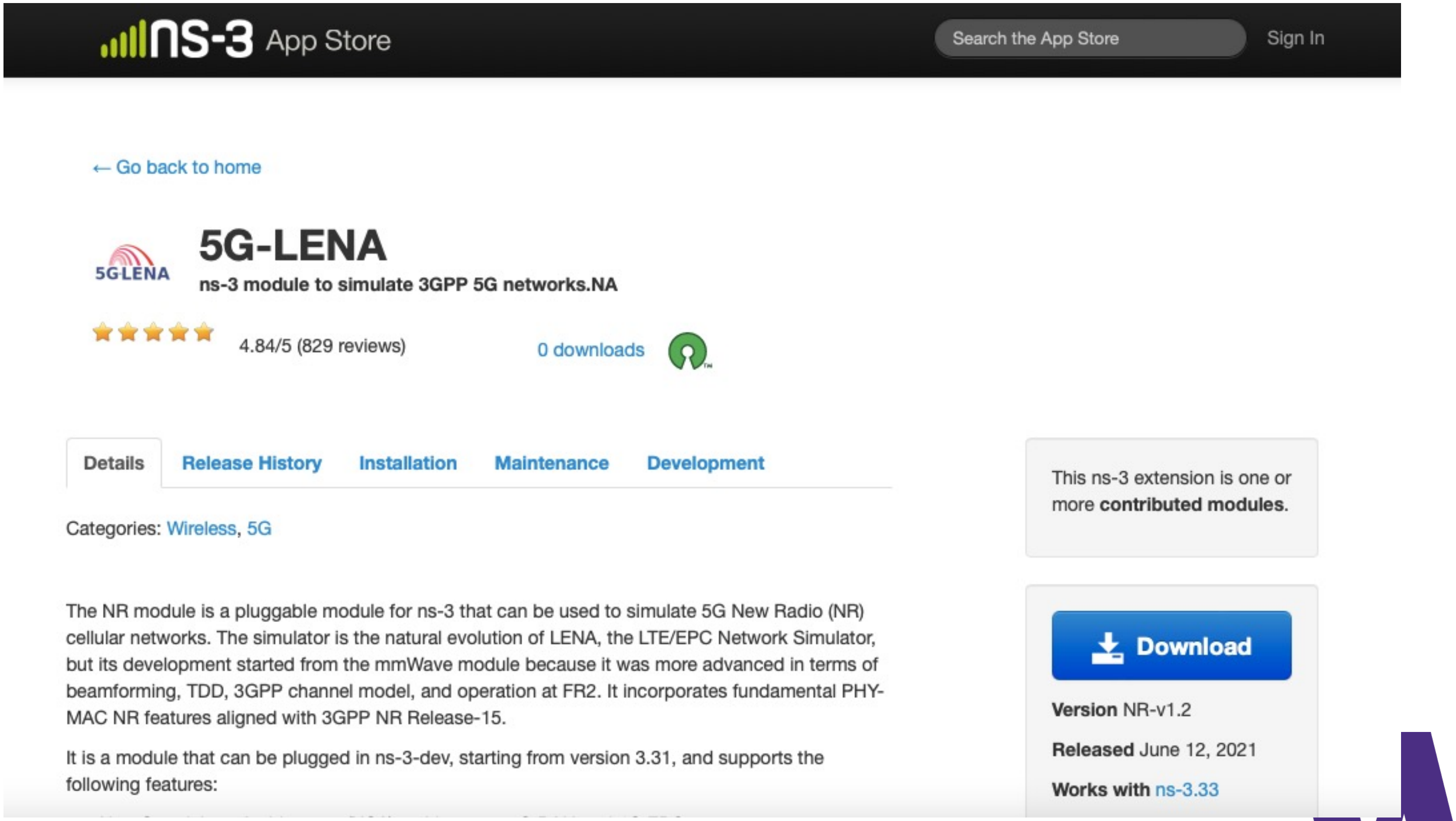
doc: Add DOI in RELEASE_NOTES.md
Katerina Koutlia authored 1 month ago · aaa7c4fb

master · nr

Name	Last commit	Last update
LICENSES	reuse: Add REUSE license to all source (non-bi...	2 months ago

ns-3 App Store

- ns-3 extensions also have a public page on the App Store:
  - https://apps.nsnam.org/app/nr/



5G-LENA

ns-3 module to simulate 3GPP 5G networks.NA

★★★★★ 4.84/5 (829 reviews) 0 downloads

Categories: Wireless, 5G

The NR module is a pluggable module for ns-3 that can be used to simulate 5G New Radio (NR) cellular networks. The simulator is the natural evolution of LENA, the LTE/EPC Network Simulator, but its development started from the mmWave module because it was more advanced in terms of beamforming, TDD, 3GPP channel model, and operation at FR2. It incorporates fundamental PHY-MAC NR features aligned with 3GPP NR Release-15.

It is a module that can be plugged in ns-3-dev, starting from version 3.31, and supports the following features:

This ns-3 extension is one or more **contributed modules**.

Version NR-v1.2

Released June 12, 2021

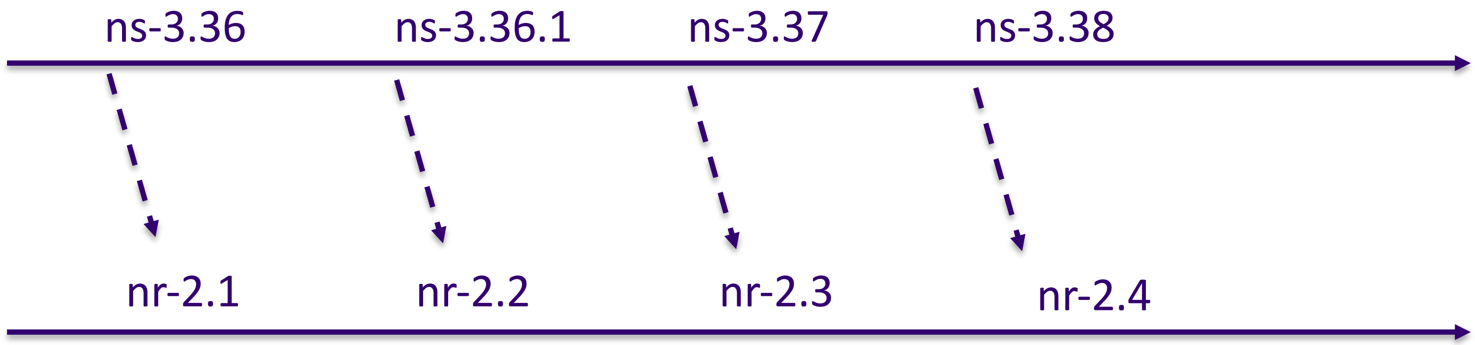
Works with ns-3.33

ns-3 and nr module synchronization

> Contrib modules must be paired with compatible ns-3 release versions

> CTTC makes an 'nr' module release after every ns-3 release

> nr modules are self-contained; require no ns-3 modifications



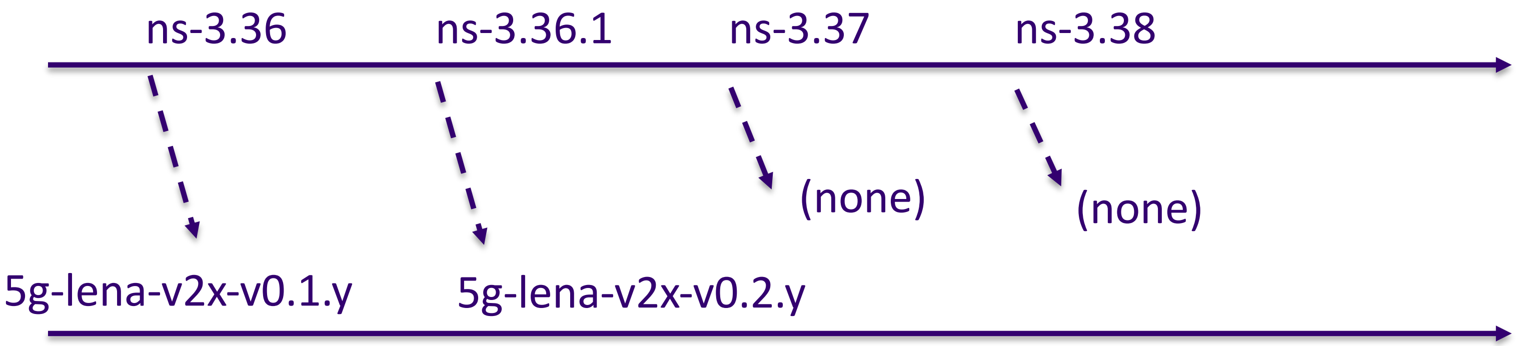
NR V2X differences

> V2X/SL extensions are not maintained in the `nr` master branch but in a separate `nr-v2x-dev` branch

> They also require a special (patched) version of ns-3

> This special version of ns-3 is maintained in another CTTC repository and branch:

  - https://gitlab.com/cttc-lena/ns-3-dev/-/tree/v2x-lte-dev
  - Currently, it lags ns-3-dev-- it is only up to ns-3.36.1 release



Instructions to obtain CTTC’s NR V2X dev version

```
$ git clone https://gitlab.com/cttc-lena/ns-3-dev.git
$ cd ns-3-dev
$ git checkout -b v2x-lte-dev origin/v2x-lte-dev
$ cd contrib
$ git clone https://gitlab.com/cttc-lena/nr.git
$ cd nr
$ git checkout -b nr-v2x-dev origin/nr-v2x-dev
```

Software in use today

> Three tutorial presentations are using customizations of the CTTC NR V2X branches

1. This introductory tutorial uses the following branches
   1. https://gitlab.com/tomhenderson/ns-3-dev.git, branch wns3-2023-tutorial
   2. https://gitlab.com/tomhenderson/nr.git, branch wns3-2023-tutorial
2. The resource allocation tutorial is based on Collin Brady’s extensions (see next tutorial)
3. The ProSe tutorial uses a special branch available at NIST’s GitHub repository (see afternoon tutorial)

Instructions to obtain this tutorial’s code

```
$ git clone https://gitlab.com/tomhenderson/ns-3-dev.git
$ cd ns-3-dev
$ git checkout -b wns3-2023-tutorial origin/wns3-2023-tutorial
$ cd contrib
$ git clone https://gitlab.com/tomhenderson/nr.git
$ cd nr
$ git checkout -b wns3-2023-tutorial origin/wns3-2023-tutorial
```

Differences with respect to upstream CTTC instructions are depicted in red

- git commit hash of ns-3-dev branch: fe082b36 (June 15, 2023)
- git commit hash of nr branch: 69d1eed4 (June 25, 2023)

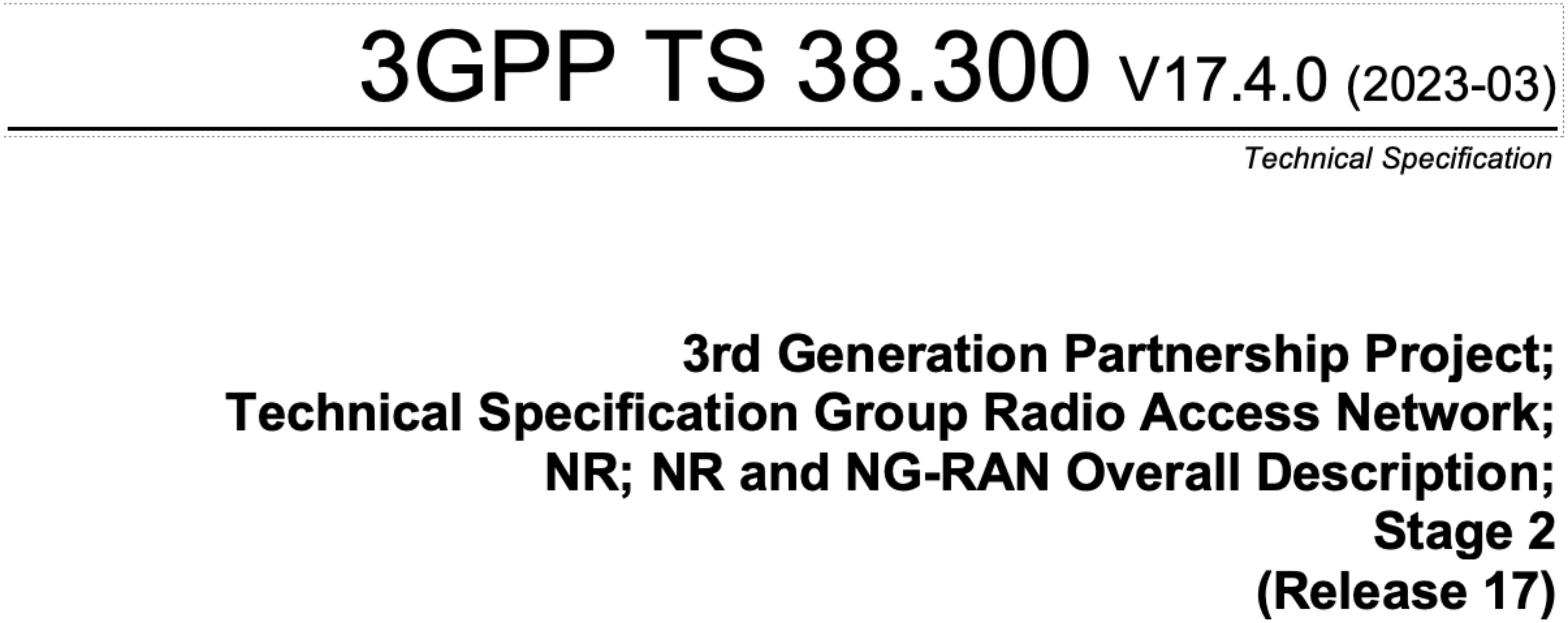
This code will be upstreamed to CTTC’s repositories once documentation and testing are completed

Remainder of this tutorial

- We will work directly with recently created example programs (nr-v2x-simple-multi-lc, sidelink-harq-example) to highlight the MAC operation and recent changes to the model
- Topics covered:
  - SL resources and terminology: symbols, slots, PRBs, subchannels, resource pools, sidelink bitmaps
  - How traffic from different applications is routed to different logical channels
  - How the scheduler prioritizes between different logical channels
  - How the scheduler is triggered to select resources, for either semi-persistent or dynamic granting
  - How the sensing process (TS 38.214 Section 8.1.4) is implemented and consulted
  - How the HARQ and PSFCH feedback channel operate

Terminology, Architecture, References

- 3GPP TS 38.300 is a good overview reference for 5G NR
  - Sections 5.7 and 16.9 pertain to sidelink



3GPP TS 38.300 V17.4.0 (2023-03)

Technical Specification

3rd Generation Partnership Project;
Technical Specification Group Radio Access Network;
NR; NR and NG-RAN Overall Description;
Stage 2
(Release 17)

NG-RAN architecture

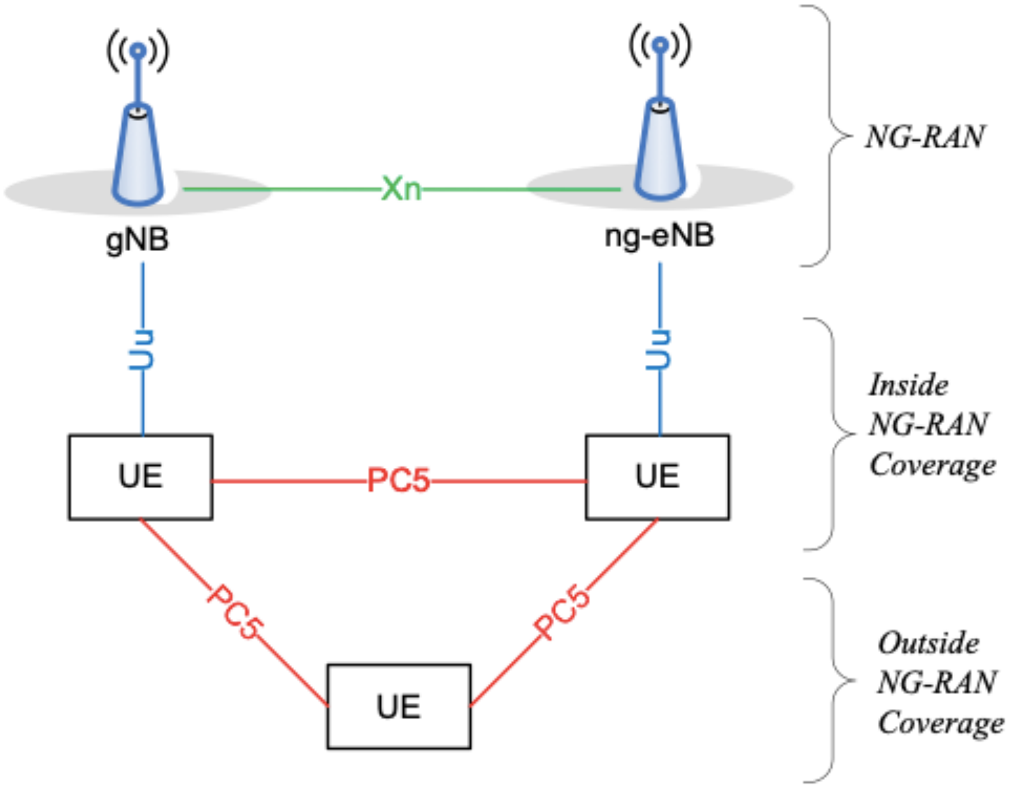


Figure 16.9.1-1: NG-RAN Architecture supporting the PC5 interface

Figure source: 3GPP TS 38.300

Protocol stack architecture

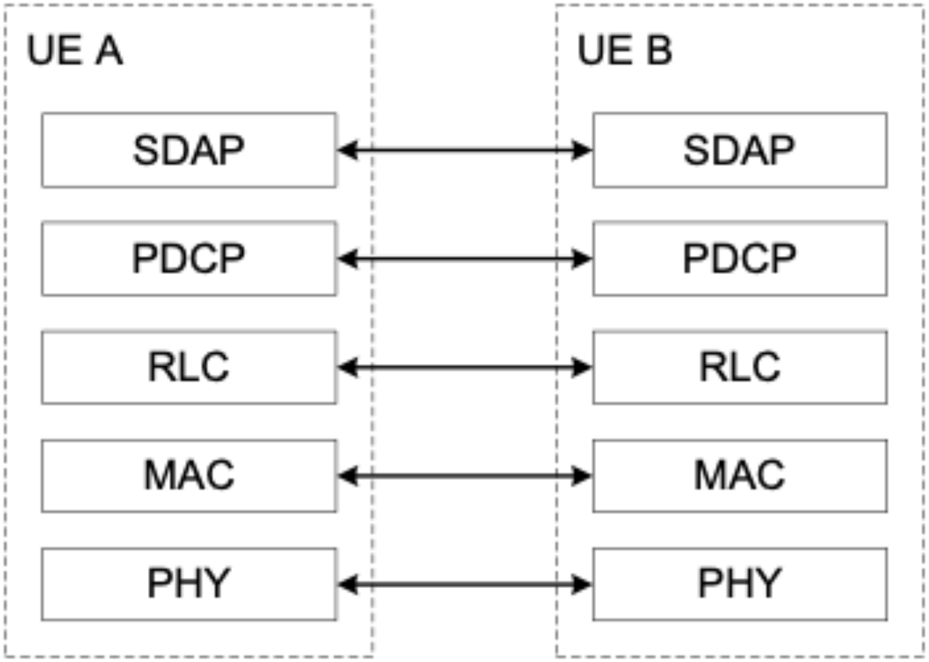


Figure 16.9.2.1-4: User plane protocol stack for STCH.

Figure source: 3GPP TS 38.300

Layer 2 architecture

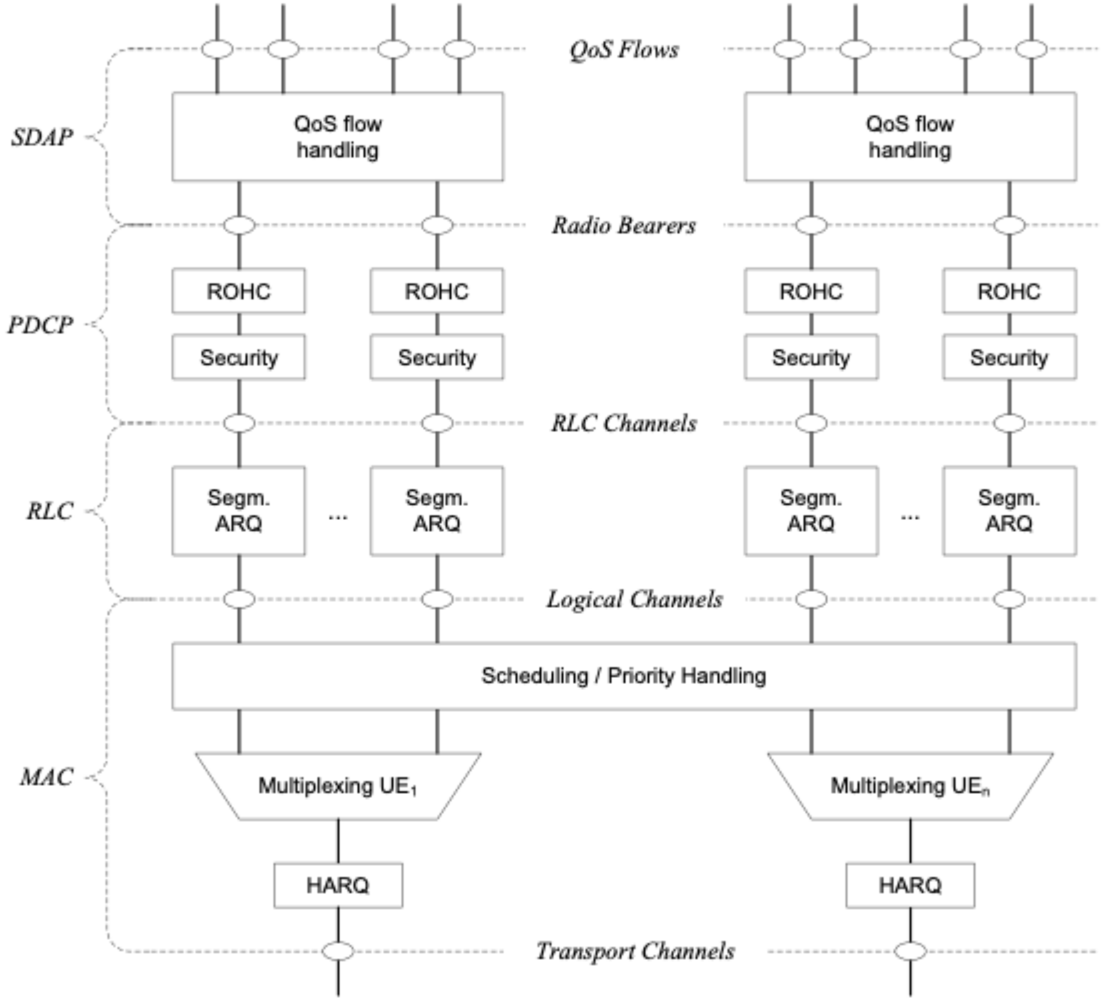


Figure 6.1-1: Downlink Layer 2 Structure

Figure source: 3GPP TS 38.300

For more information

> Last year's NR-V2X tutorial by Zoraze Ali (and NR overview by Biljana Bojovic) provides more background overview on this ns-3 model:
  – https://www.nsnam.org/research/wns3/wns3-2022/tutorials/

> IEEE Access article on the ns-3 NR V2X extensions, by Ali, Lagen, Giupponi, and Rouil
  – https://ieeexplore.ieee.org/document/9461188

> Thorough tutorial article on NR V2X in general, in IEEE Communications Surveys and Tutorials by Garcia et al:
  – https://ieeexplore.ieee.org/document/9345798

