Spectrum-aware Channel and PHY layer modeling for ns3

Nicola Baldo
Centre Tecnològic de Telecomunicacions de Catalunya
Av. Carl Friedrich Gauss 7, 08860 Castelldefels, Barcelona (Spain)
nbaldo@cttc.es

Marco Miozzo
Centre Tecnològic de Telecomunicacions de Catalunya
Av. Carl Friedrich Gauss 7, 08860 Castelldefels, Barcelona (Spain)
mmiozzo@cttc.es

ABSTRACT

The recent interest in tackling the frequency-dependent aspects of communications is pushing a demand for spectrum awareness in simulation tools. Unfortunately, most current network simulators do not deal with this aspect in a satisfactory fashion. In this paper, we present a framework that we developed for the ns3 simulator with the aim of addressing this issue. We discuss the modeling assumptions on which our design is based, and we describe in detail the implementation of the framework as well as of the PHY layer modules that we developed on top of it. Finally, we report some performance evaluation results which show that the proposed solution can achieve a good spectrum modeling accuracy with a modest increase in the computational load.

Categories and Subject Descriptors

I.6 [**Simulation and Modeling**]: General, Model Validation and Analysis, Model Development; C.2.6 [**Computer-Communications Networks**]: Internetworking

General Terms

Modeling, Simulation, Physical Layer, Channel, Propagation, Spectrum, Networking

1. INTRODUCTION

Recent advances in the physical layer technology have been driving an increase in the performance of wireless communication systems. One of the factors driving this phenomenon is the ability of the physical layer to tackle the frequency dependency of the communication channel in order to get most of the performance out of it. As a consequence, there is a strong interest in simulation tools which can model frequency-dependent phenomena and therefore support research and product design and development. Even though this type of technology normally involves the physical layer only, it is of great interest to be able to model it in a satisfactory fashion even in network simulators, in order to be able to analyze system-level issues.

In the following, we will outline some example cases to better argument this statement. The first case we consider is that of Orthogonal Frequency Division Multiplexing (OFDM), which is the modulation scheme adopted by, among others, IEEE 802.11a/g (WiFi) and IEEE 802.16 (WiMAX). With this type of modulation, the communication is divided over several subcarriers, each of which can experience different channel quality due to, e.g., frequency-selective fading and frequency-dependent interference patterns. The radio models in network simulators such as ns2 and ns3 typically assume a flat channel frequency response, thereby neglecting these aspects. As a consequence, these network simulators are not adequate for investigating some realistic problems such as Adjacent Channel Interference issues in WiFi, or interference between devices belonging to different technologies (WiFi, Bluetooth, microwave ovens, etc.) operating in the unlicensed 2.4GHz ISM band.

Moreover, emerging wireless technologies typically implement advanced physical layer solutions to tackle the frequency dependency of the communication channel and to get most of the performance out of it. For example, using the same modulation and coding schemes for all subcarriers would yield sub-optimal performance, since the subcarriers typically experience different Signal-to-Noise Ratio values. Instead, Adaptive Bit Loading strategies focus on adapting the modulation and coding scheme on a per-carrier basis; these strategies, in conjunction with the use of efficient channel coding techniques, can significantly increase the communication performance. Another example is that of Orthogonal Frequency Division Multiple Access systems (OFDMA); like in OFDM, in OFDMA the available spectrum is partitioned in subcarriers; however, unlike OFDM, at a given time different sets of subcarriers can be assigned to different users. Since the performance of different sets of carriers can vary from user to user, advanced scheduling algorithms can opportunistically allocate subcarriers to users in order to enhance the network capacity. All the aspects just mentioned are crucial components of recently introduced wireless technologies, such as IEEE 802.16e Mobile WiMAX an Long Term Evolution (LTE). It follows that an accurate modeling of those technologies in network simulators would require support for the frequency-dependent aspects of channel and physical layer modeling.

Finally, we would like to mention the recently emerged Cognitive Radio / Dynamic Spectrum Access (CR/DSA) paradigm, which consists of allowing unlicensed devices to opportunistically access those spectrum bands which are un-

NSTOOLS October 19, 2009 - Pisa, Italy.

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7647
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7647

used or underused by their license holders, subject to the constraint of not causing any harmful interference to licensed users. There are many challenges in the design of CR/DSA systems, such as for instance detecting unused/underused spectrum bands, coordinating unlicensed devices for spectrum access, and avoiding interference to licensed users. The practical use case that is typically envisioned is that of reusing underused TV bands for unlicensed wireless access networks or ad-hoc networks. Research in this field has generated a significant interest into network level simulators which are able to model how different wireless technologies make use of the spectrum.

The examples just described highlight that there is a huge need for a network simulator which can effectively model how the frequency-related aspects of the communication at the physical layer affect the performance at the upper layers. The problem is that most well-known network simulators, such as ns2 and ns3, typically feature channel models which assume a flat frequency response, thereby neglecting these aspects.

