Wi-Fi simulations with ns-3

Instructors: Tom Henderson and Hao Yin

LANC Conference

October 19, 2022

UNIVERSITY *of* WASHINGTON

Copying and Credits

> Slides (and all ns-3 documentation) are provided by the authors under Creative Commons CC-BY-SA-4.0
  - https://creativecommons.org/licenses/by-sa/4.0/

> Credit is due to ns-3's long list of Wi-Fi module maintainers
  - Mathieu Lacage, Nicola Balco, Ghada Badawy, Getachew Redietab, Matias Richart, Stefano Avallone **(current),** Sebastien Deronne **(current)**

> This tutorial development was funded by NSF award CNS-2016379

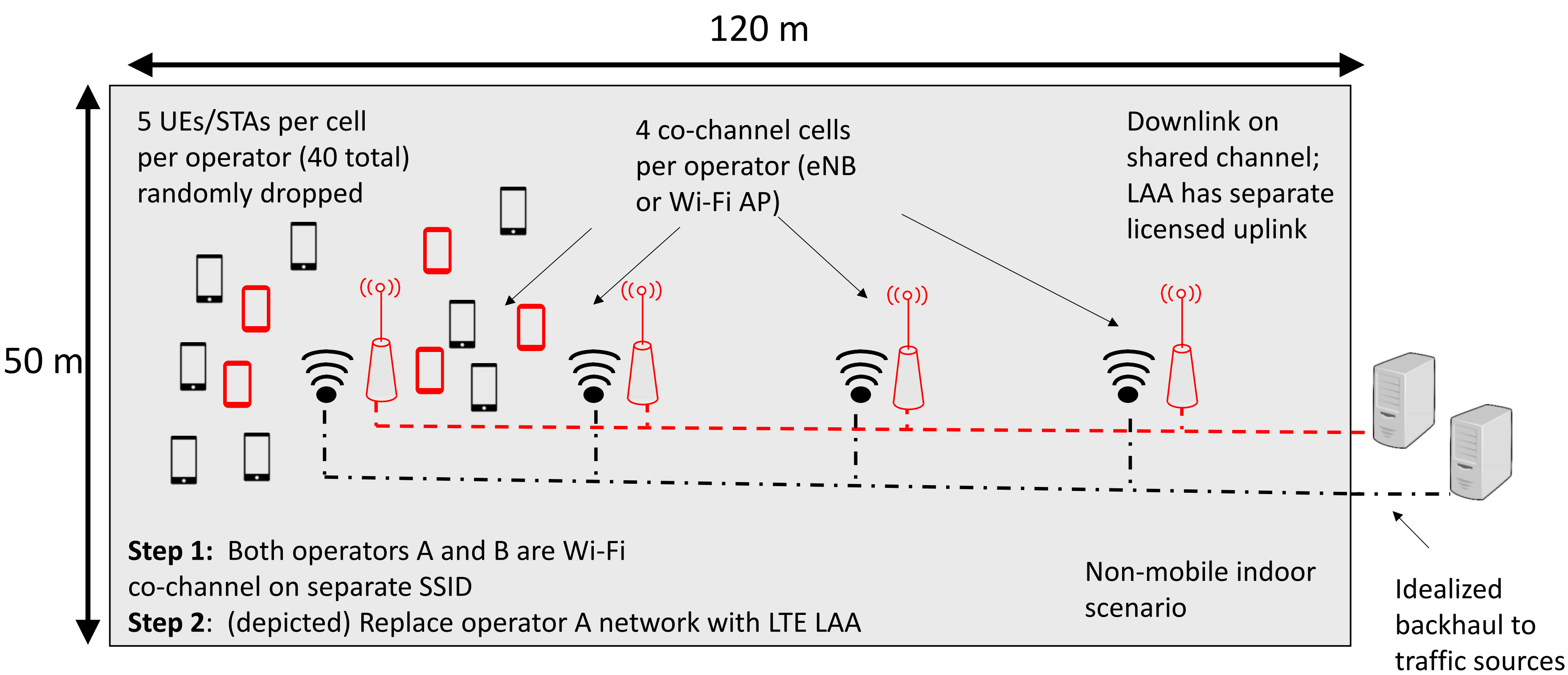
Goals of this tutorial

- Explain why you might use ns-3 to study or learn about Wi-Fi networking
- Illustrate some basic aspects of Wi-Fi operation
- Show how you can get started with ns-3 Wi-Fi simulations already written by others
- Answer your questions

What is ns-3?

> Software to build ***models of computer networks***, to conduct ***performance evaluation studies***

Question: Can LTE safely co-exist with Wi-Fi?



Network Performance Analysis Fundamentals

- Studies are conducted to try to answer questions
- "Can LTE safely co-exist with Wi-Fi?"
  - Question is too broad; need to sharpen its focus
- **Guideline 1:** Clearly state the goals of the study and define the scope
- **Guideline 2:** Select performance metrics

- ***Refined question:*** "Can a specific unlicensed variant of LTE (LAA) operate in the same spectrum as a Wi-Fi network, without impacting Wi-Fi system throughput and latency more than another co-located Wi-Fi network would impact it?"

Network Performance Analysis Fundamentals (cont.)

- What do you mean by “throughput” and “latency”?
  - How measured? (precise definition)
  - What statistics? (average throughput, 99%th percentile, worst-case, etc.)?

- **Guideline 3:** Select system and experimental parameters

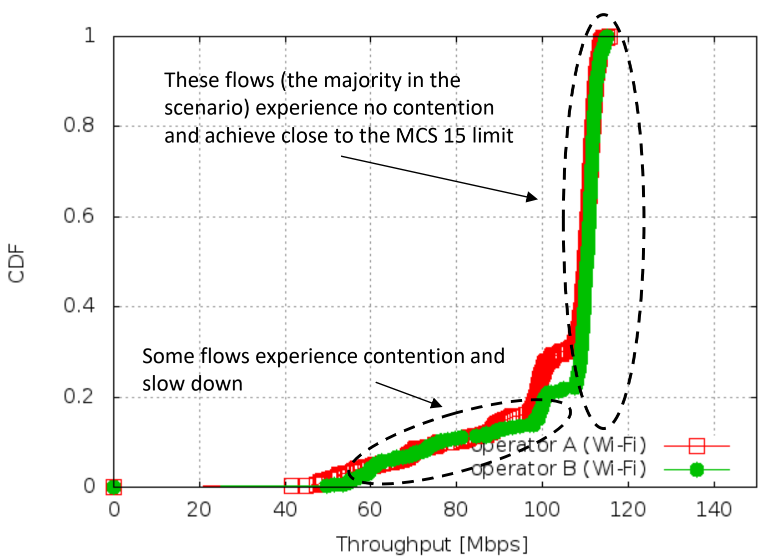
Network Performance Analysis Fundamentals (cont.)

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.

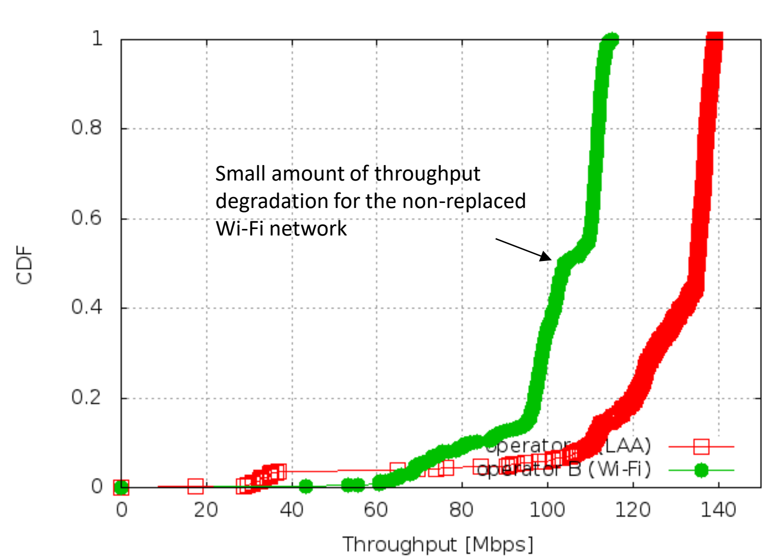
Figure from: http://arxiv.org/abs/1604.06826

Network Performance Analysis Fundamentals (cont.)

- **Guideline 4:** Design experiments
  - Select evaluation techniques
  - Select factors and their values
- Example: Place two Wi-Fi networks in same region, fully load the system, and plot a CDF of observed throughputs per station. Repeat by replacing one Wi-Fi network with LAA.



a) Step 1 (Wi-Fi)



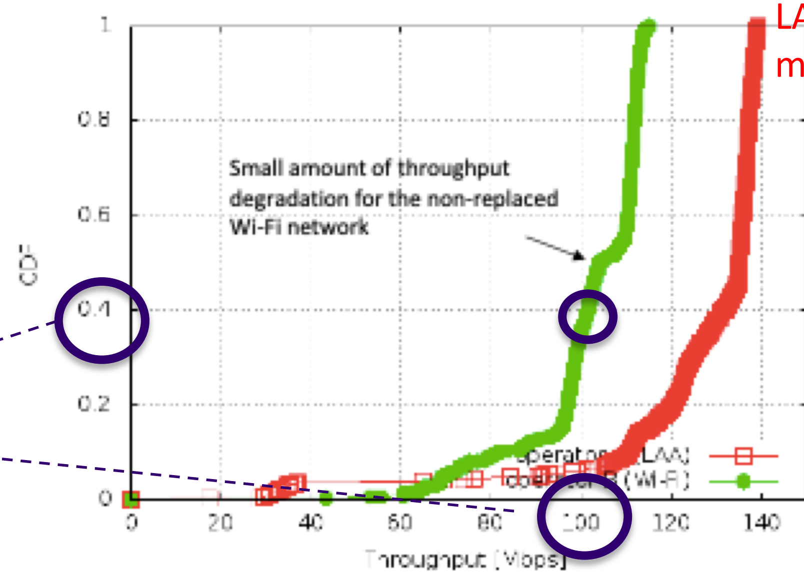
b) Step 2 (LAA)

Aside: What is a ‘CDF’?

> A cumulative distribution function measures the probability that samples fall below a specified value: For random variable X, F(x) = P(X <= x)

Wi-Fi throughput measurements

LAA throughput measurements



b) Step 2 (LAA)

40 % of Wi-Fi throughputs are below 100 Mb/s

Why interesting?

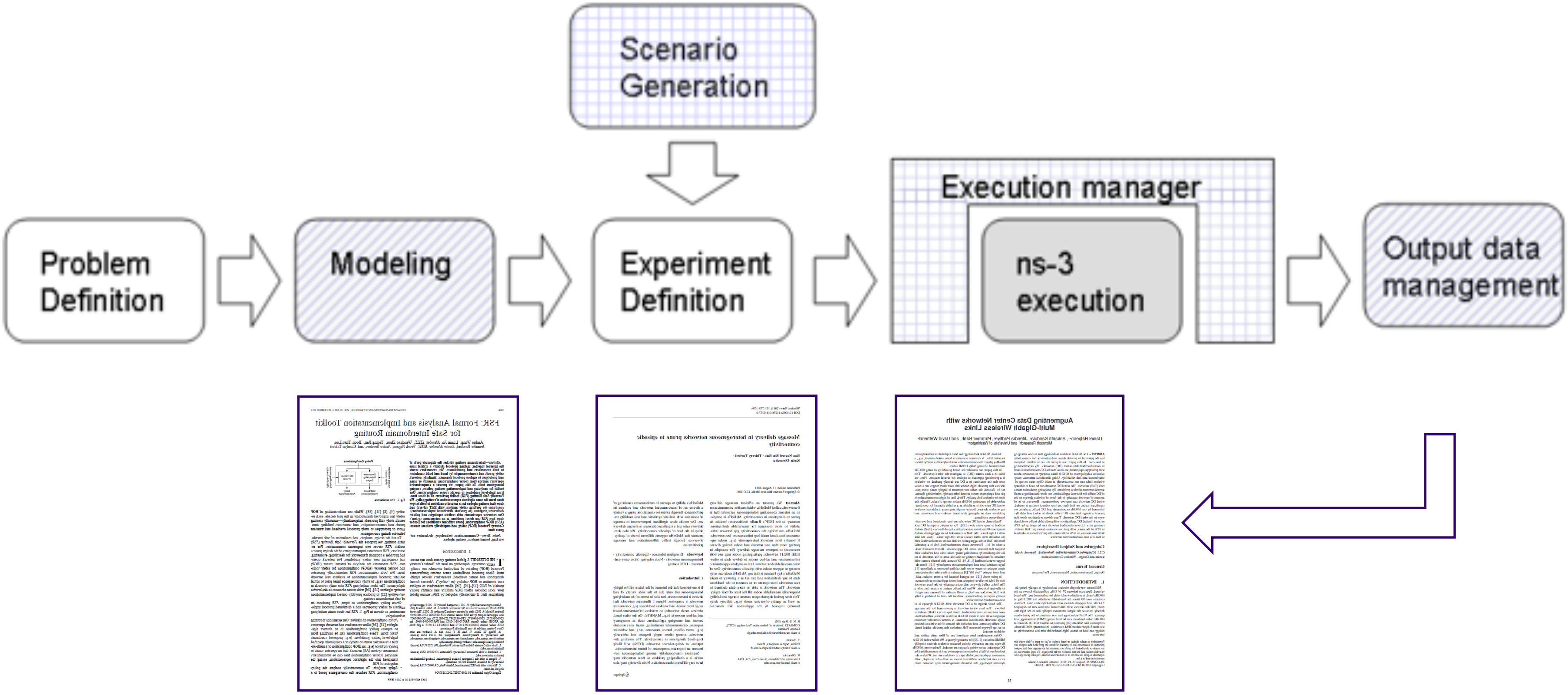
- Many networked systems are designed with concerns about worst-case behavior
- A CDF provides a sense of the spread of the data samples

Network Performance Analysis Fundamentals (cont.)

- **Guideline 5**: Analyze and interpret data, and iterate
  - Almost never a one-shot process
  - Often need to dig deeper into model or scenario, to mine it for fine-grained detail
- **Guideline 6**: Make your results easy to reproduce
  - For others, and by yourself (at a later date)

What is ns-3?

> The previous case study follows a workflow:

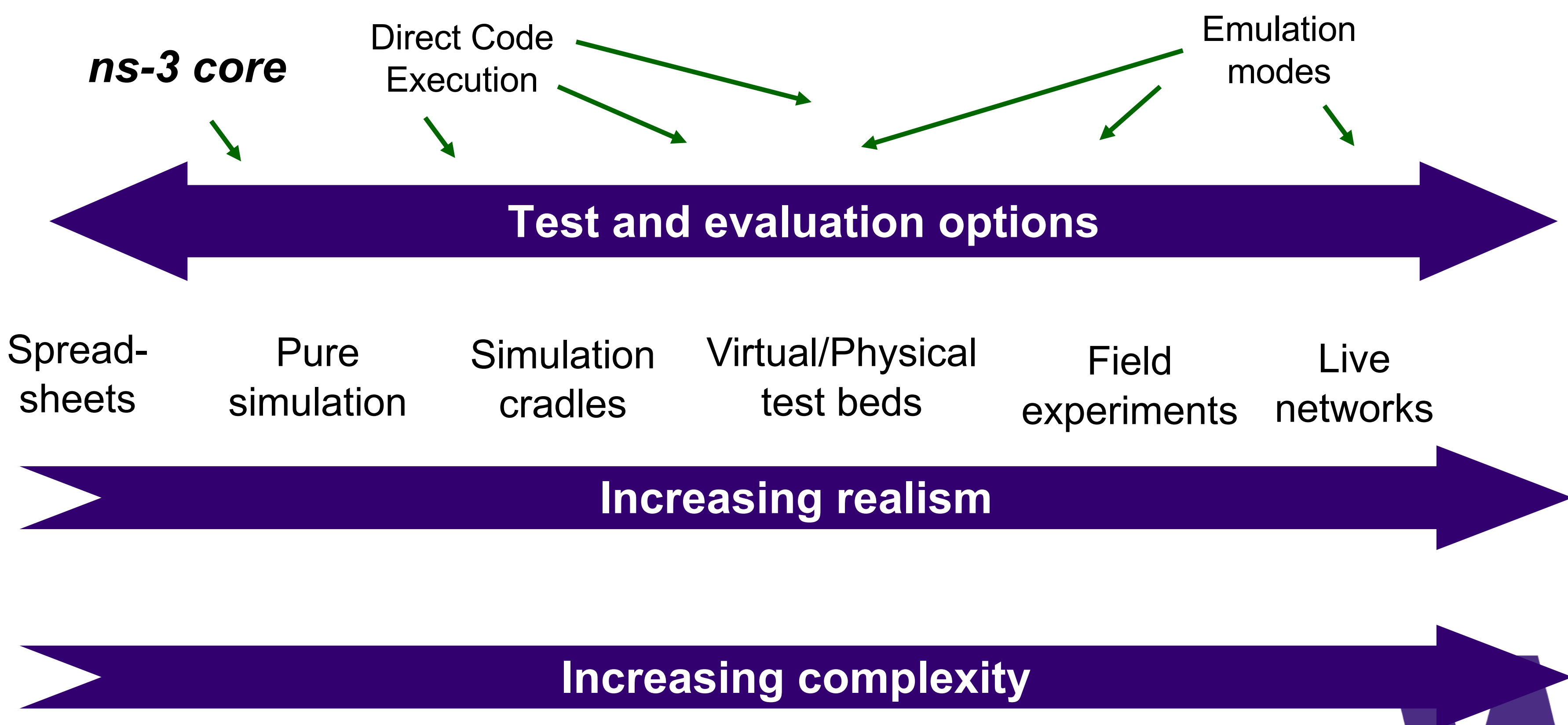


Performance evaluation alternatives

- Mathematical analysis
- Numerical computing packages (e.g., MATLAB)
- Packet-level simulators
- System-level simulators
- Testbeds, prototypes
- Field trials

