ns-3 Training

ns-3 training, June 2017

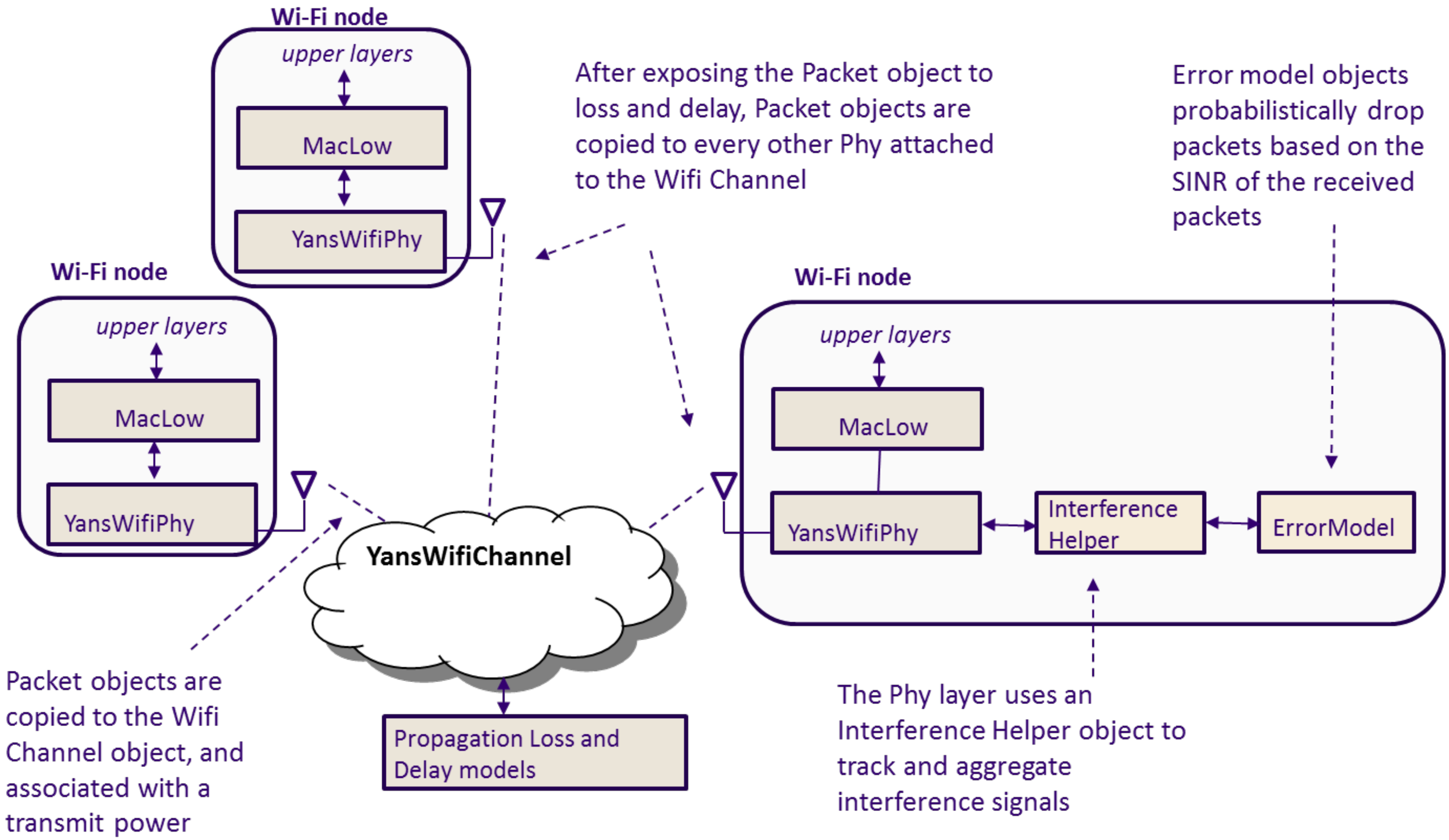
Outline

- Wi-Fi in detail
  - Support of standard features
  - Architecture
  - Configuration via helpers
- Advanced use case: LAA-Wifi-Coexistence
  - SpectrumWifiPhy
  - Adding a LBT Access Manager
  - Scenario support
  - Output data processing

Wi-Fi Overview

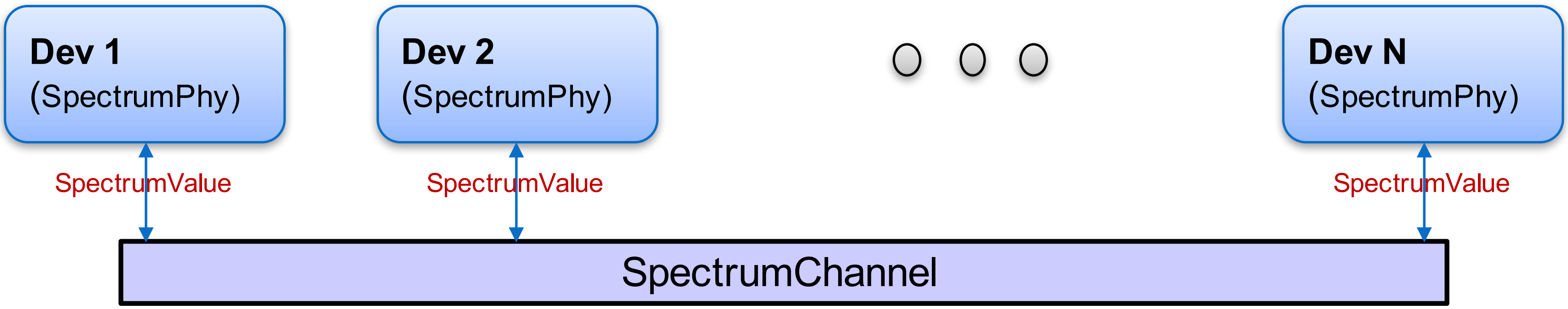
- WiFi module features
  - DCF implementation (Basic + RTS/CTS)
  - 802.11 a/b/g/n/ac (2.4 & 5 GHz) PHY
  - MSDU/MPDU aggregation
  - QoS support (EDCA)
  - Infrastructure and ad-hoc modes
  - Many rate adaptation algorithms
  - AWGN-based error models
- Unsupported features
  - MIMO
  - 11ac advanced features (Tx beamforming, Mu-MIMO)
- Related modules
  - Mesh (802.11s) and WAVE (802.11p/vehicular)

Current Wi-Fi PHY abstraction



ns-3 NETWORK SIMULATOR

Spectrum Module



- **Three key pieces**
  - SpectrumValue, SpectrumChannel, SpectrumPhy
- **SpectrumValue** is the signal abstraction being passed through to the channel (and to the users)
  - A vector of sub-bands representing center frequency, bandwidth, and sub-band power specral density
- **SpectrumChannel** delays/attenuates/modifies the transmitted signal as specified before providing copies to all receivers
  - Automatically converts signals with different resolutions
- Inheriting from **SpectrumPhy** allows different types of devices to interact through the wireless channel

Nodes, Mobility, and Position

- ALL nodes have to be created before simulation starts

- Position Allocators setup initial position of nodes
  - List, Grid, Random position…