It is to be noted that some existing simulation tool already supports, to some degree, spectrum-aware physical layer and channel models. One of them is the WiMAX simulator developed by the WiMAX Forum [1], which supports the use within a network-level simulator of performance models obtained by running a detailed multi-carrier link-level simulator. Another example is WINSE [2], which is probably one of the most complete modules for WiMAX, since it accurately models both the OFDM and OFDMA PHY layers by means of propagation traces generated with dynamic system simulators. However, we note that the fact that channel and PHY models are obtained from dedicated link-layer simulators implies that only those scenarios which have been pre-simulated by the link-layer simulator can be run in the network simulator. Furthermore, these tools are not available for public use. Among open source network simulation software, we note that the MPhy framework [3] included in Miracle [4] provides some spectrum modeling capabilities, such as the representation of the spectral mask used for transmission and the frequency response of the RF filter at the receiver; however, other aspects of channel modeling, in particular propagation modeling, are still addressed in a single-frequency fashion. The WiDe framework, recently developed within the DEMOCLES project [5], further extends Miracle's MPhy framework to provide a detailed multi-frequency channel and physical layer model for WiMAX; however, frequency-aware modeling support is not available for other wireless technologies.

To summarize, while there is a significant demand for spectrum-aware simulation tools, a publicly available network simulator which can support spectrum modeling for different wireless technologies is still lacking. In this paper, we present the framework that we developed for the ns3 simulator with the aim of filling this gap; this framework is publicly available [6] under the terms of the GPL license. In the following sections, we will first describe the general principles and modeling assumptions on which the proposed solution is based; subsequently, we will provide an overview of the implementation of both the core of the framework and the particular channel and phy models that we developed on top of it. Finally, we will evaluate the performance of the proposed framework, discussing in particular the tradeoff between computational complexity and modeling accuracy.

2. SPECTRUM MODEL

The crucial goal that drove the design of our framework is the interest in being able to model 1) how devices irradiate power into the spectrum, and 2) how to represent the variations of this irradiated power that occur due to, e.g., propagation in the channel and signal processing at the receiver. We refer to spectrum in general; most commonly, it will be the electromagnetic spectrum, but in some cases it might be something else, such as for instance the acoustic spectrum when dealing with underwater communications.

In theory, since the time and frequency domain are related each other by a transform operation, the complete time representation of a signal is all that is needed to determine its frequency occupancy. However in network simulators, due to practical complexity issues, the time-domain representation of transmitted signal is very far from being complete. In fact, normally all signals are modeled as pulses with a single amplitude value which is constant for the whole duration of the signal; physical layer aspects such as modulation and frequency upconversion are not included in this type of signal representation. As a consequence, in order to have a practical way of modeling frequency-dependent phenomena within network simulators, we choose to add to the very simple time-domain signal representation some further information about what in the real world would be its frequency-domain counterpart; in particular, we represent this information by means of the Spectral Power Density (SPD) of the signal. The reason for this choice is that the knowledge of the SPD is sufficient for reasonably accurate frequency-dependent channel and PHY layer models, such as the one we will describe in Section 5, and at the same time the representation of a SPD does not have the huge complexity of a complete time-domain signal representation.

The most generic way to represent a SPD would be to use a function space $\Phi = F \rightarrow \mathbb{R}$, where $F \subset [0, \infty) \subset \mathbb{R}$ is the portion of the spectrum we are interested into. Algebraic operation in the function space provide means for properly modeling frequency-dependent phenomena. As an example, the fact that a given node is transmitting could be modeled by its SPD $\varphi_t(f) \in \Phi$, with $f \in F$; the propagation between that node and a candidate receiver could be modeled by the frequency-dependent propagation gain $\varphi_p(f) \in \Phi$, and the SPD of the received power would then be obtained as $\phi_r(f) = \varphi_t(f)\varphi_p(f)$.

Obviously, implementing a function space defined over a set with infinite cardinality is not practical, so the practical approach we adopt is to divide the portion of spectrum we are interest into in a set of finite frequency bands, and to model frequency-dependent phenomena as being constant within each band. Clearly, the higher is the number of bands we use, the higher will be the modeling accuracy in the frequency domain, and the higher will be the computational load required to use the corresponding model.

Formally, we define a *spectrum model* $\mathcal{S}$ as a set $\mathcal{S} = \{B_i\}$, $i = 1, \ldots, N$, where $B_i = [f_{i,l}, f_{i,h}]$ is a frequency band identified by its lower and upper frequencies $f_{i,l}$ and $f_{i,h}$. Since the cardinality of $\mathcal{S}$ is finite, each function φ in the space $\mathcal{S} \rightarrow \mathbb{R}$ corresponds to a vector of finite dimension φ_i, $i = 1, \ldots, N$, and it can therefore be implemented.

