Simulaciones Wi-Fi con ns-3

Créditos

- Esta presentación es una adaptación y traducción del Tutorial: “WiFi Simulations with ns-3” realizado por Tom Henderson en el LANC 2022
  - https://lanc.top/2022/tutorials-offered/#tut-work01
- Las diapositivas se proporcionan bajo la licencia Creative Commons CC-BY-SA-4.0
  - https://creativecommons.org/licenses/by-sa/4.0/
- El mérito es de la larga lista de mantenedores del módulo Wi-Fi de ns-3
  - Mathieu Lacage, Nicola Balco, Ghada Badawy, Getachew Redietab, Matias Richart, Stefano Avallone **(current),** Sebastien Deronne **(current)**
- Esta presentación fue realizada para el 2do Workshop en Sistemas Ciber-Físicos, Montevideo, Uruguay

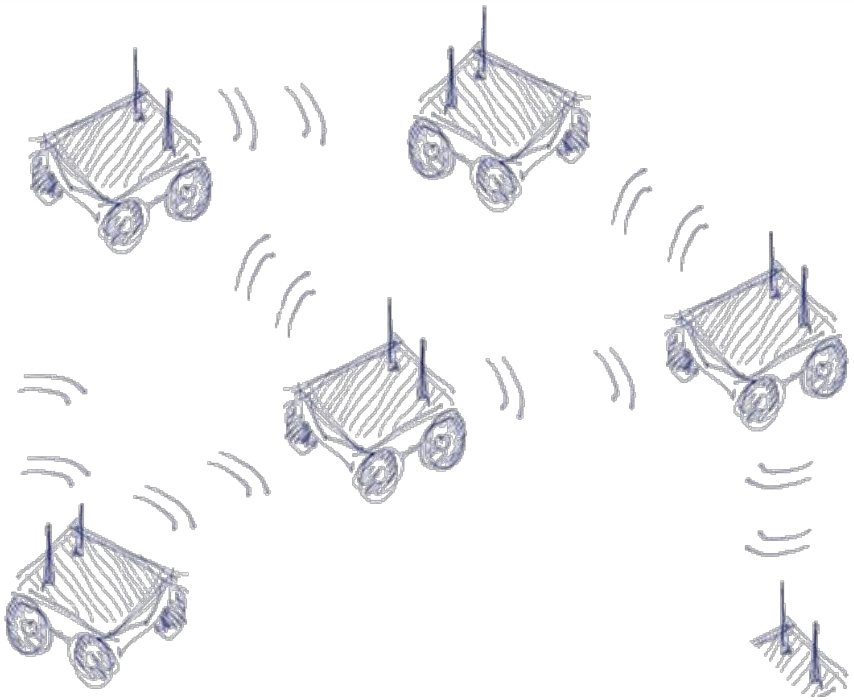
Objetivos

- Explicar por qué podría utilizar ns-3 para estudiar o aprender sobre las redes Wi-Fi
- Ilustrar algunos aspectos básicos del funcionamiento de Wi-Fi
- Mostrar cómo se puede empezar con las simulaciones de Wi-Fi de ns-3 ya escritas por otros
- Ilustrar como puede utilizarse ns-3 para desarrollar un gemelo digital de una red inalámbrica

¿Que es ns-3?

> Software para desarrollar ***modelos de redes de computadoras***, para realizar ***estudios de evaluación de rendimiento***

Pregunta: *¿Cuál es la ubicación de los robots que mejor aprovecha la red Wi-Fi?*



Fundamentos para el Análisis del Rendimiento en Redes

- Los estudios son conducidos para intentar responder a preguntas
- "¿Cuál es la ubicación de los robots que major aprovecha la red WiFi?"
  - La pregunta es muy amplia; necesitamos ajustar el foco
- **Guía 1:** Establecer claramente los objetivos del estudio y definir el alcance
- **Guía 2:** Elegir las métricas de rendimiento

- ***Pregunta redefinida:*** "¿Puedo mantener un nivel adecuado de throughput y latencia mientras los robots se mueven dentro de un cierto entorno?"

Fundamentos para el Análisis del Rendimiento en Redes

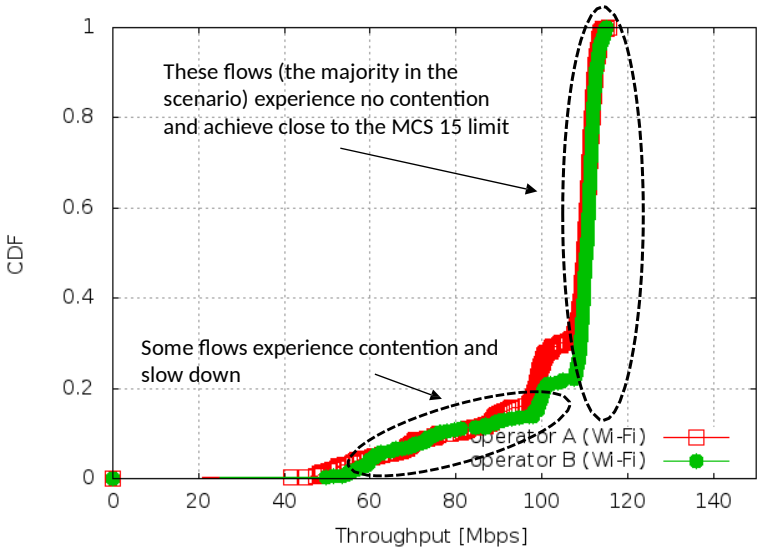
- ¿A que nos referimos con “throughput” y “latencia”?
  - ¿Como se mide? (definición precisa)
  - ¿Cuáles estadísticas? (throughput promedio, percentil 99%, pero caso, etc.)?

- **Guía 3:** Elegir los parámetros del sistema y del experimento
  - Estándar WiFi, interior o exterior, frecuencia, tipo de antena, propagación
  - Posición de nodos, tiempo, repeticiones, tráfico

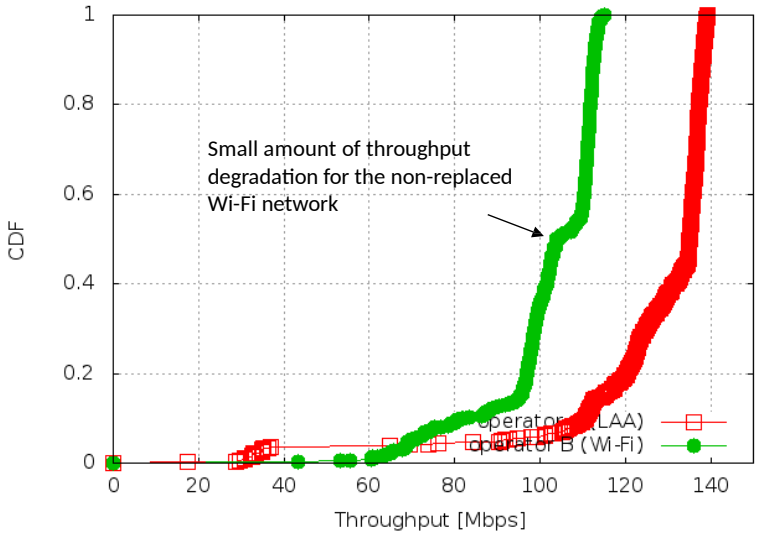
Fundamentos para el Análisis del Rendimiento en Redes

> **Guía 4:** Diseñar experimentos
  - Elegir técnicas de evaluación
  - Seleccionar las métricas y sus valores

> Ejemplo: Colocar nodos Wi-Fi en un espacio, generar tráfico y graficar la CFD del trhoughput observado. Repetir con otra ubicación de los nodos.



a) Step 1 (Wi-Fi)



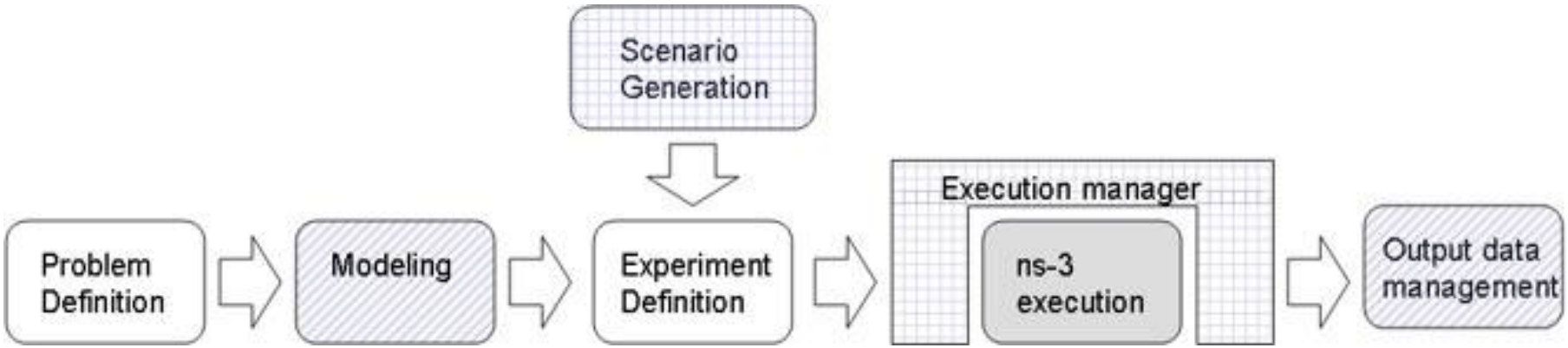
b) Step 2 (LAA)

Fundamentos para el Análisis del Rendimiento en Redes

- **Guía 5**: Analizar e interpretar los datos, e iterar
  - Casi nunca un proceso de una sola vez
  - A menudo es necesario profundizar en el modelo o en el escenario, para extraer detalles mas finos
- **Guía 6**: Hacer resultados fáciles de reproducir
  - Para otros, y para uno mismo (en un momento posterior)

¿Qué es ns-3?

