Simulating Wireless and Mobile Networks in OMNeT++ The MiXiM Vision

A. Köpke, M. Swigulski,
K. Wessel, D. Willkomm
Technische Universität Berlin
Berlin, Germany
{koepke, swigulski,
wessel, willkomm}
@tkn.tu-berlin.de

P.T. Klein Haneveld
T.E.V. Parker*
O.W. Visser*
TU Delft
Delft, the Netherlands
{P.T.KleinHaneveld,
T.E.V.Parker,
O.W.Visser}@tudelft.nl

H.S. Lichte, S. Valentin
University of Paderborn
Paderborn, Germany
{hermann.lichte,
stefanv}@upb.de

ABSTRACT

Wireless communication has attracted considerable interest in the research community, and many wireless networks are evaluated using discrete event simulators like OMNeT++. Although OMNeT++ provides a powerful and clear simulation framework, it lacks of direct support and a concise modeling chain for wireless communication. Both is provided by MiXiM. MiXiM joins and extends several existing simulation frameworks developed for wireless and mobile simulations in OMNeT++. It provides detailed models of the wireless channel (fading, etc.), wireless connectivity, mobility models, models for obstacles and many communication protocols especially at the Medium Access Control (MAC) level. Further, it provides a user-friendly graphical representation of wireless and mobile networks in OMNeT++, supporting debugging and defining even complex wireless scenarios. Though still in development, MiXiM already is a powerful tool for performance analysis of wireless networks. Its extensive functionality and clear concept may motivate researches to contribute to this open-source project [4].

1. INTRODUCTION

Discrete event simulators like OMNeT++ [22] are a standard tool to study protocols for wired and wireless networks. In contrast to the wired channel, the wireless channel has a complex influence on protocol performance and it requires in-depth knowledge of the effects before a researcher can determine the right level of detail necessary to make a sound performance analysis. If he needs a very detailed model, an implementation from scratch is a tedious and error prone task, but this implementation might be useful for other researchers as well. We therefore present MiXiM [4], a **mix**ed s**im**ulator combining various simulation frameworks developed for wireless and mobile simulations in OMNeT++. It provides detailed models and protocols, as well as a supporting infrastructure. These can be divided into five groups:

Environment models In a simulation, only relevant parts of the real world should be reflected, such as obstacles that hinder wireless communication.

Connectivity and mobility When nodes move, their influence on other nodes in the network varies. The simulator has to track these changes and provide an adequate graphical representation.

Reception and collision For wireless simulations, movements of objects and nodes have an influence on the reception of a message. The reception handling is responsible for modeling how a transmitted signal changes on its way to the receivers, taking transmissions of other senders into account.

Experiment support The experimentation support is necessary to help the researcher to compare the results with an ideal state, help him to find a suitable template for his implementation and support different evaluation methods.

Protocol library Last but not least, a rich protocol library enables researchers to compare their ideas with already implemented ones.

MiXiM provides these solutions by combining the approaches of several existing simulation frameworks into one: the mobility support, connection management, and general structure is taken from the Mobility Framework (MF) [5]; the radio propagation models are taken from the CHannel SIMulator (ChSim) [21]; and the protocol library is taken from the MAC simulator [3], the Positif framework [6], and from the Mobility Framework.

Using the experience gathered while writing each of these simulators, MiXiM introduces unique extensions like full 3D support, models for walls and obstacles that influence the mobility and the attenuation of radio signals, different frequencies and transmission media (radio waves, ultrasound), full multi-channel support in space and frequency, enabling Orthogonal Frequency Division Multiplexing (OFDM) and Multiple Input Multiple Output (MIMO) simulations, and more MAC protocols, including IEEE 802.15.4.

MiXiM is envisioned to support the simulation of networks with more than 1000 nodes, hence it has a low memory consumption and its modular structure allows the adaptation of the level of detail and thus the execution time. A graphical configuration interface helps to choose the right modules, stack them in layers, and assign values to their parameters.

*supported by the Commission of the European Union (Project No. 013790, FP6 IST Programme, RELATE)

OMNeT++ Workshop '08 Marseille, France

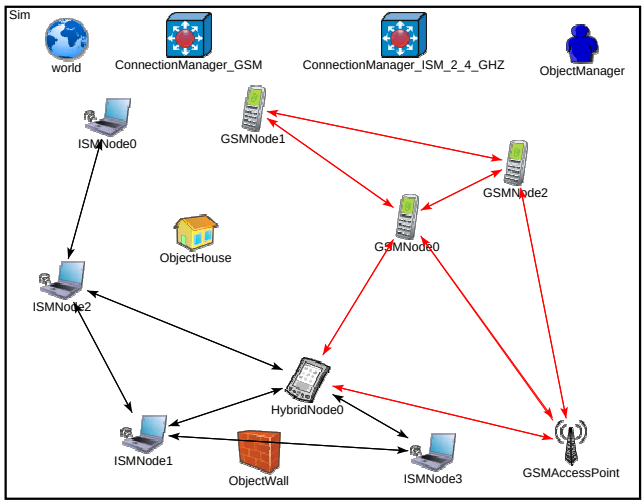
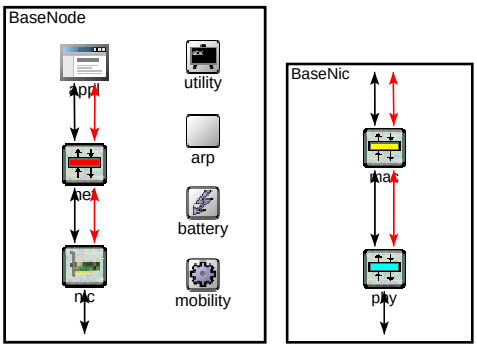


Figure 1: Simulation network



(a) Node structure (b) NIC structure

Figure 2: Node and NIC structure

This paper is structured as follows: After a general overview of the simulator in Section 2, we describe the models present in MiXiM in Section 3. However, these models can not always be implemented directly, and the implementation Section 4 introduces the base abstractions needed to program them. The protocol library uses these abstractions to actually build a number of protocols, which are shortly described in Section 5. After this we conclude our work and give an outlook on the future of MiXiM in Section 6.