- Mobility models specify how nodes will move
  - Constant position, constant velocity/acceleration, waypoint…
  - Trace-file based from mobility tools such as SUMO, BonnMotion (using NS2 format)
  - Routes Mobility using Google API (*)

(*) Presented in WNS3 - 2015

Position Allocation Examples

- List

```
MobilityHelper  mobility;
// place two nodes at specific positions (100,0) and (0,100)
Ptr<ListPositionAllocator>  positionAlloc  = CreateObject<ListPositionAllocator> ();
positionAlloc->Add  (Vector (100, 0, 0));
positionAlloc->Add  (Vector (0, 100, 0));
mobility.SetPositionAllocator(positionAlloc);
```

- Grid Position

```
MobilityHelper  mobility;
// setup the grid itself: nodes are laid out started from (-100,-100) with 20 per row, the x
// interval between each object is 5 meters and the y interval between each object is 20 meters
mobility.SetPositionAllocator  ("ns3::GridPositionAllocator",
                                 "MinX", DoubleValue (-100.0),
                                 "MinY", DoubleValue (-100.0),
                                 "DeltaX", DoubleValue (5.0),
                                 "DeltaY", DoubleValue (20.0),
                                 "GridWidth", UintegerValue (20),
                                 "LayoutType", StringValue ("RowFirst"));
```

- Random Rectangle Position

```
// place nodes uniformly on a straight line from (0, 1000)
MobilityHelper  mobility;
Ptr<RandomRectanglePositionAllocator>  positionAloc  = CreateObject<RandomRectanglePositionAllocator>();
positionAloc->SetAttribute("X",  StringValue("ns3::UniformRandomVariable[Min=0.0|Max=100.0]"));
positionAloc->SetAttribute("Y",  StringValue("ns3::ConstantRandomVariable[Constant=50.0]"));
mobility.SetPositionAllocator(positionAloc);
```

Mobility Model Example

- Constant Position

```
MobilityHelper mobility;
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (nodes);
```

- Constant Speed

```
MobilityHelper mobility;
mobility.SetMobilityModel ("ns3::ConstantVelocityMobilityModel");
mobility.Install (nodes);
Ptr<UniformRandomVariable> rvar = CreateObject<UniformRandomVariable>();
for (NodeContainer::Iterator i = nodes.Begin (); i != nodes.End (); ++i){
  Ptr<Node> node = (*i);
  double speed = rvar->GetValue(15, 25);
  node->GetObject<ConstantVelocityMobilityModel>()->SetVelocity(Vector(speed,0,0));
}
```

- Trace-file based

```
std::string traceFile = “mobility_trace.txt”;
// Create Ns2MobilityHelper with the specified trace log file as parameter
Ns2MobilityHelper ns2 = Ns2MobilityHelper (traceFile);
ns2.Install (); // configure movements for each node, while reading trace file
```

Interesting ns-3 extensions

- ns-3-highway-mobility (https://code.google.com/p/ns-3-highway-mobility/)
  - Implement IDM and MOBIL change lane, highway class, traffic-lights.
  - Based on ns-3.8
  - No longer maintained

- Virtual Traffic Lights (PROMELA) (https://dsn.tm.kit.edu/misc_3434.php)
  - Manhattan IDM mobility model
  - NLOS propagation loss models
  - (Virtual) Traffic Light applications

Propagation Models

- Propagation Loss
  - ITUR1411, LogDistance, ThreeLogDistance, Range, TwoRayGround, Friis
  - Nakagami, Jakes
  - Obstacle model (*)

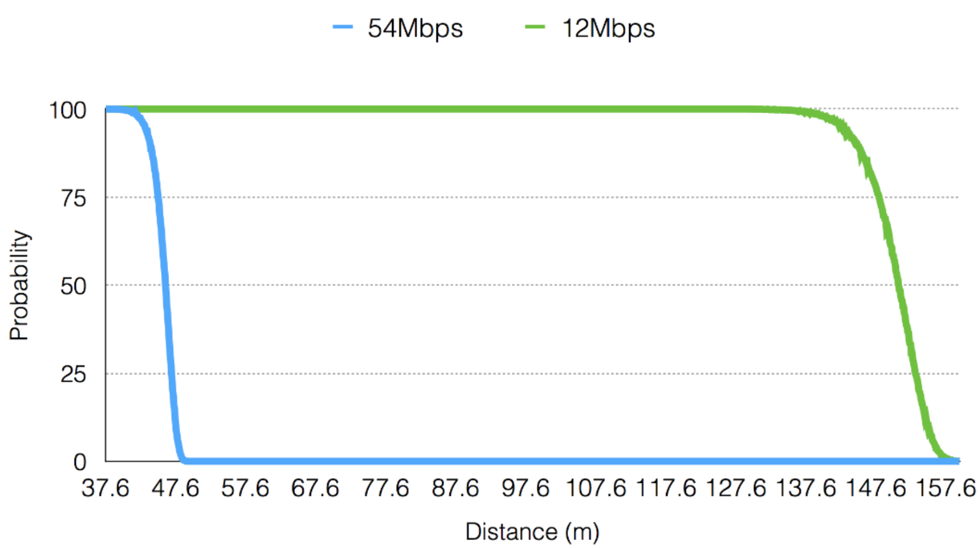
(*) Presented in WNS3 2015

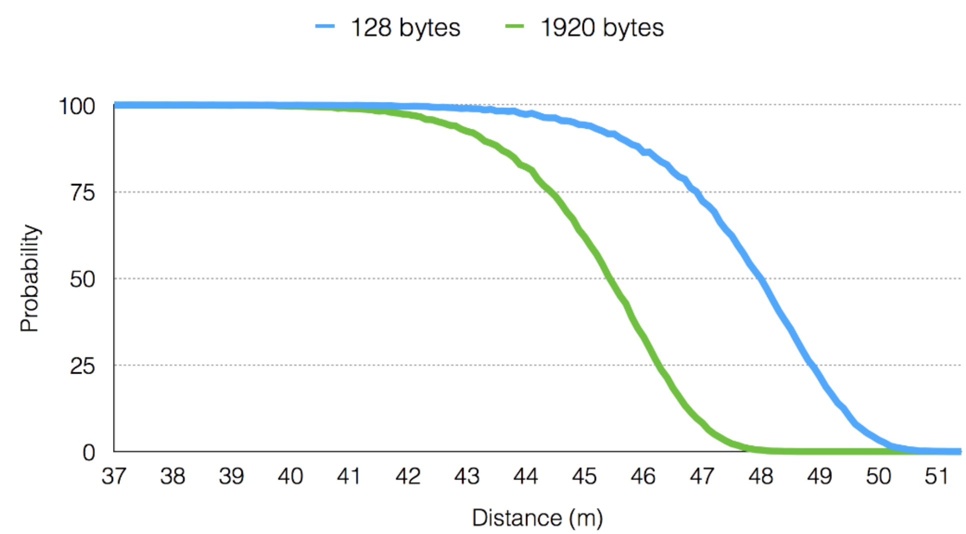
- Propagation Delay
  - Constant Speed
  - Random

- Be careful when using `YansWifiChannelHelper::Default()` the LogDistance propagation model is added. Calling `AddPropagationLoss()` again will add a *second* propagation loss model.