> El caso de estudio anterior sigue un flujo:

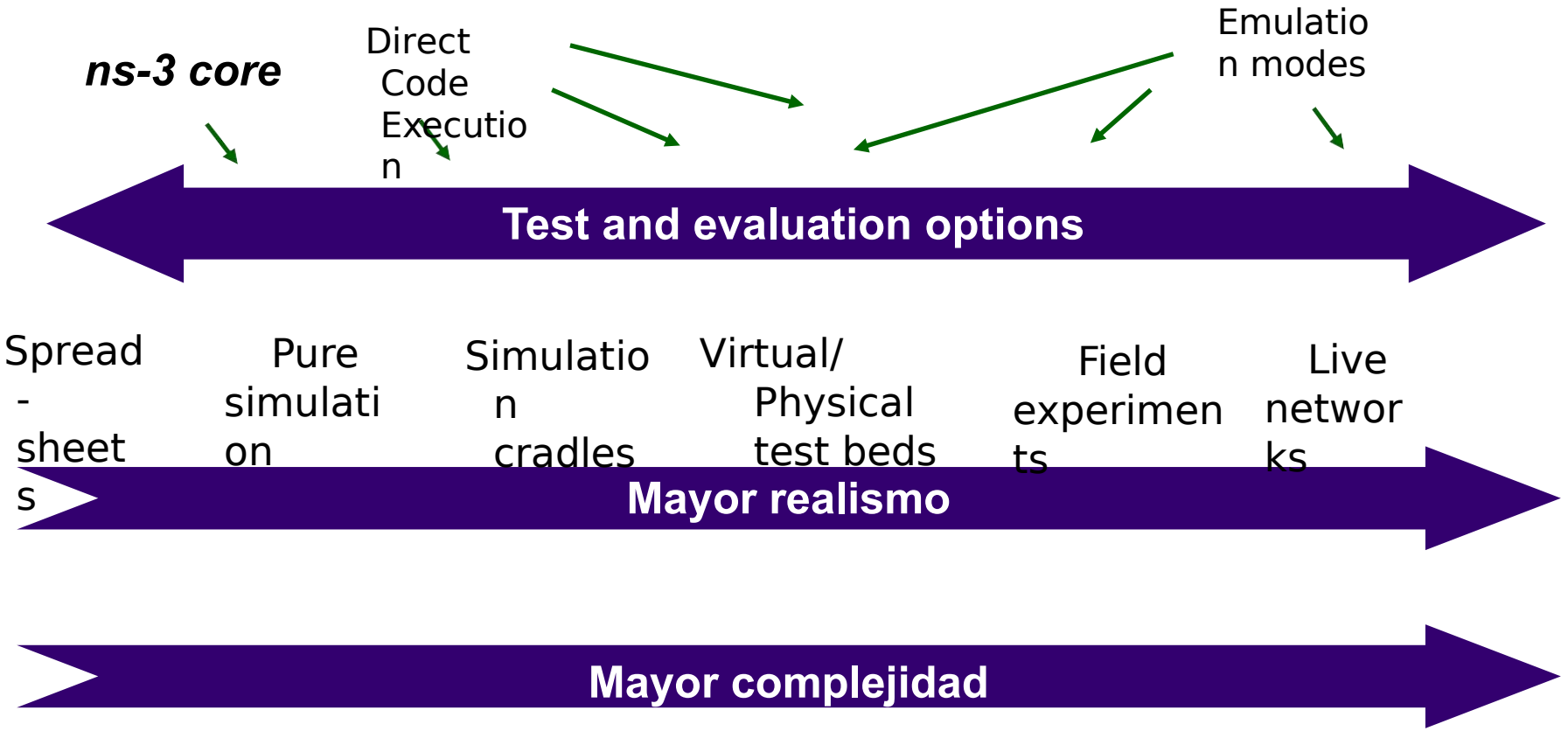


Alternativas para la evaluación de rendimiento

- Análisis matemático
- Paquetes de cómupto numérico (ej., MATLAB)
- Simuladores a nivel de paquetes (packet-level)
- Simuladores a nivel de sistema (system-level)
- Testbeds, prototipos
- Pruebas de campo

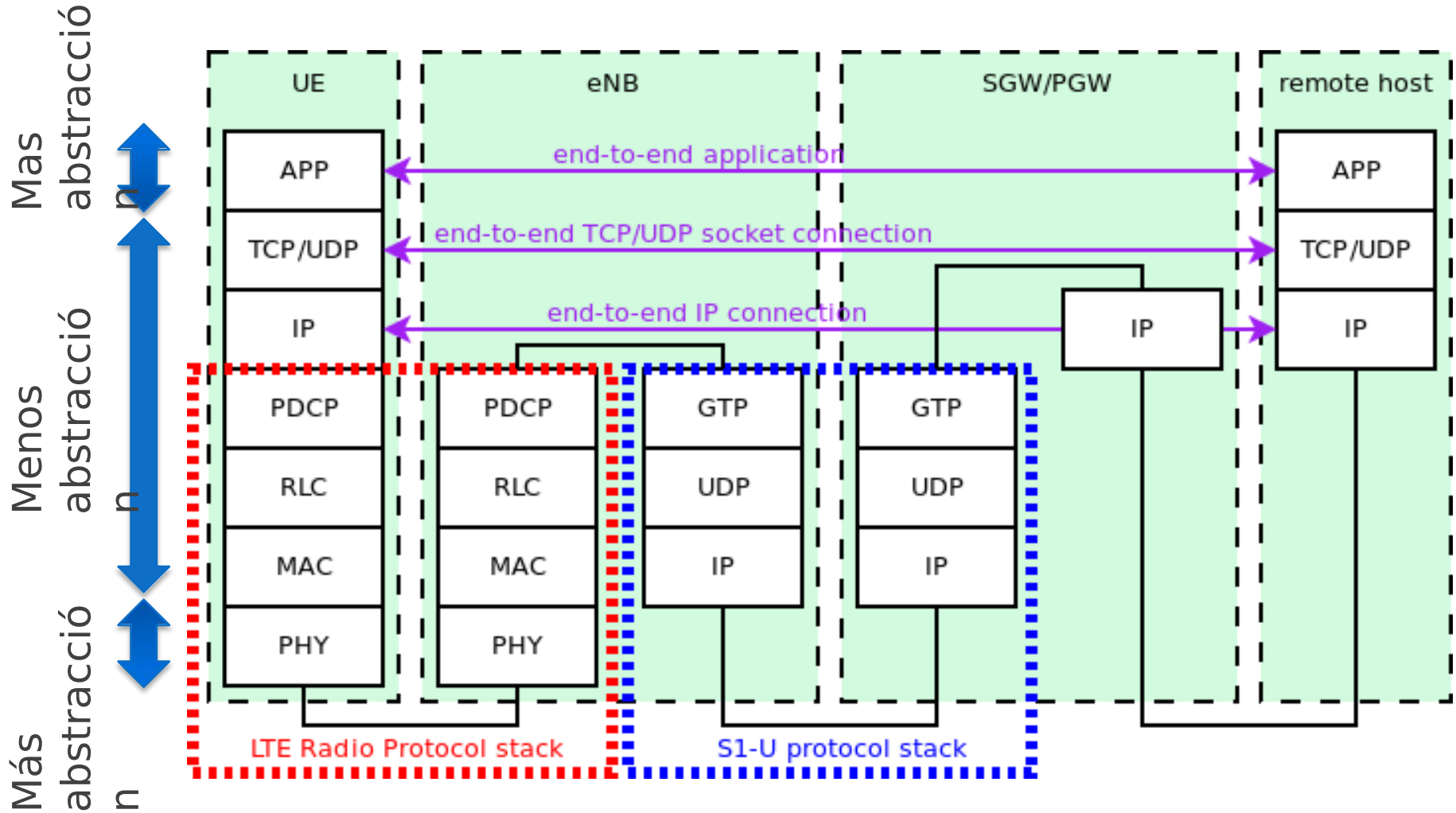
¿Qué es ns-3? (cont.)