In practice, the choice of the particular spectrum model to be used will depend on the wireless technology being considered, as well as on the assumptions made for the modeling. As an example, we might have two models for the 802.11g

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7647
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7647

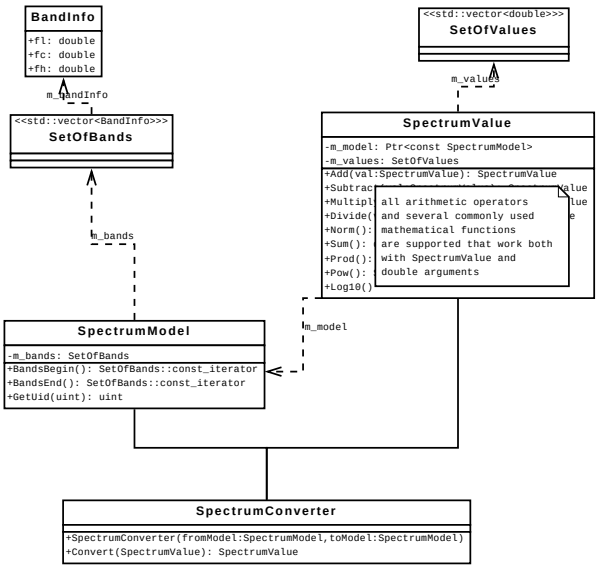

Figure 1: UML Class Diagram for the spectrum model implementation

technology in the 2.4 GHz ISM band: a coarser (and faster) one where the spectrum modeling granularity is the 5 MHz inter-channel spacing, and a finer one (but more computationally demanding) with a granularity of 312.5 kHz, which corresponds to the OFDM subcarrier spacing and therefore enables us to, e.g., simulate propagation phenomena on a per-carrier basis.

An important feature that we want to support in our spectrum modeling framework is the possibility of working with different spectrum models within the same simulation. The reason for this is that we want to leave the choice of the particular spectrum model to the developer of each wireless technology module, while making it possible to use in the same simulation modules developed with different spectrum models without requiring any porting. For this purpose, we want to define the conversion operation between spectrum models. Let $[f_{i,l}, f_{i,h}]$ and $[g_{j,l}, g_{j,h}]$ be two generic bands belonging respectively to the spectrum model $\mathcal{S}_1$ and $\mathcal{S}_2$. We define the conversion matrix $\mathbf{C} = \{c_{i,j}\}$, where the coefficients $c_{i,j}$ are determined as

$$c_{i,j} = \min\left(1, \frac{\max\left(0, \min(f_{i,h}, f_{j,h}) - \max(f_{i,l}, f_{j,l}),\right)}{f_{j,h} - f_{j,l}}\right) \quad (1)$$

With this definition, for any representation $\varphi \in \mathcal{S}_1$ we can calculate the corresponding representation $\psi \in \mathcal{S}_2$ as the matrix product $\psi = (\varphi\mathbf{C})^T$, where the superscript ^{T} indicates the transpose operation.

3. MODEL IMPLEMENTATION

The implementation of the model we just described consists of a set of `c++` classes which are represented in Figure 1.

First of all, the concept of frequency band is realized by the `BandInfo` struct. This struct contains the value of the lower and upper frequency limits which define the band, as per the definition of B_i that we gave in Section 2. Furthermore, `BandInfo` contains also the value of the reference frequency which is to be used for frequency-dependent calculations, such as for instance the calculation of propagation loss. In most practical cases, this value will be the center frequency of the band.

The concept of a set of bands is implemented by the `SetOfBands` class, which is implemented as a vector of `BandInfo`. The `SpectrumModel` class owns a `SetOfBands` instance through a private member variable; this implements the concept of spectrum model as defined in Section 2. Furthermore, the `SpectrumModel` class provides other features which are aimed at enhancing the spectrum modeling API for the developer. A set of constructors is provided, which enable the easy creation of spectrum models; as an example, a `SpectrumModel` can be created just by specifying the set of central frequencies (band boundaries will be calculated automatically). Instances of `SpectrumModel` cannot be modified after construction; for this reason, the BandInfo structure which make up a `SpectrumModel` can be accessed only via constant iterators. The reason for this choice is that `SpectrumModel`s are intended to be defined on a per-device basis by the device developer, and as such are expected to be statically instantiated from the beginning of the simulation; furthermore, all devices using the same spectrum model will be expected to use the same `SpectrumModel` instance. `SpectrumModel` instances are assigned an unique id upon creation, so that it is possible to test at run time if, e.g., two transmissions are represented using the same spectrum model, just by comparing the unique ids of their associated spectrum models. Finally, the `SpectrumModel` class inherits from the `RefCountBase` class, thus enjoying easier memory management by means of automatic reference counting.[1]

The class `SpectrumValue` implements the concept of a function space defined over a spectrum model. Each `SpectrumValue` instance possesses two particular member variables: a reference to a previously instantiated `SpectrumModel`, which represents the domain of the function space, and a vector of values implementing the codomain of a particular function in the function space. As a consequences, different values of the same spectrum model (i.e., different functions belonging to the same function space), correspond to instances of `SpectrumValue` which have references to the same `SpectrumModel`.

