openWNS - open Wireless Network Simulator

Daniel Bültmann, Maciej Mühleisen, Karsten Klagges
Department of Communication Networks
Faculty 6, RWTH Aachen University, Germany
Email: {dbn,mue,kks}@comnets.rwth-aachen.de

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

***Abstract*—This paper presents the open Wireless Network Simulator (openWNS). This simulation tool was developed in the last 5 years at the department of Communication Networks (ComNets) at RWTH Aachen University and has been released as open source software recently.**

openWNS is a dynamic event driven system level simulation platform that allows for investigation of dynamic protocol behaviour in multi-cellular scenarios with detailed interference modeling. The simulation platform follows a modular design down to protocol building blocks, which makes it possible to rapidly modify the implemented protocol stacks. openWNS currently includes models from physical to application layer. The protocol modules for IEEE 802.16m and IEEE 802.11n-draft are probably the most interesting for the wireless research community.

I. Introduction

Performance evaluation by means of simulation is an integral part of any standardization, system development or research activity. It allows for conducting repeatable experiments in a controllable low-cost environment. Typically such activities involve multiple parties, which pursue different interests. This usually leads to a situation where results of own evaluations need to be defended and evaluation results of other parties need to be reviewed. In such situations a common simulation platform has a significant potential for reduction of cost and effort, quality increase and process speed-up. This was one of the reasons for the decision to release the simulation platform used and developed at ComNets to the open source community. For additional information on openWNS see [1].

Whereas most other open source simulation tools are released under the GNU General Public License (GPL) for openWNS the Lesser GPL (LGPL) license was chosen. Compared to the GPL the LGPL additionally allows for closed source simulation modules if you only use (link against) openWNS, but still all modifications to the openWNS libraries themselves must be made open source. This relaxation was accepted to alleviate the adoption of openWNS within the industry.

The presented simulation tool is highly modular and allows users to select an extension point, which fits best to their needs. However, most of the protocol models that were released are based on an implementation of the Functional Unit Networks (FUNs) [2] [3]. Modularization is consequently applied even to protocol building blocks, such that new protocols can be easily built by selecting appropriate blocks from the Layer Development Kit (LDK) - a toolbox of protocol building blocks such as Automatic Repeat Request (ARQ), Segmentation And Reassembly (SAR), buffers, schedulers, etc.

openWNS has builtin support for simulation and compilation clusters. Simulation campaigns can be easily managed by users and results of parallel simulation runs can be browsed with a graphical front-end. The backend is built by a relational database and a grid engine such as SUN's SGE.

The rest of the paper is structured as follows. At first, an overview of other simulation tools is given. Then the simulation platform and the available simulation framework of openWNS are described. Afterwards the released simulation modules are presented and a conclusion is given.

II. Related Work

In this section, a current view on system level simulation with special respect to open source approaches is given. System level simulators for the performance evaluation of communication systems are available for almost two decades. The most prominent among them are probably ns- 2 [4] and its successor ns-3 [5] (open source), OMNeT++ [6] (open source only for academic use) and OPNET [7] (commercial). Additionally, a number of other (open-source) simulators are available: GloMoSim/QualNet (QualNet is the commercial successor of the open source simulator GloMoSim), NCTUns, GTNet, SSFNet and JiST.

In the following a summary of ns-2/ns-3, OMNeT++ and OPNET is provided to give a rough overview about the existing tools.

A. ns-2 and ns-3

The original development of ns-2 started as early as 1989 as REAL simulator. The first release of ns-2 was available in 1996. The main drawbacks 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. ns-3 in comparison to ns-2 aims at providing better support in the following items:

- Modularity of components
- Scalability of wireless simulations
- Integration/reuse of outside code
- Emulation
- Tracing and statistics
- Validation

At the time of writing this paper the current stable release is ns-3.3 as of 18th December 2008, that contains support for IEEE 802.11.

B. OMNeT++

OMNeT++ has been developed by Andras Varga and is a discrete event driven simulation environment with a number of additional models (e.g., for TCP/IP, peer-to-peer networks and LAN protocols) being available. Modeling of the physical and link layer is mainly provided by the Mobility Framework of OMNeT++ [8]. Currently, OMNeT++ support for IEEE 802.11 and sensor networks. Support to model long-term or short-term fading, are available through the MiXiM framework [9].