> ns-3 es principalmente un simulador de eventos discretos, pero también tiene modos de funcionamiento que le permiten interactuar con software y redes del mundo real



¿Qué es ns-3? (cont.)

- **Simulación de red a nivel de paquete:** La principal unidad de modelado es el *paquete* y las entidades que intercambian paquetes.



Wi-Fi y ns-3

- El diseño actual de Wi-Fi está enfocado en resolver ***problemas de asignación de recursos*** para maximizar el throughput y minimizar la latencia
  - Rate control
  - Scheduling
  - Power control
  - Interference mitigation
- Experiencia de usuario (throughput, latencia) depende en detalles de punta a punta y efectos cross-traffic
  - ej., latencia en juegos, protocolos de transporte y control de congestión, priorización de trafico y disciplinas de encolamiento

Simuladores a nivel de paquetes como ns-3 son ideales para estos estudios

Fundamentos del software ns-3

- ***ns-3 está escrito en C++***
  - Los programas de ns-3 utilizan C++ estándar, bibliotecas de ns-3 escritas en C++ y (opcionalmente) bibliotecas de C++ de terceros
- ***El sistema de construcción de ns-3 está basado en CMake y Python 3***
  - requisitos mínimos del sistema: g++/clang++, CMake, make o ninja build system, y Python
- ***ns-3 suele generar datos de salida en bruto*** y depende de otras herramientas para visualizar o procesar los datos

Esquema de la presentación

Basada en ejemplos

1. Levantar y ejecutar ns-3
2. Conceptos básicos de la simulación de eventos discretos de ns-3
3. Recorrido detallado de un sencillo programa de ejemplo de Wi-Fi
4. Ejemplos y descripciones de características adicionales del modelo Wi-Fi
5. Pasar de los ejemplos a la validación y al desarrollo de nuevas simulaciones

Prerequisitos

- Alguna experiencia de programación en Linux o macOS
- Alguna experiencia o conocimiento de C++
- Conocimientos básicos de redes Wi-Fi
- Se recomienda a los nuevos usuarios que trabajen con el tutorial de ns-3
  - HTML: https://www.nsnam.org/docs/tutorial/html/index.html
  - PDF: https://www.nsnam.org/docs/tutorial/ns-3-tutorial.pdf

Obtener ns-3

- La mayoría de los recursos están enlazados desde el sitio web principal de ns-3 en https://www.nsnam.org
- ns-3 se desarrolla y mantiene en GitLab.com en https://gitlab.com/nsnam/ns-3-dev
- Utilizaremos la versión más reciente de ns-3 (ns-3.37)
  - https://www.nsnam.org/release/ns-allinone-3.37.tar.bz2
- Si está utilizando una versión anterior o posterior de ns-3, tenga en cuenta que algunas cosas pueden haber cambiado

Compilando ns-3

- (Demo) Descargar ns-3
- (Demo) Configurar ns-3
- (Demo) Compilar ns-3
- (Demo) Ejecutar programas

Para más información, lea el tutorial Quick Start: https://www.nsnam.org/docs/tutorial/html/quick-start.html

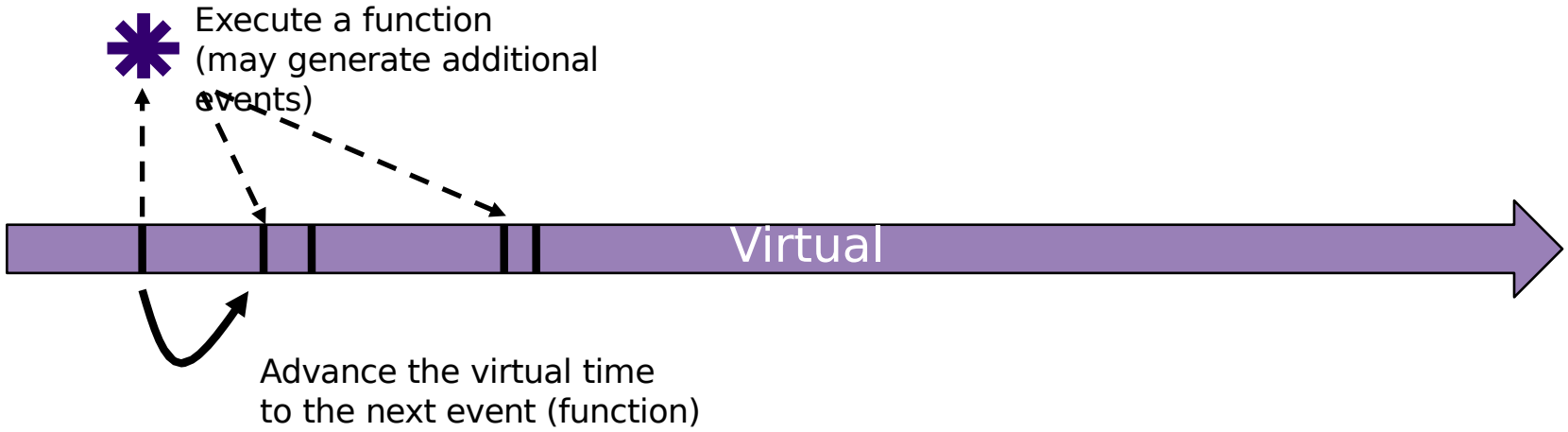
Fundamentos de la simulación basada en eventos

Intentamos representar el funcionamiento de una red dentro de un único programa C++

- Necesitamos la noción de ***tiempo virtual*** y de ***eventos*** que ocurren a tiempos (virtual) específicos
- Necesitamos una estructura de datos (***scheduler***) para mantener todos estos eventos en orden temporal
- Necesitamos un objeto (***simulador***) para avanzar sobre la lista de eventos y ejecutarlos al tiempo virtual correcto
- Podemos elegir ignorar cosas que conceptualmente puedan ocurrir entre eventos de interés, focalizando solo en los tiempos (***discreto***) con eventos interesantes

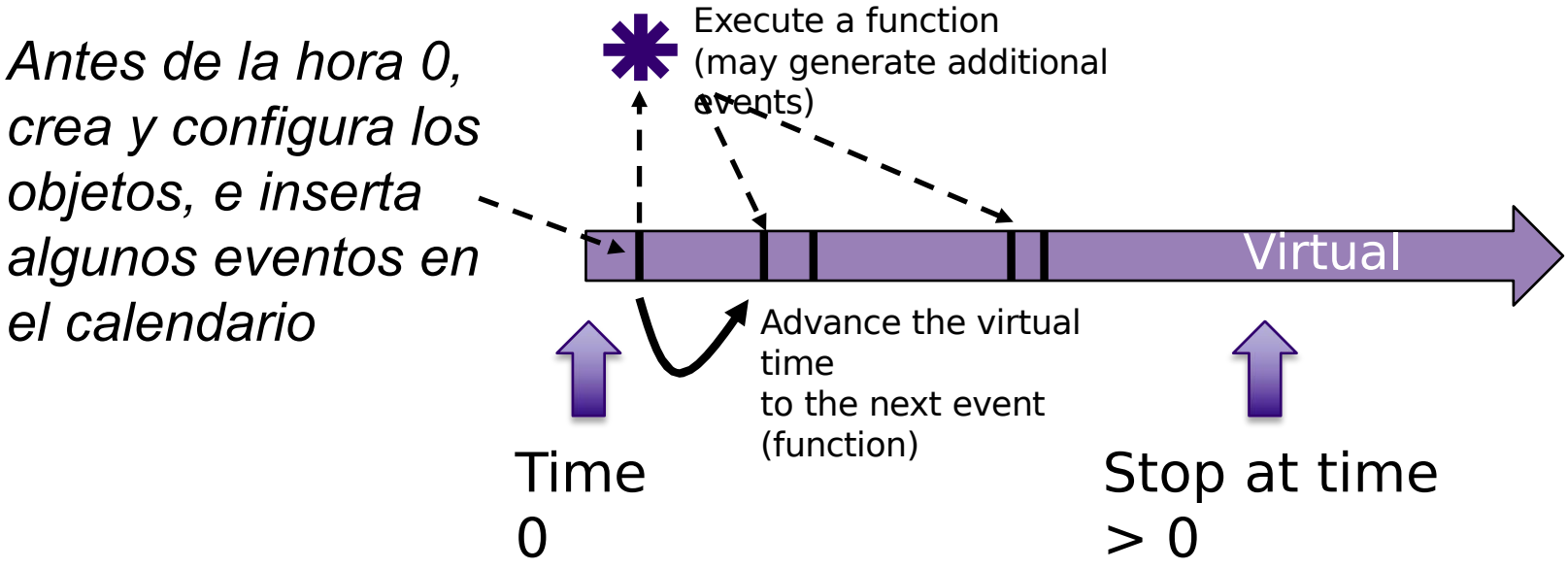
Fundamentos de la simulación basada en eventos (cont.)