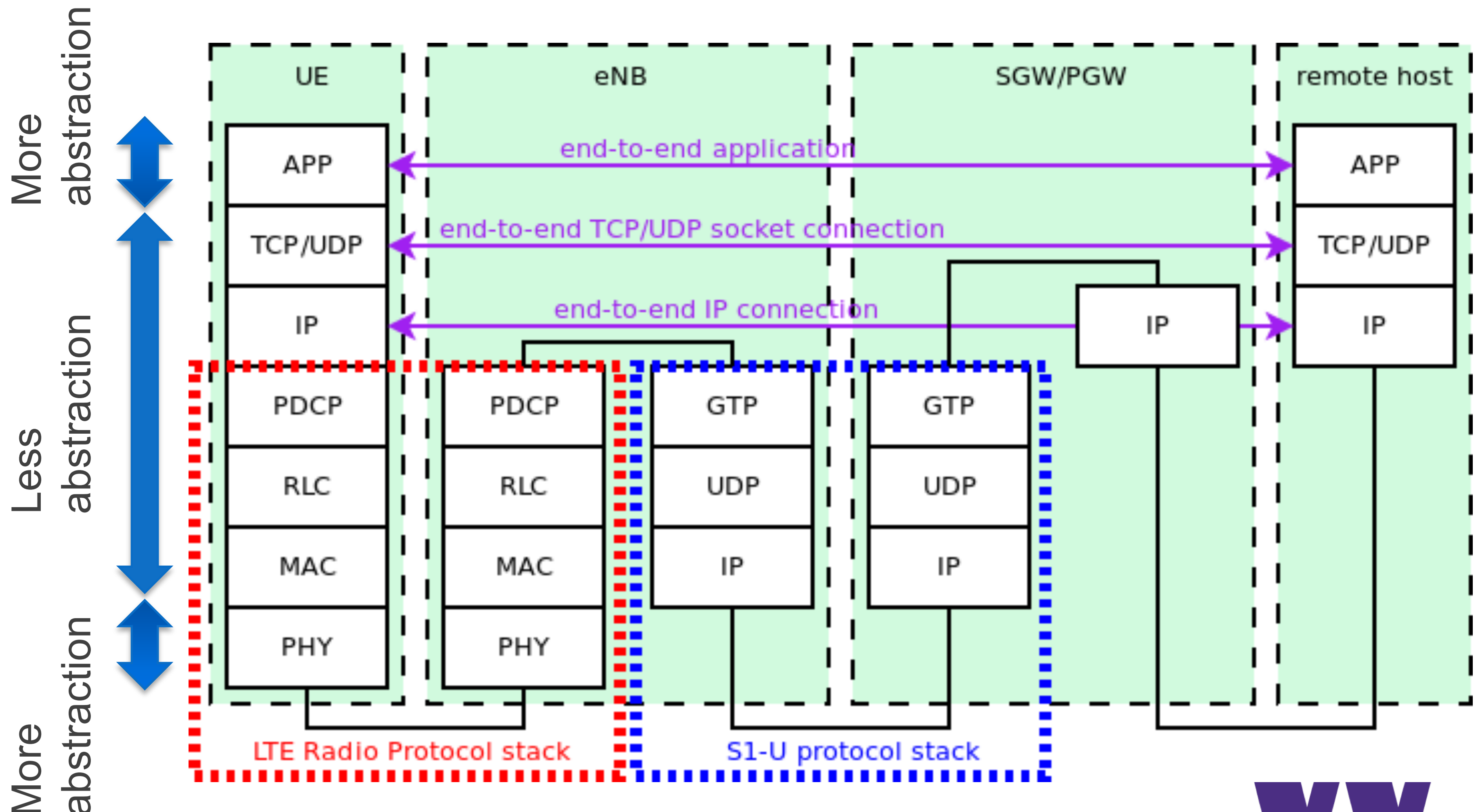
What is ns-3? (cont.)

> ns-3 is mainly a discrete-event simulator, but also has modes of operation that allows it to interact with real-world software and networks



What is ns-3? (cont.)

> **Packet-level network simulation:** The main unit of modeling is the *packet* and entities that exchange packets.



Wi-Fi and ns-3

- Modern Wi-Fi design is focused on solving ***resource allocation problems*** to maximize throughput and minimize latency
  - Rate control
  - Scheduling
  - Power control
  - Interference mitigation
- User experience (throughput, latency) depends on end-to-end details and cross-traffic effects
  - e.g., gaming latency, transport protocols and congestion control, traffic prioritization and queueing disciplines

Packet-level simulators like ns-3 are ideal for these studies

Wi-Fi simulation studies

- IEEE TGax has worked on simulation scenario and methodology guidelines for Wi-Fi simulations
- References:
  - TGax Simulation Scenarios
  - TGax Simulation Methodology

ns-3 software basics

- *ns-3 is written in C++*
  - most code conforms to C++17 or earlier standards, and makes use of the STL (standard template library)
  - ns-3 makes use of a collection of C++ design patterns and enhancements with applicability to network simulation
  - ns-3 programs make use of standard C++, ns-3 libraries written in C++, and (optionally) third-party C++ libraries
- *ns-3's build system is based on CMake and Python 3*
  - minimal system requirements: g++/clang++, CMake, either make or ninja build system, and Python
  - if you plan to develop, Git is highly recommended
- ***ns-3 typically generates raw output data*** and relies on various other tools to visualize or process the data

Outline of this tutorial

The tutorial will be example driven

1. Getting ns-3 up and running
2. Basic concepts of ns-3's discrete-event simulation
3. Detailed walkthrough of a simple Wi-Fi example program
4. Examples and descriptions of additional Wi-Fi model features
5. Progressing from examples to validation to developing new algorithms

Prerequisites

- Some experience with command-line or IDE coding on Linux or macOS
- Some experience with or understanding of C++
- Basic understanding of Wi-Fi networks
- New users are recommended to work through the ns-3 tutorial
  - HTML: https://www.nsnam.org/docs/tutorial/html/index.html
  - PDF: https://www.nsnam.org/docs/tutorial/ns-3-tutorial.pdf

Obtaining ns-3

- Most resources are linked from the ns-3 main website at https://www.nsnam.org
- ns-3 is developed and maintained on GitLab.com at https://gitlab.com/nsnam/ns-3-dev
- We will use the most recent release of ns-3 (ns-3.36.1)
  - https://www.nsnam.org/release/ns-allinone-3.36.1.tar.bz2
- If you are using an earlier or later version of ns-3, please be aware that some things may have changed
  - Note: ns-3.37 is about to be released

Building ns-3

> (Demo) Download ns-3
> (Demo) Configure ns-3
> (Demo) Build ns-3
> (Demo) Run programs

For more information, read the tutorial Quick Start:
https://www.nsnam.org/docs/tutorial/html/quick-start.html

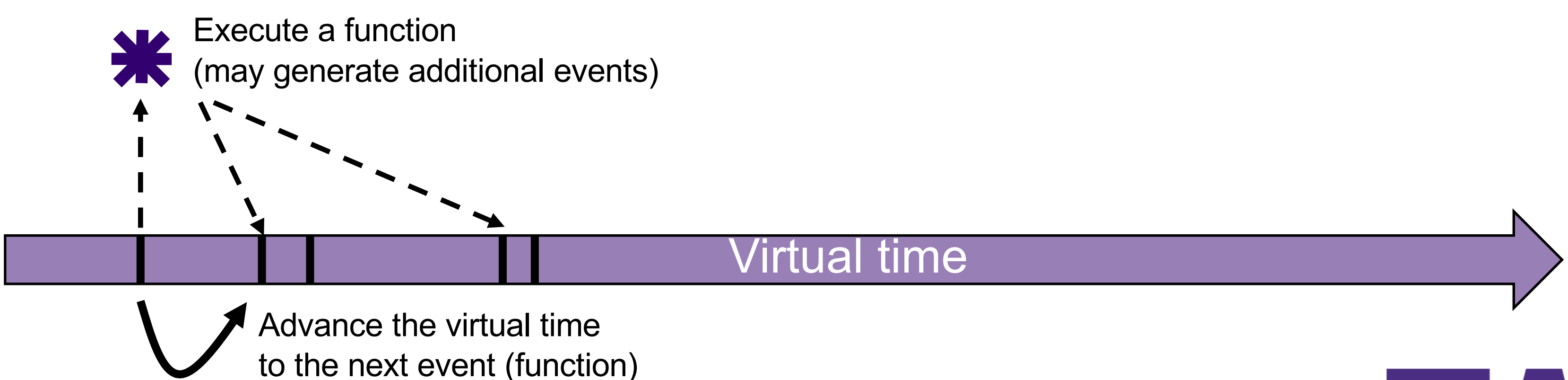
Discrete-event simulation basics

We are trying to represent the operation of a network within a single C++ program

- We need a notion of ***virtual time*** and of ***events*** that occur at specified (virtual) times
- We need a data structure (***scheduler***) to hold all of these events in temporal order
- We need an object (***simulator***) to walk the list of events and execute them at the correct virtual time
- We can choose to ignore things that conceptually might occur between our events of interest, focusing only on the (***discrete***) times with interesting events

Discrete-event simulation basics (cont.)

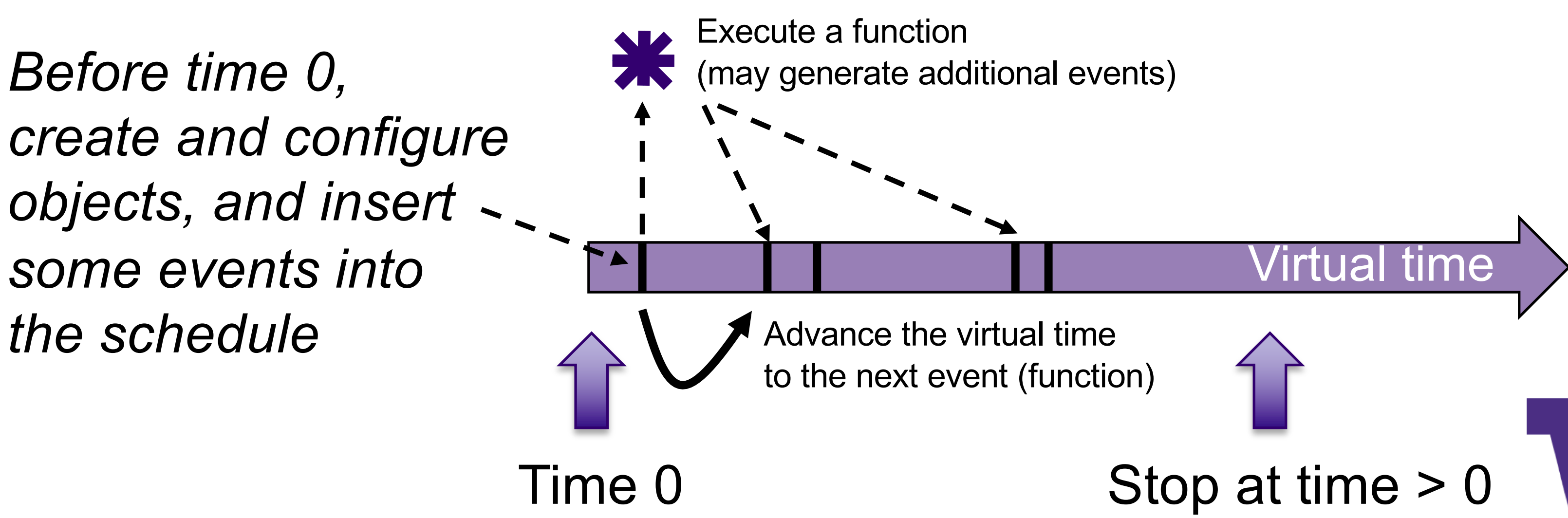
- Simulation time moves in discrete jumps from event to event
- C++ functions schedule events to occur at specific simulation times
- A simulation scheduler orders the event execution
- Simulation::Run() executes a single-threaded event list
- Simulation stops at specified time or when events end



ns-3 simulation basics and terminology

> A simulation ‘run’ or ‘replication’ usually consists of the following workflow

1. Before the notional ‘time 0’, create the scenario objects and pre-populate the scheduler with some initial events
2. Define stopping criteria; either a specific future virtual time, or when certain criteria are met
3. Start the simulation (which initializes objects, at ‘time 0’)



Virtual time in ns-3

- Time is stored as a large integer in ns-3
  - Minimizes floating point differences across platforms
- Special Time classes are provided to manipulate time (such as standard arithmetic operators)
- Default time resolution is nanoseconds, but can be set to other resolutions
  - Note: Changing resolution is not well used/tested
- Time objects can be set by floating-point values and can export floating-point values

```
double timeDouble = t.GetSeconds();
```

  - Best practice is to avoid floating point conversions where possible and use Time arithmetic operators

Key building blocks: Callback and function pointer

> C++ methods are often invoked directly on objects

```
int
main(int argc, char* argv[])
{
    CommandLine cmd(__FILE__);
    cmd.Parse(argc, argv);

    MyModel model;
    Ptr<UniformRandomVariable> v = CreateObject<UniformRandomVariable>();
    v->SetAttribute("Min", DoubleValue(10));
    v->SetAttribute("Max", DoubleValue(20));

    Simulator::Schedule(Seconds(10.0), &ExampleFunction, &model);

    Simulator::Schedule(Seconds(v->GetValue()), &RandomFunction);

    EventId id = Simulator::Schedule(Seconds(30.0), &CancelledEvent);
    Simulator::Cancel(id);
```

Unlike CommandLine::Parse(), we more generally need to call functions at some future (virtual) time.

Some program element could assign a function pointer, and a (later) program statement could call (execute) the method

Program excerpt:
src/core/examples/sample-simulator.cc (lines 101-117)

Events in ns-3

- Events are just functions (callbacks) that execute at a simulated time
  - nothing is special about functions or class methods that can be used as events
- Events have IDs to allow them to be cancelled or to test their status

Simulator and Scheduler

> The Simulator class holds a scheduler, and provides the API to schedule events, start, stop, and cleanup memory

> Several scheduler data structures (calendar, heap, list, map) are possible

> "Realtime" simulation implementation aligns the simulation time to wall-clock time

– two policies (hard and soft limit) available when the simulation and real time diverge

(Demo) sample-simulator.cc

```
int
main(int argc, char* argv[])
{
    CommandLine cmd(__FILE__);
    cmd.Parse(argc, argv);

    MyModel model;
    Ptr<UniformRandomVariable> v = CreateObject<UniformRandomVariable>();
    v->SetAttribute("Min", DoubleValue(10));
    v->SetAttribute("Max", DoubleValue(20));

    Simulator::Schedule(Seconds(10.0), &ExampleFunction, &model);

    Simulator::Schedule(Seconds(v->GetValue()), &RandomFunction);

    EventId id = Simulator::Schedule(Seconds(30.0), &CancelledEvent);
    Simulator::Cancel(id);

    Simulator::Schedule(Seconds(25.0), []() {
        std::cout << "Code within a lambda expression at time " << Simulator::Now().As(Time::S)
                  << std::endl;
    });

    Simulator::Run();

    Simulator::Destroy();
}
```

Program excerpt:
src/core/examples/sample-simulator.cc (lines 101-117)

CommandLine arguments

- Add CommandLine to your program if you want command-line argument parsing

```
CommandLine cmd(__FILE__);
cmd.AddValue("maxSsrc",
             "The maximum number of retransmission attempts for a RTS packet",
             maxSsrc);
cmd.AddValue("maxSlrc",
             "The maximum number of retransmission attempts for a Data packet",
             maxSlrc);
```

- Passing --PrintHelp to programs will display command line options, if CommandLine is enabled

```
./ns3 run "sample-simulator --PrintHelp"
```

```
sample-simulator [General Arguments]

General Arguments:
    --PrintGlobals:              Print the list of globals.
    --PrintGroups:               Print the list of groups.
    --PrintGroup=[group]:        Print all TypeIds of group.
    --PrintTypeIds:              Print all TypeIds.
    --PrintAttributes=[typeid]:  Print all attributes of typeid.
    --PrintVersion:              Print the ns-3 version.
    --PrintHelp:                 Print this help message.
```

Random Variables and Run Number

- Many ns-3 objects use random variables to model random behavior of a model, or to force randomness in a protocol
  - e.g. random placement of nodes in a topology

- Many simulation uses involve running a number of ***independent replications*** of the same scenario, by changing the random variable streams in use
  - In ns-3, this is typically performed by incrementing the simulation ***run number***

```
./ns3 run 'sample-simulator --RngRun=2'

NS_GLOBAL_VALUE="RngRun=2" ./ns3 run sample-simulator
```

Random Variables

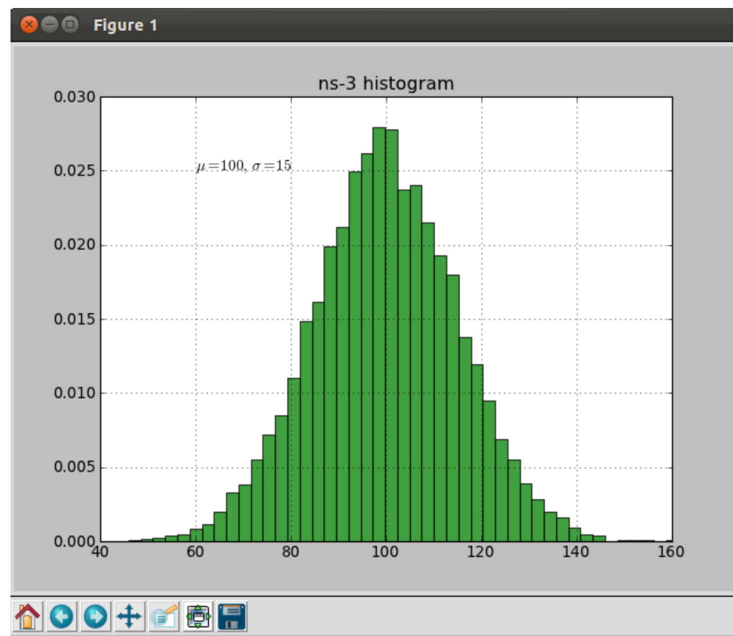
- Currently implemented distributions
  - Uniform: values uniformly distributed in an interval
  - Constant: value is always the same (not really random)
  - Sequential: return a sequential list of predefined values
  - Exponential: exponential distribution (poisson process)
  - Normal (gaussian), Log-Normal, Pareto, Weibull, Triangular, Zipf, Zeta, Deterministic, Empirical

```python
#  Demonstrate use of ns-3 as a random number generator integrated with
#  plotting tools.

import numpy as np
import matplotlib.pyplot as plt
import ns.core

# mu, var = 100, 225
rng = ns.core.NormalRandomVariable()
rng.SetAttribute("Mean", ns.core.DoubleValue(100.0))
rng.SetAttribute("Variance", ns.core.DoubleValue(225.0))
x = [rng.GetValue() for t in range(10000)]

## Make a probability density histogram
density = 1
facecolor='g'
alpha=0.75
# n, bins, patches = plt.hist(x, 50, density=1, facecolor='g', alpha=0.75)
plt.hist(x, 50, density=1, facecolor='g', alpha=0.75)
plt.title('ns-3 histogram')
plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
plt.show()
```



from src/core/examples/sample-rng-plot.py

Discrete-event simulation basics

- Events, scheduler, simulation time, random variables (✓)
- Program output
- Packets
- Nodes, NetDevices
- MobilityModel/Position