We recall that `SpectrumValue` is to be used for the representation of spectral power densities, frequency dependent propagation and processing gains, etc., and its instances are therefore expected to be manipulated by means of mathematical operations. For this purpose, the `SpectrumValue` class provides support for most common mathematical vector operations, which are implemented by means of member and friend operators and methods. Additionally, `SpectrumValue` offers vector-like semantics (index operator `[]`, mutable iterators, etc.) to support easy initialization and assignment.

Finally, support for the conversion between different spectrum models is provided by the `SpectrumConverter` class. Upon creation of this class, the user is required to provide references to two `SpectrumModel` instances; the `SpectrumConverter` class will then calculate the coefficients for the conversion of `SpectrumValue` instances between the two spectrum models according to the specification that we gave

[1]The `RefCountBase` class is part of the official ns3 simulator; the reader is referred to [7] for a detailed description of its functionality.

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7647
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7647

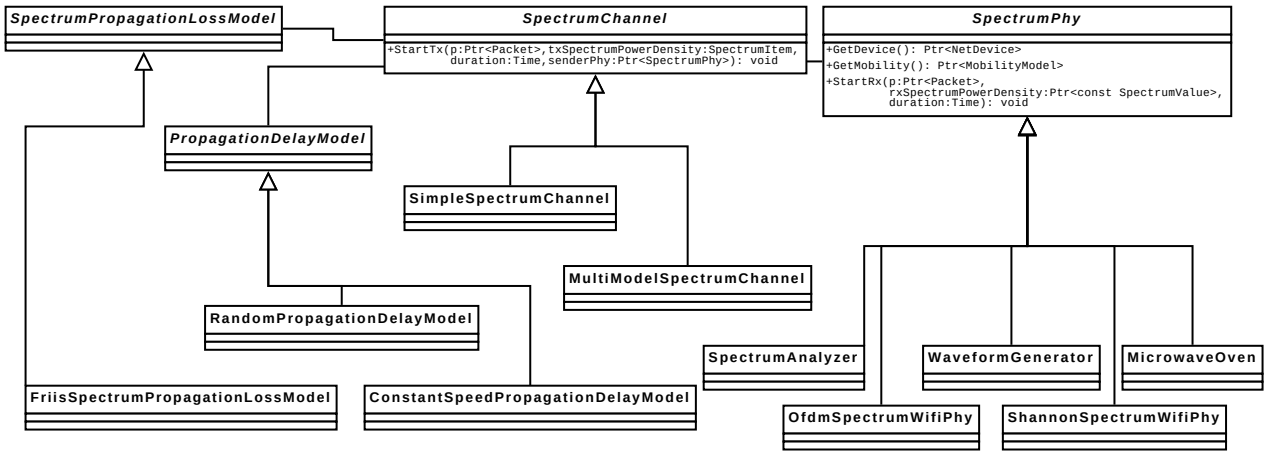

Figure 2: UML Class Diagram for the spectrum channel and PHY layer implementation

in Section 2. After creation, the `convert()` method of the `SpectrumConverter` instance can be used to convert `SpectrumValue` instances between the two `SpectrumModel`s which have been specified.

4. SPECTRUM-AWARE CHANNEL AND PHY LAYER INTERFACE

In addition to providing a set of classes implementing spectrum modeling functionality, we also provide a set of abstract classes which define a spectrum-aware interface between the PHY layer and the wireless channel. This interface enables the development of spectrum-aware devices and channel models which can interoperate with each other.

The corresponding set of classes is represented in Figure 2. The channel/PHY interface is defined by the abstract base classes `SpectrumChannel` and `SpectrumPhy`, and it is primarily devoted to the transmission and receptions of signals. For this purpose, two member methods are defined. The first one is `SpectrumChannel::StartTx`, which is to be called by a PHY layer implementation in order to perform a transmission. This methods accepts the following parameters:

- a reference to the `Packet` associated with the transmission being performed.
- the SPD of the transmission in the form of a `SpectrumValue` class instance. The SPD is to be expressed in linear units. The exact units will depend on the type of transmission medium involved; in particular, for radio communications we define that Watt units are to be used, whereas for underwater acoustic communications Pascal units are to be used. Other eventual transmission media would need to be defined.
- the duration of the transmission, expressed as a `Time` value. This is the amount of time that the SPD of the signal assumes the value expressed by the previous parameter.
- a reference to the `SpectrumPhy` instance invoking the method. This is expected to be used by `SpectrumChannel` implementations in order to avoid propagating transmissions back to the sender, e.g., by comparing the pointers of the sender and of the candidate receiver `SpectrumPhy`.