- El tiempo de simulación se mueve en saltos discretos de evento a evento
- Las funciones de C++ planifican (schedule) eventos para que ocurren en tiempos de simulación específicos
- Un planificador de simulación ordena la ejecución de eventos
- Simulation::Run() ejecuta una lista de eventos
- La simulación para a un tiempo especificado cuando no hay mas eventos



Fundamentos de simulación de ns-3 y terminología

- La ejecución de una simulación consiste usualmente del siguiente flujo de trabajo
  1. Antes del "tiempo 0" teórico, crea los objetos del escenario y pre-carga el planificador con algunos eventos iniciales
  2. Define los criterios de parada; ya sea un tiempo virtual futuro específico, o cuando se cumplan ciertos criterios
  3. Inicia la simulación (que inicializa los objetos, en "tiempo 0")



Tiempo virtual en ns-3

- El tiempo se almacena como un gran *integer* en ns-3
  - Minimiza las diferencias de punto flotante entre plataformas
- Se proporcionan clases especiales de tiempo para manipular el tiempo (como los operadores aritméticos estándar)
- La resolución de tiempo por defecto es de nanosegundos, pero puede ajustarse a otras resoluciones
  - Nota: El cambio de resolución no está bien utilizado/probado
- Los objetos de tiempo pueden ser fijados por valores de punto flotante y pueden exportar valores de punto flotante

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

  - La mejor práctica es evitar las conversiones en coma flotante siempre que sea posible y utilizar los operadores aritméticos de tiempo

Bloques de construcción clave: Callback y function pointer

> Métodos C++ son usualmente invocados directamente en objetos

```
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);
```

A diferencia de CommandLine::Parse(), generalmente necesitamos llamar a funciones en algún tiempo futuro (virtual).

Algún elemento del programa podría asignar un puntero a una función, y una sentencia (posterior) del programa podría llamar (ejecutar) el método

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

Eventos en ns-3

- Los eventos son sólo funciones (callbacks) que se ejecutan en un momento simulado
  - no hay nada especial en las funciones o métodos de la clase que puedan ser utilizados como eventos
- Los eventos tienen identificadores para permitir que se cancelen o para comprobar su estado

Simulador y Planificador

- La clase *Simulator* contiene un planificador, y provee una API para planificar eventos, iniciar, terminar y limpiar la memoria
- Varias estructuras de datos del planificador (calendar, heap, list, map) son posibles
- Simulación "realtime“ alinea el tiempo simulado con el tiempo real (wall-clock)
  - dos políticas (límite duro y suave) disponibles cuando la simulación y el tiempo real divergen

(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)

Fundamentos de la simulación de eventos discretos

- Eventos, scheduler, tiempo de simulación, variables aleatorias (✓)
- Salida del programa
- Paquetes
- Nodos, NetDevices
- Modelos de movilidad/Posición

Opciones de salida del programa

En general, ns-3 no emite datos por defecto - debe ser configurado para hacerlo

- Raw logs
- Trace sources
- PCAP, ASCII, y trazas athstats
- Trazas de animación (Animation traces)
- Archivos de gráficos Gnuplot