Current software organization

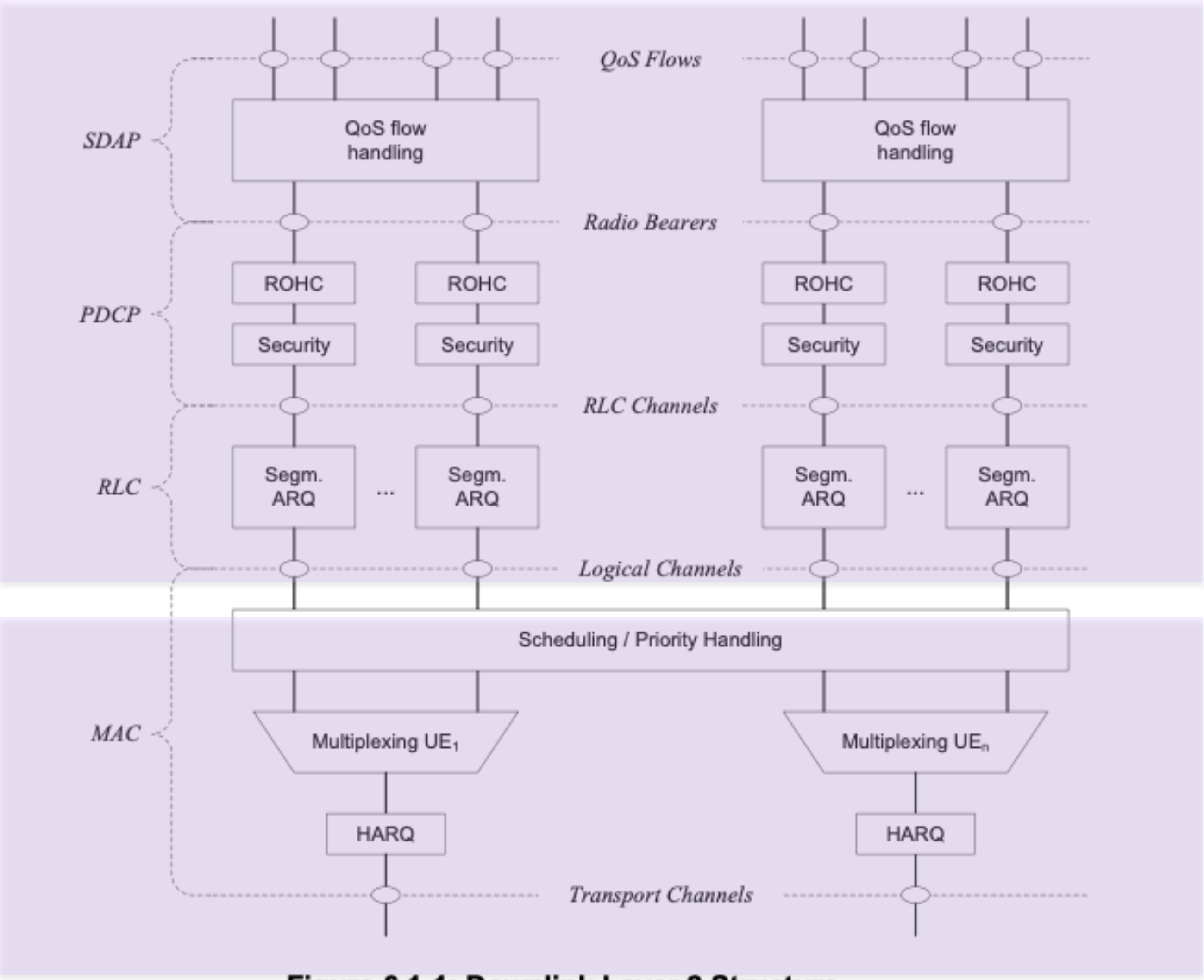


Figure 6.1-1: Downlink Layer 2 Structure

These layers reside in the src/lte directory

Lower layers reside in the contrib/nr directory

Figure source: 3GPP TS 38.300

Tutorial focus

> Today's focus: *data plane lifecycle of a packet (MAC layer)*

IP packets are classified into different logical channels

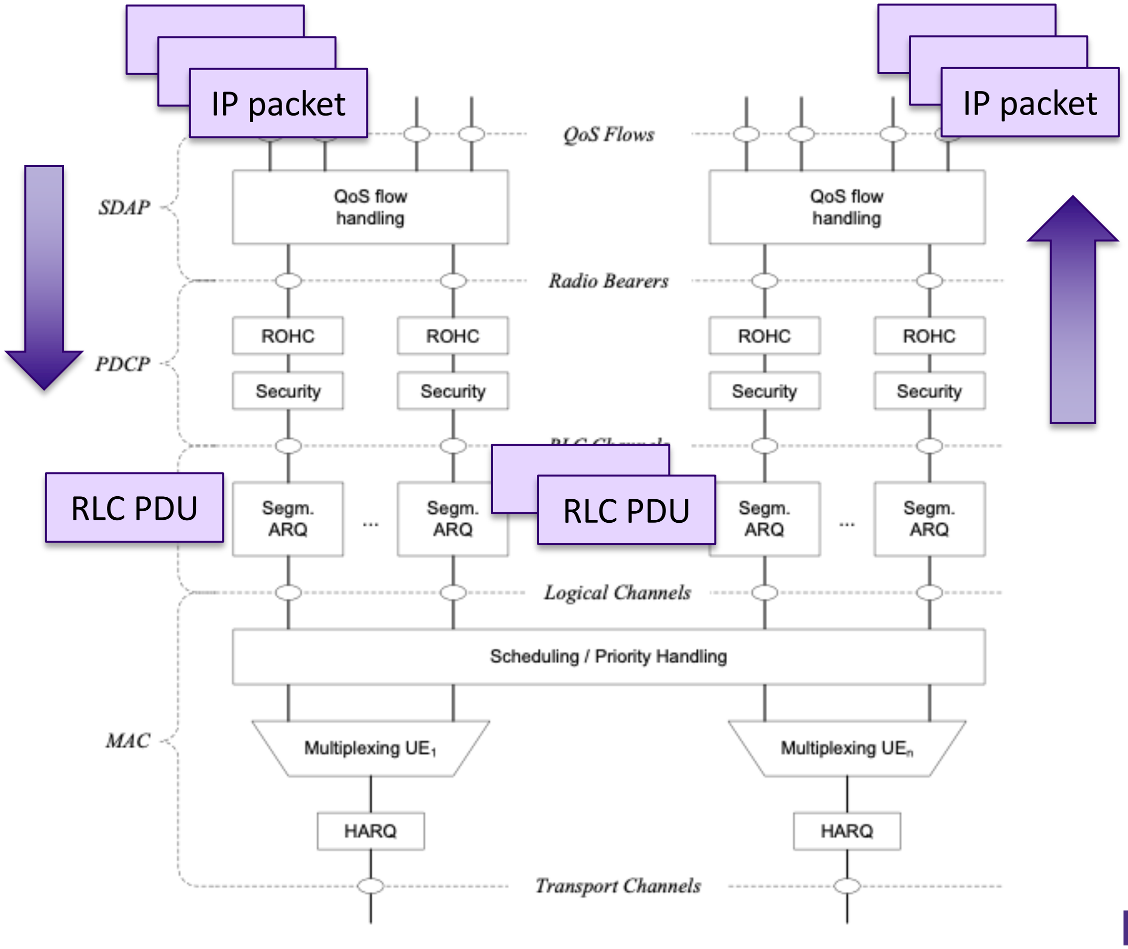


Figure 6.1-1: Downlink Layer 2 Structure

Figure source: 3GPP TS 38.300

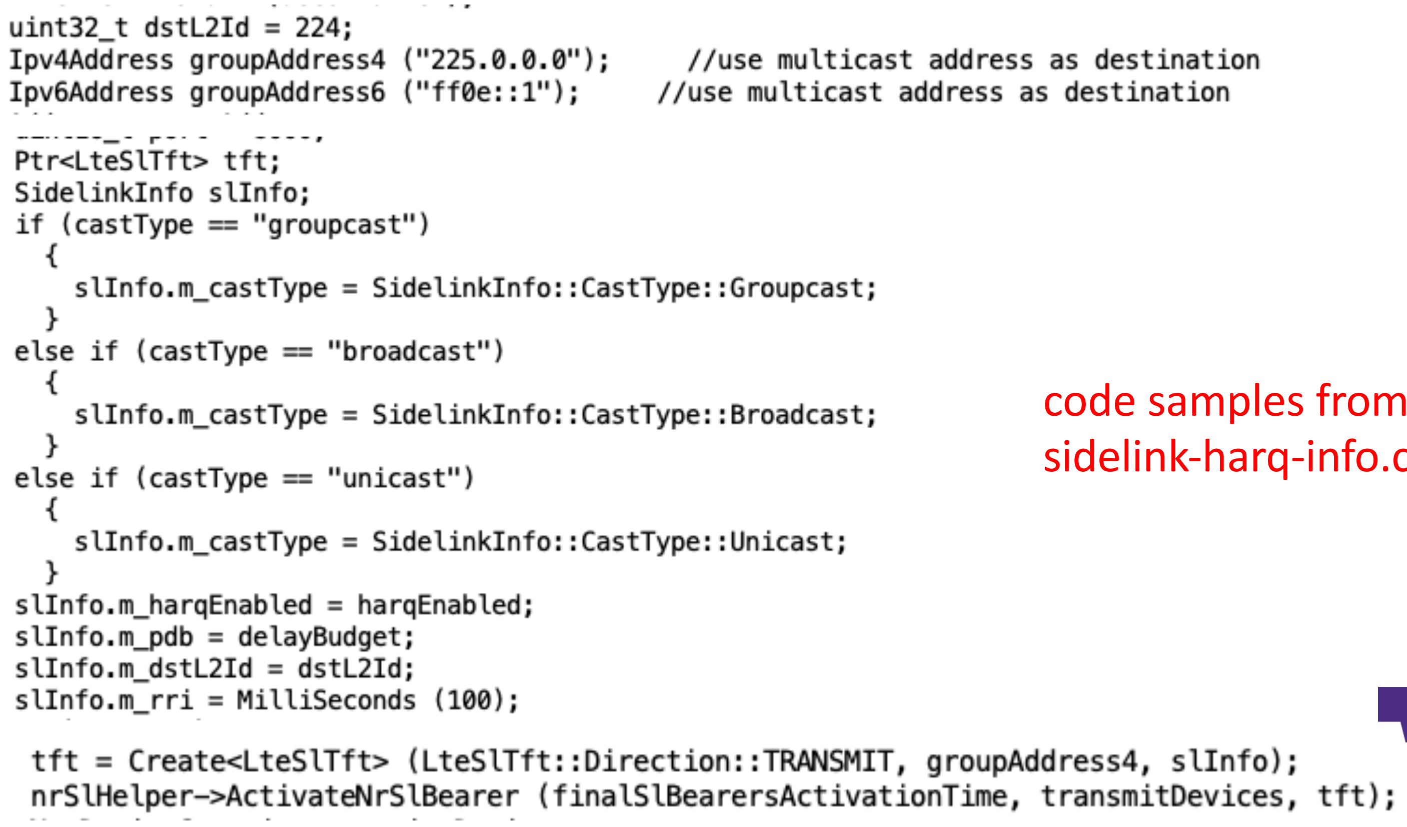
Logical channels

> Groupings of packets that receive similar service by the MAC layer
  - Destination L2 ID
  - “Cast type” (Unicast, Broadcast, Groupcast)
  - “Hybrid ARQ” (HARQ) type
  - Packet delay budget
  - Scheduling (or resource) type (semi-persistent, or dynamic)
  - Resource Reservation Interval (RRI)
  - Priority
  - Packet Error Rate requirements

> These are parameters in a SidelinkInfo structure (see next slide)

Mapping of IP packets to logical channels

> The SidelinkInfo [*] structure is passed to the LTE Sidelink Traffic Flow Template (LteSlTft)
  - [*] corresponds to "Sidelink Transmission/Identification/Other Information” defined in TS 38.321.



```
uint32_t dstL2Id = 224;
Ipv4Address groupAddress4 ("225.0.0.0");     //use multicast address as destination
Ipv6Address groupAddress6 ("ff0e::1");     //use multicast address as destination
```

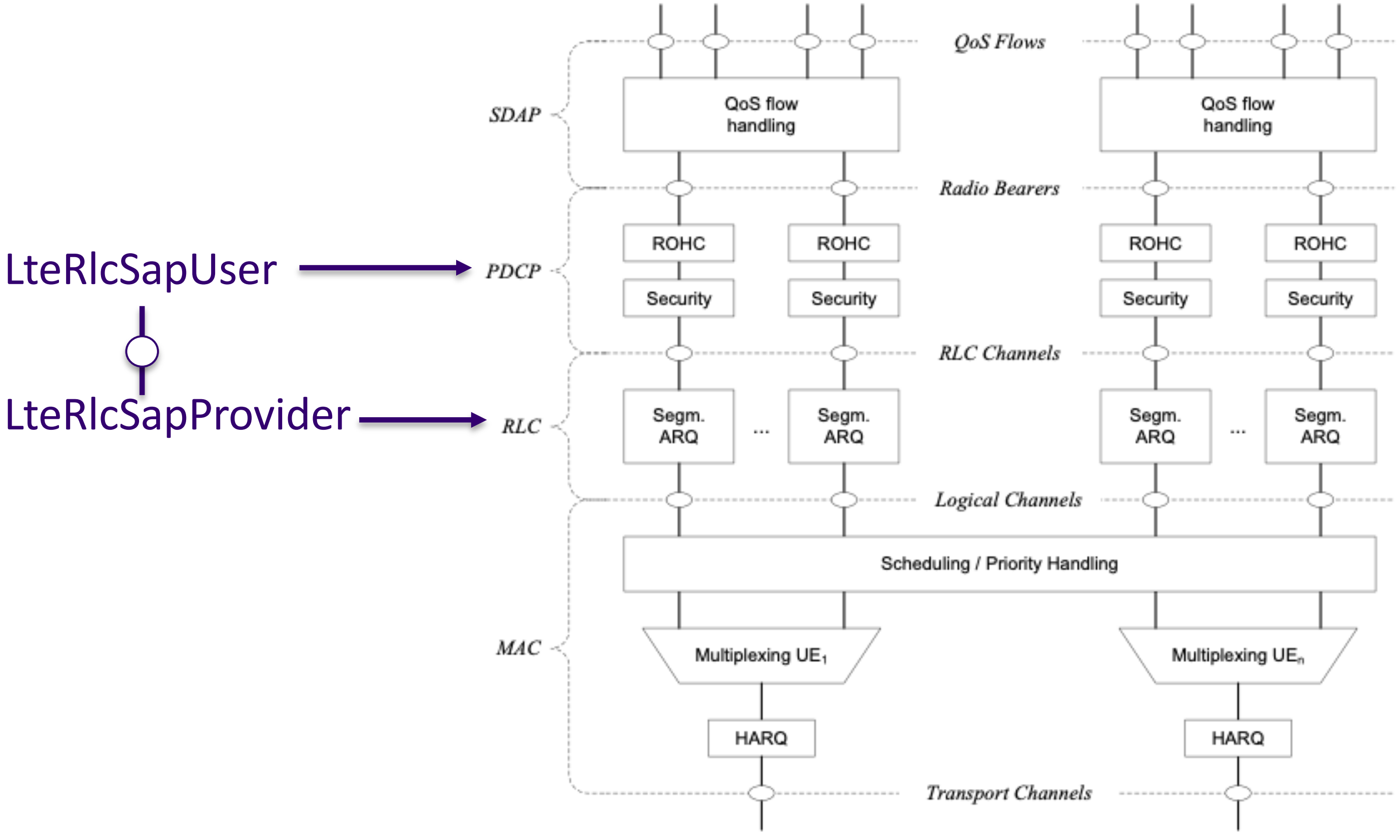
```
Ptr<LteSlTft> tft;
SidelinkInfo slInfo;
if (castType == "groupcast")
  {
    slInfo.m_castType = SidelinkInfo::CastType::Groupcast;
  }
else if (castType == "broadcast")
  {
    slInfo.m_castType = SidelinkInfo::CastType::Broadcast;
  }
else if (castType == "unicast")
  {
    slInfo.m_castType = SidelinkInfo::CastType::Unicast;
  }
slInfo.m_harqEnabled = harqEnabled;
slInfo.m_pdb = delayBudget;
slInfo.m_dstL2Id = dstL2Id;
slInfo.m_rri = MilliSeconds (100);
```

```
tft = Create<LteSlTft> (LteSlTft::Direction::TRANSMIT, groupAddress4, slInfo);
nrSlHelper->ActivateNrSlBearer (finalSlBearersActivationTime, transmitDevices, tft);
```

code samples from
sidelink-harq-info.cc

Service Access Points

- In ns-3, several interfaces are expressed formally in terms of a Service Access Point (SAP)



LteRlcSapUser

LteRlcSapProvider

Figure 6.1-1: Downlink Layer 2 Structure

Figure source: 3GPP TS 38.300

Sidelink RLC/MAC API

> In ns-3, several interfaces are expressed formally in terms of a Service Access Point (SAP)

NrSlMacSapUser →

NrSlMacSapProvider →

NrSlUePhySapUser →

NrSlUePhySapProvider →

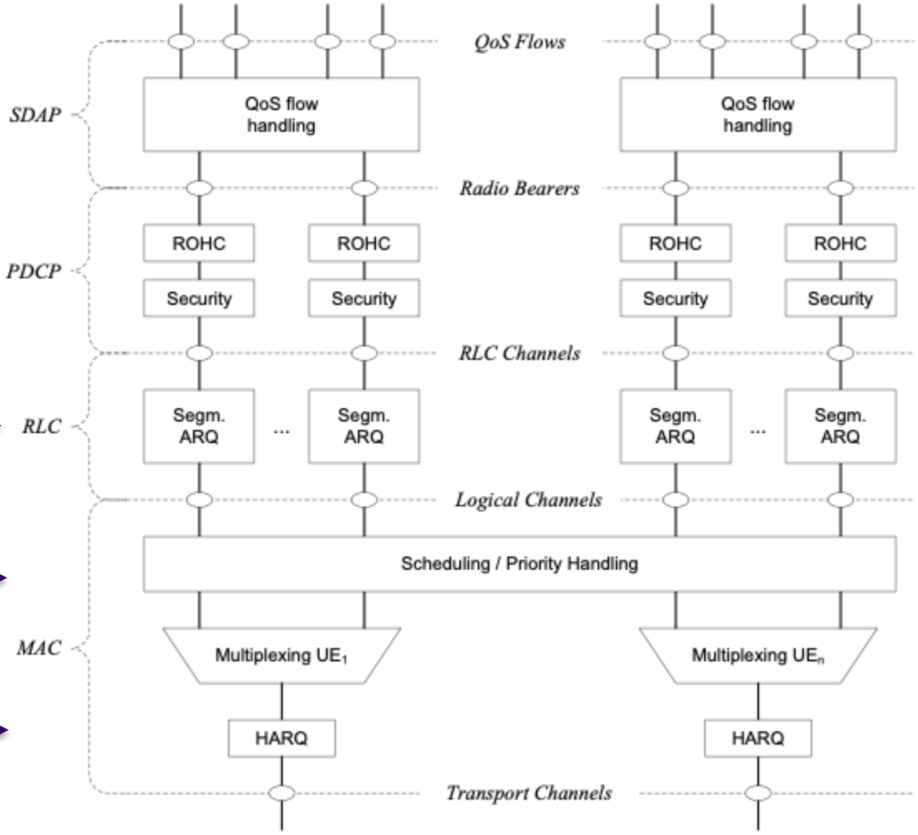


Figure 6.1-1: Downlink Layer 2 Structure

Figure source: 3GPP TS 38.300

C++ implementation of NR SAP APIs

> Model objects (e.g. NrUeMac) include API objects (e.g. NrSlMacSapProvider) that act as forwarding objects

> Model objects provide and make use of callbacks to access these APIs; these are connected together by the helper classes

```
class NrUeMac : public Object
{
  ...

public:

  ...

  /**
   * \brief Get the PHY SAP User (AKA the MAC representation for the PHY)
   * \return the PHY SAP User (AKA the MAC representation for the PHY)
   */
  NrUePhySapUser* GetPhySapUser ();

  /**
   * \brief Set PHY SAP provider (AKA the PHY representation for the MAC)
   * \param ptr the PHY SAP provider (AKA the PHY representation for the MAC)
   */
  void SetPhySapProvider (NrPhySapProvider* ptr);
```

C++ implementation of NR SAP APIs (cont.)

- The access to the model object’s public API is restricted to those forwarding methods that are implemented in the SAP class
- The nr-phy-sap.h file defines these interfaces

```
class NrUePhySapUser
{
public:

  ...

  virtual void ReceivePhyPdu (Ptr<Packet> p) = 0;

  virtual void ReceiveControlMessage (Ptr<NrControlMessage> msg) = 0;

  virtual void SlotIndication (SfnSf s) = 0;

  virtual uint8_t GetNumHarqProcess () const = 0;
};
```

C++ implementation of NR SAP APIs (cont.)

> The definition of these interfaces is in the implementation files

> Usually, they are just forwarding to public API methods

```
class MacUeMemberPhySapUser : public NrUePhySapUser
{
public:
  MacUeMemberPhySapUser (NrUeMac* mac);

  virtual void ReceivePhyPdu (Ptr<Packet> p) override;

  virtual void ReceiveControlMessage (Ptr<NrControlMessage> msg) override;

  virtual void SlotIndication (SfnSf sfn) override;

  //virtual void NotifyHarqDeliveryFailure (uint8_t harqId);

  virtual uint8_t GetNumHarqProcess () const override;

private:
  NrUeMac* m_mac;
};

}
void
MacUeMemberPhySapUser::ReceivePhyPdu (Ptr<Packet> p)
{
  m_mac->DoReceivePhyPdu (p);
}
```

Sidelink resources

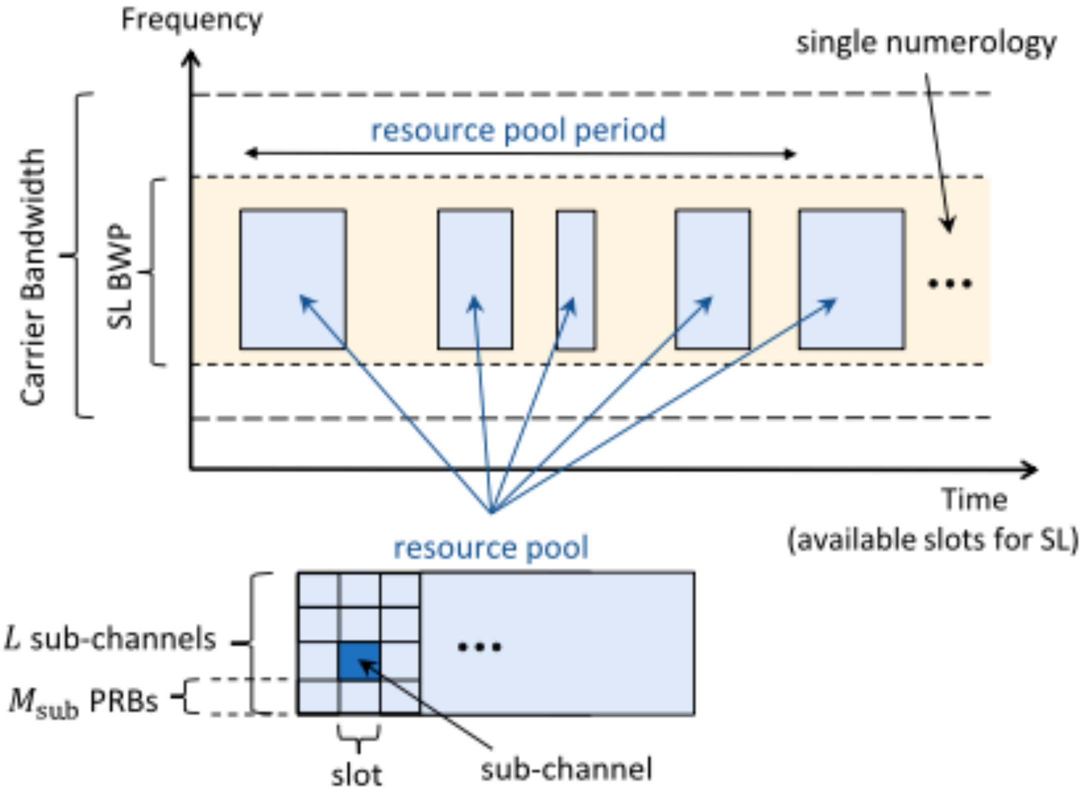


Fig. 4. SL bandwidth part and resource pool for NR V2X sidelink.

Figure source: A Tutorial on 5G NR V2X Communications, Garcia et al.

Slot structure

For the purpose of Transport Block (TB) size determination, the ns-3 code currently assumes that there are nine PSSCH symbols available for data (regardless of PSFCH)

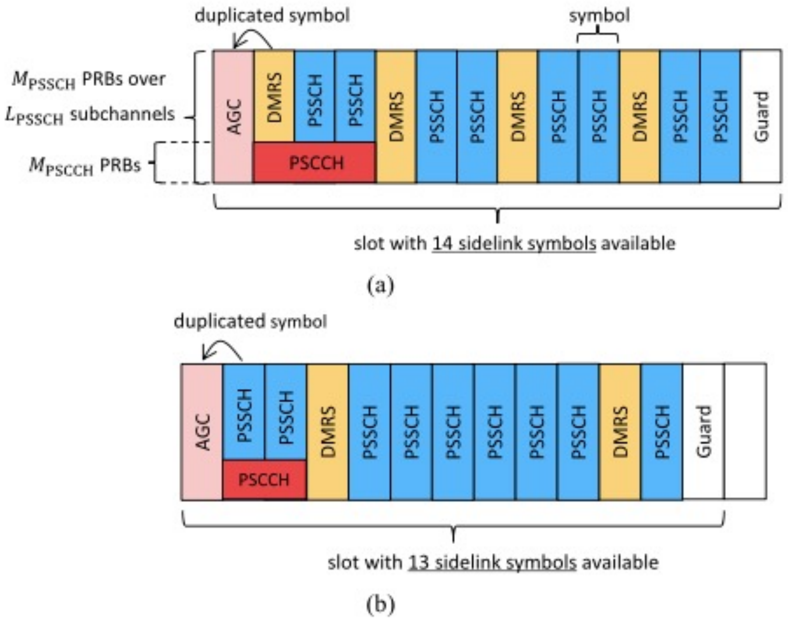


Fig. 5. Example of 2 configurations for a PSCCH/PSSCH slot (for normal CP). (a) Slot with 3 PSCCH, 12 PSSCH including 4 PSSCH DMRS symbols. (b) Slot with 2 PSCCH, 11 PSSCH including 2 PSSCH DMRS symbols.

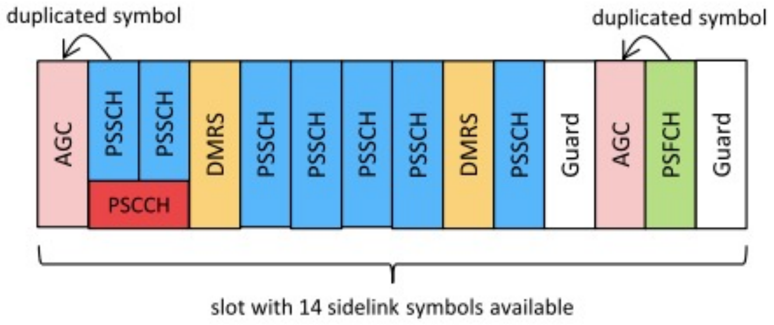


Fig. 6. Example of a PSCCH/PSSCH slot with a PSFCH, 2 PSCCH, 9 PSSCH including 2 PSSCH DMRS symbols (for normal CP).

Figure sources: A Tutorial on 5G NR V2X Communications, Garcia et al.

