Evaluation of IMT-Advanced Scenarios Using the Open Wireless Network Simulator

Sebastian Max Daniel Bültmann Ralf Jennen
Communication Networks (ComNets) Research Group
Faculty 6, RWTH Aachen University
Aachen, Germany
{smx|dbn|jen}@comnets.rwth-aachen.de

Marc Schinnenburg
PSI Transcom GmbH
Telecommunications
Düsseldorf, Germany
MSchinnenburg@psi.de

ABSTRACT

With the commence of the IMT-Advanced (IMT-A) submission and evaluation process, the vague term "4th generation wireless networks" moves towards existing standards, technologies and hardware. A significant part of the evaluation process is based on the system level simulation of reference scenarios.

In this paper, we present how the open Wireless Network Simulator (openWNS), developed in the last 5 years at the department of Communication Networks (ComNets) at RWTH Aachen University, can be used to study protocols of wireless networks. Due to the complexity of IMT-A candidate systems, the features of openWNS are described by means of the lightweight WiFiMAC. This module provides the functions of IEEE 802.11, including amendment n.

On this basis, the following details of the simulator are explained: (a) the modular simulation framework for protocol stack development, (b) the WiFiMAC module, (c) simulator calibration to ensure the reliability of the results and (d) the simulation of urban-micro scenarios based on the IMT-A evaluation guidelines.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: [Wireless Communication]

General Terms

Algorithms, Performance, Design, Experimentation

Keywords

Wireless Networks, IMT-Advanced, Simulation

1. INTRODUCTION

At the time of writing this paper, the radiocommunication sector of the International Telecommunication Union (ITU) has started its certification process of radio access technologies as IMT-Advanced (IMT-A). Among the key requirements for certification are performance capabilities that significantly exceed those of the current 3rd generation wireless networks. Evaluation of the performance is not done by the ITU-R itself, but by evaluation groups out of industry and academia - for example, ComNets is a member of the WINNER+ evaluation group [25].

Evaluation of the candidates is guided by the ITU-R report M.2135 [15] which describes in detail the test environments and deployment scenarios. Several performance measures have to be evaluated by system simulation; hence, the demand for an IMT-A compliant wireless network simulator is obvious.

The open Wireless Network Simulator (openWNS) is currently on its way to become IMT-A compliant to fill this gap. Furthermore, it shall be used to research complements and extensions which are not part of the candidate specifications, e. g. other radio access technologies, cross-layer optimisation or deployment concepts.

The paper at hand describes how openWNS is designed for the performance evaluation of wireless communication networks. After shortly reviewing other existing simulators, Section 2 introduces the simulation platform. Then, Section 3 explains the openWNS framework for protocol stack development, referred to as Functional Unit Networks (FUNs) [22, 23]. To exemplify this framework, Section 4 details the WiFiMAC module which implements the IEEE 802.11n-2009 protocol. This module is chosen because 802.11 is both simple enough to be used as an example and powerful enough to show the concepts from Section 3. Afterwards, the WiFiMAC's validation is discussed in Section 5. Finally, Section 6 explains the built-in support for the generation, execution and evaluation of large simulation campaigns.

1.1 Related Work

System simulators for the performance evaluation of wireless communication systems are available for more than two decades. In this section, a current view on simulation tools with special respect to wireless network simulation is given.

According to [16], the most prominent among them are, in descending order, ns-2 [3] and its successor ns-3 [4], GloMoSim [26, 2], QualNet [21] and OPNET [6]. Additionally, a big number of other (open-source) simulators are available, some of them developed for special purposes, e. g. in the scope of a research project.

SIMUTools 2010 March 15-19, Torrelmolinos, Malaga, Spain

1.1.1 ns-2 and ns-3

The development of ns-2 started as early as 1989 as REAL simulator. The first release of ns-2 was available in 1996. One drawback of ns-2 turned out to be the lack of wireless transmission modelling and detailed channel models. In 2006, after 10 years of research and development the team around ns-2 decided to start with a complete rewrite of the simulator. In comparison to ns-2, ns-3 aims at providing better support for modularity of components, scalability of wireless simulations, integration and reuse of external code, tracing and statistics.

As of the 21th October 2009, the current stable release is ns-3.6. Interestingly, many similarities can be found between the ns-3 and the openWNS: The programming languages (C++ and Python), a high degree of modularisation using Object-Oriented Programming (OOP) and the focus on wireless networks.

Currently, ns-3 supports as wireless link layer the IEEE 802.11 protocol, including an implementation of the amendments b, e and parts of n and s. A model to simulate WiMAX networks is currently planned. No indication is given respecting enhanced physical layer technologies like beamforming and Multiple Input Multiple Output (MIMO).

The ns-3 is released as open-source under the GNU General Public License (GPL) (version two); hence, the source code of any derived and distributed extension or modification must be made public.

1.1.2 GloMoSim

The Global Mobile Information Systems Simulation Library (GloMoSim) is built using the discrete-event simulation language parsec, which supports parallelisation of the simulation to handle large scenarios. This enables the simulation of very large scenarios with more than 1000 nodes in reasonable time by using multi-processor systems.

The GloMoSim protocol suite focuses on wireless networks; however, only the basic IEEE 802.11 MAC implementation is available as link layer together with a Bit Error Rate (BER) model that supports Modulation and Coding Schemes (MCSs) with data rates up to $18\,{}^{\mathrm{Mb}}/_{\mathrm{s}}$.

The license of the GloMoSim restricts its usage to education and non-commercial research; the reason for this is that GloMoSim's successor, QualNet, is a commercial product. Further development and support of GloMoSim seems to be ceased.

1.1.3 QualNet

Whereas ns-2, ns-3 and GloMoSim are freely available, QualNet is a closed-source commercial tool distributed by Scalable Network Technologies (SNT).

Major customers are from the telecommunication industry which use the simulator to predict the performance of wireless, wired and mixed-platform networks. For this purpose, SNT offers several closed-source model libraries that can be used to extend the capabilities of QualNet towards current network standards. For example, regarding wireless link layer protocols, there are modules available not only for IEEE 802.11a/b/g, but also for WiMAX, Zigbee and cellular networks based on GSM or UMTS. These modules have to be bought in addition to the QualNet simulation platform.