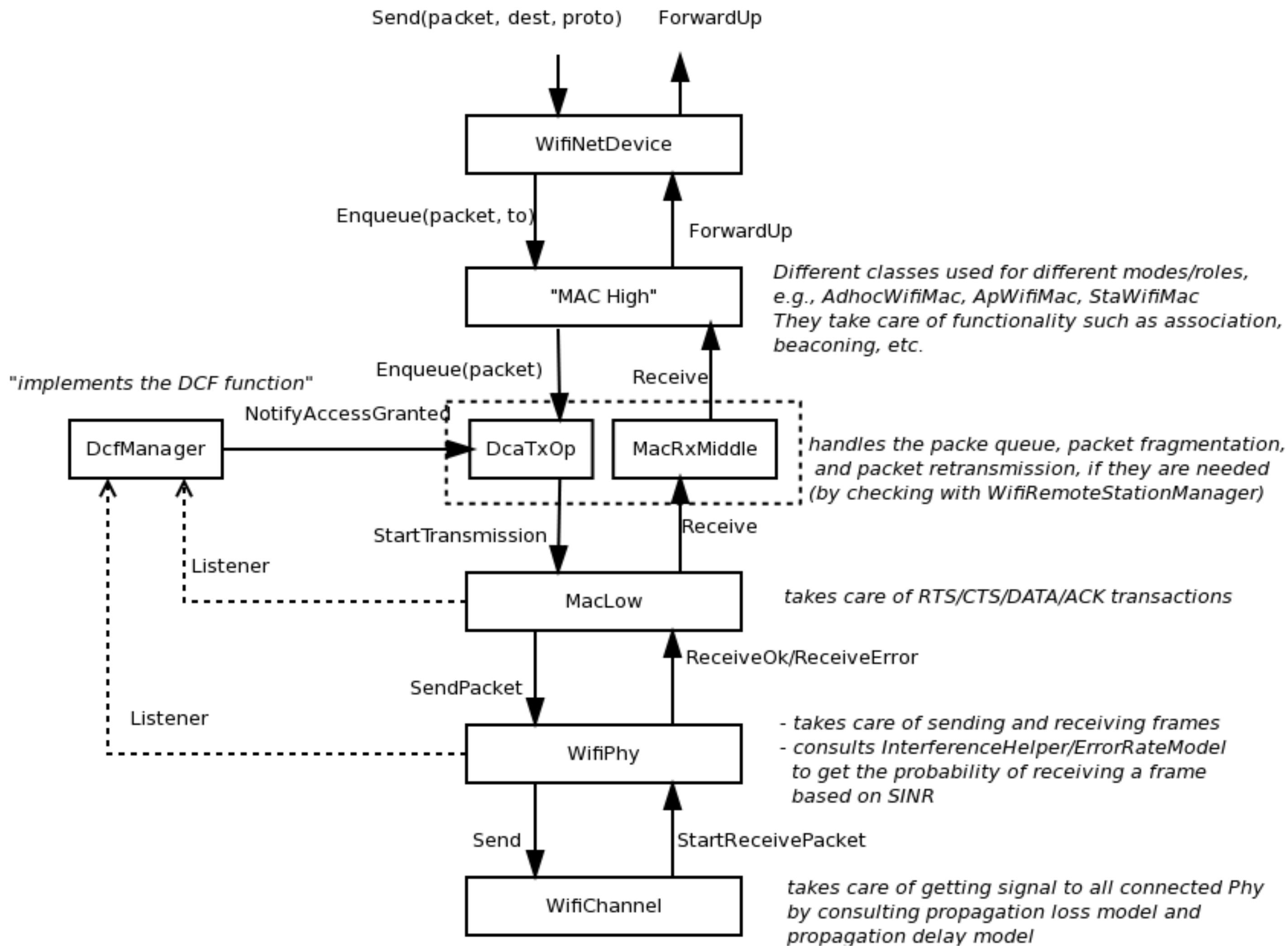
Communication Range

- Depends on many factors
  - Propagation loss model and PHY configuration
  - Frame size (big vs small)
  - Transmission mode (6Mbps vs 54 Mbps)





Wi-Fi Architecture



MAC High

- Presently, three MAC high models
  - AdhocWifiMac: simplest one
  - ApWifiMac: beacon, associations by STAs
  - StaWifiMac: association based on beacons
- All inherit from RegularWifiMac, which handles QoS and non-QoS support

Rate controls

The following rate control algorithms can be used by the MAC low layer:

- Algorithms found in real devices:
  - ArfWifiManager (default for WifiHelper), OnoeWifiManager, ConstantRateWifiManager, MinstrelWifiManager, MinstrelHtWifiManager
- Algorithms in literature:
  - IdealWifiManager, AarfWifiManager, AmrrWifiManager, CaraWifiManager, RraaWifiManager, AarfcdWifiManager, ParfWifiManager, AparfWifiManager
- Example use of constant rate

```
std::string phyMode ("OfdmRate54Mbps");
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
                              "DataMode",StringValue (phyMode),
                              "ControlMode",StringValue (phyMode));
```

MAC Middle/Low

Three components:

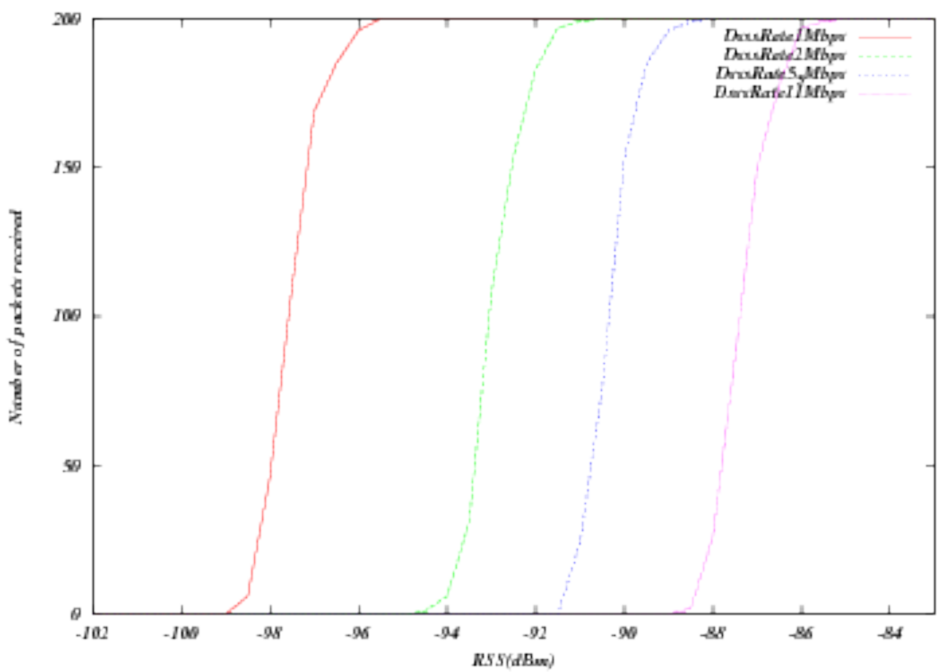
- MacLow
  - RTS/CTS/DATA/ACK transactions
  - Aggregation, Block acks
- DcfManager
  - implements the DCF
- DcaTxop and EdcaTxopN:
  - One for NQoS, the other for QoS
  - Packet queue
  - Fragmentation/Retransmissions

