NR C-V2X Mode 2 Resource Allocation & ns-3 Implementation

By: Collin Brady, Liu Cao, Sumit Roy

ns-3 Annual Meeting
June 26, 2023

UNIVERSITY *of* WASHINGTON

Overview

1. C-V2X Standards Introduction
2. Simple Scenario (MAC Collision Model)
    1. ns-3 Implementation Overview
    2. Remaining Issues
    3. Model of PRR for Simple Scenario
    4. Results for Simple Example Scenario
3. Mobile Scenario
    1. Results for Mobile Scenario

Section 1:
C-V2X Standards Introduction

- Main 3GPP standards: 38.101-1/38.101-2 (UE tx/rx), 38.204 (UE idle mode), 38.211 (modulation), 38.212 (multiplexing and coding), 38.213/38.214 (CTRL/data PHY), 38.215 (PHY measurements), 38.321 (MAC), 38.331 (RRC), 37.885 (KPIs)

Section 1: C-V2X Standards Introduction: Deployment Scenario

- Vehicular Networks
- Each vehicle broadcasts Periodic messages
  - Status messages
    - Network updates to deal with changes to network topology over time
  - Safety messages
    - Emergency braking, collision warning, etc.
- **Distributed resource allocation** for channel access via **Semi Persistent Scheduling** (SPS)
- Key Performance Indicators: Packet Reception Rate (PRR)/Throughput (reliability), Packet Inter-Reception Time (PIR) – TR 37.885
- Channel model: 3GPP V2V Highway – TR 37.885
- Receiver assumptions:
  - Half Duplex, can't tx/rx simultaneously
  - Decoding methodology left up to device manufacturer

Section 1: C-V2X Standards Introduction: NR C-V2X Use Cases

Use Case	Payload (Bytes)	Tx Rate (messages/s)	Max end-to-end Latency (ms)	Reliability (%)	Data Rate (Mbps)	Required Communications Range (m)
Vehicle Platooning	50-6000	2-5	10-25	>90	≤65	80-350
Advanced Driving	SL: 300-12000 UL: 450	SL: 10-100 UL: 50	10-100	>90	SL: 10-50 UL: 0.25-10 DL: 50	360-700
Extended Sensors	1600	10	3-100	>90	10-1000	50-1000
Remote Driving	16000-41700	33-200	5	99.999	UL: 25 DL: 1	≥1000

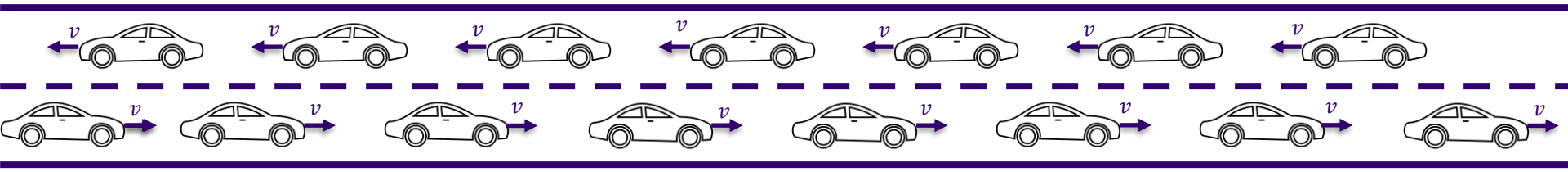
M. H. C. Garcia *et al*., "A Tutorial on 5G NR V2X Communications," in *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1972-2026, third quarter 2021

Glossary

Symbol	Definition
ρ_{UE}	UE density, UE/m
d_{TR}	Transmission range, beyond which UE can't decode transmitted packets
p_K	Resource Keeping Probability
R_c	Reselection Counter, number of times that a UE reserves a selected PRB
T_{RRI}	Duration of Resource Reservation Interval (RRI)
N_{sc}	Number of sub-channels
t_s	Slot duration
N_r	Total number of PRBs per RRI
N_{Se}	Number of duplicate packets a UE transmits per RRI
γ_{SPS}	SPS threshold, used to determine if a PRB is excluded during SPS reselection, dBm
N_{UE}	Total number of UEs
P_{COL}	MAC Collision Probability
P_{HD}	The probability that Half-Duplex error happens
PRR	Packet Reception Ratio
N_a	Number of available (unoccupied) PRBs in the selection window
N_o	Number of occupied PRBs in the selection window
N_c	Number of collided packets in a PRB where the collision happens
d_V	Inter-vehicle distance

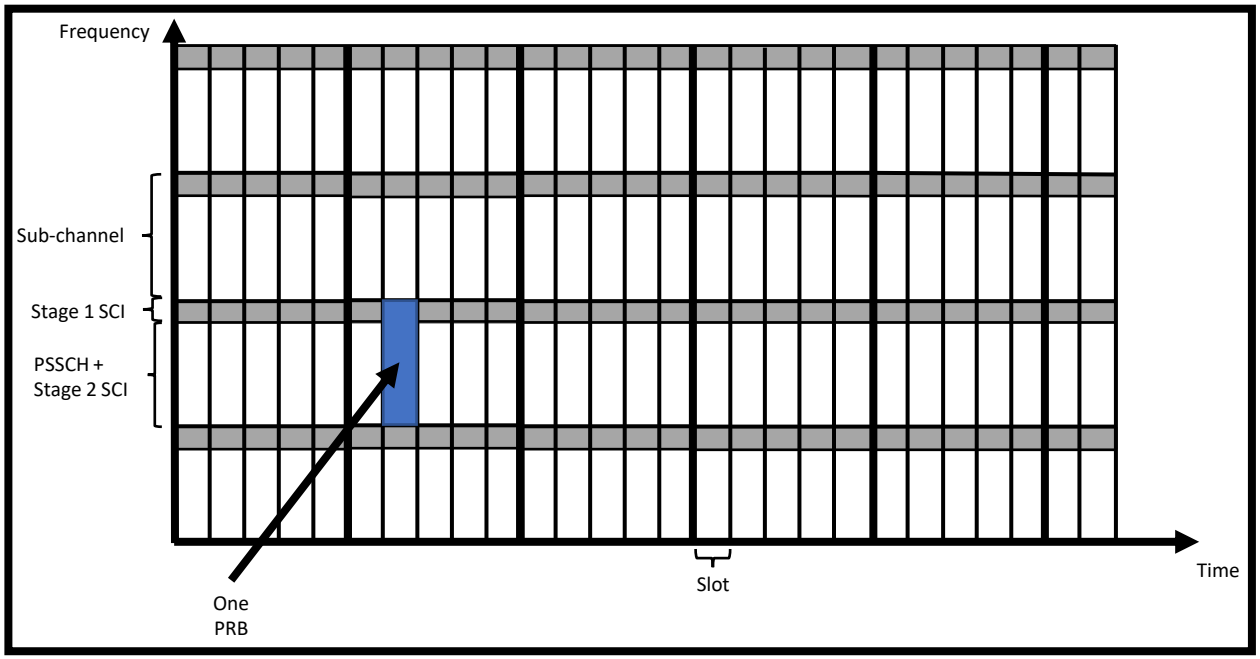
Note: One PRB supports one packet transmission.

Section 1: C-V2X Standards Introduction: Vehicular Network: Example Scenario



- User Equipment (UEs) distributed uniformly in a line with density ρ_{UE} UE/m
- 2 lanes
  - UEs move at velocity v in opposite directions
- MAC collision model
  - If two UEs occupy the same PRB, neither can be decoded
- UEs send packets periodically
  - Period = T_{RRI}
  - N_{Se} duplicate packets per period

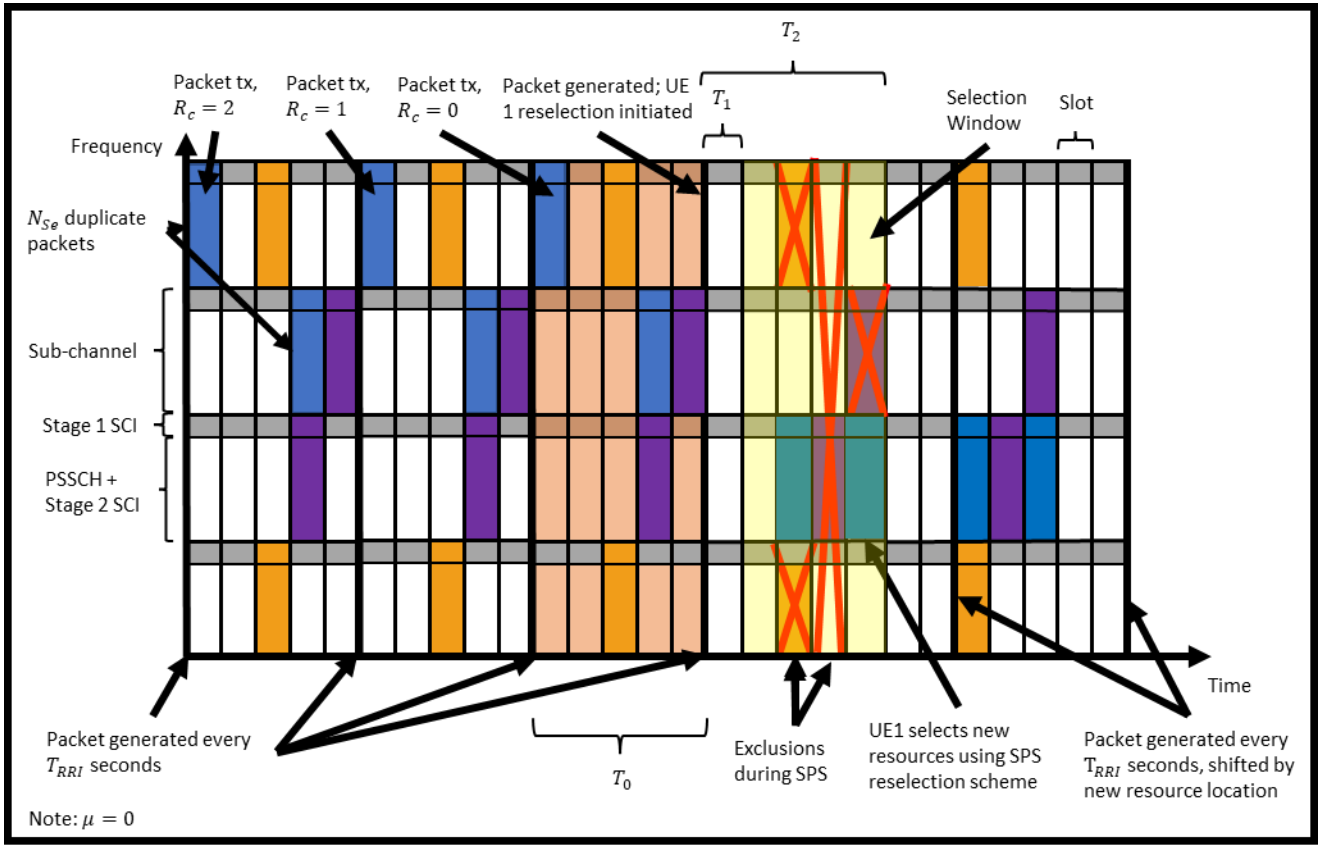
Section 1: C-V2X Standards Introduction: C-V2X Channelization



- **Channels:**
  - **Physical Sidelink Shared Channel (PSSCH)**
  - **Physical Sidelink Control Channel (PSCCH)**
    - **Split into two parts, Sidelink control information (SCI) stage 1 and 2**
  - **Physical Sidelink Feedback Channel (PSFCH)**
- **Transmission Modes:**
  - **Unicast, Groupcast, Broadcast**
  - **Focus of this presentation: Broadcast**
  - **PSFCH only exists in unicast/groupcast**

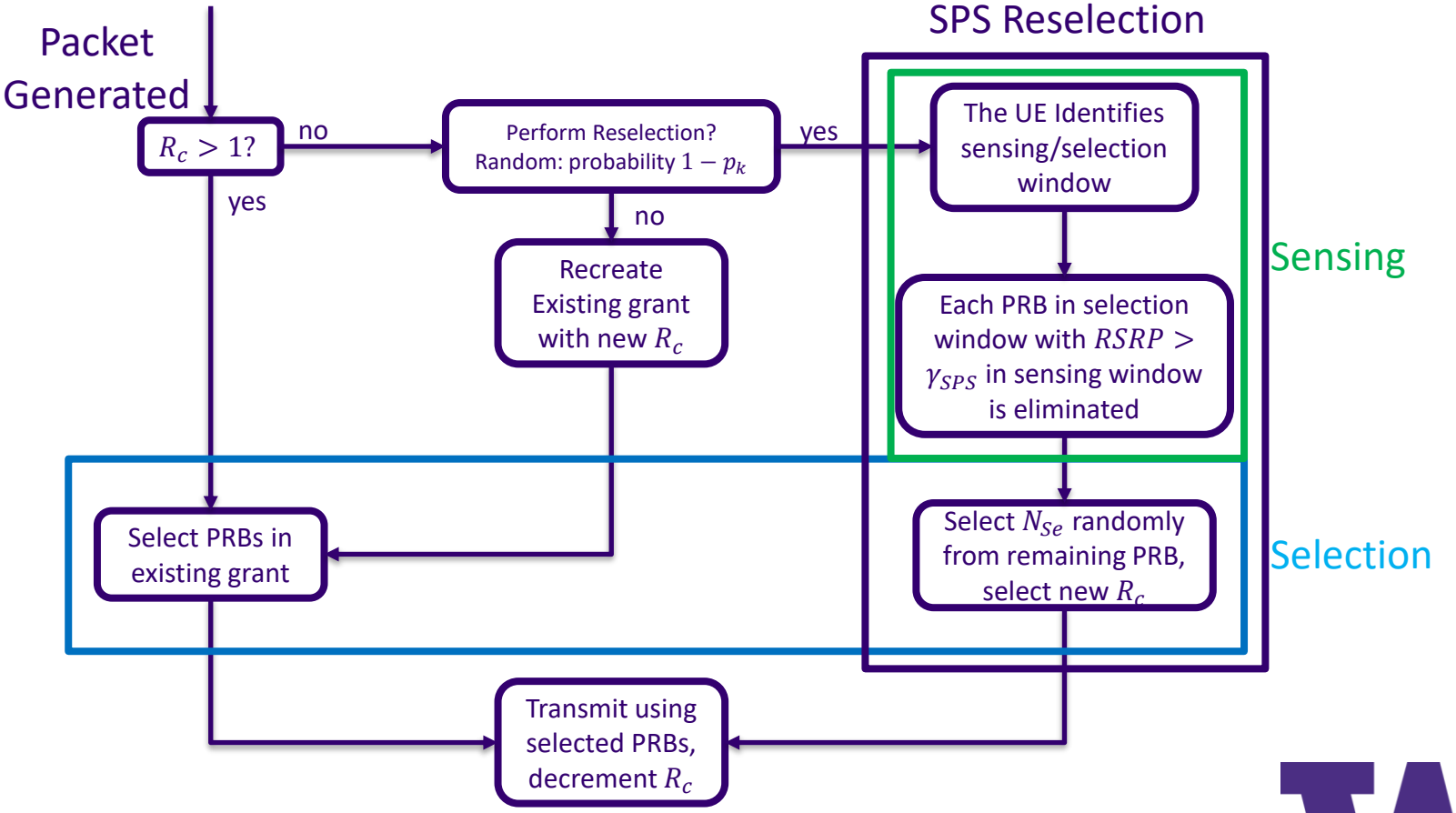
Section 1: C-V2X Standards Introduction: C-V2X MAC

- Each UE accesses the channel distributed
  - Can cause collisions!
- Data generated periodically
  - UE send $N_{Se} \geq 1$ copies as packets
- UEs use same resources for R_c transmissions
  - 1 "transmission" = N_{Se} packets

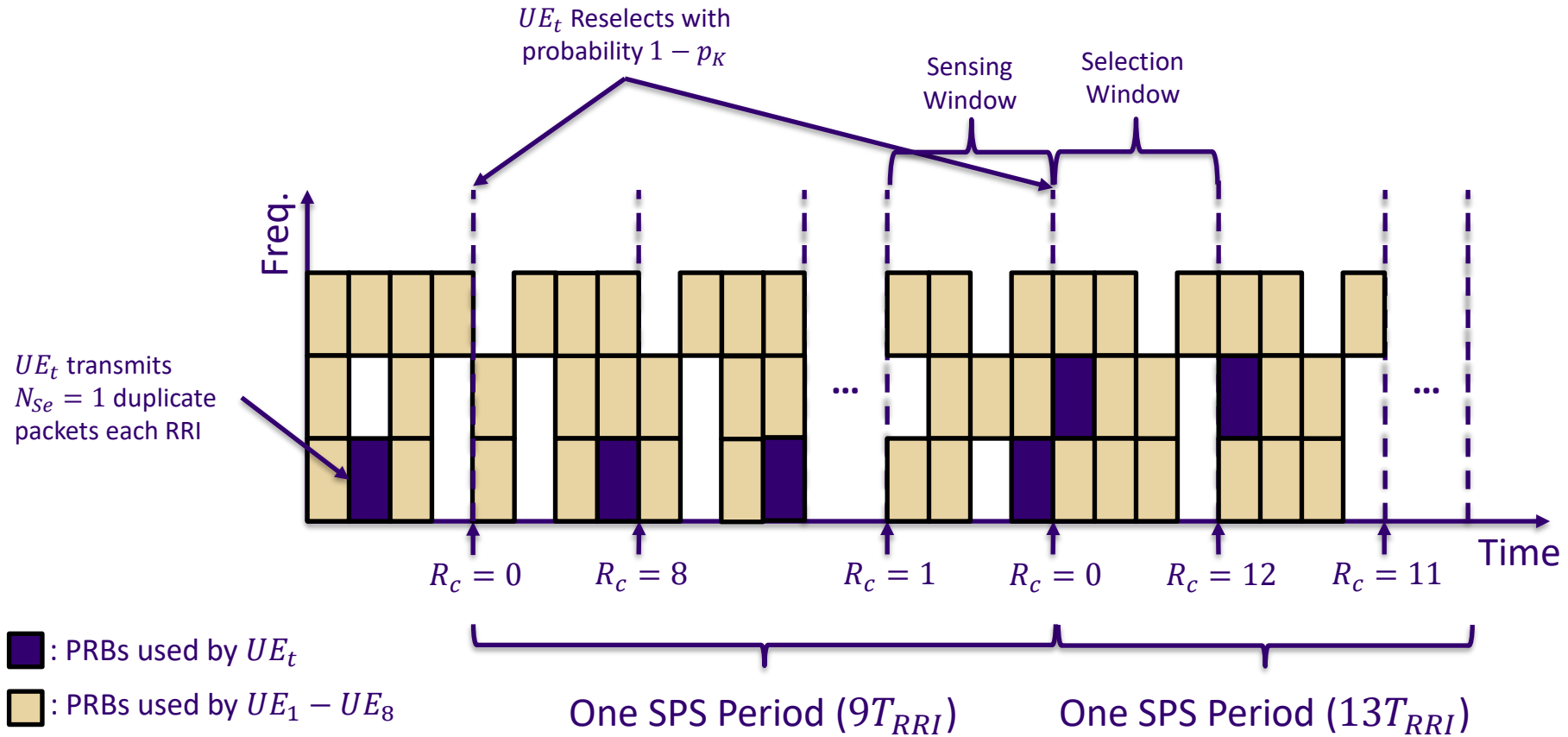


- After R_c transmissions UEs may reselect resources
- Reselection process: Semi-Persistent Scheduling (SPS)
  - "senses" channel to determine occupied resources
  - Sensing: measuring RSRP in sensing window and projecting forward to selection window
    - RSRP>γ_{SPS}=excluded resource

Section 1: C-V2X Standards Introduction: SPS flowchart



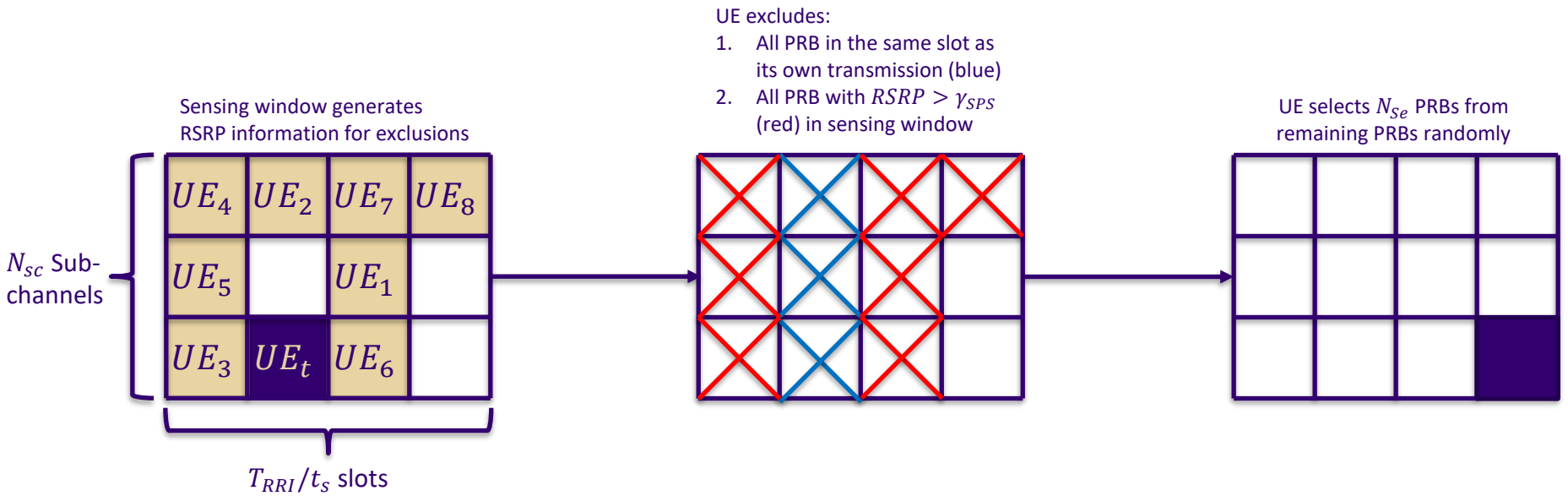
Section 1: C-V2X Standards Introduction: SPS (1)



After PRB(s) are selected, R_c is selected randomly, $R_c \sim U$ with bounds dependent on T_{RRI}, for $T_{RRI} = 100$, $R_c \sim U(5,15)$