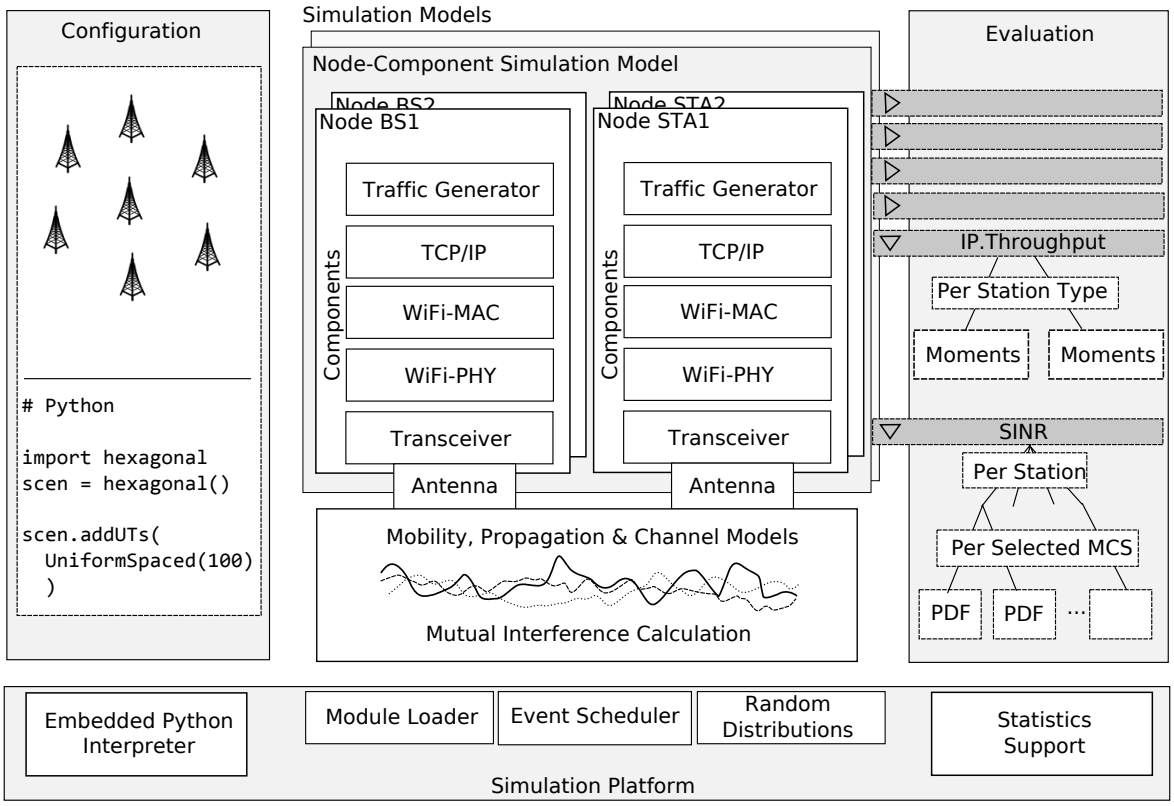

Fig. 1. openWNS Structure

C. OPNET

Whereas ns-2, ns-3 and OMNeT++ are freely available, OPNET is a commercial tool. While higher ISO/OSI layers (above layer 2) are the focus of these tools, the lower layers and especially the physical layer and the transmission characteristics are modelled very simple. The support for wireless communication systems in terms of detailed interference modelling, new channel models (e.g. for MIMO), node mobility, different receiver models (e.g. for different multiple access schemes) is very limited. The IEEE 802.11 model supports the amendments a,b,e and g. There is also a community supported IEEE 802.11n model available.

OPNET has founded a "WiMAX Model Development Consortium" which had several internal releases from September 2005 to July 2008. These releases are only available to the consortium members. OPNET states that this model supports IEEE 802.16-2004 and IEEE 802.16e-2005 [10].