Formatos de archivo de salida

- Los archivos PCAP (de captura de paquetes) pueden ser leídos por programas como ***tcpdump*** y ***wireshark***
- Los archivos de rastreo ASCII son representaciones en texto plano de las transmisiones, recepciones y descarte de paquetes
  - Ejemplo de `./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])
```

- Las trazas de athstats utilizan fuentes de trazas Wi-Fi para proporcionar un log similar a la de los controladores

Ejemplo de `./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

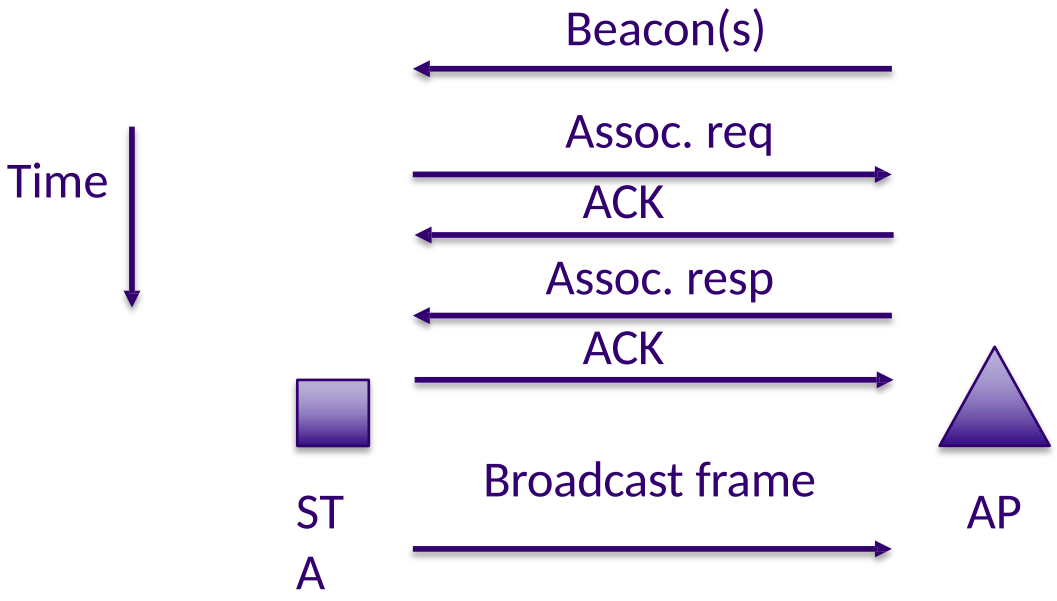
(Salida de ejemplo) wifi-simple-infra.cc

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

> Mostrar logs

> Salida del programa (pcap)

> Mirar Wireshark



Paquetes

> Un paquete es un buffer de datos especial con espacio para ***headers***, ***trailers***, ***tags***, y ***metadata***

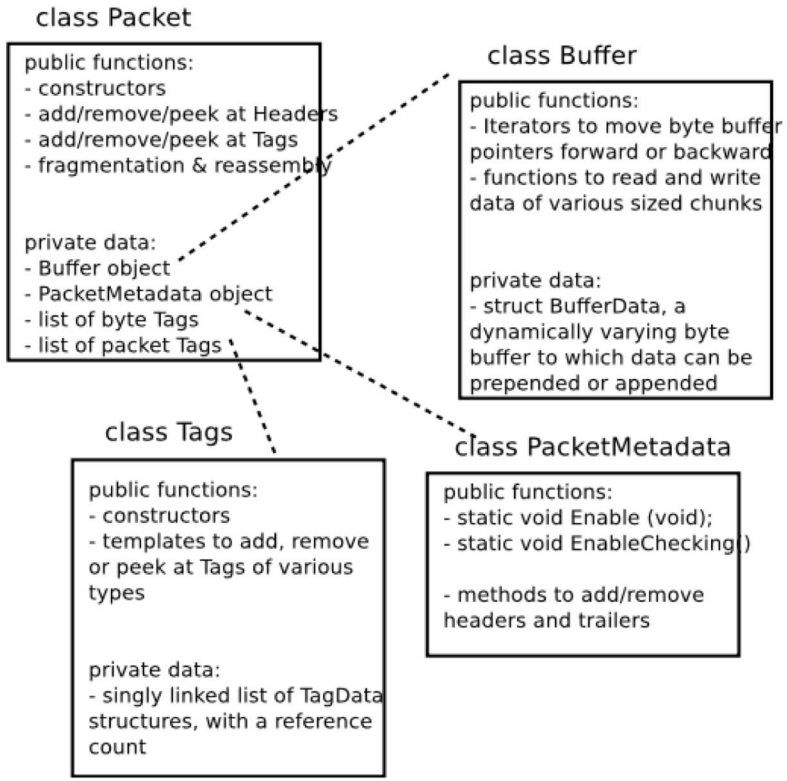
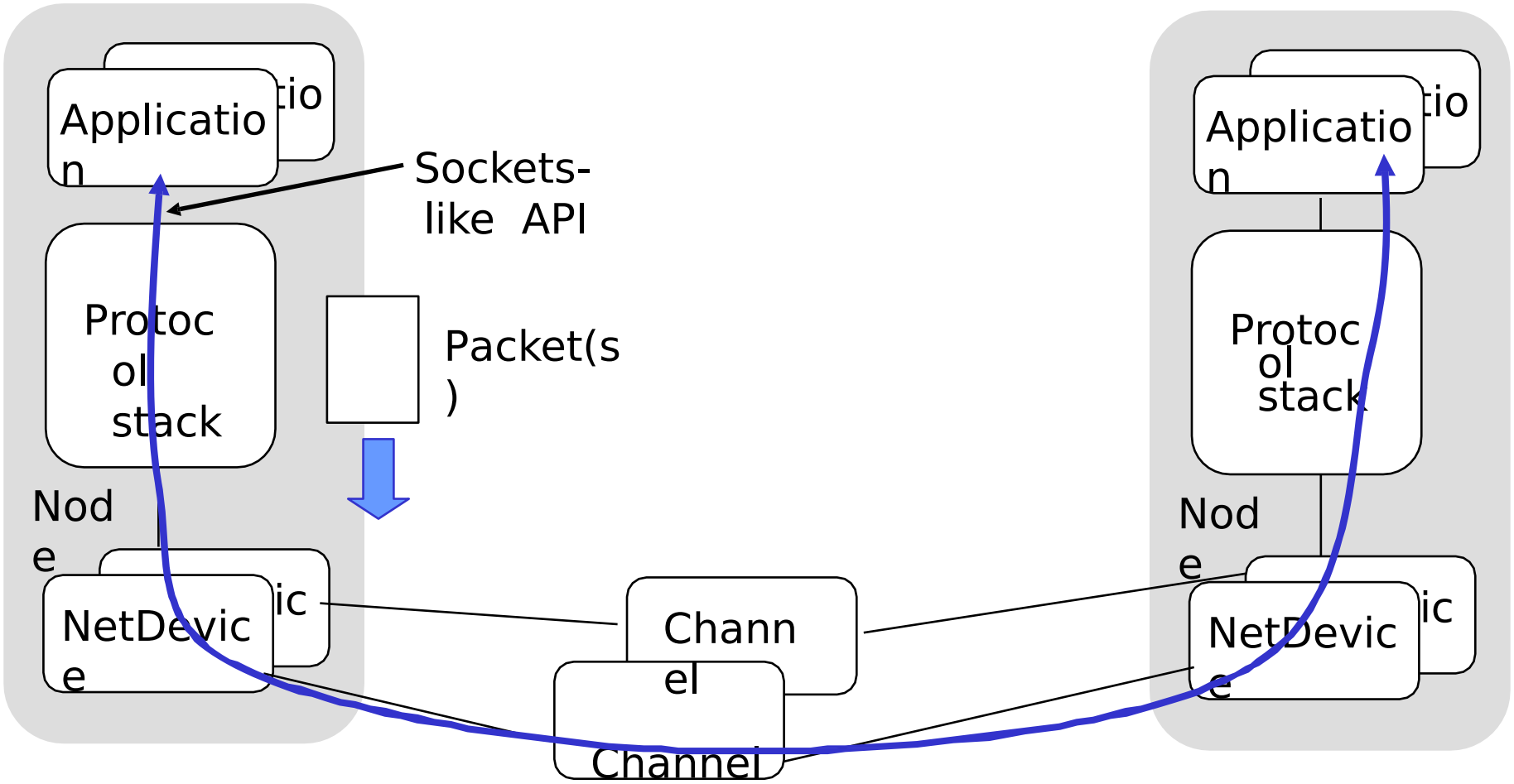


Fig. 1: Implementation overview of Packet class.

> Fuente de la figura: ns-3 Model Library documentation

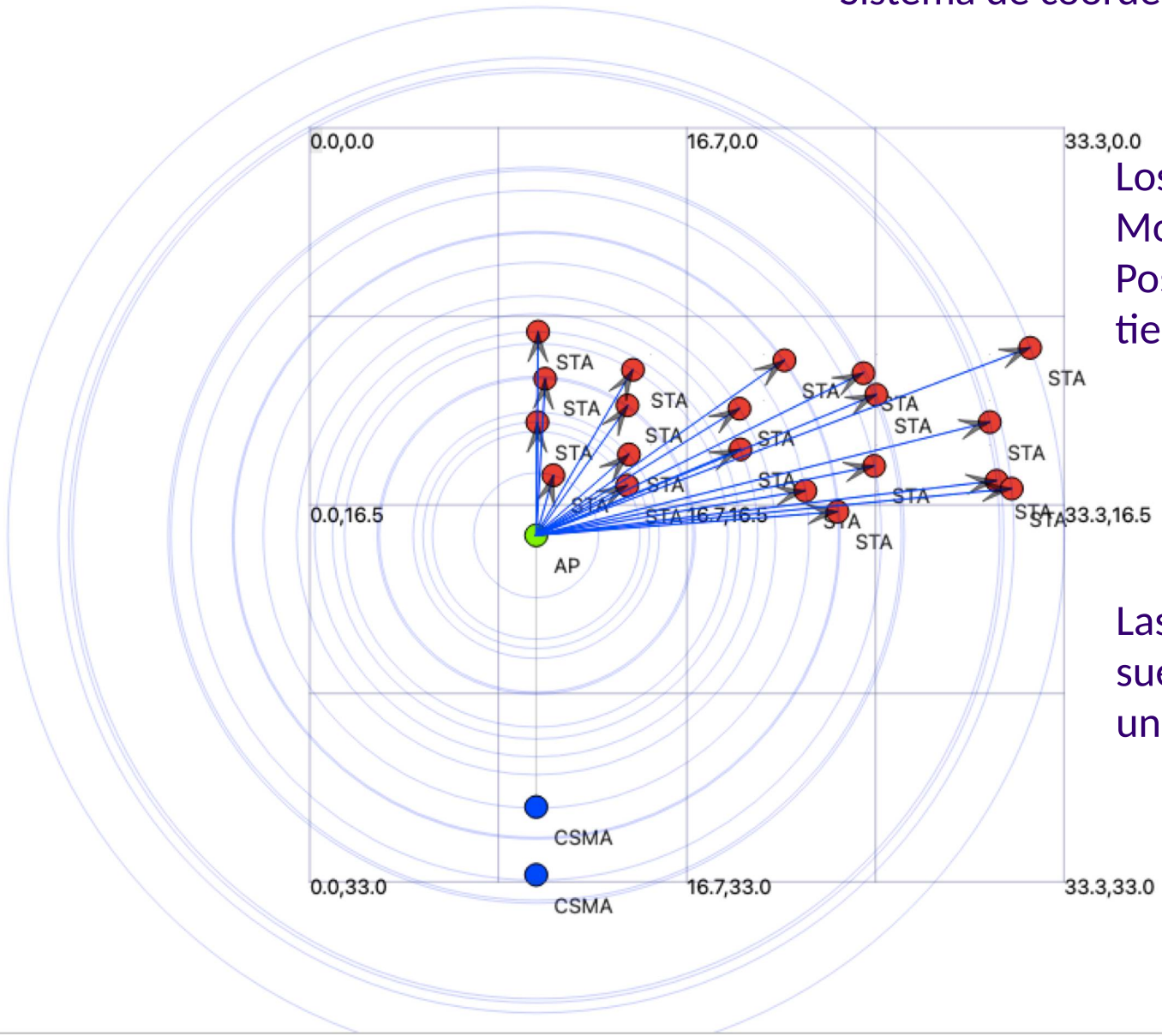
Nodos, Aplicaciones, NetDevices

> La mayoría de las simulaciones incluyen intercambios de paquetes como los que se muestran a continuación



Movilidad y posición

Sistema de coordenadas cartesianas



Los nodos suelen tener un MobilityModel con una Posición (variable en el tiempo)

Las posiciones iniciales suelen ser asignadas por un PositionAllocator

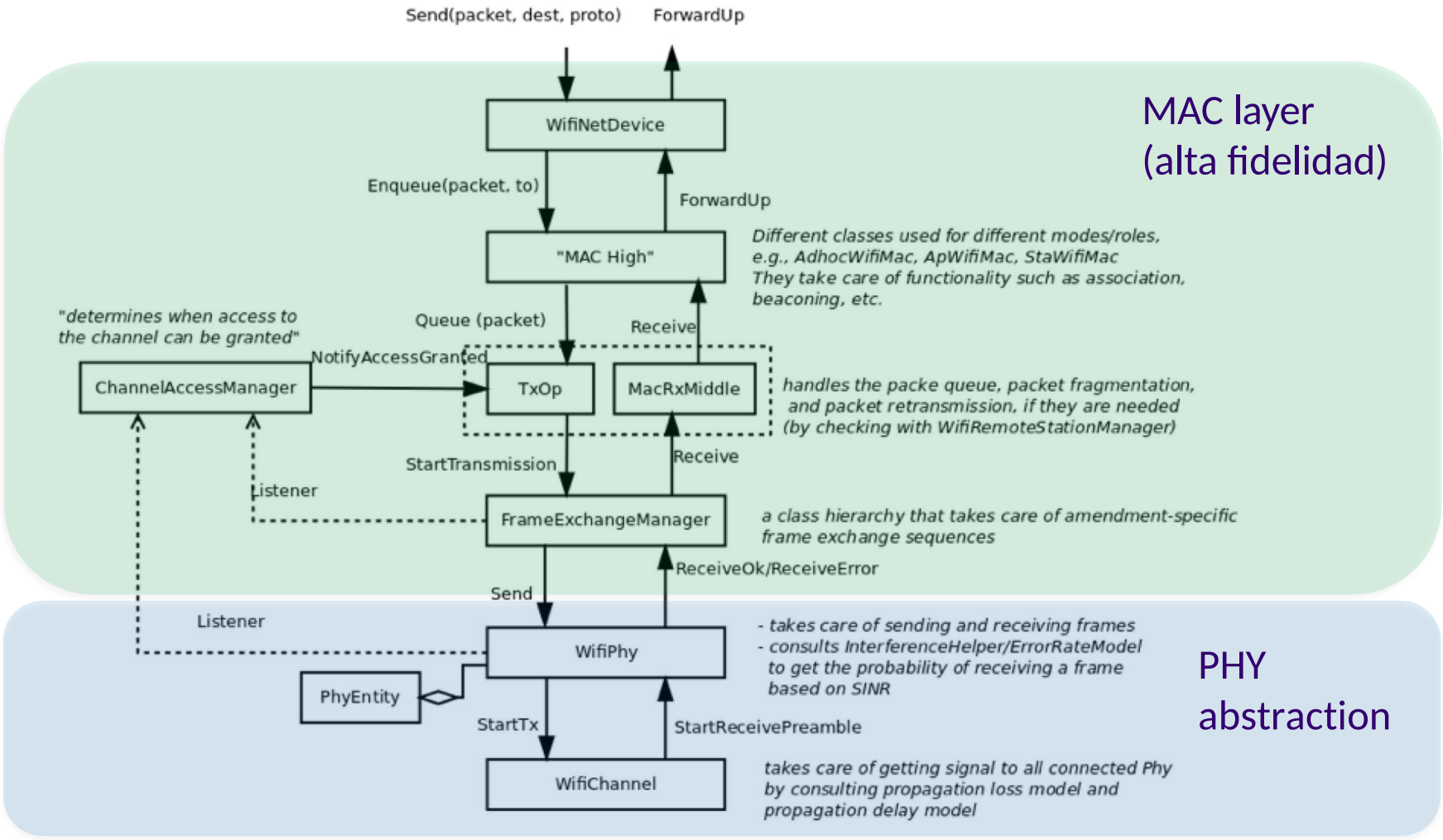
Movilidad y posición

- La posición de ns-3 se representa en un sistema de coordenadas cartesianas (x,y,z) en 3D
- El MobilityHelper combina un modelo de movilidad (**mobility model**) y un asignador de posiciones (**position allocator**).
- Los asignadores de posición establecen la posición inicial de los nodos (sólo se utilizan cuando se inicia la simulación):
  - **List:** asignar posiciones de una lista determinista especificada por el usuario;
  - **Grid:** asignar posiciones en una cuadrícula rectangular 2D (primero la fila o la columna);