Program output options

In general, ns-3 does not output data by default-- must be configured to do so

> Raw logs

> Trace sources

> PCAP, ASCII, and athstats traces

> Animation traces

> Gnuplot plot files

Output file formats

- PCAP (packet capture) files can be read by programs like ***tcpdump*** and ***wireshark***
- ASCII trace files are plain-text representations of packet transmissions, receptions, and drops
  - Example trace file from example `./ns3 run mixed-wireless`

```
t 0.00183903 /NodeList/2/$ns3::Ipv4L3Protocol/Tx(2) ns3::Ipv4Header (tos 0x0 DSC
P Default ECN Not-ECT ttl 1 id 0 protocol 17 offset (bytes) 0 flags [none] lengt
h: 68 172.16.2.1 > 172.16.2.255) ns3::UdpHeader (length: 48 698 > 698) ns3::olsr
::PacketHeader (len: 40 seqNo: 0) ns3::olsr::MessageHeader (type: HELLO TTL: 1 O
rig: 192.168.0.3 SeqNo: 0 Validity: 134 Hop count: 0 Size: 16 Interval: 5 (2s) W
illingness: 3) ns3::olsr::MessageHeader (type: MID TTL: 255 Orig: 192.168.0.3 Se
qNo: 1 Validity: 231 Hop count: 0 Size: 20 [172.16.2.1, 10.0.2.1])
```

- Athstats traces use Wi-Fi trace sources to provide debugging similar to Madwifi drivers

Example athstats output from example `./ns3 run wifi-ap`

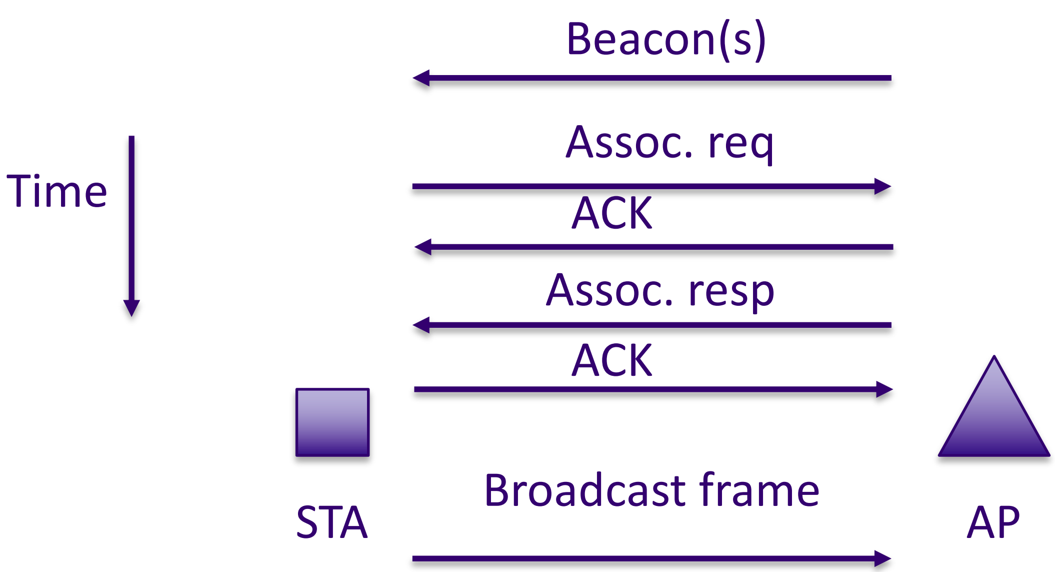
```
m_txCount  m_rxCount  unused  short  long exceeded rxError

    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
```

(Program output demo) wifi-simple-infra.cc

```
./ns3 run wifi-simple-infra
```

- Show logs
- Program output (pcap)
- View Wireshark



Packets

- A packet is a special data buffer with space for ***headers***, ***trailers***, ***tags***, and ***metadata***

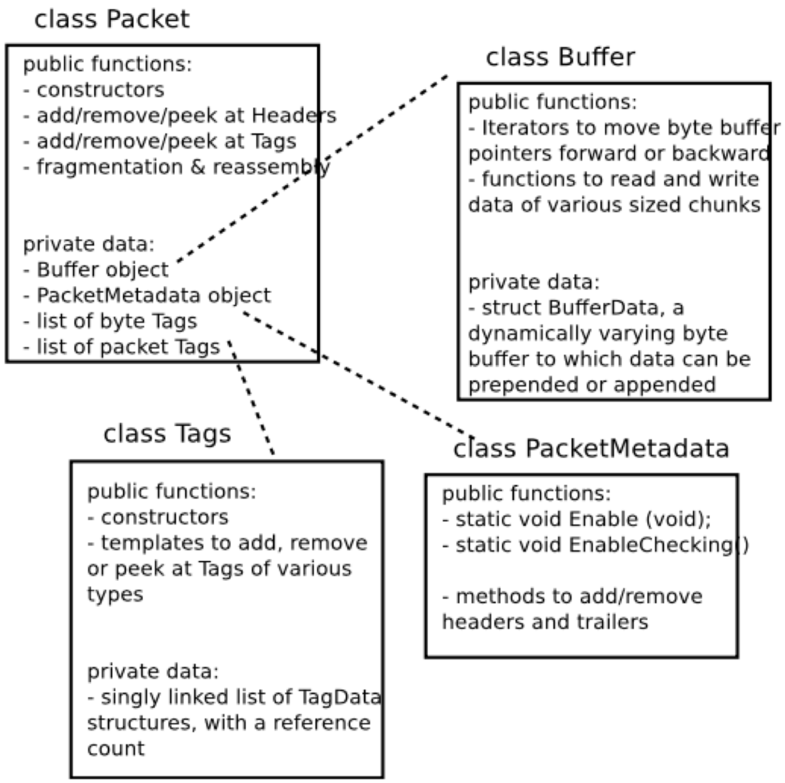
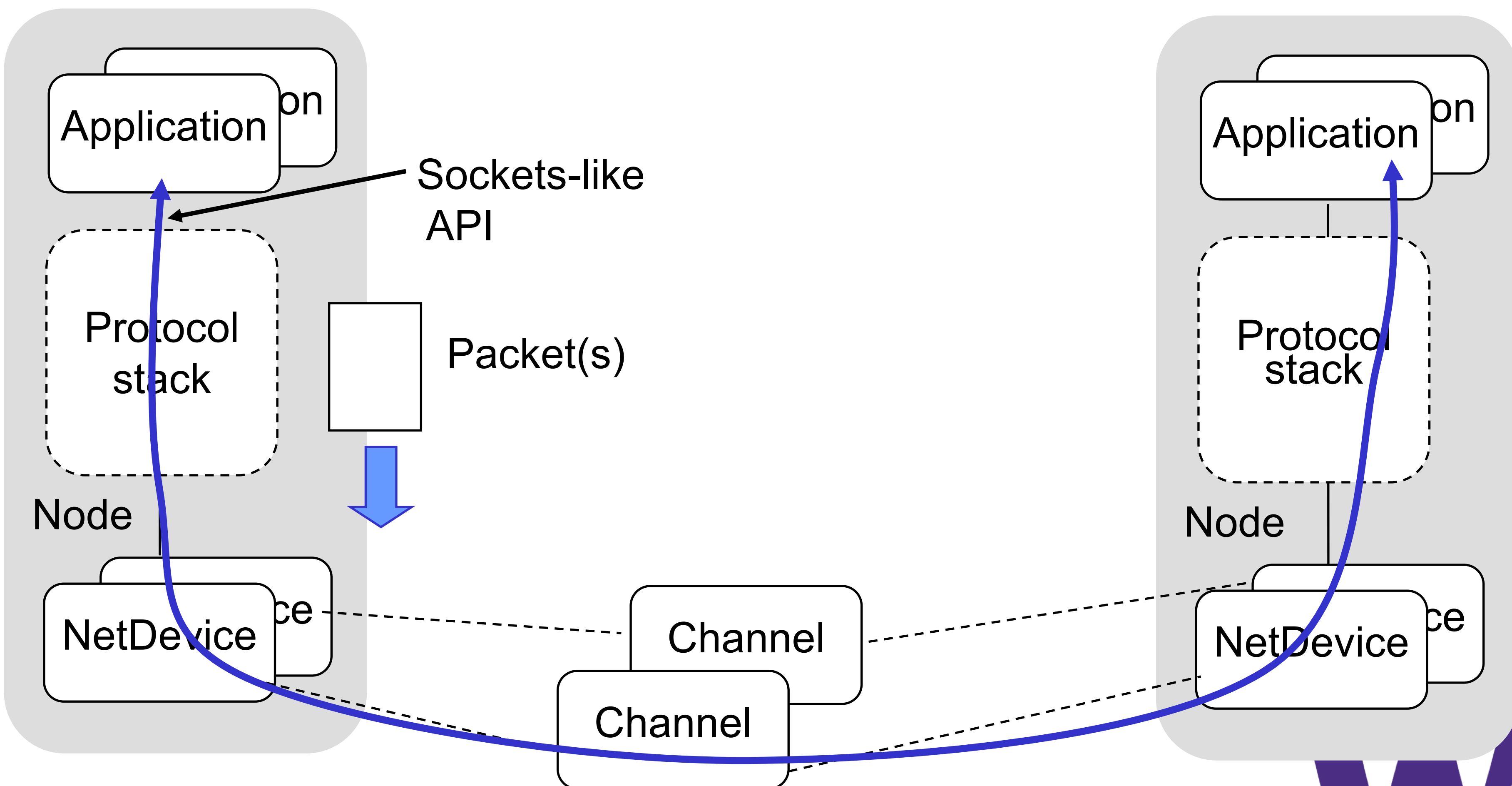


Fig. 1: Implementation overview of Packet class.

- Figure source: ns-3 Model Library documentation

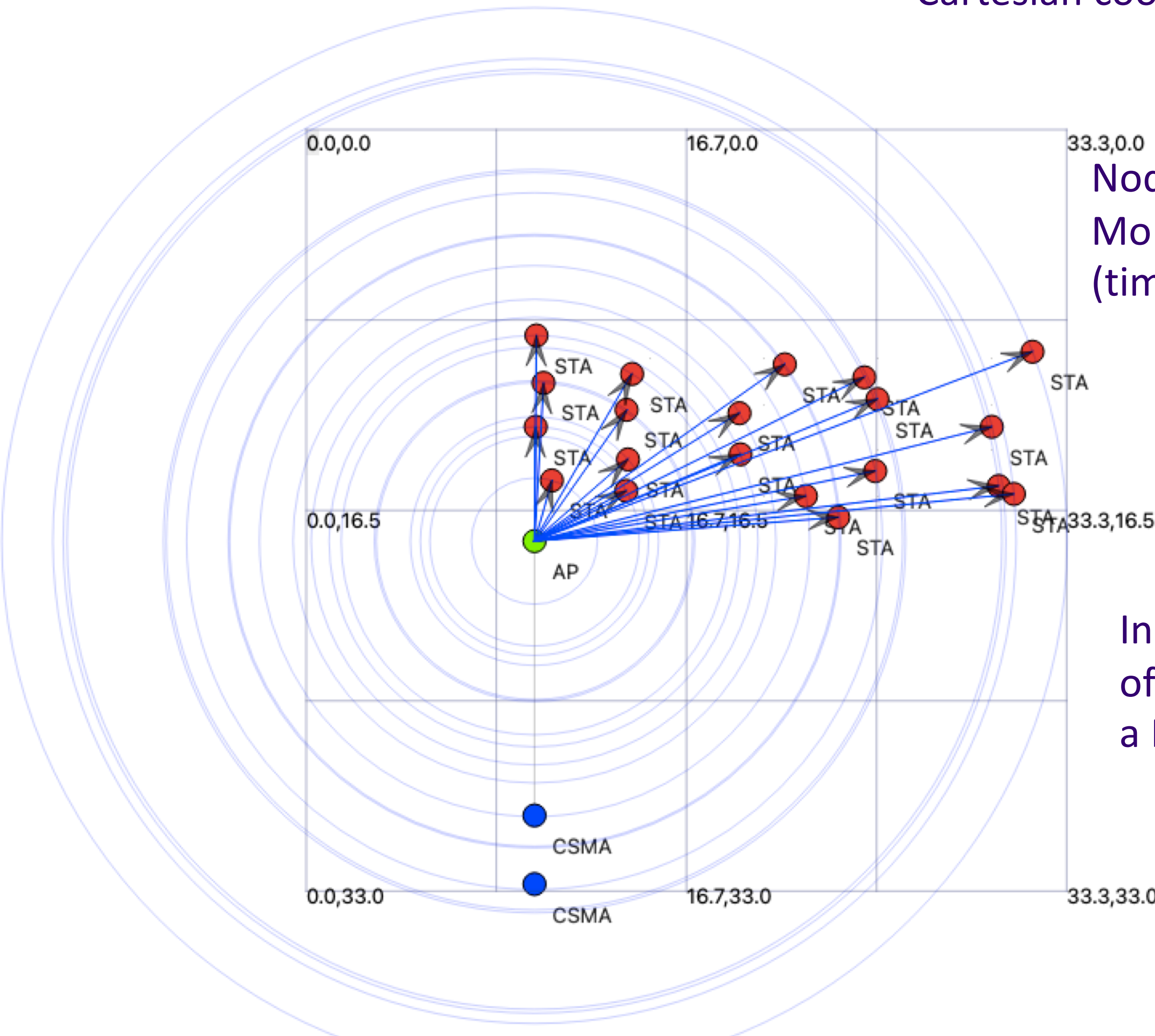
Nodes, Applications, NetDevices

- Most simulations involve packet exchanges such as depicted below



Mobility and position

Cartesian coordinate system



Nodes typically have a MobilityModel with a (time-varying) Position

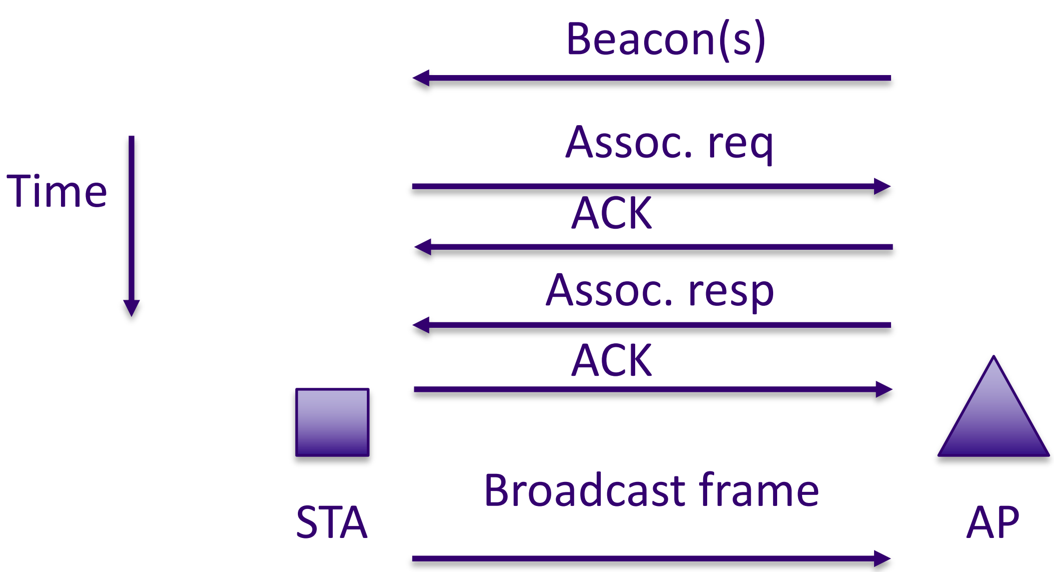
Initial positions are often allocated by a PositionAllocator