The second method of importance for the Channel-PHY interface is `SpectrumPhy::StartRx`, which is to be called by a channel implementation to notify a PHY that a signal is being received. The method accepts the following parameters:

- a reference to the `Packet` associated with the signal being received. This will normally be a reference to a copy of the packet passed by the sender PHY when performing the call to `SpectrumChannel::StartTx` that triggered this reception;
- the SPD of the signal being received, represented by a SpectrumValue instance. Note that the `SpectrumModel` being used for this SpectrumValue is expected to be one that the receiver SpectrumPhy can understand. In the case that the sender SpectrumPhy used a SpectrumModel that the receiver does not understand, it is left to the particular SpectrumChannel implementation to take care of the necessary conversion.
- the duration of the signal being received. This will normally be exactly the same value passed by the sender PHY.
- a reference to the sender `SpectrumPhy`. This is used by the receiving `SpectrumPhy` to determine, e.g. by checking its object type, whether the sender PHY belongs to a wireless technology it understands. If this is the case, reception can be attempted. Otherwise, the incoming signal will only be perceived as interference. We note that this type of perfect knowledge about the type of incoming signal is not possible in real devices, so this is a modeling abstraction that we introduce to be able to support the simulation of multiple wireless technologies operating simultaneously in the same spectrum.

The `SpectrumChannel` class is intended not only to dispatch transmissions between PHY layers, but also to host channel propagation models. Of course, it is of great interest to be able to use different types of such models interchangeably. For this purpose, an interface needs to be

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7647
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7647

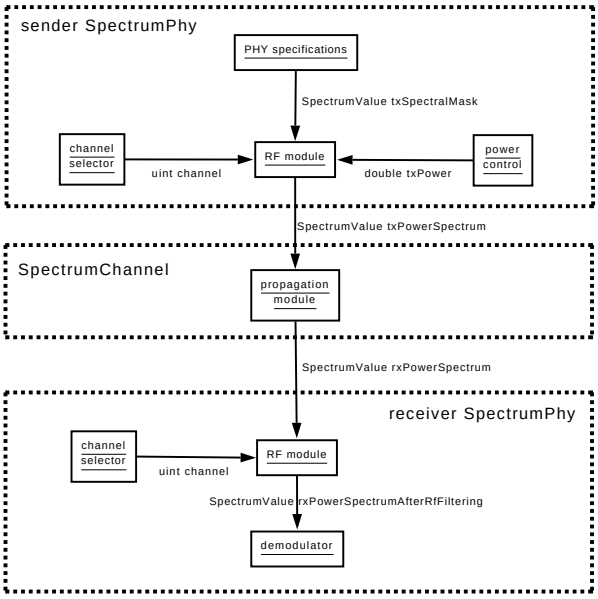

Figure 3: Example use case for the Spectrum class hierarchy

defined. We argue that the best option to model the delay in a spectrum-aware channel is to use as-is the `PropagationDelayModel` interface already provided within the ns3 wifi device; the reason for doing this is that introducing a frequency-dependent delay in an event-based network simulator would yield significant complications[2], and the benefits of the additional modeling accuracy that would be introduced are arguably not significant enough for a network simulator, once we consider all other modeling assumptions that are commonly made.

As for the propagation loss, we introduce a new interface for frequency-dependent models by means of the abstract base class `SpectrumPropagationLossModel`. In particular, the propagation loss is calculated by its `CalcRxPower` member method, which takes as input the spectral power density (represented by another `SpectrumValue`) of the signal to be attenuated, together with references to the `MobilityModel`s being used by the sender and the receiver. The output of the method is the frequency-dependent propagation loss represented as a `SpectrumValue`. We note that `SpectrumPropagationLossModel` works exactly as the `PropagationLossModel` already included in ns3, with the only difference that both the signal power and the attenuation are frequency-dependent.

To better explain the interaction between the different classes that we just introduced, we will now discuss an example use case, which is represented in Figure 3. In this example, the sender `SpectrumPhy` determines the transmitted power spectrum taking into account the spectral mask requirements of the technology it implements, as well as the current transmission power and channel settings. The transmitted power spectrum is then sent to the `SpectrumChannel` object (along with the corresponding `Packet`, which is omitted in the figure), which runs a propagation model to determine the received power spectrum for every receiving `SpectrumPhy`. Subsequently, the receiving `SpectrumPhy` applies RF filtering to the received power spectrum, and passes the resulting `SpectrumValue` to the demodulator. Finally, the demodulator object will classify all received signals into useful signals and interfering signals, and will determine whether the reception of useful signals is successful according to some error model like the ones which we will describe in the next section. As a side note, we would like to point out that the framework we present in this paper is by no means limited to the example architecture just described – in fact, we expect that the architecture of `SpectrumPhy` implementations will vary significantly among different wireless technologies.

[2]e.g., a transmission, which is normally modeled as a single packet, would need to be split in several fragments, each for a given portion of the spectrum, and each fragment would experience a different delay.

5. IMPLEMENTED CHANNEL AND PHY LAYER MODELS

In this section we describe the actual models that we implemented using the framework described in the previous section.