1.1.4 OPNET

Similar to QualNet, OPNET is a commercial simulator, offered by OPNET Technologies. While OPNET's structure and modules are very similar to QualNet – e. g. support of UMTS – its licensing options include a university program for teaching and non-commercial research. With this program, chosen technology modules are delivered with the source code, so that own protocol improvements and extensions can be made. A lot of these extensions, e. g. for mesh networks, are available free for download on the OPNET webpage [6].

In contrast to this, some of the developed modules are released to restricted customer groups only. For example, OPNET Technologies has founded a "WiMAX Model Development Consortium" which had several internal releases from September 2005 to July 2008. OPNET states that this model supports IEEE 802.16-2004 and IEEE 802.16e-2005 [8].

1.2 Licensing & Availability of openWNS

Typically, during system simulation obtained results must be evaluated, reviewed and defended. This was one of the reasons to release openWNS as open source.

Whereas most other open source simulation tools are released under the GPL for openWNS the Lesser GPL (LGPL) license was chosen. Compared to the GPL the LGPL additionally allows for closed source extensions. Still, all modifications to the openWNS libraries themselves must be made open source. This relaxation intends to alleviate the adoption of openWNS within the academical research community and the industry.

The home page of the openWNS can be found at [5]. The web page contains an installation guide, the documentation, including user and developer manuals, the web-based bugtracking, FAQs and the online frontend of the code version management system.

2. SIMULATION PLATFORM

The simulation platform of openWNS includes the core components of an event-driven stochastic simulation tool and is the basis for the simulation framework and simulation modules (see Figure 1). It is written in C++ and is heavily based on the Boost libraries [1] which provide already many features of the upcoming C++ standard [7]. openWNS includes multiple modules for different protocol layers as runtime plugins. While this paper focuses on the WiFiMAC module a more detailed description of the simulation platform (e. g. the pseudo-random number generator, event handling, etc.) and available modules (e. g. different traffic generators, TCP/IP, WiMAX, OFDM(A)) can be found in [12].

2.1 Configuration

The Python programming language is used for configuration of simulation scenarios, as shown on the left hand side of Figure 1). By chosing a programming language instead of a data representation language such as XML, instruments like loops, functions, inheritance, polymorphism and encapsulation are available. Especially during the setup of large scenarios with complex node types, this saves time and reduces errors.

Each simulation module is accompanied by reasonable default configurations. This allows users to setup their first simulations quickly and then start changing parameters incrementally.

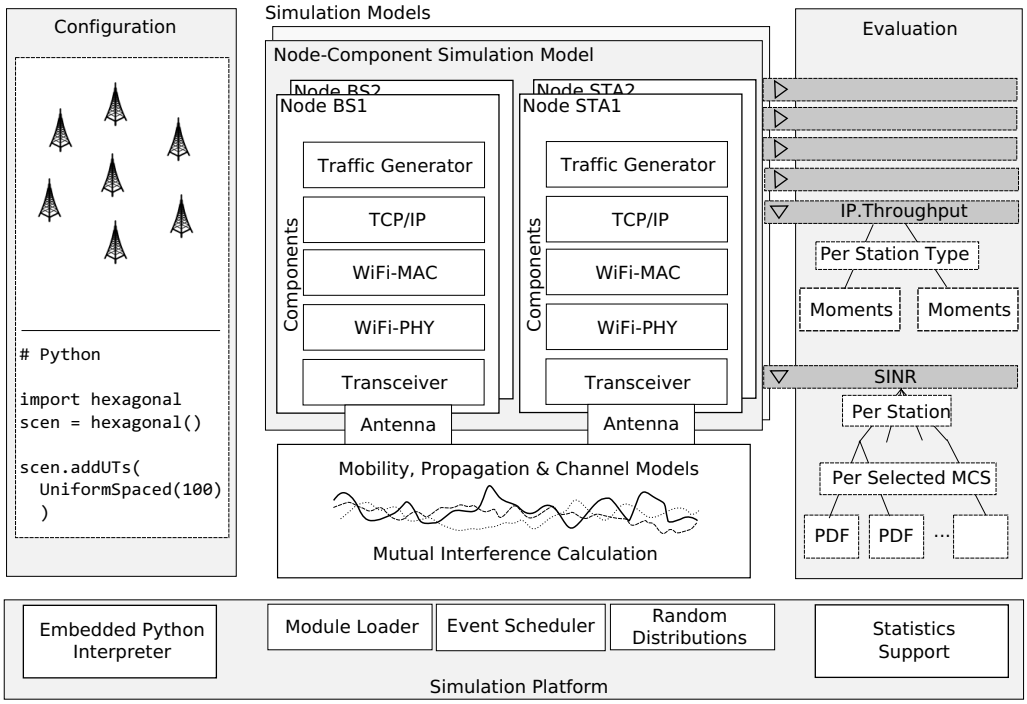


Figure 1: The openWNS simulation platform.

2.2 Evaluation

The evaluation subsystem of openWNS provides means to sort measurements according to the context and compress the data by statistically processing the measurements during the simulation on the fly. This is illustrated on the right hand side of Figure 1.

At compile time the developer defines measurement sources within the model and also the context information that accompanies each measurement (e. g. the node position).

At configuration time the user of the model can decide on the kind of evaluation that suits his investigation best. For instance, the user could configure an evaluation for a Signal to Interference plus Noise Ratio (SINR) measurement source. Then, the Probability Density Function (PDF) of false scheduling decisions can be gathered, sorted by the nodes and the modulation and coding scheme.

The online statistical evaluation saves memory. Furthermore, the clear distinction between the measurement source and the sorting stages makes it easy for users to quickly implement their desired evaluation while keeping the modules unchanged.

For publication-quality evaluation, the openWNS provides an implementation of the Discrete Limited Relative Error (DLRE) algorithm [24, 13] as evaluation sink. With the help of the DLRE, not only the confidence intervals of the mean, but of complete PDFs can be measured. Hence, reliable simulation results can be assured.