2. GENERAL STRUCTURE

In this section we describe the general structure of the MiXiM framework and its constituent components.

2.1 Simulation modules

Figure 1 shows an example MiXiM network. The environmental model is contained in the `world` utility module, which is mainly used to collect global parameters like the dimensions of the network (`playground`), whether it is 2D or 3D, etc. Furthermore, MiXiM uses `objects` to model the environment of a simulation. Figure 1 shows a house (`ObjectHouse`) and wall (`ObjectWall`) as examples. Objects influence the radio propagation of signals and the mobility of other objects and nodes. A wall, for example, cannot be crossed by a pedestrian. The `ObjectManager` is responsible for managing `objects`, providing services to the rest of the simulation including calculating which objects interfere within a given line-of-sight between two nodes. Details on the environmental modeling are described in Section 3.1.

The `ConnectionManager` module is responsible for dynamically managing the connections between interfering nodes. It knows the position of all nodes and can query object positions from the `ObjectManager`. In general, MiXiM supports multiple connection managers, responsible for different frequency ranges such as radio waves and ultra sound. Details on connection modeling are given in Section 3.2. The `ConnectionManager` implementation of MiXiM is detailed in Section 4.2.2.

Finally, a network also contains `nodes`, i.e. entities desiring to communicate with each other. In MiXiM, different kinds of nodes can be specified, such as Access Points (APs) and terminals. Additionally, a node can have different communication capabilities, e.g. bluetooth and ultrasound.

2.2 Node modules

An example of a `node` module can be seen in Figure 2(a). On the left side, the standard layers according to the IP model can be found, namely, the application layer (`appl`), the network layer (`netw`), the MAC layer (`mac`), and the physical layer (`phy`). The physical layer is the place to take care of the reception and collision handling and is described in Section 4.3 in detail. Adjacent layers are connected by two pairs of OMNeT++ "gates". The first pair is used for passing up and down data messages as in the real world, including control messages between nodes. The second pair is used to exchange control messages between the layers, to support control communication. Examples are the request of the MAC layer to the physical layer to perform carrier sensing, or the indication of the physical layer to the MAC layer that the transmission of a message is over.

Note that the physical and MAC layer are grouped into a Network Interface Card (NIC) module (Figure 2(b)). Physical and MAC layer design is usually tightly coupled and very specific for different communication techniques (e.g. bluetooth, GSM). We thus decided also to couple the modeling tightly. Although there is only one NIC shown in Figure 2(a), a node can have several NICs. In MiXiM, thus, a laptop with different NICs, like bluetooth, GSM, and IEEE 802.11 can be modeled.

The `mobility` module is responsible for the movements of a node or an object. Details on mobility modeling and the core implementation in MiXiM can be found in Sections 3.1 and 4.2, respectively. Different mobility models implemented in MiXiM are described in Section 5.3. The `battery module` is used for energy related issues. For a sensor node, e.g., the battery drainage due to communication and processing can be simulated. The `arp` module handles the Address Resolution Protocol (ARP), i.e. the translation between network and MAC addresses.

The `utility` module is derived from the *blackboard* module introduced in the Mobility Framework [10]. It has two main tasks: Firstly, it provides a general interface for collecting statistical data of a simulation. Using the utility module for statistical data collection only has minimal impact on the performance of the simulation and leaves full flexibility for different analysis methods. Secondly, the utility module maintains parameters that need to be accessed by more than one module within a node. One example is the position of a node, which is calculated and updated by the mobility module, but also needed by the physical layer and potentially the localization module. Details on the utility module can be found in Section 4.1.

2.3 Base framework and protocol library

Logically, MiXiM can be divided into two parts: the base framework and the protocol library. The base framework provides the general functionality needed for almost any wireless simulation, such as connection management, mobility, and wireless channel modeling. Details on the implementation concepts of the base MiXiM framework can be found in Section 4. The protocol library complements the base framework with a rich set of standard protocols, including mobility models, and is detailed in Section 5.

In order to have clearly defined interfaces between the base framework and the protocol library, MiXiM provides a *base* module for each OMNeT++ module described above. Following this concept makes it easy to implement new protocols for MiXiM while facilitating re-usability.

3. MIXIM BASE MODELS

Simulating wireless communication systems requires a suitable abstraction of the environment, the radio channels, and the physical layer. For these parts of the scenario, MiXiM provides a *modeling* framework. In this section, we discuss the basic modeling approaches, the assumptions behind these approaches, and implementation relevant aspects such as model abstraction level and model support for trading off accuracy and calculation complexity.

3.1 Environmental model

Simulations are usually carried out on a limited area, a `playground`, on which nodes and objects are placed. Nodes represent the wireless devices with their protocol stack and are modeled as isotropic radiators not having any physical dimension. An object, in contrast, is anything with a physical dimension that resides in the propagation environment and can possibly attenuate a wireless signal. Both, objects and nodes may be mobile. Nodes may even be combined with objects, e.g. to model a sensor node mounted on a car. In the following, we use the term *entity* to refer to both, nodes and objects.