Physical layer

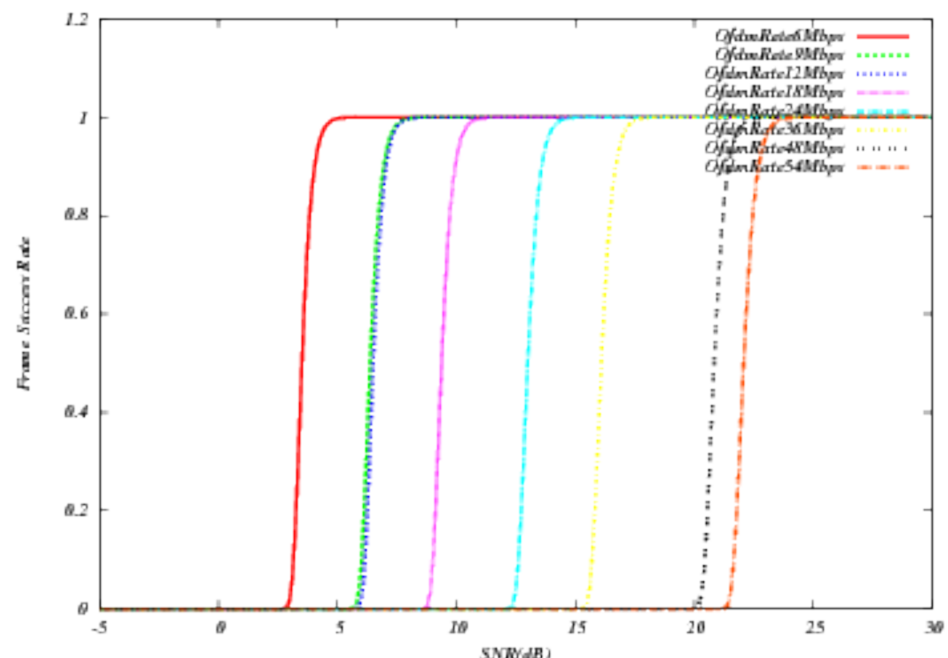
- No AGC model
- Sync on first RX with energy > detection threshold
- Collision: the error model will likely cause a drop of the packet
- No capture effect: won't re-sync on a stronger packet

Error models

- Based on analytical models with error bounds
- Three implementations with different bounds: YansErrorRateModel, NistErrorRateModel, DssErrorRateModel



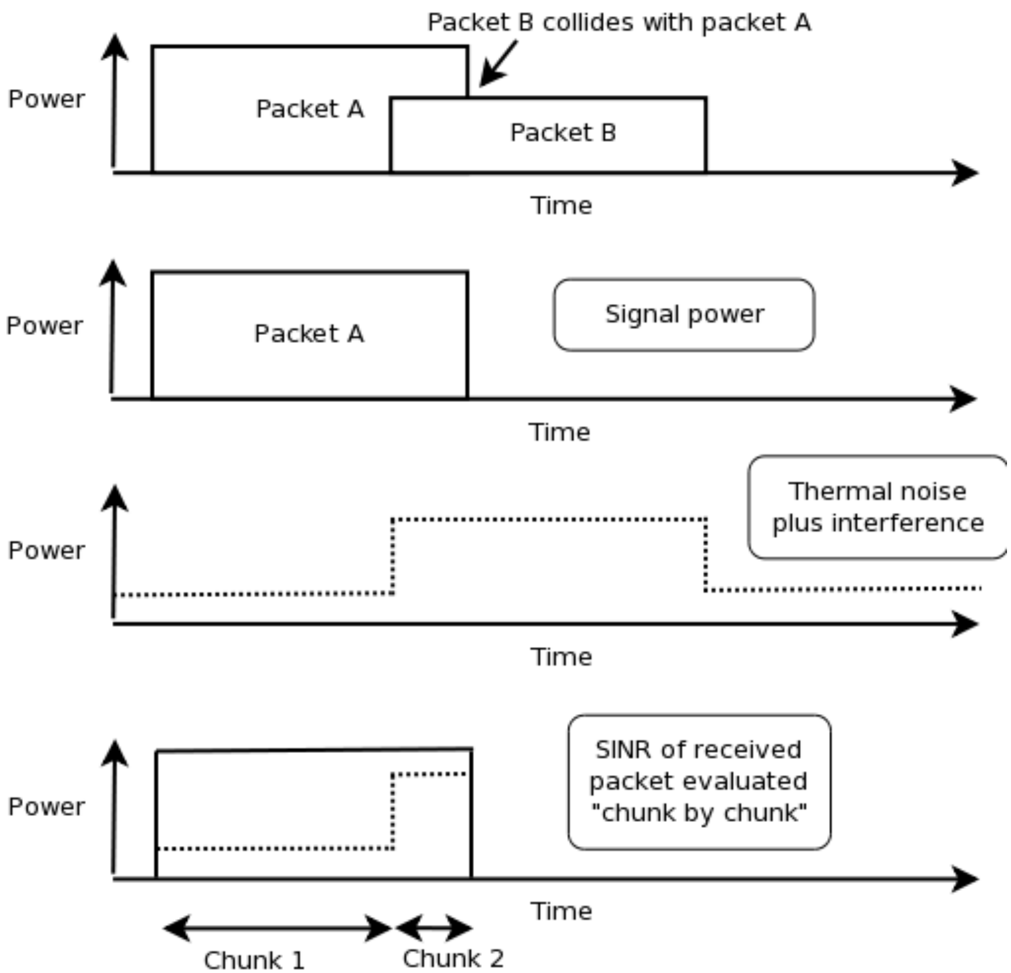
DsssErrorRateModel



NistErrorRateModel (OFDM)

Interference helper

- SINR evaluated on chunk-by-chunk basis



ns-3 NETWORK SIMULATOR

Configuring 802.11n/ac

- Example programs include
  - examples/wireless/ht-wifi-network.cc
  - examples/wireless/vht-wifi-network.cc
  - examples/wireless/wifi-aggregation.cc
- Setting the WifiPhyStandard will set most defaults reasonably

```
WifiHelper wifi;
wifi.SetStandard (WIFI_PHY_STANDARD_80211ac);
WifiMacHelper mac;
```