  - **Random position allocators:** asignar posiciones aleatorias dentro de una forma seleccionada (rectángulo, círculo, ...).
- Los modelos de movilidad especifican cómo se moverán los nodos durante la simulación:
  - **Constant:** posición, velocidad o aceleración;
  - **Waypoint:** especificar la ubicación para un tiempo determinado (pares de tiempo-posición);
  - **Trace-file based:** parsear archivos y convertirlos en eventos de movilidad ns-3, soportar herramientas de movilidad como SUMO, BonnMotion (usando el

Resumen

- Eventos, scheduler, tiempo de simulación, variables aleatorias (✓)
- Salida del programa (✓)
- Paquetes (✓)
- Nodos, NetDevices (✓)
- Modelos de movilidad/Posición (✓)

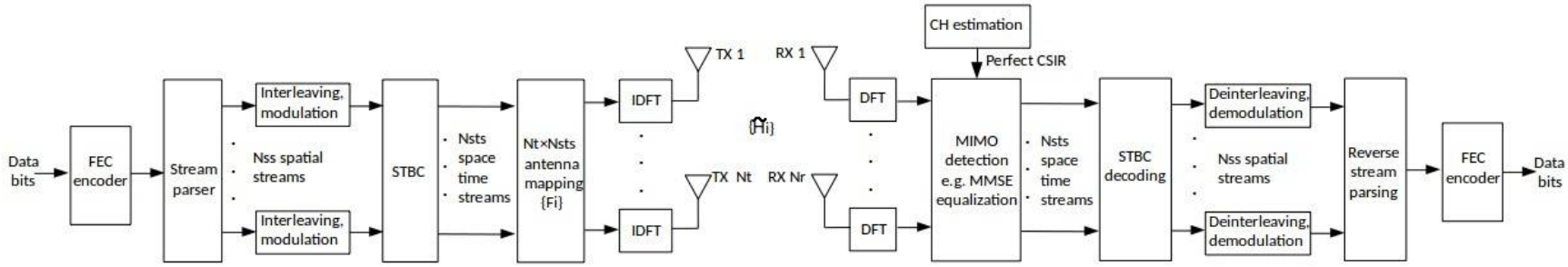
Capas MAC y PHY en ns-3 Wi-Fi



WifiNetDevice architecture

Abstracción PHY

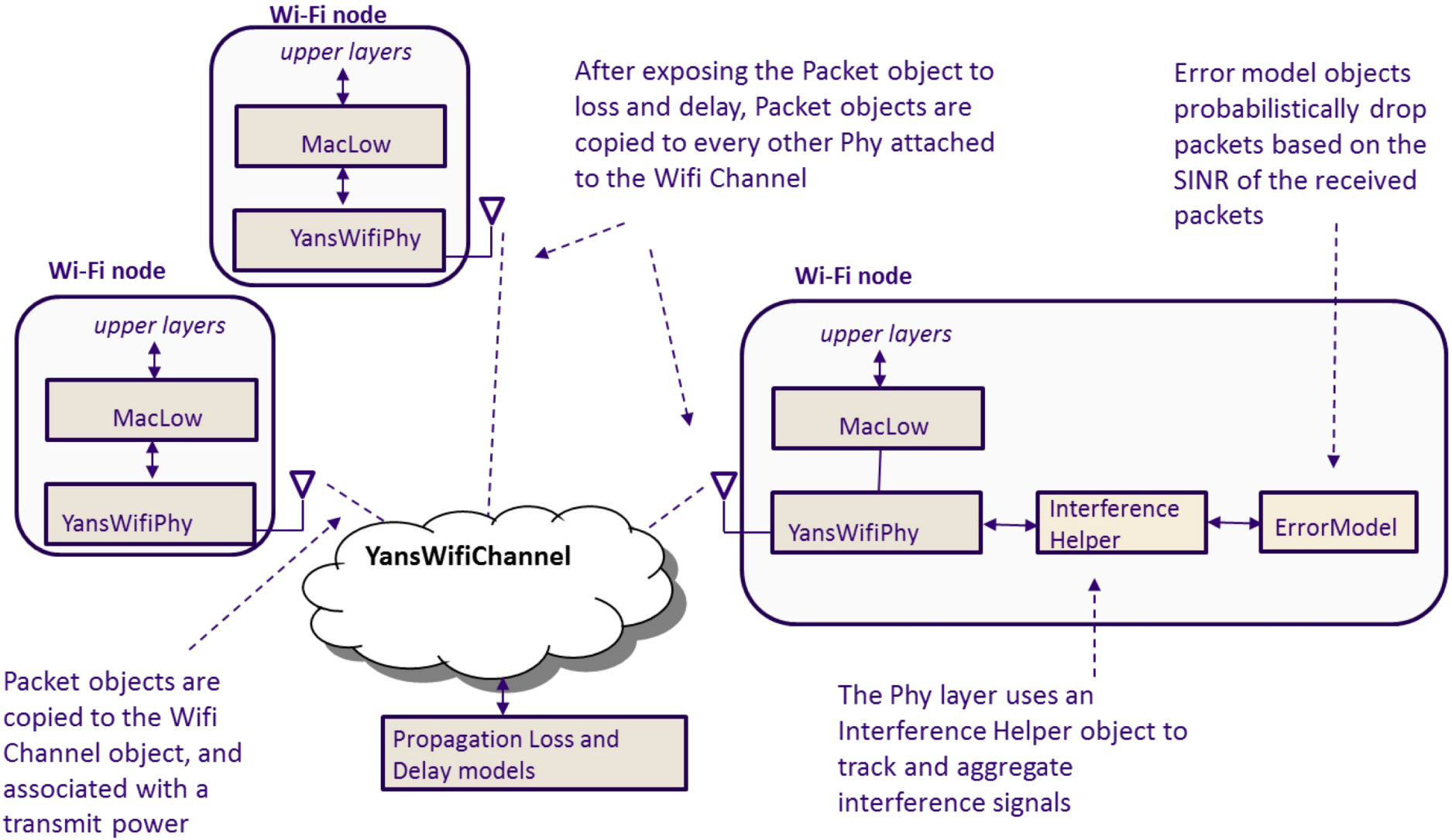
- Un modelo completo de capa física descompondría los paquetes en símbolos de RF y modelaría la ecualización, la sincronización, etc.



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

- En ns-3, los objetos "paquete" se transmiten a través de canales que añaden pérdidas de propagación, se calculan los SNR y éstos se traducen en ratios de error de paquete (PER)

Abstracción PHY en ns-3



Canales Wi-Fi

> Se admiten dos opciones:

1. YansWifiChannel (modelo simple, single-band)
   - Se utiliza si no hay un modelo de “fading” selectivo en frecuencia y si no hay interferencias de fuentes externas
   - YansWifiChannelHelper agregará por defecto el “LogDistancePropagationLossModel” con un valor del exponente de “path-loss” de 3
2. SpectrumChannel (descomposición en bandas fine-grained)
   - Utilizar si se necesitan modelos selectivos de frecuencia más detallados, o en un entorno de señal mixta
   - SpectrumWifiChannelHelper agregará por defecto el “FriisSpectrumPropagationLossModel” (potencia decae como el cuadrado de la distancia)

Propagación

- El módulo de propagación define:
  - Modelos de pérdida de propagación (propagation loss):