Depending on the scenario to investigate, the limited area of the playground may cause undesired border effects, which can – in the worst case – dominate the simulation results. In order to avoid such effects, the playground may be modeled as the surface of a torus. Wrapping the edges of a rectangular area (i.e. connecting the top edge with the bottom edge as well as the left edge with the right edge) results in a torus, where an entity leaving the playground on one side would reappear on the other side. The same applies for wireless connectivity between nodes.

The mobility of objects is a time-continuous process, which raises a trade-off between accuracy and computational complexity. In MiXiM, the level of accuracy (and, thus, the computational complexity) of modeling mobility can be chosen by the user. MiXiM provides a user-defined `update interval` for mobility modeling. This parameter specifies how often the position of an object is updated. Additionally, the position information of an entity contains the start time, start position, direction, and speed of the entity. Thus, intermediate positions can be easily interpolated if needed.

Mobility also requires to handle collisions with entities and to handle border crossing of the playground. MiXiM provides four different strategies for handling collisions and border crossing. First, a collision may simply raise an error. Second, a new position may be randomly chosen upon collision and the entity is placed there afterwards, as long as this position does not coincide with another entity. Third, the entity may be reflected in an angle that it had when it collided. The last strategy only applies to the borders of the playground: When modeled as a torus, wrapping causes the entity to re-enter the playground on the opposite side.

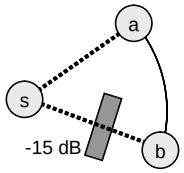


Figure 3: An object within the line-of-sight between two nodes *s* and *b* yields a weaker received signal than that of a non-obstructed pair *s* and *a* at the same distance.

The shadowing effect of objects that reside in the propagation environment results in different received signal strengths at equal distances as shown in Figure 3. These variations can be described by a stochastic model (e.g. log-normal shadowing), which must be adapted to the characteristics of the environment that is to be simulated. Unfortunately, such a model is too generic and cannot adequately describe particular settings, e.g. nodes placed around a building.

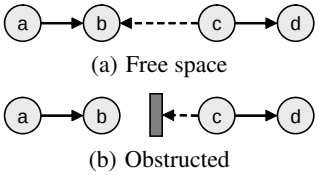
As a solution, MiXiM provides the `ObjectManager` as a central authority for managing objects in the propagation environment. Objects are characterized by dimensions, position, angle of rotation (optionally), and frequency-dependent attenuation factors. An object that obstructs the line-of-sight between any pair of interconnected nodes causes additional signal losses during transmission as shown in Figure 3. Since entities can be mobile, intersections of the line-of-sight of two nodes with one or more objects must be determined at runtime. For any intersection with an object, its frequency-dependent attenuation factor is added up to yield the additional attenuation caused by the objects.

3.2 Connection modeling

In contrast to wired simulations, connectivity modeling is a challenging task in wireless simulations. In wired simulations, two nodes are connected by wires, which can be easily modeled (e.g. in OMNeT++ by connections). In wireless simulations, however, the "channel" between two nodes is the air, which is a broadcast medium and cannot be easily represented by **one** connection. In MiXiM we decided to divide the modeling into two parts. The first part is the wireless channel and its attenuation property, which is described in detail in Section 3.3. The second part is the connectivity between nodes, which is described in the following.

Theoretically, a signal sent out by one node affects all other nodes in the simulation (if operating in the same frequency range). However, the signal is attenuated, so that the received power at nodes very far away from the sending node may be so low that it is negligible. In order to reduce the computational complexity in MiXiM, nodes are connected only when they are within the `maximal interference distance`. The maximal interference distance is a conservative bound on the maximal distance at which a node can still possibly disturb the communication of a neighbor. Please note that the maximal interference distance does not specify the maximal distance at which messages can be (correctly) received. A MiXiM connection is probably better defined by its complement: *All nodes that are **not** connected, definitely do not interfere with each other*. Following this concept, a node that wants to receive a message from a communication peer, also receives all (interfering) signals and can, thus, decide on the interference level and resulting bit errors.

The presence of objects in the propagation environment also impacts the maximal interference distance. Objects may shield two nodes from each other as shown in Figure 4 because the additional



(a) Free space

(b) Obstructed

Figure 4: The presence of an object within the line-of-sight between two pairs of connected nodes, a,b and c,d, decreases the maximal interference distance of node c.

attenuation that they imply reduces the maximal interference distance. Thus, objects may cause two nodes to be disconnected, increasing the probability of the hidden node problem.

3.3 Wireless channel models

MiXiM's channel models express radio propagation effects as time variant factors of the instantaneous Signal-to-Noise Ratio (SNR) γ of the received signal[1]. Although such SNR-based models abstract the exact signal behavior, e.g. the current phase shift, they enable the separate calculation of channel effects and, thus, adjusting the required accuracy by selecting the modeled effects and time-scale. On this SNR-level, MiXiM already includes the following widely accepted channel models for path-loss, shadowing, large and small-scale fading [8, 20].