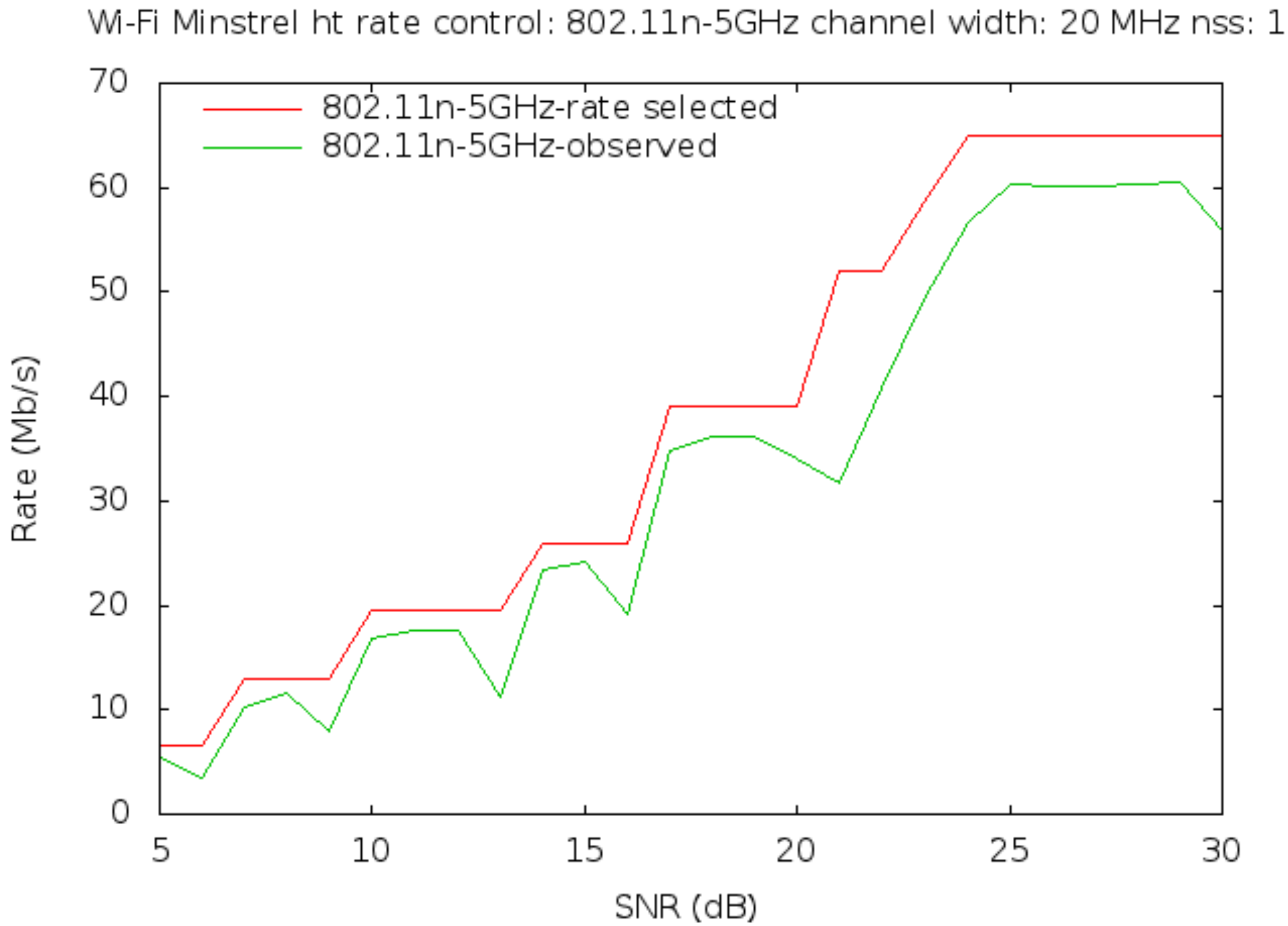
- 802.11ac uses 80 MHz channel by default; 802.11n uses 20 MHz

802.11n/ac rate controls

- Possible options are IdealWifiManager, MinstrelHtWifiManager, and ConstantRateWifiManager
- examples:
  - src/wifi/examples/ideal-wifi-manager-example.cc
  - src/wifi/examples/minstrel-ht-wifi-manager-example.cc

Example output for MinstrelHt

```
$ ./waf --run "minstrel-ht-wifi-manager-example --standard=802.11n-5GHz"
$ gnuplot minstrel-ht-802.11n-5GHz-20MHz-LGI-1SS.plt
```



ns-3 NETWORK SIMULATOR

Typical configuration

```
std::string phyMode ("DsssRate1Mbps");

NodeContainer ap;
ap.Create (1);
NodeContainer sta;
sta.Create (2);

WifiHelper wifi;
wifi.SetStandard (WIFI_PHY_STANDARD_80211b);

YansWifiPhyHelper wifiPhy =  YansWifiPhyHelper::Default ();
// ns-3 supports RadioTap and Prism tracing extensions for 802.11
wifiPhy.SetPcapDataLinkType (YansWifiPhyHelper::DLT_IEEE802_11_RADIO);

YansWifiChannelHelper wifiChannel;
// reference loss must be changed since 802.11b is operating at 2.4GHz
wifiChannel.SetPropagationDelay
("ns3::ConstantSpeedPropagationDelayModel");
wifiChannel.AddPropagationLoss ("ns3::LogDistancePropagationLossModel",
                                "Exponent", DoubleValue (3.0),
                                "ReferenceLoss", DoubleValue (40.0459));
wifiPhy.SetChannel (wifiChannel.Create ());
```

ns-3 NETWORK SIMULATOR

Typical configuration (cont.)

```
// Add a non-QoS upper mac, and disable rate control
WifiMacHelper wifiMac;
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
                              "DataMode",StringValue (phyMode),
                              "ControlMode",StringValue (phyMode));

// Setup the rest of the upper mac
Ssid ssid = Ssid ("wifi-default");
// setup ap.
wifiMac.SetType ("ns3::ApWifiMac",
                 "Ssid", SsidValue (ssid));
NetDeviceContainer apDevice = wifi.Install (wifiPhy, wifiMac, ap);
NetDeviceContainer devices = apDevice;

// setup sta.
wifiMac.SetType ("ns3::StaWifiMac",
                 "Ssid", SsidValue (ssid),
                 "ActiveProbing", BooleanValue (false));
NetDeviceContainer staDevice = wifi.Install (wifiPhy, wifiMac, sta);
devices.Add (staDevice);
```

Typical configuration (cont.)

```
// Configure mobility
MobilityHelper mobility;
Ptr<ListPositionAllocator> positionAlloc =
CreateObject<ListPositionAllocator> ();
positionAlloc->Add (Vector (0.0, 0.0, 0.0));
positionAlloc->Add (Vector (5.0, 0.0, 0.0));
positionAlloc->Add (Vector (0.0, 5.0, 0.0));
mobility.SetPositionAllocator (positionAlloc);
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (ap);
mobility.Install (sta);

// other set up (e.g. InternetStack, Application)
```

Athstats helper

Hooks Wi-Fi traces to provide debugging similar to Madwifi drivers

Example athstats output from example `./waf --run wifi-ap`

```
m_txCount   m_rxCount  unused  short  long exceeded rxError

      0       0     0      0      0      0      0      0      0     0    0M
      0      60     0      0      0      0      0      0      0     0    0M
      0     123     0      0      0      0      0      0      0     0    0M
      0     122     0      0      0      0      0      0      0     0    0M
      0     122     0      0      0      0      0      0      0     0    0M
      0     122     0      0      0      0      0      0      0     0    0M
      0     122     0      0      0      0      2      0      0     0    0M
      0     122     0      0      0      0     23      0      0     0    0M
      0     122     0      0      0      0     14      0      0     0    0M
      0     122     0      0      0      0     14      0      0     0    0M
      0     122     0      0      0      0     26      0      0     0    0M
      0     122     0      0      0      0     12      0      0     0    0M
```