Sidelink slot pattern

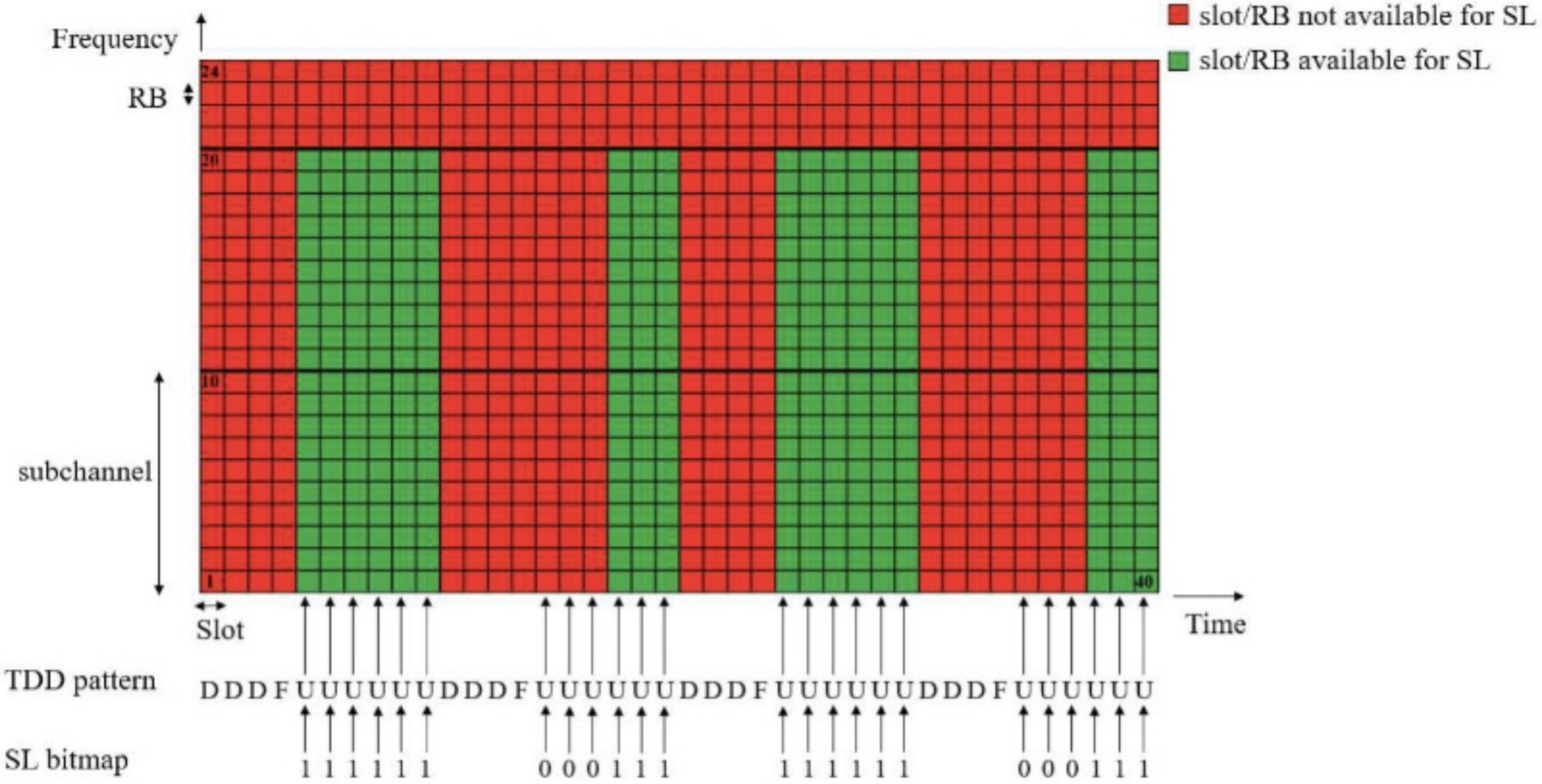
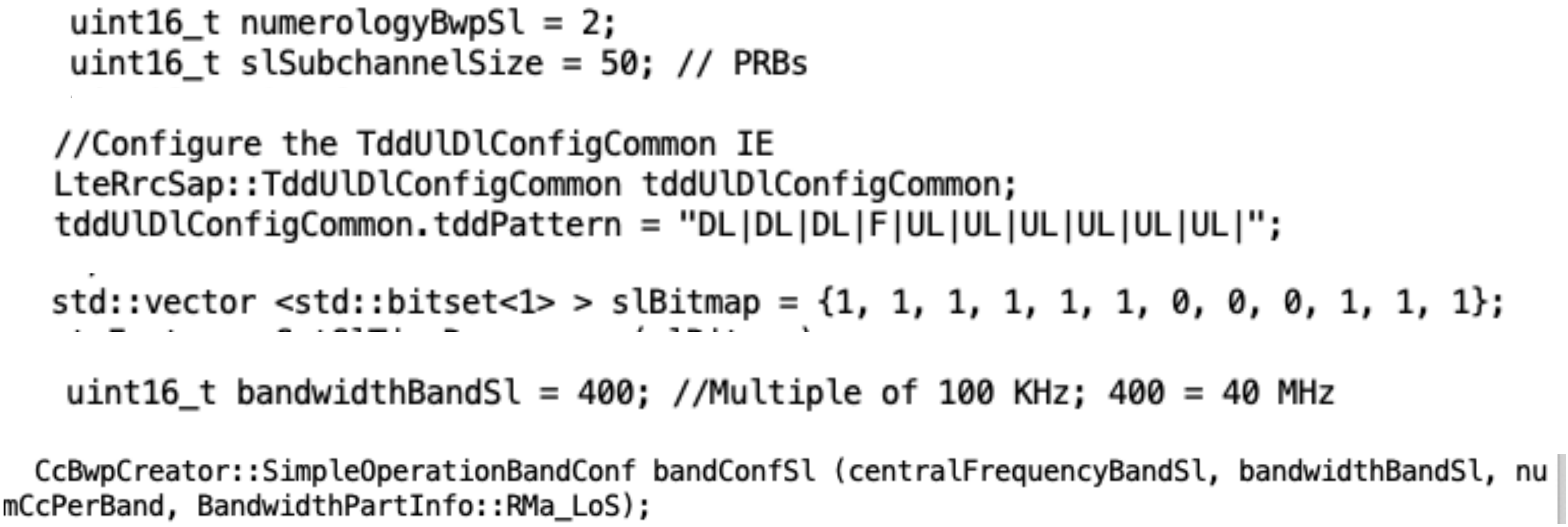

🟥 slot/RB not available for SL
🟩 slot/RB available for SL
Frequency
RB
24
20
10
1
40
subchannel
Slot
Time
TDD pattern
D D D F U U U U U D D D F U U U U U D D D F U U U U U D D D F U U U U U
SL bitmap
1 1 1 1 1 1
0 0 0 1 1 1
1 1 1 1 1 1
0 0 0 1 1 1

FIGURE 1. Time/frequency frame structure and definition of sidelink resource pool for NR V2X TDD. Example with 2 subchannels of 10 RBs each, using TDD pattern of [D D D F U U U U U] and sidelink bitmap of [1 1 1 1 1 1 0 0 0 1 1 1].

Figure source: 3GPP NR V2X Mode 2: Overview, Models and System-Level Evaluation, Ali, Lagen, Giupponi, Rouil

Sidelink resource definition in ns-3 code



```
uint16_t numerologyBwpSl = 2;
uint16_t slSubchannelSize = 50; // PRBs
```

```
//Configure the TddUlDlConfigCommon IE
LteRrcSap::TddUlDlConfigCommon tddUlDlConfigCommon;
tddUlDlConfigCommon.tddPattern = "DL|DL|DL|F|UL|UL|UL|UL|UL|UL|";
```

```
std::vector <std::bitset<1> > slBitmap = {1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1};
```

```
uint16_t bandwidthBandSl = 400; //Multiple of 100 KHz; 400 = 40 MHz
```

```
  CcBwpCreator::SimpleOperationBandConf bandConfSl (centralFrequencyBandSl, bandwidthBandSl, nu
mCcPerBand, BandwidthPartInfo::RMa_LoS);
```

The combination of these parameters largely defines the resources that the MAC has to work with

Guidance as to what are reasonable values to use
In ns-3 simulations would be helpful

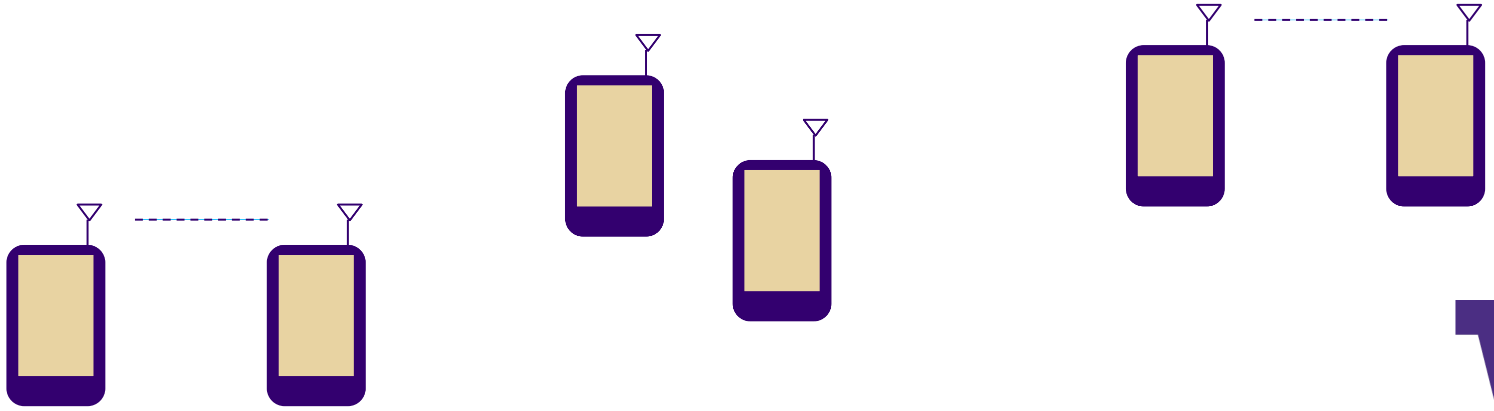
Multiple BWP and multiple RP

- Although not covered in these examples, it is possible to define multiple bandwidth parts (BWP) and multiple resource pools (RP)
  - Different numerologies can be assigned to different BWP
- In ns-3, mappings are therefore needed between logical channels and BWPs, and multiple instances of NrUeMac
- This is managed by the BwpManagerAlgorithm class
  - Currently, only a static mapping based on bearer QCI value is implemented
  - It is unclear if this has been tested or exercised much in the current V2X codebase

Resource allocation and scheduling problem

Consider when a new RLC PDU arrives

- Scheduler must find future resources that
  1. do not conflict with its own scheduled transmissions
  2. do not conflict with planned receptions
  3. are unlikely to collide with other UE transmissions
  4. allow for retransmissions and feedback (if configured)
- Each UE’s interference environment can be different
- Modulation and power control could be factors



Resource allocation and scheduling approach

- Scheduler takes hints from past receptions in a ***sensing window*** to avoid possible future collisions
- Scheduler selects from available resources in the ***selection window*** (and beyond, for semi-persistent scheduling)

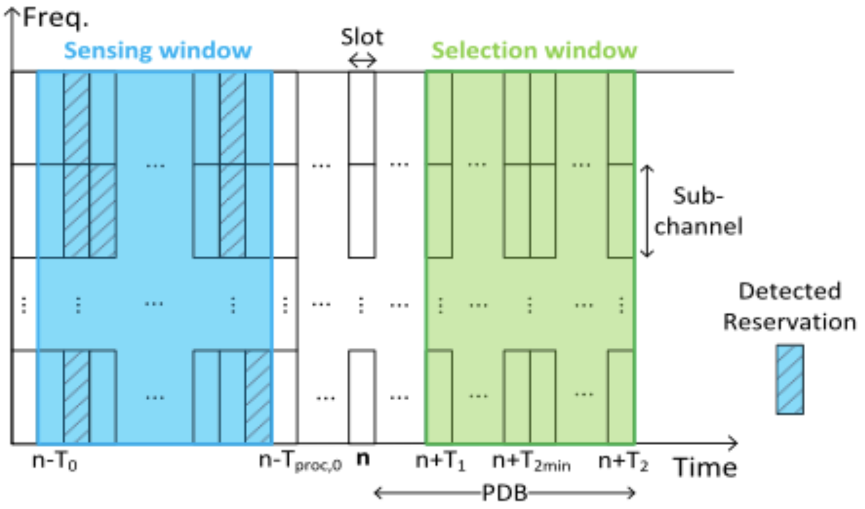


Fig. 18. Sensing and selection windows of NR V2X SL mode 2 when T_2 = PDB.

Figure source: A Tutorial on 5G NR V2X Communications, Garcia et al.

Demonstration of ns-3 sidelink operation

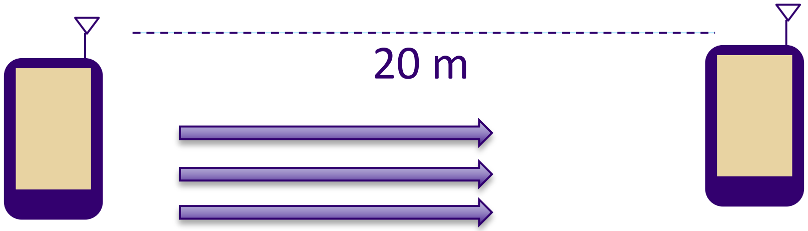
- We will use two example programs to demonstrate how packets are handled by the NR UE MAC layer
  - `contrib/nr/examples/nr-v2x-examples/nr-sl-simple-multi-lc.cc` (authored by Samantha Gamboa)
  - `contrib/nr/examples/nr-v2x-examples/sidelink-harq-example.cc` (authored by Tom Henderson)
- Both programs derive from an original program from Zoraze Ali
  - `contrib/nr/examples/nr-v2x-examples/cttc-nr-v2x-demo-simple.cc`

```
$ simulation commands will be depicted in Courier font, enclosed like this
```

nr-sl-simple-multi-lc.cc overview

> Purpose: Demonstrate how different sidelink traffic profiles can be configured, and how those affect resource selection

> Topology: Two UEs separated by 20 meters



- Three traffic flows, configured differently
- Constant bit rate traffic (default 16 Kb/s, 200 byte UDP packets)
- No Hybrid ARQ (HARQ) configured, but blind retransmissions enabled

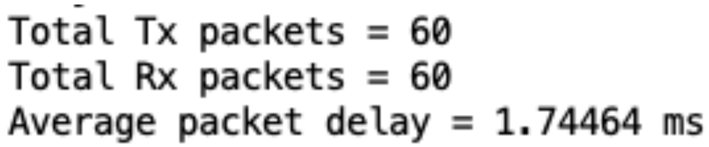
> Traffic profile parameters varied:

- Scheduling type (semi-persistent scheduling (SPS), or dynamic)
- Destination L2 ID
- Priority (and relative priority to SPS/dynamic)
- Resource reservation interval (RRI)

nr-sl-simple-multi-lc.cc overview (cont)

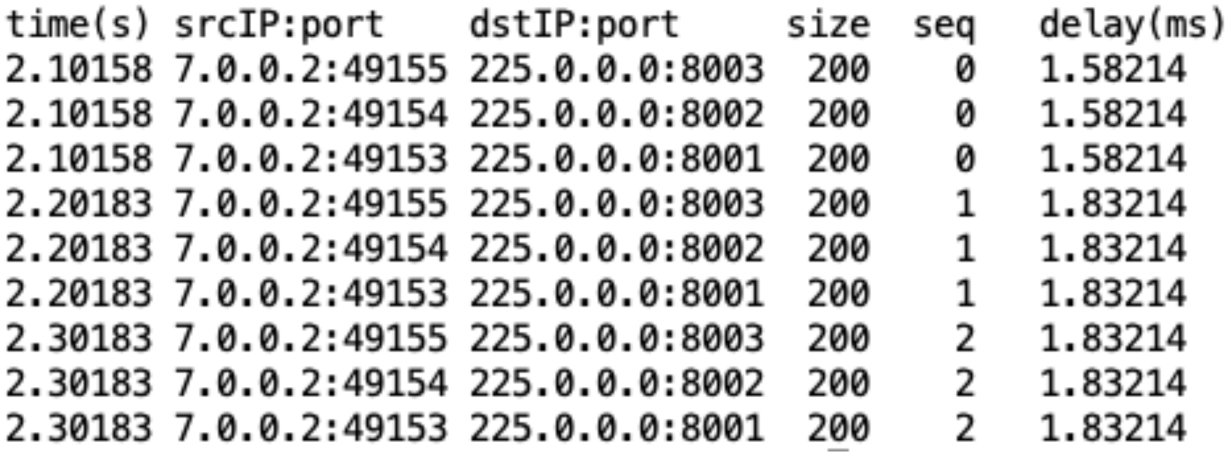
> Output: Terminal output and a delay trace

- Terminal output: Packets sent and received, and average latency (measured at application layer) across all flows. Sample:



```
Total Tx packets = 60
Total Rx packets = 60
Average packet delay = 1.74464 ms
```

- Delay trace (NrSlAppRxPacketDelayTrace.txt). Per-packet delay trace mesured at application layer. Sample:

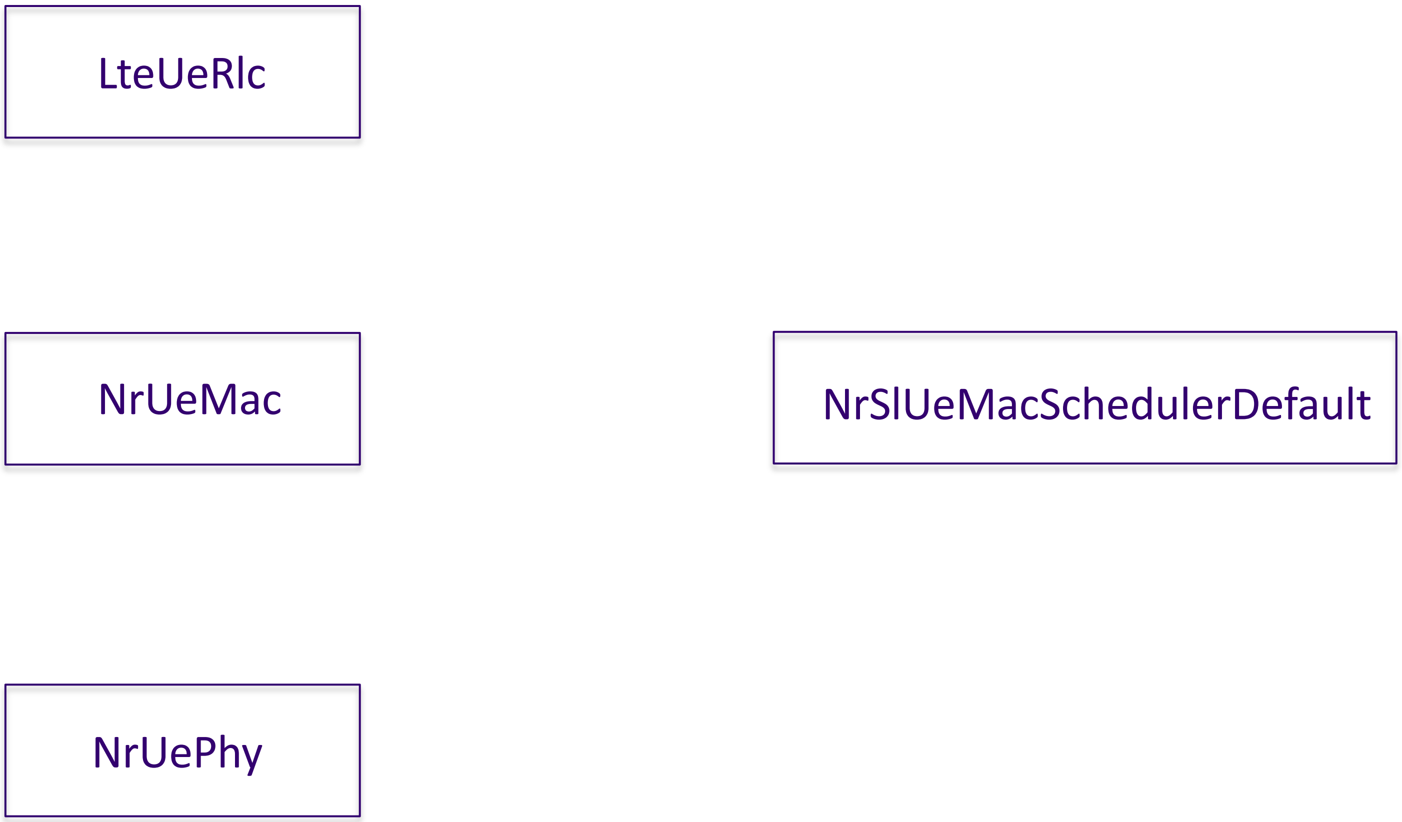


```
time(s) srcIP:port     dstIP:port      size  seq   delay(ms)
2.10158 7.0.0.2:49155 225.0.0.0:8003  200    0   1.58214
2.10158 7.0.0.2:49154 225.0.0.0:8002  200    0   1.58214
2.10158 7.0.0.2:49153 225.0.0.0:8001  200    0   1.58214
2.20183 7.0.0.2:49155 225.0.0.0:8003  200    1   1.83214
2.20183 7.0.0.2:49154 225.0.0.0:8002  200    1   1.83214
2.20183 7.0.0.2:49153 225.0.0.0:8001  200    1   1.83214
2.30183 7.0.0.2:49155 225.0.0.0:8003  200    2   1.83214
2.30183 7.0.0.2:49154 225.0.0.0:8002  200    2   1.83214
2.30183 7.0.0.2:49153 225.0.0.0:8001  200    2   1.83214
```

nr-sl-simple-multi-lc.cc parameter choices

Simulation configuration		Resulting traffic profile configuration per flow		
Parameter	Value	Flow 1	Flow 2	Flow 3
`schedTypeConfig`	1	Dynamic	Dynamic	Dynamic
`schedTypeConfig`	2	SPS	SPS	SPS
`schedTypeConfig`	3	Dynamic	Dynamic	SPS
`schedTypeConfig`	4	SPS	SPS	Dynamic
`dstL2IdConfig`	1	255	255	255
`dstL2IdConfig`	2	255	254	255
`dstL2IdConfig`	3	254	254	255
`priorityConfig`	1	1	1	1
`priorityConfig`	2	1	2	3
`priorityConfig`	3	2	2	1
`priorityConfig`	4	1	1	2
`rriConfig`	1	100	100	100
`rriConfig`	2	100	50	100