Small-scale fading, i.e. determining the variation of a wireless channel at small time scale, is caused by mobility in the propagation environment. We model small-scale fading using the typical "Jakes-like" method [8] with the "land mobile" Autocorrelation Function (ACF) (Table 2.1 in [20]). This model is parameterized by a maximum Doppler shift according to carrier frequency f_c and velocity v of the fastest moving object in the propagation environment, e.g. a moving user. This fading model is based on an Non-Line Of Sight (NLOS) assumption modeled by Rayleigh-distributed signal amplitudes resulting in an exponentially distributed instantaneous SNR $\gamma_{i,j}$ for the channel from user i to user j. The model supports frequency-selective fading (as occurring in wide band systems, e.g. WLANs), which is parameterized by the mean delay spread and can be easily extended to further dimensions of the signal, e.g. spatial fading for multi antenna systems. While the model as such supports reciprocal channels and correlation between multiple signal dimensions, in typical scenarios, the instantaneous SNR is assumed to be independent and identically distributed (i.i.d.) for different channels (i, j). This means, first, that all users are sufficiently separated in space and, second, that the channels are non-reciprocal, i.e. $\gamma_{i,j} \neq \gamma_{j,i}$.

The resulting i.i.d. autocorrelated Rayleigh fading model reflects a typical channel scenario found in office or urban environments with many small stationary and uniformly distributed scatterers and moving users. With moving users, naturally, the distance to the destination d may change over time. Therefore, path loss, i.e. modeling the attenuation of signals during propagation, becomes time selective – an effect also known as large-scale fading. Here, standard methods are employed incorporating path loss as an environment-dependent negative exponent $(-\alpha)$ of the time-selective distance $d(t)$ between the communicating terminals.

Finally, the effect shadowing abstracts many physical effects such as reflection, diffraction, scattering, and absorption. Typically, shadowing is modeled by i.i.d. log-normal attenuation reflecting urban environments.

To implement shadowing and fading a block model has to be used requiring that $\gamma_{i,j}$ stays constant during a so-called block time. This interval typically refers to a Physical layer (PHY) interleaver block or to the minimal coherence time of the channel. Therewith, each of these blocks experiences a quasi-static channel while the ACF defines whether consecutive blocks fade independently.

Although the above models are widely-used, they are of course not suitable for *any* situation. For example, a Nakagami-type amplitude distribution may be preferred for scenarios with a LOS component, or a different ACF may be chosen to model an open-space scattering environment. Nonetheless, due to their clear separation, in MiXiM, all these components can be exchanged easily and can be used to derive further model variants. A standalone implementation of these models for OMNeT++ is available in [21].

[1]Unless noted otherwise, we employ *linear* SNR values.

3.4 Physical layer models

At the physical layer, essentially the used modulation and Forward Error Correction (FEC) coding and decoding functions define the bit error rate and throughput of a system. As for the effects of wireless channels, the effect of these functions can be modeled at SNR-level.

At this level, FEC introduces a so-called coding gain at the receiver, which can be expressed by a factor g to the SNR of the detected signal. This coding gain depends on the used code, its rate R_c, and the employed decoding algorithm [19]. While an uncoded transmission is expressed by $g = 1$, typical channel codes provide coding gains larger than 2 (Table 8.2-15 in [19]). For a single channel observed at the receiver, this simply results in $\hat{\gamma} = \gamma \cdot g$ for the SNR after decoding. This SNR value is then compared to an SNR threshold (th_γ) to model transmission errors in the decider (Section 4.3.4) and a transmission error is assumed at the receiver if $\hat{\gamma} < \text{th}_\gamma$. Typically, the SNR threshold th_γ calibrates the system to stay below a given Packet Error Rate (PER) bound, e.g. as defined in the standard of the communication system [15]. It is selected a-priori depending on the receiver sensitivity for the chosen modulation scheme as given in the transceiver data sheet or approximated [19]. By selecting thresholds and coding gain independently per terminal, terminals employing different PHY parameters can be modeled. Furthermore, by varying thresholds and coding gain over time, rate adaptation is supported.

In addition to systems detecting a bit from a single channel, this SNR-based model easily extends to diversity receivers where several channels are joined before the bit detection is made. Such systems exploit differently faded channels and employ a filter, e.g. Maximum Ratio Combining (MRC) [19], to combine the signals received from L channels to a single signal used for bit detection. On SNR-level this signal combining can be modeled as $\hat{\gamma} = \sum_{l=1}^{L} \gamma_l \cdot g_l$ defining the instantaneous SNR reached after combining and decoding. For each employed channel l, a different code or modulation may be chosen by defining an independent coding gain g_l or threshold $\text{th}_{\gamma,l}$. This combining model can be used to model diversity receivers combining signals in different dimensions, e.g. OFDM subcarriers, or multi-antenna systems. Using this diversity receiver model for simulating cooperative multi-antenna relaying networks is described in [14] in more detail.

4. MIXIM BASE IMPLEMENTATION

In this section we introduce the base implementation concepts of the MiXiM framework. Specifically, we give details on the `utility` module concept, the mobility and connection management, the channel implementation, and some basics about the physical layer implementation. Furthermore, a timer abstraction is introduced as an optional way to use timers in a simulation.

4.1 Utility module

The purpose of simulation is to gain insight in the function and performance of a protocol or system. To this end, the experimenter needs to collect data, like estimated node locations, throughput or delay. When the protocol is finished, another researcher may want to use the code and compare it with other protocols under different circumstances. Often he needs to collect similar data as the first researcher. It is hence desirable that the performance instrumentation remains in the protocol code, but it should have a negligible impact if it is not needed. Furthermore, the instrumentation code should be independent from the analysis tool used to gain insight. One researcher may prefer to write trace files for offline analysis; another researcher may use Akaroa [11] to simulate until a certain confidence level is reached; and yet another researcher may prefer to calculate all results online.