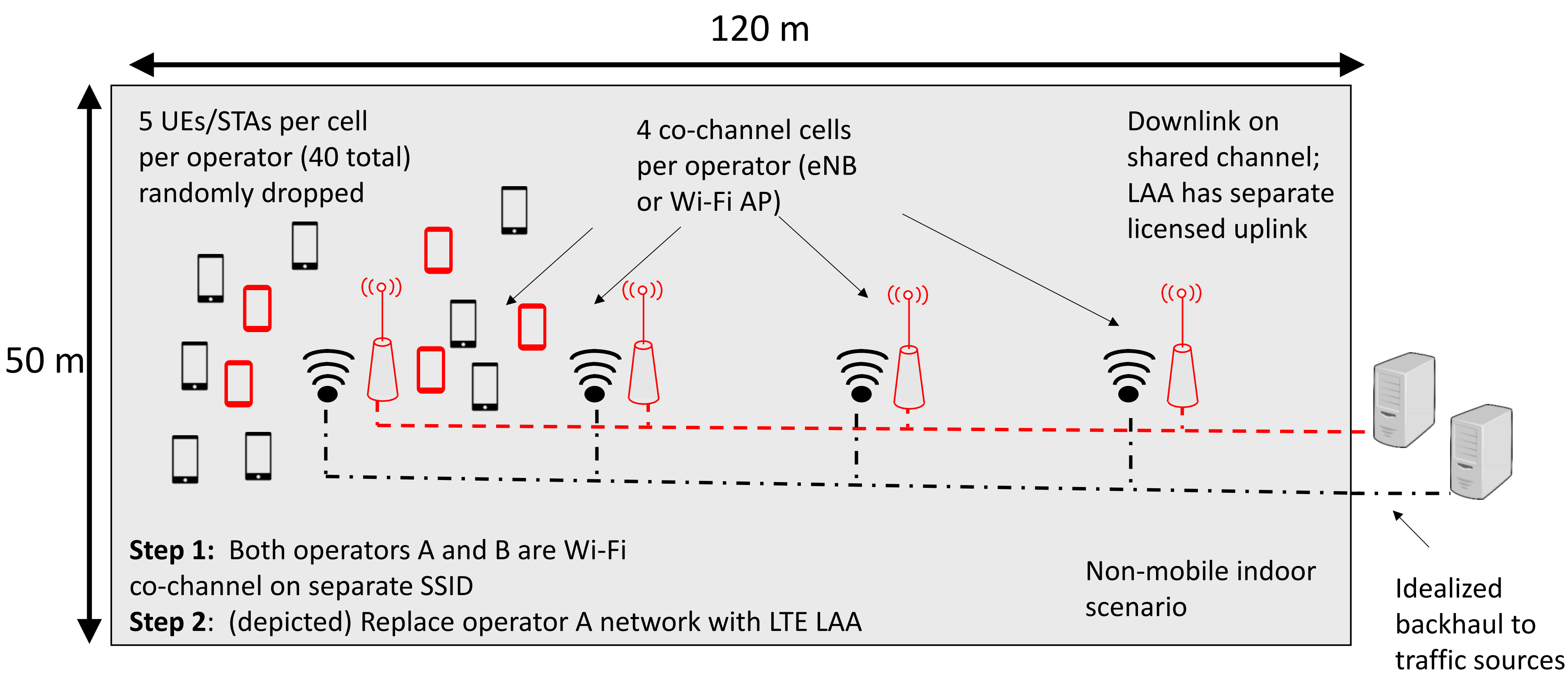
ns-3 NETWORK SIMULATOR

LTE/Wi-Fi Coexistence

Use case: LAA Wi-Fi Coexistence

- ns-3 has been extended to support scenarios for LTE LAA/Wi-Fi Coexistence
- Methodology defined in 3GPP Technical Report TR36.889
- Enhancements needed:
  - Wireless models (LBT access manager, SpectrumWifiPhy, propagation/fading models)
  - Scenario support (traffic models)
  - Output data processing