III. Simulation Platform

This section introduces the simulation platform of openWNS, which includes the core components of an event-driven stochastic simulation tool and is the basis for the simulation framework and simulation modules (cmp. Figure 1). It is written in C++ and is heavily based on the Boost libraries [11] which provide already many features of the upcoming C++ standard [12], today. openWNS is separated in multiple modules (shared libraries) which can by loaded dynamically by the simulation platform if needed.

A. Event Scheduler

The simulation platform provides both real-time and non-real-time schedulers. The real-time scheduler can be used to build demonstrators and implement interaction with the simulation host (see Section V-D for first steps to do this). The non-real-time event scheduler uses a two layer data structure as shown in Figure 2. For each distinct simulation time a map entry is created. Each map entry contains a list of all events that are queued at this time. With this structure the following complexities are achieved:

- Schedule an event at current simulation time: $O(1)$ within the current bucket the new event simply needs to be appended to the end of the list.
- Schedule a delayed event: $O(log(N))$ where N is the number different simulation time instances for which at least one event already exists.
- Cancel an event: $O(1)$ after an event has been scheduled the scheduler returns a handle to the event which must be used to cancel the event. The handle makes searching for the event prior to deleting obsolete.
- Retrieve next queued event: $O(1)$ just retrieve the head of the queue at the current simulation time.

It is important to state here that the programmers interface does not put any restrictions on the event types. The event scheduler only requires the call operator to be implemented by queued events, i.e. all events must be functors. To further facilitate the usage it is recommended to apply the function argument binding mechanisms provided by the Boost bind, function and lambda libraries [13].

B. Random Distributions

The random number generator is based on the Mersenne Twister algorithm [14]. The implementation that is used is the one provided by the Boost random library (i.e. mt19937). The

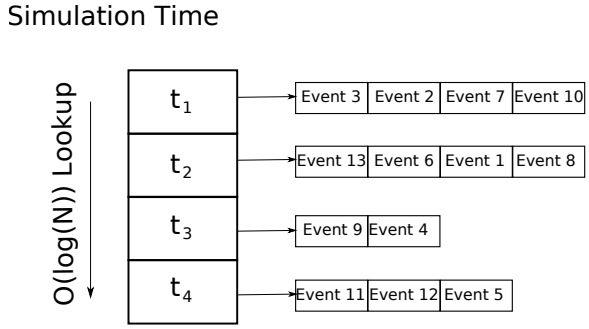

Fig. 2. Event Scheduler Data Structures

algorithm provides a period of of $2^{19937} - 1$ and passed a number of stringent statistical tests.

The available random number distributions include Uniform, Normal, Exponential, Poisson, Ricean, Pareto, Erlang, and Binomial. Furthermore, the random number distributions provided by the boost random library are available.

For debugging purposes it is possible to use multiple base generators. In this way one could use a fixed seed for the mobility components but use random seeds within the link to system level interface packet error rate experiments.

C. Configuration

The Python language is used for configuration of simulation scenarios (cmp. Figure 1). The most important advantage to choose a programming language instead of a data representation language such as XML is its scalability. To be useful for a wide range of users the configuration mechanism must be capable to scale with the scenario size and also scale with increasing complexity of simulation models. With an object oriented programming language the first scale-up can be achieved by functional decomposition of the scenario setup task, while the second can be achieved through sub-classing or structural composition of class hierarchies. Python was chosen for its syntactical clarity and its wide support within the open source community. Each simulation module (cmp. Section V) is accompanied by reasonable default configurations. New users can quickly setup their first simulations and then start changing parameters incrementally. PyTree is the graphical configuration viewer that is included in openWNS.

D. Evaluation

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

At compile time the developer defines measurement sources within the model and also defines context information that accompanies each measurement (i.e. the node position, the base station to which it is associated, the used modulation and coding scheme, etc.).

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. The Probability Density Function (PDF) for each station and for each modulation coding scheme can be gathered to determine the number of false scheduling decisions. Unused measurement sources have neglible overhead.