3. FUNCTIONAL UNIT NETWORKS

The development of a simulator often requires the implementation of recurring software patterns. The openWNS provides a framework that makes development of simulation models and often used parts of protocol stacks easy. This is achieved by a component-based development approach: Traditionally, each layer in a protocol stack represents a component that can be exchanged depending on the required functionality, e. g. TCP instead UDP on layer 4 to support a reliable delivery of data segments.

As proposed in [23], the openWNS takes this idea one step further: Here, a component – called Functional Unit (FU) – is a microscopic part of a protocol that implements a single function. This allows for a high level of reuse; furthermore, the most common FUs can be collected in a toolbox. Section 3.2 describes the Layer Development Kit (LDK) of the openWNS, as an example of such a toolbox.

In the end, a protocol stack is built of multiple connected FUs that make up a Functional Unit Network (FUN). For example, Figure 2 shows a small FUN composed of FUs to buffer data, handle acknowledgements (ARQ) and segment frames (SAR). Furthermore, the figure indicates that all FUs provide a set of methods to handle data flow, intra layer flow control and FU management within a FUN. This FU interface is described in the following sections.

3.1 Functional Units

If FUs are supposed to be composed and connected in an arbitrary way, the necessity for generalised interfaces arises. How should FUs be organised to support a wide range of different tasks as demanded by current and future protocols? How can these units be connected in a generic way to support the configuration of larger systems based on such units only?

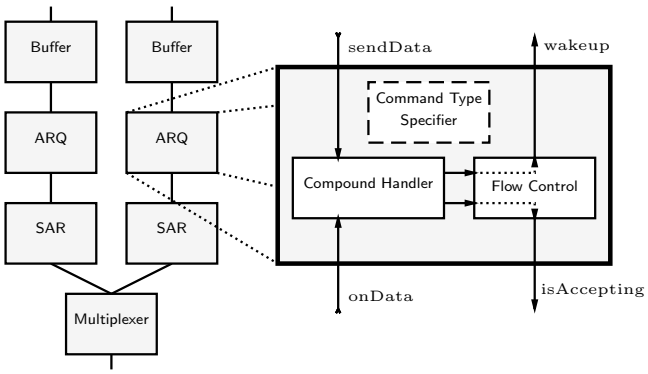


Figure 2: An exemplary Functional Unit Network (FUN)

The authors of the initial paper on FUNs identified four interfaces which are at least necessary to realize such an architecture [23]. These interfaces of an FU are depicted in Figure 3a. In the following sections these interfaces are described in more depth.

3.1.1 Data Handling Interface

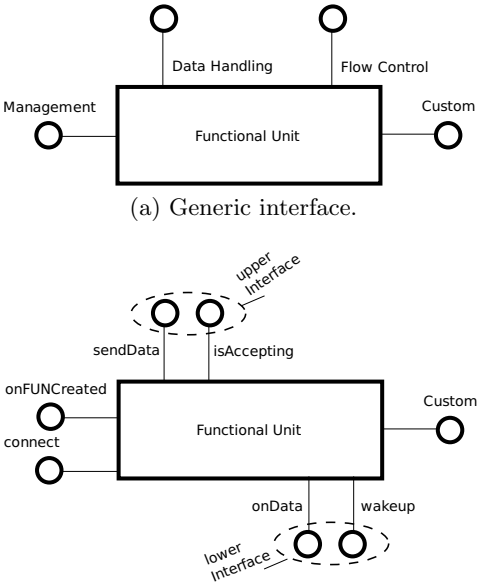
The most fundamental requirement for FUs is the ability to handle data. The basic data unit that is transmitted between FUs is referred to as *compound*. For now a compound can be seen as a chunk of data of variable size. FUs as part of a protocol stack may receive compounds for processing before and after such a compound has been transmitted over the air-interface. The first case is called outgoing data flow, while the latter case is referred to as incoming data flow. To support differentiation of the two directions the interface provides two methods: `sendData(Compound)` and `onData(Compound)` for compounds in the outgoing and incoming flow as depicted in Figure 3b.

3.1.2 Flow Control Interface

Every FU has only a limited capacity to store compounds and often FUs do not need to store compounds at all. However, the physical layer introduces a bottleneck, limiting the amount of information transmitted and thus the rate at which compounds can be handled. Thus the need for an intra layer flow control of outgoing compounds arises: FUs must have the ability to prevent other units from delivering compounds to them, when they decide not to accept additional ones. The implementation of this flow control mechanism is realized with two methods

- `isAccepting(Compound)`
- `wakeup()`

Before each `sendData` call the FU wishing to send data must make sure the target FU is accepting via the `isAccepting` method (see Figure 3b). The lower FU is thus able to stop the data flow if it cannot accept any more compounds. The `wakeup` of a FU is called by lower FUs to indicate that the lower (calling) FU is accepting data again. A good example for this is a Stop-And-Wait ARQ FU: the FU implementing the Stop-And-Wait ARQ is not accepting data while it is waiting for the acknowledgement of a transmission. After



(a) Generic interface.

(b) Logical interfaces towards upper and lower FUs.

Figure 3: Functional Unit interface.

the ARQ has received the acknowledgement for the transmission it can change its state to accepting again and will in turn call the `wakeup` method of the upper FU to signal its state change.

Note that intra layer flow control is only applied for outgoing flows. Flow control for incoming flows is basically the control of data rates between communication partners; hence, the implementation of a flow control protocol using explicit and/or implicit information exchange between the data source and the sink is needed.

3.1.3 Management Interface

The management interface of a FU offers the necessary functionality to manage the composition and configuration of a FUN. Since this paper does not focus on the management of FUs, the interface presented in Figure 3b does not show the complete set of functionality which is currently available. Two methods have been chosen as an example:

- `connect(FunctionalUnit)`
- `onFUNCreated()`

The `connect` method takes another FU as argument which should be connected to the FU in question. Special containers to support multiplexing and demultiplexing of data when the protocol stack is in full operation are maintained by this method.