Indoor 3GPP scenario



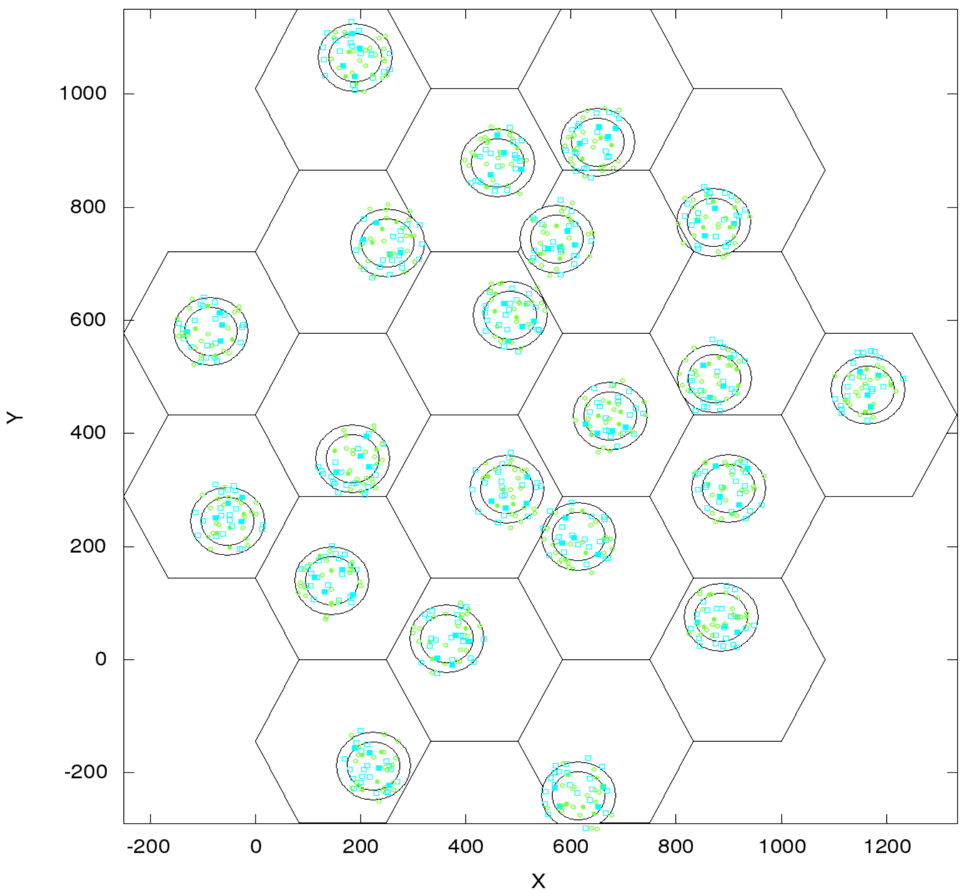
Indoor scenario details

Unlicensed channel model	3GPP TR 36.889	ns-3 implementation
Network Layout	Indoor scenario	Indoor scenario
System bandwidth	20 MHz	20 MHz
Carrier frequency	5 GHz	5 GHz (channel 36, tunable)
Number of carriers	1, 4 (to be shared between two operators) 1 for evaluations with DL+UL Wi-Fi coexisting with DL-only LAA	1 for evaluations with DL+UL Wi-Fi coexisting with DL-only LAA
Total Base Station (BS) transmission power	18/24 dBm	18/24 dBm Simulations herein consider 18 dBm
Total User equipment (UE) transmission power	18 dBm for unlicensed spectrum	18 dBm
Distance dependent path loss, shadowing and fading	ITU InH	802.11ax indoor model
Antenna pattern	2D Omni-directional	2D Omni-directional
Antenna height	6 m	6 m (LAA, not modelled for Wi-Fi)
UE antenna height	1.5 m	1.5 m (LAA, not modelled for Wi-Fi)
Antenna gain	5 dBi	5 dBi
UE antenna gain	0 dBi	0 dBi
Number of UEs	10 UEs per unlicensed band carrier per operator for DL-only 10 UEs per unlicensed band carrier per operator for DL-only for four unlicensed carriers. 20 UEs per unlicensed band carrier per operator for DL+UL for single unlicensed carrier. 20 UEs per unlicensed band carrier per operator for DL+UL Wi-Fi coexisting with DL-only LAA	Supports all the configurations in TR 36.889. Simulations herein consider the case of 20 UEs per unlicensed band carrier per operator for DL LAA coexistence evaluations for single unlicensed carrier.
UE Dropping	All UEs should be randomly dropped and be within coverage of the small cell in the unlicensed band.	Randomly dropped and within small cell coverage.
Traffic Model	FTP Model 1 and 3 based on TR 36.814 FTP model file size: 0.5 Mbytes. Optional: VoIP model based on TR36.889	FTP Model 1 as in TR36.814. FTP model file size: 0.5 Mbytes Voice model: DL only
UE noise figure	9 dB	9 dB
Cell selection	For LAA UEs, cell selection is based on RSRP (Reference Signal Received Power. For Wi-Fi stations (STAs), cell selection is based on RSS (Received signal power strength) of WiFi Access Points (APs). RSS threshold is -82 dBm.	RSRP for LAA UEs and RSS for Wi-Fi STAs
Network synchronization	For the same operator, the network can be synchronized. Small cells of different operators are not synchronized.	Small cells are synchronized, different operators are not synchronized.

ns-3 NETWORK SIMULATOR

Outdoor 3GPP scenario

Outdoor layout: hexagonal macrocell layout. 7 macro sites and 3 cells per site. 1 Cluster per cell. 4 small cells per operator per cluster, uniformly dropped. ITU UMi channel model.



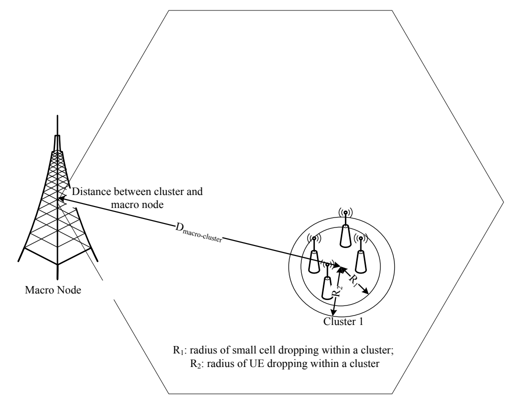


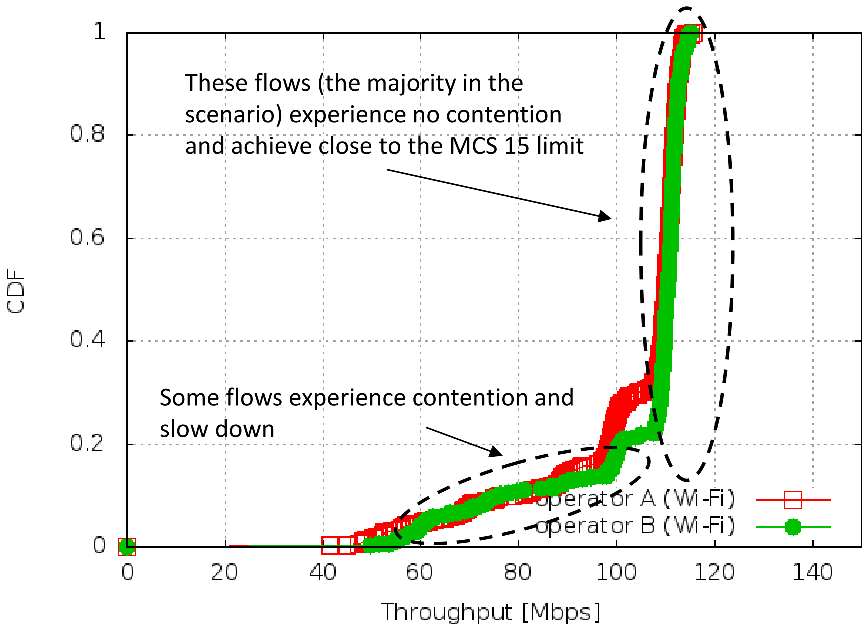
Figure source: 3GPP TR 36.889 V13.0.0 (2015-05)

References

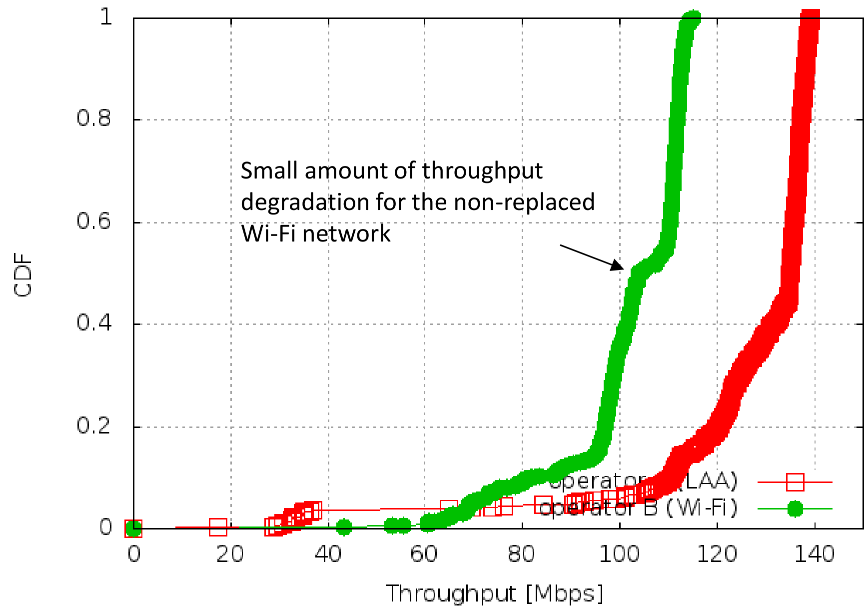
- ns-3 Wiki page:
  - https://www.nsnam.org/wiki/LAA-WiFi-Coexistence
    - module documentation
    - references to various publications
    - documentation on reproducing results
- Code:
  - http://code.nsnam.org/laa/ns-3-lbt