Mobility and position

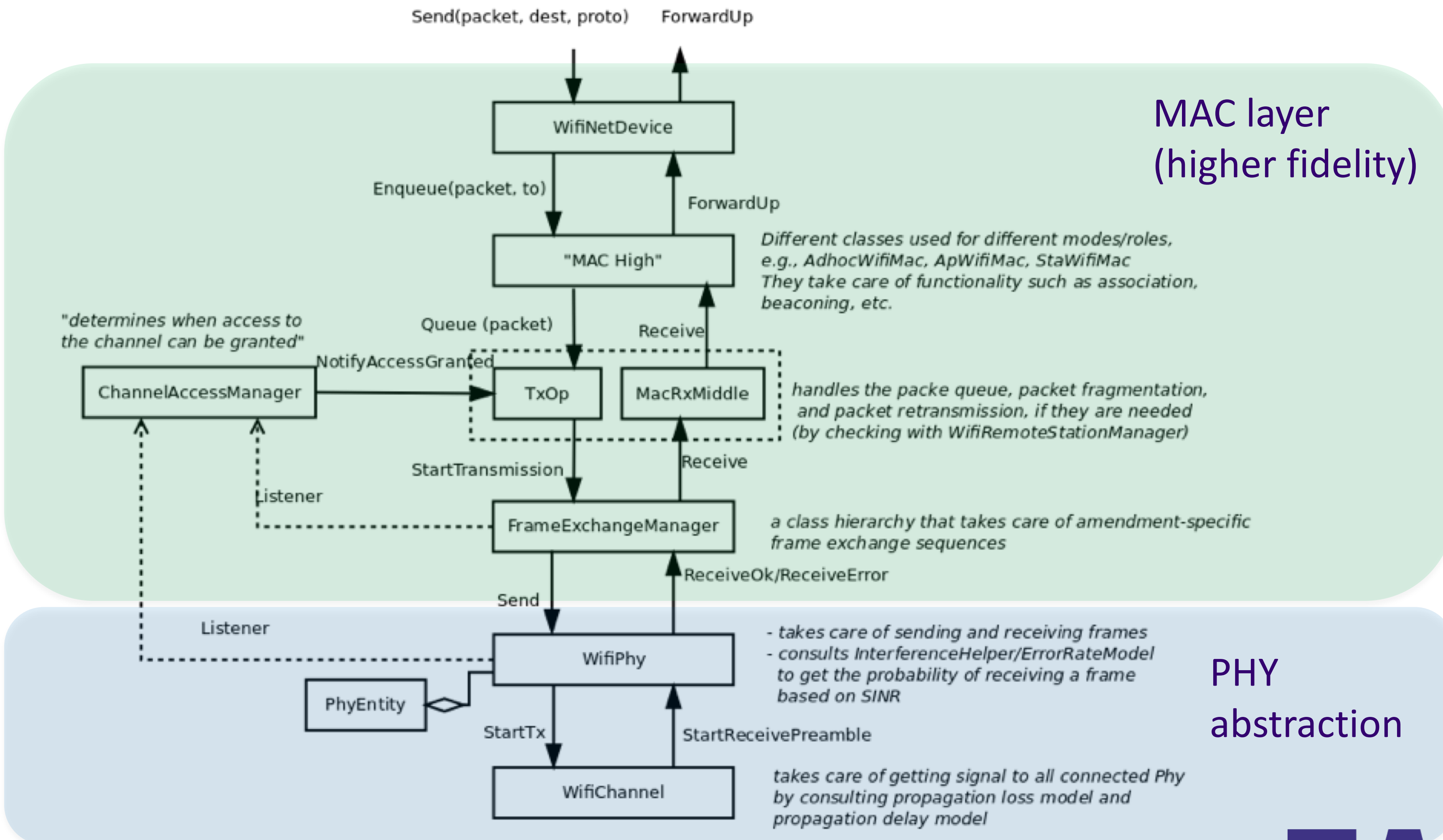
- ns-3 position is represented on a 3D Cartesian (x,y,z) coordinate system
- The MobilityHelper combines a **mobility model** and **position allocator**.
- Position Allocators setup initial position of nodes (only used when simulation starts):
  - **List:** allocate positions from a deterministic list specified by the user;
  - **Grid:** allocate positions on a rectangular 2D grid (row first or column first);
  - **Random position allocators:** allocate random positions within a selected form (rectangle, circle, …).
- Mobility models specify how nodes will move during the simulation:
  - **Constant:** position, velocity or acceleration;
  - **Waypoint:** specify the location for a given time (time-position pairs);
  - **Trace-file based:** parse files and convert into ns-3 mobility events, support mobility tools such as SUMO, BonnMotion (using NS2 format) , TraNS

Recap

- Events, scheduler, simulation time, random variables (✓)
- Program output (✓)
- Packets (✓)
- Nodes, NetDevices (✓)
- MobilityModel/Position (✓)



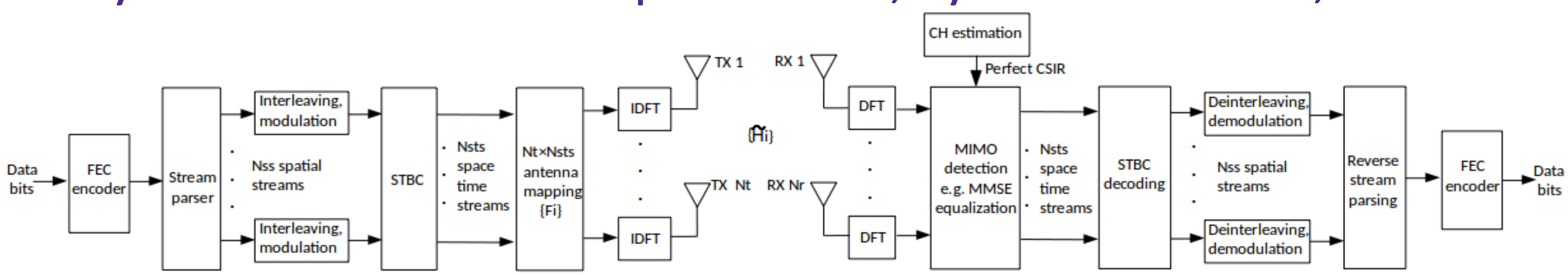
ns-3 Wi-Fi MAC and PHY layers



WifiNetDevice architecture

PHY abstraction

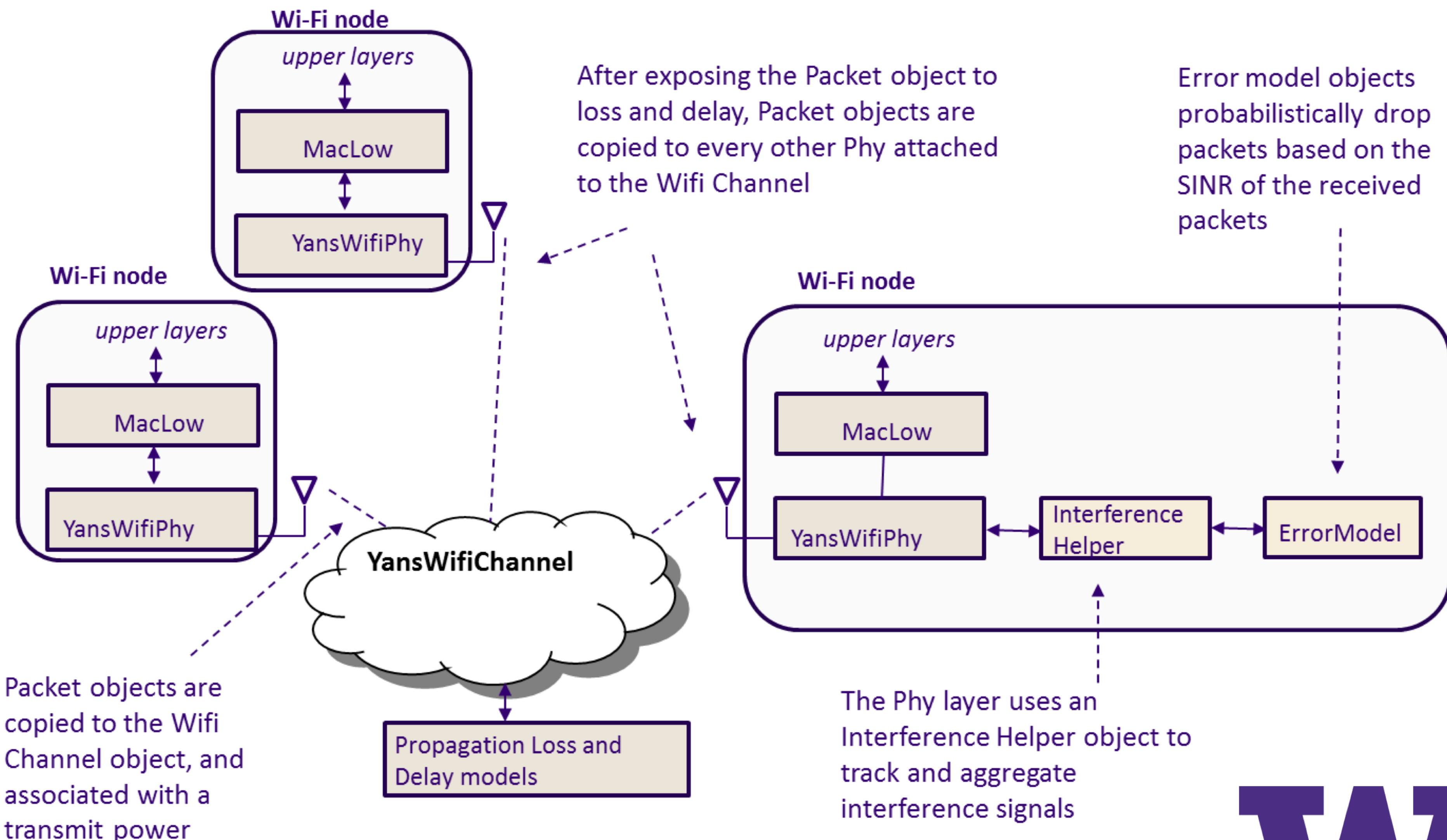
> A full physical layer model would decompose packets into RF symbols and model equalization, synchronization, etc.



– Figure source: Sian Jin et al, 2020 Workshop on ns-3 paper.

> In ns-3, "packet" objects are transmitted over channels that add propagation losses, SNRs are calculated, and SNRs are translated to packet error ratios (PERs)

ns-3 PHY abstraction



Wi-Fi channels

> Two options are supported:

1. YansWifiChannel (simple single-band model)
   - Use if there is no frequency-selective fading model, and if there is no interference from foreign sources
   - Default YansWifiChannelHelper will add a "LogDistancePropagationLossModel" with path loss exponent value of 3
2. SpectrumChannel (fine-grained band decomposition)
   - Use if more detailed frequency selective models are needed, or in a mixed-signal environment
   - Default SpectrumWifiChannelHelper wil add a "FriisSpectrumPropagationLossModel" (power falls as square of distance)

SpectrumChannel illustration

> Figure source: ns-3 Model Library documentation

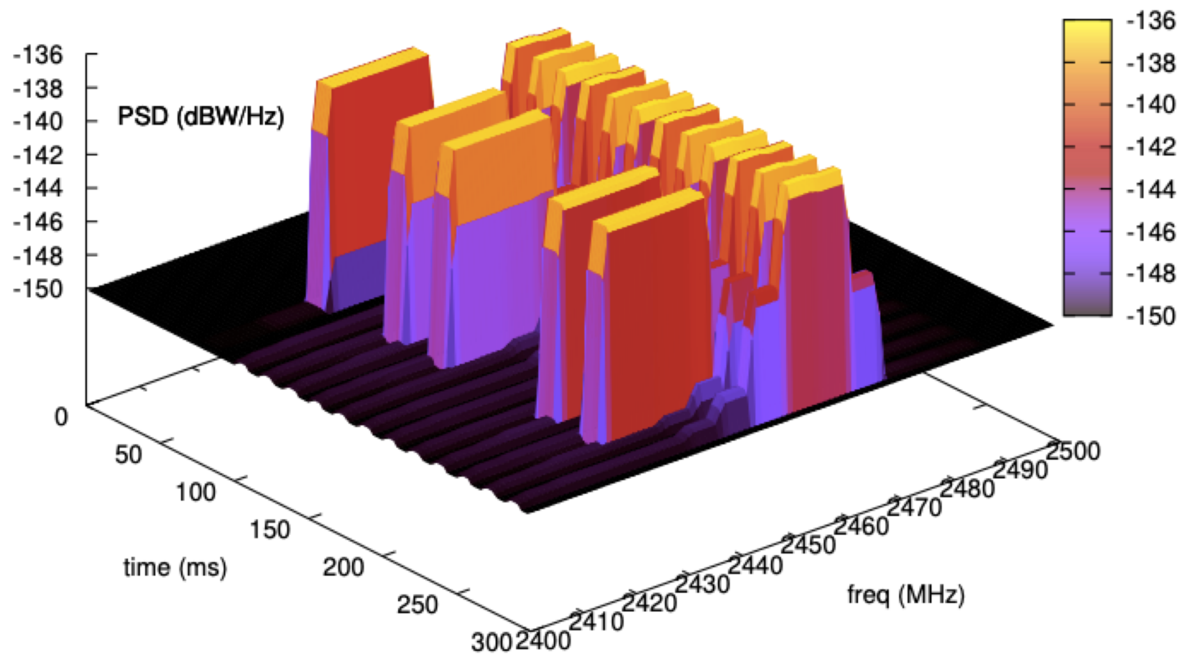


Fig. 1: Spectrogram produced by a spectrum analyzer in a scenario involving wifi signals interfered by a microwave oven, as simulated by the example `adhoc-aloha-ideal-phy-with-microwave-oven`.

Propagation

> Propagation module defines:
  - Propagation loss models:
    Calculate the Rx signal power considering the Tx signal power and the respective Rx and Tx antennas positions.
  - Propagation delay models:
    Calculate the time for signals to travel from the TX antennas to RX antennas.

> Propagation delay models almost always set to:
  - ConstantSpeedPropagationDelayModel: In this model, the signal travels with constant speed (defaulting to speed of light in vacuum)

Propagation (cont.)

> Propagation loss models:

- Many propagation loss models are implemented:
  - ✓ Abstract propagation loss models:
    FixedRss, Range, Random, Matrix, …
  - ✓ Deterministic path loss models:
    Friis, LogDistance, ThreeLogDistance, TwoRayGround, …
  - ✓ Stochastic fading models:
    Nakagami, Jakes, …

Propagation (cont.)

– A propagation loss model can be "chained" to another one, making a list. The final Rx power takes into account all the chained models.
Example: path loss model + shadowing model + fading model

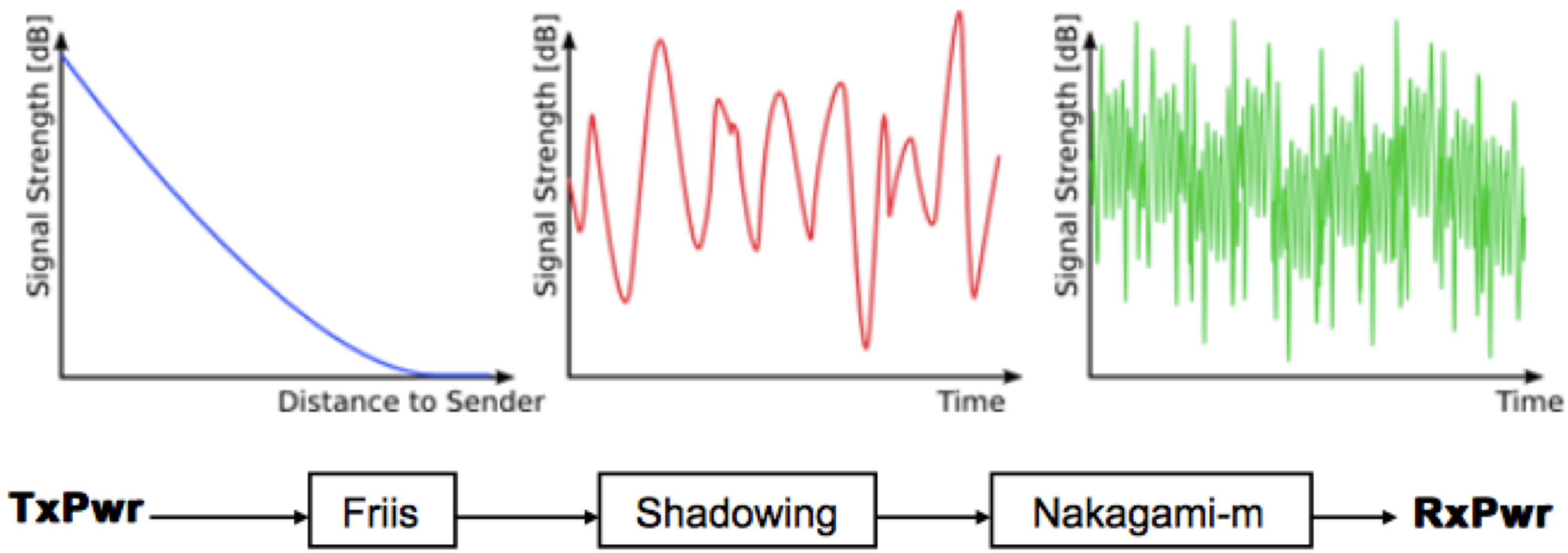
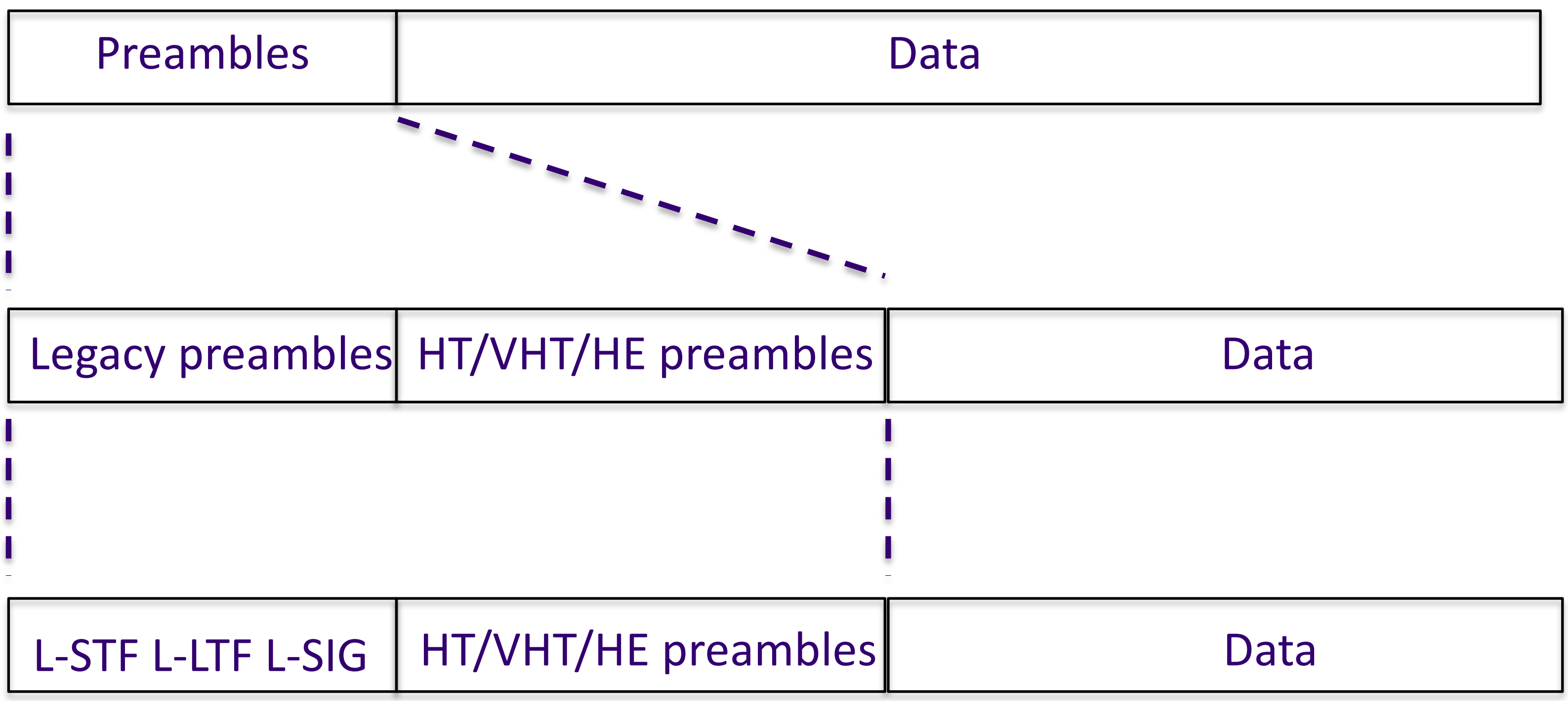