`onFUNCreated` is a hook which is called by the surrounding framework of a FU to signal the successful creation of a FUN to the FU. Any special tasks the FU may need to undertake to get into a proper state for operation can be handled here.

3.1.4 Custom Interface

All aforementioned interfaces are generic interfaces of a FU which need to be supported by each FU in order to

ensure proper working as part of the described framework. However, it can be beneficial for FUs to offer additional interfaces. These interfaces are summarised under the Custom interface.

3.2 Layer Development Kit

Although existing protocols differ significantly from each other, several basic building blocks appear in each; e. g. the buffer to store outgoing data. The FUN architecture provides the framework to implement these building blocks as FUs that can be configured by the user according to the specific protocol requirements.

The openWNS includes the LDK which contains several different FUs that are unit-tested and easily configurable. Furthermore, these FUs may be used as starting points for more specialised implementations. Currently, the LDK includes FUs for the following tasks:

- Automatic Repeat Requests (ARQs): The following ARQ strategies are implemented and many parameters such as window sizes, retransmission timeouts or field lengths of sequence numbers may be configured.
  - Stop and Wait
  - Selective Repeat
  - Go Back N
  - Cumulative ACK
- Buffers: The implementations for buffers have bounded capacity and can either drop data in an overflow situation or use the FU flow control interface to stop the data flow.
- Multiplexing: There are FUs that allow for Many-to-One, or One-To-One connections in the outgoing direction of FUs.
- Flow Separation: A flow separator is the base functionality for implementing Quality of Service (QoS) aware FUNs. Compounds are categorised into different flows which then are processed by a dedicated FU or Sub-FUN within the flow separator.
- Flow Gates: Should be used in conjuction with flow separators. Gates give control on the flow control of distinct flows, e.g. in high load situation block all best-effort traffic.
- Concatenation: Multiple compounds are concatenated in the outgoing data path and incoming compounds are restored.
- Segmentation And Reassemblys (SARs): Both fixed and variable segment sizes are supported.
- Finite State Machine: Allows to build complex FUs. The users implementation is provided by a state machine.
- Probing: By inserting probing FUs into the FU the user may gather measurement results very quickly. There are statistic probes available that measure packet sizes, packet delays, incoming and outgoing throughput, error rates, etc.

4. THE WiFiMAC MODULE

To demonstrate the benefits of the simulation framework and the LDK toolbox the simulator module "WiFiMAC" is presented in this section. This module implements functions of the standard IEEE 802.11 for Wireless Local Area Networks (WLANs) so that it can be used for the evaluation of IMT-A scenarios. This module was chosen for presentation for several reasons: First, its Medium Access Control (MAC) function defined as the Distributed Coordination Function (DCF) is based on Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). This algorithm is sufficiently well-known so that a functional description can be omitted. Second, multiple amendments change the standard towards current and future demands. Within the openWNS, the modular FUN architecture requires only small and local changes to include the amendments, which becomes visible when comparing the FUNs of the "legacy" IEEE 802.11 MAC and the one including the enhancements from amendment n. Third, the standard IEEE 802.11 with its outdoor range of approximately 200 m (using the IMT-A Urban Micro pathloss model and 30 dBm transmission power) is the "smallest" radio access technology applicable to the small scale IMT-A scenarios.

4.1 The Basic WiFiMAC

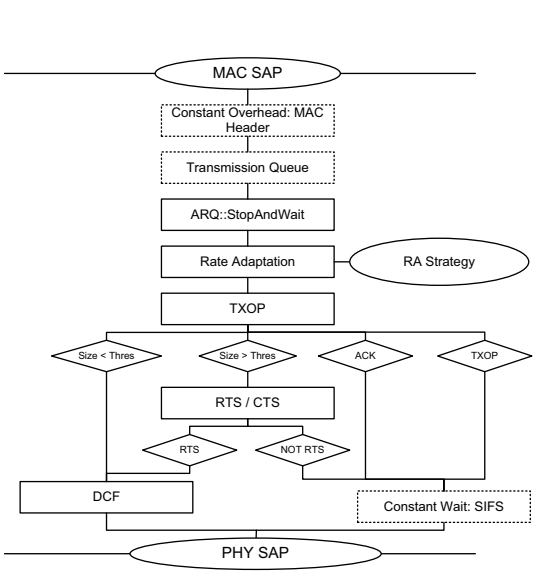
Figure 4a shows the WiFiMAC FUN in a configuration according to IEEE 802.11-2007. From a functional viewpoint, the FUs in the FUN can be sorted into two groups: The lower FUs, everything below the *TXOP*, are responsible for the timing of the MAC, i. e. when to give a compound to the Physical Layer (PHY) for transmission. All FUs above and including the *TXOP* act independently from the timing of the channel access.

Clearly, the different functions of the IEEE 802.11 MAC can be found in the FUN:

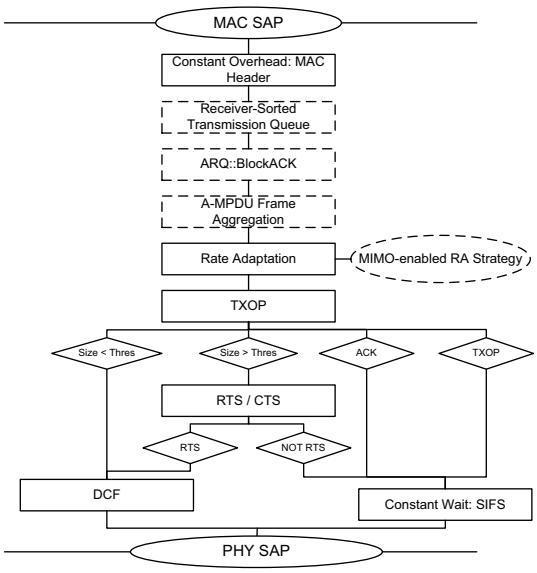
- The *Transmission Queue* is a size-limited buffer that stores outgoing compounds.
- The *StopAndWait* FU is derived from a similar FU available in the LDK and adopted to the special requirements of the IEEE 802.11 ARQ: The retransmission timer expires if no reception-start indication is received from the PHY within a constant delay, signalling the start of the Acknowledgement (ACK).
- The *RateAdaptation*, which can be configured using several different strategies, e. g. based on Packet Error Rate (PER) statistics, SINR measurements or a constant configuration.
- The *TXOP* allows to transmit several compounds in a row without repeating the time-consuming backoff, realizing the Transmission Opportunity (TxOP) procedure.

