Integrating an 802.11 physical layer simulator in NS-3

Stylianos Papanastasiou

Signals and Systems
stypap@chalmers.se
Outline

Motivation

Planned activity

The implementation

Validation, tradeoffs

Future Work
Physical layer abstractions in network level simulators

- Network simulators typically abstract the physical layer
- They use frame-level statistics
  - What happens to the frame?
  - (c.f. what happens to the signal?)
- Approach works well enough if the abstraction is reasonable
  - Ok in wired, not so well in wireless
Reasonable abstractions

- Not always clear what to abstract and what not to
- It is not easy to extrapolate “general” characteristics
- Scenario-specific minute measurements may matter
- May want to examine network performance of PHY techniques
  - Receiver structure (Interference cancellation)
  - Synchronisation
  - Channel estimation
Typical channel model development

- Channel Measurements $\rightarrow$ Channel Model $\rightarrow$ BER $\rightarrow$ PER
- Lost in translation? BER to PER conversion not trivial
- Would be useful to utilise Channel Model or Measurements directly
  - Expediency (no need to make PER calculations)
  - Accuracy (no compromises)
Big picture - Layer division

- Network vs physical layer communities
- Packet vs signal time samples
- Coarse vs fine grained
- Network vs Link
- Join the two perspectives in a single simulation tool
Integrate physical layer simulator for OFDM-based IEEE 802.11 communication in a network simulator
Course of action

1. Use NS-3 – a popular, well maintained simulator
2. Develop and validate a physical layer simulator
   ▶ Based on well-established signal processing library (IT++)
   ▶ May have value as a standalone deliverable
3. Integrate the two – release to community for feedback
Why use NS-3

▶ Very well maintained
▶ Open source (GPL)
▶ Speed and accuracy are a priority for maintainers
▶ Flexible architecture and very good documentation
▶ Actively used for research
NS-3 architecture

Higher Layers

WifiNetDevice
- handles probing and (re)association with base

WifiMac

DcaTxOp

MacRxMiddle
- packet queueing
- fragmentation
- retransmissions

MacLow
- handles RTS/CTS/DATA/ACKs

DCF Manager
- Notify when access is granted

WifiPhy

WifiChannel
NS-3 architecture (implementation)
A new physical layer for NS-3

MAC
(bits)

PHYCICAL
(time samples)

CHANNEL
(time samples)

MacLow

PhySimEmuWifiPhy

Convolutional Encoder
Block Interleaver
Scrambler
Modulator

Interference Manager
Signal Detection
Channel Estimation
Demodulator
Block Deinterleaver
Convolutional Decoder

Propagation Delay Model
Propagation Loss Model
Channel modelling

- Channel models operate on time sample level
- Several have been implemented already
- Channel models can be “chained” in sequence
Frame reception

- Frame reception handled realistically
- Signal detector tries to “lock-on” to frame
- Header is decoded and if successful the payload is examined too
At all times surrounding interference is accounted for.
For the superposition of signals some alignment is needed.
Frame reception (Old vs New)
Integration into NS-3 state machine

- Reception process represented by 4 events
- Computations performed at each event
- Possibilities of computational “shortcuts”
Example usage (standard model)

```
YansWifiChannelHelper wifiChannel;
YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default();

wifiPhy.SetChannel(wifiChannel.Create());

WifiHelper wifi = WifiHelper::Default();
NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default();
wifiMac.SetType("ns3::AdhocWifiMac");

NodeContainer nodes;
nodes.Create(2);
// Install WifiPhy and set up mobility
wifi.Install(wifiPhy, wifiMac, nodes);
```
Example usage (new model)

```cpp
PhySimWifiChannelHelper wifiChannel;
PhySimWifiPhyHelper wifiPhy = PhySimWifiPhyHelper::Default();

wifiPhy.SetChannel(wifiChannel.Create());

WifiHelper wifi = WifiHelper::Default();
NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default();
wifiMac.SetType("ns3::AdhocWifiMac");

NodeContainer nodes;
nodes.Create(2);
// Install WifiPhy and set up mobility
wifi.Install(wifiPhy, wifiMac, nodes);
```
Validation

- Validate produced samples against 802.11 Annex G.
- Common-sense cases (BER vs SINR)
- Verification against Octave/Matlab simulator and theoretical BER curves
- Against real transceivers done by KIT at CMU
Static pathloss, 500 bytes frame size, 10 Hz transmission freq.
What is the tradeoff?

▶ Simulation becomes much more demanding
▶ Computational costs + memory overheads
  ▶ Preliminary results indicate slowdown factor of 200 for simple pathloss
  ▶ ...for complex channel slowdown factor increases to $10^3$
  ▶ Have to store whole frame representation in memory
▶ Work underway at Karlsruhe to increase speed for single user systems (OpenCL)
Future work

▶ Integrate several new channel estimators (sometime in October/November)
▶ Implement more channel/frontend effects (phase noise, antenna diversity etc...)
▶ Implement environment/node geometry effects
Where can I get all this?

- All items are open source (GPL) and available online
- Standalone physical layer simulator available
- ...as well as full NS-3 implementation
- Code is under review for inclusion in NS-3 in the near future
Thank you for your attention

Questions?

- **Standalone physical layer simulator**
  Does not include high-level protocols, mobility, etc...

- **NS-3 simulator with new physical layer model**
  Ported to the latest version of NS-3 (3.9)
  [http://www.lefrog.at/NS-3.9-PhysimWiFi-1.0.tar.bz2](http://www.lefrog.at/NS-3.9-PhysimWiFi-1.0.tar.bz2)