Figure source: Unknown

Preamble detection and frame capture models

> In practice, a WiFi frame is first detected (and synchronized) via a ***preamble*** field

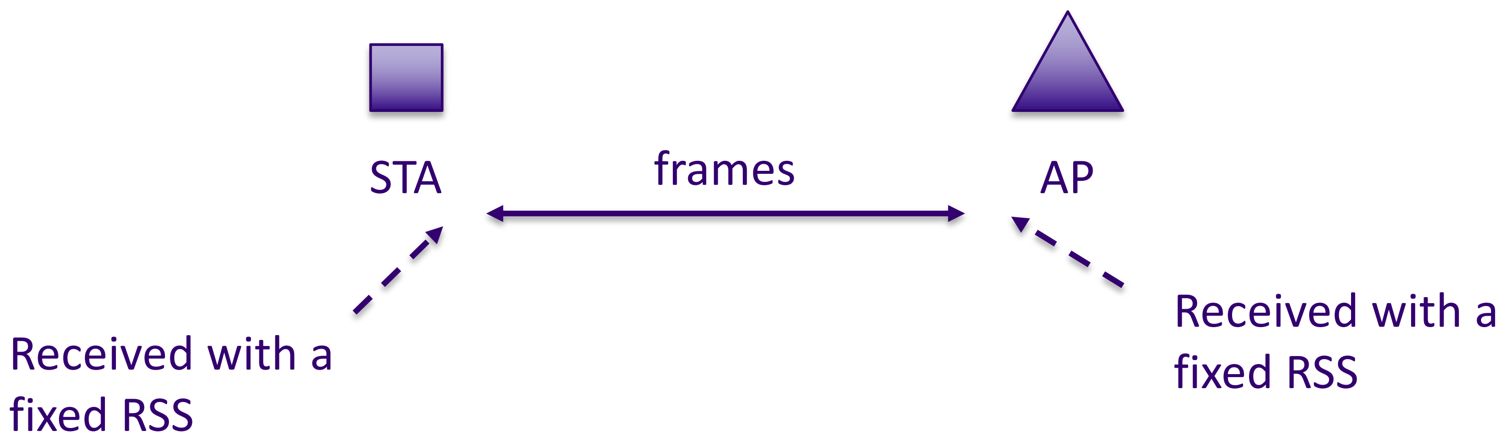


Preamble detection and frame capture models

- A **‘ThresholdPreambleDetectionModel’** is configured by default by the Wi-Fi helpers
  - ”Threshold” attribute: default 4 dB
  - “MinimumRssi” attribute: default -82 dBm
- A **‘SimpleFrameCaptureModel’** is available but must be added (WifiHelper::SetFrameCaptureModel())
  - Only enabled for YansWifiPhyHelper
  - “Window” attribute: default 16us
  - “Margin” attribute: default 5 dB

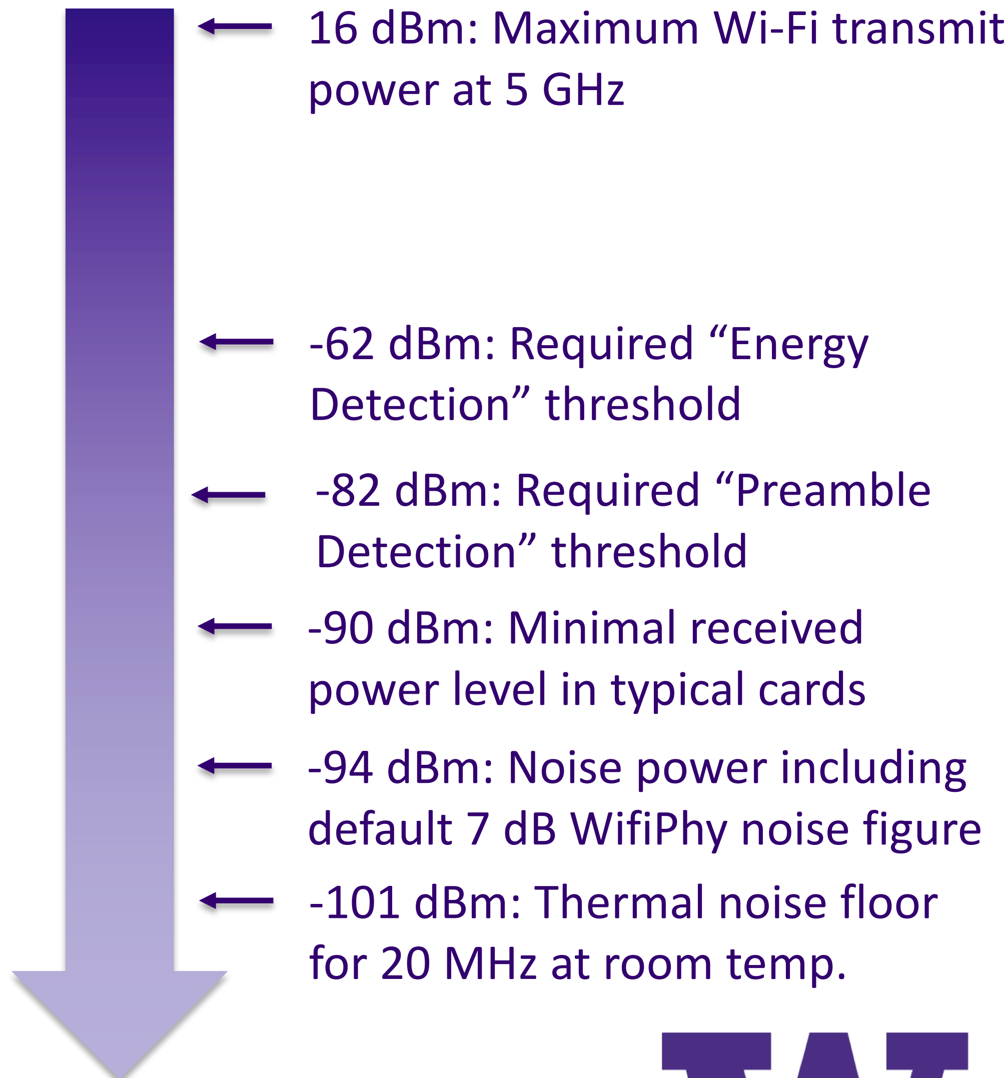
(Demo) wifi-simple-infra.cc

- wifi-simple-infra.cc uses a special ‘FixedRss’ propagation loss model that enforces that the received signal strength (RSS) is a configured value
- Packet delivery is governed by a **preamble detection model** and a **Wi-Fi error model**



Signal strength and Wi-Fi

> dBm is reference to decibels over 1 mW
> 0 dBm = 1 mW
> +/- 3 dB = */÷ a factor of 2 on a linear scale
> +/- 10 dB = */÷ a factor of 10 on a linear scale

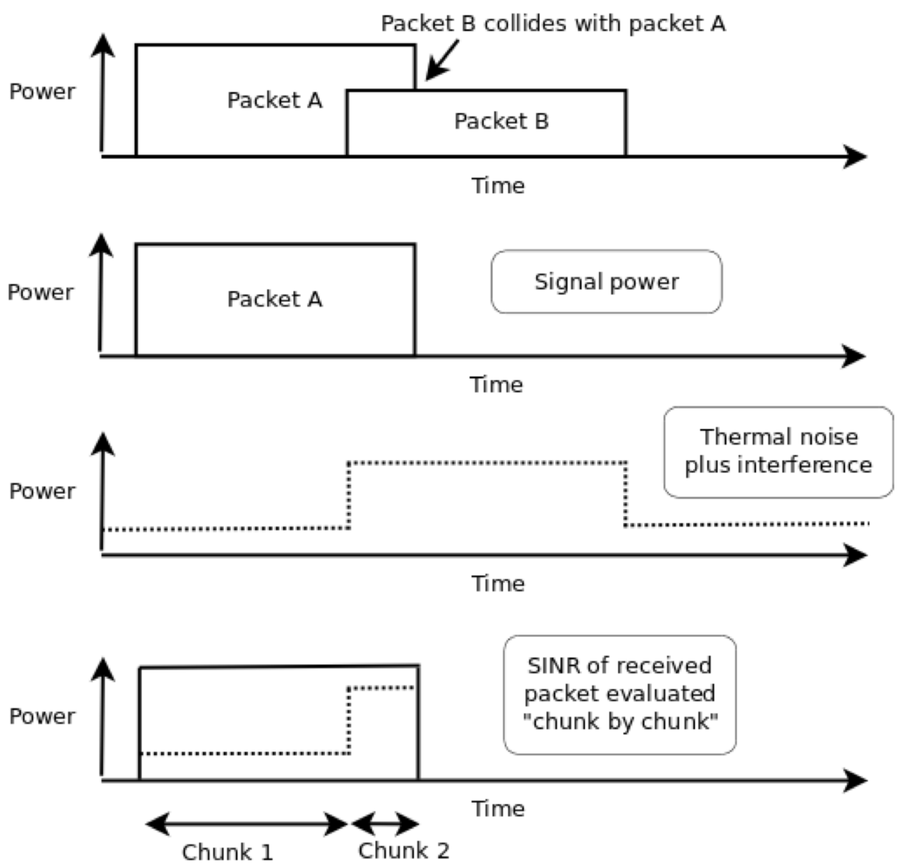


Signal to noise ratio

- (Signal + gain) power/(Noise + interference) power
- Typically expressed in decibels (dB)
- 0 dB -> signal power equals the noise power (ratio of 1)
- Different modulations require different levels of SNR to decode successfully
- Gains (e.g. directional antennas, amplifiers) can contribute to the numerator
- Propagation losses reduce the signal power at the receiver
- Thermal noise and noise figure contribute to the denominator

Interference handling and error models

- Interference (if any) is handled by adding the interfering signal's power to the noise power
- Figure source: ns-3 Model Library documentation



SNIR function over time

Error models

- DSSS error models are derived analytically
  - See: https://www.nsnam.org/~pei/80211b.pdf
- OFDM error models are derived from MATLAB(TM) Wireless LAN System Toolbox
  - See: https://depts.washington.edu/funlab/wp-content/uploads/2017/05/Technical-report-on-validation-of-error-models-for-802.11n.pdf
- P_{error} (probability of packet error)

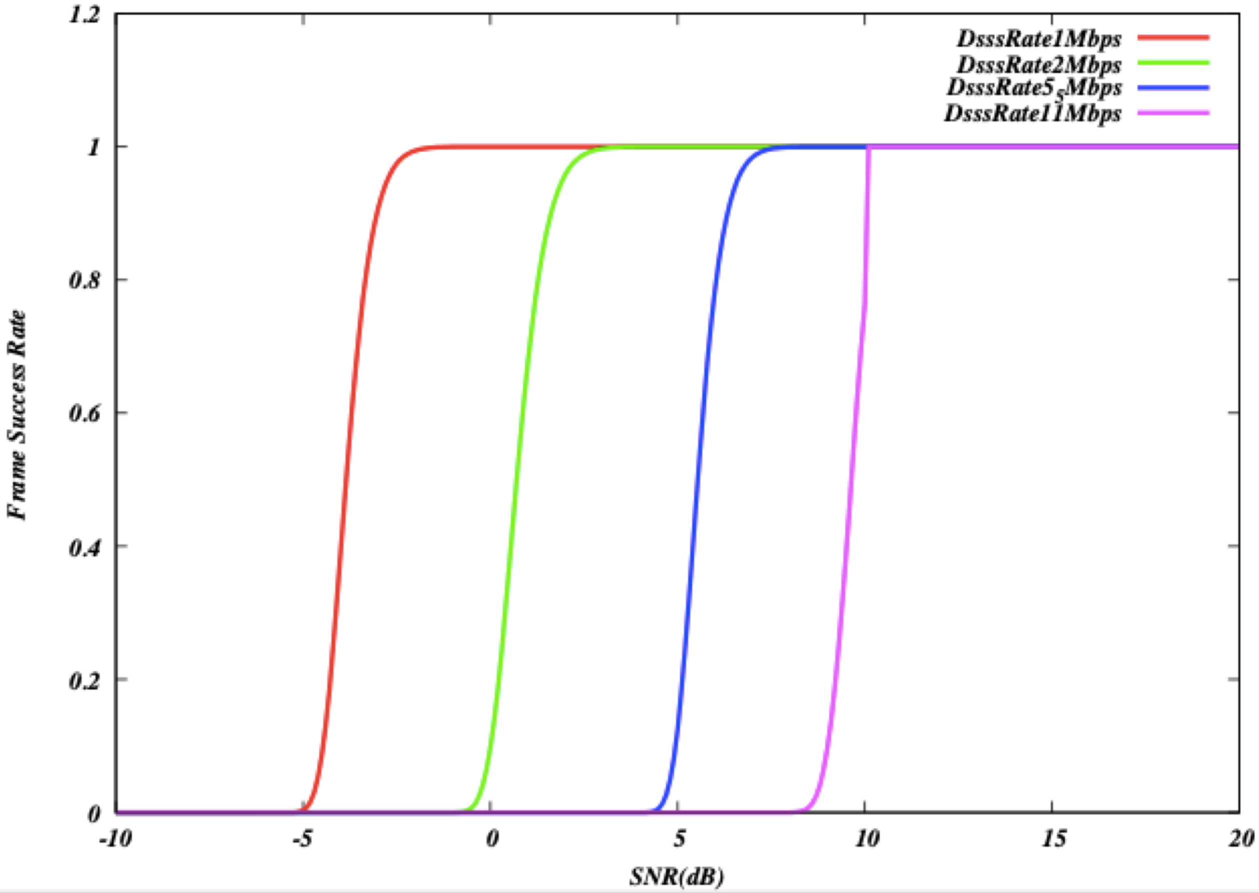
  $= 1 - (P_{success1})(P_{success2})(P_{success3})\ldots$ (for all chunks)
- $P_{success}$ (N-bit chunk at given BER)

  $= 1 - (1 - BER)^N$

Example PER curves

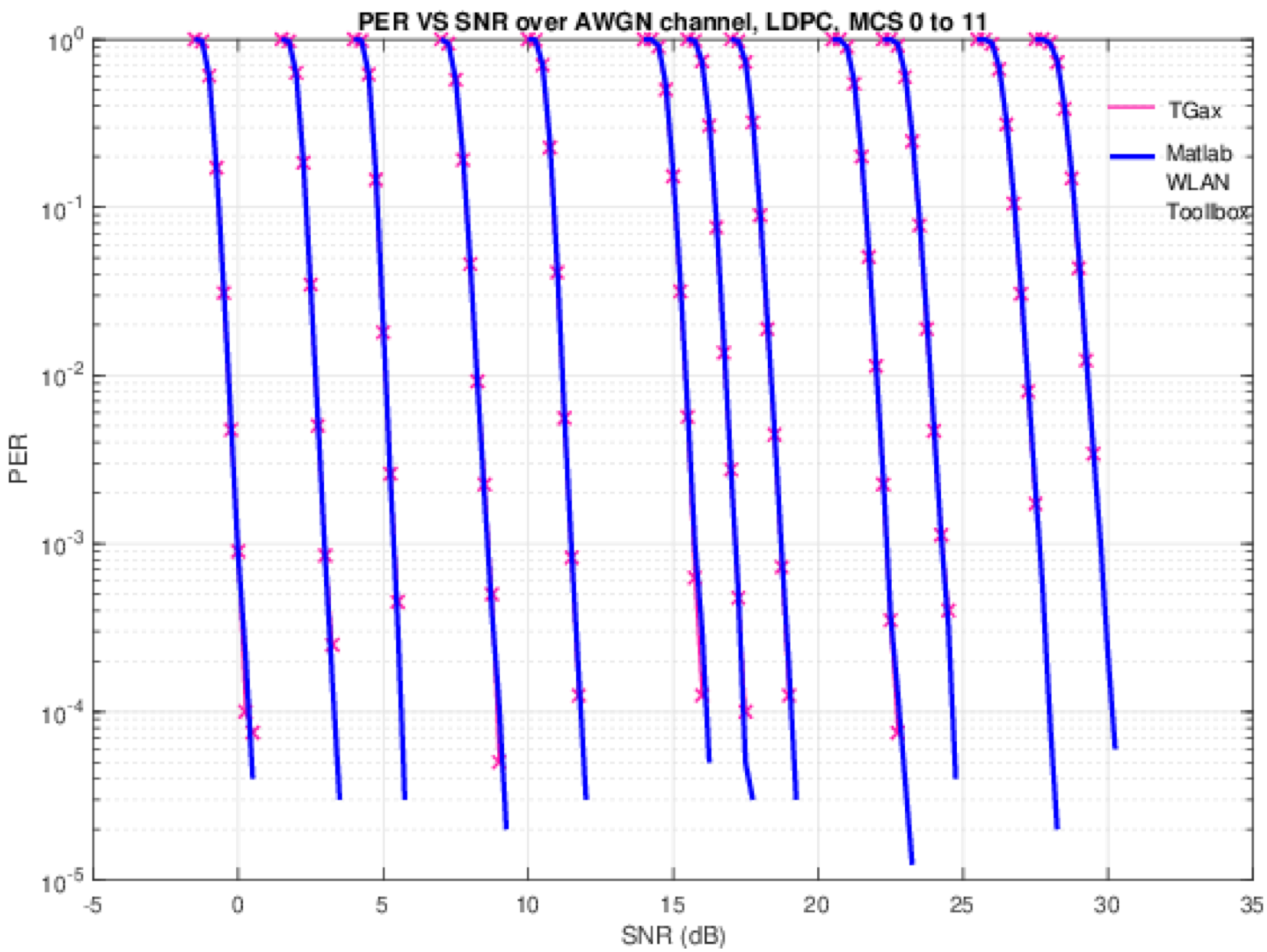
> Figure from ‘examples/wireless/wifi-dsss-validation.cc’

```
$ ./ns3 run wifi-dsss-validation
$ gnuplot frame-success-rate-dsss.plt
```