5.1 Spectrum channel models

We provide two implementation of the `SpectrumChannel` class, which are named `SimpleSpectrumChannel` and `MultiModelSpectrumChannel`. As the name states, the first one is simpler, due to the fact that it relies on the assumption that all devices attached to the channel will be using the same `SpectrumModel`. The second class, on the other hand, supports the use of different `SpectrumModel`s by the attached devices. This is realized by asking the user to specify, when attaching devices to the channel at simulation setup, the particular `SpectrumModel` that each device will be using; upon this action, a `SpectrumConverter` will be automatically created for each pair of different `SpectrumModel`s. At run time, `MultiModelSpectrumChannel` will use the `SpectrumConverter` instances previously created to convert each transmission from its native spectrum model into every other spectrum model being used. Thus, the number of spectrum conversion operations for each transmitted packet is equal to the number of spectrum models being used in the simulation minus one.

5.2 Spectrum propagation loss models

The only model we implement so far is the well-known Friis propagation model [8]. We note that, strictly speaking, this model is frequency-dependent, so in spite of its simplicity it is still representative of the rationale of our proposed framework. In practice however, as we will discuss in Section 6, the difference in attenuation with respect to frequency that is obtained with this model is almost negligible for most commonly encountered wireless technologies, due to the fact that they use a relatively narrow band. We plan to develop more complex propagation models, in particular frequency-selective fading models such as the one described in [9], in the near future.

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7647
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7647

5.3 Spectrum devices

In this section we describe the PHY layer implementations that we developed by implementing the `SpectrumPhy` interface. Each implementation is described in the following subsections.

5.3.1 WaveformGenerator

The `WaveformGenerator` class implements a simple device radiating power into the spectrum. Upon simulation configuration, the user is expected to provide it with the SPD and duration of its transmissions, and to schedule start and stop events for the generator. During the simulation a `WaveformGenerator`, during its active periods, will transmit back-to-back copies of the signal whose parameter had been previously specified. This can be useful, for example, to generate interference which will disturb the operation of other wireless technologies. We note that, since a waveform generator is not a communicating device, this class is not intended to be connected to any upper protocol, and for this reason it is not used within any particular NetDevice implementation.

5.3.2 MicrowaveOven

Like `WaveformGenerator`, also the `MicrowaveOven` class implements a non-communicating device whose primary purpose is to be used as a source of interference. The main feature of this class is that it models the emissions of microwave ovens, according to the model in [10]. This model was developed with the aim of supporting the study of interference between microwave ovens and WiFi stations in the 2.4 GHz ISM band.

5.3.3 SpectrumAnalyzer

The aim of this device is to perform one of the tasks for which spectrum analyzer are normally used for, i.e., to measure and record the total power irradiated into the spectrum at a particular location. In particular, the `SpectrumAnalyzer` class averages the SPD of all incoming signals over a user-configurable time duration. The resulting cumulative SPD for every measurement interval is then recorded to a file for later analysis.

5.3.4 OfdmSpectrumWifiPhy

Starting from the code of the `YansWifiPhy` class provided within ns3, we developed a PHY layer implementation for a WiFi device which uses the `SpectrumPhy`/`SpectrumChannel` interface instead of the single frequency `WifiPhy`/`WifiChannel` interface. This feature is then exploited for the implementation of an error model based on the Effective SNR (ESNR) calculation. ESNR is an abstraction which has been introduced in PHY layer modeling for the purpose of characterizing the performance of OFDM systems in which each subcarrier $n = 1 \dots N$ experiences a different SNR value γ_n[11]. In particular, the ESNR γ is defined to be the value of SNR that, in a case where SNR is constant over all subcarriers and equal to γ, would give the same error performance that we get for the case in which SNR is equal to γ_n and varies between subcarriers. In practice, ESNR is approximated with functions of the type

$$\gamma = \Theta^{-1}\left(\frac{1}{N}\sum_{n=1}^{N}\Theta(\gamma_n)\right) \quad (2)$$

where the particular function Θ depends on the wireless technology being used. In our case, we adopted the model given in [12], which is valid for the OFDM modulation schemes used in IEEE 802.11g. The obtained ESNR value is then fed to the error model already implemented in YansWifiPhy in order to infer whether a packet is in error or can be correctly received.

5.3.5 ShannonSpectrumWifiPhy

The research and development effort invested in the physical layer over past decades has been effective in bringing the capacity of real communication devices close to the Shannon capacity limit. This has come at the cost of a huge increase in device complexity. In wireless network simulation, the standard practice for improving the accuracy of physical layer models has always been to try to mimic in a closer way the behavior of real PHY layer implementation. However, due to the increased complexity of the PHY layer, this approach is currently not any more viable for network simulators.