Key Parameters:
N_{Se}, p_K

Section 1: C-V2X Standards Introduction: SPS (2): Resource Reselection



Process Defined in TS 38.214

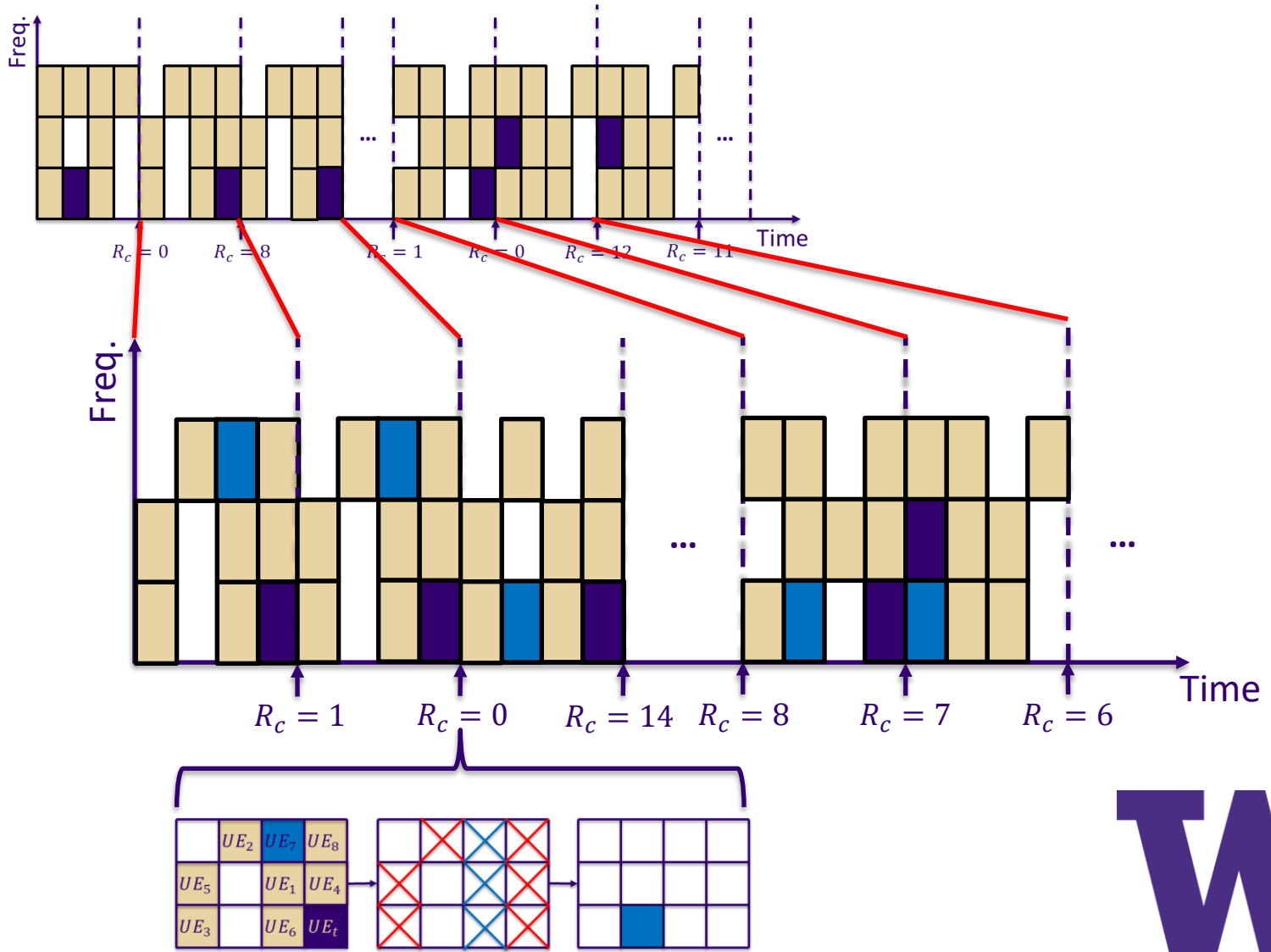
Key Parameters: N_r

Total PRBs in selection window: $N_r = N_{sc}\mathrm{T_{RRI}/t_s}$

: Exclusions due to HD

: Exclusions due to excess RSRP

Section 1: C-V2X Standards Introduction: Meanwhile… Other UEs perform SPS Simultaneously

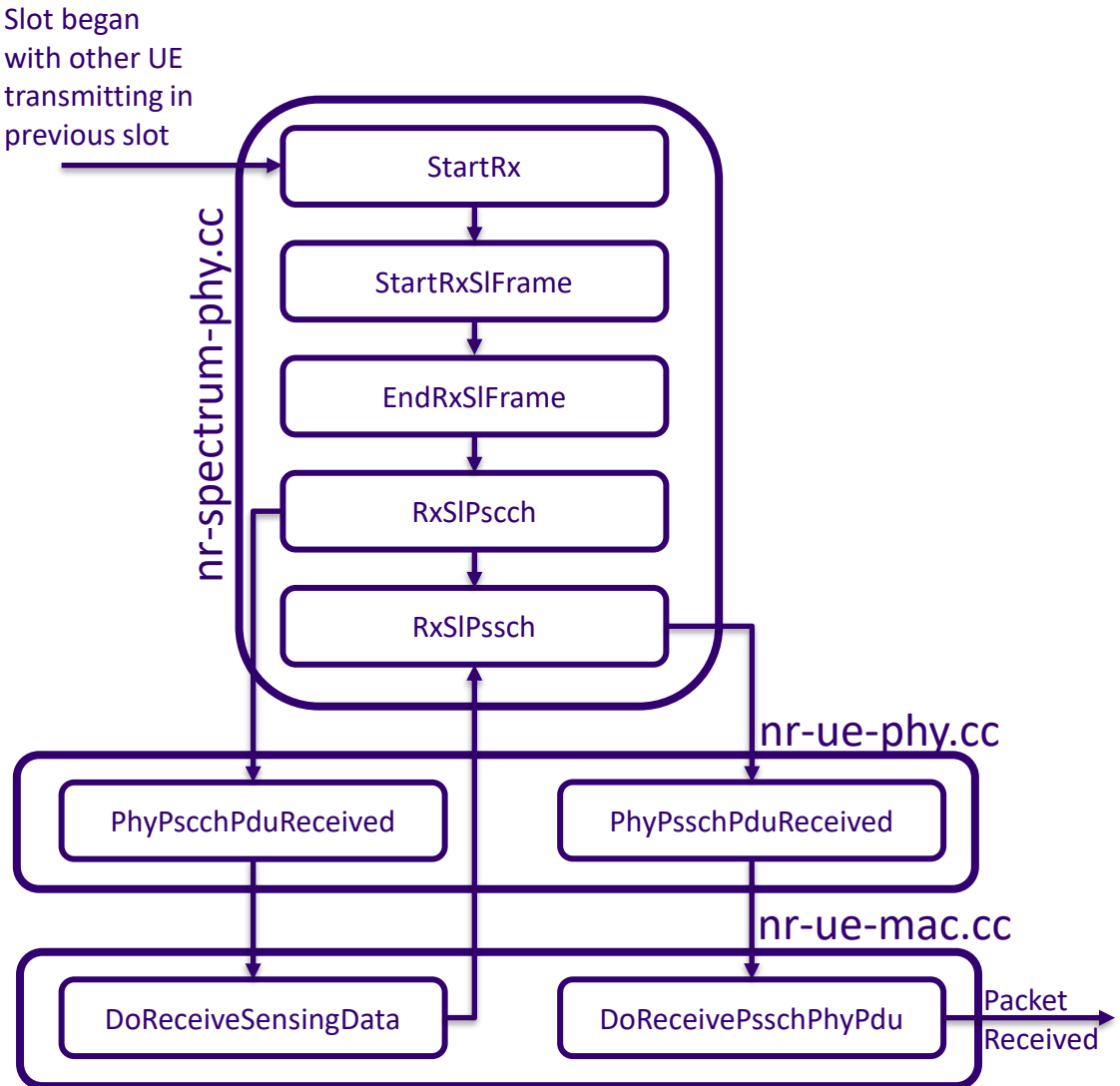


Section 2:

Simple Scenario (MAC Collision Model)

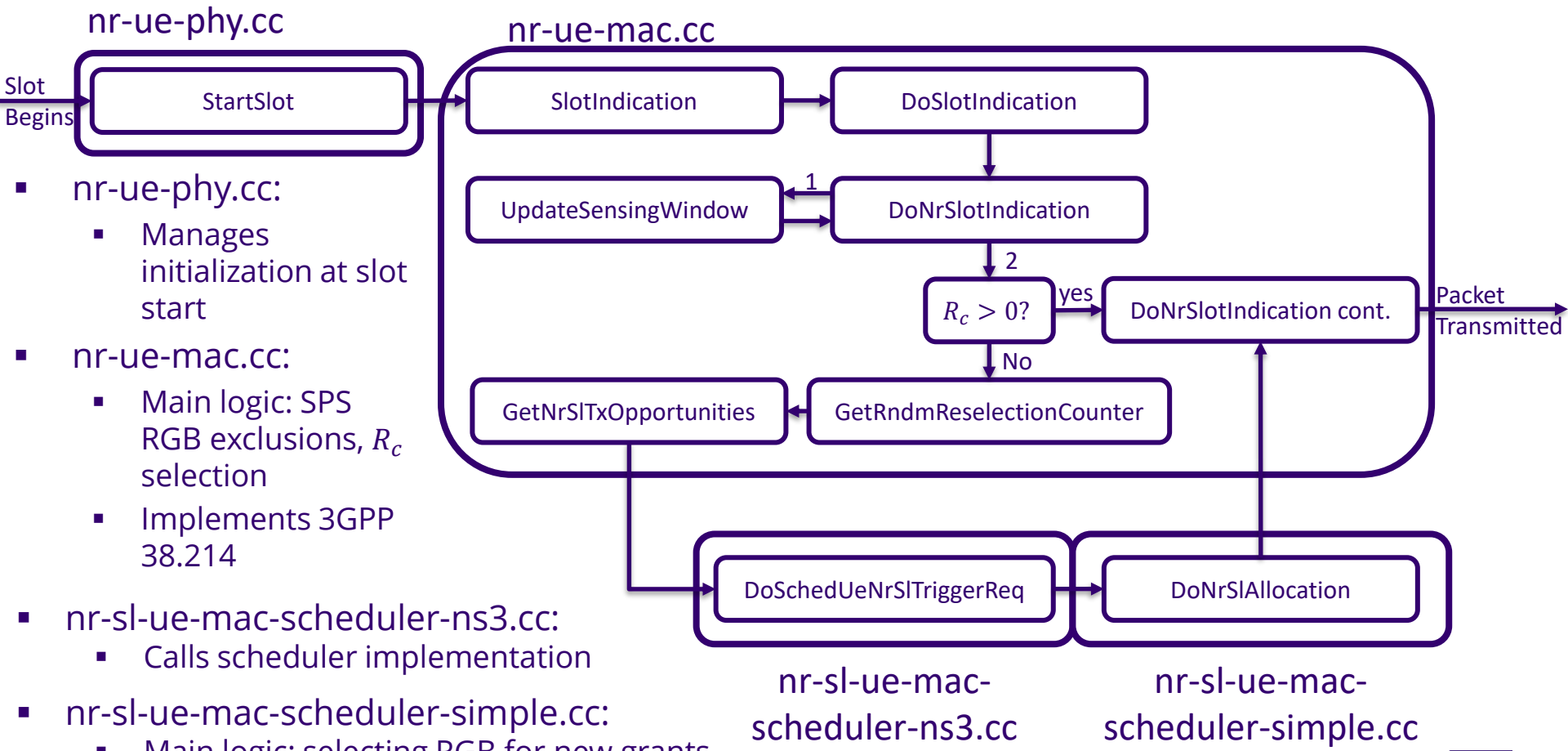
All code used to generate all data can be found at
https://github.com/CollinBrady1993/Code-for-NR-C-V2X-Tutorial

Section 2.1: ns-3 Implementation Overview: Packet Reception



- nr-spectrum-phy:
  - Decomposes packet into constituent parts
    - PSCCH/PSSCH
  - Computes relevant channel effects
    - Interference, RSRP, SINR, etc.
- nr-ue-phy.cc:
  - Creates *sensingData* object
    - Used during SPS to calculate exclusions
- nr-ue-mac.cc
  - Logs packet reception and updates sensing window

Section 2.1: ns-3 Implementation Overview: Packet Generation/Transmission Flow + SPS



- nr-ue-phy.cc:
  - Manages initialization at slot start
- nr-ue-mac.cc:
  - Main logic: SPS RGB exclusions, R_c selection
  - Implements 3GPP 38.214
- nr-sl-ue-mac-scheduler-ns3.cc:
  - Calls scheduler implementation
- nr-sl-ue-mac-scheduler-simple.cc:
  - Main logic: selecting RGB for new grants

Section 2.1: ns-3 Implementation Overview: Loss and Error Models

- Phy Model from "New Radio Physical Layer Abstraction for System-Level Simulations of 5G Networks" by Lagen et al.
- three-gpp-propagation-loss-model.cc used for propagation losses
  - MIMO features in three-gpp-spectrum-propagation-loss-model.cc disabled for simplicity
- Nr Module uses tables in nr-eesm-t2.cc to obtain BLER from SINR after propagation losses
  - Tables recreate curves in Lagen et al.

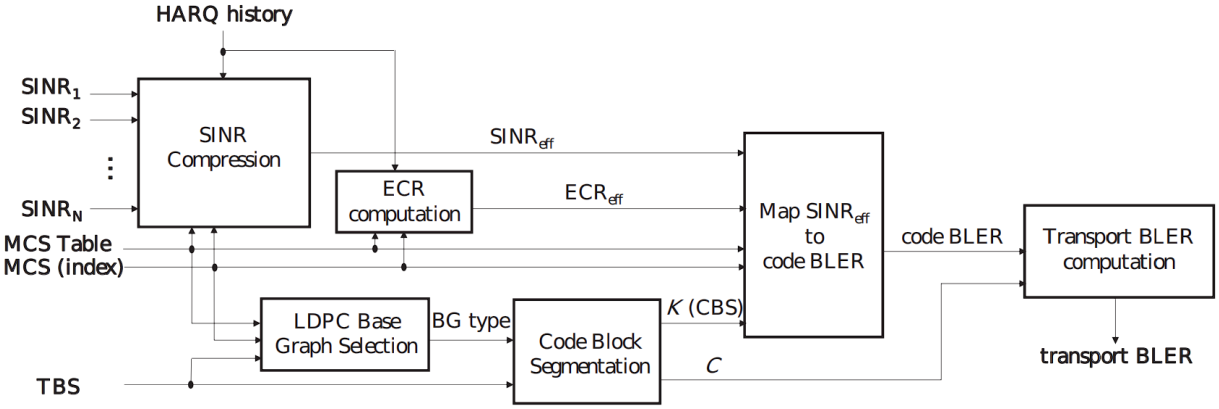
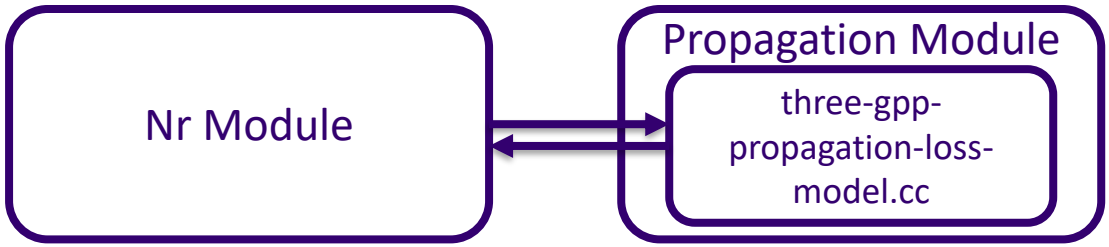


Fig. 1: NR PHY abstraction model.

Figure from "New Radio Physical Layer Abstraction for System-Level Simulations of 5G Networks" by Lagen et al.



Section 2.1: ns-3 Implementation Overview: Example Simulation File

- Uses nr-v2x-west-to-east-highway.cc as base
  - Main changes:
    - Under the hood, fixing bugs
    - Mobility changes
      - More compact, no movement – mac collision model
      - Opposite direction movement – more realistic model
- Main sim: nr-v2x-west-to-east-highway.cc-BradyCao/ nr-v2x-west-to-east-highway.cc-BradyCao-PHYCol
- Github: https://github.com/CollinBrady1993/Code-for-NR-C-V2X-Tutorial

```
int
main (int argc, char *argv[])
{
  /*
   * Variables that represent the parameters we will accept as input by the
   * command line. Each of them is initialized with a default value.
   */
  uint16_t numVehiclesPerLane = 130;
  uint16_t numLanes = 2;
  uint16_t interVehicleDist = 20; //meters
  uint16_t interLaneDist = 4; //meters
  double speed = 0; //meter per second, default 140 km/h - 44.7 m/s=100mph, 22
  bool enableOneTxPerLane = false;
  bool logging = false;

  // Traffic parameters (that we will use inside this script:)
  bool useIPv6 = false; // default IPV4
  uint32_t udpPacketSizeBe = 200;
  double dataRateBe = 16; //16 kilobits per second

  // Simulation parameters.
  Time simTime = Seconds (100);
  Time reportingInterval = Seconds (1);
  //Sidelink bearers activation time
  Time slBearersActivationTime = Seconds (2.0);


  // NR parameters. We will take the input from the command line, and then we
  // will pass them inside the NR module.
  double centralFrequencyBandSl = 5.89e9; // band n47  TDD //Here band is anal
  uint16_t bandwidthBandSl = 100; //Multiple of 100 KHz; 400 = 40 MHz, N_sc*10
  double txPower = 15; //dBm
  std::string tddPattern = "UL|";//"UL|UL|UL|UL|UL|UL|UL|UL|UL|UL|";
  std::string slBitMap = "1|1|";//"1|1|1|1|1|1|1|1|1|1|1|1|1|1|1|1|1|1|1|1|";/
  uint16_t numerologyBwpSl = 0;
  uint16_t slSensingWindow = 100; // T0 in ms
  uint16_t slSelectionWindow = 5; // T2min
  uint16_t slSubchannelSize = 50;
  uint16_t slMaxNumPerReserve = 1;
  double slProbResourceKeep = 0.8;//pKeep
  uint16_t slMaxTxTransNumPssch = 1;//NSe
  uint16_t reservationPeriod = 50; // in ms
  bool enableSensing = true;
  uint16_t t1 = 0;
  uint16_t t2 = 49;
  int slThresPsschRsrp = -164;
  bool enableChannelRandomness = true;
  uint16_t channelUpdatePeriod = 1; //ms
  uint8_t mcs = 14;
```

Section 2.1: ns-3 Implementation Overview: Simulation Output Files

- Additional traces not shown: pscchTxUeMac, pscchRxUePhy
  - Similar to pssch versions shown here
- For all traces:
  - Some entries hidden for clarity; more data available
- psschTxUeMac:
  - Main trace for p_{COL} calculation
- pktTxRx:
  - Main trace for PRR calculation
  - Collates duplicate retransmission receptions into one entry

psschTxUeMac

timeMs	imsi	rbStart	reselCounter
39884.0	223	150	0
39884.0	21	100	6
39884.0	216	50	2
39884.0	174	50	7
39884.0	156	150	11
39885.0	129	0	1
39885.0	42	50	3
39885.0	122	100	8
39886.0	163	100	2
39886.0	147	0	11
39887.0	33	0	0
39887.0	229	0	0
39888.0	137	100	7
39888.0	224	50	13
39889.0	38	50	12
39890.0	171	50	1
39891.0	50	50	8
39891.0	199	100	3
39892.0	119	150	8
39892.0	249	0	2
39892.0	196	50	14
39893.0	298	50	3
39893.0	275	150	7
39894.0	159	150	10
39894.0	205	50	8
39894.0	15	100	2
39895.0	284	50	10
39895.0	31	150	6
39895.0	197	0	7
39896.0	194	150	1
39896.0	245	0	2
39896.0	233	50	4

pktTxRx

timeSec	txRx	nodeId	srcIp	pktSeqNum
39.83	tx	263	7.0.1.9	371
39.83	tx	73	7.0.0.75	370
39.83	tx	173	7.0.0.175	370
39.83	tx	273	7.0.1.19	370
39.83	tx	83	7.0.0.85	369
39.83	tx	183	7.0.0.185	369
39.83	tx	283	7.0.1.29	369
39.83	tx	93	7.0.0.95	368
39.83	tx	193	7.0.0.195	368
39.83	tx	293	7.0.1.39	368
39.830028563	rx	59	7.0.0.228	373
39.830028563	rx	67	7.0.0.228	373
39.830028563	rx	81	7.0.0.228	373
39.830028563	rx	83	7.0.0.228	373
39.830028563	rx	89	7.0.0.228	373
39.830028563	rx	101	7.0.0.114	375
39.830028563	rx	111	7.0.1.18	369
39.830028563	rx	115	7.0.1.18	369
39.830028563	rx	118	7.0.0.114	375
39.830028563	rx	119	7.0.0.114	375
39.830028563	rx	134	7.0.1.18	369
39.830028563	rx	213	7.0.0.228	373
39.830028563	rx	234	7.0.0.228	373
39.830028563	rx	254	7.0.0.114	375
39.830028563	rx	268	7.0.0.114	375
39.830028563	rx	284	7.0.1.18	369
39.830028563	rx	285	7.0.1.18	369
39.830028563	rx	67	7.0.0.228	374
39.830028563	rx	69	7.0.0.228	374
39.830028563	rx	70	7.0.0.228	374
39.830028563	rx	71	7.0.0.228	374
39.830028563	rx	72	7.0.0.228	374

psschRxUePhy

timeMs	rnti	txRnti	psschRbStart	psschCorrupt	sci2Corrupt
39729.928563	79	227	0	0	0
39729.928563	80	227	0	0	0
39729.928563	81	227	0	0	0
39729.928563	82	227	0	0	0
39729.928563	83	227	0	0	0
39729.928563	84	227	0	0	0
39729.928563	90	227	0	0	0
39729.928563	102	113	100	0	0
39729.928563	106	113	100	0	0
39729.928563	107	113	100	0	0
39729.928563	108	113	100	0	0
39729.928563	109	113	100	0	0
39729.928563	110	113	100	0	0
39729.928563	111	113	100	0	0
39729.928563	112	274	0	0	0
39729.928563	112	113	100	0	0
39729.928563	114	274	0	0	0
39729.928563	114	113	100	0	0
39729.928563	115	274	0	0	0
39729.928563	115	113	100	0	0
39729.928563	116	274	0	0	0
39729.928563	116	113	100	0	0
39729.928563	117	274	0	0	0
39729.928563	117	113	100	0	0
39729.928563	118	274	0	0	0
39729.928563	118	113	100	0	0
39729.928563	119	274	0	0	0
39729.928563	119	113	100	0	0
39729.928563	120	274	0	0	0
39729.928563	120	113	100	0	0
39729.928563	121	274	0	0	0
39729.928563	122	274	0	0	0