Calibration against link simulations

> OFDM AWGN error models have been generated and tabulated using MATLAB link simulations



Fading channels require special techniques

> Link-to-system mapping (LSM) is a technique to use link simulations of more complicated frequency-selective fading channels to map received SNR to a different "effective" SNR
  - See https://www.nsnam.org/research/wns3/wns3-2021/tutorials/

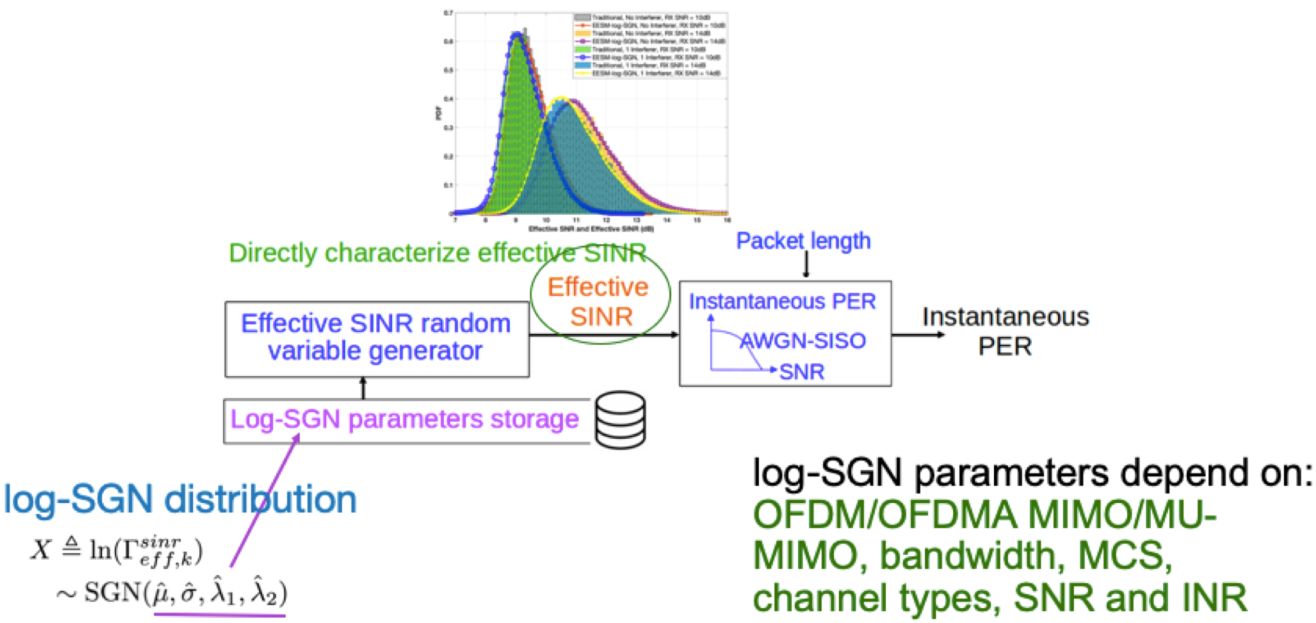
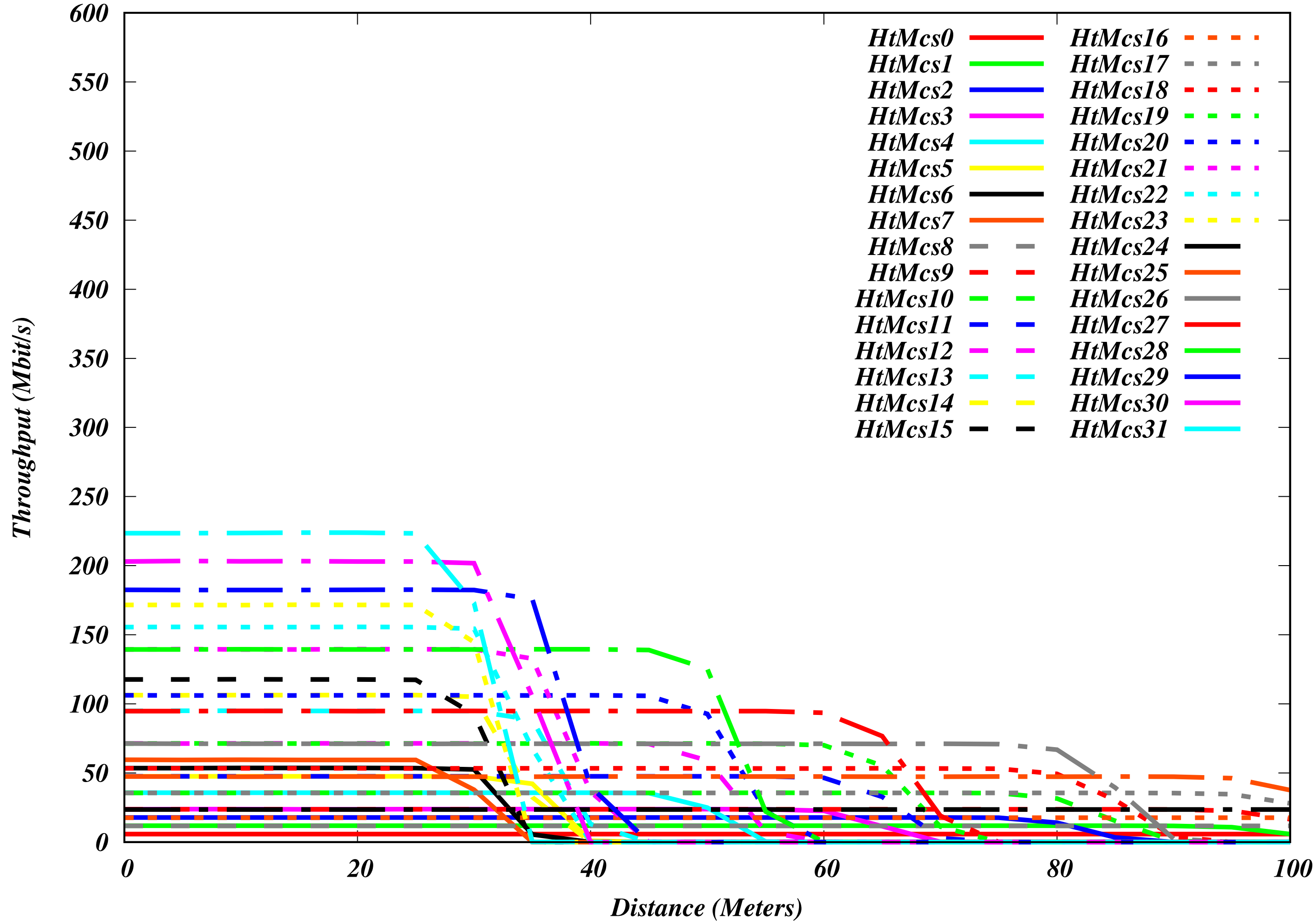


Figure source: Sian Jin, WNS3 2021 PHY abstraction tutorial

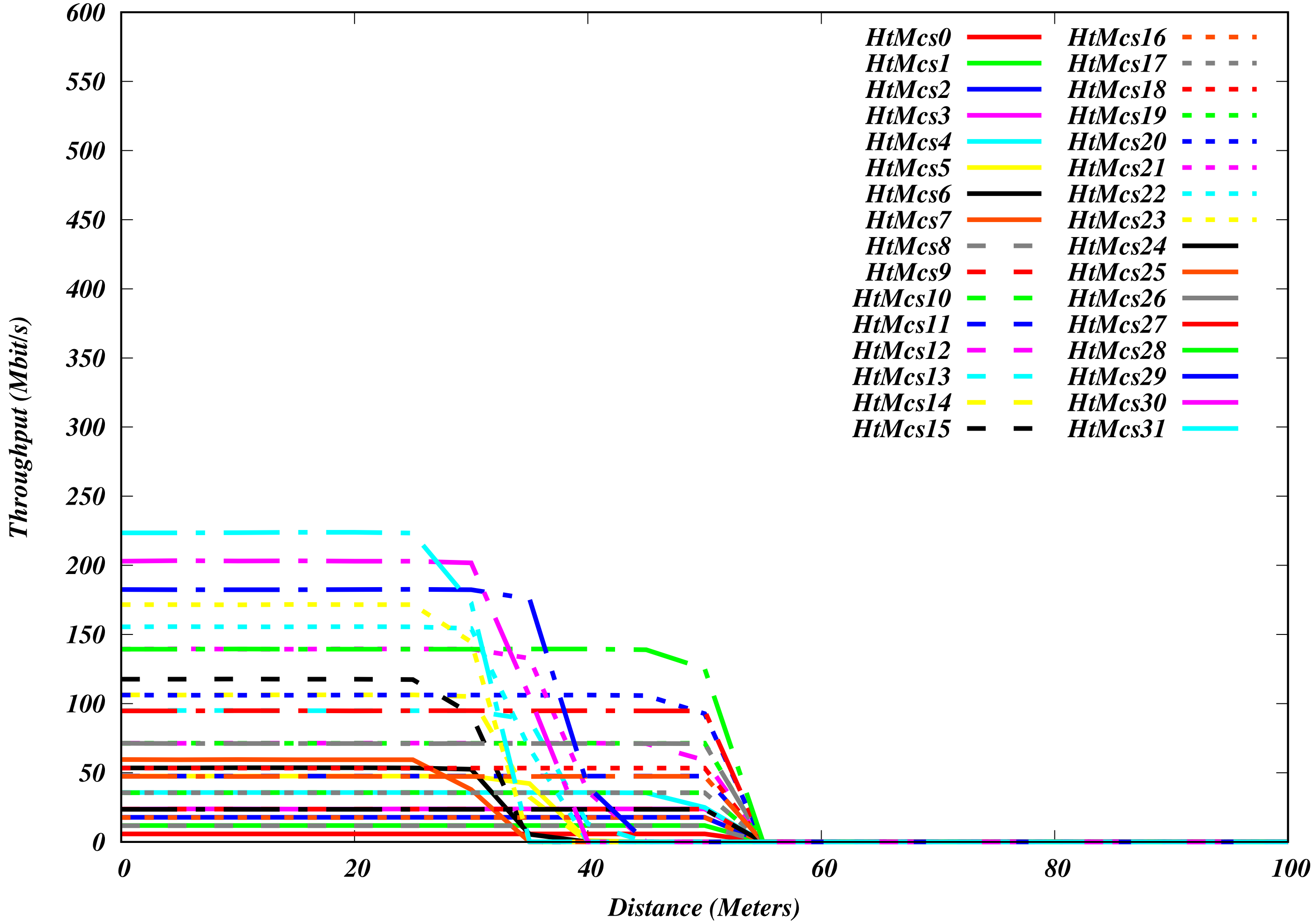
Throughput vs distance for 802.11n modulation

```
./ns3 run 'wifi-80211n-mimo --preambleDetection=0'
```



Throughput vs distance for 802.11n modulation

```
./ns3 run 'wifi-80211n-mimo --preambleDetection=1'
```



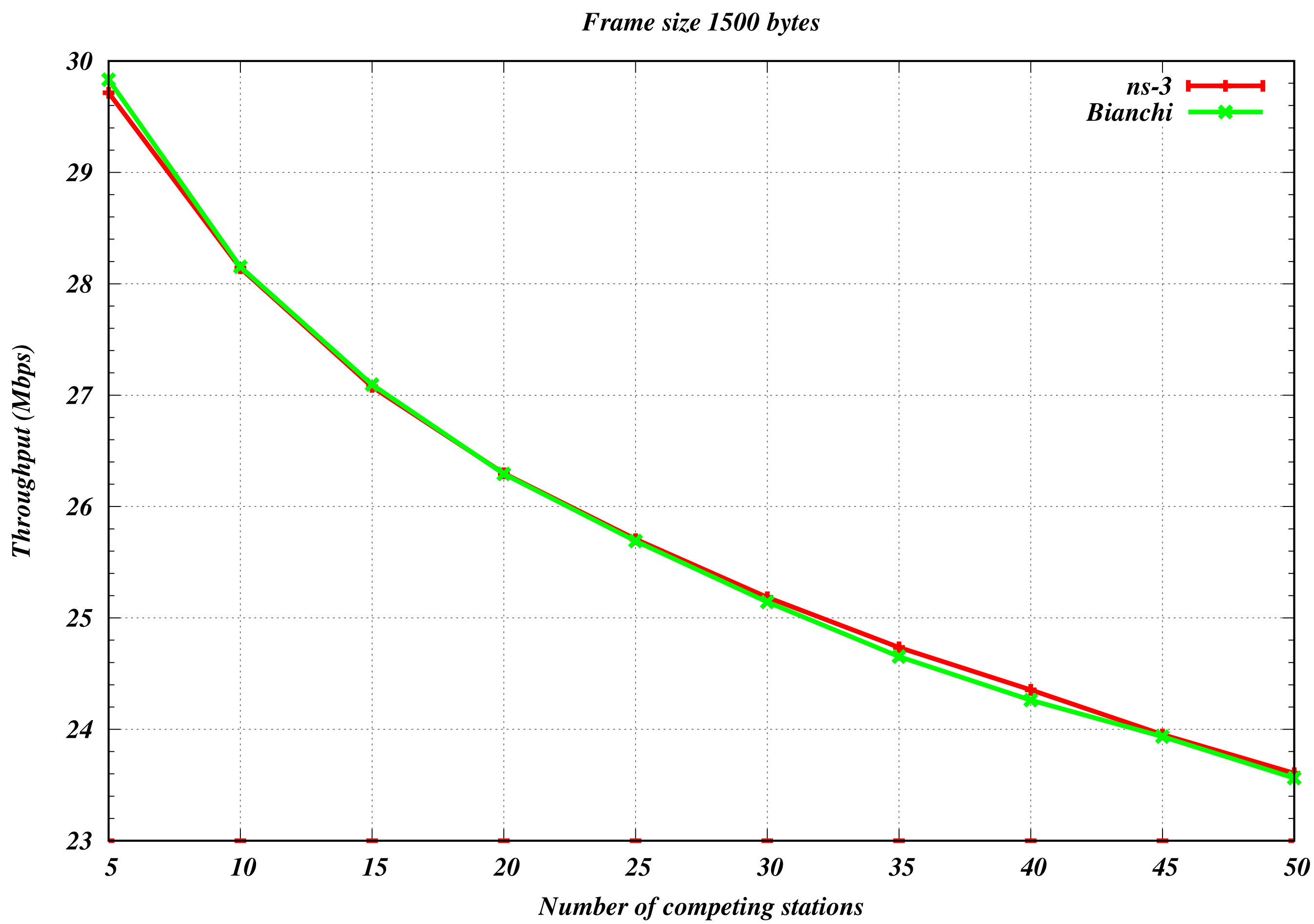
Bianchi analysis/validation

- Analytical work by Bianchi [*] bounded the performance of the Wi-Fi DCF under saturating traffic
- ns-3 simulations (src/wifi/examples/wifi-bianchi.cc) have been used to validate the simulator against this analysis, for many versions of the Wi-Fi standard
  - accounting for differences in overhead and operation

[*] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," in *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, pp. 535-547, March 2000

Example Bianchi plot

- Default results for 802.11a, 5 to 50 nodes, adhoc network
- ./ns3 run ‘wifi-bianchi’



Rate control

- Rate control refers to the algorithm used to select an appropriate modulation and coding scheme (MCS) to send a packet (first transmission attempt, and possible retries)
- Rate control is a subset of the generalized problem of resource allocation, which can include also power control, scheduling, carrier sense range, beamforming, etc. and can be performed jointly with rate control (in theory)
- In current practice, either pure rate control (selection of per-packet and per-retry MCS only) or joint rate and power control is reportedly performed
  - Only newer chipsets appear to support per-frame power control, possibly with some switching latency constraints

Rate control in ns-3

- ns-3 has many rate control algorithms that only work for non-HT (i.e., pre-802.11n) Wi-Fi, including joint rate/power controls
- For 802.11n and newer, there are only four rate controls available
    1) ConstantRateWifiManager
    2) IdealWifiManager
    3) MinstrelHtWifiManager
    4) ThomsonSamplingWifiManager

ConstantRateWifiManager

- Many ns-3 programs disable dynamic rate control and provide specific rates for both the data and control/management frames
- Sample code is shown below:

```
std::ostringstream oss;
oss << "HeMcs" << mcs;
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager","DataMode", StringValue (oss.str ()),
                              "ControlMode", StringValue (oss.str ()));
```