After the *TXOP* FU, a *compound switch* (available in the LDK) sorts the compounds according to different keys:

- Compounds that represent ACK frames or belonging to a TxOP are transmitted after a constant delay, the Short Interframe Space (SIFS).
- Compounds with a size below a configurable threshold use the *DCF*, which implements the CSMA/CA as defined in the standard.



(a) WiFiMAC in the IEEE 802.11-2007 configuration. Dotted FUs are taken directly from the LDK toolbox, solid FUs are taken from the toolbox and specialised for the WiFiMAC.



(b) WiFiMAC in the IEEE 802.11n configuration. Only the dashed FUs differ from the IEEE 802.11-2007 implementation.

Figure 4: FUNs of the WiFiMAC data link layer according to IEEE 802.11.

- The remaining compounds are stored in the *RTS/CTS* FU which precedes the transmission with the handshake of Request To Send (RTS)/Clear To Send (CTS) frames; only the RTS is transmitted using the *DCF*.

Flow control in this FUN is implemented entirely based on the operation of the *DCF* FU: Until the channel is idle and the FU's backoff has counted to zero, its interface does not accept any new compounds, which is propagated up to the *Transmission Queue*. After the backoff has reached zero, the FU sends a *wakeup* to the upper FUs, which finally reaches the buffer and thus initiates the transmission.

This transmission is only aborted if the RTS/CTS handshake is used and the CTS is not received: In this case, the *RTS/CTS* FU drops the stored compound and signals the transmission failure to the *ARQ* FU, using the specialised interface. Thus, the responsibility of retransmissions is solely at the *ARQ*.

4.2 Extensions for IEEE 802.11n

The amendment n of IEEE 802.11 increases the throughput of the standard beyond 100 $^{\mathrm{Mb}}/_{\mathrm{s}}$ measured at IP layer. To achieve this, several PHY improvements are incorporated, most importantly the MIMO capability with up to four parallel streams.

To deliver the gains of the PHY improvements to the application layer, an efficiency improvement of the MAC is required. For this aim, the amendment introduces two types of frame aggregation methods and extends the block acknowledgement procedure. Both functions can be found in specialised FUs that are integrated into the IEEE 802.11-2007 FUN, as given in Figure 4b:

- The *BlockACK* FU implements a selective reject ARQ, including one outgoing queue for transmitted, not acknowledged compounds and several incoming queues for out of order receptions.

- The *A-MPDU Frame Aggregation* FU implements aggregation of multiple compounds such that a single decoding error does not compromise the complete aggregation container. The simpler aggregation type, *A-MSDU*, can be easily implemented by inserting an aggregation FU from the LDK before the *Transmission Queue*.

Two other optional functions conclude the differences of the two configurations: The single-queue buffer of the basic configuration is replaced by the *Receiver-Sorted Transmission Queues* FU, which manages one buffer per receiver. In this way, the block acknowledgement and the frame aggregation can operate more efficiently as the probability for multiple frames in a row targeted for one receiver is increased. The receiver-sorted queue contains a strategy which is responsible for the selection of the next queue.

The remaining difference extends the ability of the rate adaptation strategy to select more than one spatial stream if the number of antennas at the transmitter and receiver and the expected SINR allow for this.

Obviously, the FUN framework enables a straightforward inclusion of the features of the amendment n into the existing FUN without major structural changes. This allows for the assessment that the inclusion of future amendments, e. g. those of the current task group ac, is possible with similar efforts and that the simulator can be timely adapted to changes in the standardisation process.

4.3 Other MAC Extensions

Some of the existing IEEE 802.11 hardware, especially Access Points (APs), support simultaneous transmissions

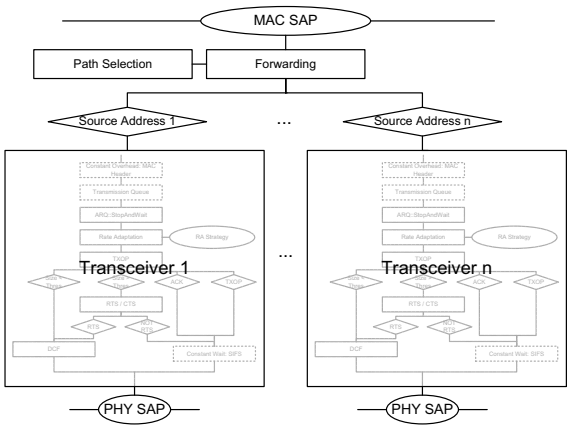


Figure 5: The WiFiMAC FUN realizing path-selection (for IEEE 802.11 amendment s) and multi-transceiver operation.

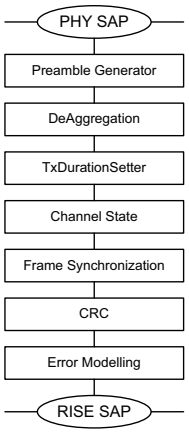


Figure 6: The WiFiMAC PHY Layer FUN

using two or more different transceivers, tuned to different frequency channels. This is for example used to support Stations (STAs) at 2.4 GHz and 5.5 GHz at the same time. A multi-transceiver capability is even more important in the case of IEEE 802.11 based Wireless Mesh Networks (WMNs): One transceiver, tuned to 2.4 GHz, is responsible for the last hop to the mobile STAs, whereas another (or even several other) transceiver relays data on 5.5 GHz in the mesh backbone.