    Calcula la potencia de la señal Rx teniendo en cuenta la potencia de la señal Tx y las respectivas posiciones de las antenas Rx y Tx.
  - Modelos de retardo de propagación (propagation delay):
    Calcular el tiempo de recorrido de las señales desde las antenas de transmisión hasta las de recepción.
- Los modelos de retardo de propagación casi siempre se ajustan a:
  - ConstantSpeedPropagationDelayModel: En este modelo, la señal viaja con velocidad constante (por defecto la velocidad de la luz en el vacío)

Propagación (cont.)

> Modelos en ns-3:

- Se aplican muchos modelos de pérdida de propagación:
  - ✓ Abstract propagation loss models: FixedRss, Range, Random, Matrix, …
  - ✓ Deterministic path loss models: Friis, LogDistance, ThreeLogDistance, TwoRayGround, …
  - ✓ Stochastic fading models: Nakagami, Jakes, …

Propagación (cont.)

- Un modelo de pérdida de propagación puede "encadenarse" a otro, formando una lista. La potencia Rx final tiene en cuenta todos los modelos encadenados.

Ejemplo: path loss model + shadowing model + fading model

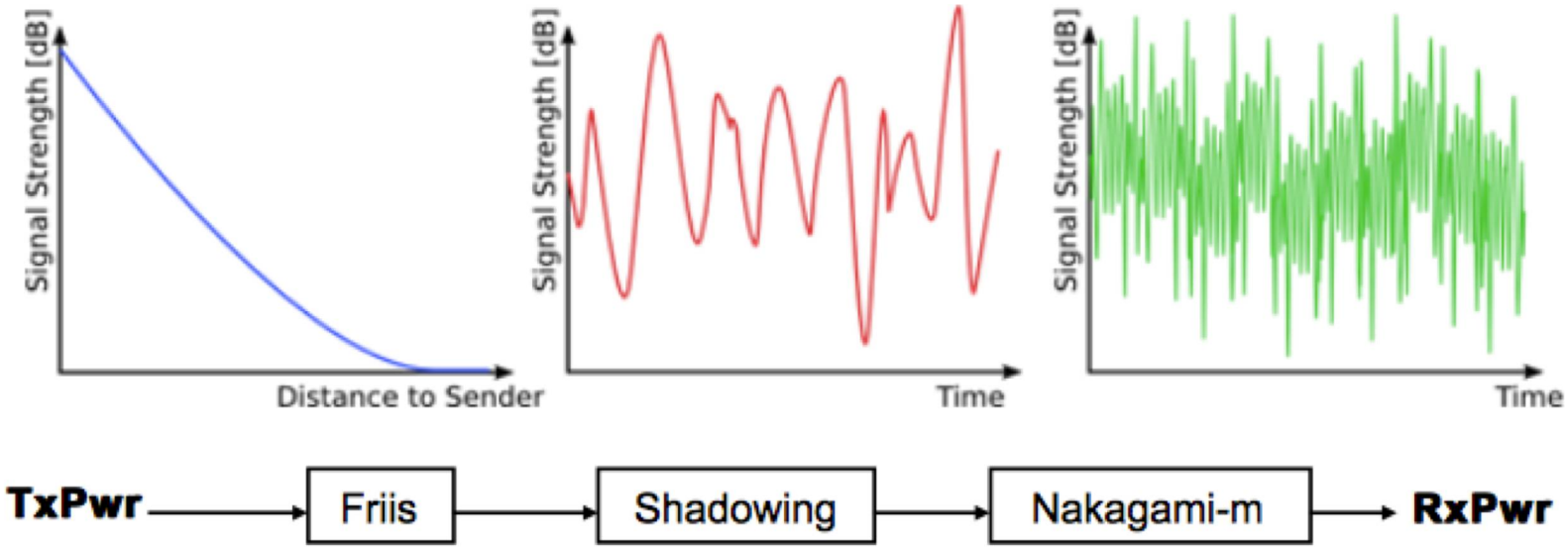
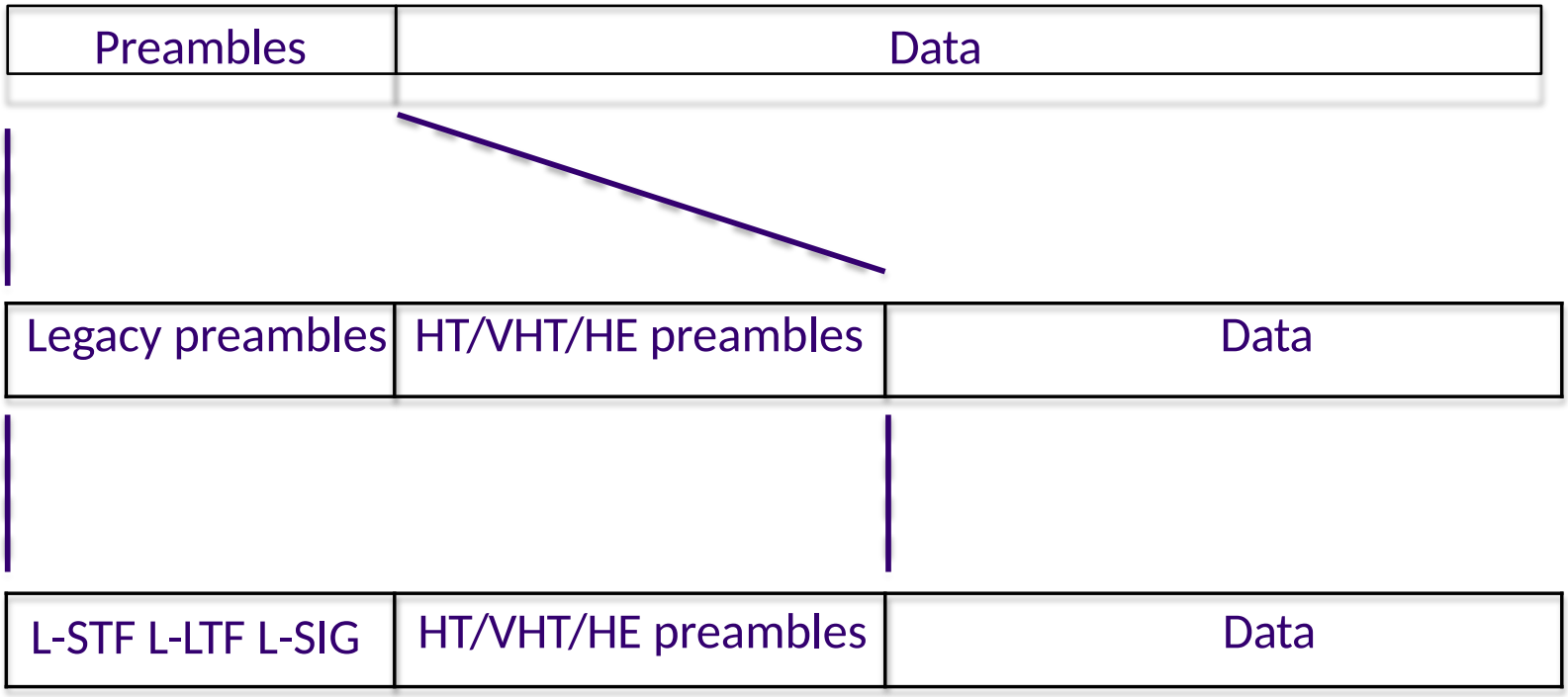


Figure source: Unknown

Modelos de detección de preámbulos y captura de tramas

> En la práctica, una trama WiFi se detecta primero (y se sincroniza) mediante un campo de **preámbulo**

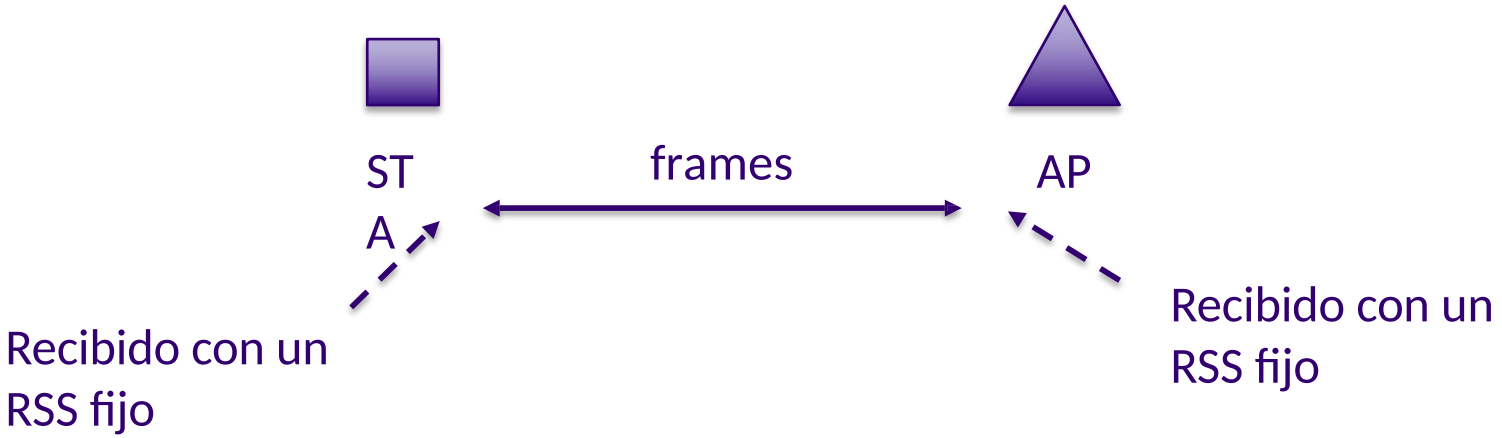


Modelos de detección de preámbulos y captura de tramas

- Un **'ThresholdPreambleDetectionModel'** es configurado por defecto por los helpers Wi-Fi
  - Atributo "Threshold": default 4 dB
  - Atributo "MinimumRssi": default -82 dBm
- Un **'SimpleFrameCaptureModel'** está disponible pero debe ser agreagado (WifiHelper::SetFrameCaptureModel())
  - Solo habilitado para YansWifiPhyHelper
  - Atributo "Window" : default 16us
  - Atributo "Margin" : default 5 dB

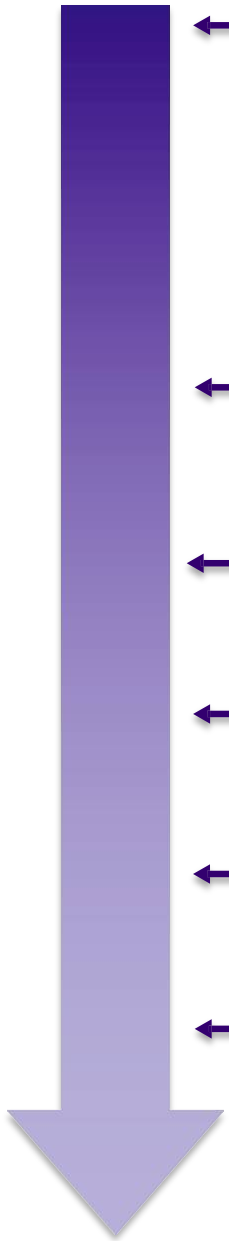
(Demo) wifi-simple-infra.cc

- wifi-simple-infra.cc utiliza un modelo especial de pérdida de propagación 'FixedRss' que impone que la intensidad de la señal recibida (RSS) sea un valor configurado