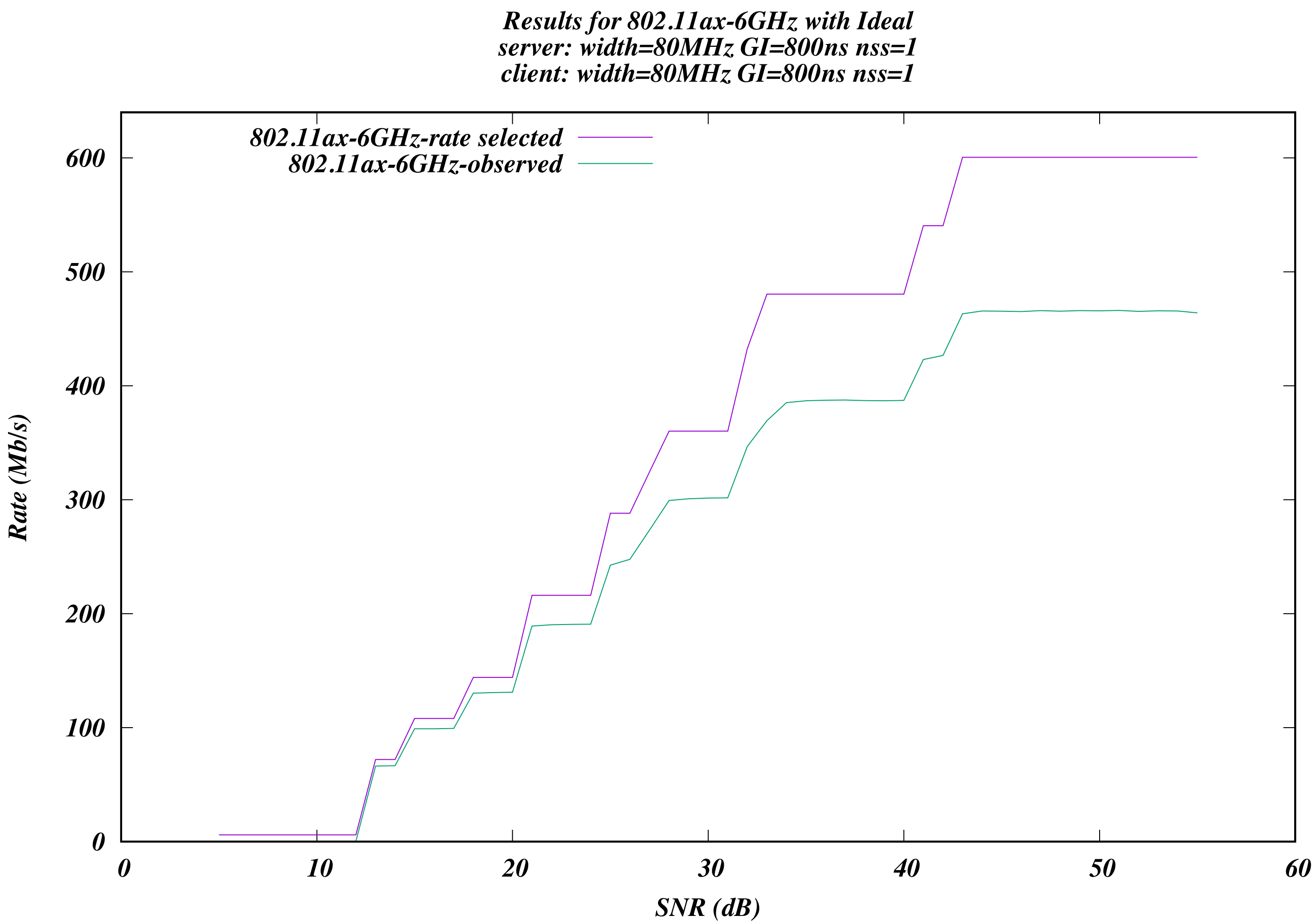
IdealWifiManager

- ns-3 contains an idealized dynamic rate control manager that adjusts the next sending rate based on the last SNR received on the remote STA
  - The sender has access to the receiver's statistics, which may be possible if IEEE 802.11k is supported, but is generally not available
  - The highest throughput MCS that is supported by the provided SNR is selected
  - A configurable BER threshold (default 1e-6) is used for deciding whether an MCS (SNR) is viable

Example Ideal plot

> 802.11ax in 6GHz with IdealWifiManager

```
./ns3 run 'wifi-manager-example --standard=802.11ax-6GHz'
```



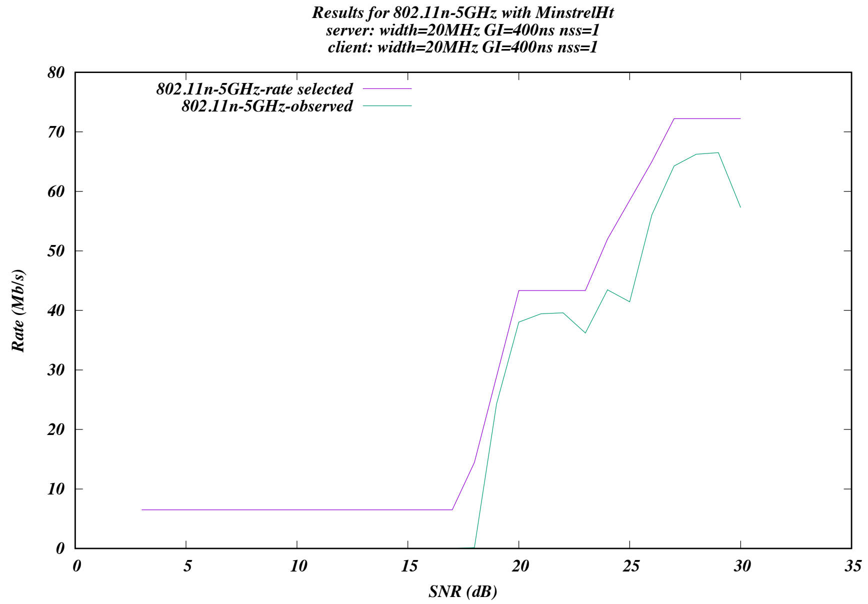
MinstrelHtWifiManager

- Overall philosophy is that it is hard to pick a rate based on available SNR figures from Linux drivers, and instead a better approach is to search for good rates via trial-and-error
- Minstrel dedicates 10% of its packets to probe for other rates that might offer an improved performance
  - called "Lookaround" rates
  - makes use of an exponentially weighted moving average (EWMA) on packet success statistics
  - Details are available in ns-3, or Yin et al, "Rate control in the mac80211 framework: Overview, evaluation and improvements," Computer Networks 81, 2015.
- ns-3 contains MinstrelWifiManager for legacy 802.11 standards, and MinstrelHtWifiManager for 802.11n/ac/ax

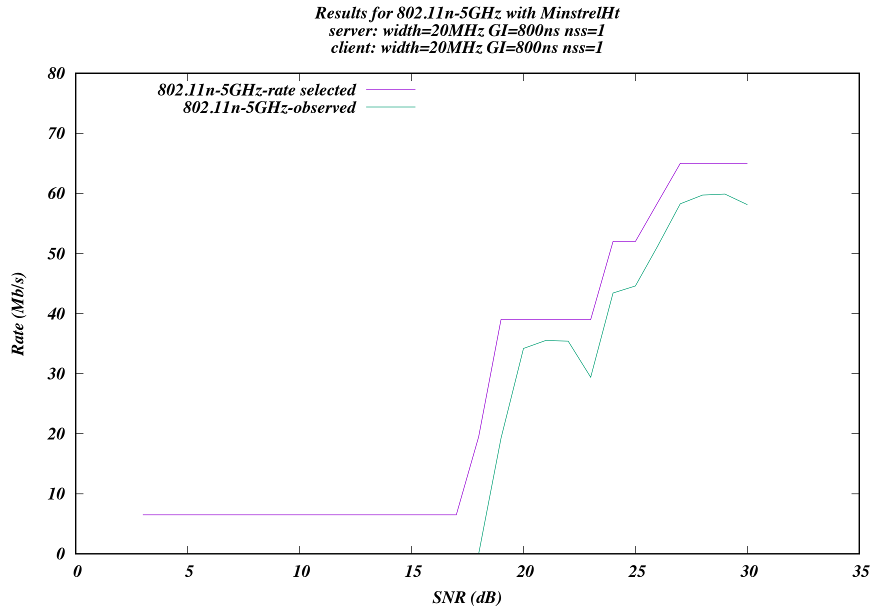
Example Minstrel plot

> Compare short and long guard interval performance for MinstrelHt at 802.11n-5GHz, 20 MHz channel, 1 stream

```
./ns3 run 'wifi-manager-example --standard=802.11n-5GHz --serverShortGuardInterval=800 --clientShortGuardInterval=800 --wifiManager=MinstrelHt'
```



400ns



800ns

Wi-Fi 6 (802.11ax) support

- **11ax frame formats**
- **OBSS PD spatial reuse** for dense networks
- **DL OFDMA** and **UL OFDMA** (including support for the MU EDCA Parameter Set)
- **Multi-user management frames** (e.g. MU-BAR)
- Round-robin **multi-user scheduler**
- **Limitations:**
  - 802.11ax/be MU-RTS/CTS is not yet supported
  - 802.11ac/ax/be MU-MIMO is not supported, and no more than 4 antennas can be configured
  - 802.11n/ac/ax/be beamforming is not supported
  - Power-save and energy consumption features
  - Cases where RTS/CTS and ACK are transmitted using HT/VHT/HE/EHT formats

Upcoming Wi-Fi extensions

Initial Wi-Fi 7 (802.11be) components are under development by Stefano Avallone and Sebastien Deronne

> New frame formats, support for new modulation types, wider channels

> **Multi-link operation** (MLO)

> **Multi-AP coordination** (TXOP sharing)

> **Channel State Information** (CSI)

Finish integration of new **fast fading MIMO error models**

– https://www.nsnam.org/research/wns3/wns3-2021/tutorials/

Integrate 802.11ad (WiGig) extensions

– https://gitlab.com/sderonne/ns-3-dev/-/tree/wigig_module

Examples to review (time permitting)

- wifi-simple-infra.cc
- wireless-animation.cc (netanim)
- wifi-80211n-mimo.cc
- wifi-hidden-terminal.cc
- wifi-manager-example.cc
- wifi-spatial-reuse.cc

References

> General: Eldad Perahia and Robert Stacey, "Next Generation Wireless LANs," Second Edition, Cambridge University Press, 2013

> Standards documents (IEEE 802.11-2016, IEEE 802.11ax-2021)

> ns-3 specific:

– Lacage, Henderson,"Yet another network simulator." *Proceeding from the 2006 workshop on ns-2: the IP network simulator*. 2006.

– Lanante Jr., Roy, Carpenter, Deronne, Improved Abstraction for Clear Channel Assessment in ns-3 802.11 WLAN Model, WNS3 2019.

– Avallone, Imputato, Redieteab, Ghosh and Roy, "Will OFDMA Improve the Performance of 802.11 Wifi Networks?," in *IEEE Wireless Communications*, vol. 28, no. 3, pp. 100-107, June 2021.

– Magrin, Avallone, Roy, and Zorzi, Validation of the ns-3 802.11ax OFDMA implementation, WNS3 2021.

Conduct Research with ns-3 Wi-Fi models

UNIVERSITY *of* WASHINGTON

How to use ns-3 Wi-Fi models to conduct our own research?

- Phase 1: Validate the modules in ns-3
  - Start with the existing examples
  - System level validation
  - Compare with well known theoretical model/other simulation tools

- Phase 2: Build new scenarios and explore with different parameters
  - Investigate the impact on different parameters: power, moving speed…
  - Build more complex scenarios : single cell->multi cells
  - Evaluate the performance and verify the guess

- Phase 3: Build and test new algorithms
  - Machine learning algorithms in wireless communication
  - Optimization approaches
  - New modules and new features

Phase 1: Validation work for Wi-Fi modules in ns-3

> Validation Examples

Validate the development of ns-3 Wi-Fi module against the well-known analytical model for different network setups.

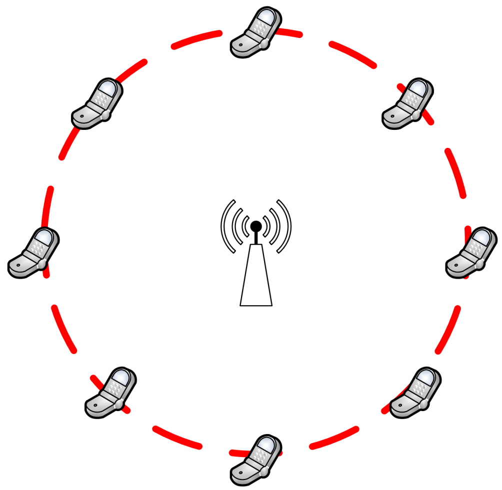
- DCF validation for different Wi-Fi standards: 802.11 a/b/g/ax
  - https://gitlab.com/nsnam/ns-3-dev/-/blob/master/src/wifi/examples/wifi-bianchi.cc
- 802.11ax OFDMA validation [1]:
  - https://github.com/signetlabdei/ofdma-validation

[1] Davide Magrin, Stefano Avallone, Sumit Roy, and Michele Zorzi. 2021. Validation of the ns-3 802.11ax OFDMA implementation. *In Proceedings of the Workshop on ns-3 (WNS3 '21).* Association for Computing Machinery, New York, NY, USA, 1–8. DOI:https://doi.org/10.1145/3460797.3460798

Phase 1: Validate the modules in ns-3

> Basic DCF validation recap

- Simulation setup:
  - Infrastructure mode: One AP and multiple stations
  - Traffic: Uplink traffic only.
  - Stations locate at the same distance (close) to the AP
  - Transmission with same power and MCS
  - Saturation mode



- Key assumptions for the analytical model:
  - No PHY errors, so packet losses only caused by the collision
  - Stations are all the same

- AP and stations may run on different powers
- Increase distances, PHY error may also occur and change the backoff window procedure.

Phase 2: Build new scenarios and explore with different parameters

> 6 GHz Power Role and Unequal Power Setup [2]

- U.S Federal Communications Commission (FCC) has adopted new rules to open the 6 GHz bands for unlicensed access.
- The new ruling limits operation by a Power Spectral Density (PSD) limit in 6 GHz bands that differs from the total average power independent of the channel bandwidth in 5 GHz bands.
- Unequal power of the Access Points (AP) and stations (STA) also impact the system performance in wireless local area networks (WLANs).

Table 1: (Max) Average Transmit power vs. channel bandwidth: Indoor Operation

Device type	Frequency	Max power for bandwidth			
		20 MHz	40 MHz	80 MHz	160 MHz
Low power AP	6 GHz	18.01 dBm	21.02 dBm	24.03 dBm	27.04 dBm
	5 GHz	30 dBm	30 dBm	30 dBm	30 dBm
Low power STA	6 GHz	12.04 dBm	15.05 dBm	18.06 dBm	21.07 dBm
	5 GHz	24 dBm	24 dBm	24 dBm	24 dBm

[2] Hao Yin, Sumit Roy, and Sian Jin. 2022. IEEE WLANs in 5 vs 6 GHz: A Comparative Study. In Proceedings of the 2022 Workshop on ns-3 (WNS3 '22).

Phase 2: Build new scenarios and explore with different parameters

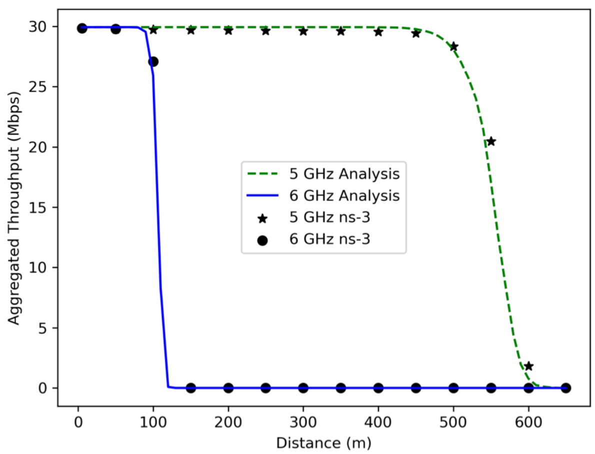
> **6 GHz Power Rule and Unequal Power Setup**

- How can we build the new scenario to test these two setups? (Demo and codes)
  - Downlink setups
  - Power rules

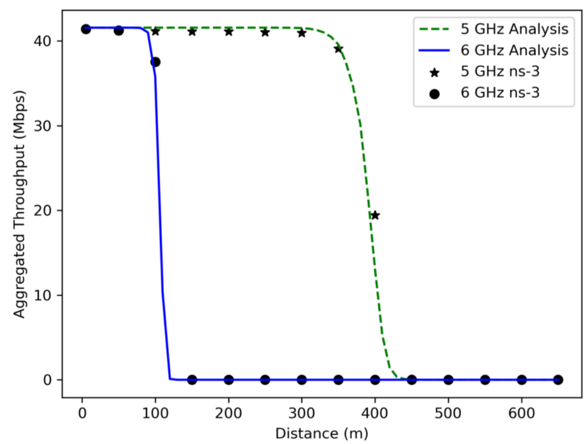
Phase 2: Build new scenarios and explore with different parameters

> 6 GHz Power Rule and Unequal Power Setup

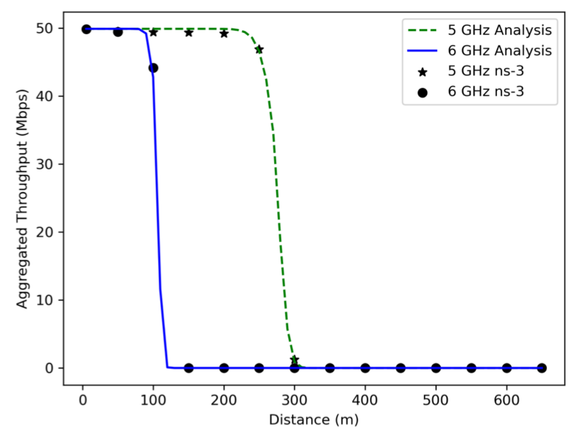
- 6 GHz power rule results   Codes: https://github.com/Mauriyin/ns3



20 MHz



40 MHz



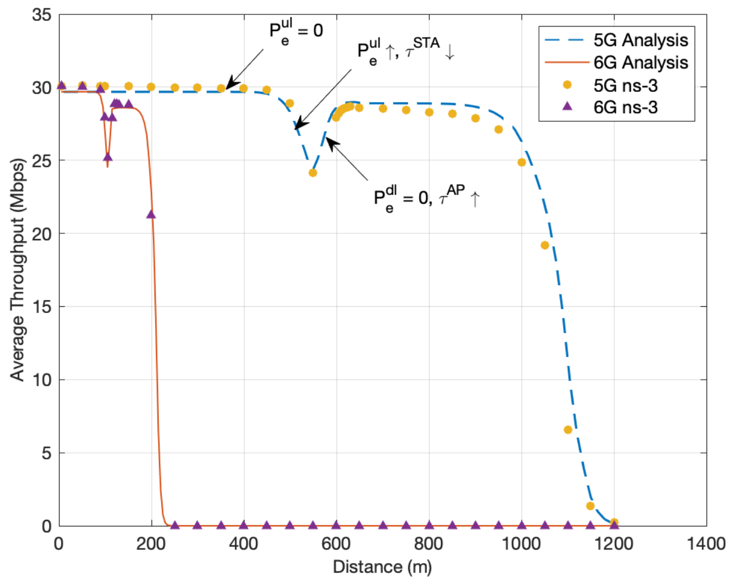
80 MHz

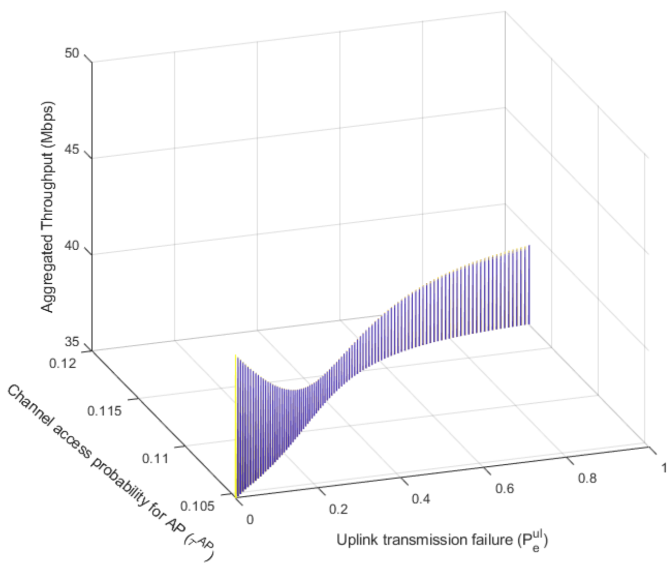
- As the distance increases, the received power and SNR decreases, the packet error rate increases, and the aggregated throughput drops.
- As the channel bandwidth increases, the transmission range of the 5 GHz band decreases while the transmission range in the 6 GHz band remains the same.