Section 2.1: ns-3 Implementation Overview: Matlab Analysis

p_{COL} analysis main logic

```
channel = cell((round(max(time)-min(time)))*1000+1,max(sc)+1)';
for i = 1:max(imsi)
    timeTemp = time(imsi == i) - min(time);
    scTemp = sc(imsi == i);
    RcTemp = Rc(imsi == i);

    for j = 1:length(timeTemp)
        channel{scTemp(j)+1,round(timeTemp(j)*1000)+1} = [channel{scTemp(j)+1,round(timeTemp(j)*1000)+1};[i,RcTemp(j)]];
    end
end
collisionData = zeros(numUe,2);
collisionData1 = zeros(numUe,2);
collisionData2 = zeros(numUe,2);
for ueNum = 1:numUe
    collisions = zeros(size(channel,1)*size(channel,2),3);
    a = 1;
    collision1 = zeros(size(channel,1)*size(channel,2),3);
    b = 1;
    collision2 = zeros(size(channel,1)*size(channel,2),3);
    c = 1;
    for i = 1:size(channel,1)
        for j = 1:size(channel,2)
            if size(channel{i,j},1)>1 && ismember(ueNum,channel{i,j}(:,1))
                temp = [round((time(imsi==channel{i,j}(find(channel{i,j}(:,1) == ueNum),1))-min(time))*1000)+1,sc(imsi==channel{i,j}(find(channel{i,j}(:,1) == ueNum),1)),Rc(imsi==channel{i,j}(find(channel{i,j}(:,1) == ueNum),1)
                lastReselection = find(temp(:,3)==0 & temp(:,1) < j,1,'last');%index of the last reselection event for this UE
                if mod(temp(temp(:,1)==j,1) - temp(lastReselection,1),RRI*1000)~=0 | (temp(temp(:,1)==j,2) - temp(lastReselection,2)) ~= 0 %if the slot of subchannel has changed
                    collision1(b,:) = [i,j,size(channel{i,j},1)];%[collision1;[i,j,size(channel{i,j},1)]];
                    b = b+1;
                    %break
                else
                    collision2(c,:) = [i,j,size(channel{i,j},1)];%[collision2;[i,j,size(channel{i,j},1)]];
                    c = c+1;
                end
                collisions(a,:) = [i,j,size(channel{i,j},1)];%[collisions;[i,j,size(channel{i,j},1)]];
                a = a+1;
            end
        end
    end
    collisionData(ueNum,:) = [size(collisions(1:(a-1),:),1),length(imsi(imsi == ueNum))];
    collisionData1(ueNum,:) = [size(collision1(1:(b-1),:),1),length(imsi(imsi == ueNum))];
    collisionData2(ueNum,:) = [size(collision2(1:(c-1),:),1),length(imsi(imsi == ueNum))];
end
pCOL1 = [mean(collisionData1(:,1)./collisionData1(:,2)),std(collisionData1(:,1)./collisionData1(:,2)),numUe];
pCOL2 = [mean(collisionData2(:,1)./collisionData2(:,2)),std(collisionData2(:,1)./collisionData2(:,2)),numUe];
pCOL = [mean(collisionData(:,1)./collisionData(:,2)),std(collisionData(:,1)./collisionData(:,2)),numUe];
```

- Matlab used to analyze ns-3 data
- All plots produced using Matlab

PRR analysis main logic part 1

```
rawData = zeros(size(txRecord,1)*numUe,5);
a = 1;
for i = 1:numUe
    temp = rxRecord(rxRecord(:,2)==i-1,:);%all receptions with matching transmitter
    if mod(i,10) == 0 || i == 1
        i;
        disp(['UE ',num2str(i),' of ',num2str(numUe)])
        datestr(now)
    end
    for j = (min(temp(:,4))+1):(max(temp(:,4))-1)%the first and last packets tend to only be partial data, so we ignore them
        temp2 = temp(temp(:,4)==j,:);%all receptions with matching seqNum and transmitter

        rtx = initialPos(i-1+1,:);%current tx Pos
        rxList = [0:(numUe-1)]';
        rrx = initialPos;%current rx Pos

        rxBool = rxList~=(i-1);

        dr = vecnorm((rtx-rrx)')';

        receptions = ismember(rxList,temp2(:,3));
        dataTime = zeros(length(rxList),1);
        dataTime(receptions) = min(temp2(:,1));

        if isempty(temp2(:,3))%if NOBODY recieved the packet
            dataTime(dataTime == 0) = temp(1,1);
        else
            dataTime(dataTime == 0) = min(temp2(:,1));
        end

        rawData(a:(a+sum(rxBool)-1),:) = [dataTime(rxBool),repmat(i-1,sum(rxBool),1),rxList(rxBool),dr(rxBool),receptions(rxBool)];
        a = a + sum(rxBool);
    end
end

rawData(a:end,:) = [];
```

PRR analysis main logic part 2

```
posMat = zeros(numUe);
for i = 1:numUe
    for j = (i+1):numUe
        posMat(i,j) = vecnorm((initialPos(i,:)-initialPos(j,:)));
    end
end

disp('calculating PDR')
PRR = zeros(length(binEdges)-1,3);

for i = 1:length(binEdges)-1
    binData = rawData(rawData(:,4) <= binEdges(i+1) & rawData(:,4) > binEdges(i),5);

    if isempty(binData)
        PRR(i,1) = 0;
        PRR(i,2) = 0;
        PRR(i,3) = 2*sum(sum(posMat > binEdges(i) & posMat < binEdges(i+1)));
    else
        PRR(i,1) = mean(binData);
        PRR(i,2) = std(binData);
        PRR(i,3) = 2*sum(sum(posMat > binEdges(i) & posMat <= binEdges(i+1)));
    end
end
```

Section 2.2: Remaining Issues: Standards Ambiguous on when R_c Decrementation Occurs

- From the standards (38.321 5.22.1.3.1a):
  - "If the Sidelink HARQ Entity requests a new transmission, the Sidelink process shall:
    - Generate a transmission as described below.
    - if this transmission corresponds to the last transmission of the MAC PDU:
      - decrement SL_RESOURCE_RESELECTION_COUNTER by 1, if available."
- Ambiguous if decrementation of R_c occur before or after transmission is generated
- ns-3 decrements before, produces strange results for $N_{se} > 1$
- Nonissue while R_c is not transmitted but if it were, it would cause erroneous results.
  - Packet sent at 2188 ms will not occupy resource for 11 more transmissions, only 10
- Decrementing after - UEs use resource one more time than current implementation

timeMs	imsi	reselCounter	cReselCounter
Filter	=1	Filter	Filter
2188.0	1	10	100
2207.0	1	9	99
2288.0	1	9	99
2307.0	1	8	98
2388.0	1	8	98
2407.0	1	7	97
2488.0	1	7	97
2507.0	1	6	96
2588.0	1	6	96
2607.0	1	5	95
2688.0	1	5	95
2707.0	1	4	94
2788.0	1	4	94
2807.0	1	3	93
2888.0	1	3	93
2907.0	1	2	92
2988.0	1	2	92
3007.0	1	1	91
3088.0	1	1	91
3107.0	1	0	90
3130.0	1	11	110
3185.0	1	10	109
3230.0	1	10	109
3285.0	1	9	108
3330.0	1	9	108
3385.0	1	8	107
3430.0	1	8	107
3485.0	1	7	106
3530.0	1	7	106
3585.0	1	6	105

Section 2.2: Remaining Issues:
Standards Sensing Ignores PRB with no Packet Decode, Waste of Information and Problematic in NR-U

- By Standards only PRB with decoded packets get RSRP measured
  - Other PRB can still have packets occupying them
    - If a collision occurs no decoded packet, but high RSRP
    - UEs should want to avoid these PRBs too
  - In a coexistence scenario packets from other standards cannot be decoded in any circumstance, PRBs should still be avoided
- Lazy ns-3 fix for C-V2X only: include failed decodes in sensingData in nr-spectrum-phy
  - Only works if sensing/selection don't need to differentiate between decoded and undecoded packets
  - Not clear that NR C-V2X code can handle coexistence case currently

```
if (!corrupt)
  {
    error = false;       //at least one control packet is OK
    SpectrumValue psd = m_slSigPerceived.at (paramIndex);
    PscchPduInfo pduInfo;
    pduInfo.packet = packet;
    pduInfo.psd = psd;
    rxControlMessageOkList.push_back (pduInfo);
    //Store the indices of the decoded RBs
    rbDecodedBitmap.insert ( m_slRxSigParamInfo.at (paramIndex).rbBitmap.begin (), m_slRxSigParamInfo.at (paramIndex).rbBitmap.end ());
  }
```

```
SpectrumValue psd = m_slSigPerceived.at (paramIndex);
PscchPduInfo pduInfo;
pduInfo.packet = packet;
pduInfo.psd = psd;
rxControlMessageOkList.push_back (pduInfo);//This will make it so ever packet is considered during SPS reselection, rather than just ones which were successfully decoded

if (!corrupt)
  {
    error = false;       //at least one control packet is OK
    //Store the indices of the decoded RBs
    rbDecodedBitmap.insert ( m_slRxSigParamInfo.at (paramIndex).rbBitmap.begin (), m_slRxSigParamInfo.at (paramIndex).rbBitmap.end ());
  }
```

Section 2.2: Remaining Issues: No Energy Detection

- SL UEs always attempt to decode packets, regardless of RSRP
  - No mechanism to determine if packet exists in PRB, unlike uplink downlink no gNb to tell the UE where to decode
- No energy detection mechanism defined for NR for sidelink
  - TSG RAN WG1 #109-e (https://www.3gpp.org/ftp/TSG_RAN/WG1_RL1/TSGR1_109-e/Docs/R1-2205183.zip) indicates support for reuse of 37.213 (Physical layer procedures for shared spectrum channel access, LTE-U)

Section 2.2: Remaining Issues:

Spectrum Loss Model Hardcoded in nr-helper

- three-gpp-spectrum-propagation-loss-model implements TR 37.885 Section 6.2.3
- Nr-helper defaults to three-gpp-spectrum-propagation-loss-model
- Issue: rest of nr-helper assumes use of three-gpp-spectrum-propagation-loss-model and creates variables of that type, not easy to change spectrum loss model to something else
- (Bad) Solution: Comment out DoCalcRxPowerSpectralDensity, pass input as output
  - Better Solution: Use constant-spectrum-propagation-loss, but hard to implement and tedious to change out later

```
67  NrHelper::NrHelper (void)
68  {
69    NS_LOG_FUNCTION (this);
70    m_channelFactory.SetTypeId (MultiModelSpectrumChannel::GetTypeId ());
71    m_gnbNetDeviceFactory.SetTypeId (NrGnbNetDevice::GetTypeId ());
72    m_ueNetDeviceFactory.SetTypeId (NrUeNetDevice::GetTypeId ());
73    m_ueMacFactory.SetTypeId (NrUeMac::GetTypeId ());
74    m_gnbMacFactory.SetTypeId (NrGnbMac::GetTypeId ());
75    m_ueSpectrumFactory.SetTypeId (NrSpectrumPhy::GetTypeId ());
76    m_gnbSpectrumFactory.SetTypeId (NrSpectrumPhy::GetTypeId ());
77    m_uePhyFactory.SetTypeId (NrUePhy::GetTypeId ());
78    m_gnbPhyFactory.SetTypeId (NrGnbPhy::GetTypeId ());
79    m_ueChannelAccessManagerFactory.SetTypeId (NrAlwaysOnAccessManager::GetTypeId ());
80    m_gnbChannelAccessManagerFactory.SetTypeId (NrAlwaysOnAccessManager::GetTypeId ());
81    m_schedFactory.SetTypeId (NrMacSchedulerTdmaRR::GetTypeId ());
82    m_ueAntennaFactory.SetTypeId (UniformPlanarArray::GetTypeId ());
83    m_gnbAntennaFactory.SetTypeId (UniformPlanarArray::GetTypeId ());
84    m_gnbBwpManagerAlgoFactory.SetTypeId (BwpManagerAlgorithmStatic::GetTypeId ());
85    m_ueBwpManagerAlgoFactory.SetTypeId (BwpManagerAlgorithmStatic::GetTypeId ());
86    m_gnbUlAmcFactory.SetTypeId (NrAmc::GetTypeId ());
87    m_gnbDlAmcFactory.SetTypeId (NrAmc::GetTypeId ());
88    m_gnbBeamManagerFactory.SetTypeId (BeamManager::GetTypeId());
89    m_ueBeamManagerFactory.SetTypeId (BeamManager::GetTypeId());
90    m_spectrumPropagationFactory.SetTypeId (ThreeGppSpectrumPropagationLossModel::GetTypeId ());
91    m_bwpManagerFactory.SetTypeId (BwpManagerUe::GetTypeId ());
```

```
291  // Iterate over all CCs, and instantiate the channel and propagation model
292  for (const auto & cc : band->m_cc)
293    {
294      for (const auto & bwp : cc->m_bwp)
295        {
296          // Initialize the type ID of the factories by calling the relevant
297          // static function defined above and stored inside the lookup table
298          initLookupTable.at (bwp->m_scenario) (&m_pathlossModelFactory, &m_channelConditionModelFactory);
299
300          auto channelConditionModel  = m_channelConditionModelFactory.Create<ChannelConditionModel>();
301
302          if (bwp->m_propagation == nullptr && flags & INIT_PROPAGATION)
303            {
304
305              bwp->m_propagation = m_pathlossModelFactory.Create <ThreeGppPropagationLossModel> ();
306              bwp->m_propagation->SetAttributeFailSafe ("Frequency", DoubleValue (bwp->m_centralFrequency));
307              DynamicCast<ThreeGppPropagationLossModel> (bwp->m_propagation)->SetChannelConditionModel (channelConditionModel);
308            }
309
310          if (bwp->m_3gppChannel == nullptr && flags & INIT_FADING)
311            {
312              bwp->m_3gppChannel = m_spectrumPropagationFactory.Create<ThreeGppSpectrumPropagationLossModel>();
313              DynamicCast<ThreeGppSpectrumPropagationLossModel> (bwp->m_3gppChannel)->SetChannelModelAttribute ("Frequency", DoubleValue (bwp->m_centralFrequency));
314              DynamicCast<ThreeGppSpectrumPropagationLossModel> (bwp->m_3gppChannel)->SetChannelModelAttribute ("Scenario", StringValue (bwp->GetScenario ()));
315              DynamicCast<ThreeGppSpectrumPropagationLossModel> (bwp->m_3gppChannel)->SetChannelModelAttribute ("ChannelConditionModel", PointerValue (channelConditionModel));
316            }
317
318          if (bwp->m_channel == nullptr && flags & INIT_CHANNEL)
319            {
320              bwp->m_channel = m_channelFactory.Create<SpectrumChannel> ();
321              bwp->m_channel->AddPropagationLossModel (bwp->m_propagation);
322              bwp->m_channel->AddPhasedArraySpectrumPropagationLossModel (bwp->m_3gppChannel);
323            }
324        }
325    }
326  }
```

```
1329  int64_t
1330  NrHelper::DoAssignStreamsToChannelObjects (Ptr<NrSpectrumPhy> phy, int64_t currentStream)
1331  {
1332    Ptr<ThreeGppPropagationLossModel> propagationLossModel = DynamicCast<ThreeGppPropagationLossModel> (phy->GetSpectrumChannel ()->GetPropagationLossModel ());
1333    NS_ASSERT (propagationLossModel != nullptr);
1334
1335    int64_t initialStream = currentStream;
1336
1337    if (std::find (m_channelObjectsWithAssignedStreams.begin(),
1338                   m_channelObjectsWithAssignedStreams.end (),
1339                   propagationLossModel) == m_channelObjectsWithAssignedStreams.end ())
1340      {
1341        currentStream += propagationLossModel->AssignStreams (currentStream);
1342        m_channelObjectsWithAssignedStreams.push_back (propagationLossModel);
1343      }
1344
1345    Ptr<ChannelConditionModel> channelConditionModel = propagationLossModel->GetChannelConditionModel();
1346
1347    if (std::find (m_channelObjectsWithAssignedStreams.begin(),
1348                   m_channelObjectsWithAssignedStreams.end (),
1349                   channelConditionModel) == m_channelObjectsWithAssignedStreams.end ())
1350      {
1351        currentStream += channelConditionModel->AssignStreams (currentStream);
1352        m_channelObjectsWithAssignedStreams.push_back (channelConditionModel);
1353      }
1354
1355    Ptr<ThreeGppSpectrumPropagationLossModel> spectrumLossModel = DynamicCast<ThreeGppSpectrumPropagationLossModel> (phy->GetSpectrumChannel ()->GetPhasedArraySpectrumPropagationLossModel ());
1356
1357    if (spectrumLossModel)
1358      {
1359        if (std::find (m_channelObjectsWithAssignedStreams.begin(),
1360                       m_channelObjectsWithAssignedStreams.end (),
1361                       spectrumLossModel) == m_channelObjectsWithAssignedStreams.end ())
1362          {
1363            Ptr <ThreeGppChannelModel> channel = DynamicCast<ThreeGppChannelModel> (spectrumLossModel->GetChannelModel());
1364            currentStream += channel->AssignStreams (currentStream);
1365            m_channelObjectsWithAssignedStreams.push_back (channel);
1366          }
1367      }
1368
1369    return currentStream - initialStream;
1370  }
```

Section 2.2: Remaining Issues: nr-spectrum-phy tries to save time, ignores decodable packets

```
//Compute the error and check for collision for each expected TB
for (auto &tbIt : m_slTransportBlocks)
  {
    NS_ABORT_MSG_IF (tbIt.second.sinrUpdated == false, "SINR not updated for the expected TB from RNTI " << tbIt.first);
    Ptr<Packet> sci2Pkt = RetrieveSci2FromPktBurst (tbIt.second.pktIndex);
    NrSlSciF2aHeader sciF2a;
    sci2Pkt->PeekHeader (sciF2a);
    if (sciF2a.GetNdi ())
      {
        NS_LOG_DEBUG ("RemovePrevDecoded: " << +sciF2a.GetHarqId () << " for the packets received from RNTI " << tbIt.first << " rv " << +sciF2a.GetRv ());
        m_harqPhyModule->RemovePrevDecoded (tbIt.first, sciF2a.GetHarqId ());
      }
    //For blind reTxs, we do not dispatch already decode TBs to UE PHY
    bool isPrevDecoded = m_harqPhyModule->IsPrevDecoded (tbIt.first, sciF2a.GetHarqId ());
    if ((!m_slDataErrorModelEnabled || isPrevDecoded) && (!m_dropTbOnRbCollisionEnabled || isPrevDecoded))
      {
        continue;
      }
```

- RxSlPssch in nr-spectrum-phy tries to save time by ignoring duplicates if another packet in burst is already decoded
  - Highlighted above in red
- Condition used results in ignoring packets under certain conditions
- Quick fix: comment out section
  - Effects shown on right, UE doesn't attempt to decode some packets, by forcing it to check everything more packets are decoded
  - Only an issue for $N_{Se} > 1$
  - Unsure why above code skips packets

	timeMs	rnti	txRnti	psschCorrupt	sci2Corrupt
	Filter	=1	=2	Filter	Filter
214	24494.928563	1	2	0	0
215	24594.928563	1	2	0	0
216	24705.928563	1	2	1	1
217	24783.928563	1	2	0	0
218	24805.928563	1	2	1	1
219	24883.928563	1	2	0	0
220	24905.928563	1	2	1	1
221	24983.928563	1	2	0	0
222	25005.928563	1	2	1	1
223	25083.928563	1	2	0	0
224	25105.928563	1	2	1	1
225	25183.928563	1	2	0	0
226	26005.928563	1	2	0	0
227	26105.928563	1	2	0	0
228	26205.928563	1	2	0	0
229	26305.928563	1	2	0	0
230	26405.928563	1	2	0	0
231	26505.928563	1	2	0	0
232	26605.928563	1	2	0	0
233	26705.928563	1	2	0	0
234	26805.928563	1	2	0	0
235	26905.928563	1	2	0	0
236	27005.928563	1	2	0	0
237	27105.928563	1	2	0	0
238	27249.928563	1	2	0	0
239	27349.928563	1	2	0	0
240	27449.928563	1	2	0	0
241	27549.928563	1	2	0	0
242	27649.928563	1	2	0	0
243	27751.928563	1	2	0	0
244	27851.928563	1	2	0	0
245	27951.928563	1	2	0	0

	timeMs	rnti	txRnti	psschCorrupt	sci2Corrupt
	Filter	=1	=2	Filter	Filter
427	24600.928563	1	2	0	0
428	24705.928563	1	2	1	1
429	24783.928563	1	2	0	0
430	24805.928563	1	2	1	1
431	24883.928563	1	2	0	0
432	24905.928563	1	2	1	1
433	24983.928563	1	2	0	0
434	25005.928563	1	2	1	1
435	25083.928563	1	2	0	0
436	25105.928563	1	2	1	1
437	25183.928563	1	2	0	0
438	25283.928563	1	2	0	0
439	25383.928563	1	2	0	0
440	25483.928563	1	2	0	0
441	25583.928563	1	2	0	0
442	25683.928563	1	2	0	0
	25783.928563	1	2	0	0
444	25883.928563	1	2	0	0
445	25983.928563	1	2	0	0
446	26005.928563	1	2	0	0
447	26009.928563	1	2	0	0
448	26105.928563	1	2	0	0
449	26109.928563	1	2	0	0
450	26205.928563	1	2	0	0
451	26209.928563	1	2	0	0
452	26305.928563	1	2	0	0
453	26309.928563	1	2	0	0
454	26405.928563	1	2	0	0
455	26409.928563	1	2	0	0
456	26505.928563	1	2	0	0

Section 2.2: Remaining Issues: IPV4 does not support >254 UE

Error Triggered

```
Ptr<NrSlDataRadioBearerInfo>
LteUeRrc::AddNrSlDrb (uint32_t srcL2Id, uint32_t dstL2Id, uint8_t lcid)
{
  NS_LOG_FUNCTION (this);

  NS_ABORT_MSG_IF ((srcL2Id == 0 || dstL2Id == 0), "Layer 2 source or destination Id shouldn't be 0");
```

Simple Solution

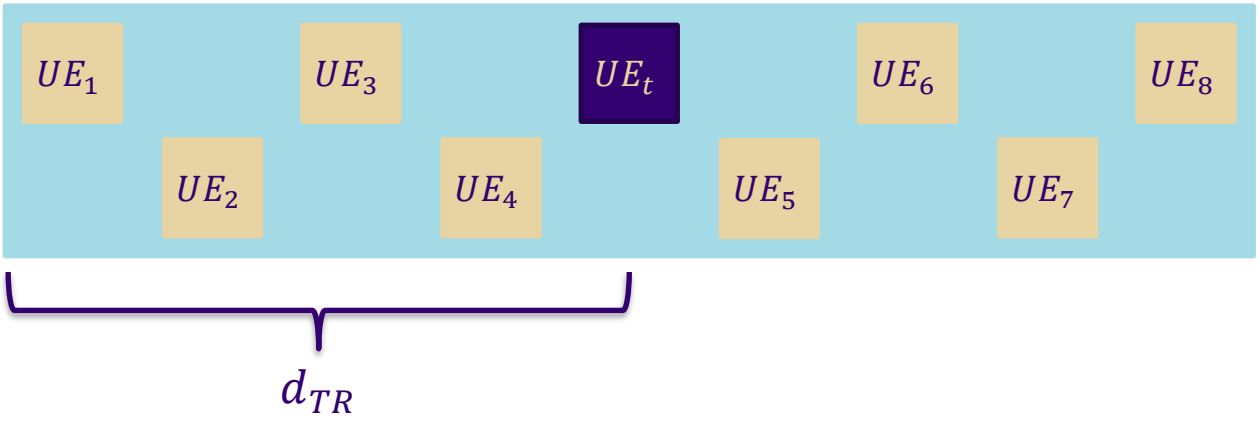
```
NS_ABORT_MSG_IF (m_imsiCounter >= 0xFFFFFFFF, "max num UEs exceeded");
uint64_t imsi = ++m_imsiCounter;
if (m_imsiCounter % 256 == 254)
{
  ++m_imsiCounter;
  ++m_imsiCounter;
}
```

- When using >254 UE IPV4 SL C-V2X triggers above error
  - Caused by using only bottom digits of IP address
  - Not sure on location in code
- Simple solution: skip IP addresses ending in .0 or .1

Section 2.3: Model of PRR for Simple Scenario: MAC Collision Model and Validation

- A pure MAC-level sensitivity analysis for the collision probability (P_{COL}) and packet reception ratio (PRR) in terms of
  - Resource keeping probability, p_K;
  - UE density, N_{UE} (or ρ_{UE}) ;
  - Number of duplicate packet transmissions, N_{Se}.
- The collision events are validated by ns-3.
- Key takeaways from the model and ns-3.