Typical sequence of simulation events



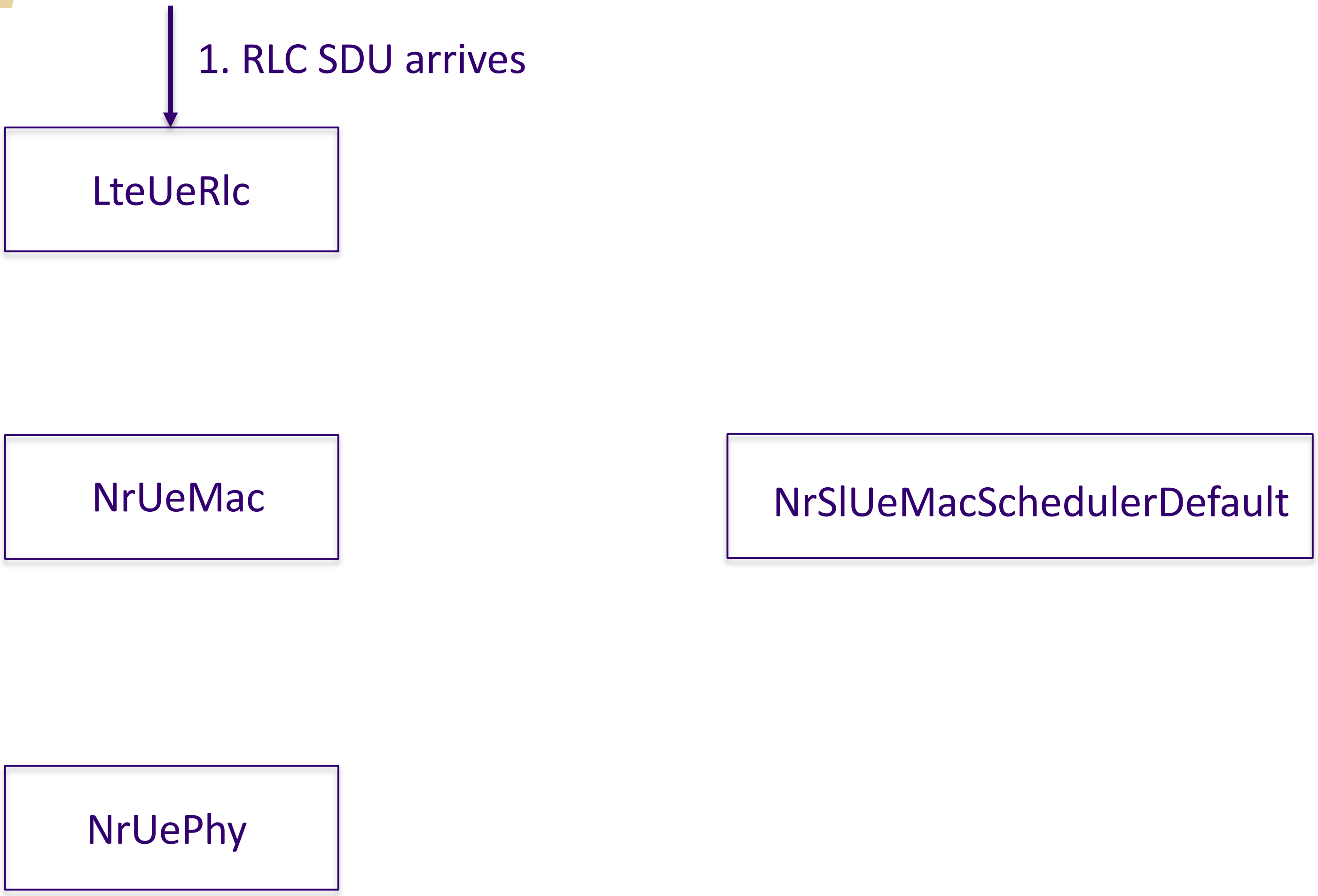
LteUeRlc

NrUeMac

NrSlUeMacSchedulerDefault

NrUePhy

Typical sequence of simulation events



1. RLC SDU arrives

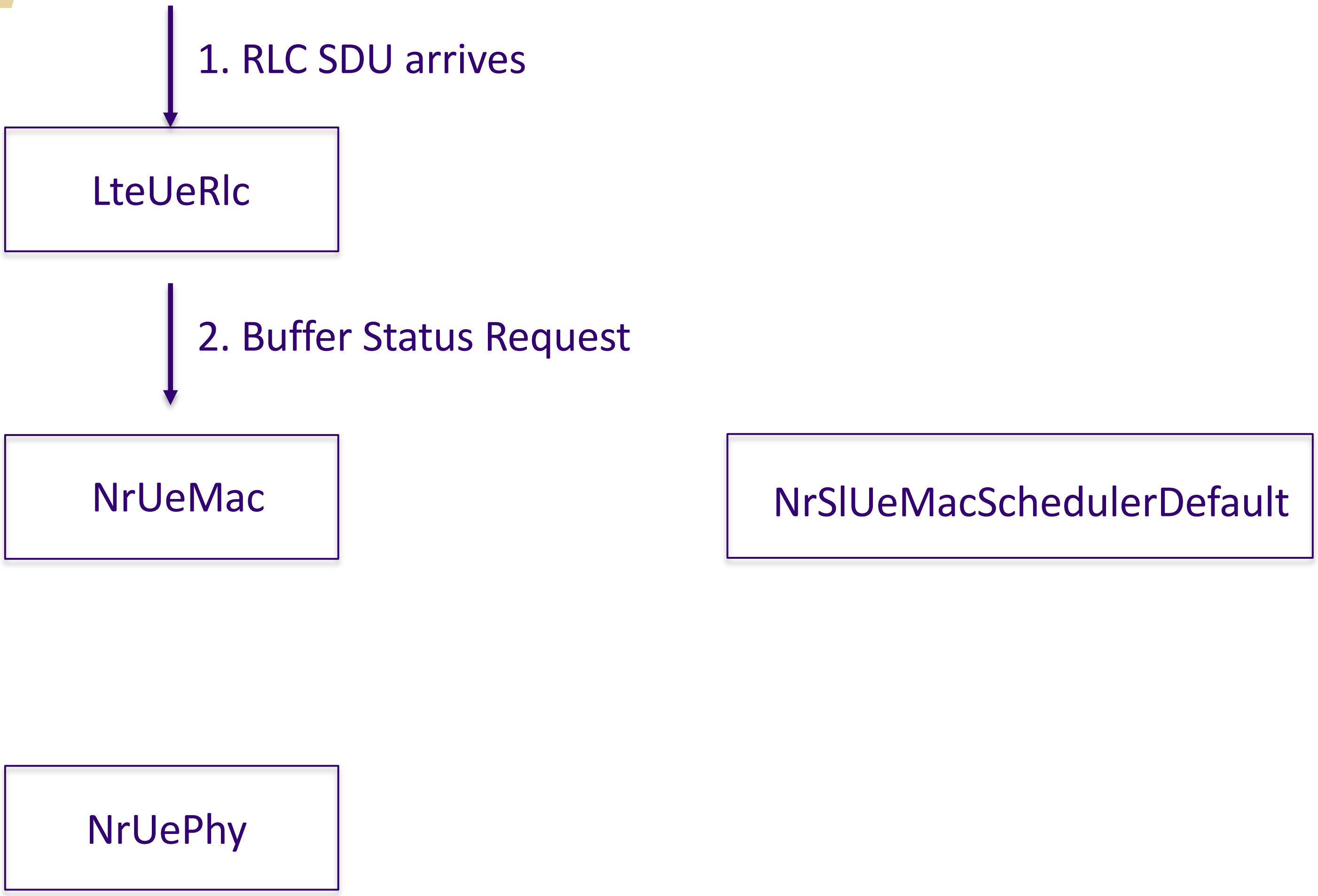
LteUeRlc

NrUeMac

NrSlUeMacSchedulerDefault

NrUePhy

Typical sequence of simulation events



1. RLC SDU arrives

LteUeRlc

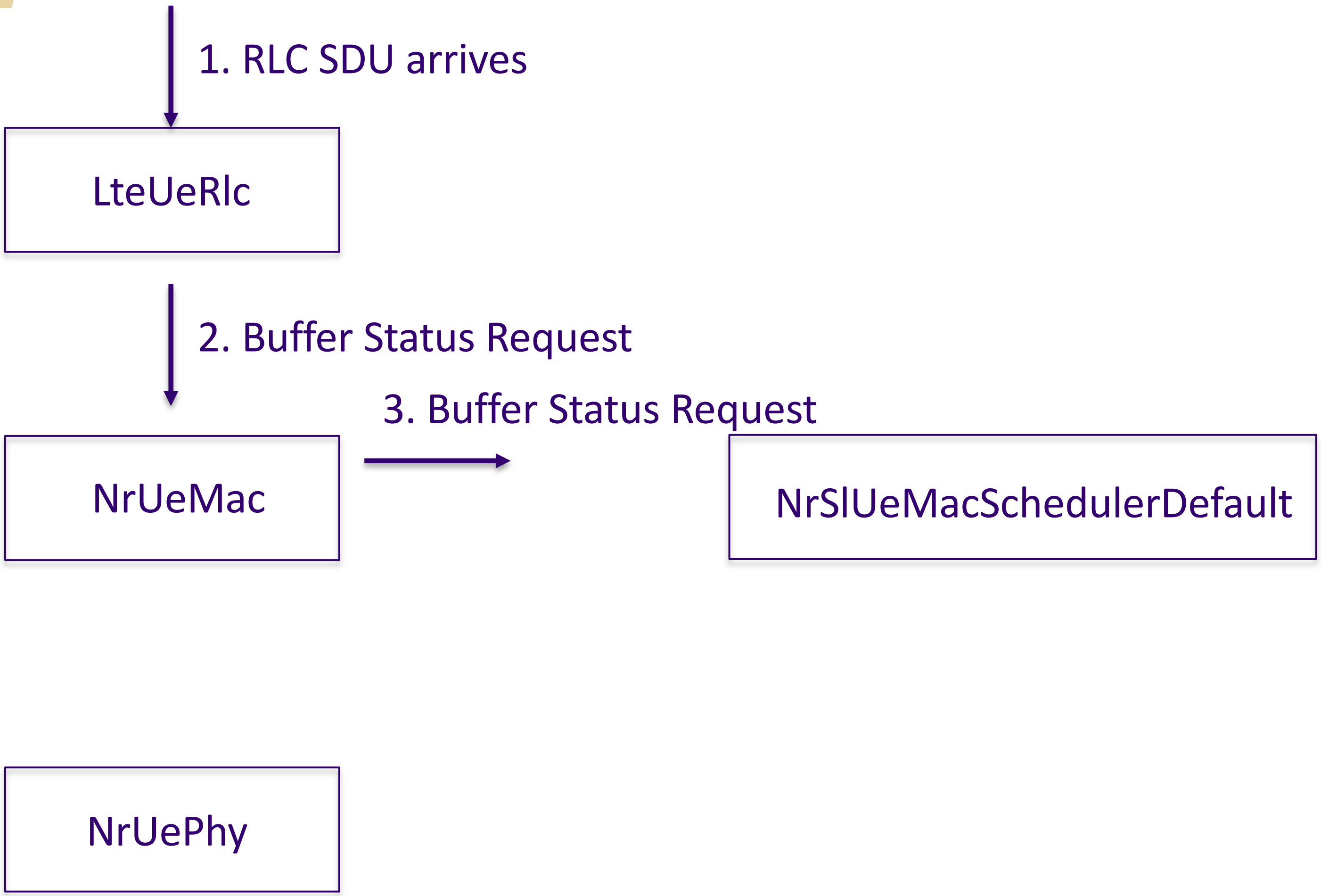
2. Buffer Status Request

NrUeMac

NrSlUeMacSchedulerDefault

NrUePhy

Typical sequence of simulation events



1. RLC SDU arrives

LteUeRlc

2. Buffer Status Request

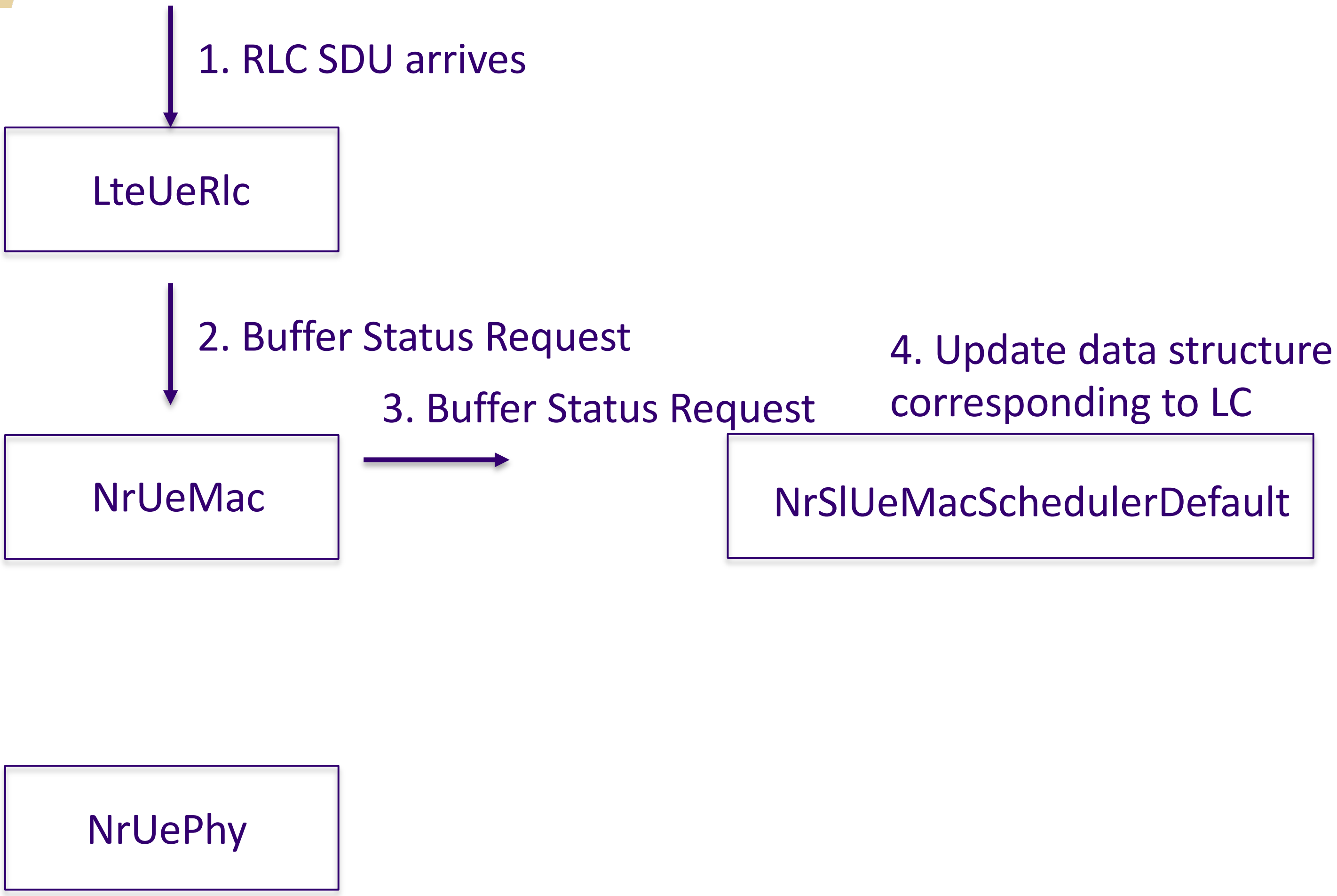
3. Buffer Status Request

NrUeMac

NrSlUeMacSchedulerDefault

NrUePhy

Typical sequence of simulation events



1. RLC SDU arrives

LteUeRlc

2. Buffer Status Request

NrUeMac

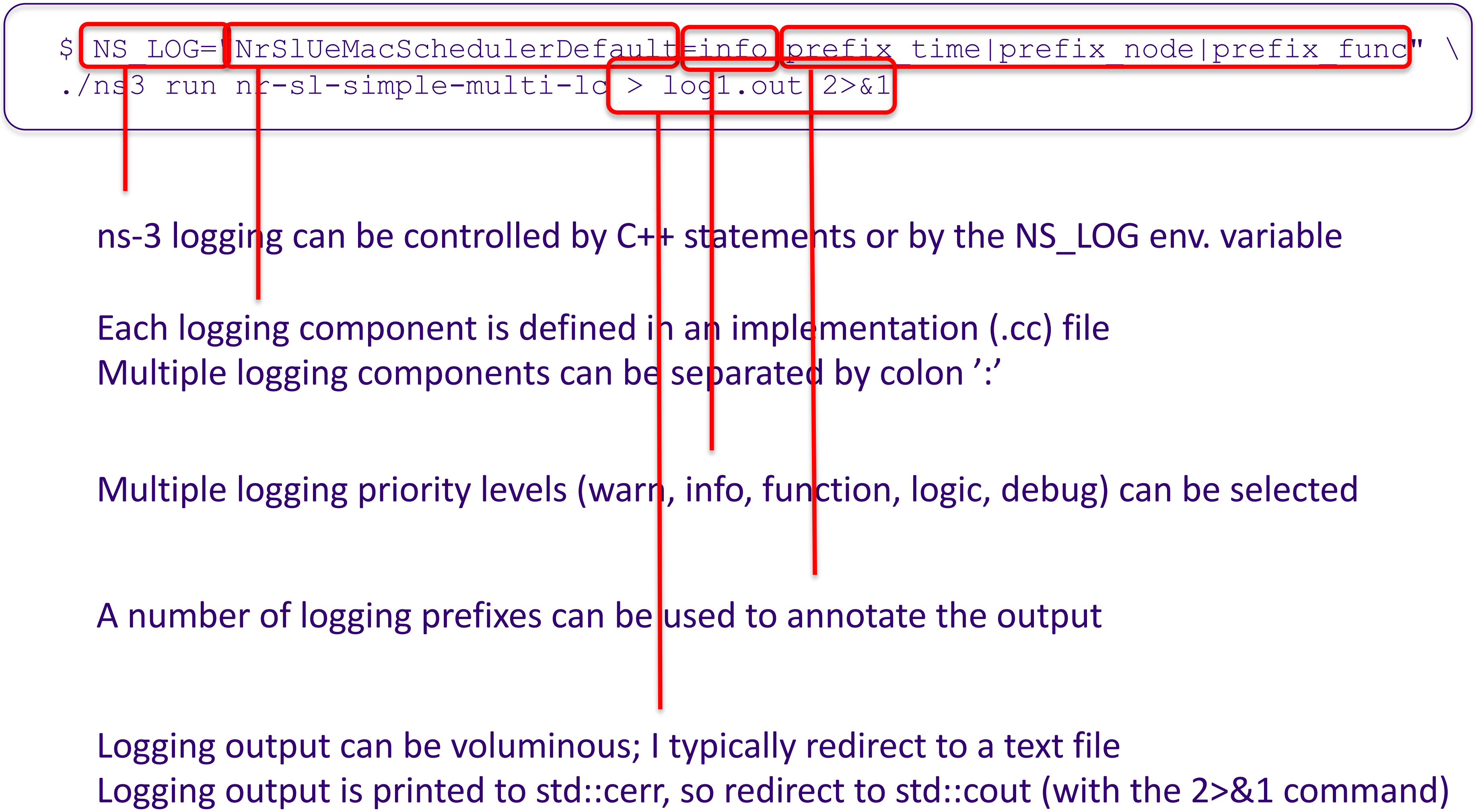
3. Buffer Status Request

4. Update data structure corresponding to LC

NrSlUeMacSchedulerDefault

NrUePhy

Log output



```
$ NS_LOG="NrSlUeMacSchedulerDefault=info|prefix_time|prefix_node|prefix_func" \
./ns3 run nr-sl-simple-multi-lc > log1.out 2>&1
```

ns-3 logging can be controlled by C++ statements or by the NS_LOG env. variable

Each logging component is defined in an implementation (.cc) file
Multiple logging components can be separated by colon ’:’

Multiple logging priority levels (warn, info, function, logic, debug) can be selected

A number of logging prefixes can be used to annotate the output

Logging output can be voluminous; I typically redirect to a text file
Logging output is printed to std::cerr, so redirect to std::cout (with the 2>&1 command)

Log output

```
$ NS_LOG="NrSlUeMacSchedulerDefault=info|prefix_time|prefix_node|prefix_func" \
./ns3 run nr-sl-simple-multi-lc > log1.out 2>&1
```

Log output

```
$ NS_LOG="NrSlUeMacSchedulerDefault=info|prefix_time|prefix_node|prefix_func" \
./ns3 run nr-sl-simple-multi-lc > log1.out 2>&1
```

ns-3 logging can be controlled by C++ statements or by the NS_LOG env. variable

Log output

```
$ NS_LOG="NrSlUeMacSchedulerDefault=info|prefix_time|prefix_node|prefix_func" \
./ns3 run nr-sl-simple-multi-lc > log1.out 2>&1
```

ns-3 logging can be controlled by C++ statements or by the NS_LOG env. variable

Each logging component is defined in an implementation (.cc) file
Multiple logging components can be separated by colon ’:’

Log output

```
$ NS_LOG="NrSlUeMacSchedulerDefault=info|prefix_time|prefix_node|prefix_func" \
./ns3 run nr-sl-simple-multi-lc > log1.out 2>&1
```

ns-3 logging can be controlled by C++ statements or by the NS_LOG env. variable

Each logging component is defined in an implementation (.cc) file
Multiple logging components can be separated by colon ’:’

Multiple logging priority levels (warn, info, function, logic, debug) can be selected

Log output

```
$ NS_LOG="NrSlUeMacSchedulerDefault=info|prefix_time|prefix_node|prefix_func" \
./ns3 run nr-sl-simple-multi-lc > log1.out 2>&1
```

ns-3 logging can be controlled by C++ statements or by the NS_LOG env. variable

Each logging component is defined in an implementation (.cc) file
Multiple logging components can be separated by colon ':'

Multiple logging priority levels (warn, info, function, logic, debug) can be selected

A number of logging prefixes can be used to annotate the output

Log output

```
$ NS_LOG="NrSlUeMacSchedulerDefault=info|prefix_time|prefix_node|prefix_func" \
./ns3 run nr-sl-simple-multi-lc > log1.out 2>&1
```

ns-3 logging can be controlled by C++ statements or by the NS_LOG env. variable

Each logging component is defined in an implementation (.cc) file
Multiple logging components can be separated by colon ’:’

Multiple logging priority levels (warn, info, function, logic, debug) can be selected

A number of logging prefixes can be used to annotate the output

Logging output can be voluminous; I typically redirect to a text file
Logging output is printed to std::cerr, so redirect to std::cout (with the 2>&1 command)

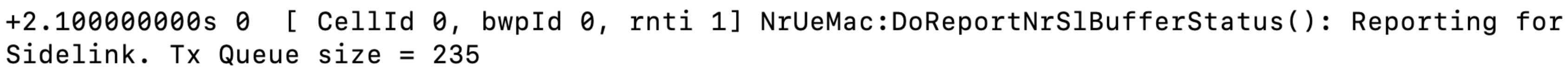
Log output: Buffer Status Request

```
$
NS_LOG="NrSlUeMacSchedulerDefault=info|prefix_time|prefix_node|prefix_func:
NrUeMac=info|prefix_time|prefix_node|prefix_func:NrSlUeMacSchedulerLCG=info
|prefix_time|prefix_node|prefix_func" ./ns3 run nr-sl-simple-multi-lc >
log1.out 2>&1
```

1. RLC SDU arrives

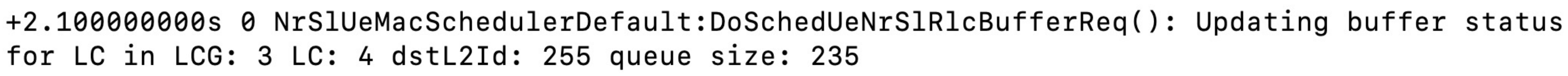
Not shown... in LTE module (logs not cleaned up yet)

2. NrUeMac BufferStatusRequest (line 17)



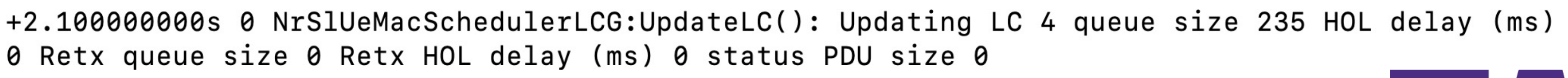
```
+2.100000000s 0  [ CellId 0, bwpId 0, rnti 1] NrUeMac:DoReportNrSlBufferStatus(): Reporting for
Sidelink. Tx Queue size = 235
```

3. NrSlUeMacSchedulerDefault BufferStatusRequest (line 18)



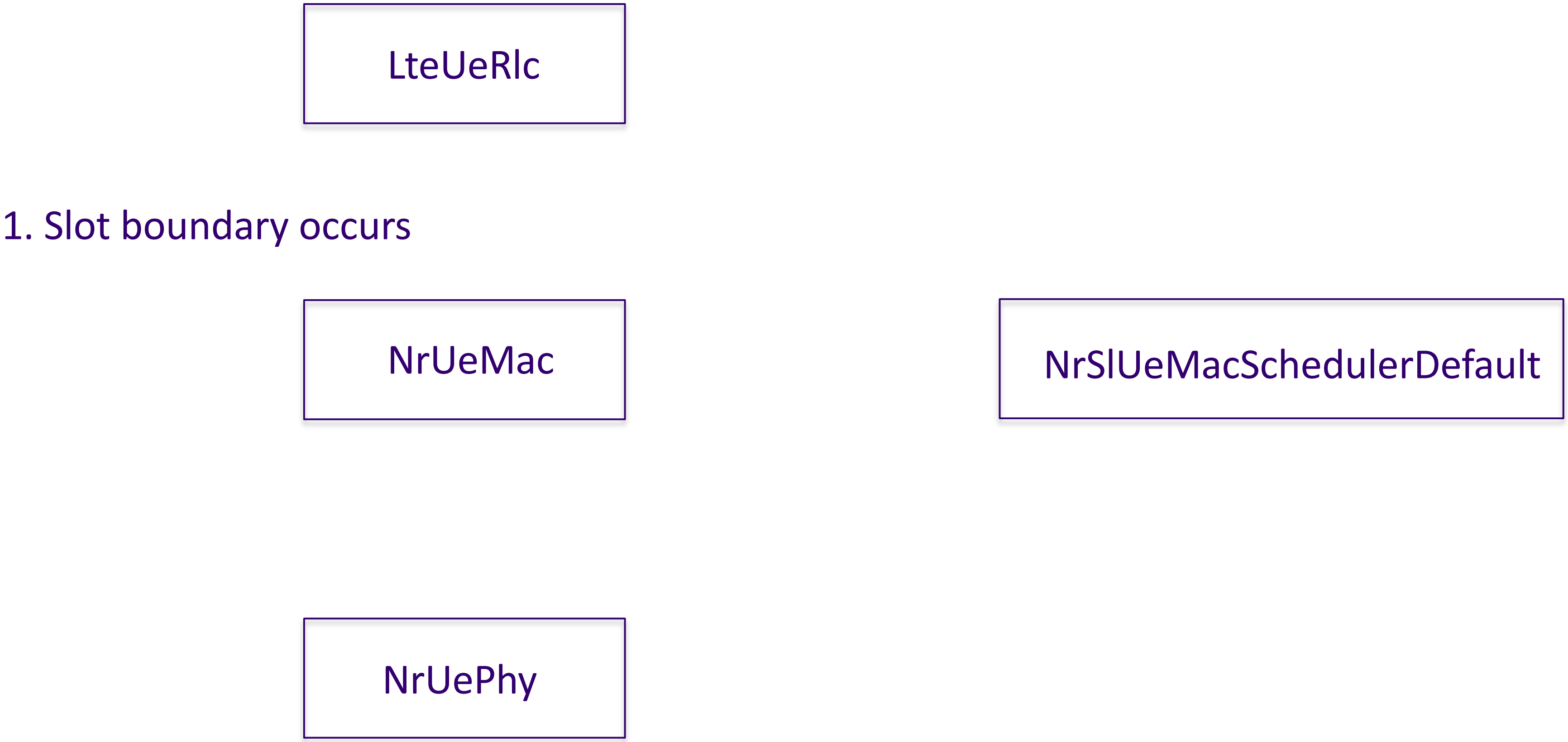
```
+2.100000000s 0 NrSlUeMacSchedulerDefault:DoSchedUeNrSlRlcBufferReq(): Updating buffer status
for LC in LCG: 3 LC: 4 dstL2Id: 255 queue size: 235
```

4. Update LC/DstL2Id structures (line 19)



```
+2.100000000s 0 NrSlUeMacSchedulerLCG:UpdateLC(): Updating LC 4 queue size 235 HOL delay (ms)
0 Retx queue size 0 Retx HOL delay (ms) 0 status PDU size 0
```

Typical sequence of simulation events



LteUeRlc

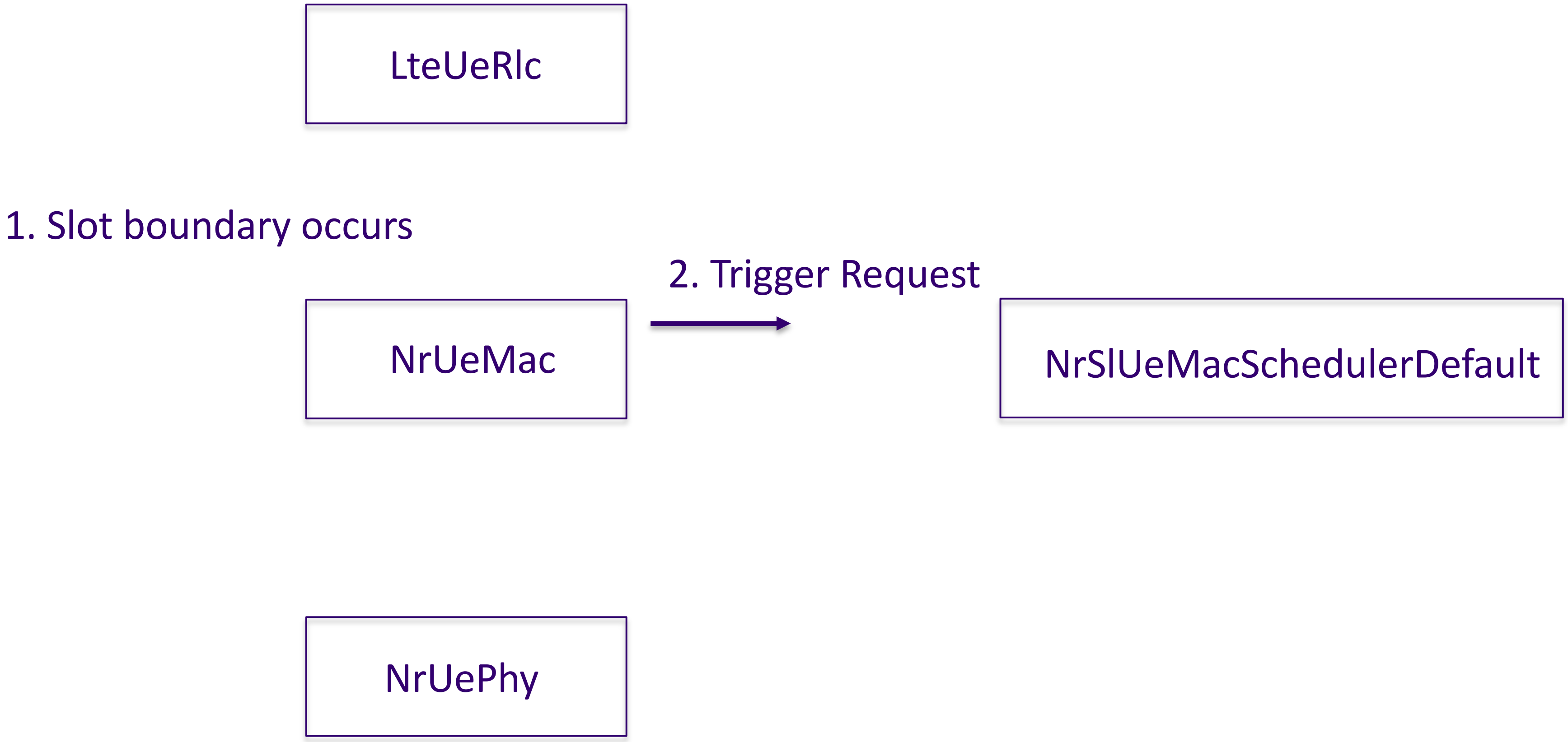
1. Slot boundary occurs

NrUeMac

NrSlUeMacSchedulerDefault

NrUePhy

Typical sequence of simulation events



LteUeRlc

1. Slot boundary occurs

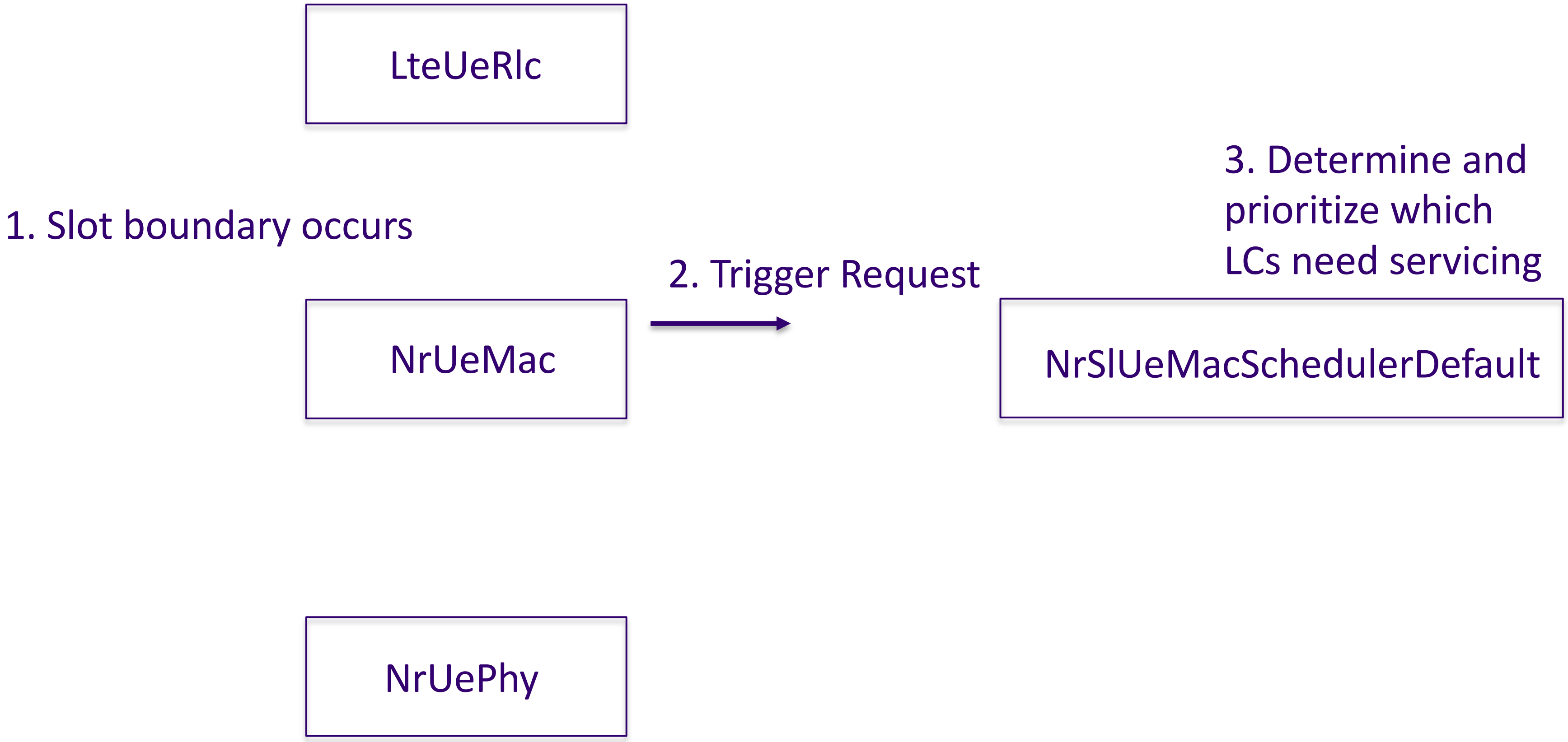
2. Trigger Request

NrUeMac

NrSlUeMacSchedulerDefault

NrUePhy

Typical sequence of simulation events



LteUeRlc

1. Slot boundary occurs

NrUeMac

2. Trigger Request

3. Determine and
prioritize which
LCs need servicing

NrSlUeMacSchedulerDefault

NrUePhy

Logical channel prioritization

- When multiple LCs need to be scheduled at the same time, some prioritization order is needed
- The explicit priority (QoS) of LCs (encoded in the SidelinkInfo of the LC) can be used as the primary prioritization
- When LCs have the same QoS priority, other priortization heuristics are needed
  - e.g., closest deadline to packet delay budget, largest queue, longest time since allocation, round robin
- Some high priority LCs may not be able to be scheduled if there is congestion, despite pre-existing grants to lower priority LCs
- A feature called ‘preemption’ (not implemented) can override

Logical channel prioritization (cont.)