The blackboard contained within the `utility` module provides a general solution for this problem. The instrumented module ("publisher") publishes the observed parameter on the blackboard. The meaning of a parameter is encoded in its class, as standard for object oriented languages. The blackboard then informs all parties that are interested in this particular parameter ("subscribers"). Often, the publishing module is the instrumented protocol, while the subscriber is usually a module that performs statistical analysis. This allows each researcher to plug in his preferred statistical analysis method. At the same time, the overhead for the instrumentation (publishing) is a simple function call that has a constant execution time for each parameter. Of course, the execution time increase if more subscribers are interested in this particular parameter.

The same concept is used for parameters, which need to be accessed or updated from multiple modules within a node. The publisher can publish the parameter as described above. Every subscriber will get a notification about the change and can take appropriate action. Details on the publish-subscribe interface used can be found in [10] and the Mobility Framework manual [5]. The `utility` module can also keep local copies of parameters published on the blackboard. This way, the `utility` module complements the push interaction of the blackboard with a pull interaction for appropriate parameters.

4.2 Mobility and connectivity

Mobility and connectivity management is one of the main tasks of the MiXiM base framework. It is crucial for a simulation to be able to provide a given level of accuracy and at the same time keeping the computational complexity at a reasonable level.

We decided to handle mobility in a distributed manner. It is handled locally by a `mobility` module in every entity. Decisions how and where to move neither affect other entities nor do they require global knowledge. Connectivity management is handled centrally by the `ConnectionManager`. In order to set up and tear down connections, the distances between nodes have to be calculated, for which the global knowledge of the positions of all nodes is required.

4.2.1 Mobility module

Each entity has a `mobility` sub-module, responsible for the movements of the node or object. As already mentioned, the accuracy (and with this the computational complexity) of the mobility can be adjusted using the `update interval` parameter.

The `BaseMobility` module is responsible for the graphical representation of an entity. Every time the position is changed, it updates the graphical representation. It also publishes the new position on the `utility` module of the node, so that other modules get informed. One of the modules that needs to get informed about changes in the position of a node is the physical layer which is responsible for updating the position with the `ConnectionManager`. Another functionality implemented in the `BaseMobility` module is the border handling as described in Section 3.1.

The only task left to be implemented for specific mobility models is the actual movement pattern of the nodes. Some of the models available in MiXiM are described in Section 5.3.

4.2.2 Connection management

The `ConnectionManager` module is responsible for establishing connections between nodes that are within the maximal interference distance of each other and tearing down these connections once they exceed this distance. The loss of connectivity can be due to mobility (i.e. the nodes move too far apart) or due to a change in transmission power or a crashed node etc.

An important factor influencing the maximal interference distance is the attenuation caused by objects within the line-of-sight of two nodes as shown in Figure 4. As already introduced in Section 3.1, each object has a frequency dependent attenuation factor. All objects have to register with the `ObjectManager` (the central authority for managing objects in the simulation). The object manager implements a *line segment intersection* algorithm that checks whether a line connecting two points in the propagation environment intersects with the borders of one or more objects. The additional attenuation is then determined by adding up the frequency-dependent attenuation factors of the objects intersecting the line. The connection manager uses this value for adjusting the maximal interference distance of two nodes and, thus, the presence of objects may cause two nodes to be disconnected. Since MiXiM supports multiple frequency ranges by means of multiple connection managers, objects can be defined with attenuation factors per connection manager.

MiXiM provides two possibilities how to establish communication channels between nodes. The first version – mainly for visualization and debugging – uses OMNeT++ connections. While this approach is not very memory efficient (6 gates are needed for every connection), it enables the user to visualize the network for debugging reasons. The other option is to use the `sendDirect` function of OMNeT++, only requiring one gate per NIC. This option is especially useful for running the simulation in command line mode without visualization. To further speed up connection updating, the connection manager implements a grid based approach. The playground is divided into quadrants with the edge length set to the maximal interference distance. When a node changes its position, the connection manager only has to update the connections of this node to all other nodes within its own and neighboring quadrants. Depending on the size and density of the network, this can result in a significant performance improvement. For details please refer to [10].

As previously mentioned, it is possible to have multiple connection managers in MiXiM. This enables the simulation of different orthogonal spectrum ranges while reducing the memory consumption and computational complexity. In Figure 1, e.g., a GSM network and a WLAN network are presented. The assumption is that the two spectral ranges are sufficiently separated, so that we can assume that GSM communication does not effect WLAN communication going on at the same time. As can be seen in Figure 1, it is also possible to have hybrid nodes, i.e. nodes having both a GSM and a WLAN NIC. It is the responsibility of the NIC (and **not** the node) to register with the desired connection manager. Only registered NICs will be connected by the respective connection managers.

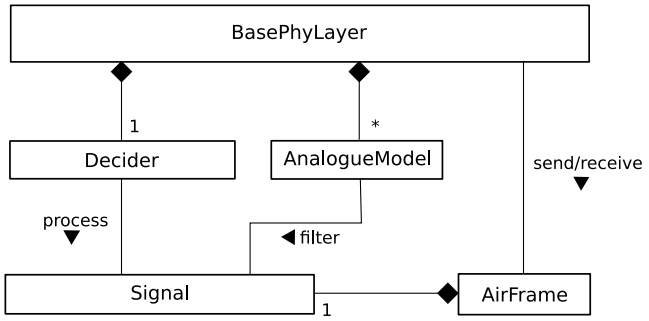


Figure 5: Physical layer class graph

4.3 Physical layer

The physical layer is the core part of a wireless node in MiXiM. It is responsible for message sending and receiving, collision detection, and bit error calculation. Additionally, it is responsible for applying the channel models used in the simulation.