Section 2.3: Model of PRR for Simple Scenario: Physical Scenario – Demonstration Example



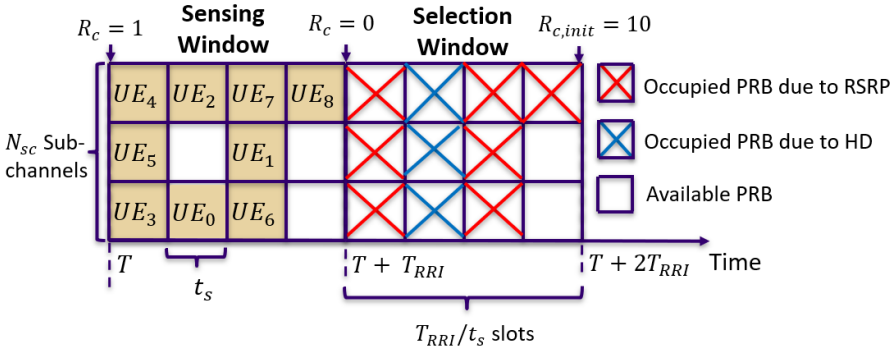
- UEs distributed uniformly in a line with density ρ_{UE} UE/m
- UEs synchronized in time/frequency
- Fully connected network
  - All UEs within d_{TR} of the central UE can decode packets perfectly in the absence of interference
- MAC collision assumption
  - If two (or more) UEs select the same PRB for transmission all UEs fail to decode those packets

Key Parameter:

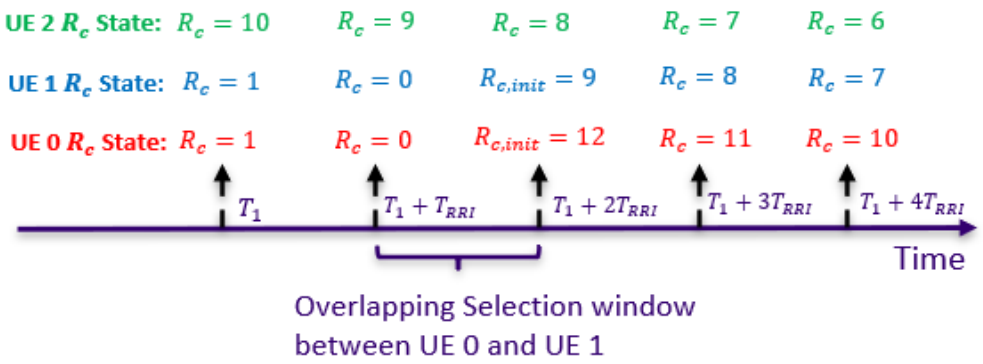
ρ_{UE}

Section 2.3: Model of PRR for Simple Scenario: Setup and Assumption

- No PHY errors (ideal PHY)
- Sensing window size = Selection window size = T_{RRI}
- Total # of PRBs in each RRI: $N_r = N_{sc} T_{RRI} / t_s$
- UE obtains available PRBs by excluding occupied PRBs
  - The available PRBs are assumed to be unoccupied;
- Synchronous Reselection Counter (R_c) decrementation
  - All UEs' R_c decrement (or change) simultaneously.



Sensing window & Selection window



Synchronous R_c Decrementation

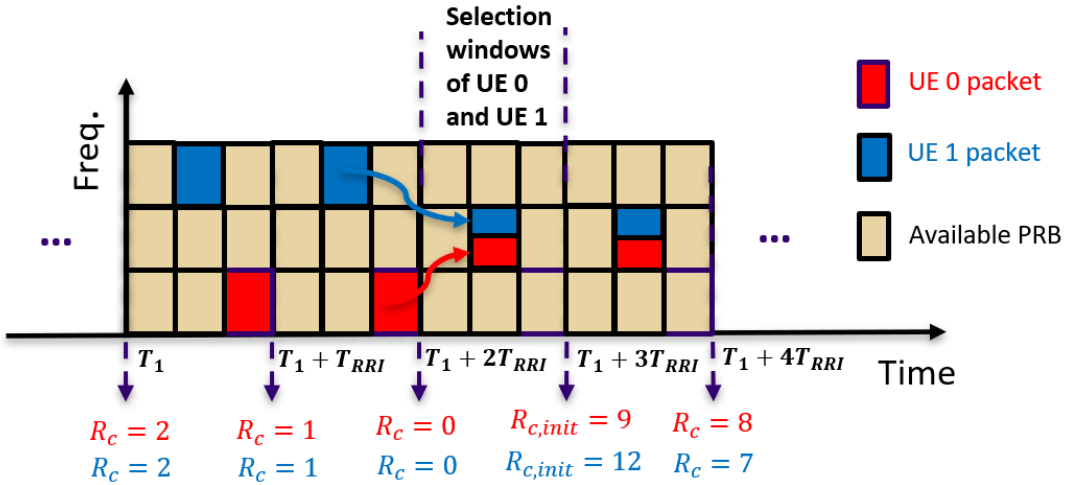
Section 2.3: Model of PRR for Simple Scenario: Collision Events

- **MAC Collision Probability (P_{COL})**
  - MAC collision happens if
    - Two (or more) UEs transmit in the same PRB → All UEs fail to decode any of the overlapped packets;
  - MAC collision event starts in the selection window
    - UE 0 (reference UE)'s R_c decrements to 0 → UE enters the selection window
      - UE 0 performs reselection → **Collision Event 1**;
      - UE 0 does not perform reselection → **Collision Event 2**;
    - Always followed by consecutive collisions right after the selection window.
  - P_{COL} = Pr{Collision Event 1} + Pr{Collision Event 2}

    = $P_{COL,1}$ + $P_{COL,2}$

Section 2.3: Model of PRR for Simple Scenario: Collision Events

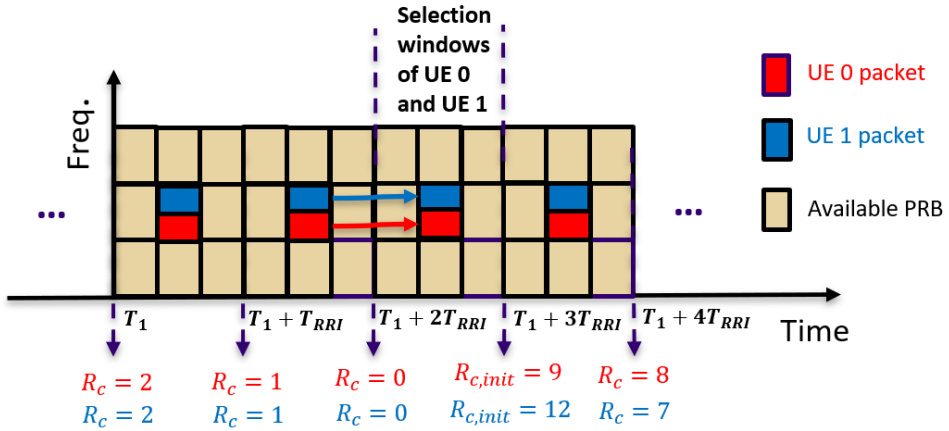
- **Collision Event 1**: UE 0 performs reselection (with probability $1 - p_K$) in the selection window
  - Meanwhile, UE 1's R_c decrements to 0 → UE 1 enters the selection window;
  - UE 1 performs reselection → UE 1 and UE 0 select the same PRB among available PRBs→ Collision happens;
  - $\min\{R_{c,init}^{UE\ 0}, R_{c,init}^{UE\ 1}\} \in [5, 15]$ consecutive collisions must happen right after the selection window.



Collision Event 1

Section 2.3: Model of PRR for Simple Scenario: Collision Events

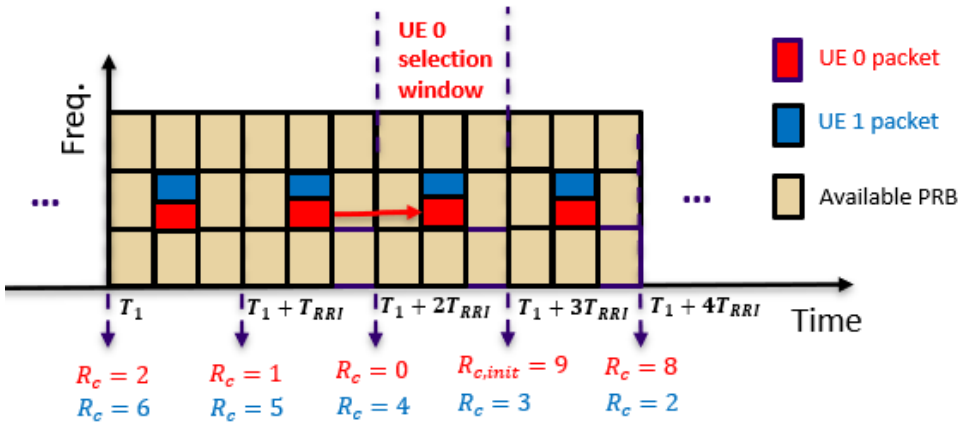
- **Collision Event 2**: UE 0 does not perform reselection (with probability p_K) in the selection window
  - UE 1 has collided with UE 0 in the prior RRIs;
    - Collision Event 2 must be preceded by Collision Event 1;
  - **Sub-Event 1**: Collided UE's $R_c = 0$ when UE 0's $R_c = 0$;
    - UE 1 does not perform reselection → UE 1 and UE 0 stick with the same PRB → Collision happens;
    - $\min\{R_{c,init}^{UE\ 0}, R_{c,init}^{UE\ 1}\} \in [5, 15]$ consecutive collisions must happen right after the selection window.



Collision Event 2: Sub-Event 1

Section 2.3: Model of PRR for Simple Scenario: Collision Events

- **Collision Event 2**: UE 0 does not perform reselection (with probability p_K) in the selection window
  - UE 1 has collided with UE 0 in the prior RRIs;
    - Collision Event 2 must be preceded by Collision Event 1;
  - **Sub-Event 2**: Collided UE's $R_c \neq 0$ when UE 0's $R_c = 0$;
    - UE 1 uses the previously reserved PRB→ UE 1 and UE 0 stick with the same PRB → Collision happens;
    - $\min\{R_{c,init}^{UE\ 0}, R_c^{UE\ 1}\} \in [0, 14]$ consecutive collisions must happen right after UE 0's selection window.

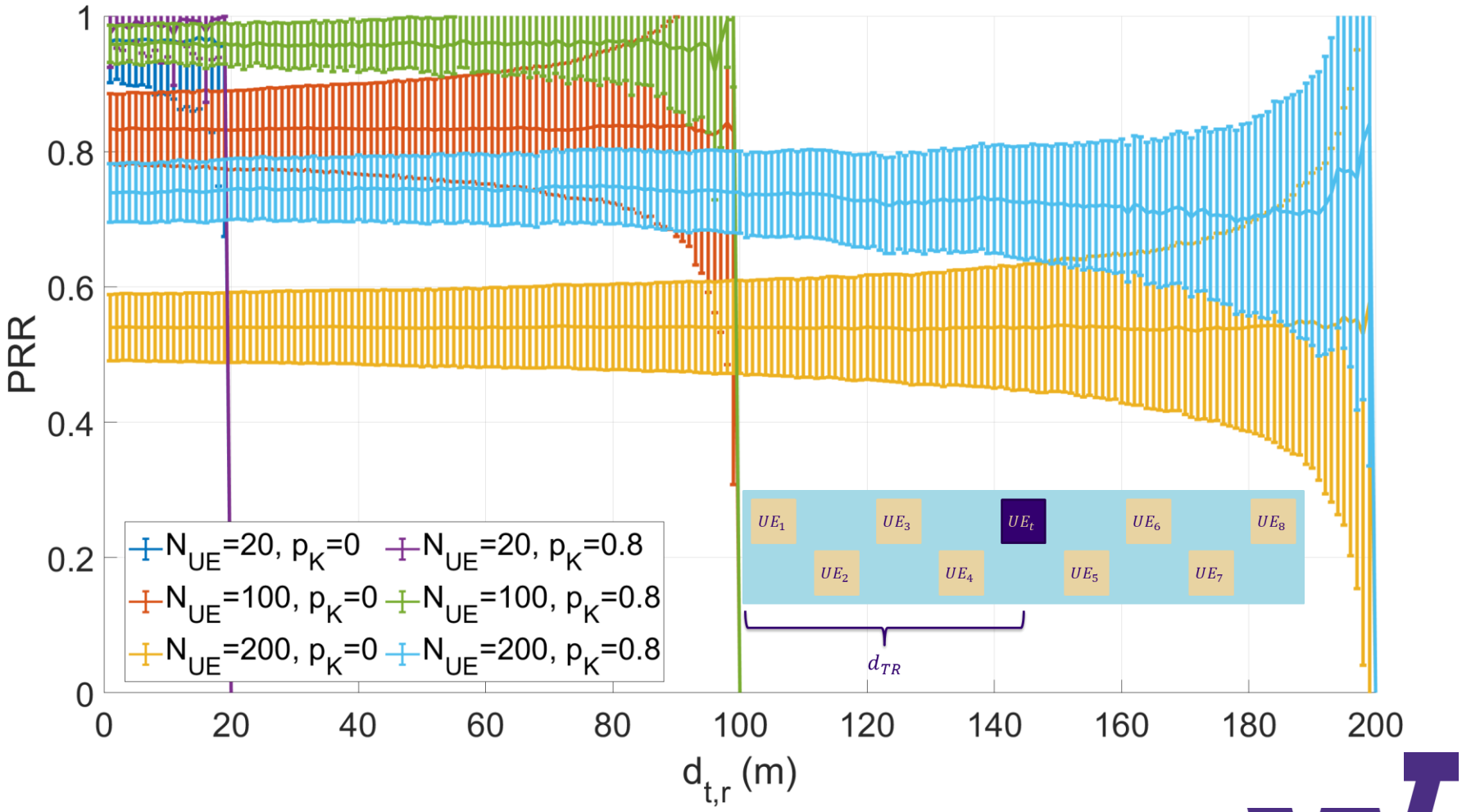


Collision Event 2: Sub-Event 2

Section 2.3: Model of PRR for Simple Scenario: P_{COL} and PRR

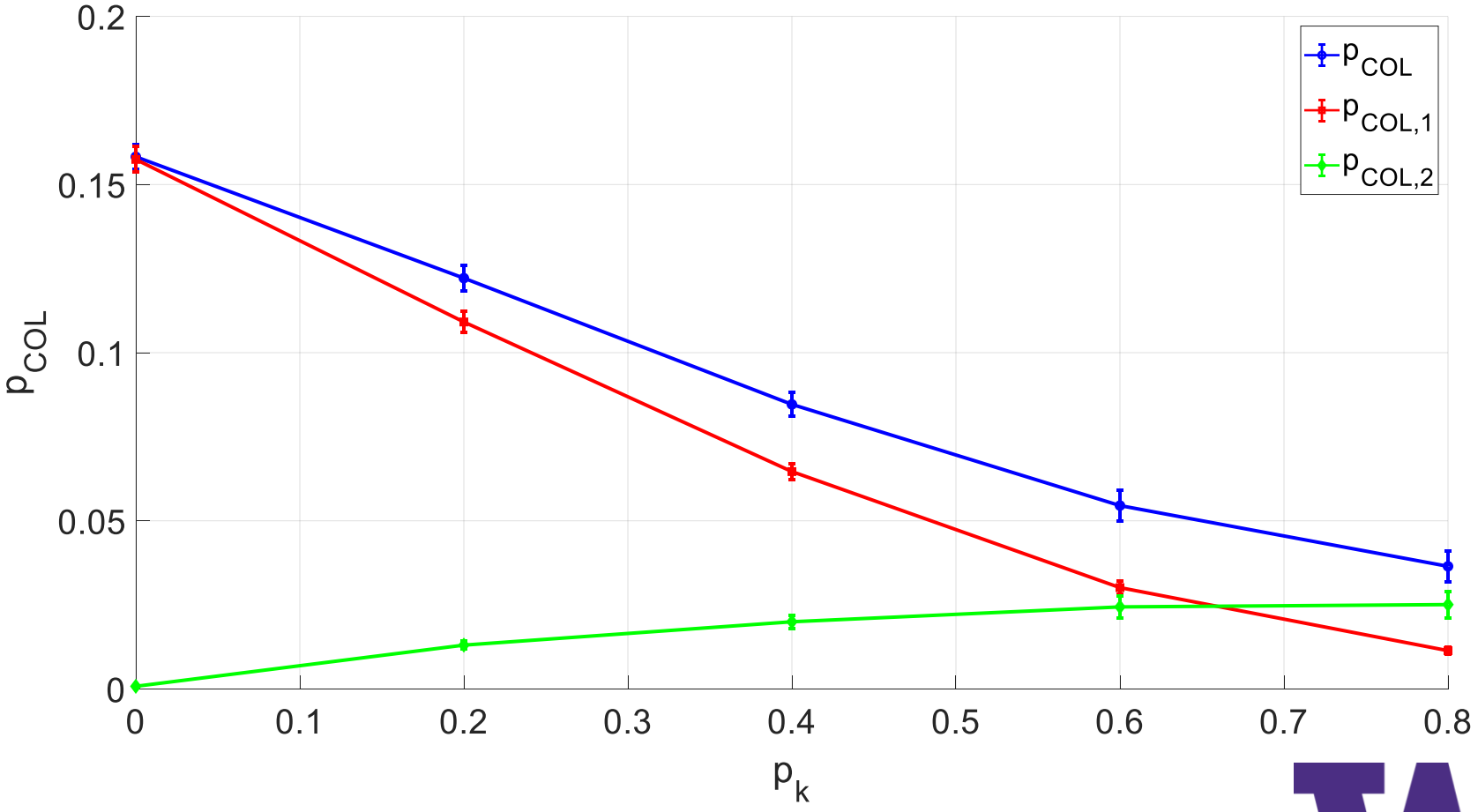
- **Packet Reception Ratio (PRR)**
  - **MAC collision error (P_{COL})**: Two UEs transmit in the same PRB;
  - **Half-duplex error (P_{HD})**: In capable of transmitting and receiving in the same slot;
    - $P_{HD} = \frac{t_s}{T_{RRI}}$
  - **N_{Se} = 1**: Both errors do not happen → This packet received successfully;
    - $PRR = (1 - P_{COL}(N_{Se}, p_K, N_{UE}))(1 - P_{HD})$
  - **N_{Se} > 1**: At least one of N_{Se} duplicate packets received successfully → This packet received successfully;
    - $PRR = 1 - \left(1 - (1 - P_{COL}(N_{Se}, p_K, N_{UE}))(1 - P_{HD})\right)^{N_{Se}}$

Section 2.4: Results for Simple Example Scenario: UEs are fully connected – no distance dependance on PRR