Given that all this increase in PHY layer complexity is aimed at getting closer to the Shannon capacity, using the Shannon capacity itself for the characterization of the wireless link could actually yield a better tradeoff between modeling accuracy and computational complexity. The `ShannonSpectrumWifiPhy` class that we developed takes exactly this approach to implement an error model for the IEEE 802.11 OFDM PHY. In particular, let $t = 1 \dots T$ be an index denoting the time intervals over which the noise and interference that the considered transmission is facing remain constant, and let D_t be the duration of this interval. The constant value of the SINR on a given subcarrier n in interval t is then denoted as as $\gamma_{n,t}$. `ShannonSpectrumWifiPhy` calculates the maximum capacity C_t within a given time interval t using the Shannon-Hartley theorem, i.e.:

$$C_t = \sum_{n=1}^{N} B_n \log_2(1 + \gamma_{n,t}) \quad (3)$$

where B_n is the bandwidth of subband n. The maximum amount of bits L_t which could be delivered in interval t is then calculated as $L_t = C_t/D_t$, and the total amount of bits which could be delivered within that transmission is given by $L = \sum_{t=1}^{T} L_t$. This value is then compared with the actual packet size $\overline{L}$ to model communication errors; in particular, the packet is said to be in error if $L < \overline{L}$, since this means that, due to insufficient SNR, reliable delivery would not have been feasible even with the best possible channel modulation and coding strategy.

As a final note, we would like to point out that the name `ShannonSpectrumWifiPhy` is representative of the fact that we used the physical layer model just described within an implementation of the WifiPhy interface. However, the model itself is generic and can be used for other wireless technologies as well.

6. PERFORMANCE EVALUATION

To evaluate the computational load of our framework, we ran some simulations with different levels of spectrum modeling accuracy, and compared the results with `YansWifiPhy`. For this purpose, we used two modified versions of the `examples/wifi-ap` simulation script included within ns3. The execution times obtained for different values of the number of STAs and of the number of bands used for spectrum modeling are reported in Figure 4. We note that, when the

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7647
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7647

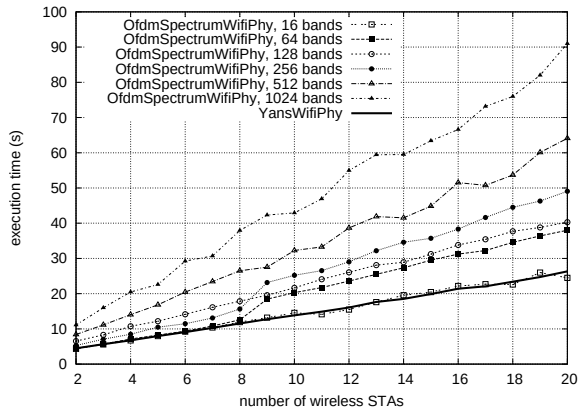

Figure 4: Computational load comparison between a spectrum-aware model (OfdmSpectrumWifiPhy) and a traditional single-frequency model (YansWifiPhy)

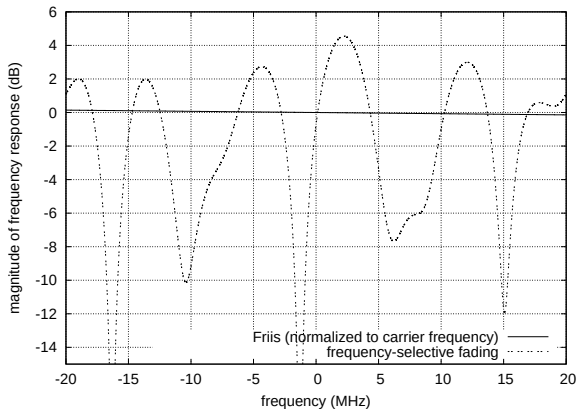

Figure 5: Baseband frequency response of the considered propagation models.

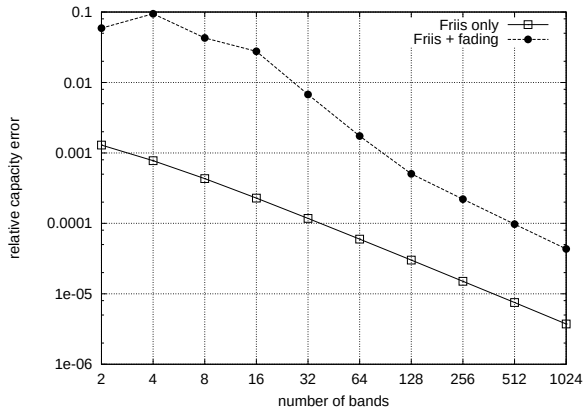

Figure 6: Relative error in the calculated channel capacity as a function of the number of bands used for the spectrum model

number of bands in the spectrum model is limited, the execution time is almost the same as `YansWifiPhy`. When the number of bands increases, the use of `OfdmSpectrumWifiPhy` is computationally more demanding; however, even with such a high number of bands as 1024, the required execution time is roughly only 4 times that of Yans.