The major advantage of this approach over post-mortem measurement evaluation is the support for longer simulation runs. When running large simulation campaigns storage capacity is soon a problem. For example, consider a simulation campaign which collects the mean SINR for 100 user terminals within a cell. Assuming that a double precision float value (8 bytes) is used for the value and that every frame (2ms) a new measurement is generated, the data rate for this scenario would be 800 bytes per frame. With 100 drops (terminal positions are fixed but choosen randomly) and 100 seconds simulation time for each drop the necessary storage capacity would be 4GB - only for the SINR values.

The online statistical evaluation saves space. Furthermore, the clear distinction between the measurement source and the sorting stages makes it easy for users to quickly implement their desired evaluation. No changes to the original models have to be made.

IV. Simulation Framework

The development of a simulator often requires the implementation of recurring software patterns. The openWNS provides a framework that makes developing of protocols easy. The goal of the simulation framework is to make development of simulation models and often used parts of protocol stacks easy to implement and to configure. This is achieved through well-defined clear interfaces, a rich set of predefined protocol building blocks and a high degree of code reuse, which is achieved by a component-based development approach.

A. Simulation Model

There is an indispensable need to simulate both, simple queueing systems as well as complex simulation scenarios with an entirely equipped protocol stack. The openWNS provides a software architecture that supports both. Each simulation is defined through the simulation model which specifies two basic methods: `start()` and `shutdown()`. These methods define the entry point of the simulation model and a point of notification about the end of the simulation. At startup a simulation model will typically setup the investigation scenario and schedule events. Thereafter, the simulation model is driven by the event scheduler's main event loop. The shutdown method is used to properly shutdown the model and to gather final statistics.

B. Node-Component Model

As stated above the simulation is based on a simulation model which can be a simple queueing system or a more complex scenario with several stations. The Node-Component model allows for the flexible specification of protocol stacks. Therefore, each station is represented by a `Node` class. Each Node contains a set of components which represent the protocol layers, equivalent to protocol layers of the ISO/OSI reference model. Figure 1 shows the structure of the Node-Component Simulation Model. Usually each simulator module defines a specific component type, that can be instantiated inside a node, see also Section V.

C. Layer Development Kit

Protocol layer development is often the fundamental step of developing an openWNS module. Protocol layers in openWNS correspond to ISO/OSI layers and are subdivided into Functional Units (FUs). There is a simple mechanism to connect FUs. These connected FUs form a Functional Unit

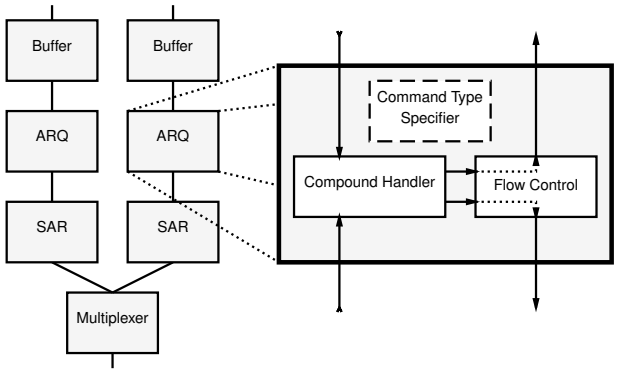

Fig. 3. Functional Unit Networks

Network (FUN) and represent the central packet processor of the openWNS layer, see Figure 3.

Messages between and inside layers are transmitted through compounds of commands, which is similar to the blackboard software pattern [15]. Each FU defines a unique command type. Each compound contains a single instance of the specific command, that is defined by the command type specifier and which can only be accessed by the FU.

The compound handler is the central element of the FU. It defines the actions that are performed for incoming and outgoing compounds. Often, the developer has only to define the function of the compound handler. Other elements of the FU can easily be aggregated by predefined components of the LDK toolbox.

Another important aspect of the LDK is flow control. FUs provide flow control for both, incoming and outgoing packets. Each FU provides an interface that gives information, whether a compound would be accepted. Hence upper FUs ask lower FUs before they send outgoing compounds. For incoming compounds, flow control in terms of blocking is not necessary.