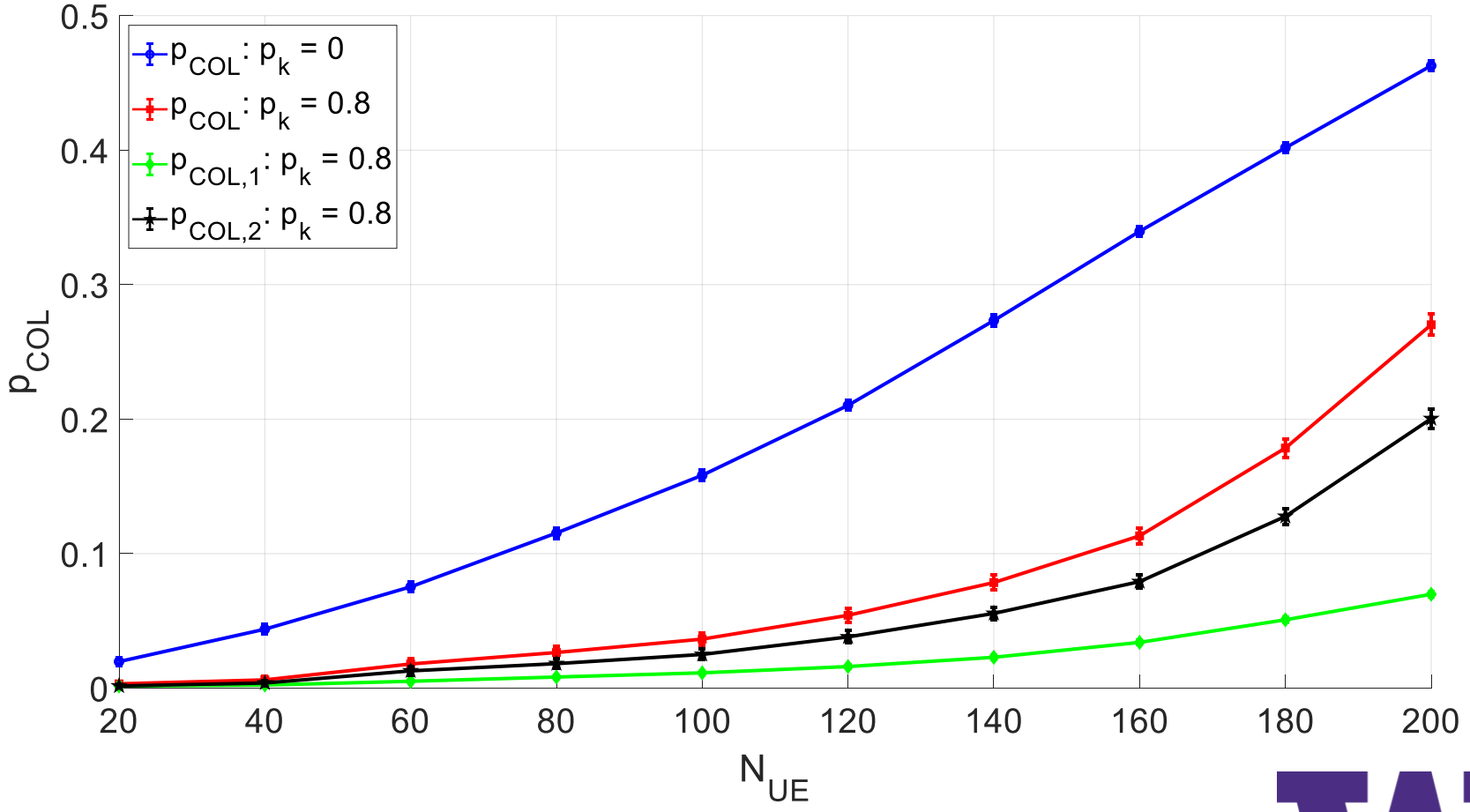


Note: $N_r = 200$ in our simulations

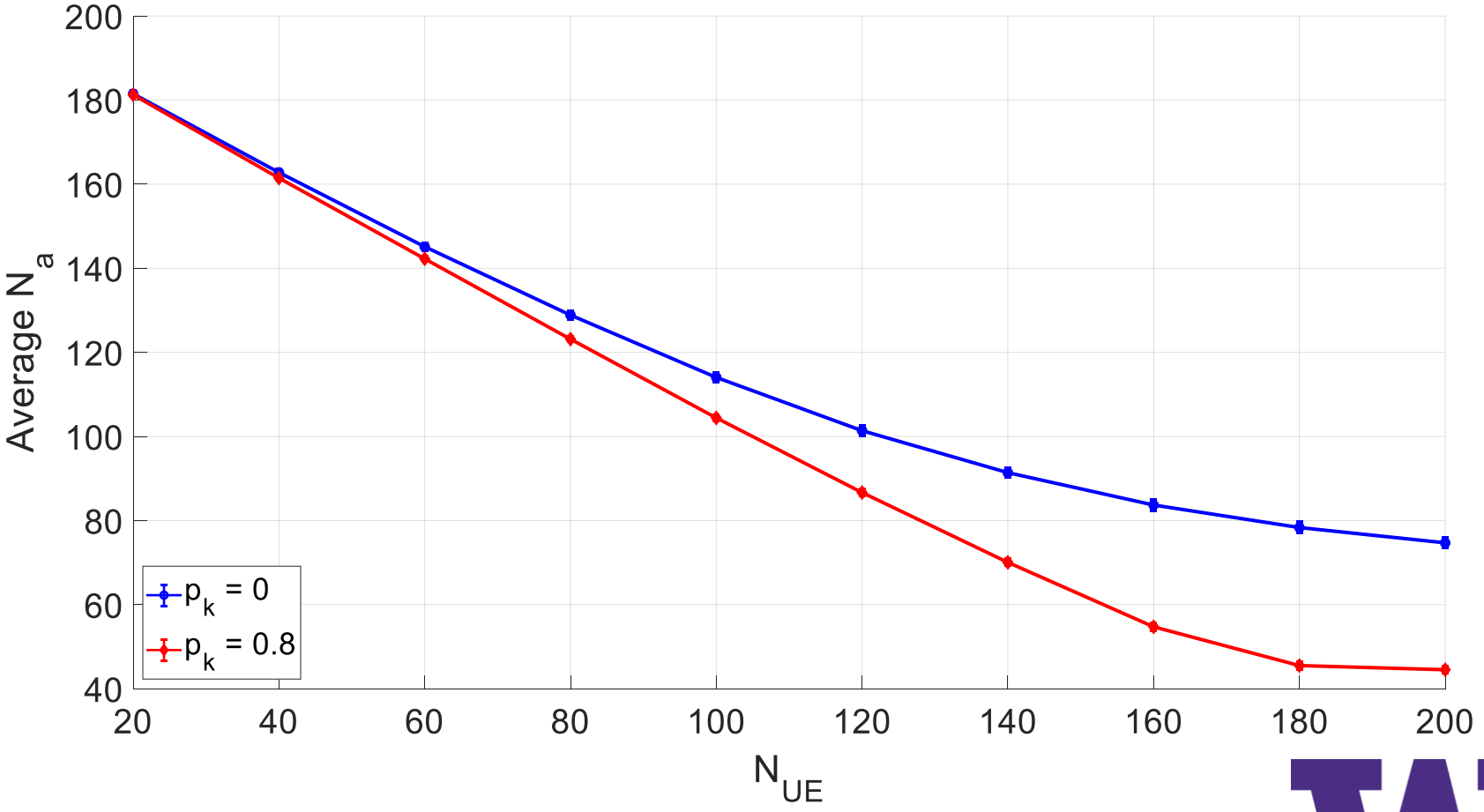
Section 2.4: Results for Simple Example Scenario: Effect of p_K on p_{COL}: $N_{UE} = 100$, $N_{Se} = 1$



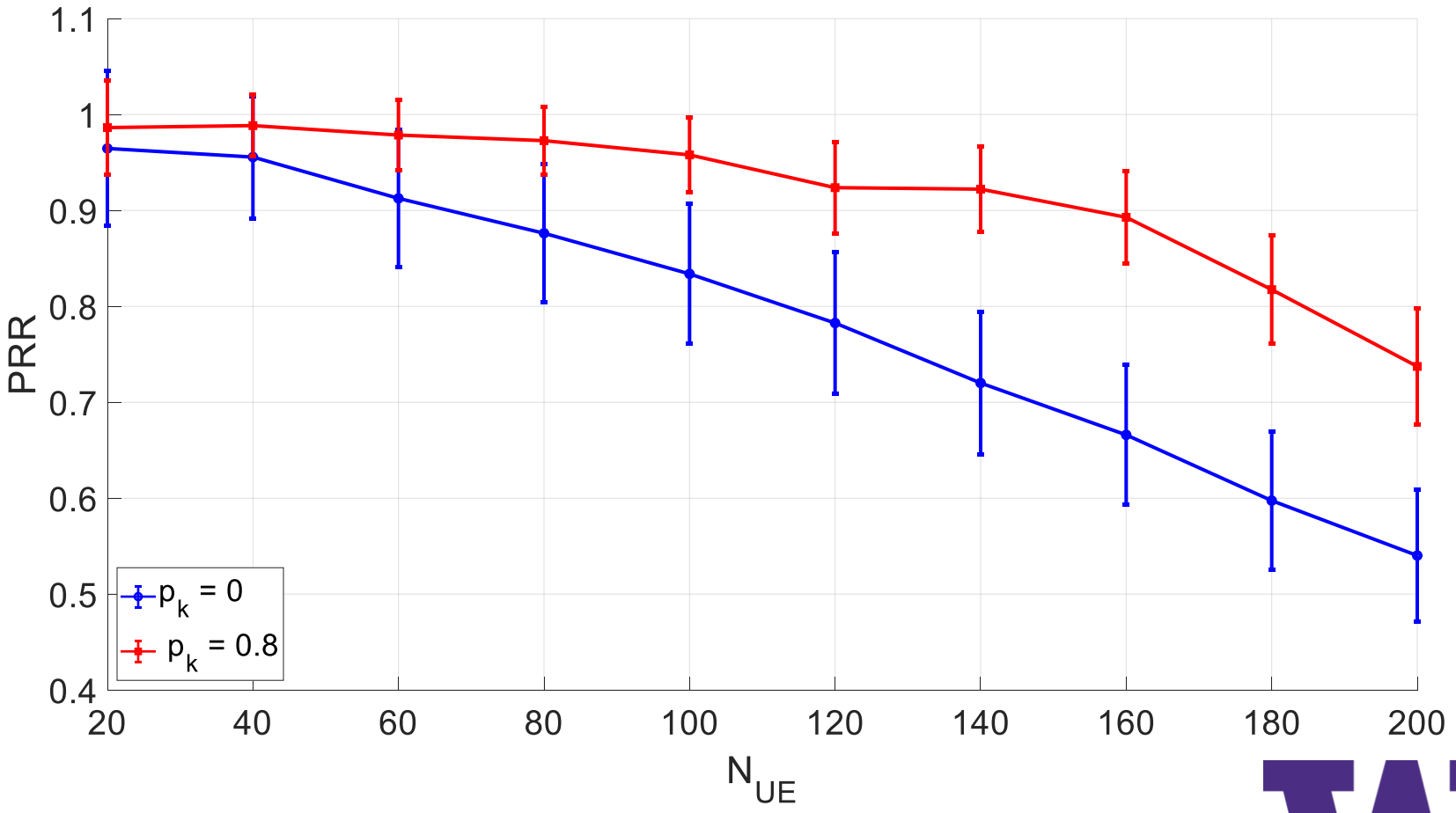
Section 2.4: Results for Simple Example Scenario: Effect of p_K and N_{UE} on p_{COL}: $N_{Se} = 1$



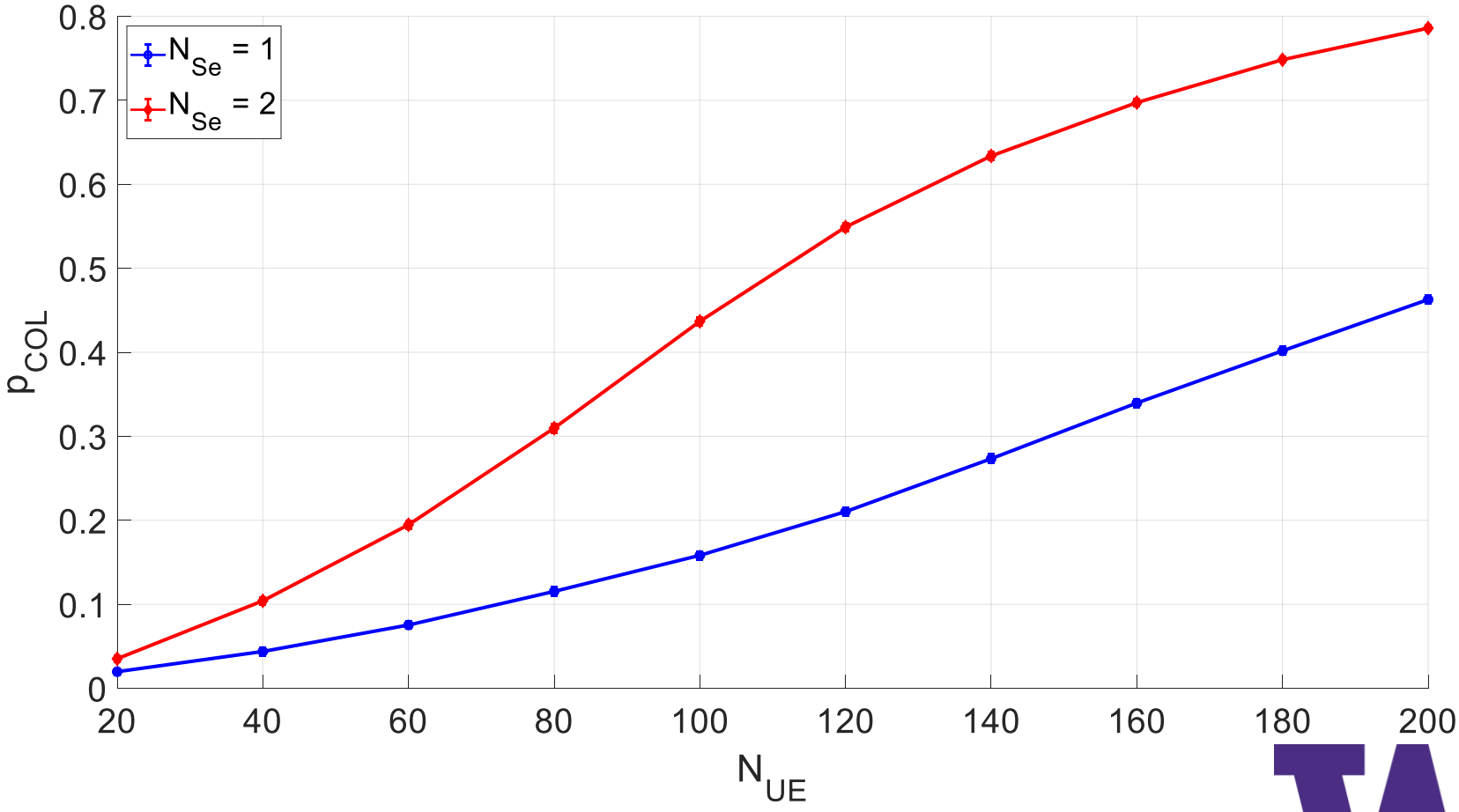
Section 2.4: Results for Simple Example Scenario: Effect of p_K and N_{UE} on $\bar{N}_a$: $N_{Se} = 1$



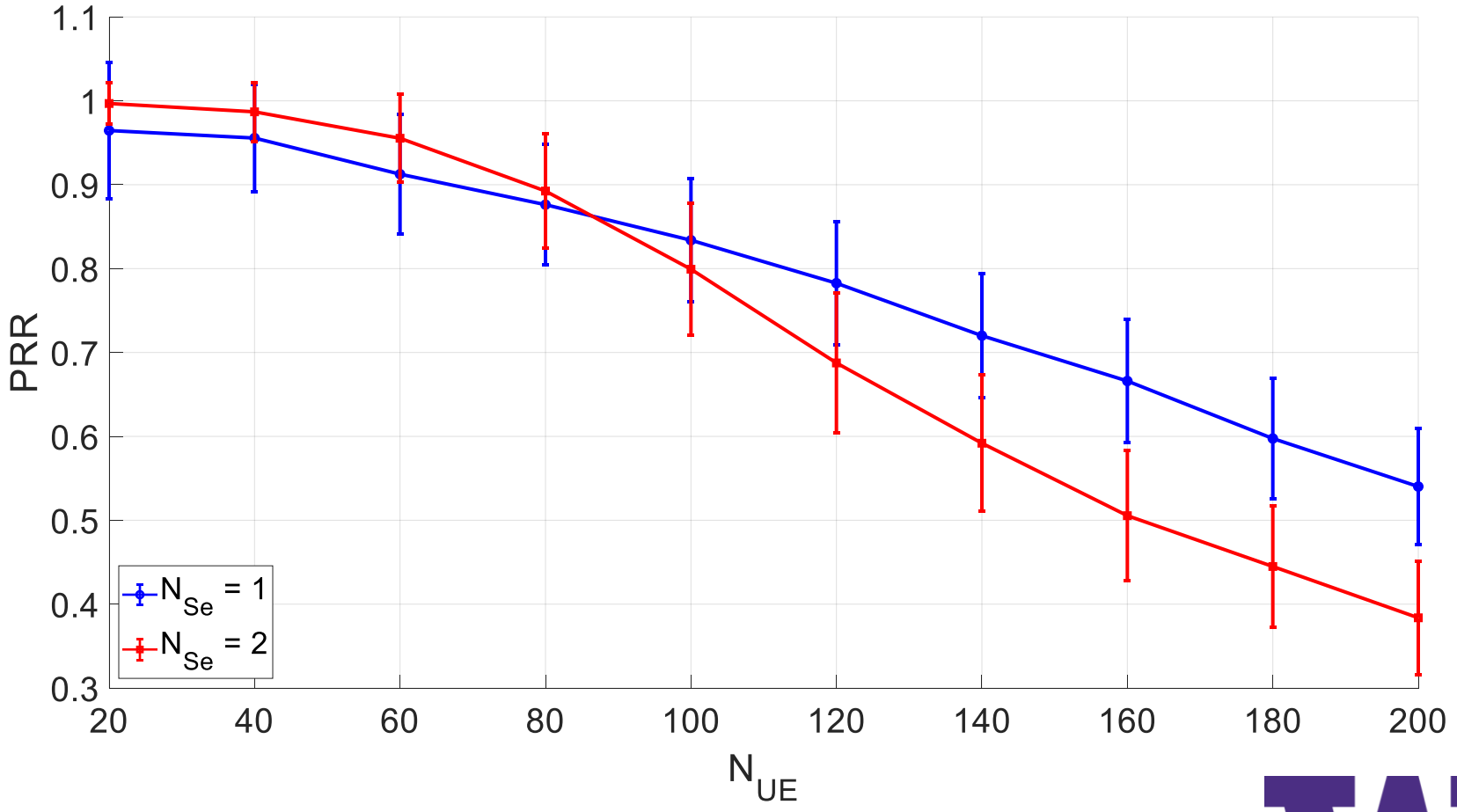
Section 2.4: Results for Simple Example Scenario: Effect of p_K and N_{UE} on PRR: $N_{Se}= 1$



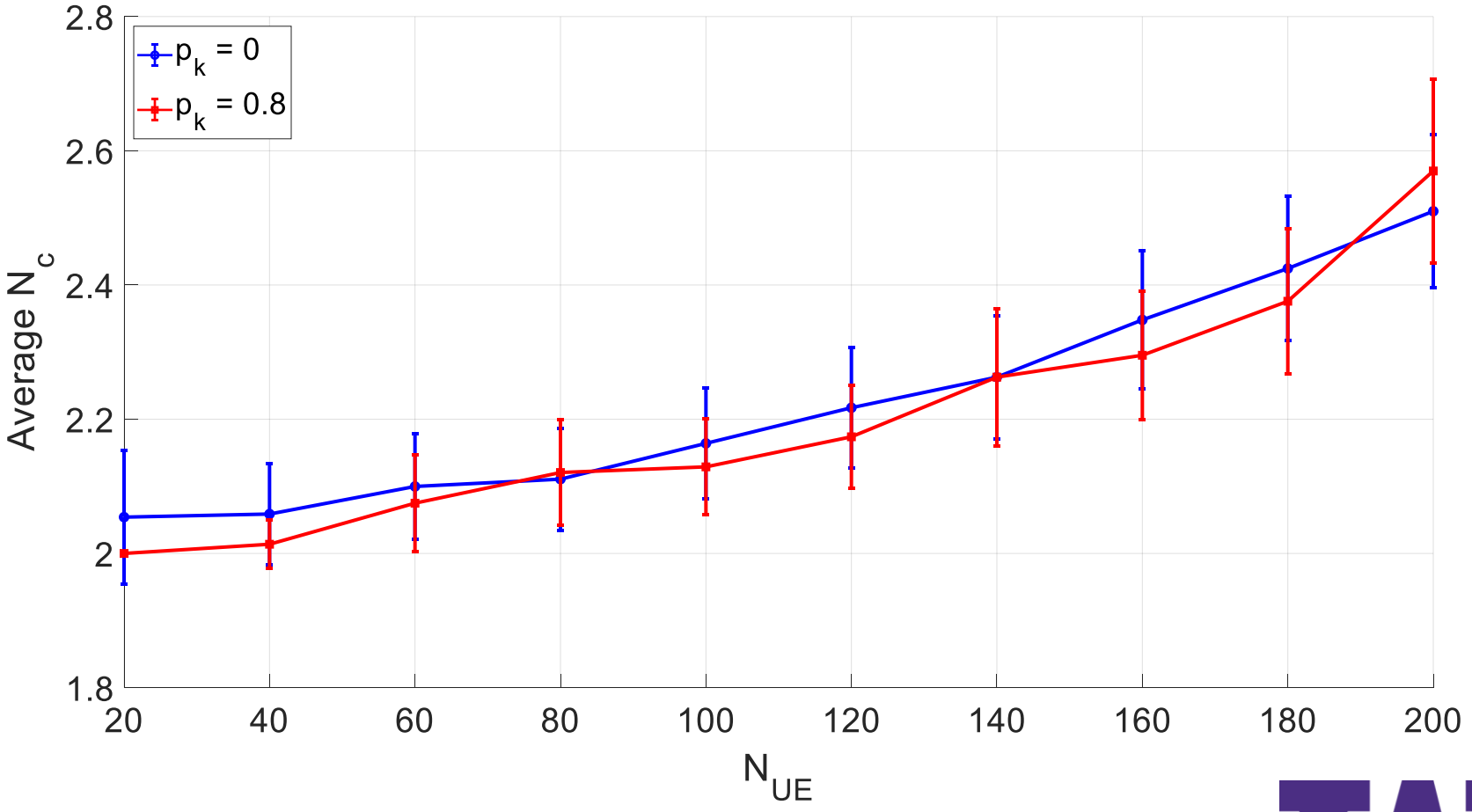
Section 2.4: Results for Simple Example Scenario: Effect of N_{Se} and N_{UE} on p_{COL}: $p_K = 0$



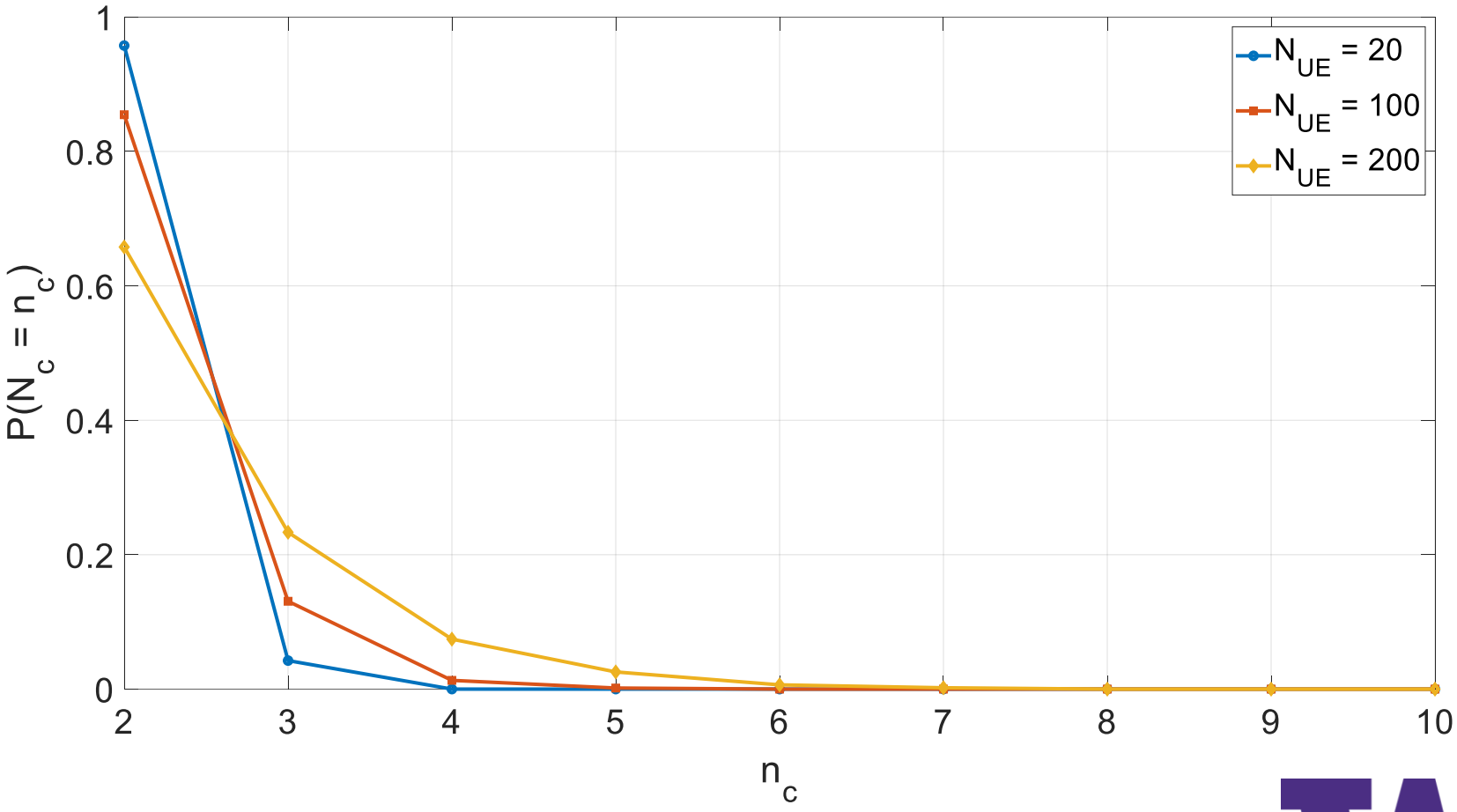
Section 2.4: Results for Simple Example Scenario: Effect of N_{Se} and N_{UE} on PRR: $p_K = 0$



Section 2.4: Results for Simple Example Scenario: Effect of p_K and N_{UE} on $\overline{N}_c$: $N_{Se}= 1$



Section 2.4: Results for Simple Example Scenario: Effect of N_{UE} on $\mathbf{P}(N_c)$: $N_{Se}= 1, p_k = 0$

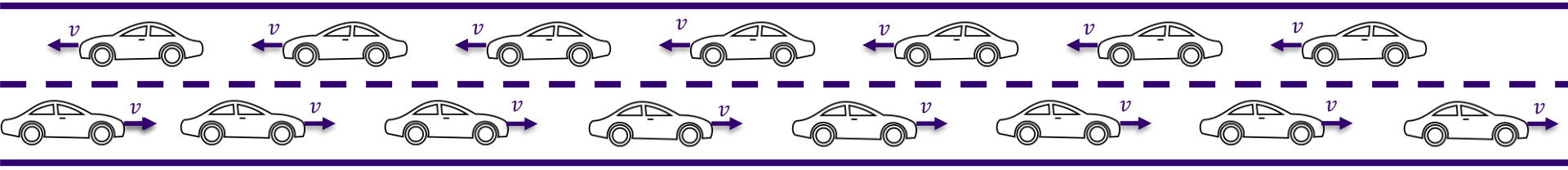


Section 2.4: Results for Simple Example Scenario: Key Takeaways from Model and ns-3

- If $N_{UE}N_{Se} << N_r$: $\uparrow N_{Se} \rightarrow\uparrow P_{COL} \rightarrow\uparrow PRR$
  - $PRR(N_{Se}) \propto 1 - P_{COL}(N_{Se})^{Nse}$: $P_{COL}(1) > P_{COL}(N_{Se})^{Nse}$
    - Example: $P_{COL}(1)$ = 0.3; $P_{COL}(2)$ = 0.5 → $P_{COL}(2)^2$ = 0.25
  - But if $N_{UE}N_{Se} \approx N_r$ (or $N_{UE}N_{Se} > N_r$): $\uparrow N_{Se} \rightarrow\uparrow P_{COL} \rightarrow\downarrow PRR$
    - $P_{COL}(1) < P_{COL}(N_{Se})^{Nse}$ in the (over)saturated condition;
    - Example: $P_{COL}(1)$ = 0.6; $P_{COL}(2)$ = 0.8 → $P_{COL}(2)^2$ = 0.64
- $\uparrow p_K \rightarrow\downarrow P_{COL} \rightarrow\uparrow PRR$.
  - $\uparrow p_K$ → the behavior of the transmitter becomes more predictable (lower reselection frequency).
  - No tradeoff in a fully connected network but,
    - If UEs are entering and exiting the transmission range, there is an optimal p_K for a given rate of change (represented by velocity).
- Collisions always happen consecutively (lasts 1 to16 T_{RRI} → 100 to1600 milliseconds) in SPS
  - The time-sensitive applications may suffer.