In general, the higher is the number of bands used for spectrum modeling, the higher will be the spectrum modeling accuracy. In practice, however, the exact amount of accuracy improvement will depend on how much the channel and physical layer phenomena that we want to model vary with respect to frequency. To better understand this issue, we ran a simple simulation using Matlab.[3] We considered a communication system operating at a carrier frequency of 2.447 GHz, with a bandwidth of 20 MHz. We considered a free-space pathloss model, with a received spectral power density of 2.5×10^{-19} W/Hz at the carrier frequency. The frequency response of this pathloss model was calculated in accordance with the formula by Friis [8], and is reported in Figure 5; as evident from the figure, the frequency response for the pathloss model alone is almost flat. In addition to pathloss, we considered multipath fading; in particular, we used the Rayleigh fading channel model provided by Matlab's Communication Toolbox to simulate a 5 path channel with path delays equal to 167, 233, 303, 320 and 357 ns, an exponential power delay profile with a delay constant of 333 ns, and a mobile node speed of 4 Km/h. The result of this choice of parameters is a slow-fading frequency-selective channel, whose frequency response is also reported in Figure 5. We considered AWGN noise with a spectral power density of $4{\times}10^{-21}$ W/Hz, and we calculated the channel capacity according to (3) for a varying number of bands, i.e., for different levels of spectrum modeling accuracy. This calculation was performed twice, first considering path loss only, and then considering path loss together with fading.

In Figure 6 we report the obtained relative error between the capacity value calculated using the approximate spectrum model and the actual capacity.[4] We observe that the error is in general much lower when path loss only is considered; in particular, even when a single band is used for spectrum modeling, the relative error is below 1%, which we consider an acceptable level of accuracy. This fact is due to the very weak dependency of the path loss gain on the frequency. On the other hand, when frequency-selective fading is considered in addition to path loss, the relative error is generally higher, and it becomes acceptable (below 1%) only when the modeling resolution is ≥ 32 bands. Of course, different values of the channel modeling parameters might lead to quantitatively different results; however, based on our experience, we argue that a resolution in the order of one hundred bands will provide the accuracy required for most practical purposes. Given the modest increase in the computational load that we experienced for this level of resolution, we can conclude that the proposed framework is expected to achieve a good spectrum modeling accuracy in most cases with a sustainable complexity.

[3]The reason for this choice is that a `SpectrumPropagationLossModel` providing a frequency-selective fading model is still to be implemented.

[4]the "actual capacity" was calculated using the approximate spectrum model with a very high resolution, i.e., 2^{18} bands

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7647
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7647

7. CONCLUSIONS

In this paper we presented the framework that we developed to support spectrum-aware simulation in the ns3 simulator, discussing its design, implementation and performance. The code is publicly available [6] under the terms of the GPL license. Future research directions include the development of additional models to be used with this framework, such as, for example, frequency selective fading propagation models, and PHY layer models for wireless technologies like Mobile WiMAX, LTE and Cognitive Radio.

8. ACKNOWLEDGEMENTS

This work was supported in part by the Spanish Ministry of Science and Innovation under grant number TEC2008-06826/TEC (project ARTICO).

9. REFERENCES

[1] WiMAX Forum, "WiMAX System Evaluation Methodology Doucument - Version 1.0," January 2007.

[2] A. Sayenko, O. Alanen, H. Martikainen, V. Tykhomyrov, and O. Puchko, "Winse: Wimax ns-2 extension," in *ICST SIMUTools*, Rome, Italy, March 2009.

[3] N. Baldo, F. Maguolo, and M. Miozzo, "A new approach to simulating PHY, MAC and Routing," in *ACM WNS2*, Athens, Greece, October 2008.

[4] N. Baldo, F. Maguolo, M. Miozzo, M. Rossi, and M. Zorzi, "ns2-MIRACLE: a Modular Framework for Multi-Technology and Cross-Layer Support in Network Simulator 2," in *ACM NSTools*, Nantes, France, October 2007.

[5] DEMOCLES®, Simulation-based testbed for dynamic RRM, http://www.cttc.es/en/projects/testbeds/project/DEMOCLES.jsp.

[6] http://iptechwiki.cttc.es/Ns3_Spectrum.

[7] http://www.nsnam.org/docs/release/manual.html.

[8] J. D. Kraus, *Antennas*. McGraw-Hill, 1950.

[9] D. Wu and R. Negi, "Effective capacity channel model for frequency-selective fading channels," *Wireless Networks*, vol. 13, no. 3, pp. 299–310, 2007.

[10] T. Taher, M. Misurac, J. LoCicero, and D. Ucci, "Microwave Oven Signal Modelling," in *IEEE WCNC*, April 2008.

[11] S. He and M. Torkelson, "Effective SNR estimation in OFDM system simulation," in *IEEE GLOBECOM*, November 1998.

[12] Ericsson, "System-level evaluation of ofdm - further considerations," 3GPP TSG-RAN WG1 #35 R1-031303, Nov 2003.

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7647
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7647