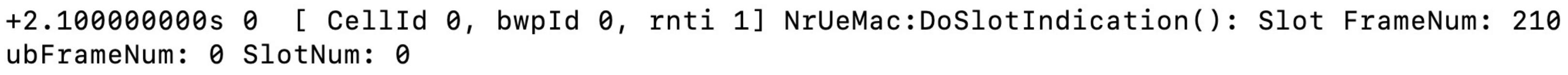
The current ns-3 implementation prioritizes first by sorting destinations, then by LC within each destination:

1. Select the destination:
    1. with the LC with the highest priority
    2. if multiple destination share the same highest priority, select the one with the smallest dstL2Id
2. Select the LC to that destination with highest priority
3. Select all LCs with the same grant attributes (scheduling type, scheduling attributes, and HARQ feedback type) as the LC with highest priority
    1. if multiple LCs with different scheduling type share the same highest priority, select the one(s) with scheduling type priority indicated by m_prioToSps attribute
    2. if m_prioToSps and multiple LCs with SPS scheduling type and different RRI share the same highest priority, select the one(s) with RRI equal to the LC with lowest LcId

Log output: Trigger request

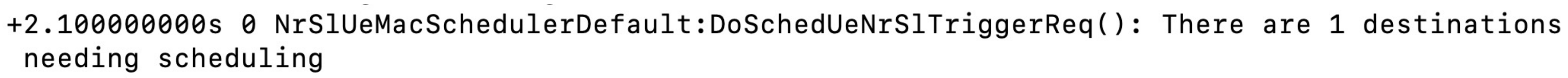
```
$
NS_LOG="NrSlUeMacSchedulerDefault=logic|prefix_time|prefix_node|prefix_func
:NrUeMac=debug|prefix_time|prefix_node|prefix_func" ./ns3 run nr-sl-simple-
multi-lc > log2.out 2>&1
```

1. Slot boundary occurs (line 39197)



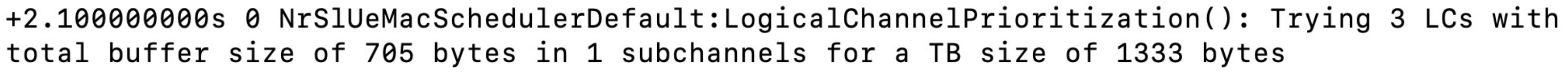
```
+2.100000000s 0  [ CellId 0, bwpId 0, rnti 1] NrUeMac:DoSlotIndication(): Slot FrameNum: 210
ubFrameNum: 0 SlotNum: 0
```

2. Trigger request (line 39205)



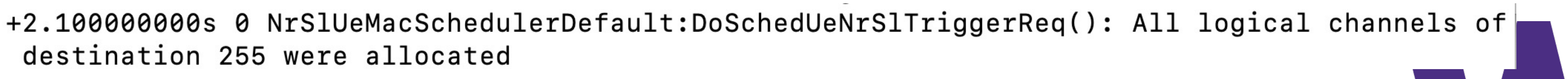
```
+2.100000000s 0 NrSlUeMacSchedulerDefault:DoSchedUeNrSlTriggerReq(): There are 1 destinations
 needing scheduling
```

3. Prioritize LCs in scheduler (line 39212-39252)



```
+2.100000000s 0 NrSlUeMacSchedulerDefault:LogicalChannelPrioritization(): Trying 3 LCs with
total buffer size of 705 bytes in 1 subchannels for a TB size of 1333 bytes
```

...



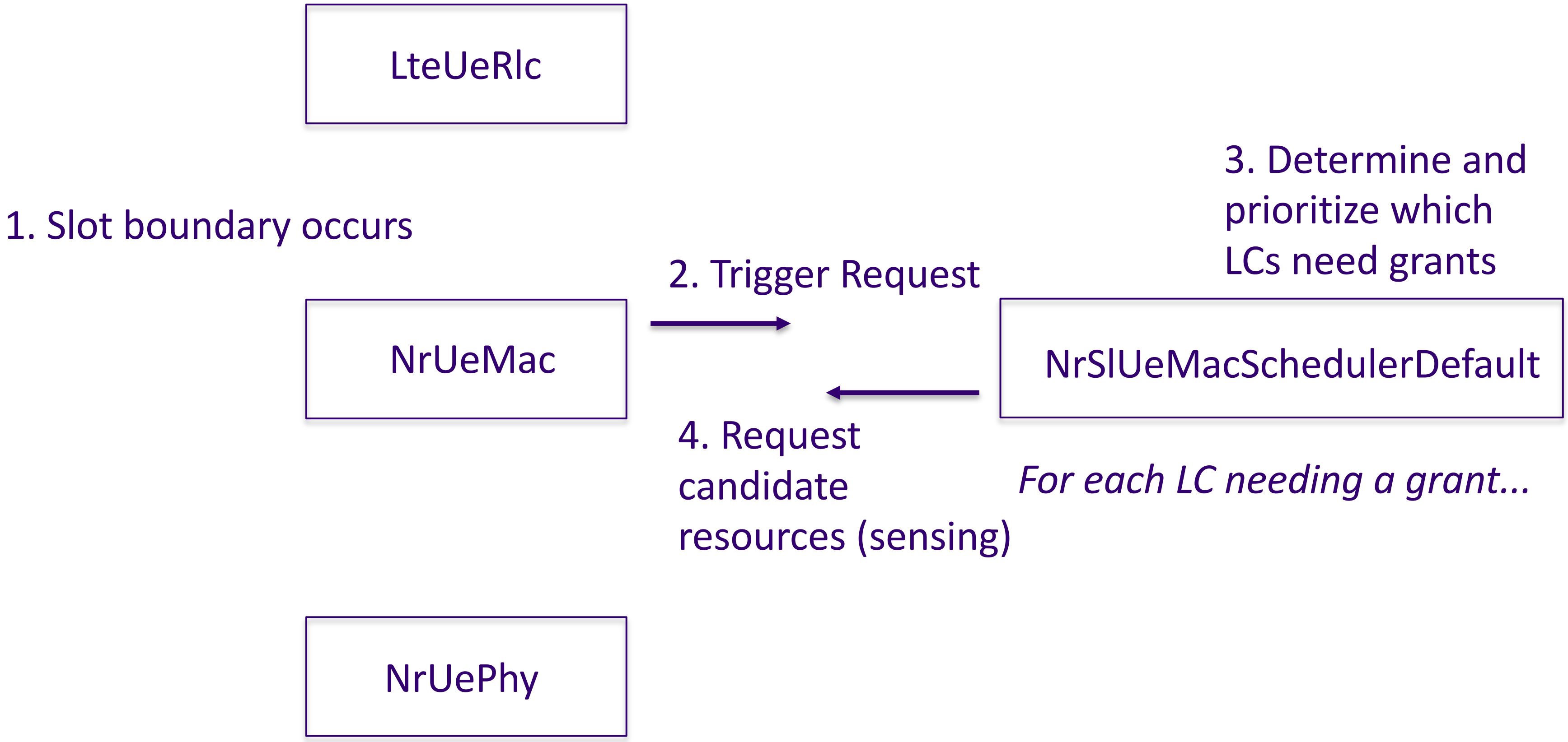
```
+2.100000000s 0 NrSlUeMacSchedulerDefault:DoSchedUeNrSlTriggerReq(): All logical channels of
 destination 255 were allocated
```

nr-sl-simple-multi-lc.cc parameter choices

Simulation configuration		Resulting traffic profile configuration per flow		
Parameter	**Value**	**Flow 1**	**Flow 2**	**Flow 3**
`schedTypeConfig`	1	Dynamic	Dynamic	Dynamic
`schedTypeConfig`	2	SPS	SPS	SPS
`schedTypeConfig`	3	Dynamic	Dynamic	SPS
`schedTypeConfig`	4	SPS	SPS	Dynamic
`dstL2IdConfig`	1	255	255	255
`dstL2IdConfig`	2	255	254	255
`dstL2IdConfig`	3	254	254	255
`priorityConfig`	1	1	1	1
`priorityConfig`	2	1	2	3
`priorityConfig`	3	2	2	1
`priorityConfig`	4	1	1	2
`rriConfig`	1	100	100	100
`rriConfig`	2	100	50	100

All three LCs in this (default) example are scheduled in the same grant, because they each have the same LC properties

Typical sequence of simulation events



LteUeRlc

1. Slot boundary occurs

NrUeMac

2. Trigger Request

3. Determine and prioritize which LCs need grants

NrSlUeMacSchedulerDefault

4. Request candidate resources (sensing)

For each LC needing a grant...

NrUePhy

Sensing overview

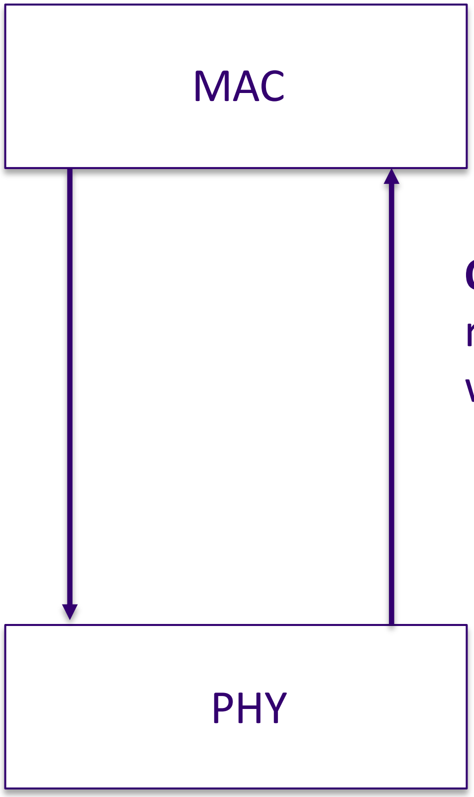
- Sensing is based on a combination of 1) PSCCH decoded (SCI 1-A), and 2) RSRP measured in the slots within window
- NR Sidelink is assumed to be a half-duplex device--cannot transmit and receive in the same (time) slot
- Sensing for NR SL Mode 2 is specified in a six-step algorithm in 3GPP TS 38.214

Sensing overview (cont.)

> Described as a MAC/PHY interaction in the specification, to obtain candidate resources for scheduling

Inputs:

- resource pool
- priority
- packet delay budget
- number of subchannels
- RRI
- cResel



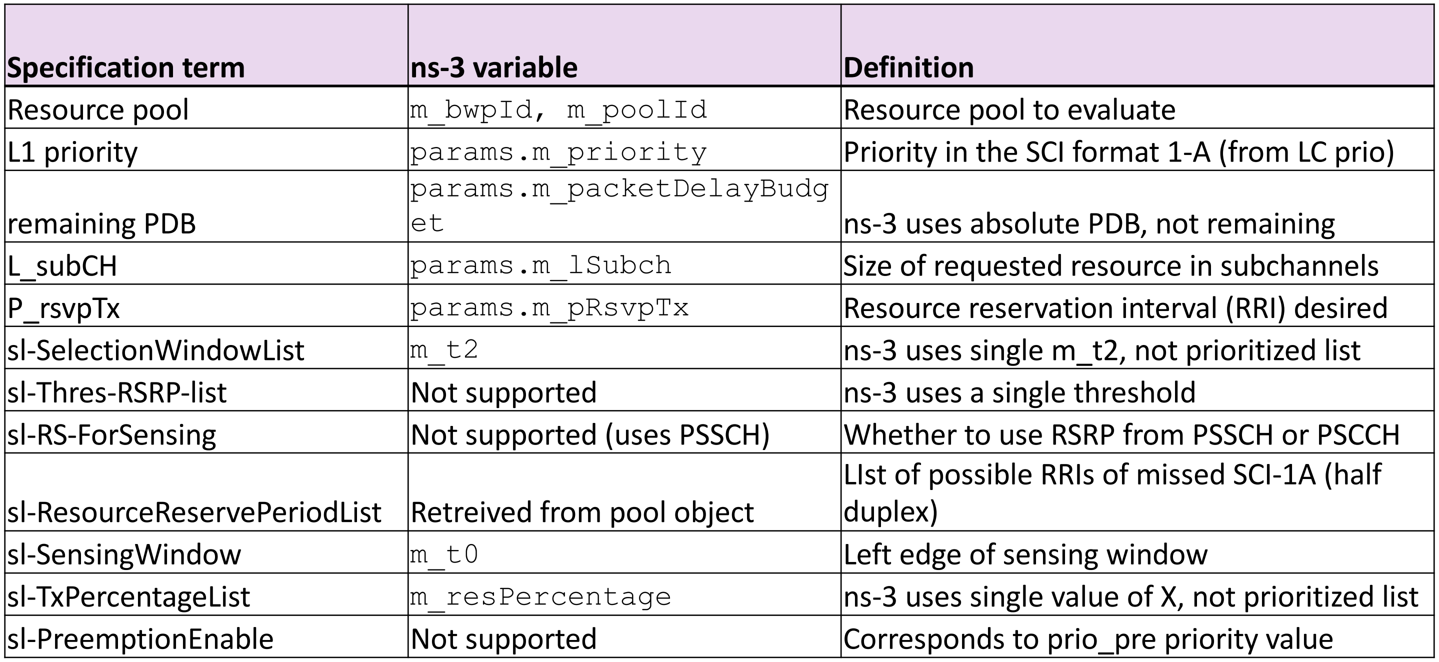
MAC

PHY

Output: Set of candidate resources in the selection window

Sensing implementation in ns-3

- `NrUeMac::NrUeMac::GetNrSlCandidateResourcesPrivate()` follows 3GPP TS 38.214 Section 8.1.4 closely
- Algorithm inputs:

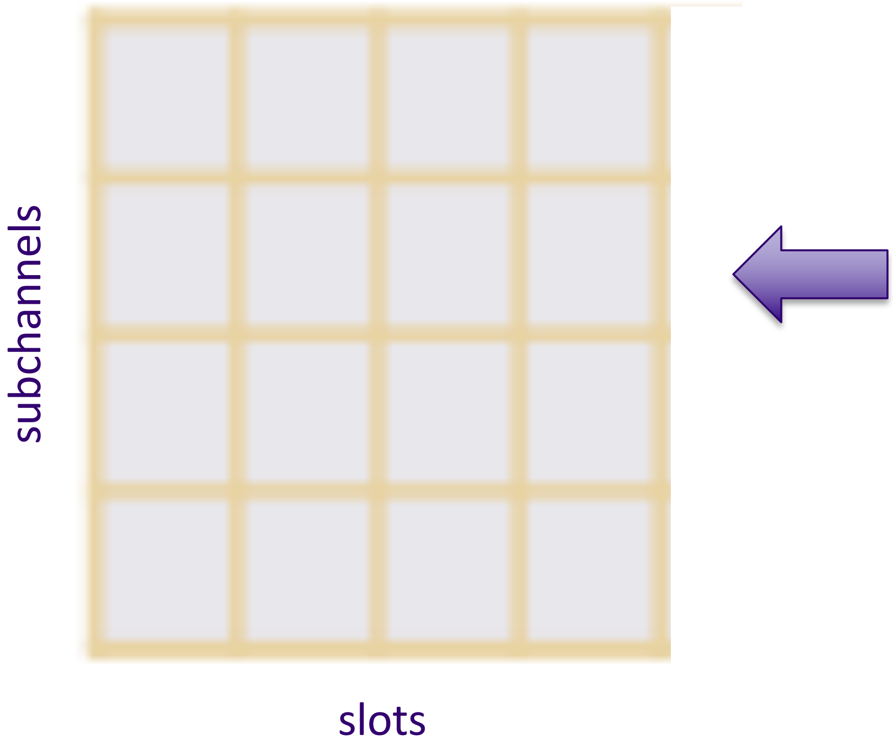


Specification term	ns-3 variable	Definition
Resource pool	`m_bwpId, m_poolId`	Resource pool to evaluate
L1 priority	`params.m_priority`	Priority in the SCI format 1-A (from LC prio)
remaining PDB	`params.m_packetDelayBudget`	ns-3 uses absolute PDB, not remaining
L_subCH	`params.m_lSubch`	Size of requested resource in subchannels
P_rsvpTx	`params.m_pRsvpTx`	Resource reservation interval (RRI) desired
sl-SelectionWindowList	`m_t2`	ns-3 uses single m_t2, not prioritized list
sl-Thres-RSRP-list	Not supported	ns-3 uses a single threshold
sl-RS-ForSensing	Not supported (uses PSSCH)	Whether to use RSRP from PSSCH or PSCCH
sl-ResourceReservePeriodList	Retreived from pool object	LIst of possible RRIs of missed SCI-1A (half duplex)
sl-SensingWindow	`m_t0`	Left edge of sensing window
sl-TxPercentageList	`m_resPercentage`	ns-3 uses single value of X, not prioritized list
sl-PreemptionEnable	Not supported	Corresponds to prio_pre priority value

- Algorithm outputs list S_A of all candidate single-slot resources

Interpretation of ”all candidate resources”

- Candidate resources may overlap with each other
- Example request for resource with three subchannels

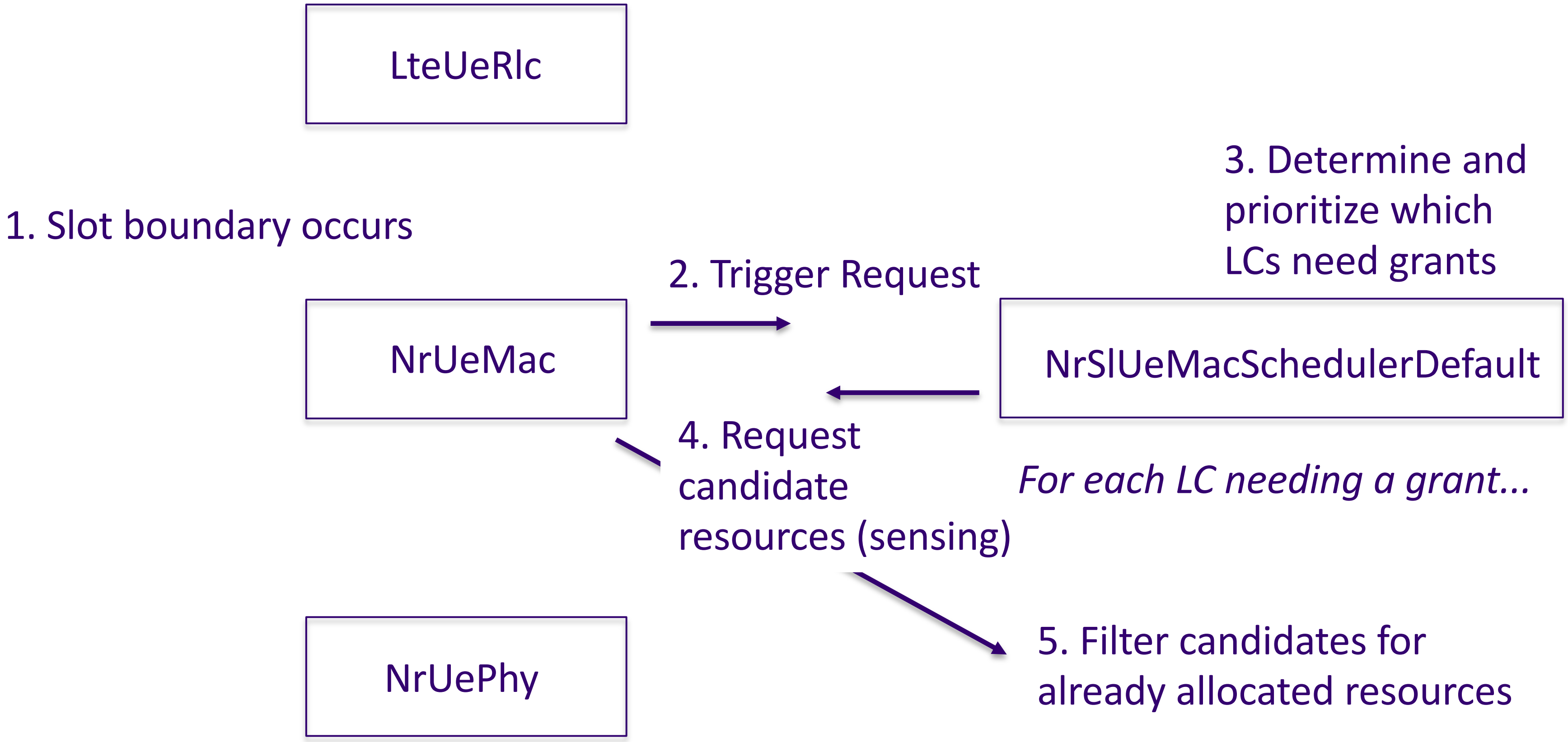


There are eight candidate three-subchannel single-slot resources (with overlap)

Algorithm summary (TS 38.214 Section 8.1.4)

- (Step 4) Initialize S_A to all candidate resources
- (Step 5-- ***for missed sensing opportunities***) For all resources that were not listened to (due to transmitting in that slot), use the ResourceReservePeriodList to exclude, from S_A, all possible future SPS transmissions that may have been unheard
  - If resulting S_A is less than a threshold percentage (e.g. 50%), restore S_A to its value from Step 4
- (Step 6-- ***for decoded signals***) For all resources known in the sensing window due to received SCI-1A (and PSSCH):
  - If the received RSRP > current RSRP threshold, then exclude future projections of those resources from S_A
- (Step 7) If resulting S_A is less than a threshold percentage, increase threshold by 3 dB and go to step 4

Typical sequence of simulation events



LteUeRlc

1. Slot boundary occurs

NrUeMac

2. Trigger Request

3. Determine and prioritize which LCs need grants

NrSlUeMacSchedulerDefault

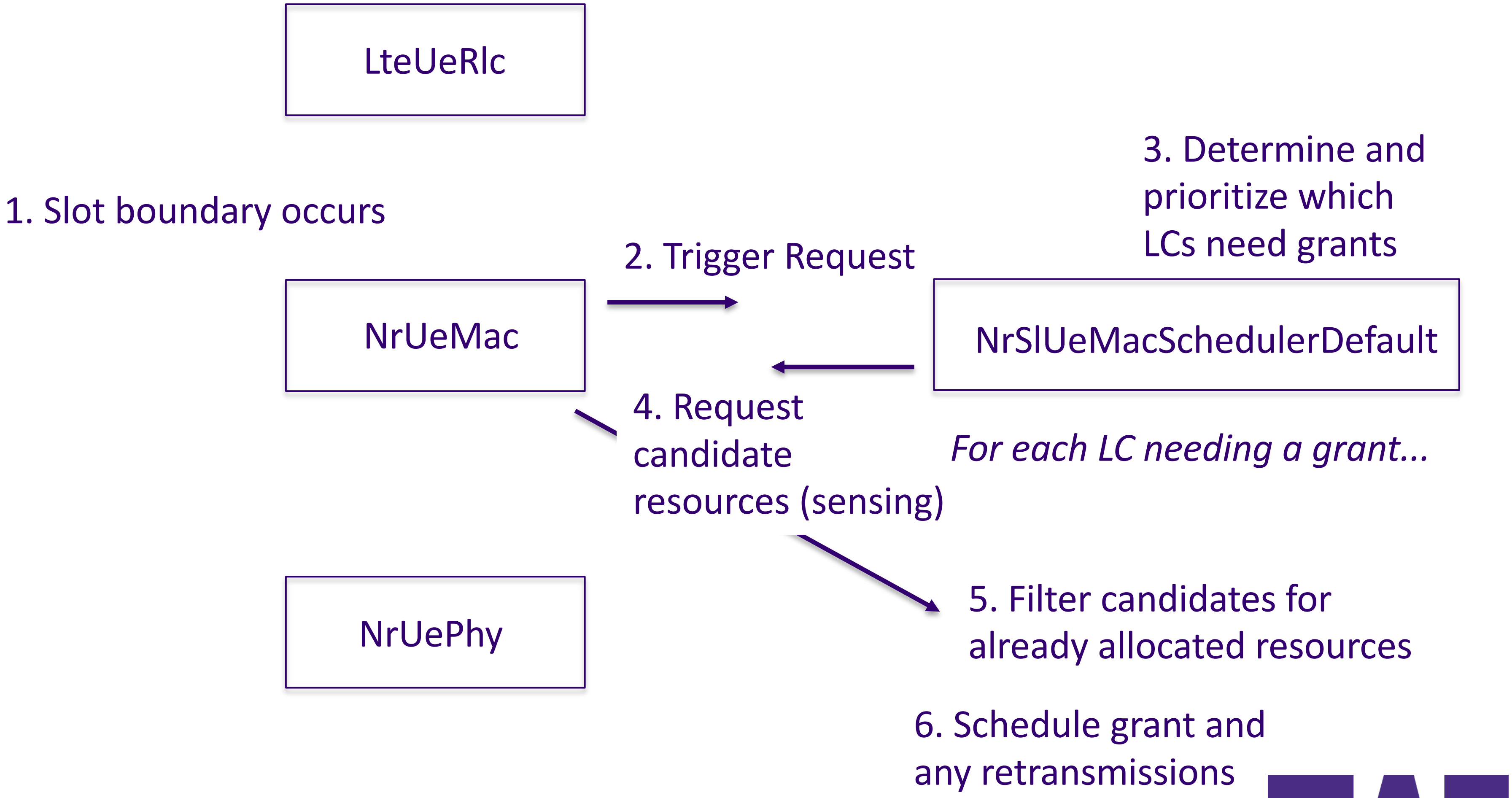
4. Request candidate resources (sensing)

For each LC needing a grant…

5. Filter candidates for already allocated resources

NrUePhy

Typical sequence of simulation events



LteUeRlc

1. Slot boundary occurs

NrUeMac

2. Trigger Request

3. Determine and prioritize which LCs need grants

NrSlUeMacSchedulerDefault

4. Request candidate resources (sensing)

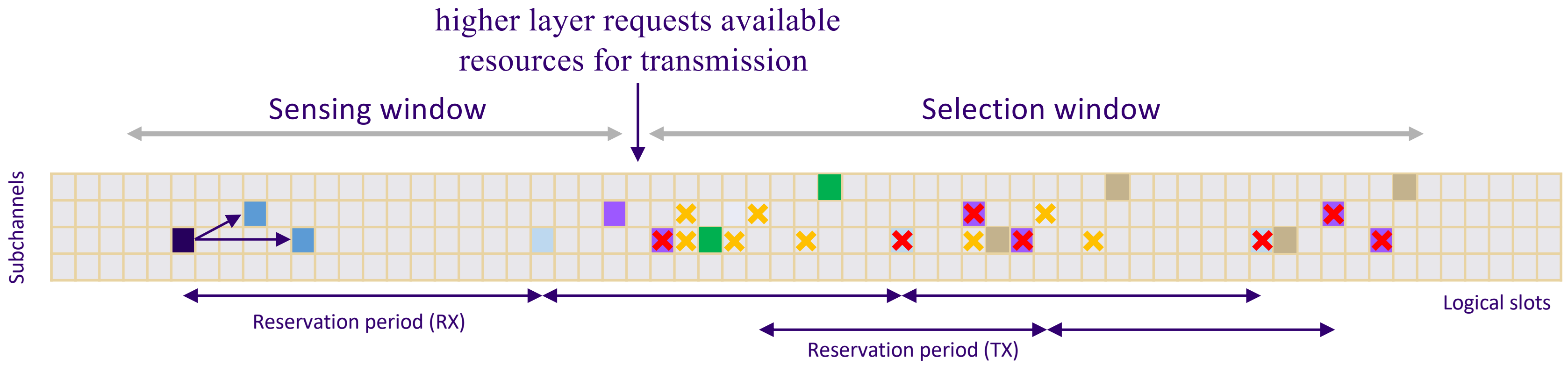
For each LC needing a grant...