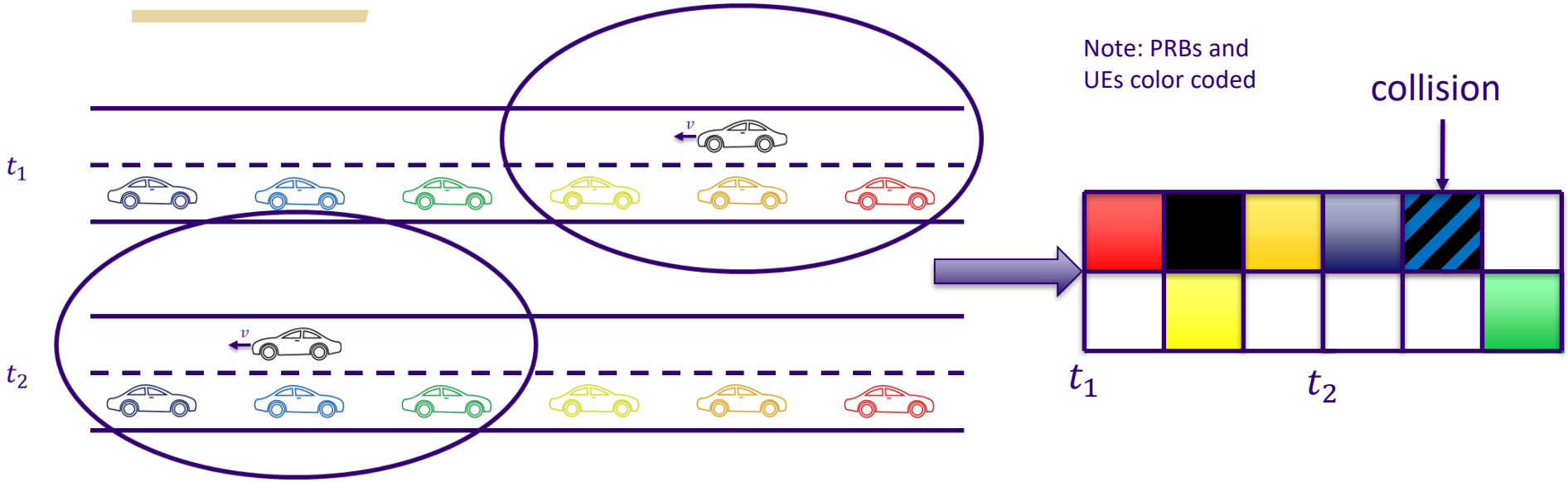
Section 3:
Mobile Scenario

Section 3: Mobile Scenario: Physical Scenario Refresher



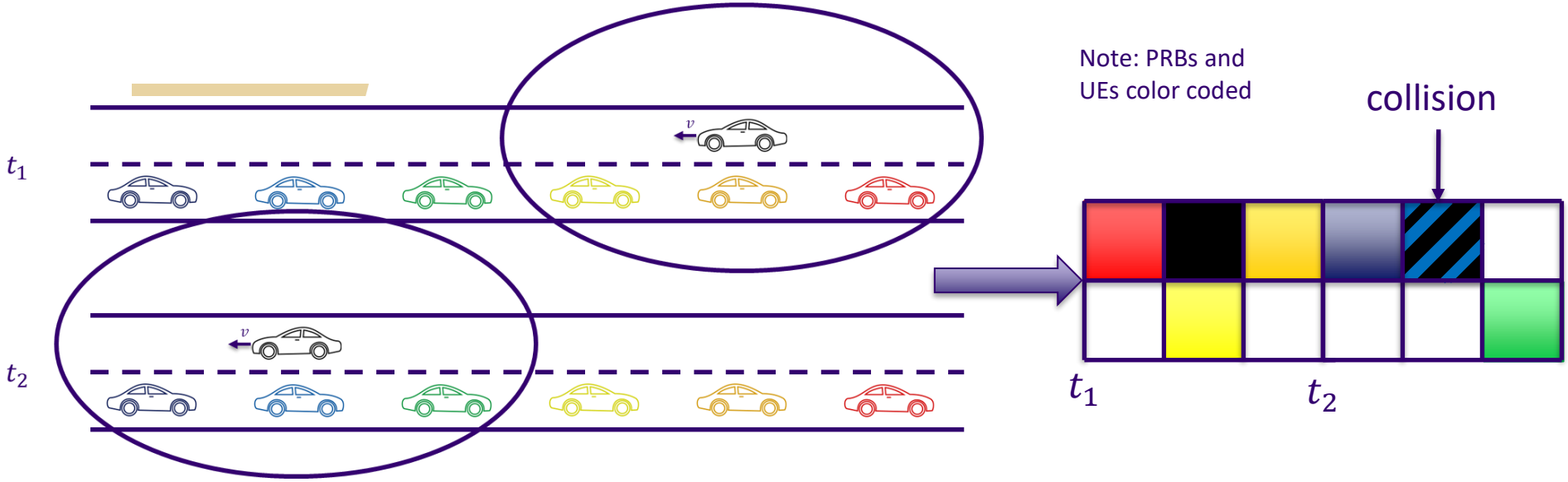
- UEs distributed uniformly in a line with density ρ_{UE} UE/m
- 2 lanes
  - UEs move at velocity v in opposite directions
- PHY collision model
  - If two UEs occupy the same PRB, one may be decoded based on SINR
- Parameters under test (PRR as a function of):
  - N_{Se} vs ρ_{UE}
  - N_{Se} vs v
  - p_k vs ρ_{UE}
  - p_k vs v

Section 3: Mobile Scenario: Motivation for Mobility



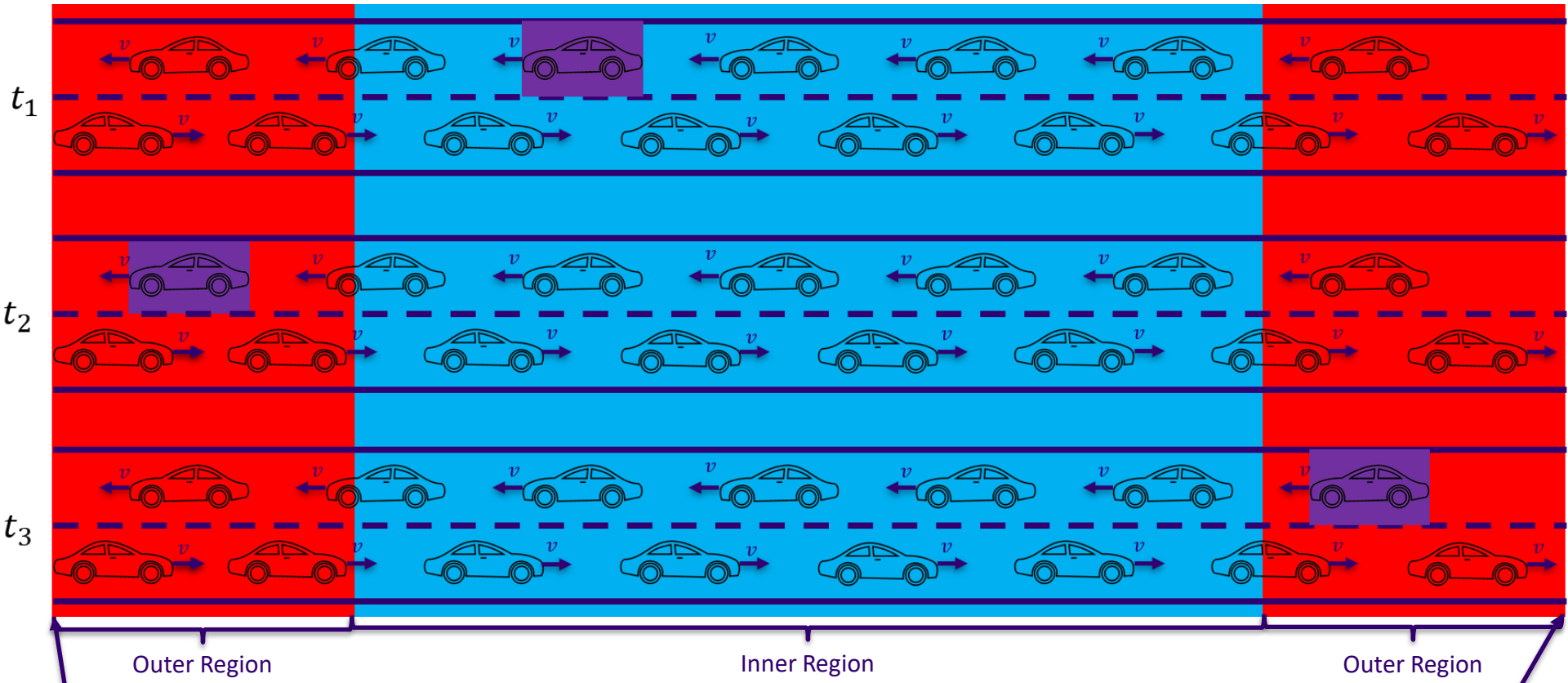
- Simple Model (MAC Collision) has fixed topology, full connectivity
  - No hidden nodes, no new nodes
- Goal of mobility:
  - Introduce a mechanism represent topology changes over time
- Topology affects network state:
  - PRB choices made with info from topology during reselection
  - Topology changes → collisions with unknown (during reselection) UEs

Section 3: Mobile Scenario: Expected Effects of Mobility



- In all cases:
  - From fully connected network: reselect less often (higher p_k) = more predictable = higher PRR
  - Faster topology changes → reselect more often
    - Previous reselection (p_k) ignores new information, suboptimal PRB choice for current situation
  - Key Tradeoff – Predictability vs Topology Rate of Change
    - At what point (new p_k) are gains from being more predictable overcome by losses from not reacting to new information?
- If topology changes number of UE in decode region
  - Different N_{Se} will be preferred - Need algorithm to tune over time
  - Outside the scope of this tutorial

Section 3: Mobile Scenario: Mobility Model Specifics



- Uses constant-velocity-mobility-model as base
  - UEs move in one direction, reach predefined boundary, then move instantaneously to other boundary
- Regions (outer and inner) are used for data processing
  - UEs in outer regions lack sufficient neighbors, ignored when computing PRR

Section 3: Mobile Scenario: ns-3 New Mobility Model in Code

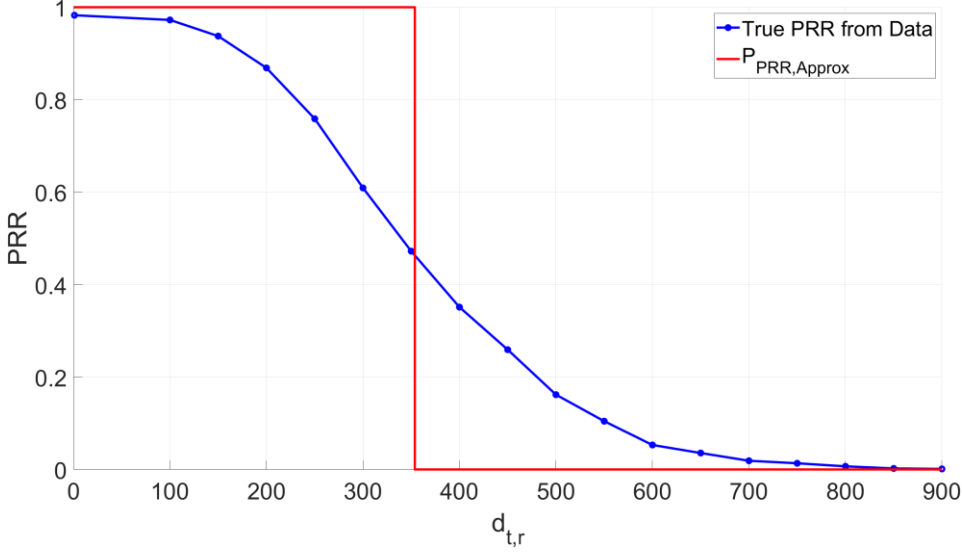
```
void
ConstantVelocityHelper::UpdateWithWrappingBounds (const Rectangle &bounds) const
{
  NS_LOG_FUNCTION (this << bounds);
  Update ();
  if (m_position.x > bounds.xMax)
  {
    m_position.x = bounds.xMin;
  }
  else if (m_position.x < bounds.xMin)
  {
    m_position.x = bounds.xMax;
  }
  else if (m_position.y > bounds.yMax)
  {
    m_position.y = bounds.yMin;
  }
  else if (m_position.y < bounds.yMin)
  {
    m_position.y = bounds.yMax;
  }
}
```

- New mobility model:
  - Constant-velocity-mobility-model-bounded
- Same as Constant-velocity-mobility-model but uses new update function
- Update function checks bounds in mobility model, moves UE to other side if bounds exceeded

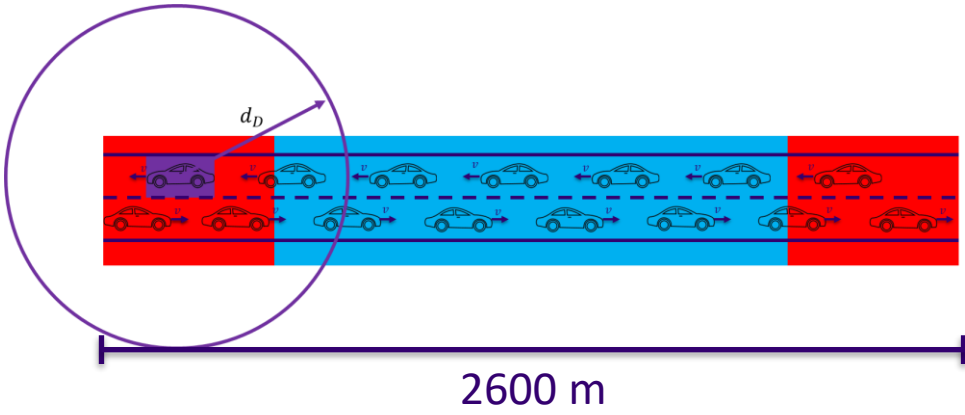
```
#include "constant-velocity-mobility-model-bounded.h"
#include "ns3/simulator.h"

namespace ns3 {

NS_OBJECT_ENSURE_REGISTERED (ConstantVelocityMobilityModelBounded);

TypeId ConstantVelocityMobilityModelBounded::GetTypeId (void)
{
  static TypeId tid = TypeId ("ns3::ConstantVelocityMobilityModelBounded")
    .SetParent<MobilityModel> ()
    .SetGroupName ("Mobility")
    .AddConstructor<ConstantVelocityMobilityModelBounded> ()
    .AddAttribute ("Bounds", "The 2d bounding area",
                   RectangleValue (Rectangle (0, 2600, 0, 10)),
                   MakeRectangleAccessor (&ConstantVelocityMobilityModelBounded::m_bounds),
                   MakeRectangleChecker ())
    ;
  return tid;
}

ConstantVelocityMobilityModelBounded::ConstantVelocityMobilityModelBounded ()
{
}

ConstantVelocityMobilityModelBounded::~ConstantVelocityMobilityModelBounded ()
{
}

void
ConstantVelocityMobilityModelBounded::SetVelocity (const Vector &speed)
{
  m_helper.Update ();
  m_helper.SetVelocity (speed);
  m_helper.Unpause ();
  NotifyCourseChange ();
}


Vector
ConstantVelocityMobilityModelBounded::DoGetPosition (void) const
{
  m_helper.UpdateWithWrappingBounds (m_bounds);
  return m_helper.GetCurrentPosition ();
}
void
ConstantVelocityMobilityModelBounded::DoSetPosition (const Vector &position)
{
  m_helper.SetPosition (position);
  NotifyCourseChange ();
}
Vector
ConstantVelocityMobilityModelBounded::DoGetVelocity (void) const
{
  return m_helper.GetVelocity ();
}

} // namespace ns3
```

Section 3: Mobile Scenario: PHY Layer Model Specifics

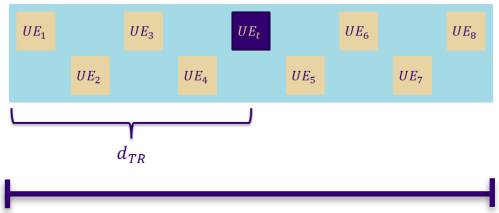
- Ns-3 uses BLER curves from [2] by Lagen et al. for PSSCH/PSCCH
  - Path loss model – 3GPP V2V Highway
- In absence of collisions – UEs cannot decode packets beyond ≈800 m
  - Outer region is 800 m from simulation boundary
- For approximation introduce simplified 0/1 decode model (red curve)
  - $P_{PRR,approx} = \begin{cases} 1; d_{t,r} < d_D \\ 0; d_{t,r} > d_D \end{cases}$
  - $d_D = \int PRR(d)dd \rightarrow \mathrm{d_D} \approx 350m$
- $P_{PRR,approx}$ used to calculate N_D - number of "decodable" UE
  - $N_D = \left\lfloor \frac{2d_D}{d_V} \right\rfloor \cdot N_{Lanes}$



Mobility Scenario

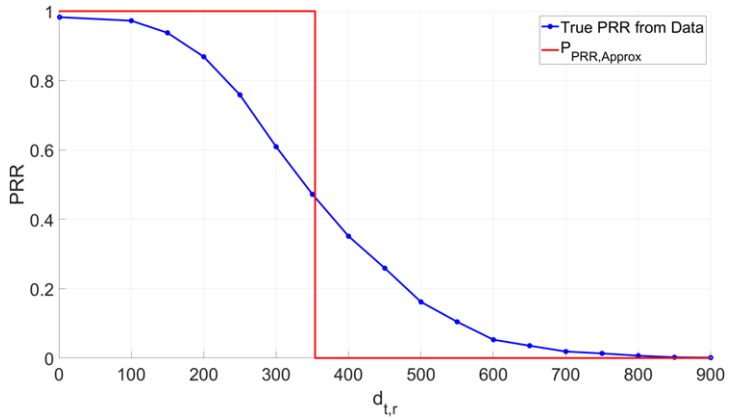


Fully Connected Scenario



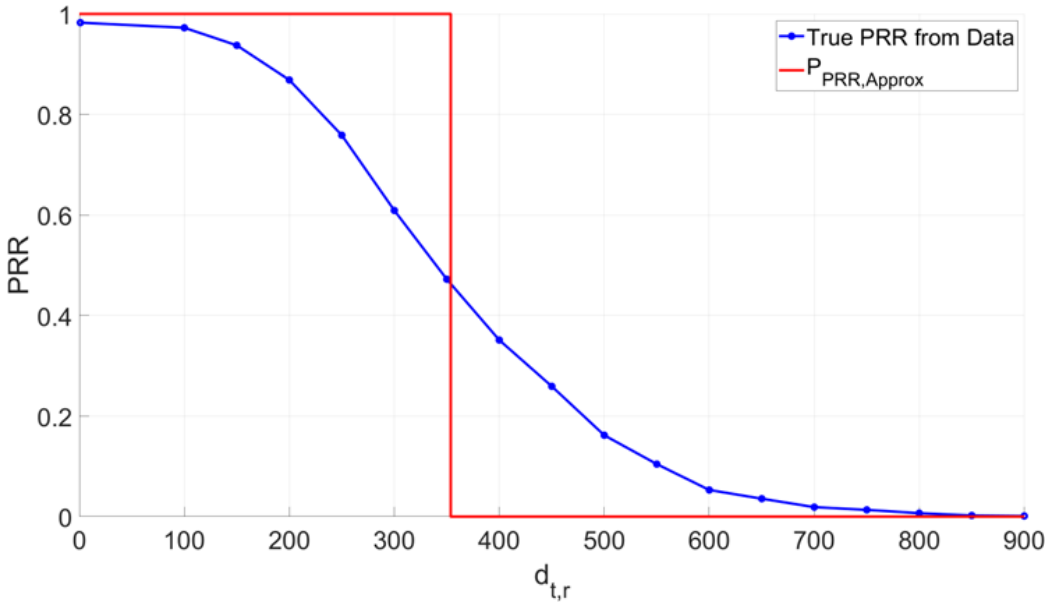
Section 3: Mobile Scenario: PHY Layer Model Specifics 2