- La entrega de paquetes se rige por un modelo de **detección de preámbulos** y un **modelo de error Wi-Fi**



Potencia de la señal y Wi-Fi

- dBm es la referencia a los decibelios sobre 1 mW
- 0 dBm = 1 mW
- +/- 3 dB = */÷ un factor de 2 en una escala lineal
- +/- 10 dB = */÷ un factor de 10 en una escala lineal



16 dBm: Potencia máxima de transmisión Wi-Fi a 5 GHz

-62 dBm: Umbral de "detección de energía" requerido

-82 dBm: Umbral de "Detección de preámbulo" requerido

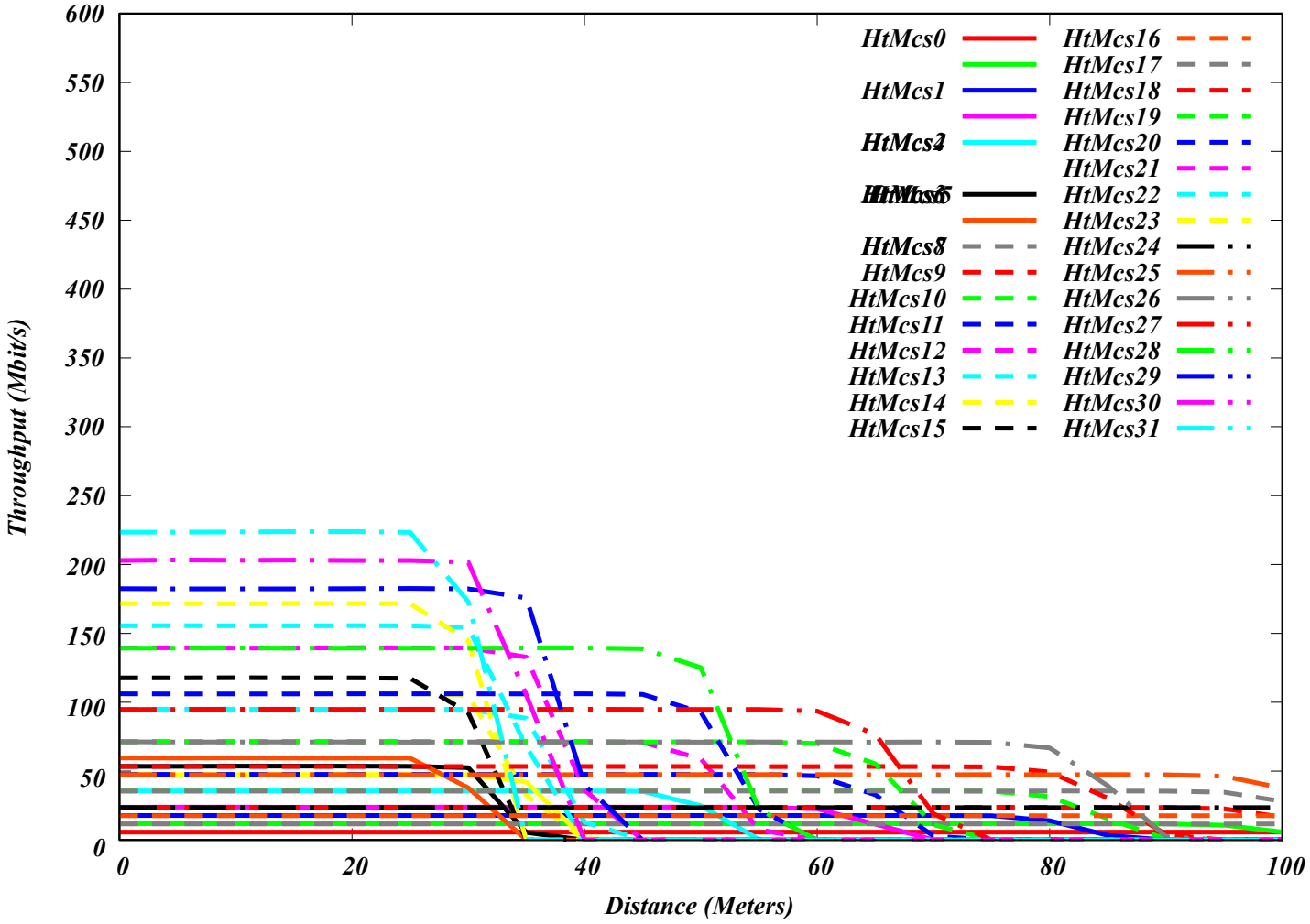
-90 dBm: Nivel mínimo de potencia recibida en tarjetas típicas

-94 dBm: Potencia de ruido incluyendo la figura de ruido WifiPhy de 7 dB por defecto

-101 dBm: Suelo de ruido térmico para 20 MHz a temperatura ambiente.

Throughput vs distance for 802.11n modulation

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



Rate control

- Rate control se refiere al algoritmo utilizado para seleccionar un esquema de modulación y codificación (MCS) adecuado para enviar un paquete (primer intento de transmisión y posibles reintentos)
- Rate control es un subconjunto del problema generalizado de asignación de recursos, que puede incluir también el control de la potencia, el scheduling, el rango de detección de la portadora, el beamforming, etc. y puede realizarse conjuntamente con el control del rate (en teoría)
- En la práctica actual, se realiza un control de velocidad puro (selección de MCS por paquete y por reintento solamente) o un control conjunto de velocidad y potencia

Rate control in ns-3

- ns-3 tiene muchos algoritmos de control de rate que sólo funcionan para Wi-Fi no HT (es decir, anterior a 802.11n), incluyendo controles conjuntos de velocidad y potencia
- Para 802.11n y más recientes, sólo hay cuatro controles de rate disponibles
  1) ConstantRateWifiManager
  2) IdealWifiManager
  3) MinstrelHtWifiManager
  4) ThomsonSamplingWifiManager

Ejemplos para revisar (si el tiempo lo permite)

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