The modularity and the standardised FU interface allow for a simple extension of the WiFiMAC towards multiple transceivers. As shown in Figure 5, the FUN of the MAC can be replicated as many times as needed. When a compound enters the MAC, a *Forwarding* FU determines, according to a path selection table, which next hop needs to be used and sets the appropriate source address of the transceiver. Similarly, if a compound is received by the *Forwarding* FU that has not reached its final destination, it is relayed accordingly. The population of the path selection table is done by a separate management FU.

4.4 The WiFiMAC PHY Layer

As a part of its framework, the openWNS simulator provides a Radio Interference Simulation Engine (RISE), which models the peculiarities of the wireless channel: pathloss (including stochastic Line of Sight (LOS)/Non-LOS models), spatially correlated random shadowing, fast fading and of course the calculation of interference during simultaneous transmissions.

The RISE accepts at its interface the start and end points of transmission events; similarly, it indicates the end of a transmission to receiving nodes, together with the calculated SINR.

To model the capabilities of an IEEE 802.11 a/g or n PHY, a convergence FUN between the RISE interface and the discussed MAC is required. This FUN contains the following FUs, see Figure 6:

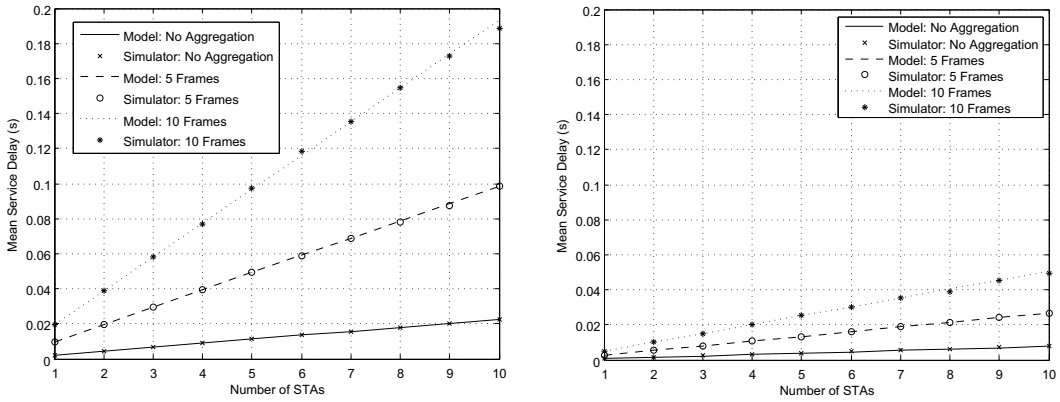
- A *PreambleGenerator* delays every compound and prefixes a special preamble compound, modelling the PHY preamble and header. This compound is used at the receiver to signal the start of a transmission (e. g. to the ARQ that waits for a pending ACK) and to model the successful signal synchronisation.
- The *DeAggregation* FU splits, in case the A-MPDU aggregation is used, the aggregated compound into multiple segments so that the SINR (and thus the error probability) is calculated separately for each segment. In this way, the ability of A-MPDU aggregation to decode frames in the aggregation container although a decoding error has corrupted one segment is modelled.
- A *TxDurationSetter* calculates from the compound length and the selected MCS the transmission duration.
- The *ChannelState* implements the IEEE 802.11 channel state detection, comprising the physical- and the virtual clear channel assessment.
- Finally, three FUs model decoding errors of incoming compound based on the indicated SINR: First, an *ErrorModelling* FU calculates the PER based on the SINR, the selected MCS, the number of spatial streams and receive antennas and the compound length (detailed description and further analysis can be found in [20, 19]; the MIMO model is described in [14, 18]). Then, a *CRC* FU drops compounds according to the PER. Finally, a *FrameSynchronization* FU models the capture effect as described in [17].

5. SIMULATOR VALIDATION

As with every other nontrivial software, programming errors may occur during the development phase. However, to assure sound simulation results, validation of the implementation using black-box testing is performed: Simulation results of reference scenarios are compared against results available from reviewed literature. To ensure comparability of the results, all details of the reference scenarios must be available, which is often not the case in results created by closed-source simulators or simulator extensions.

Therefore, we have selected the well known analytical IEEE 802.11 DCF model from [10] (refined in [11]) and have implemented it in Matlab. While it is possible that this implementation also contains errors, the probability to archive the same incorrect results using two fundamentally different approaches is low.

The limit of the analytical model in comparison to the openWNS is the restriction to scenarios where all STAs are



(a) No spatial multiplexing.

(b) 4x4 MIMO.

Figure 7: Comparison of the mean service delay at saturation according to [11] and simulated by the openWNS.

in mutual reception range of each other, i. e. the effect of hidden nodes (or even rate adaptation) is not included. Hence, the reference scenario consists of a single AP, closely surrounded by STAs with saturated uplink traffic sources.

A key result from [11] is the derivation of an expression for the mean service delay in a saturated network, dependent on the number of STAs and the PHY parameters. By setting the correct values for the frame and preamble durations, it is possible to apply the model even for the amendment n. The mean service delay measurements are easily obtained by inserting a *Delay Probe* FU (available in the LDK) in the FUN directly below the transmission queue. This probe adds a time stamp to outgoing compounds; the peer FU at the receiver can thus calculate the service delay of the compound.

Figure 7 shows the mean service delay with increasing number of STAs and different aggregation lengths. The two Figures 7a and 7b differ in the number of antennas used for MIMO transmission, and thus in the number of spatial streams: While Figure 7a uses the common 1x1 setup, the results from Figure 7b assume a 4x4 MIMO configuration.

All results generated by the openWNS show a precise match of the graph given by the analytical model. Hence, the most important FUs of the WiFiMAC (*ARQ*, *RTS/CTS*, *DCF*, *A-MPDU Frame Aggregation*) and their combination into the FUN as given in Figure 4b reflect the model correctly.

The testing framework of the openWNS supports the setup of system tests, which performs the described validation automatically by comparing freshly generated simulation results with stored reference values. In this way it is possible to ensure the correctness of existing code if new functionality is added and FUs are changed during the development of new protocol features.