- High Density = Oversaturated Channel - $\frac{N_{Se}N_D}{N_r} > 1.25$
  - $\rho_{UE} = \frac{1}{20}\frac{UE}{m \cdot Lane} \rightarrow d_V = 20m \rightarrow N_D = \left\lfloor \frac{2d_D}{d_V} \right\rfloor \cdot N_{Lanes} = \left\lfloor \frac{2(350)}{20} \right\rfloor 2 = 70$
- Medium Density = saturated Channel - $.5 < \frac{N_{Se}N_D}{N_r} < 1.25$
  - $\rho_{UE} = \frac{1}{40}\frac{UE}{m \cdot Lane} \rightarrow d_V = 40m \rightarrow N_D = \left\lfloor \frac{2d_D}{d_V} \right\rfloor \cdot N_{Lanes} = \left\lfloor \frac{2(350)}{40} \right\rfloor 2 = 34$
- Low Density = Undersaturated Channel - $\frac{N_{Se}N_D}{N_r} < .5$
  - $\rho_{UE} = \frac{1}{100}\frac{UE}{m \cdot Lane} \rightarrow d_V = 100m \rightarrow N_D = \left\lfloor \frac{2d_D}{d_V} \right\rfloor \cdot N_{Lanes} = \left\lfloor \frac{2(350)}{100} \right\rfloor 2 = 14$

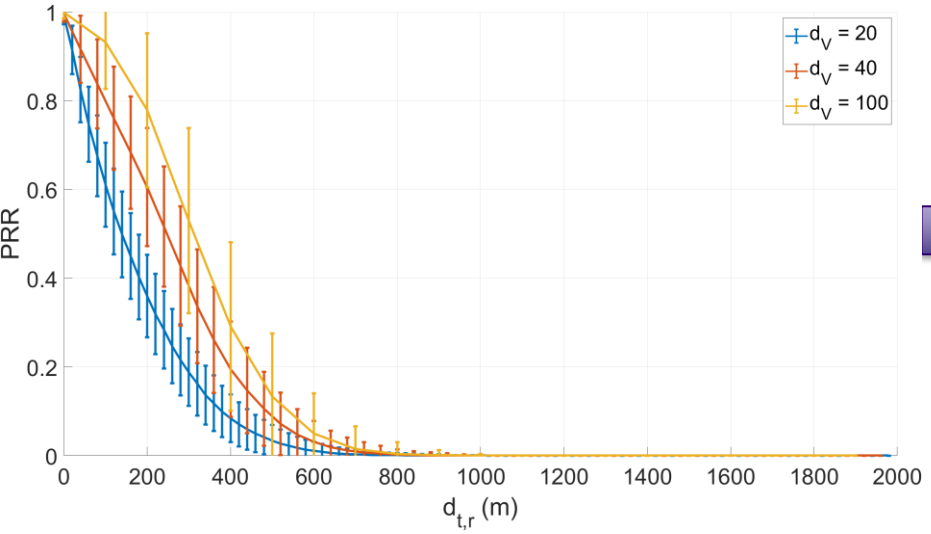


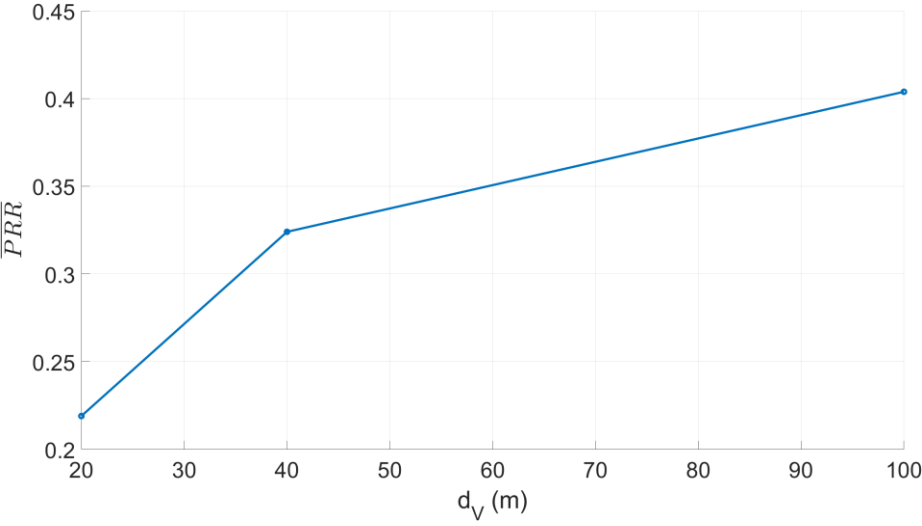
Section 3: Mobile Scenario: PHY Layer Model Specifics 3

- Speed defined in terms of new UE after $E[R_c]T_{RRI} = 1$ seconds
  - $\frac{Number\ of\ New\ UE\ After\ 1\ Seconds}{N_D} = \frac{(vN_D/(2d_D))}{N_D} = \frac{v}{2d_D}$
  - $0 \leq \frac{v}{2d_D} \leq .5$
- High speed = $\frac{v}{2d_D} \geq .25$
  - $v = 2 * .35 * 350 \approx 250\ m/s$
- Low Speed = $0 \leq \frac{v}{2d_D} < .25$
  - $v = 2 * .175 * 350 \approx 125\ m/s$
- Static = $\frac{v}{2d_D} = 0$
  - $v = 0\ m/s$



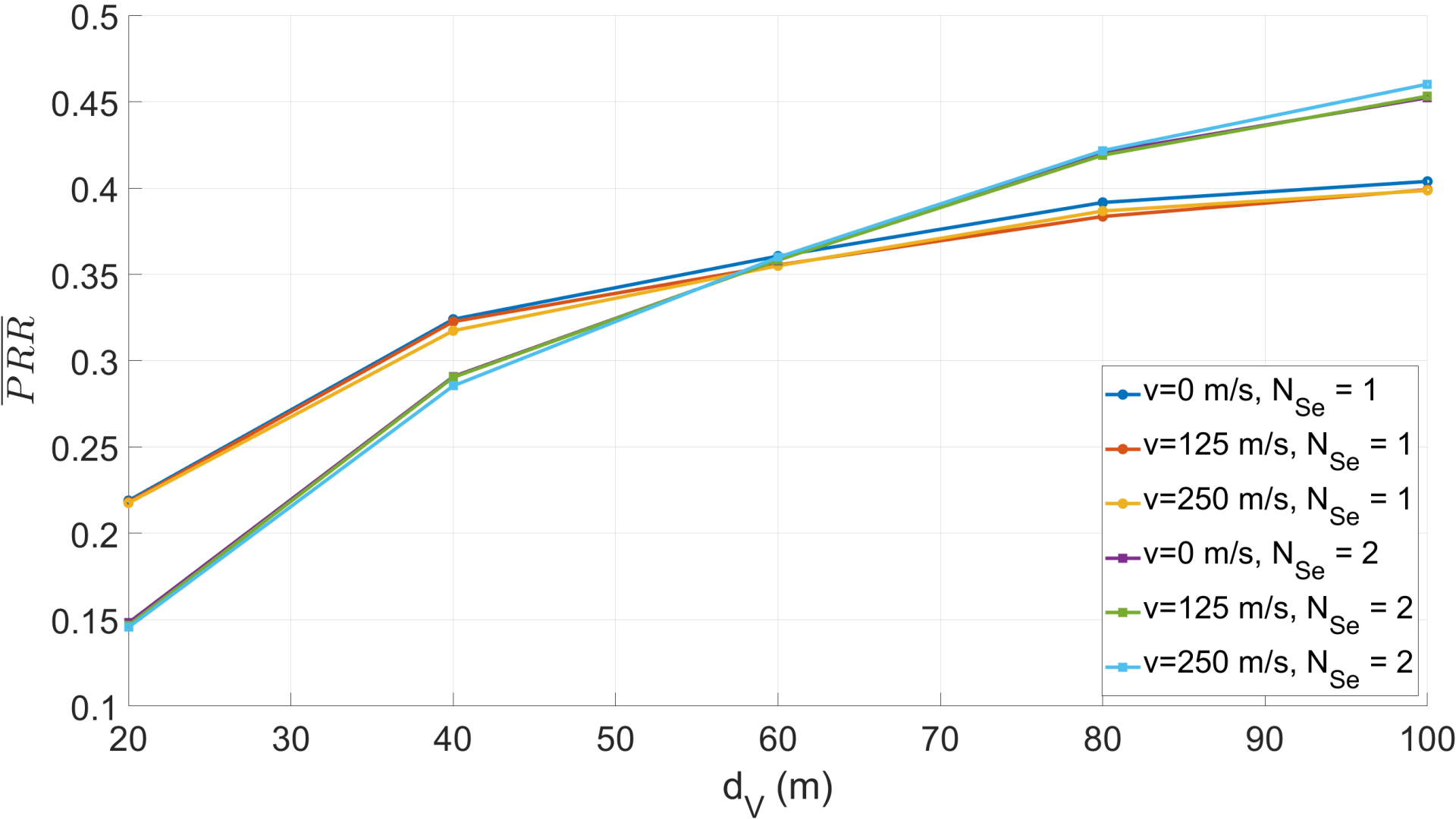
Section 3.1: Results for Mobile Scenario: Distance Dependance Necessitates New Metric



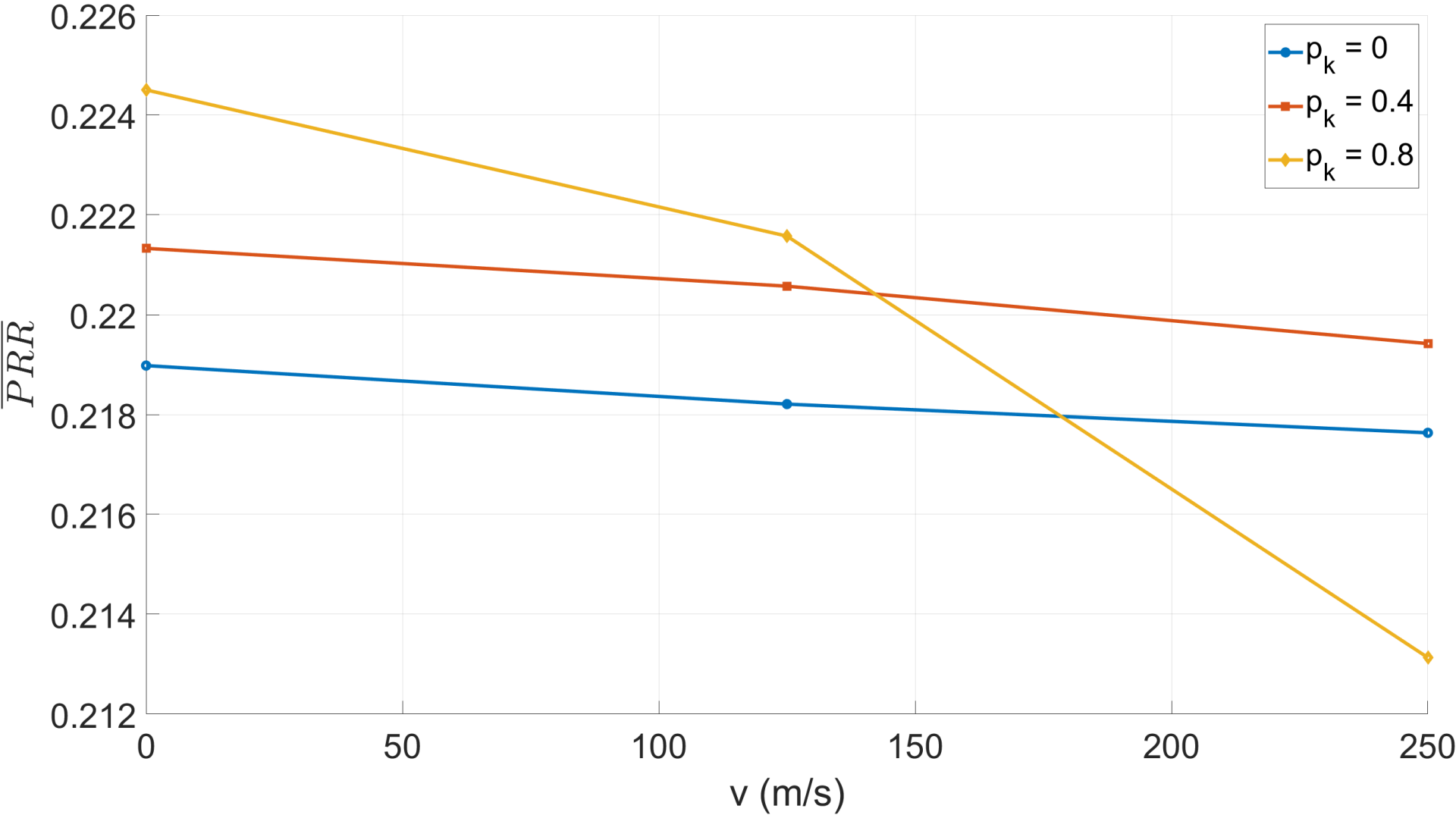


- PRR now a function of $d_{t,r}$ - inter tx/rx distance in m
  - Can't average like fully connected case
- Define:
  - $\overline{PRR} = \frac{1}{x}\int_{d=0}^{x} PRR(d)dd$
  - x not important so long as $PRR(x) \approx 0$
    - For rest of slides $x = 800$

Section 3.1: Results for Mobile Scenario: Effect of N_{Se} and v on $\overline{PRR}$, $d_V = 20, p_k = 0$



Section 3.1: Results for Mobile Scenario: Effect of p_k and v on $\overline{PRR}$, $d_V = 20, N_{Se} = 1$



Section 3.1: Results for Mobile Scenario: Summary of Major Findings

Tested Parameter	Finding for Fully Connected Network	Finding for Mobile Network
UE density (d_V and N_{UE})	Increasing UE density reduces the PRR due to an increase in collisions. (Slide 39)	Increasing UE density reduces the PRR due to an increase in collisions. (Slide 55)
Speed (v m/s)	(untested)	Speed alone is not a major contributor to PRR (Slide 55). Differences in PRR as a function of speed are a function of BLER model.
Reselection Probability ($1 - p_k$)	Reducing reselection probability improves PRR in all tested scenarios. This occurs because it makes the behavior of the transmitting UE more predictable to other UEs, thus reducing collisions. (Slide 39)	At static speeds $p_k = .8$ maximizes $\overline{PRR}$ (same as fully connected network). The effect is less pronounced than fully connected because the UEs at the edge of a reference UEs decode range are only considered during reselection sometimes, creating a natural, but small, topology change (slide 56) At low and high speeds $p_k = .4$ maximizes $\overline{PRR}$ because rapid topology changes introduces losses from not reselecting, like in slide 46, changes to physical topology can cause collisions. (slide 56)
Number of Blind Retransmissions (N_{Se})	Increasing the number of blind retransmissions improves PRR at low UE densities due to repetition coding gains, at high UE densities this improvement is negated by an increase in collisions (Slide 41)	Increasing the number of blind retransmissions improves PRR at low UE densities due to repetition coding gains, at high UE densities this improvement is negated by an increase in collisions (Slide 55)

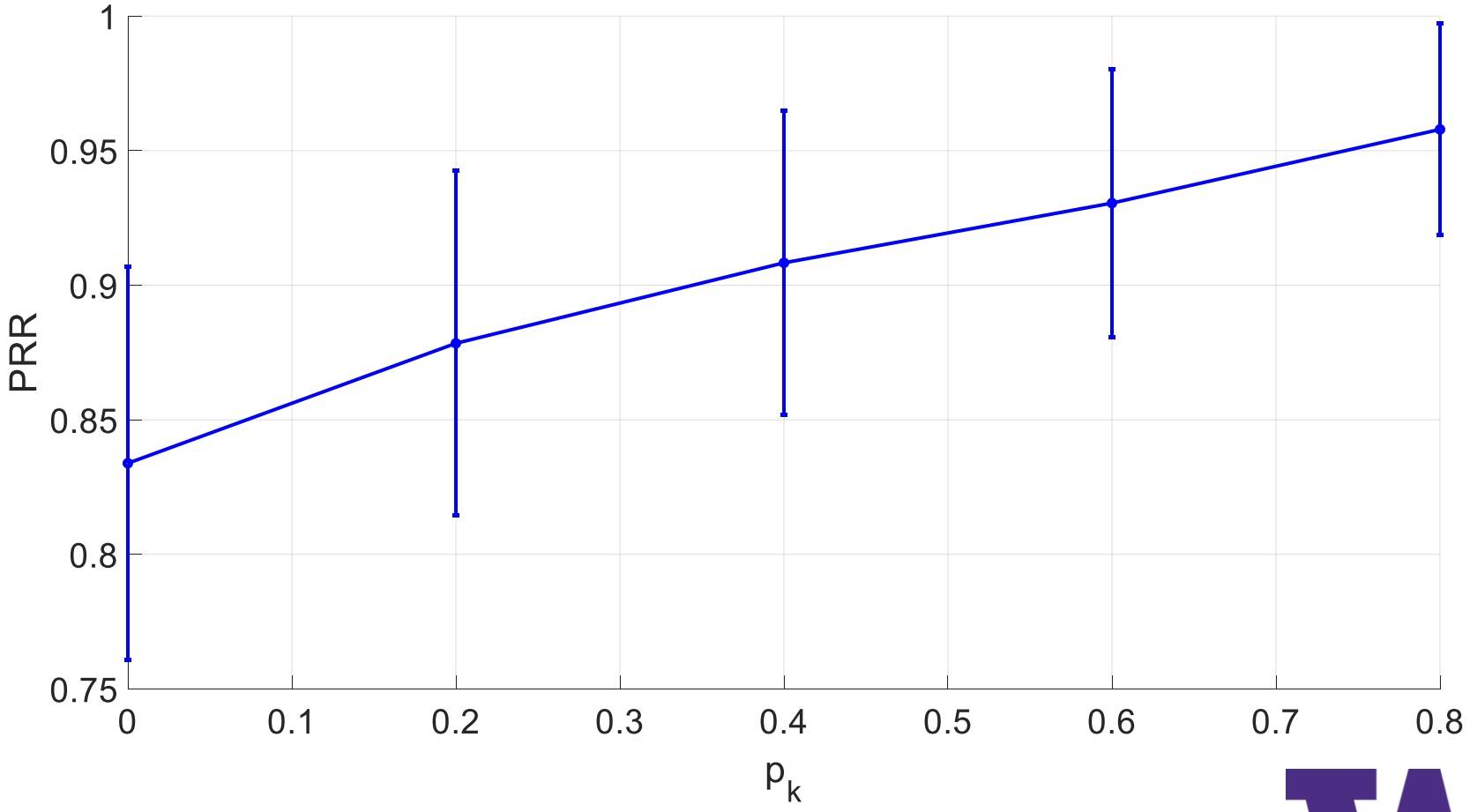
All findings assume a fixed N_r and T_{RRI}

Citations

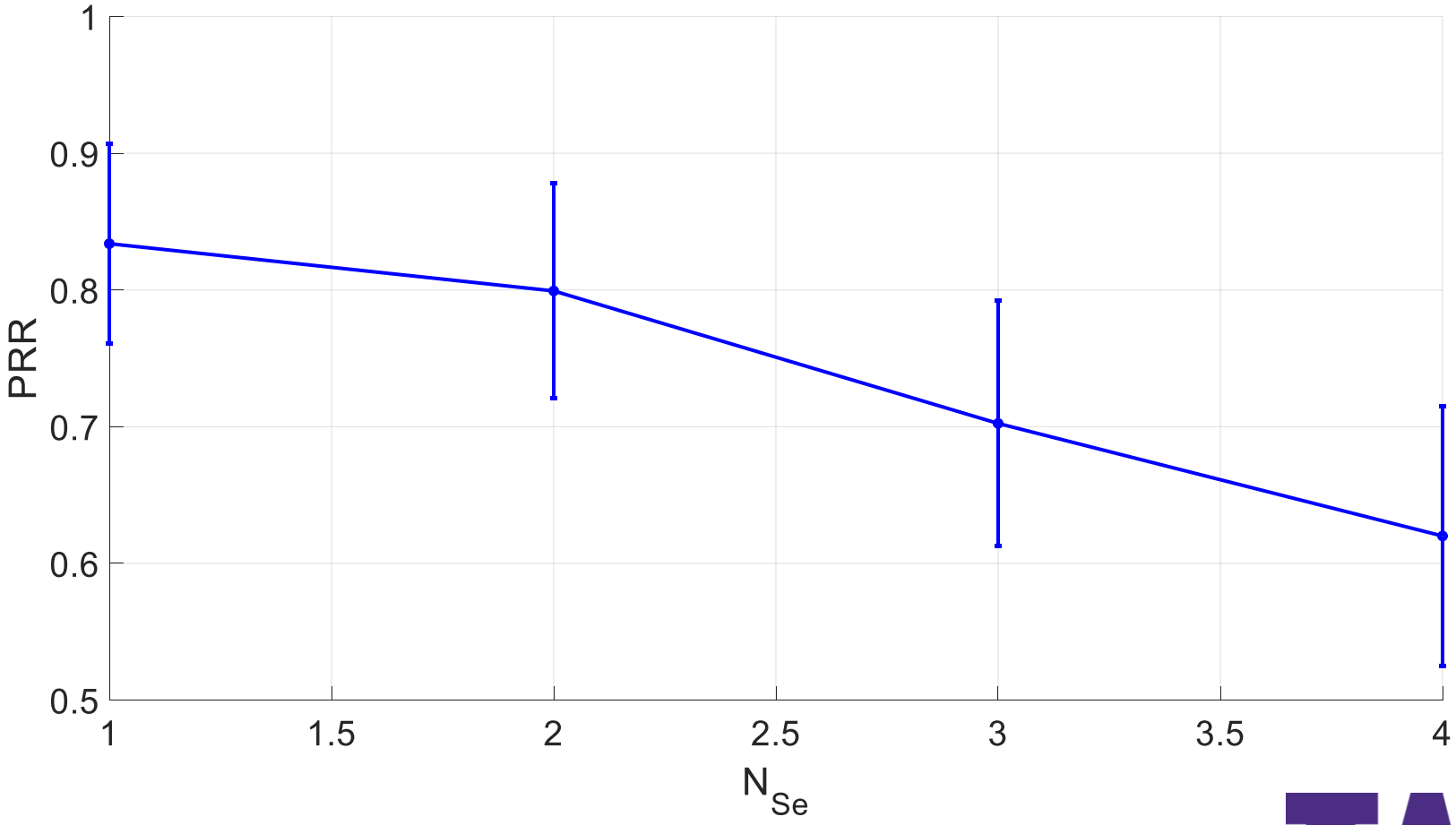
- [1]: 5GAA. White Paper C-V2X Use Cases: Methodology, Examples and Service Level Requirements. Accessed: Jun. 2019. [Online]. Available: https://5gaa.org/wpcontent/uploads/2019/07/5GAA_191906_WP_CV2X_UCs_v1.pdf
- [2]: S. Lagen, K. Wanuga, H. Elkotby, S. Goyal, N. Patriciello, and L. Giupponi, “New Radio Physical Layer Abstraction for System-Level Simulations of 5G Networks,” in ICC 2020 - 2020 IEEE International Conference on Communications (ICC), pp. 1–7, 2020.