New protocols can be easily built by selecting appropriate blocks from the LDK, that contains a predefined toolbox of protocol building blocks such as ARQ, SAR, buffers, schedulers, multiplexers, de-multiplexers, etc. A detailed description of the FU concept can be found in [3].

V. Simulation Modules

This section presents the simulation modules included in openWNS. Starting with the channel and interference modelling, the WiMAX and WiFi data link layer modules are presented. openWNS allows for simulations that include multi-standard nodes that may operate concurrently below the IP network layer. The transport layer modules for TCP and UDP are introduced. At the end of this section the available traffic models are presented which can be operated either on top of the data link, network or transport layer.

A. RISE - Radio Interference Simulation Engine

RISE manages node mobility (brownian movement, roadmap, polygon paths, etc.) and interference calculation. The channel model is used to calculate total received signal strength for every transmission by using the formula

$$P_R = P_T - L_{PL} - L_{Sh} - L_{FF} + G_T + G_R \qquad (1)$$

P_R is the received power, P_T the total emitted power by the transmitter, L_{PL}, L_{Sh}, L_{FF} the losses due to path-loss, shadowing, and fast fading, and $G_T(\phi, \theta)$, $G_R(\phi, \theta)$ are the antenna gains at the transmitter and receiver. The radio propagation model can be independently chosen for each transceiver type pair. This can be used for example to have different models for different moving speeds or to define Line of Sight (LOS) and Non Line of Sight (NLOS) connections.

It is possible to include directive antenna models which depend on ϕ and θ. Two antenna types are distinguished. The static antenna is described by its gain in all directions. The beamforming antenna allows to dynamically adjust its directivity. The algorithm used to calculate the gain is the optimal beamformer algorithm described in [16].

Several models to calculate the path loss between transmitter and receiver are available. Those are:

- Constant (distance independent)
- Free space
- Single slope
- Multi slope

Distance ranges can be defined and a model applied for each range. The single slope model is described by the equation $L_{PL} = (\frac{\lambda}{4\pi d})^{\gamma}$. d is the distance between transmitter and receiver, λ the electromagnetic wavelength and γ the propagation coefficient. In a logarithmic notation γ becomes the slope. Free space propagation is a special case of the single slope model with $\gamma = 2$.

The multi slope model is created by defining multiple distance ranges using single slope propagations with different propagation factors. Constant, distance independent path loss is usually applied for very short or very long distances. The pathloss models for the IMT-Advanced evaluation have already been partly included [17].

Different shadowing models to describe the scenario are available. These models describe the influence of solid obstacles on radio wave propagation. Three different models are available:

- Map based
- Scenery object based
- Spatially correlated

The map based model assumes fixed base station positions. The shadowing is pre-calculated for each base station by a map of the signal degradation due to shadowing at several sampling points on the scenario. The signal strength between sampling points is interpolated.

The scenery object based model includes geometric obstructions with a fixed penetration loss. The total shadowing is defined by the total penetration loss of all penetrated walls assuming LOS propagation. This is typically used to create indoor scenarios with walls or outdoor scenarios with whole buildings. In contrast to the map based model this model does not require fixed base stations to be one communication end point. It can therefore be used for mobile-to-mobile station communication.

Spatially correlated shadowing is modeled stochastically. A description of the model can be found in [18]. It is based on a sequence of correlated, log-normally distributed random values.

Additionally to shadowing and path-loss, a fast fading model can be enabled. Currently, rician fading [19] as well as time correlated and frequency selective models are available. Time correlation is modelled according to the Jakes model [20]. The frequency selective fading process is modelled according to [21].

B. WiFi

openWNS includes a model for the simulation of IEEE 802.11 Wireless Local Area Networks. The physical layer model allows to simulate frame transmissions on wireless channels with 20/40 MHz bandwidth. It is based on the *RISE* (see Section V-A) module and thus supports pathloss, shadowing, fast fading and interference calculation.

The physical layer model supports both virtual and physical carrier sensing. The link-to-system level interface is based on a SINR to packet error rate mapping, which takes into account the modulation and coding scheme and the packet length. A simple stochastic model of MIMO gains [22] has been included.