6. SIMULATION OF IMT-A SCENARIOS

The ITU-R report M.2135 [15] contains an in-depth description of test environments and deployment scenarios for evaluation of IMT-A candidate technologies. The report describes thirteen performance measures; three of them have to be evaluated by system simulation: Cell spectral efficiency, cell edge user spectral efficiency and Voice over IP capacity. Furthermore, each evaluation has to be done in five deployment scenarios, ranging from an indoor hotspot to rural macro-cells. The scenarios are defined by the device capabilities (e. g. height, number of antennas, maximum transmission power, receiver noise figure) as well as the channel model, user distribution and user mobility.

```
for params.numChannels in [1, 2, 3, 4, 5]:
    for params.numAntennas in [1,4]:
        for params.seed in range(1,11):
            params.write()
```

Figure 8: Campaign setup using Python.

In the following, the procedure how to simulate IMT-A scenarios is explained, showing how the user can benefit from the available simulation and evaluation framework. As a complete evaluation of the scenario types and performance measures would exceed the scope of this paper, we use as an example the evaluation of the downlink cell spectral efficiency in the urban micro scenario.

The used radio access technology is IEEE 802.11 (including amendment n) as described in Section 4.2; the evaluation should assess the impact of the number of antennas (1 or 4) and channels (1 to 5).

6.1 Campaign Setup

In the openWNS terminology, a set of parameters with different values together with a base configuration is named a "campaign". In our case, the campaign parameters are the number of antennas and channels as given above, plus (a) the random number generator seed to generate several different "drops", i. e. scenarios that differ in the positions of the STAs, as required by the evaluation guidelines, and (b) different values for the offered traffic per STA.

The starting point for the setup is a campaign configuration file that defines the required parameter sweeps. As the programming language Python is used for the simulator configuration, all language constructs can be employed. The simple Python fragment given as Figure 8 creates $5 \cdot 2 \cdot 10 = 100$ different simulations.

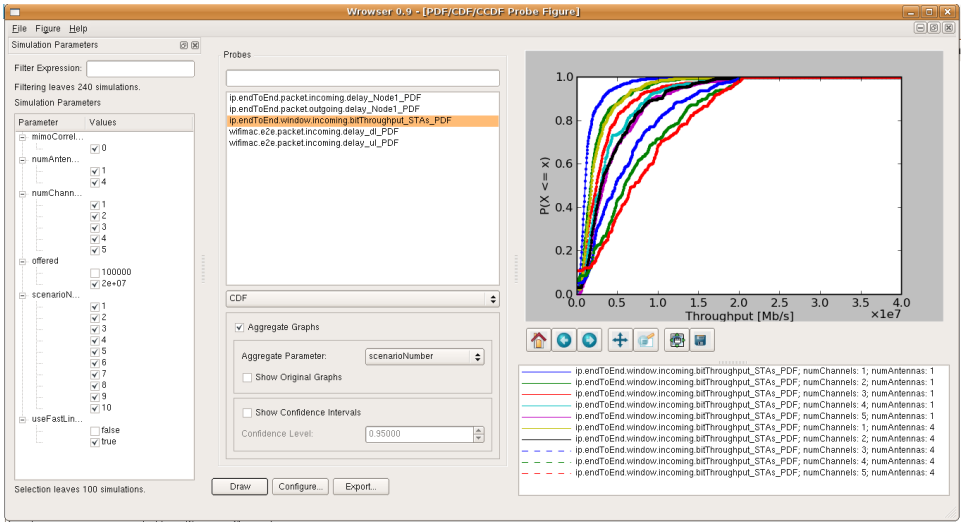


Figure 9: Screenshot of the openWNS graphical evaluation frontend "Wrowser".

6.2 Cluster Simulations

With a duration of 2 to 9 h per simulation (depending on the number of channels: less channels result in more interference calculations and thus more runtime), the campaign requires approximately 16 days of processor time.

One of the most advanced features of the openWNS simulation platform is its support for cluster computing. Currently, openWNS offers an interface to the Sun Grid Engine (SGE), which allows for executing the simulations of a campaign in parallel on as many cores as available. Thus, by using a cluster with 100 cores, the simulations are finished after 9 h.

After completion, simulation results can be written into a Postgresql database, so that data mining methods can be used (e. g. slicing, dicing, aggregation) for the evaluation.

6.3 Campaign Evaluation

Collecting results, extracting measurements and generating parameter plots is often very time consuming and error prone. The openWNS offers the *Wrowser* (an acronym for *W*ireless network simulator *R*esult Br*owser*, see [9]). which solves this problem and lets users focus on the research rather than on the scripts that collect their measurements.

As soon as a simulation is finished, the Wrowser can be used to access the results from the central database. Wrowser is aware of all the simulation parameters and aggregated parameter plots can be generated within a few steps. Figure 9 shows the GUI, plotting the cumulative probability function of the STAs' throughput, depending on the number of channels and antennas. Results for drops with the same parameter values are automatically aggregated into one graph.

The Wrowser supports the export of the plots as comma-separated value files and Matlab programs, so that further processing of selected results is possible. With a little knowledge of Matlab, publication-quality graphs are generated in few minutes. For example, Figure 10 shows the results of the simulation campaign, where the carried traffic per STA is converted to the cell spectral efficiency by dividing the results by the number of 20 MHz-channels and cells in the

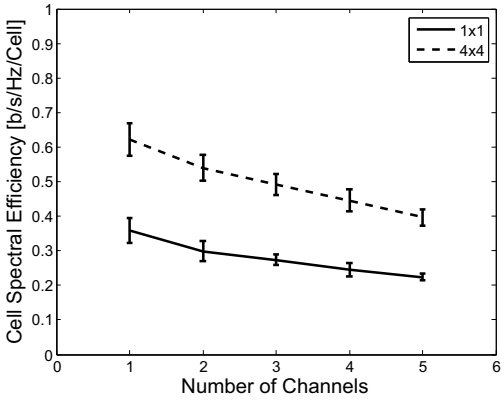


Figure 10: Downlink cell spectral efficiency of IEEE 802.11n-2009 in the IMT-A Urban Micro scenario with different antenna and channel configurations.

scenario. As the results are averaged over multiple drops, the Wrowser is able to add confidence intervals to the mean values. Here, the 0.95-confidence interval is plotted.