Appendix A:
Additional Simple Scenario Results

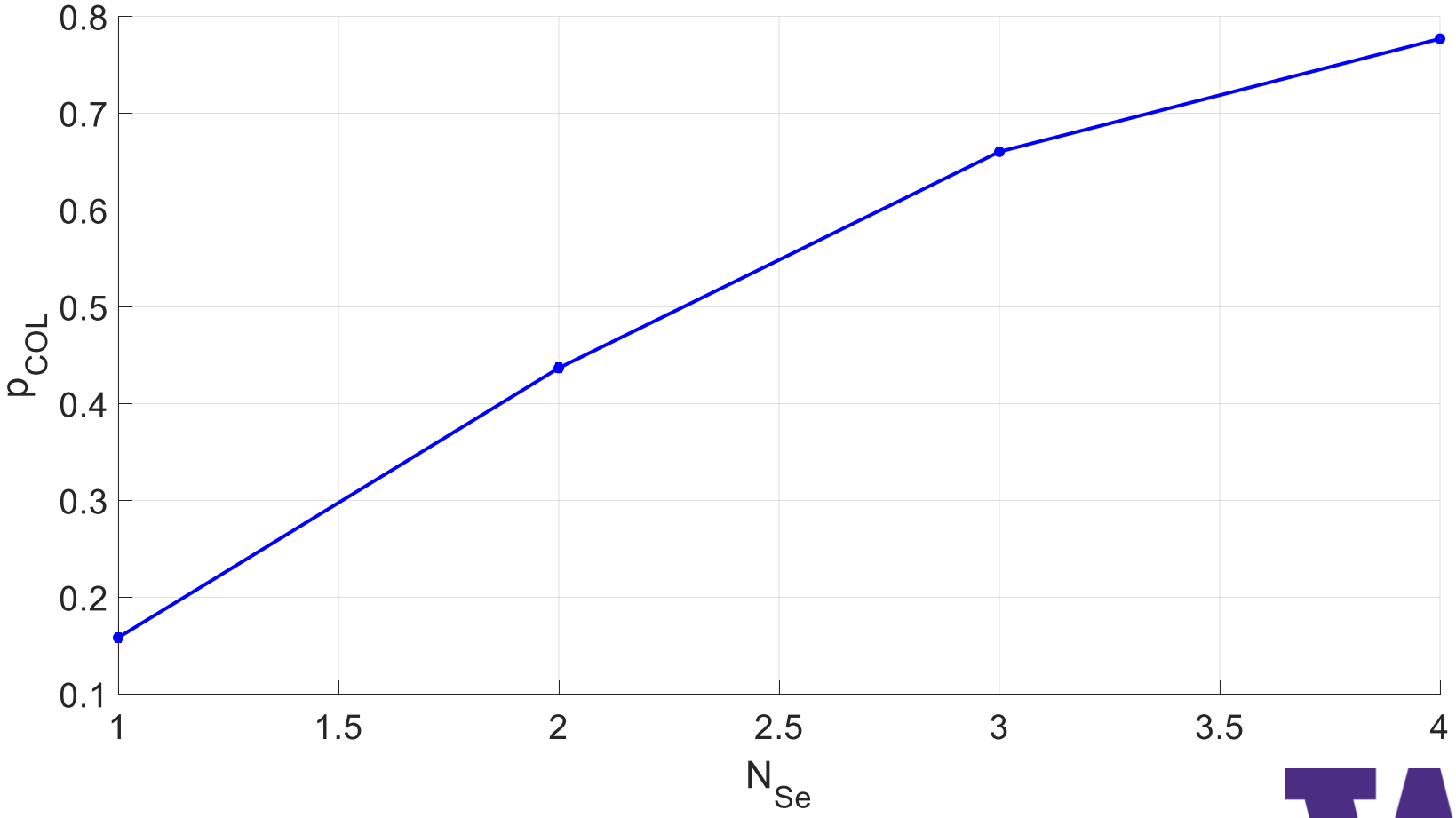
Appendix A: Additional Simple Scenario Results: Effect of p_K on PRR: $N_{UE} = 100$, $N_{Se} = 1$



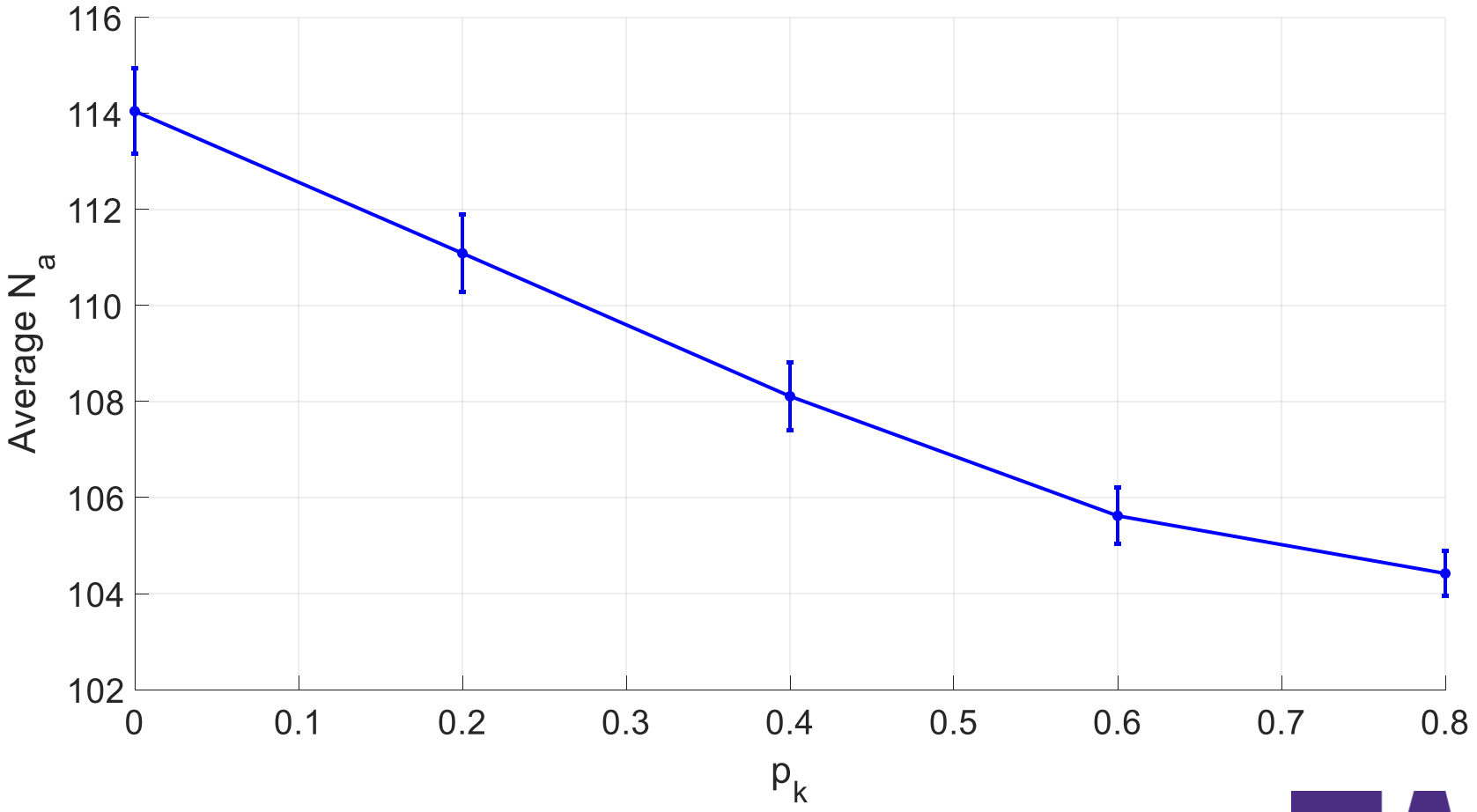
Appendix A: Additional Simple Scenario Results: Effect of N_{Se} on $PRR: p_K = 0, N_{UE} = 100$



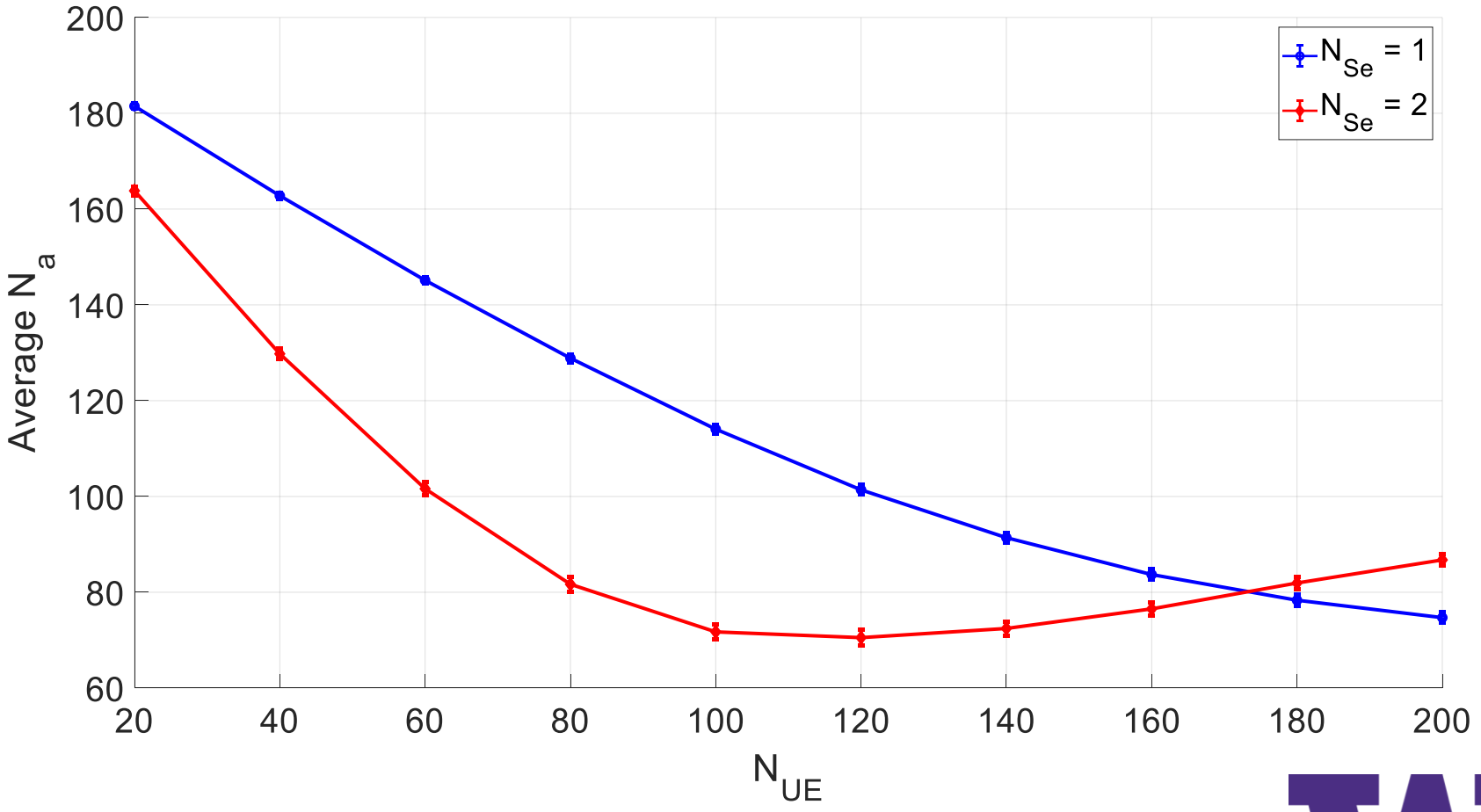
Appendix A: Additional Simple Scenario Results: Effect of N_{Se} on p_{COL}: $p_K = 0, N_{UE} = 100$



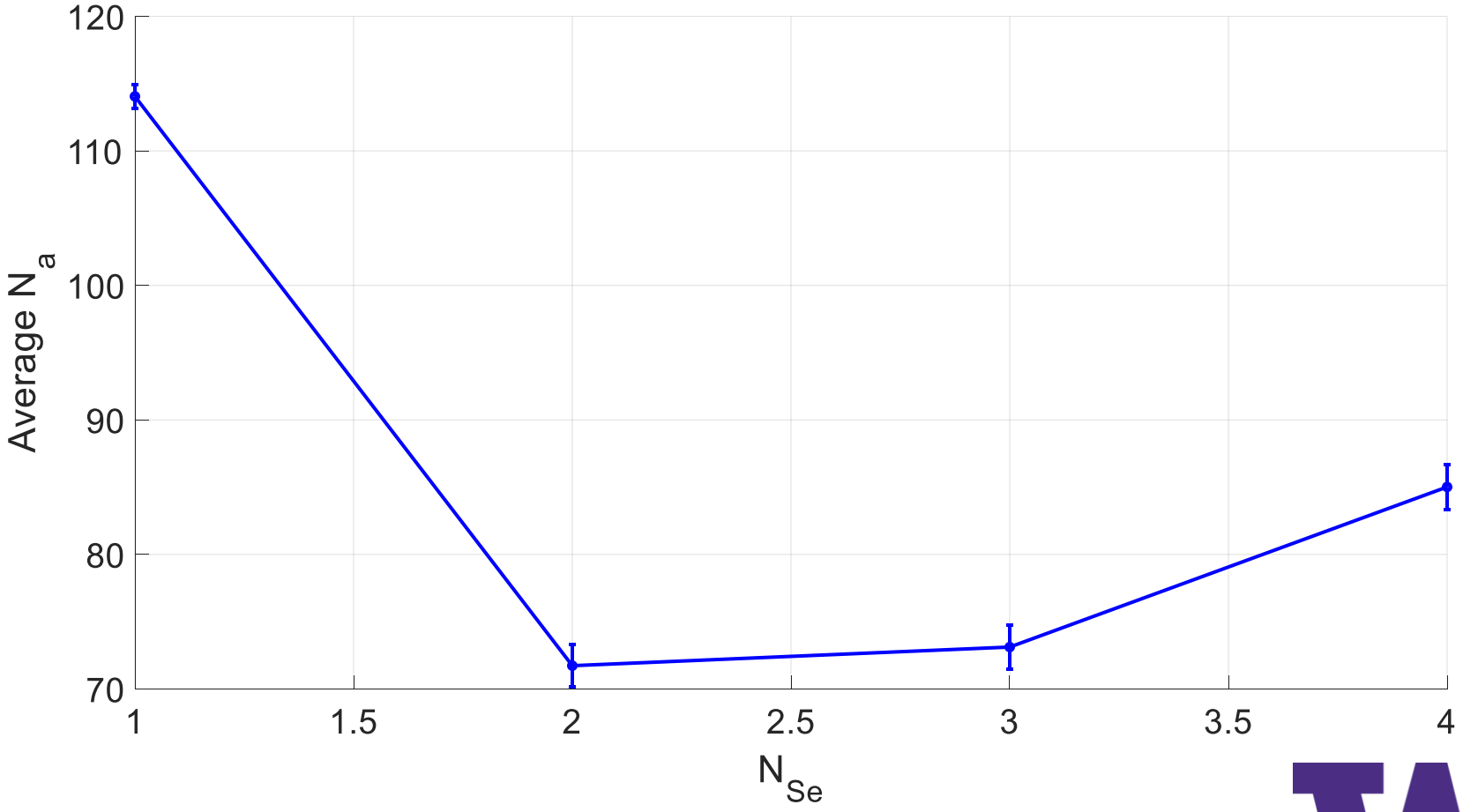
Appendix A: Additional Simple Scenario Results: Effect of p_K on $\overline{N}_a$: $N_{Se} = 1, N_{UE} = 100$



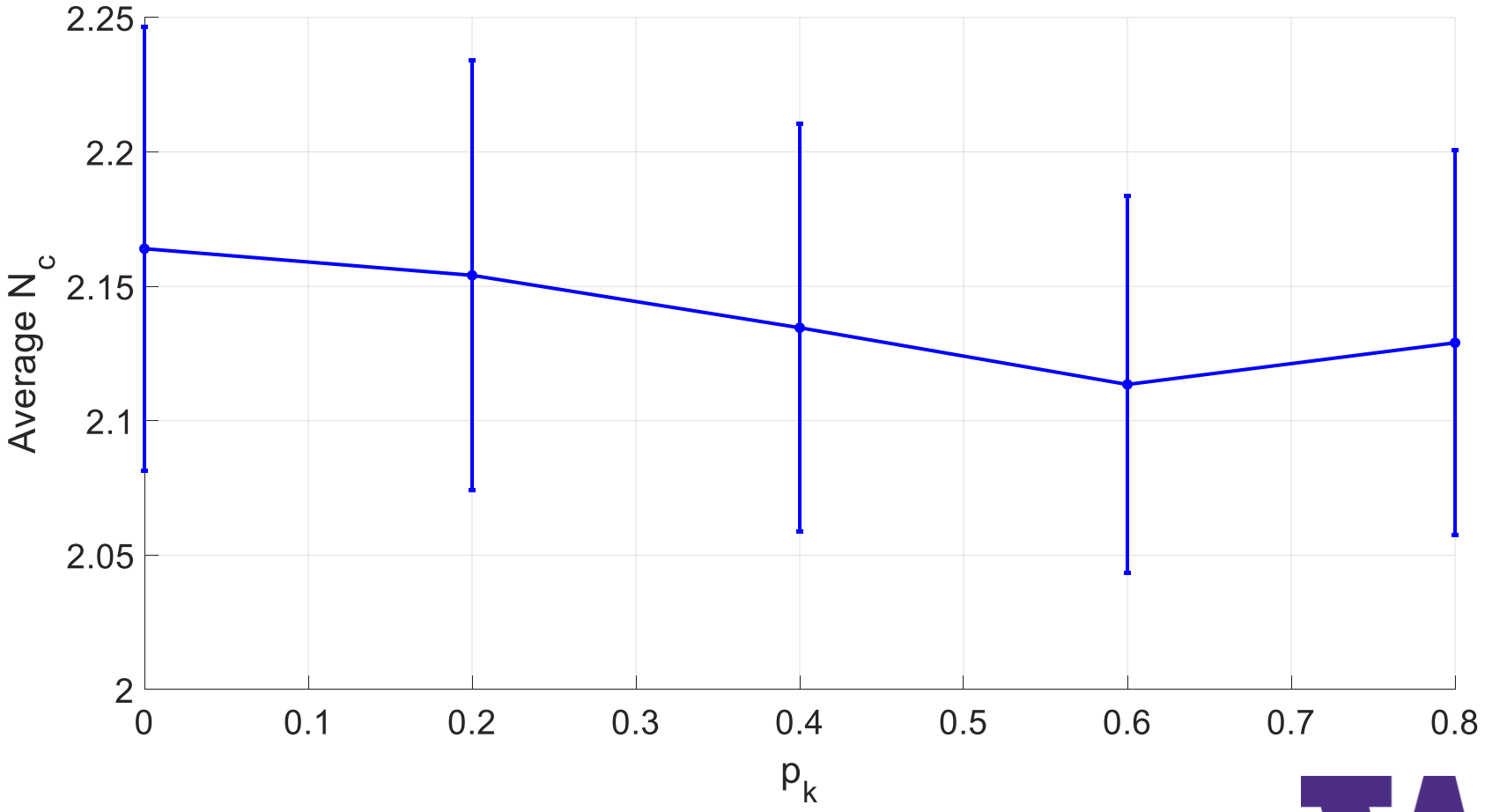
Appendix A: Additional Simple Scenario Results: Effect of N_{Se} and N_{UE} on $\bar{N}_a$: $p_K = 0$



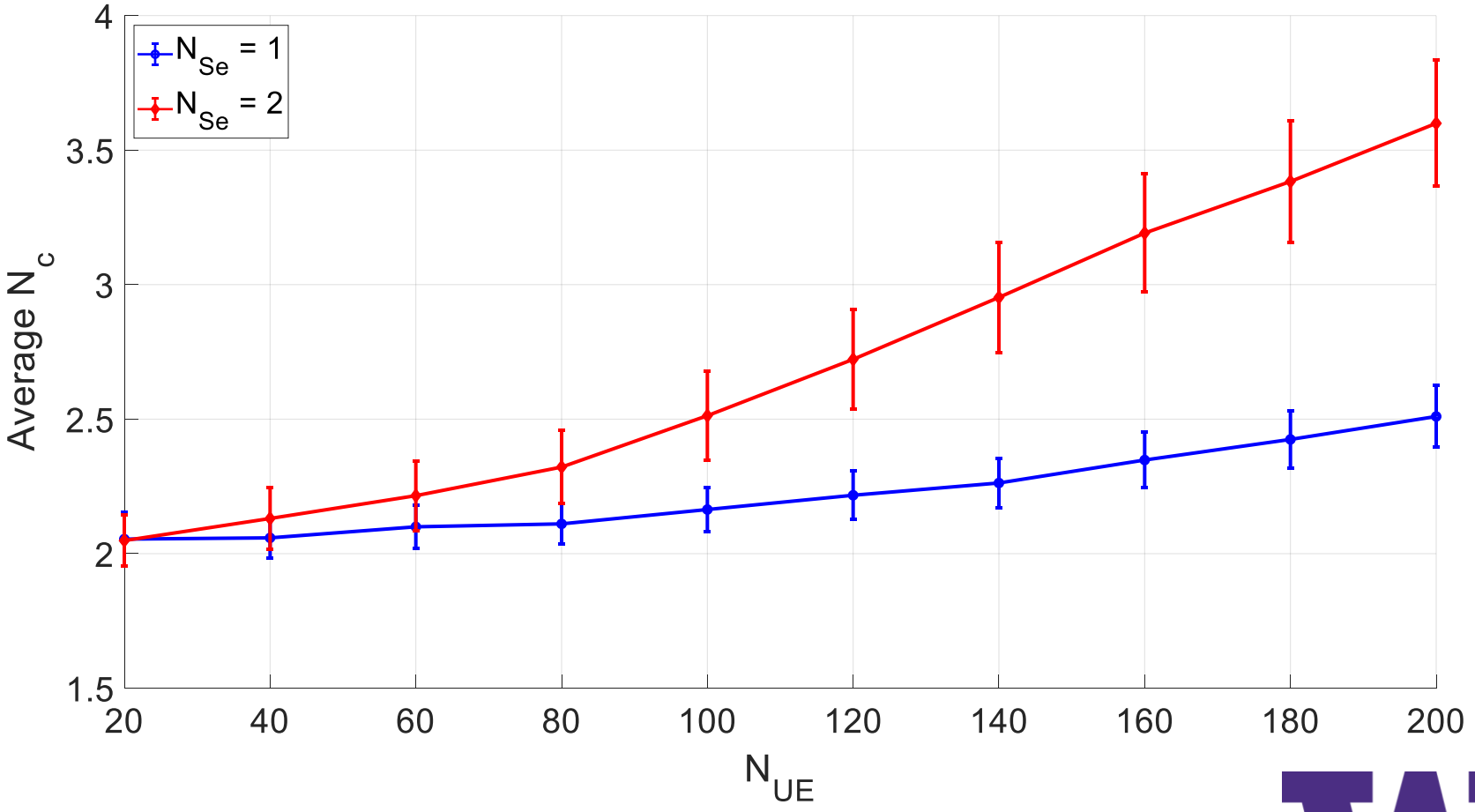
Appendix A: Additional Simple Scenario Results: Effect of N_{Se} on $\overline{N}_a$: $p_K = 0, N_{UE} = 100$



Appendix A: Additional Simple Scenario Results: Effect of p_K on $\bar{N}_c$: $N_{UE} = 100$, $N_{Se} = 1$



Appendix A: Additional Simple Scenario Results: Effect of N_{Se} and N_{UE} on $\bar{N}_c$: $p_K = 0$



Appendix A: Additional Simple Scenario Results: Effect of N_{Se} on $\overline{N}_c$: $p_K = 0, N_{UE} = 100$