NrUePhy

5. Filter candidates for already allocated resources

6. Schedule grant and any retransmissions

Sensing and Resource Selection



higher layer requests available resources for transmission

Sensing window

Selection window

Subchannels

Logical slots

Reservation period (RX)

Reservation period (TX)

■ Resources indicated by PSCCH (from 1st-stage SCI)

🟪 Resources indicated through reservation period (from 1st-stage SCI)

❌ Resource excluded due to direct collision

✖ Resource excluded due to collisions in future transmissions

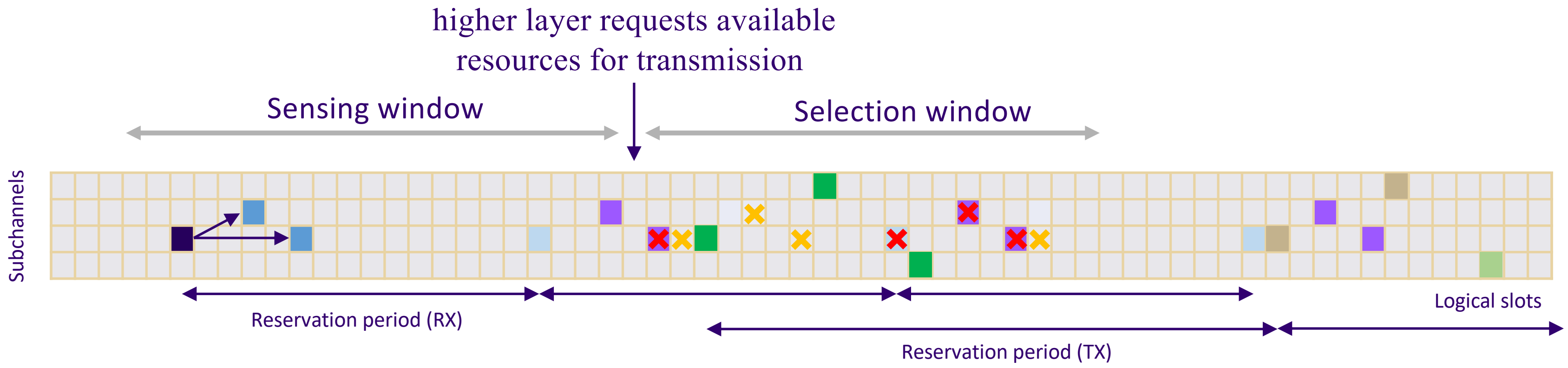
🟩 Resource randomly selected

🟫 Resource selected through reservation period

Additional factors affecting the sensing and selection process include:

- Priorities of the transmissions
- Preemption capabilities
- Signal strength
- Remaining packet delay budget
- Size of the transmission (i.e., number of subchannels)
- Half duplex mode

Sensing and Resource Selection



higher layer requests available resources for transmission

Sensing window

Selection window

Subchannels

Reservation period (RX)

Logical slots

Reservation period (TX)

■ Resources indicated by PSCCH (from 1st-stage SCI)

🟪 Resources indicated through reservation period (from 1st-stage SCI)

❌ Resource excluded due to direct collision

✖ Resource excluded due to collisions in future transmissions

🟩 Resource randomly selected

🟫 Resource selected through reservation period

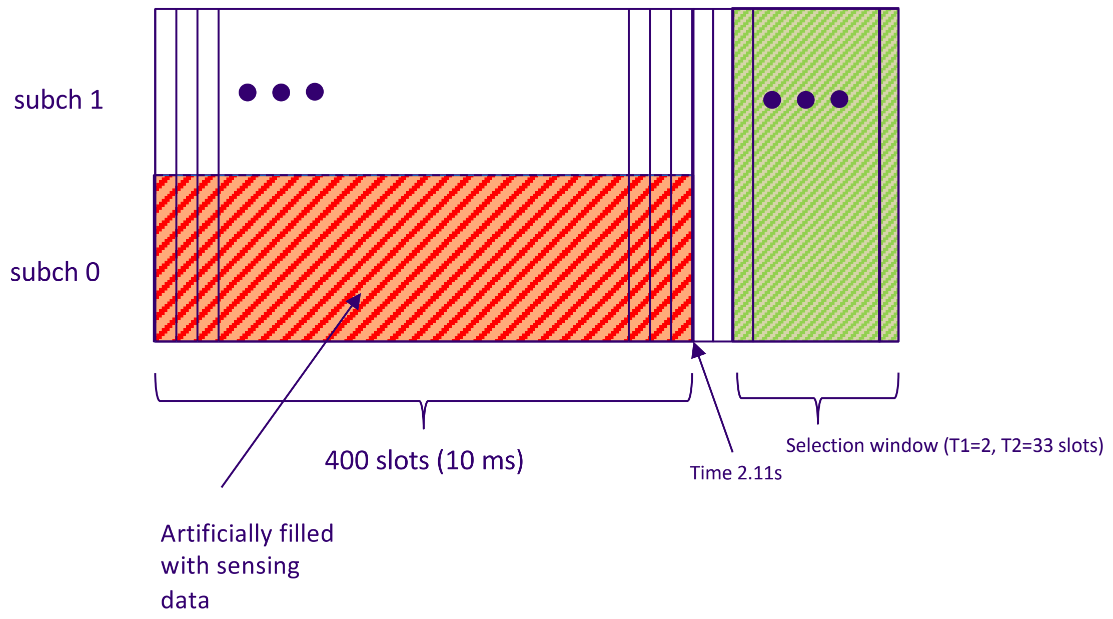
Additional factors affecting the sensing and selection process include:

- Priorities of the transmissions
- Preemption capabilities
- Signal strength
- Remaining packet delay budget
- Size of the transmission (i.e., number of subchannels)
- Half duplex mode

Recent sensing improvements

- Current nr-v2x branch code selects candidate ‘slots’, not candidate ‘resources’
- For simulations with a single subchannel, no difference, but necessary for simulations with multiple subchannels

Test scenario to illustrate the difference



subch 1
subch 0
400 slots (10 ms)
Time 2.11s
Selection window (T1=2, T2=33 slots)
Artificially filled with sensing data

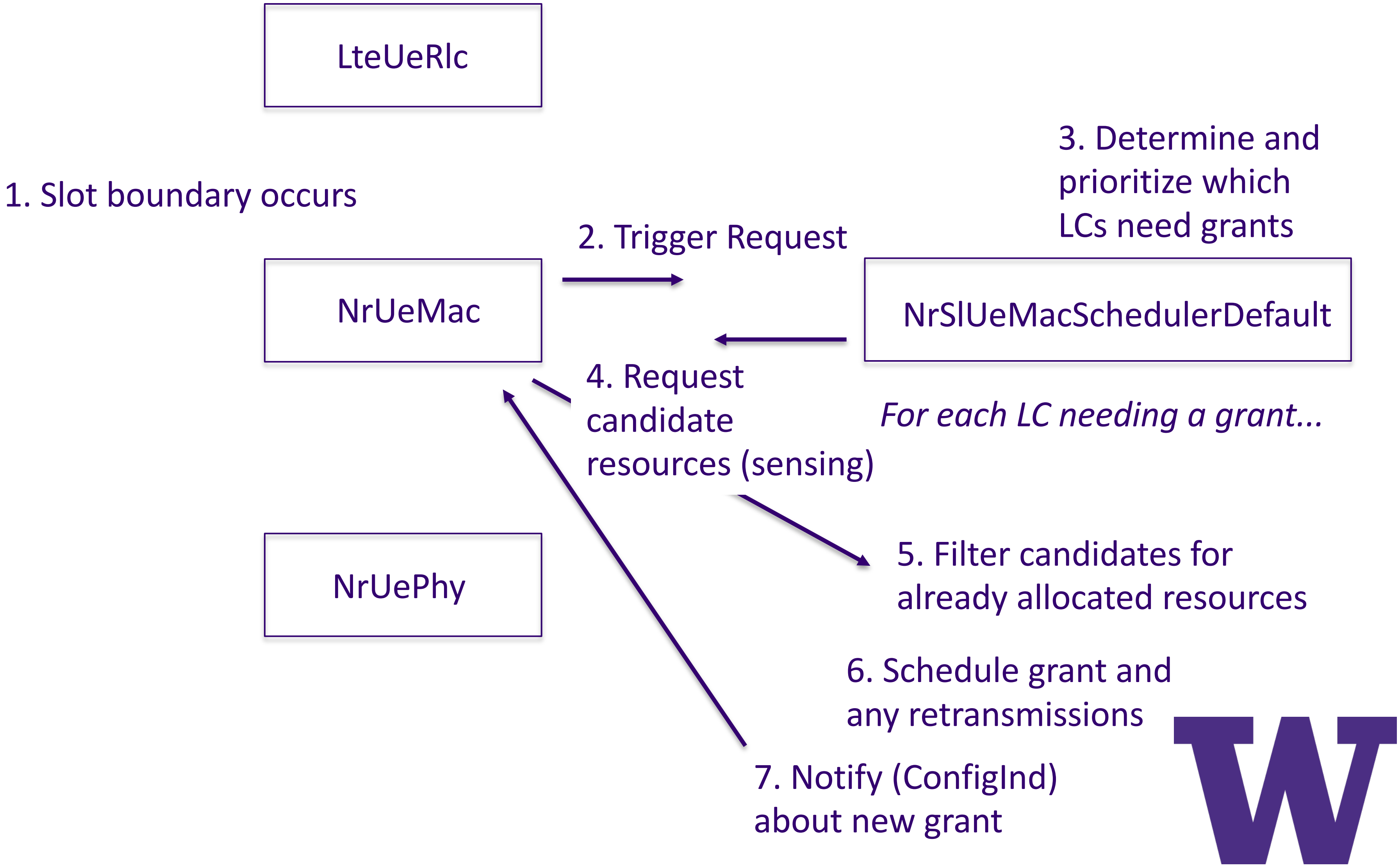
Scheduling (resource selection)

- Previous ns-3 scheduler implementation was called NrSlUeMacSchedulerSimple
  - Capable of semi-persistent scheduling only
  - Capable of handling single LC only
  - Used same assumption of sensing; logic based on slots occupied, not resources occupied (in multi-subchannel case)
  - Only supported blind retransmissions (HARQ-based retransmissions not supported)
- Proposed new scheduler implementation, NrSlUeMacSchedulerDefault, removes above limitations
- Schedulers are expected to be an area of active research by users

Scheduling (resource selection)

- Some features are not implemented
  - Re-evaluation
  - Preemption
  - Handling of differently sized slots (some with PSFCH, some without)
    - for transport block size calculation, a conservative assumption is made that all slots only have nine symbols available for PSSCH data

Typical sequence of simulation events



LteUeRlc

NrUeMac

NrUePhy

NrSlUeMacSchedulerDefault

1. Slot boundary occurs

2. Trigger Request

3. Determine and prioritize which LCs need grants

For each LC needing a grant...

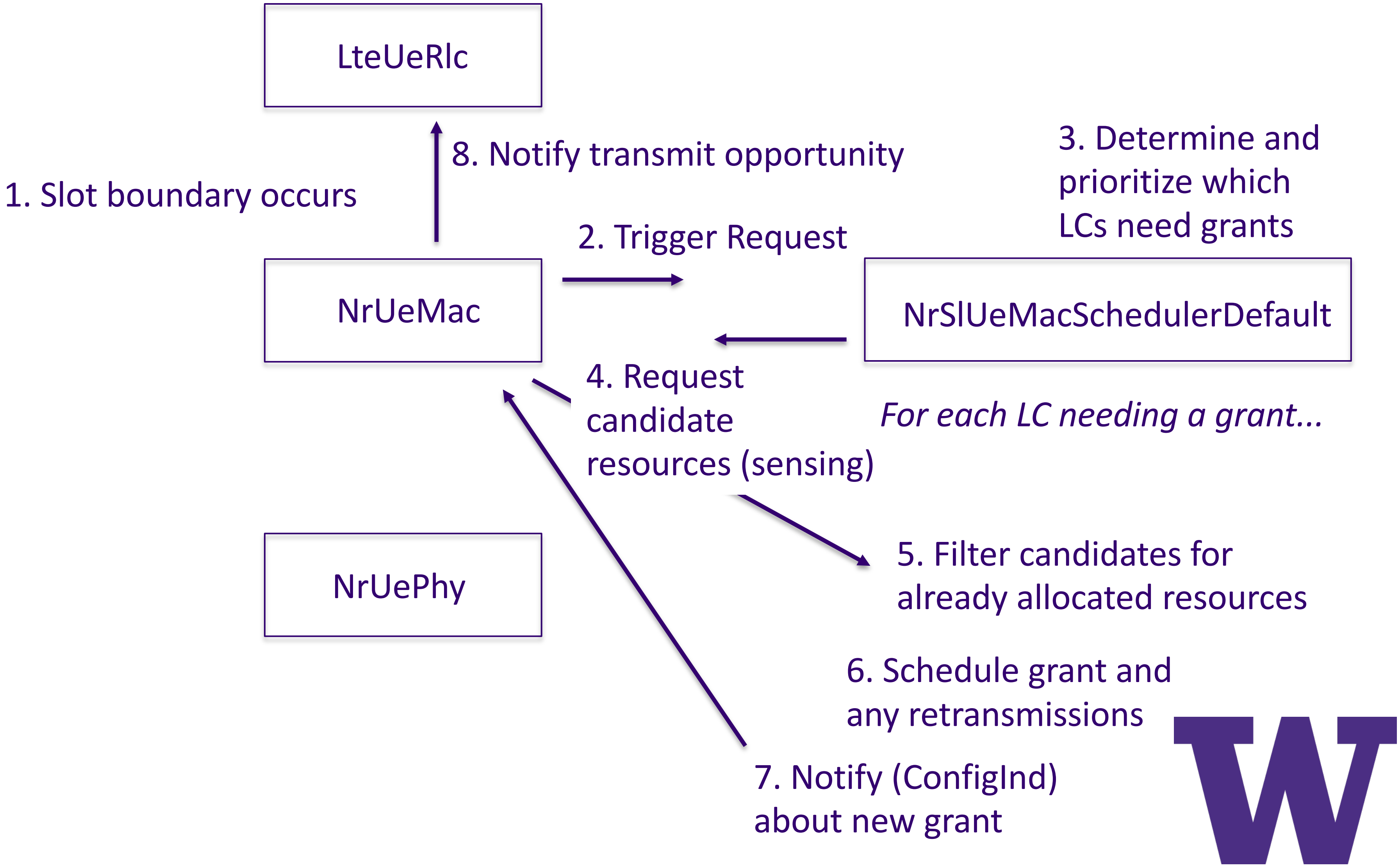
4. Request candidate resources (sensing)

5. Filter candidates for already allocated resources

6. Schedule grant and any retransmissions

7. Notify (ConfigInd) about new grant

Typical sequence of simulation events



LteUeRlc

NrUeMac

NrUePhy

NrSlUeMacSchedulerDefault

1. Slot boundary occurs

2. Trigger Request

3. Determine and prioritize which LCs need grants

4. Request candidate resources (sensing)

For each LC needing a grant...

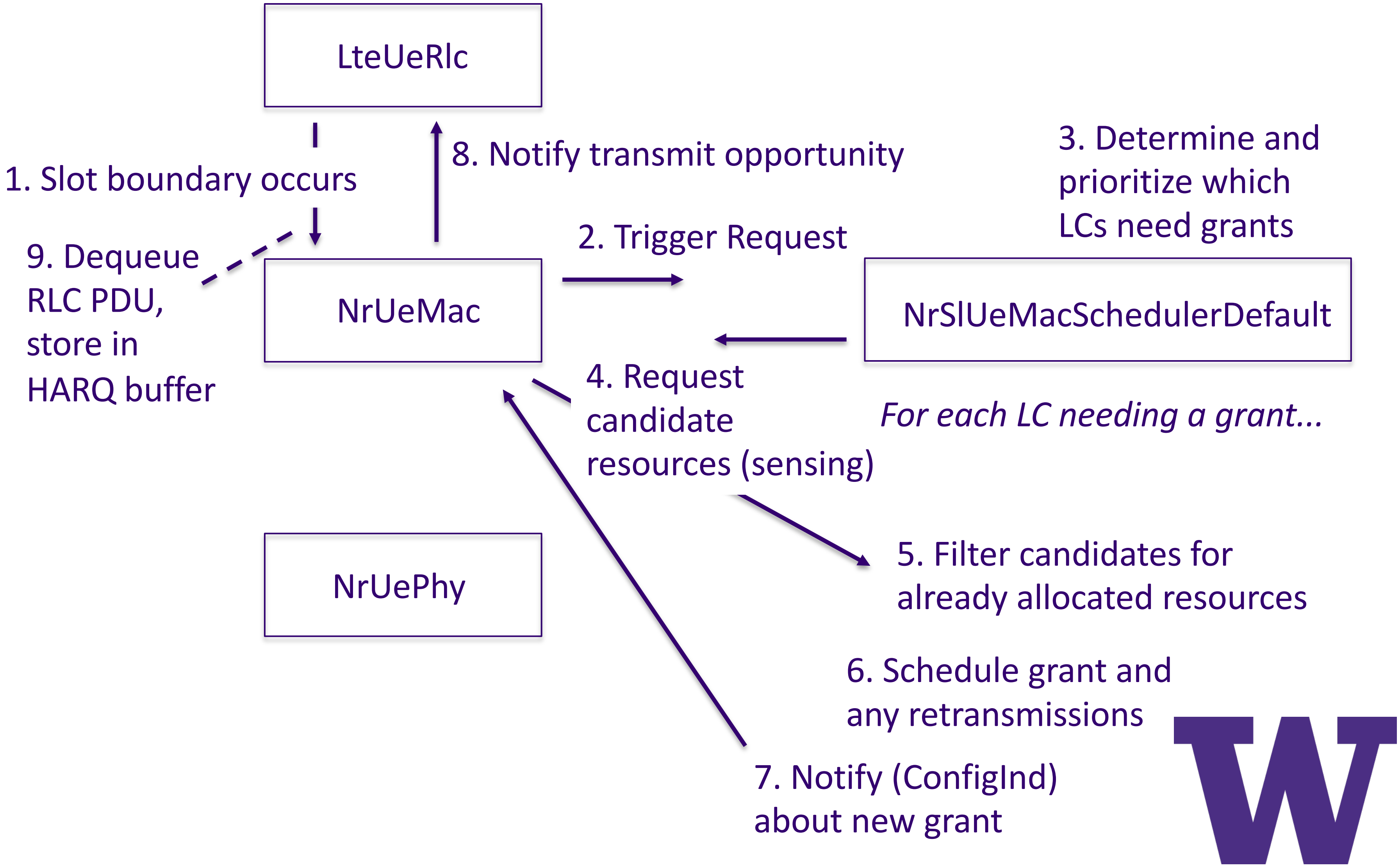
5. Filter candidates for already allocated resources

6. Schedule grant and any retransmissions

7. Notify (ConfigInd) about new grant

8. Notify transmit opportunity

Typical sequence of simulation events



LteUeRlc

NrUeMac

NrSlUeMacSchedulerDefault

NrUePhy

1. Slot boundary occurs

2. Trigger Request

3. Determine and prioritize which LCs need grants

For each LC needing a grant...

4. Request candidate resources (sensing)

5. Filter candidates for already allocated resources

6. Schedule grant and any retransmissions

7. Notify (ConfigInd) about new grant

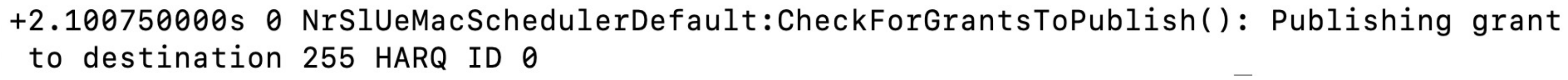
8. Notify transmit opportunity

9. Dequeue RLC PDU, store in HARQ buffer

Log output: Publishing grants

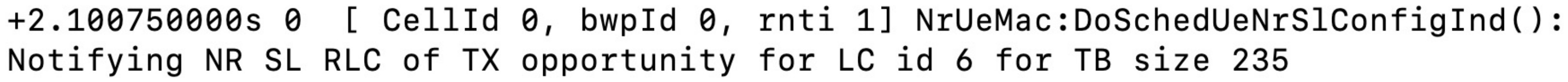
```
$
NS_LOG="NrSlUeMacSchedulerDefault=info|prefix_time|prefix_node|prefix_func:
NrUeMac=info|prefix_time|prefix_node|prefix_func" ./ns3 run nr-sl-simple-
multi-lc > log3.out 2>&1
```

7. Notify (ConfigInd) about new grant (line 33)



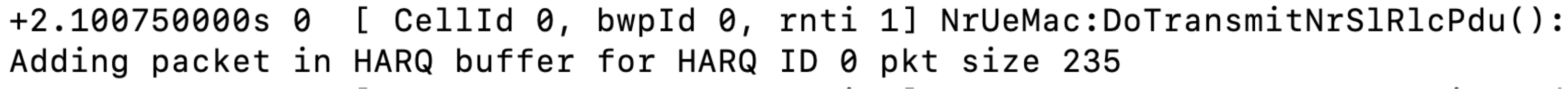
```
+2.100750000s 0 NrSlUeMacSchedulerDefault:CheckForGrantsToPublish(): Publishing grant
 to destination 255 HARQ ID 0
```

8. Notify transmit opportunity (line 34)



```
+2.100750000s 0  [ CellId 0, bwpId 0, rnti 1] NrUeMac:DoSchedUeNrSlConfigInd():
Notifying NR SL RLC of TX opportunity for LC id 6 for TB size 235
```

9. Dequeue from LC, store in HARQ buffer (line 35)

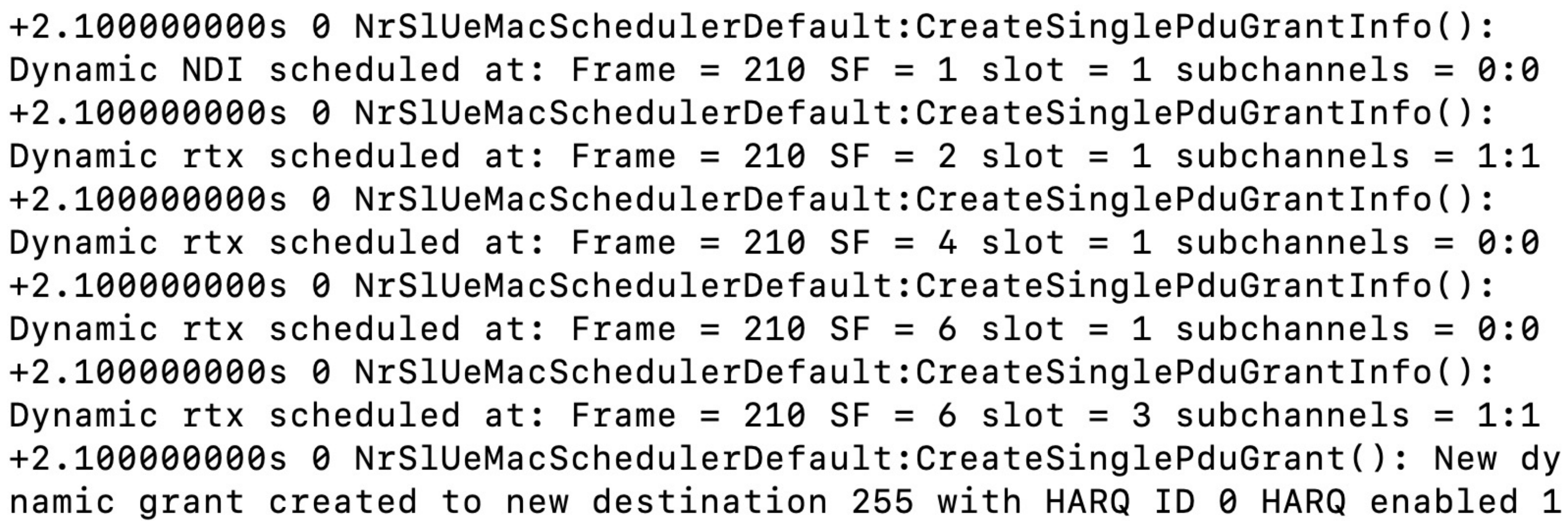


```
+2.100750000s 0  [ CellId 0, bwpId 0, rnti 1] NrUeMac:DoTransmitNrSlRlcPdu():
Adding packet in HARQ buffer for HARQ ID 0 pkt size 235
```

Log output: Publishing grants (cont.)