The cell spectral efficiency decreases with an increase of channels. The reason for this is the following: although the throughput is increased with the help of more channels, the bandwidth is not used efficiently enough to compensate for the increase of the denominator of the spectral efficiency. Furthermore, the step from a 1×1 to a 4×4 configuration only achieves twofold spatial efficiency increase.

7. CONCLUSION

The paper at hand provides a complete overview of openWNS: First, the FUN architecture which facilitates a rapid prototyping of current and future protocol stacks is presented. The architecture is illustrated using the WiFiMAC as an example – especially the implementation of the MAC enhancements of IEEE 802.11n-2009. Second, the builtin

support to generate, configure, execute and evaluate large-scale simulation campaigns as required by the IMT-A evaluation criteria is shown.

A description of all aspects of the current openWNS would require significantly more space than available for this paper. For example, the presentation of the modules for the other radio access technologies that are currently under development – WiMAX and LTE-Advanced – is much more complex than the description of the WiFiMAC. Hence, we refer again to the online documentation that can be found at [5].

8. ACKNOWLEDGEMENTS

The development and release of openWNS would not have been possible without the support, hard work and endless efforts of a large number of diploma thesis workers and PhD students. We are particularly grateful to our colleagues Maciej Mühleisen, Ralf Pabst, Arif Otyakmaz, Klaus Sambale, Rainer Schoenen and Matthias Malkowski for their dedication and contribution to openWNS. Finally, we would like to thank Prof. Walke who made the work on this simulator possible.

9. REFERENCES

[1] Boost C++ Libraries. Web Page `http://www.boost.org/` (Retrieved 2009-10-15).

[2] Global Mobile Information Systems Simulation Library (GloMoSim). Web Page, `http://pcl.cs.ucla.edu/projects/glomosim/` (Retrieved 2009-10-15).

[3] ns-2. Web Page, `http://www.isi.edu/nsnam/ns/` (Retrieved 2009-10-15).

[4] ns-3. Web Page, `http://www.nsnam.org/` (Retrieved 2009-10-15).

[5] open Wireless Network Simulator (openWNS). Web Page, `http://www.openwns.org` (Retrieved 2009-10-15).

[6] OPNET. Web Page, `http://www.opnet.com/` (Retrieved 2009-10-15).

[7] TR19768 Technical Report on C++ Library Extensions.

[8] WiMAX (IEEE 802.16) Specialized Model. Description available at `http://www.opnet.com/solutions/brochures/wimax_model.pdf` (Retrieved 2009-10-15).

[9] Wireless network simulator Result Browser (Wrowser). Web Page, `http://launchpad.net/openwns-wrowser` (Retrieved 2009-10-15).

[10] G. Bianchi. Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE Journal on selected areas in communications*, 18(3):535–547, 2000.

[11] G. Bianchi and I. Tinnirello. Remarks on IEEE 802.11 DCF performance analysis. *IEEE Communications Letters*, 9(8):765–767, 2005.

[12] D. Bültmann, M. Mühleisen, K. Klagges, and M. Schinnenburg. openWNS - open Wireless Network Simulator. In *15th European Wireless Conference*, Aalborg, Denmark, May 2009.

[13] F. Schreiber, C. Görg. Stochastic Simulation: a simplified LRE-algorithm for Discrete Random Sequences. *AEÜ*, 1996.

[14] D. Gore, J. Heath, R.W., and A. Paulraj. On performance of the zero forcing receiver in presence of transmit correlation. *Information Theory, 2002. Proceedings. 2002 IEEE International Symposium on*, pages 159–, 2002.

[15] ITU-R. M.2135 : Guidelines for evaluation of radio interface technologies for IMT-Advanced. Technical report, ITU, 2008.

[16] S. Kurkowski, T. Camp, and M. Colagrosso. Manet simulation studies: The Incredibles. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(4):50–61, 2005.

[17] J. Lee, W. Kim, S. Lee, D. Jo, J. Ryu, T. Kwon, and Y. Choi. An experimental study on the capture effect in 802.11a networks. In *Proceedings of the second ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*, pages 19–26, Montreal, Quebec, Canada, 2007.

[18] J. Mirkovic. *Design and Performance Analysis of MIMO Based WLANs*. PhD thesis, RTWH Aachen University, Department of Communication Networks, Dec 2008.

[19] G. Orfanos. *Development and Performance Evaluation of an MAC Protocol for MC-CDMA Wireless LANs with QoS Support*. PhD thesis, RWTH Aachen University, Department of Communication Networks, 2006.

[20] G. Orfanos, J. Habetha, and W. Butsch. Error probabilities for radio transmissions of MC-CDMA based W-LANs. In *Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st*, volume 3, pages 1706–1710 Vol. 3, 2005.

[21] Scalable Network Technologies. Qualnet. Web Page, `http://www.scalable-networks.com/` (Retrieved 2009-10-15).

[22] M. Schinnenburg, F. Debus, A. Otyakmaz, L. Berlemann, and R. Pabst. A framework for reconfigurable functions of a multi-mode protocol layer. In *Proceedings of SDR Forum 2005*, page 6, Los Angeles, U.S., Nov 2005.

[23] M. Schinnenburg, R. Pabst, K. Klagges, and B. Walke. A Software Architecture for Modular Implementation of Adaptive Protocol Stacks. In *MMBnet Workshop*, pages 94–103, Hamburg, Germany, Sep 2007.

[24] F. Schreiber. Time efficient simulation: the LRE-algorithm for producing empirical distribution functions with limited relative error. *AEÜ*, 38, 1984.

[25] WINNER: Wireless World Initiative New Radio+. Web Page, `http://projects.celtic-initiative.org/winner+` (Retrieved 2009-10-15).

[26] X. Zeng, R. Bagrodia, and M. Gerla. GloMoSim: A Library for Parallel Simulatin of Large-scale Wireless Networks. In *12th Workshop on Parallel and Distributed Simulations (PADS'98)*, May 1998.