Phase 2: Build new scenarios and explore with different parameters

> 6 GHz Power Rule and Unequal Power Setup

- Unequal Power results



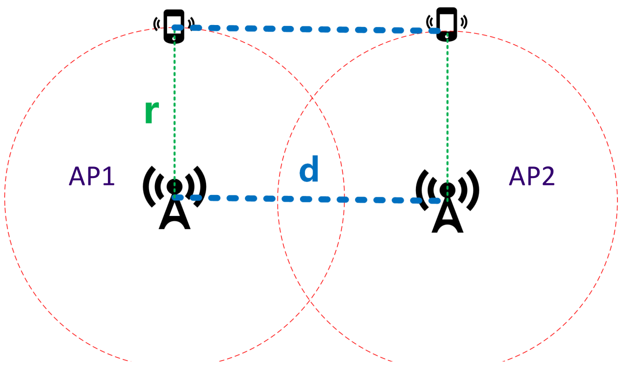


- 1st Drop: STA PER increases. STA power decreases to margin, the rate starts to drop
- Increase :The STA has some packets successfully transmitted but not to 0 (still 5 nodes, backoff window [CWmin, CWmax]); But since STAs have larger backoff windows, the AP can access channel more often.
- Flat: All the STAs’ tpt drops to 0 (backoff window CWmax, lower collision probability), only AP sending packets successfully
- 2nd Drop: AP power decreases to margin, AP PER increases

Phase 2: Build new scenarios and explore with different parameters

> Multi-BSS Setup [3]

Axis:
AP1 (0, 0)   AP2 (d, 0)
STA1 (0, r)   STA2 (d, r)



2 Overlapping BSS:

- ALL STAs are in the same position for each BSS
- CCA: -82 dBm, TX power: 20 dBm
- Log distance path loss (PL) model -> PL is a function of distance: *PL(dis)*
- Change *d* and *r* to simulate different cases.
- Uplink Only

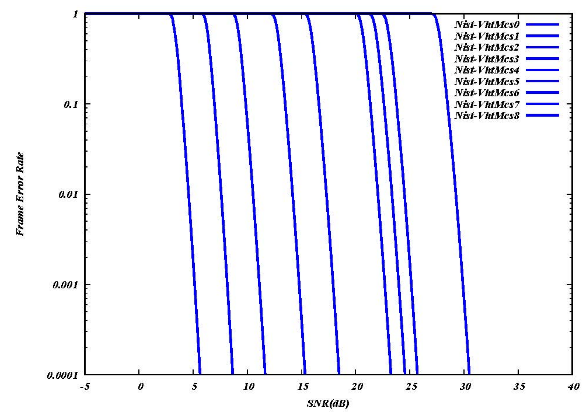
$$\text{SNIR} = \frac{P_{rx}}{(P_{in}+Noise)}$$

$$P_{rx} = P_{tx} - PL(r)$$

$$P_{in} = P_{tx} - PL(\sqrt{r^2 + d^2})$$

Conditions that 2 STAs can transmit at the same time:

- 2 STAs are in different BSS
- SINR > *Threshold(MCS)*, for example, we need around 5 dB SNIR for MCS 0



SINR vs PER

[3] R. Kajihara, H. Wenkai, L. Lanante, M. Kurosaki and H. Ochi, "Performance Analysis Model of IEEE 802.11 CSMA/CA for Multi-BSS Environment," 2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications, 2020, pp. 1-7, doi: 10.1109/PIMRC48278.2020.9217235.

Phase 2: Build new scenarios and explore with different parameters

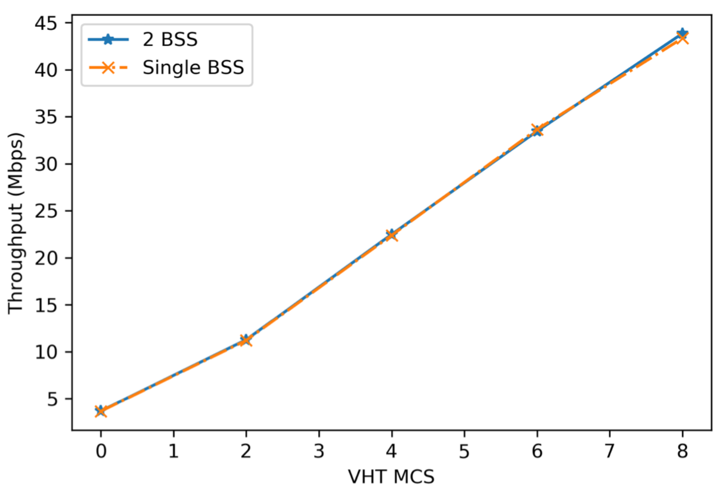
> Multi-BSS Setup

Codes: https://gitlab.com/haoyinyh/ns-3-dev/-/tree/multibss

Case 1: Equivalent case:

- Setup: r = 8m, d = 5m
  - Every node is in the carrier sensing range (can sense each other)
  - SINR = 2 dB -> No simultaneous transmission for ALL MCS
  - Expectation: 2 BSS is equivalent to one larger cell
- Results:

Parameters	Value
P_{rx}	-61.6 dBm
P_{in}	-64.6 dBm
Noise	-128 dBm
SINR	2 dB



Total 50 STAs

- 2 BSS is equivalent to one larger cell
- All the results are validated against the Bianchi model

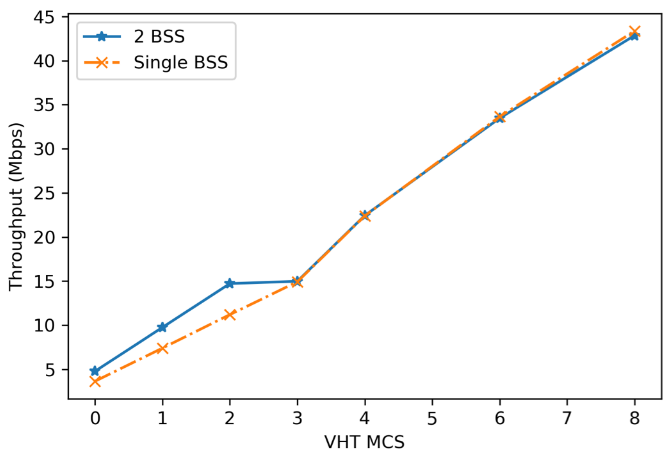
Phase 2: Build new scenarios and explore with different parameters

> Multi-BSS Setup

Case 2: Simultaneous transmission

- Setup: r = 10m, d = 20m
  - Every node is in the carrier sensing range (can sense each other)
  - SINR = 12 dB -> Can support simultaneous transmission at MCS 0/1/2
  - Expectation: 2 BSS has larger throughput in MCS 0/1/2 than one large cell
- Results:

Parameters	Value
P_{rx}	-65 dBm
P_{in}	-77.2 dBm
Noise	-128 dBm
SINR	12 dB



Total 50 STAs

Simultaneous transmission happens when MCS < 3

The multi-BSS throughput is larger when MCS < 3

Large single BSS throughput is also validated against the Bianchi model (similar with case 1)

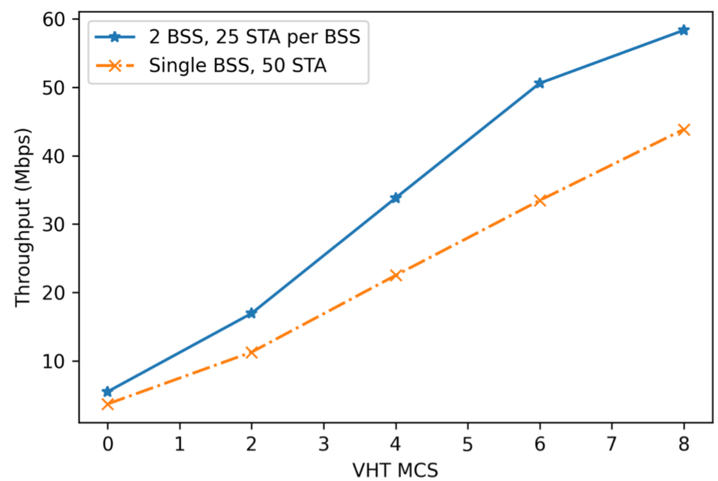
Phase 2: Build new scenarios and explore with different parameters

> Multi-BSS Setup

Case 3: Simultaneous transmission

- Setup: r = 3m, d = 20m
  - Every node is in the carrier sensing range (can sense each other)
  - SINR = 28.9 dB -> Can support simultaneous transmission at all MCSs
  - Expectation: 2 BSS has larger throughput in all MCSs than one large cell
- Results:

Parameters	Value
P_{rx}	-46.7 dBm
P_{in}	-75 dBm
Noise	-128 dBm
SINR	28.9 dB



Total 50 STAs

- Simultaneous transmission happens for all MCSs
- The multi-BSS throughput is larger
- Large single BSS throughput is also validated against the Bianchi model (similar with case 1)

Phase 3: Build and test new algorithms

> Wi-Fi Rate Control Algorithms [4]

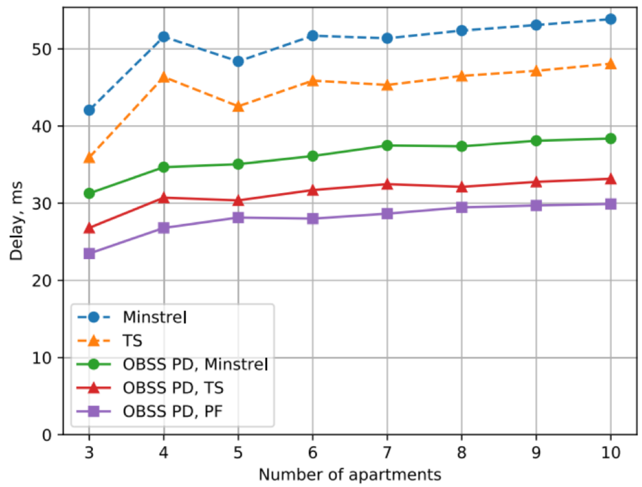


FIGURE 13. Results for the residential building.

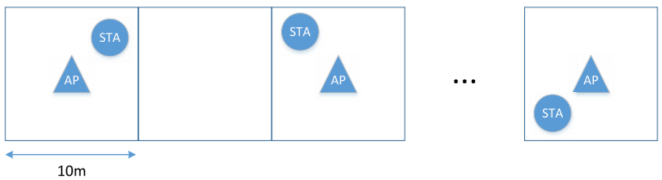


FIGURE 12. Residential building scenario.

- **TS:** MAB algorithm, using binomial distribution to approximate the success probability and then select the MCS (arm). Using Thompson sampling (TS) approach to calculate reward.
- **PF:** Estimate the channel SINR, then using TS to approach to approximate the SINR distribution, and then select the MCS based on the SINR.
- **OBSS PD:** Using OBSS PD to enable spatial reuse setup. The same way to calculate the OBSS PD: Threshold = Average RSSI – Margin (Margin is a positive value that considers channel quality fluctuations).

Benefits from RL (reinforcement learning):

- Explore the optimal way to search the (sub-)optimal setup <-> randomly search in traditional ways .
- Learn from the environment -> ‘remember’ similar situations.
- Capable for the optimization in large and complex scenario.

Deep RL? MAB?

[4] A. Krotov, A. Kiryanov and E. Khorov, "Rate Control With Spatial Reuse for Wi-Fi 6 Dense Deployments," in IEEE Access, vol. 8, pp. 168898-168909, 2020, doi: 10.1109/ACCESS.2020.3023552.

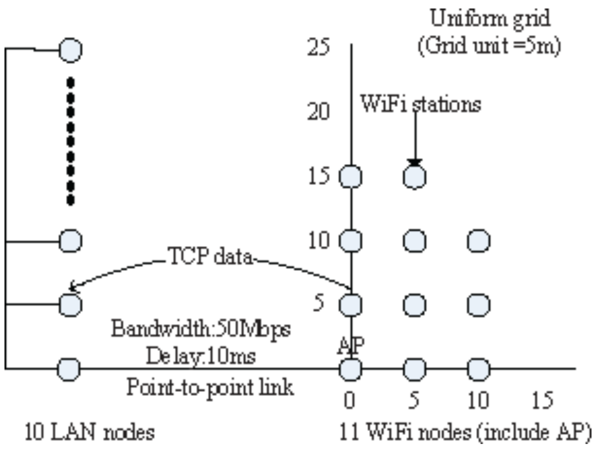
Phase 3: Build and test new algorithms

> Simulation Scenario

- Created by modifying the file " examples/tutorials/third.cc" in ns-3.

- The topology contains 10 wired LAN nodes connected to each other and one of the nodes is connected to the stationary Access Point(AP) of the Wireless Network using a point to point link with 50Mbps bandwidth and 10ms delay.

- Reference code: https://github.com/DodiyaParth/802.11ac_compatible_RAAs_Performa nce_Analysis_in_NS3



Simulation Scenario [3]

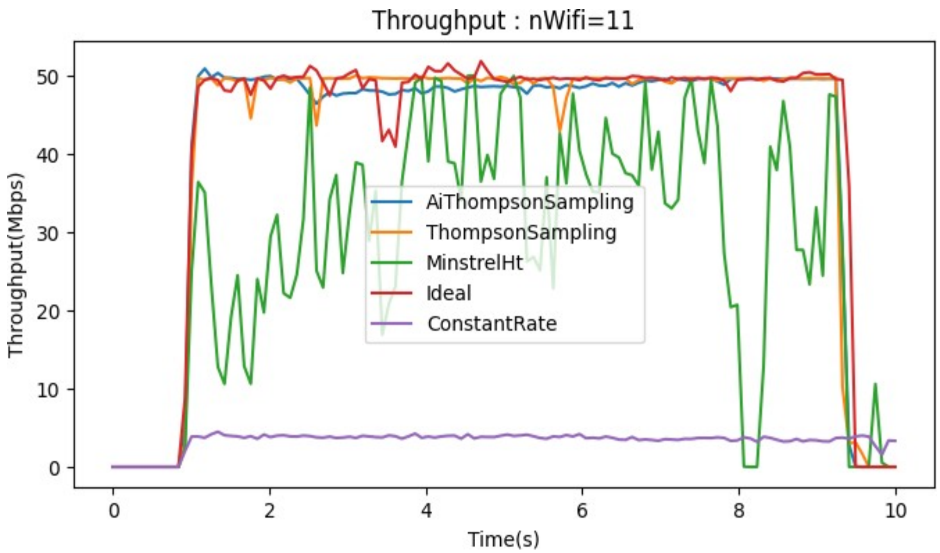
[5] Huang, Tingpei, et al. "A comparative simulation study of rate adaptation algorithms in wireless LANs." *International Journal of Sensor Networks* 14.1 (2013): 9-21.

Phase 3: Build and test new algorithms

> Simulation

Codes: https://github.com/hust-diangroup/ns3-ai

Error Rate Model	NistErrorRateModel
Channel Delay Model	ConstantSpeedPropagationDelay Model
Channel Loss Model	LogDistancePropagationLoss Mode l
MAC(Station/AP) Type	Sta WifiMac/ ApWifiMac
Application Data Rate	1 Mbps
Packet Size	1024 bytes
Mobility Model	RandomDirectional2dMobilityMo del
Mobility Speed	Random Variable : U(15.0 mps, 20.0 mps)
Simulation Topology of Wifi nodes	Grid, rectangle range: (-100m, 100m, -100m, 100m)



Under same scenario, how's the performance of different algorithms.

- Calculate the throughput every second with different rate control algorithms.
- Change the total node numbers and simulation duration to compare the results.

Thank you!

UNIVERSITY *of* WASHINGTON