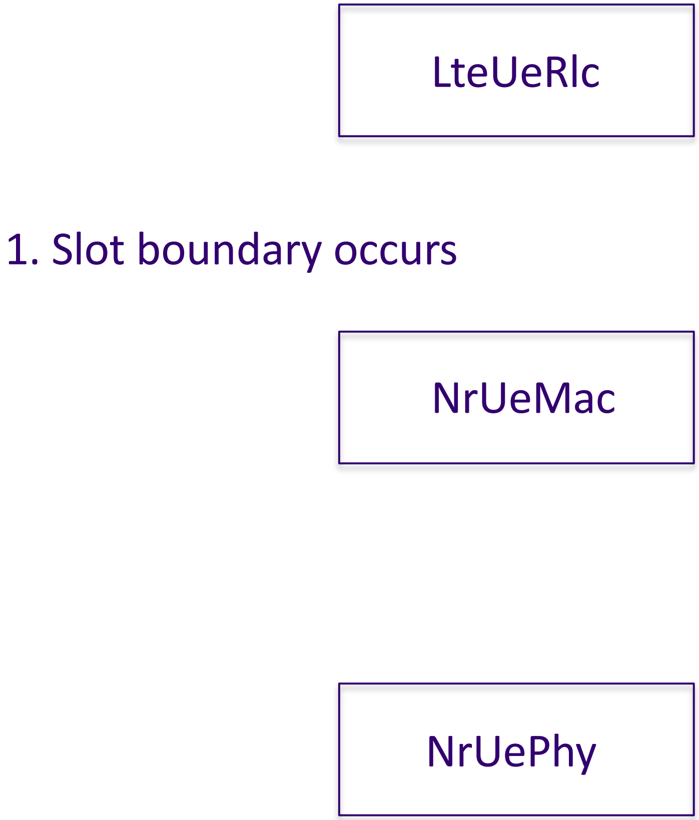
```
$
NS_LOG="NrSlUeMacSchedulerDefault=info|prefix_time|prefix_node|prefix_func:
NrUeMac=info|prefix_time|prefix_node|prefix_func" ./ns3 run nr-sl-simple-
multi-lc > log3.out 2>&1
```

7. Notify (ConfigInd) about new grant (more detail: lines 27-32)



```
+2.100000000s 0 NrSlUeMacSchedulerDefault:CreateSinglePduGrantInfo():
Dynamic NDI scheduled at: Frame = 210 SF = 1 slot = 1 subchannels = 0:0
+2.100000000s 0 NrSlUeMacSchedulerDefault:CreateSinglePduGrantInfo():
Dynamic rtx scheduled at: Frame = 210 SF = 2 slot = 1 subchannels = 1:1
+2.100000000s 0 NrSlUeMacSchedulerDefault:CreateSinglePduGrantInfo():
Dynamic rtx scheduled at: Frame = 210 SF = 4 slot = 1 subchannels = 0:0
+2.100000000s 0 NrSlUeMacSchedulerDefault:CreateSinglePduGrantInfo():
Dynamic rtx scheduled at: Frame = 210 SF = 6 slot = 1 subchannels = 0:0
+2.100000000s 0 NrSlUeMacSchedulerDefault:CreateSinglePduGrantInfo():
Dynamic rtx scheduled at: Frame = 210 SF = 6 slot = 3 subchannels = 1:1
+2.100000000s 0 NrSlUeMacSchedulerDefault:CreateSinglePduGrant(): New dy
namic grant created to new destination 255 with HARQ ID 0 HARQ enabled 1
```

Typical sequence of simulation events



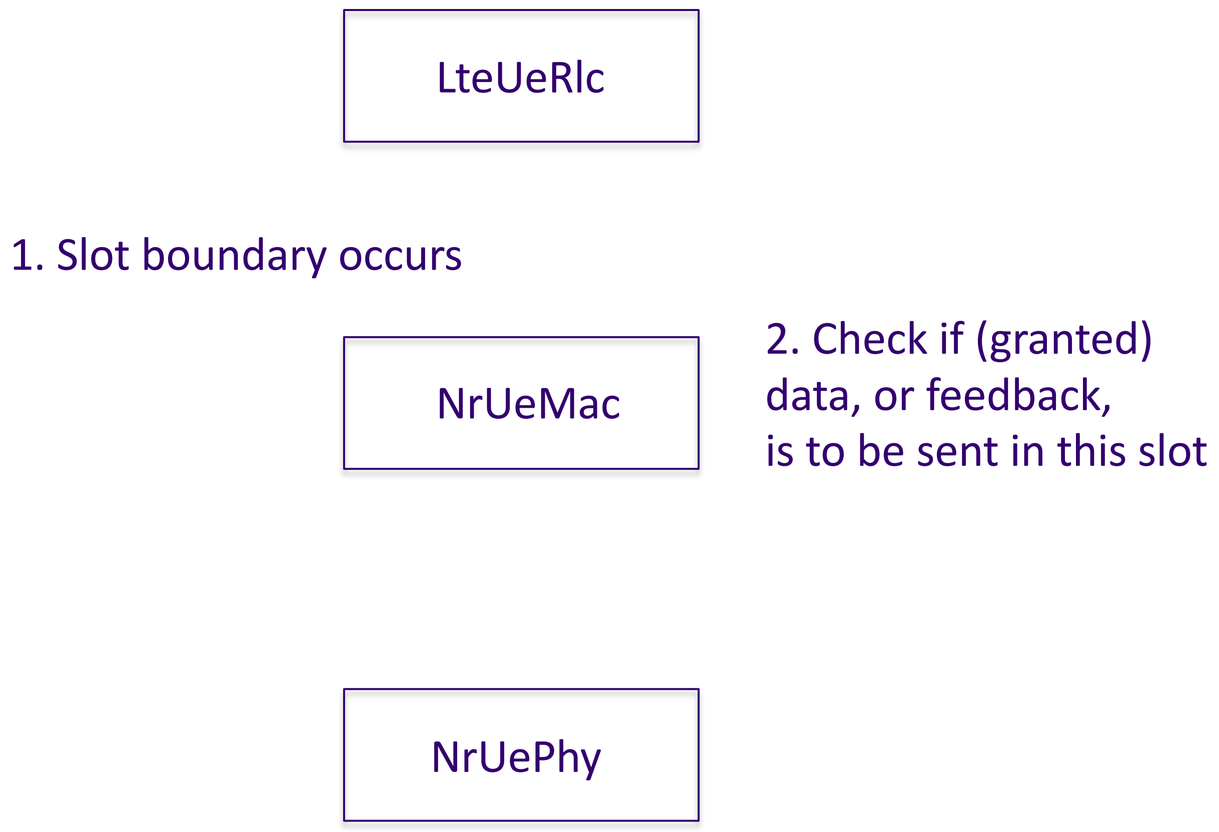
LteUeRlc

1. Slot boundary occurs

NrUeMac

NrUePhy

Typical sequence of simulation events



LteUeRlc

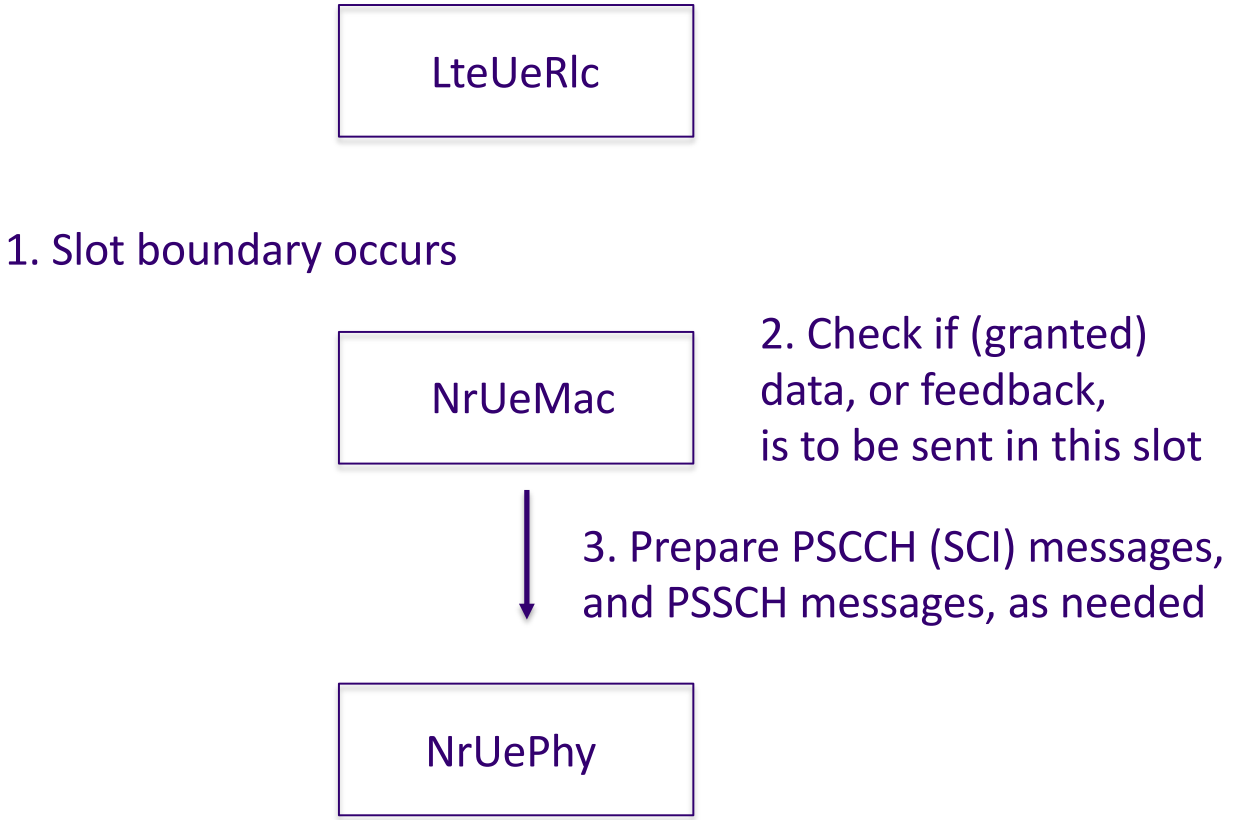
1. Slot boundary occurs

NrUeMac

2. Check if (granted)
data, or feedback,
is to be sent in this slot

NrUePhy

Typical sequence of simulation events



LteUeRlc

1. Slot boundary occurs

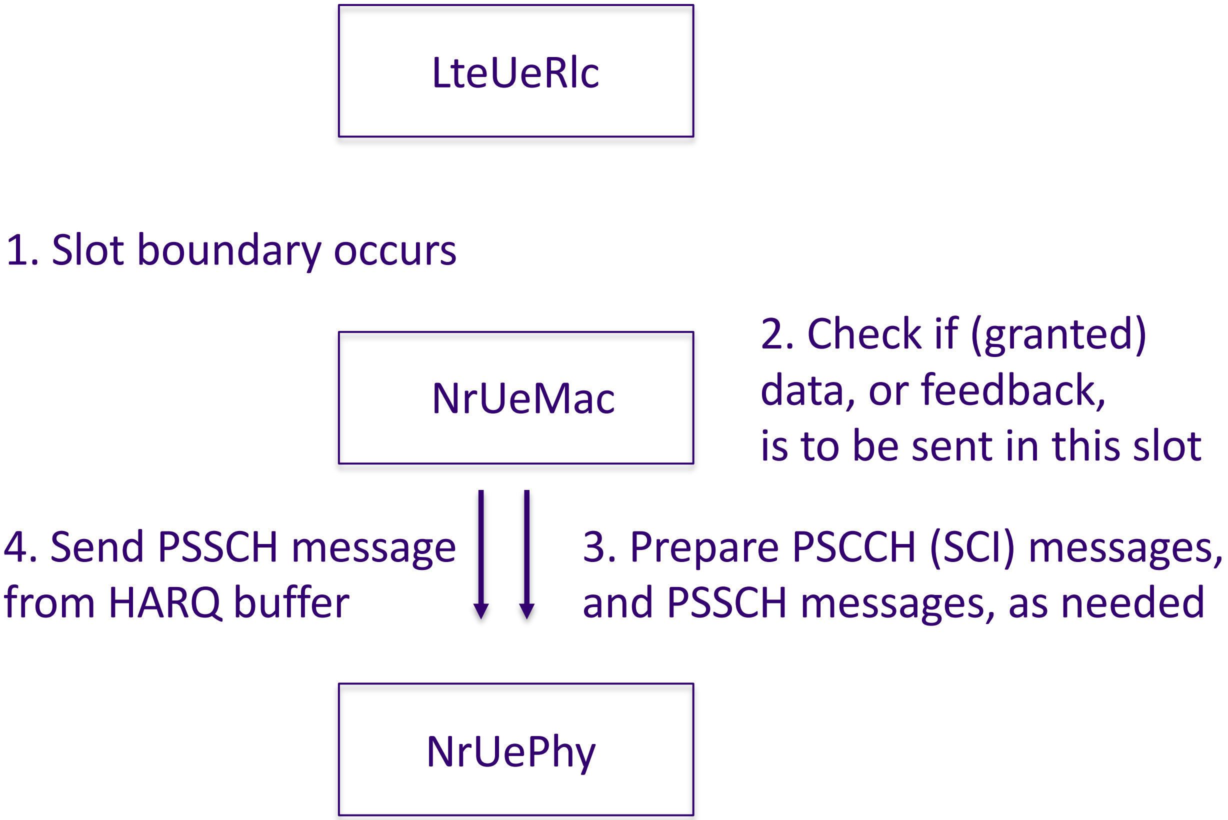
NrUeMac

2. Check if (granted)
data, or feedback,
is to be sent in this slot

3. Prepare PSCCH (SCI) messages,
and PSSCH messages, as needed

NrUePhy

Typical sequence of simulation events



LteUeRlc

1. Slot boundary occurs

NrUeMac

2. Check if (granted)
data, or feedback,
is to be sent in this slot

4. Send PSSCH message
from HARQ buffer

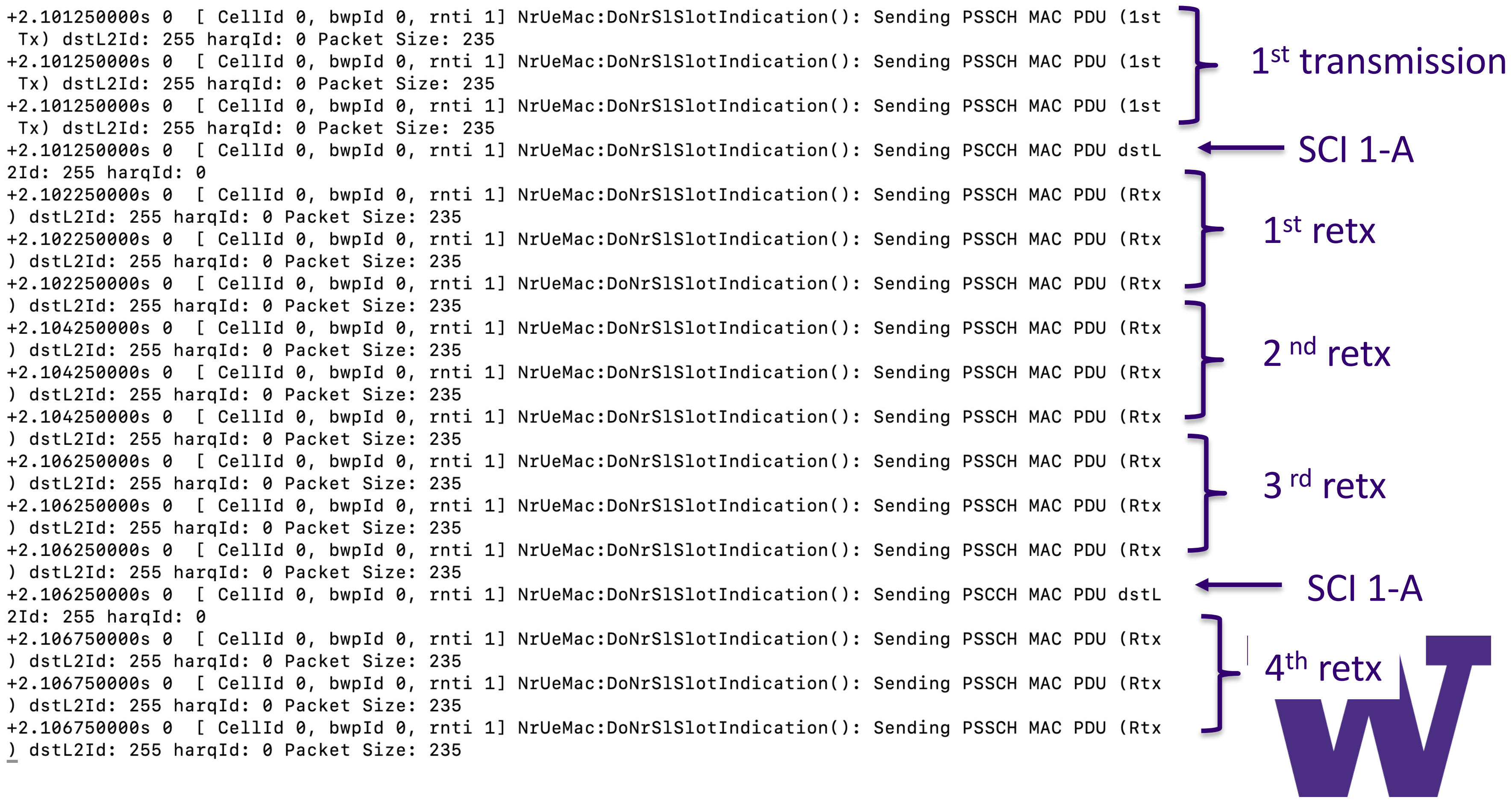
3. Prepare PSCCH (SCI) messages,
and PSSCH messages, as needed

NrUePhy

Log output: Sending data

```
$  NS_LOG="NrUeMac=info|prefix_time|prefix_node|prefix_func" ./ns3 run nr-
sl-simple-multi-lc > log4.out 2>&1
```

Only one event on previous slide (#3) captured by logging at INFO level
Sending of PSSCH and PSCCH is logged at this level (lines 16-32)



```
+2.101250000s 0  [ CellId 0, bwpId 0, rnti 1] NrUeMac:DoNrSlSlotIndication(): Sending PSSCH MAC PDU (1st
 Tx) dstL2Id: 255 harqId: 0 Packet Size: 235
+2.101250000s 0  [ CellId 0, bwpId 0, rnti 1] NrUeMac:DoNrSlSlotIndication(): Sending PSSCH MAC PDU (1st
 Tx) dstL2Id: 255 harqId: 0 Packet Size: 235
+2.101250000s 0  [ CellId 0, bwpId 0, rnti 1] NrUeMac:DoNrSlSlotIndication(): Sending PSSCH MAC PDU (1st
 Tx) dstL2Id: 255 harqId: 0 Packet Size: 235
+2.101250000s 0  [ CellId 0, bwpId 0, rnti 1] NrUeMac:DoNrSlSlotIndication(): Sending PSCCH MAC PDU dstL
2Id: 255 harqId: 0
+2.102250000s 0  [ CellId 0, bwpId 0, rnti 1] NrUeMac:DoNrSlSlotIndication(): Sending PSSCH MAC PDU (Rtx
) dstL2Id: 255 harqId: 0 Packet Size: 235
+2.102250000s 0  [ CellId 0, bwpId 0, rnti 1] NrUeMac:DoNrSlSlotIndication(): Sending PSSCH MAC PDU (Rtx
) dstL2Id: 255 harqId: 0 Packet Size: 235
+2.102250000s 0  [ CellId 0, bwpId 0, rnti 1] NrUeMac:DoNrSlSlotIndication(): Sending PSSCH MAC PDU (Rtx
) dstL2Id: 255 harqId: 0 Packet Size: 235
+2.104250000s 0  [ CellId 0, bwpId 0, rnti 1] NrUeMac:DoNrSlSlotIndication(): Sending PSSCH MAC PDU (Rtx
) dstL2Id: 255 harqId: 0 Packet Size: 235
+2.104250000s 0  [ CellId 0, bwpId 0, rnti 1] NrUeMac:DoNrSlSlotIndication(): Sending PSSCH MAC PDU (Rtx
) dstL2Id: 255 harqId: 0 Packet Size: 235
+2.104250000s 0  [ CellId 0, bwpId 0, rnti 1] NrUeMac:DoNrSlSlotIndication(): Sending PSSCH MAC PDU (Rtx
) dstL2Id: 255 harqId: 0 Packet Size: 235
+2.106250000s 0  [ CellId 0, bwpId 0, rnti 1] NrUeMac:DoNrSlSlotIndication(): Sending PSSCH MAC PDU (Rtx
) dstL2Id: 255 harqId: 0 Packet Size: 235
+2.106250000s 0  [ CellId 0, bwpId 0, rnti 1] NrUeMac:DoNrSlSlotIndication(): Sending PSSCH MAC PDU (Rtx
) dstL2Id: 255 harqId: 0 Packet Size: 235
+2.106250000s 0  [ CellId 0, bwpId 0, rnti 1] NrUeMac:DoNrSlSlotIndication(): Sending PSSCH MAC PDU (Rtx
) dstL2Id: 255 harqId: 0 Packet Size: 235
+2.106250000s 0  [ CellId 0, bwpId 0, rnti 1] NrUeMac:DoNrSlSlotIndication(): Sending PSCCH MAC PDU dstL
2Id: 255 harqId: 0
+2.106750000s 0  [ CellId 0, bwpId 0, rnti 1] NrUeMac:DoNrSlSlotIndication(): Sending PSSCH MAC PDU (Rtx
) dstL2Id: 255 harqId: 0 Packet Size: 235
+2.106750000s 0  [ CellId 0, bwpId 0, rnti 1] NrUeMac:DoNrSlSlotIndication(): Sending PSSCH MAC PDU (Rtx
) dstL2Id: 255 harqId: 0 Packet Size: 235
+2.106750000s 0  [ CellId 0, bwpId 0, rnti 1] NrUeMac:DoNrSlSlotIndication(): Sending PSSCH MAC PDU (Rtx
) dstL2Id: 255 harqId: 0 Packet Size: 235
```

HARQ overview

- Hybrid Automatic Repeat Request (HARQ) is provided in sidelink for unicast and groupcast
- ACK or NACK feedback provided from receiver to transmitter
- Two modes of groupcast HARQ are defined
  - Range-based NACK-only feedback (mode 1)
  - ACK/NACK feedback (mode 2)

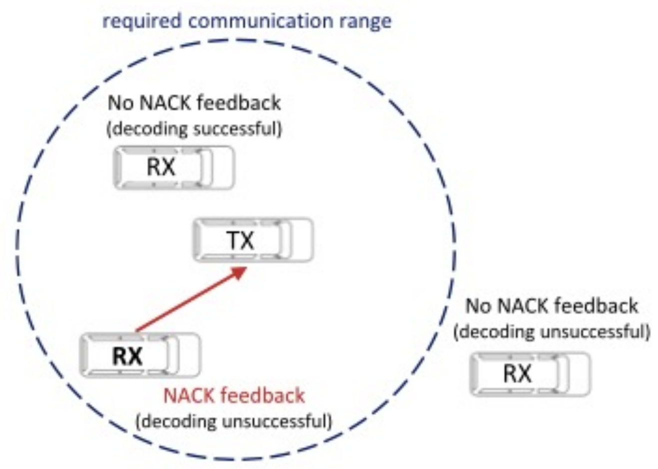


Fig. 9. NACK-only feedback for groupcast NR V2X sidelink (option 1).

In ns-3, only mode 2 is implemented

Figure source: A Tutorial on 5G NR V2X Communications, Garcia et al.

PSFCH channel

- The feedback channel is a symbol that is periodically inserted every PsfchPeriod (1, 2, or 4) sidelink slots
- Feedback is delivered on the next available PSFCH symbol after the 'MinTimeGapPsfch' slots have occurred

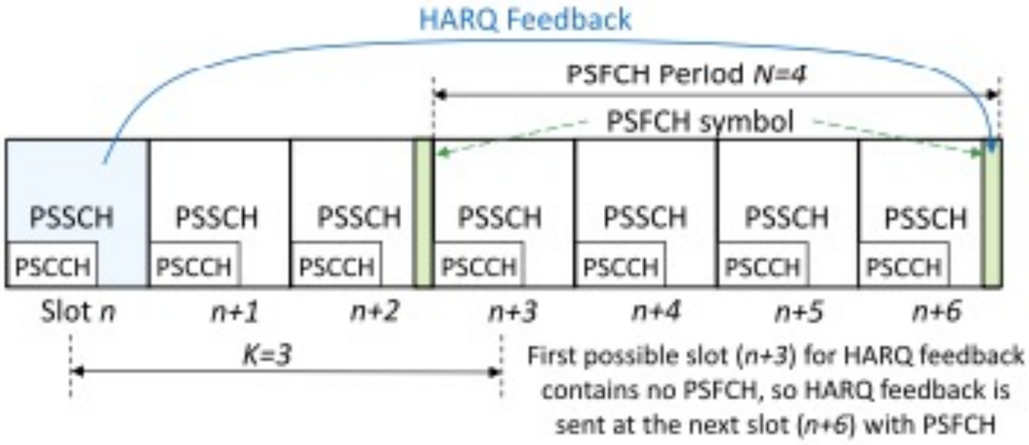

HARQ Feedback
PSFCH Period N=4
PSFCH symbol
PSSCH
PSCCH
Slot n
n+1
n+2
n+3
n+4
n+5
n+6
K=3
First possible slot (n+3) for HARQ feedback contains no PSFCH, so HARQ feedback is sent at the next slot (n+6) with PSFCH

Fig. 10. PSSCH-to-HARQ feedback timing based on at least K = 3 slots. For simplicity, the figure depicts only one sub-channel within the resource pool. We also omit the detailed structure of a PSCCH/PSSCH slot with or without PSFCH including PSSCH DMRS, AGC symbols and guard symbols.

Figure sources: A Tutorial on 5G NR V2X Communications, Garcia et al.

PSFCH encoding

- In practice, a complicated encoding exists to convey the feedback for a set of PRBs into the limited PSFCH symbol
    - In ns-3, we currently assume a perfect PSFCH channel

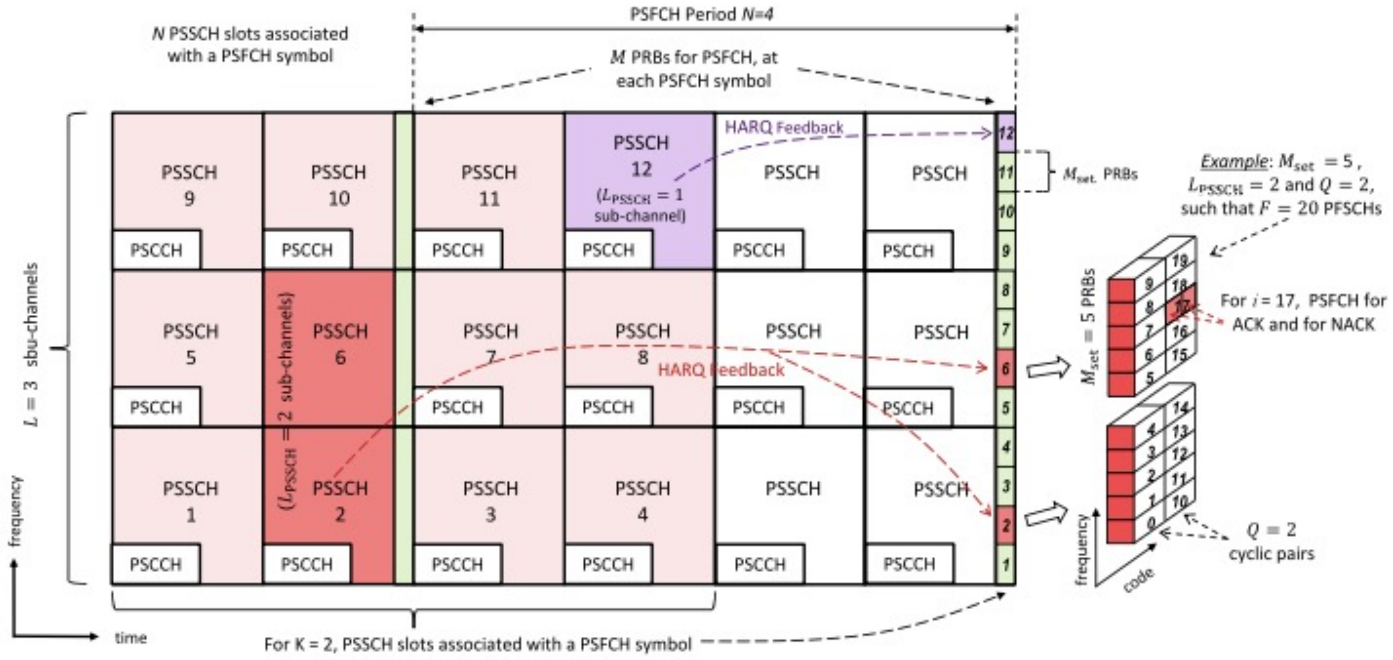
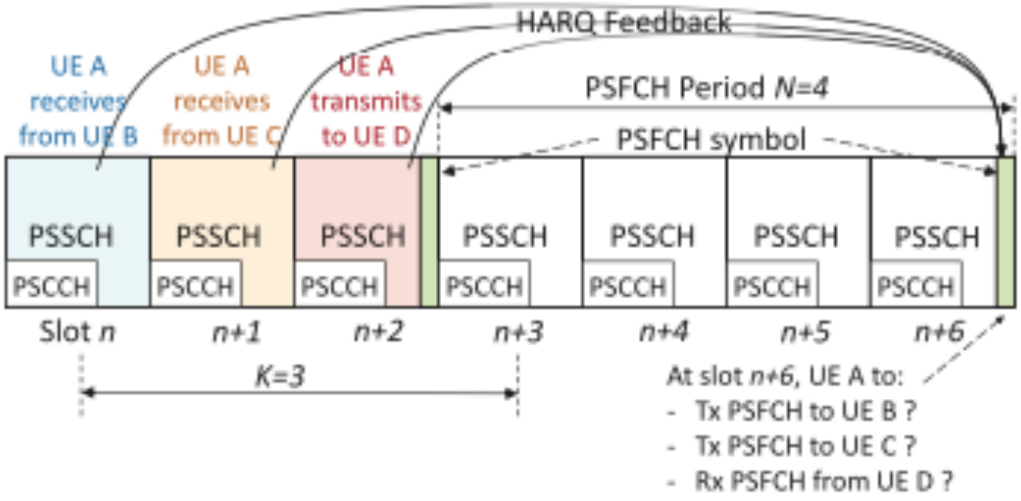

PSFCH Period N=4
N PSSCH slots associated with a PSFCH symbol
M PRBs for PSFCH, at each PSFCH symbol
HARQ Feedback
PSSCH 9
PSSCH 10
PSSCH 11
PSSCH 12 (L_{PSSCH} = 1 sub-channel)
PSSCH 5
PSSCH 6
PSSCH 7
PSSCH 8
PSSCH 1
PSSCH 2
PSSCH 3
PSSCH 4
PSSCH
PSCCH
(L_{PSSCH} = 2 sub-channels)
HARQ Feedback
L = 3 sbu-channels
frequency
time
For K = 2, PSSCH slots associated with a PSFCH symbol
12
11
10
9
8
7
6
5
4
3
2
1
M_{set} PRBs
M_{set} = 5 PRBs
Example: M_{set} = 5, L_{PSSCH} = 2 and Q = 2, such that F = 20 PFSCHs
For i = 17, PSFCH for ACK and for NACK
Q = 2 cyclic pairs
frequency
code

Fig. 11. PSFCHs for HARQ feedback associated with different transmissions.