The MiXiM physical layer is divided into three parts, which are described in detail in the following sub-sections. The `BasePhyLayer` itself provides the interfaces to the MAC layer and the physical layers of other nodes. The `AnalogueModels` are responsible for simulating the attenuation (like shadowing, fading and path loss) of a received signal. Finally, the `Decider` is responsible for evaluation (classification as noise or signal) and demodulation (bit error calculation) of the received messages. To provide a clear interface and to avoid memory overhead we have designed the analogue models and the decider as pure C++ classes instead of separate OMNeT++ modules. The class graph of a physical layer of MiXiM is shown in Figure 5.

4.3.1 The signal concept

The signal strength of a message sent from one node to another is influenced by the environment it travels through. As shown in Section 3.3, this can be modeled with attenuation factors caused by path loss, shadowing and fading. Furthermore, a message can be sent using multiple frequencies (e.g. OFDM) and using multiple antennas (MIMO). Adding up all those possibilities, a message can have varying sending power, attenuation, and bit-rate (modeling modulation and coding) in time, space and frequency.

In MiXiM we decided to create the `signal` class to model this complex process. Each message has an attached signal object representing sending power, attenuation, and bit-rate in the three dimensions time, frequency, and space. An example for the sending power (TX) is shown in Figure 6. In order to send a message, a node has to specify the sending power and bit-rate in the appropriate dimensions. The receiving node then adds the attenuation. Based on the whole signal, bit errors can be calculated.

4.3.2 BasePhyLayer

Apart from message sending and receiving, the `BasePhyLayer` acts as an interface between physical layer messages (`AirFrames`) and the `AnalogueModel` and the `Decider`. For maximal modularity and flexibility, different analogue models and deciders can be plugged into the physical layer.

When receiving a message, the physical layer first passes the message to the analogue model, which calculates the attenuation part of the signal. The physical layer is then responsible for simulating the propagation and transmission delay of the message. The message is passed at least twice to the decider: at the beginning and at the end of the message. However, the decider can also request

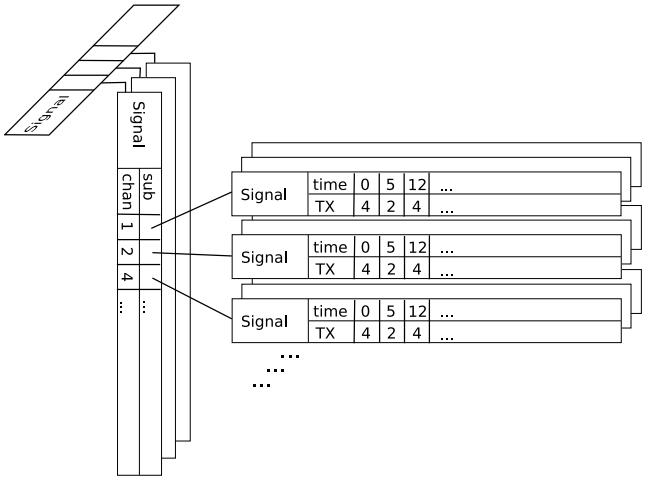


Figure 6: Example signal for sending power (TX)

to get the message at arbitrary times in-between. Finally, after the decider calculates the bit errors, the message has to be handed to the MAC layer.

Additionally, the physical layer stores all messages in the `ChannelInfo` class. The `ChannelInfo` class is a service provider that keeps track of all `AirFrames` on the channel. `ChannelInfo` provides a function that returns all `AirFrames` intersecting with a given time interval. The decider uses this function in order to calculate the SNR of a given message. `AirFrames` are disposed by `ChannelInfo` once all other time-intersecting `AirFrames` are completely received.

4.3.3 Analogue models

The real receiving power of a received message is a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ from time, frequency and space to receiving power. Since we do not have attenuation in OMNeT++, MiXiM has to simulate features like path loss, shadowing and fading.

Each of these attenuation sources can be represented by another function $a_i : \mathbb{R}^n \rightarrow \mathbb{R}$ from time, frequency and space to attenuation. The attenuation of a signal is calculated by implementations of fading, shadowing and path-loss models. Note that an arbitrary number of analogue models can be plugged into the physical layer. Each analogue model is basically a filter class for signals.

Summing up the attenuation of all analogue models gives the attenuation part of the signal, which is calculated at the start of the reception of a message. Together with the sending power of a received packet the decider can later on calculate the SNR and, thus, bit errors.

4.3.4 Decider

There are three main tasks to the `Decider`. First of all, the decider has to classify incoming messages into receivable messages or noise. Second, at the end of receiving a receivable message, the decider has to calculate the bit errors for the message. Third, it has to provide information about the current state of the channel.

There are several possible models determining how and when a physical layer decides whether a message can be received or is just noise. The MiXiM decider supports all of these models. Upon the start of the reception of a message, the physical layer passes the message to the decider. The decider can then either decide right away whether to treat the message as noise or not, and / or it can request the physical layer to resubmit the packet after a certain time.

This concept even enables the decider to revise its decision (e.g. if a second, much stronger message arrived in the meantime).

The latest time that the decider can request the message from the physical layer is at the end of the receiving process of the message. This is also the time when the decider has to calculate the bit errors for the message. In order to do so, it requests all intersecting messages from the `ChannelInfo` in order to calculate the SNR for the message. It then can either make a simple binary decision (received correctly or not) or it can calculate bit errors and positions, depending on the complexity of the particular decider model.