ns-3 NETWORK SIMULATOR

Sample results



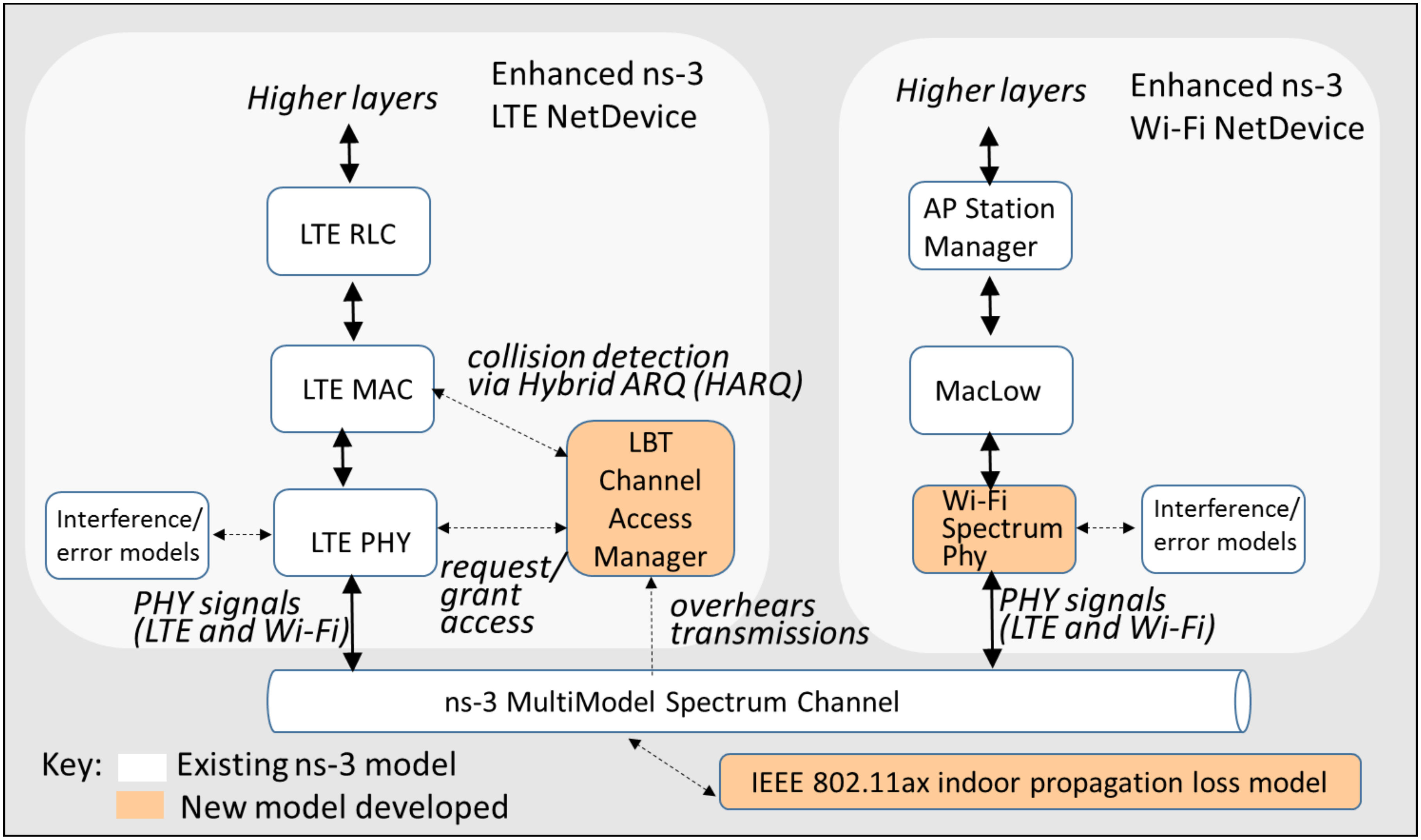
a) Step 1 (Wi-Fi)



b) Step 2 (LAA)

NETWORK SIMULATOR

Wi-Fi enhancements



Model enhancements: Wi-Fi

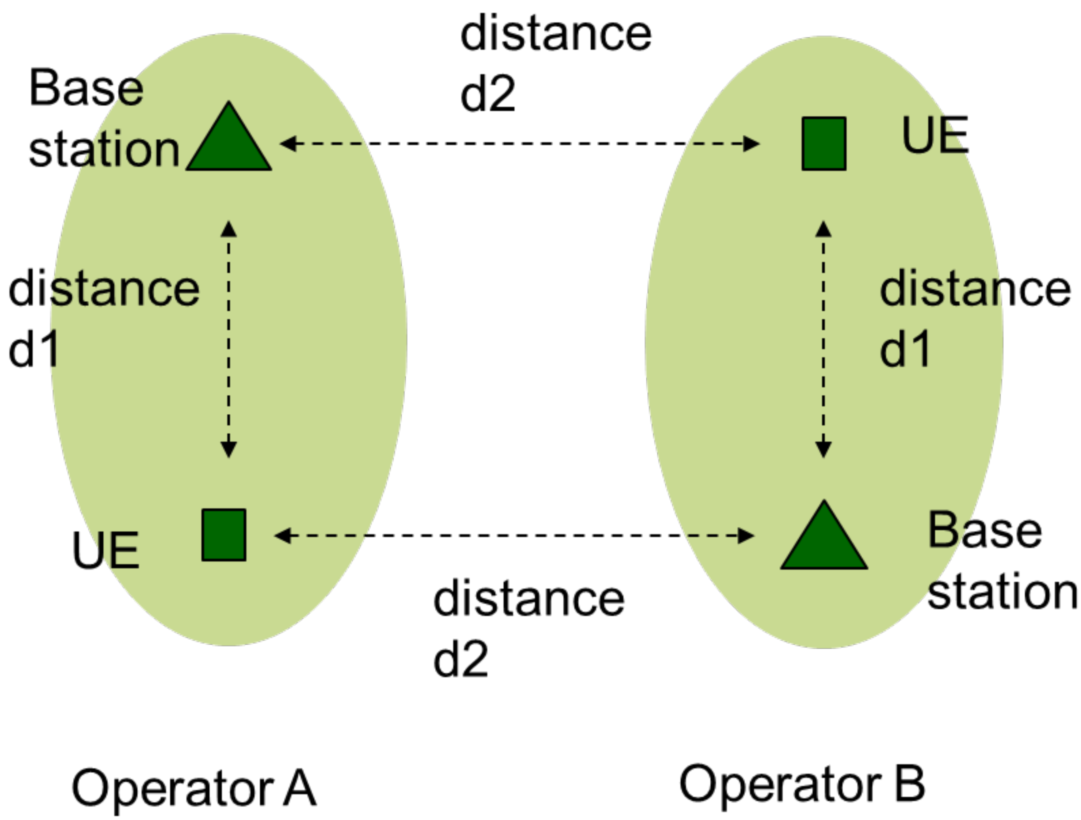
- Spectrum Wi-Fi Phy implementation
- Wi-Fi preamble detection based on AWGN and TGn Channel Model D
- Wi-Fi RSS-based AP selection and roaming
- Wi-Fi MIMO approximations to support 2x2 DL, 1x2 DL on AWGN and TGn Model D

Model enhancements: LTE

- LTE interference model relies on the simplifying assumption that all interfering signals are LTE and are synchronized at the subframe level.
- LTE inteference model has been enhanced to handle inteference by signals of any type.
- This relies on the ns-3 Spectrum framework.
- The reception of LTE signals is evaluated by chunks, where each chunk is identified by a constant power spectral density.

Scenario

- An initial test scenario, useful for testing basic model operation in a small scale setting, grew into TR36.889-like indoor and outdoor scenarios
- D1 and d2 can vary and operator A and B can be both LTE or Wi-Fi



Output experiment scripts

- Shell scripts and gnuplot helpers to manage configuration and data output
- *(demonstrate)*

Status

- Portions are being migrated into ns-3-dev
  - SpectrumWifiPhy in ns-3.26
  - LTE components, propagation model likely in ns-3.27
  - Scenario helper may be rewritten
- Trying to decompose into pieces easy to merge
- For more information:
  - https://www.nsnam.org/wiki/LAA-WiFi-Coexistence