Figure sources: A Tutorial on 5G NR V2X Communications, Garcia et al.

PSFCH contention

- UEs may have multiple transmissions to make in the same PSFCH symbol, or may want to transmit ***and*** receive in the same symbol
  - In ns-3, we do not model PSFCH collisions or contention



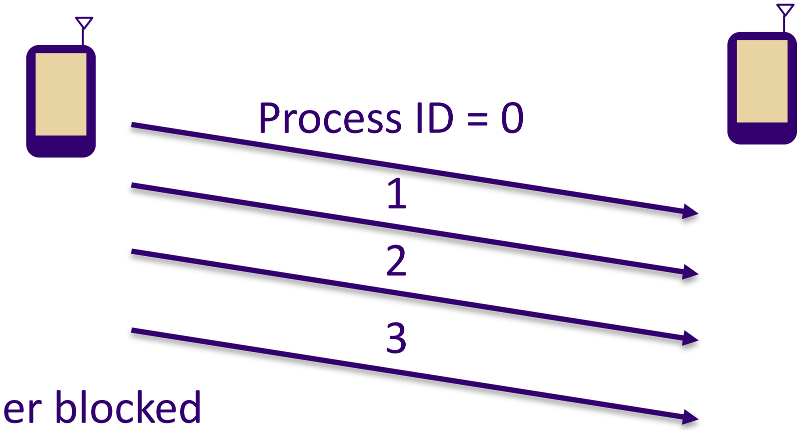
At slot n+6, UE A to:
- Tx PSFCH to UE B ?
- Tx PSFCH to UE C ?
- Rx PSFCH from UE D ?

Fig. 12. Collisions between PSFCH transmissions or between a PSFCH transmission and a PSFCH reception.

Figure sources: A Tutorial on 5G NR V2X Communications, Garcia et al.

HARQ processes

- HARQ is a ”stop-and-wait” protocol, using a small number of “process IDs”
- Each process ID corresponds to a transport block (TB)
- A MAC instance has a maximum of four SL HARQ processes

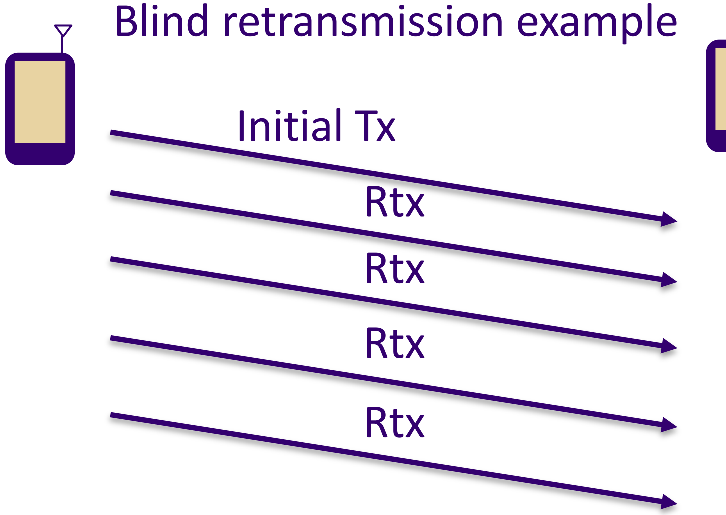


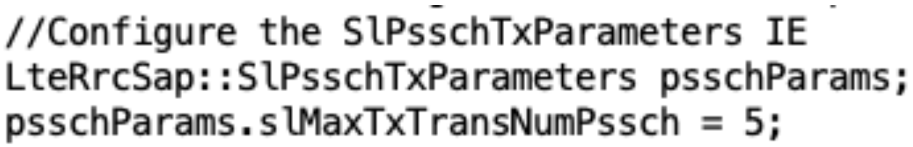
Sender blocked until one of the HARQ processes is ACKed or times out

Number of HARQ (and blind) retransmissions

- A logical channel may be configured from between 1 and 32 transmissions per TB. (ns-3 default is 5 transmissions)
- In ns-3, blind retransmissions can be configured by enabling HARQ in the SidelinkInfo parameter, and setting PSFCH period to zero (disabling PSFCH)
  - Blind retransmission configuration is not discussed in the standards
- Blind retransmissions are selected at random within the selection window

Blind retransmission example





```
//Configure the SlPsschTxParameters IE
LteRrcSap::SlPsschTxParameters psschParams;
psschParams.slMaxTxTransNumPssch = 5;
```

No feedback is returned

HARQ scheduling implications

> When HARQ is enabled, TS 38.321 states that retransmission slots must be selected such that there is an opportunity for PSFCH feedback to be returned before retransmission
  - This is implemented in ns-3 in the scheduler (`NrSlUeMacSchedulerDefault::IsMinTimeGapSatisfied()`)
  - This may limit the number of retransmissions available in a selection window

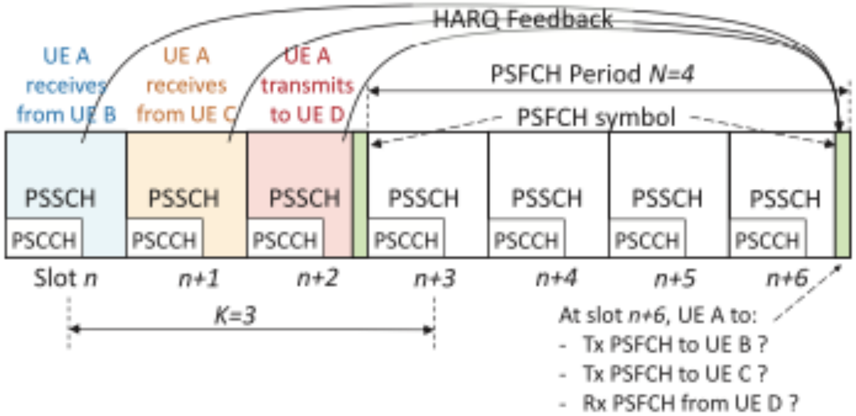
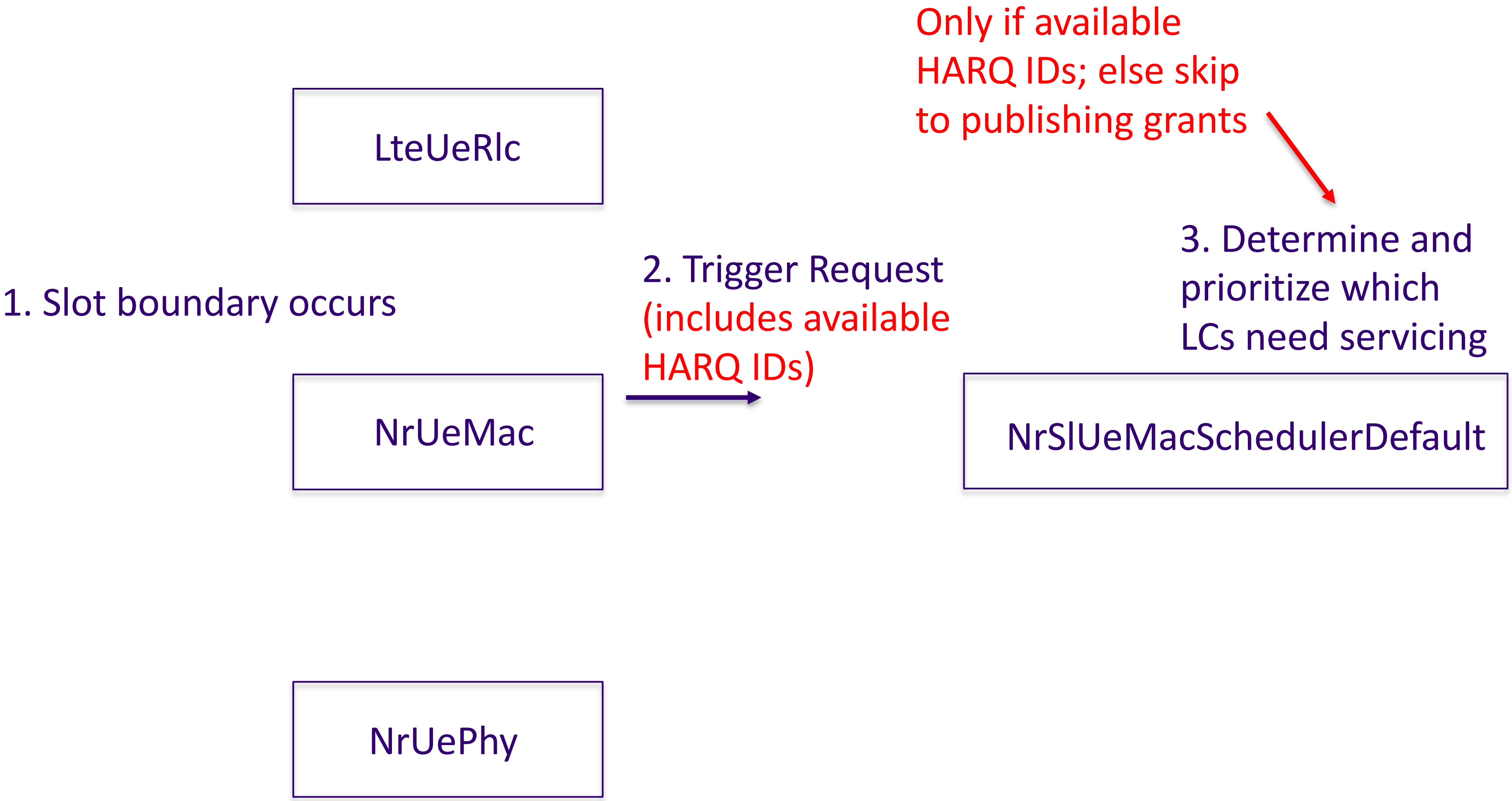


Fig. 12. Collisions between PSFCH transmissions or between a PSFCH transmission and a PSFCH reception.

Figure sources: A Tutorial on 5G NR V2X Communications, Garcia et al.

Typical sequence of simulation events



Only if available HARQ IDs; else skip to publishing grants

LteUeRlc

1. Slot boundary occurs

NrUeMac

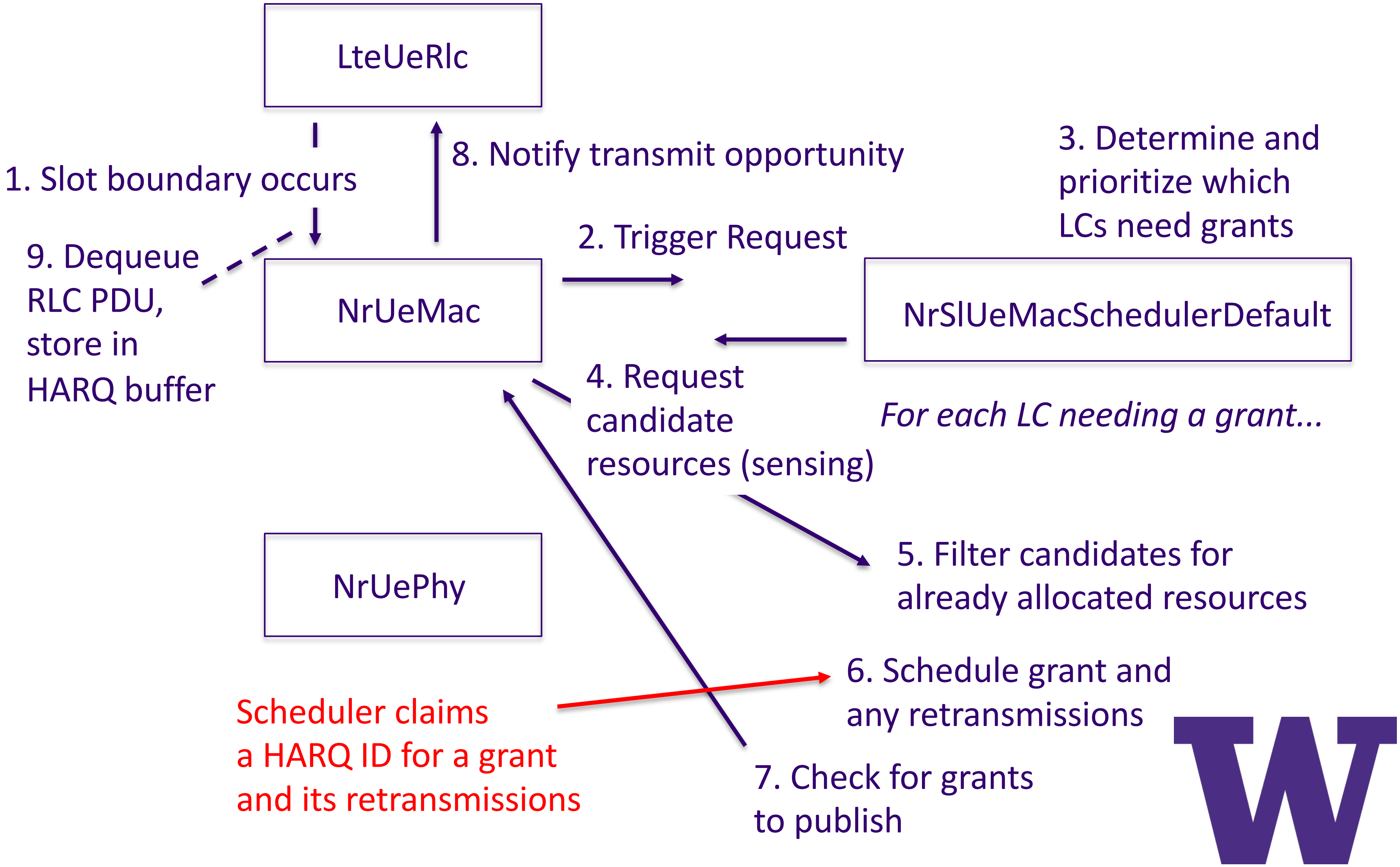
2. Trigger Request (includes available HARQ IDs)

3. Determine and prioritize which LCs need servicing

NrSlUeMacSchedulerDefault

NrUePhy

Typical sequence of simulation events



LteUeRlc

NrUeMac

NrSlUeMacSchedulerDefault

NrUePhy

1. Slot boundary occurs

2. Trigger Request

3. Determine and prioritize which LCs need grants

For each LC needing a grant…

4. Request candidate resources (sensing)

5. Filter candidates for already allocated resources

6. Schedule grant and any retransmissions

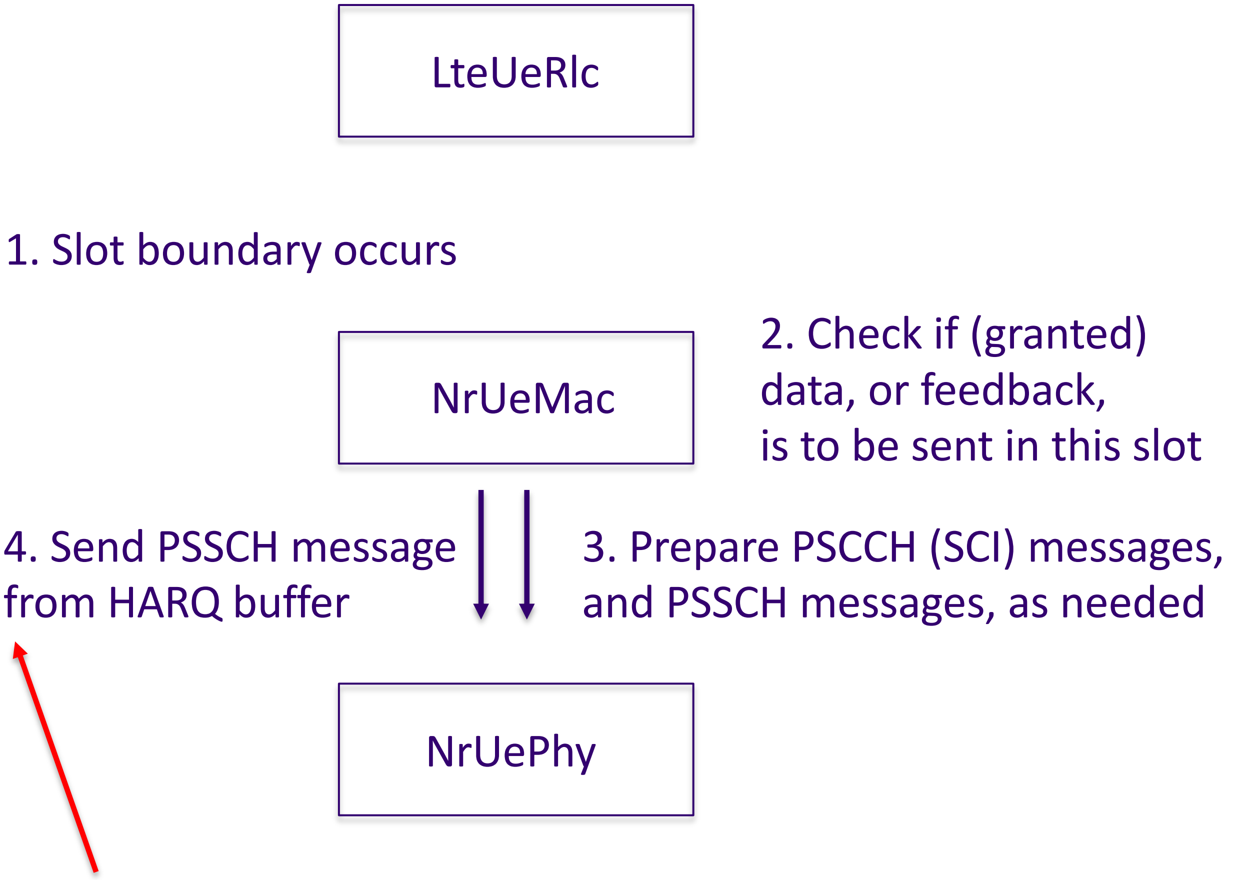
Scheduler claims a HARQ ID for a grant and its retransmissions

7. Check for grants to publish

8. Notify transmit opportunity

9. Dequeue RLC PDU, store in HARQ buffer

Typical sequence of simulation events



LteUeRlc

1. Slot boundary occurs

NrUeMac

2. Check if (granted) data, or feedback, is to be sent in this slot

3. Prepare PSCCH (SCI) messages, and PSSCH messages, as needed

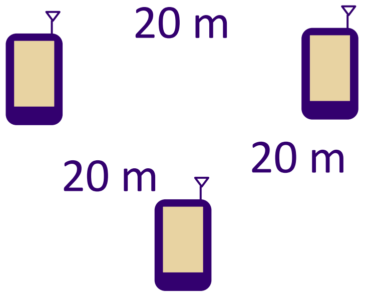
4. Send PSSCH message from HARQ buffer

NrUePhy

- Start a timer to free the process ID in case of lack of response
- In case of positive ACK, cancel future retransmissions and timer (and free the process ID)

sidelink-harq-example.cc overview

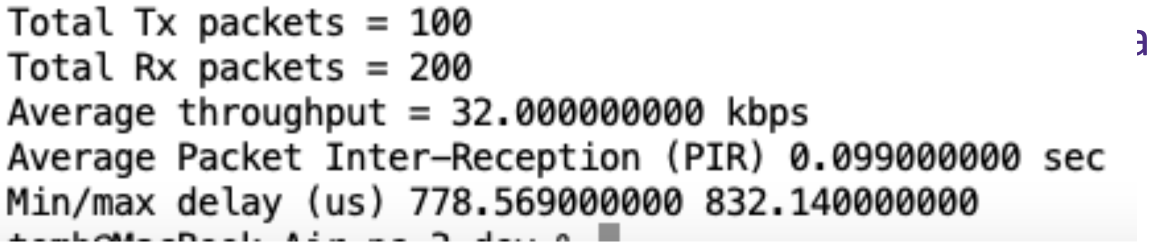
- Purpose: Demonstrate/contrast how groupcast mode 2, unicast, and broadcast (blind retransmission) can be configured in a small network
- Topology: Two (unicast) or three (groupcast) UEs separated by 20 meters



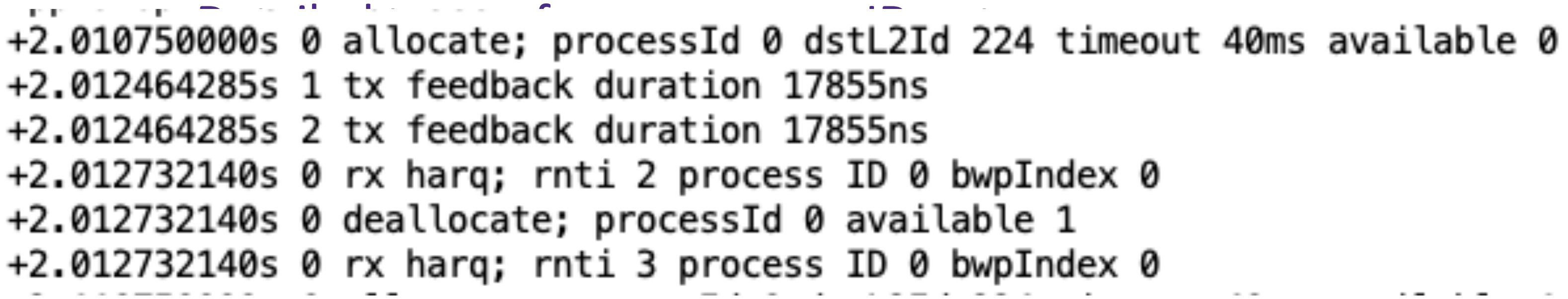
- One constant bit rate traffic (default 16 Kb/s, 200 byte UDP packets)
- Simulation runs for configured number of transmit packets (default 100)

sidelink-harq-example.cc overview (cont)

- Output: Summary output and detailed terminal output
  - Summary output: Packets sent and received, average throughput, TR and max/min delay



```
Total Tx packets = 100
Total Rx packets = 200
Average throughput = 32.000000000 kbps
Average Packet Inter-Reception (PIR) 0.099000000 sec
Min/max delay (us) 778.569000000 832.140000000
```

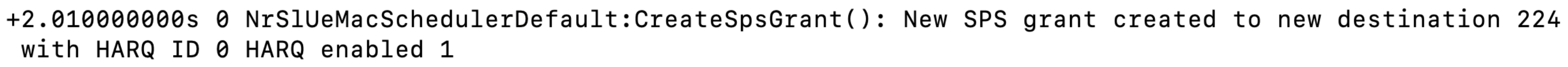


```
+2.010750000s 0 allocate; processId 0 dstL2Id 224 timeout 40ms available 0
+2.012464285s 1 tx feedback duration 17855ns
+2.012464285s 2 tx feedback duration 17855ns
+2.012732140s 0 rx harq; rnti 2 process ID 0 bwpIndex 0
+2.012732140s 0 deallocate; processId 0 available 1
+2.012732140s 0 rx harq; rnti 3 process ID 0 bwpIndex 0
```

Log output: HARQ events

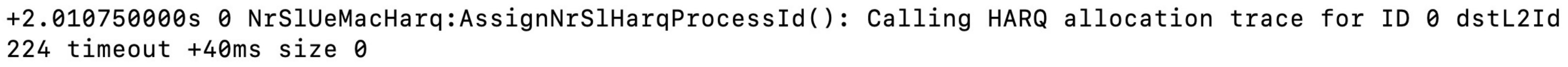
```
$
NS_LOG="NrSlUeMacSchedulerDefault=info|prefix_time|prefix_node|prefix_func:
NrSlUeMacHarq=info|prefix_time|prefix_node|prefix_func" ./ns3 run sidelink-
harq-example > log5.out 2>&1
```

Scheduler claims an unused HARQ ID (line 192)



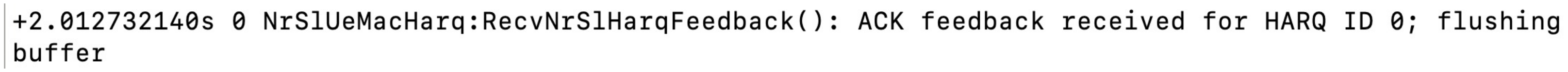
```
+2.010000000s 0 NrSlUeMacSchedulerDefault:CreateSpsGrant(): New SPS grant created to new destination 224
 with HARQ ID 0 HARQ enabled 1
```

HARQ process manager later assigns the ID and starts timer of 40 ms (line 194)



```
+2.010750000s 0 NrSlUeMacHarq:AssignNrSlHarqProcessId(): Calling HARQ allocation trace for ID 0 dstL2Id
224 timeout +40ms size 0
```

Positive ACK causes HARQ buffer to be flushed (line 199)



```
+2.012732140s 0 NrSlUeMacHarq:RecvNrSlHarqFeedback(): ACK feedback received for HARQ ID 0; flushing
buffer
```

Summary

- CTTC’s NR V2X branches have been extended for improved sensing, scheduling, and HARQ model operation
- Some level of abstraction still exists (e.g., PSFCH perfect feedback channel), but sensing, scheduling, and HARQ improvements should support a more accurate latency model for sidelink
- Updates reviewed in this tutorial will be upstreamed to CTTC’s public branches in the next month or two
- For more information, follow the CTTC-LENA nr branch (and merge requests) evolve this summer
  - User mailing list: 5g-lena-users@googlegroups.com
  - Merge requests: https://gitlab.com/cttc-lena/nr/-/merge_requests