The last task of the decider is to provide information about the channel state. This channel state is needed at the MAC layer, e.g. for Carrier Sense Multiple Access (CSMA) protocols. The MAC layer can request the decider to sense the channel for a certain amount of time. The decider then returns whether the channel is currently idle or busy.

4.4 Timer

MiXiM provides a number of timer modules for use by protocols, allowing for simple use of timers without having to know about the underlying systems, and providing common functionality for creation and destruction of timers with minimal user intervention. Currently, MiXiM provides three types of timers: (1) Simple one-shot `Timer` that fire once *n* seconds after they are started. (2) `RepeatTimer`, i.e. timers that fire every *n* seconds. (3) `FrameTimer`, which are like `RepeatTimer`, but all of the separate timers on different nodes are guaranteed to fire at about the same time (typically within a few milliseconds of each other).

In addition, OMNeT++ objects can be "attached" to timers to allow for storing arbitrary timer-related information. These are support modules, which can be included by any MiXiM modules that require their functionality, but do not slow down modules that do not require their functions.

5. MIXIM PROTOCOL LIBRARY

MiXiM allows every module in the simulation to be replaced by another module, adding or overriding functionality to the base implementation. For some of these modules there is already a wide choice of implemented protocols available.

5.1 MAC protocols

A Medium Access Control (MAC) protocol is designed to make decisions about the sharing of a medium for communication between nodes of a system. For wireless systems, that shared medium is the air. A MAC protocol needs to decide when a node should send out messages, such that the messages do not interfere with messages of other nodes. Additionally – especially for low power devices – the MAC protocol should determine at what times the radio can be switched off to avoid listening to the medium (which consumes power) when no other nodes are sending.

Support for such MAC protocols, especially designed for low power sensor networks, is based upon the MAC Simulator [3] developed by the TU Delft, originally for work with the T-MAC protocol [9]. Inherited from the MAC Simulator, MiXiM provides a wide variety of different MAC protocols encompassing a significant proportion of the current design space for sensor network MACs. There are two base classes for building sensor network specific MAC protocols:

BaseMACLayer - basic MiXiM-style layering interface, providing en/decapsulation of packets, but no other functionality.

EyesMACLayer - provides a number of support functions for sensor network MACs, including support functions for low-power listening [18] and Sift-inspired [12] carrier-sense period choosing, as well as generating statistical information about MAC protocol performance.

The MAC Simulator was unable to simulate motion of nodes, or a detailed radio model, but now these functionalities are provided by MiXiM and can be used to further explore the effect on MAC protocols. We further intend to implement our more recent work with the λMAC framework [16] for MiXiM, which will also help with easing MAC protocol implementation.

MiXiM also already implements standard MAC protocols for wireless Local Area Networks (LANs) and Personal Area Networks (PANs). The IEEE 802.11b/g family, as well as the IEEE 802.15.4 standard are ported from the Mobility Framework to MiXiM.

The base structure of MiXiM also makes it very easy to implement new MAC protocols. For instance, the "FrameTimer" support modules can be used to easily implement Time Division Multiple Access (TDMA) based or hybrid protocols (e.g. T-MAC).

5.2 Network layer protocols

MiXiM supports networking protocols for a wide variety of traffic paradigms (source-to-sink; any-to-any; local neighborhood; etc), and these are further supported by the other simulation modules, e.g. localization data (Section 5.4) for geographic routing, motion data derived from the mobility module (Section 4.2.1) for decisions regarding when to update routes, or network-wide timers (Section 4.4) for synchronized protocols. MiXiM is currently being used to test the Ley Line Routing protocol [7].

5.3 Mobility models

There is already a rich library of mobility modules implemented for MiXiM, which includes simple modules like "constant speed mobility" and "circle mobility", but also modules that parse ANSim [1] trace files and BonnMotion [2] files. It is also very easy to create new mobility modules, by sub-classing from the `BaseMobility` class. The `BaseMobility` class provides all the functionality needed for mobility handling in MiXiM – only the specific mobility pattern has to be implemented in order to create a new mobility module.

5.4 Localization

Localization (i.e. determining position information of the current node) is a service provided by the optional localization layer. *Optional* means that it is not included in the standard software protocol stack, but must be added to the `node` module explicitly. Thus, simulations that do not use localization do not have a negative performance impact from the overhead of a localization protocol. This optional nature allows the placement of the localization layer beneath the network layer, such that geographic routing algorithms can use the localization layer for position information.

Support for existing localization algorithms is ported from the Positif [6] localization framework developed at the TU Delft. Positif targets static wireless networks and contains the algorithms discussed in [13] as well as a statistics based algorithm [17]. Positif gathers and outputs statistics on the performance of the algorithms and comes with a range of analysis tools to visualize the output. The advantages of running Positif algorithms on MiXiM are: they can be tested with different NICs and the influence of objects on the algorithms can be analyzed. Furthermore, the experiments are more scalable since MiXiM uses the more scalable `handleMessage` compared to `activity` used in Positif. As Positif algorithms require a specific interface, a new base layer called `PositifLayer` was developed specifically for Positif algorithms. Consequently, there are two base layer to build localization algorithms on:

BaseLocalization - the base layer following MiXiM conventions. It simply forwards messages from upper and lower layers and maintains a list of anchors and unknown neighbors. Algorithms based on `BaseLocalization` can take advantage of the full potential of MiXiM.

PositifLayer - an interface layer between the Positif algorithms and the MiXiM simulation framework. Algorithms based on `PositifLayer` can use the Positif analysis tools, but support only static networks.

The combination of localization with the mobile abilities of MiXiM enables a wide range of simulation scenarios, such as initializing a static network with a mobile anchor, or motion detection and object tracking with either static or mobile anchors; all in combination with the range of other protocols that MiXiM offers.

The `BaseLocalization` layer aims to provide basic functionality for a wide class of localization algorithms. The localization layer includes position information in the header of messages from upper layers, such that algorithms can be developed that have little or no message overhead. Several algorithms implemented on `BaseLocalization` are available.

6. CONCLUSION

In this paper we introduce MiXiM to the OMNeT++ community – a powerful simulation framework and concise modeling chain for mobile and wireless networks.

Although MiXiM is still in development, it already provides a solid base of models and implementations for simulating wireless and mobile networks, including models for mobile environments, nodes and objects, radio propagation models for multiple signal dimensions, physical layer models for modulation, coding and diversity receivers as well as an extensive library of MAC protocols and localization algorithms. Furthermore, most of the work done in the predecessors of MiXiM is being integrated. Thus, MiXiM profits from the rich experience gathered from writing wireless simulation frameworks for OMNeT++. MiXiM's modular design and pioneering concepts such as the signal functionality enable the straightforward definition of complex scenarios as well as the easy integration of new models and protocol implementations.

It is, therefore, our hope that MiXiM's clear structure and its extensive modeling base motivates researchers to contribute to this open-source project hosted at `http:\\mixim.sf.net`. Joining forces in the MiXiM simulator will quickly provide a powerful simulation framework and performance analysis tool to the wireless R&D community.

7. REFERENCES

[1] ANSim - Ad-Hoc network simulation. [online]. Available: http://www.ansim.info/.

[2] BonnMotion. [online]. Available: www.cs.uni-bonn.de/IV/BonnMotion/.

[3] MAC simulator. [online]. Available: http://www.consensus.tudelft.nl/software.html.

[4] MiXiM simulator for wireless and mobile networks using OMNeT++. [online]. Available: http://mixim.sourceforge.net/.

[5] Mobility framework (MF) for simulating wireless and mobile networks using OMNeT++. [online]. Available: http://mobility-fw.sourceforge.net/.

[6] Positif localization simulation framework. [online]. Available: http://www.consensus.tudelft.nl/software.html.

[7] M. Ali, T. Parker, A. Dunkels, K. Langendoen, and P. Levis. L2R: Routing in Low Power and Lossy Networks with Constant State. [online]. Available: http://rl2n.com/.

[8] J. Cavers. *Mobile Channel Characteristics*. Kluwer Academic, 2000.

[9] T. van Dam and K. Langendoen. An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *1st ACM Conf. on Embedded Networked Sensor Systems (SenSys 2003)*, pages 171–180, Los Angeles, CA, USA, Nov. 2003.

[10] W. Drytkiewicz, S. Sroka, V. Handziski, A. Koepke, and H. Karl. A Mobility Framework for OMNeT++. 3rd International OMNeT++ Workshop, at Budapest University of Technology and Economics, Department of Telecommunications Budapest, Hungary, Jan. 2003.

[11] G. C. Ewing, K. Pawlikowski, and D. McNickle. Akaroa2: Exploiting Network Computing by Distributing Stochastic Simulation. In *Proc. European Simulation Multicoference ESM'99*, pages 175–181. International Society for Computer Simulation, June 1999.

[12] K. Jamieson, H. Balakrishnan, and Y. Tay. Sift: A MAC Protocol for Event-Driven Wireless Sensor Networks. In *3rd European Workshop on Wireless Sensor Networks (EWSN'06)*, pages 260–275, Zurich, Switzerland, Feb. 2006.

[13] K. Langendoen and N. Reijers. Distributed Localization in Wireless Sensor Networks: A Quantitative Comparison. *Computer Networks*, 43(4):500–518, 2003.

[14] H. S. Lichte and S. Valentin. Implementing MAC protocols for cooperative relaying: A compiler-assisted approach. In *Proc. Int. Conf. on Simulation Tools and Techniques for Commun., Networks and Systems (SIMUTools)*, 2008. submitted for review.

[15] B. O'Hara and A. Petrick. *IEEE 802.11 Handbook: A designers companion*. IEEE Press, 1999.

[16] T. Parker, M. Bezemer, and K. Langendoen. The λMAC framework: redefining MAC protocols. PDS Technical Report PDS-2007-004, Delft University of Technology, Sept. 2007.

[17] T. Parker and K. Langendoen. Refined Statistic-based Localisation for Ad-Hoc Sensor Networks. In *IEEE Workshop on Wireless Ad Hoc and Sensor Networks (associated with Globecom 2004)*, Dallas, TX, Nov. 2004.

[18] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *2nd ACM Conf. on Embedded Networked Sensor Systems (SenSys 2004)*, pages 95–107, Baltimore, MD, USA, 2004.

[19] J. G. Proakis. *Digital Communications*. McGraw-Hill, 4 edition, 2000.

[20] M. K. Simon and M.-S. Alouini. *Digital Communications over Fading Channels*. John Wiley & Sons, Inc., 2 edition, 2004.

[21] S. Valentin. ChSim – a wireless channel simulator for OMNeT++. TKN Simulation Workshop 2006, Technical University of Berlin, Germany, Sept. 2006. Available at project website: http://wwwcs.upb.de/cs/chsim.

[22] A. Varga. *OMNeT++ Discrete Event Simulation System*. Available: http://www.omnetpp.org/doc/manual/usman.html.