The medium access control layer implements the IEEE 802.11 Distributed Coordination Function (DCF). Additionally, the RTS/CTS transmission mode, transmission opportunities according to IEEE 802.11e are implemented. Furthermore, the block acknowledgements and frame aggregation methods of IEEE 802.11n are available, as well as SINR and ARQ-based rate adaptation with MIMO support.

An additional path selection module allows for simulation of mesh networks according to IEEE 802.11s and has already successfully been used for performance evaluation [23].

C. WiMAX

The openWNS additionally supports the IEEE 802.16 protocol, also known as WiMAX. Since the WiMAX protocol realizes a frame based medium access scheme, the openWNS has been enhanced to support periodically timed frames. The WiMAX medium access control module (WiMAC) supports the Orthogonal Frequency Division Multiplex (OFDM) physical layer Time Division Duplex (TDD) profile of the IEEE 802.16e standard. Also, WiMAC supports the Orthogonal Frequency Division Multiple Access (OFDMA) profile for flat channels. The implementation provides special packets for Frame Control Header (FCH), downlink (DL) and uplink (UL) maps, ranging messages, association and connection establishment packets and bandwidth requests.

In the recent years, WiMAC has also be enhanced to support relay enhanced multihop communication in cellular scenarios. The IEEE 802.16j task group has put significant effort in developing medium access techniques for the relay enhanced cellular system. WiMAC implements the transparent relay mode, which makes multihop operations possible, even for unmodified subscriber stations.

D. TCP/IP Module

The Internet Protocol (IP) module included in openWNS implements a subset of IP version 4. Within each simulation node an unlimited number of data link layers may be included. Each is handled similar to a device node in a real computer system. This allows for simulation of hybrid multi-technology nodes. *Virtual* services for ARP, Domain Name System (DNS) and Dynamic Host Configuration Protocol (DHCP) have been implemented, whereby *virtual* denotes that there are no Protocol Data Units (PDUs) actually transmitted, but the service is realized transparently within the simulation tool. It is possible to include delay models for each of these services.

By now, only static routing tables with Time To Live (TTL) support have been implemented, but the flexible architecture allows for extension of routing protocols. Furthermore, the module implements IP Tables and provides internal tunnel devices (similar to Linux's tunnel device) to support IP in IP encapsulation. There is no Internet Control Message Protocol (ICMP) implemented.

The support of DNS and DHCP has been added to make the scenario configuration as easy as possible. Higher layers address their traffic streams by using domain names. Tedious IP address mangling is not needed. The DHCP sub-module takes care of address allocation and also automatically updates lookup tables within the DNS service.

UDP and TCP models with accurate UDP and TCP headers are available. The congestion avoidance and slow start algorithms have been implemented as strategies and can be exchanged by configuration. Currently Tahoe and Reno are available.

One very beneficial feature of the TCP/IP modules is their capability to write Wireshark [24] compatible trace files. In this way the powerful network analysis tool can be used to visulaize protocol behaviour. There is also a TUN device available that actually connects the simulator to the operating system, allowing for live captures during the simulation run.

E. Traffic Models

The openWNS load generator is named *Constanze*. Basically, it consists of *traffic generators* and *bindings*. Traffic generators create packets while the binding ties the generator to a specific lower layer. Within openWNS it is possible to connect the traffic generator either to the data link layer, network layer or transport layer depending on the scenario. The traffic models you can choose from are:

- Simplistic Point Process (PP) models including Constant Bitrate, Poisson distributed traffic or the more generic version that allow for arbitrary random distributions for both packet inter-arrival time and packet size.
- Markov-Modulated Poisson Process (MMPP) models. The IMT-Advance VoIP model [17] or variable bit-rate models like MPEG2.
- Autoregressive Moving Average (ARMA) models. These are typically used to model variable bit rate video or ATM traffic but have also been applied to model online game traffic.

Constanze's traffic generator bindings take care of adapting the traffic source and sink to the desired protocol layer. Traffic sinks record throughput and delay statistics and are called *listener bindings*. Generators can be bound to the

- Data Link Layer (DLL). In this case the binding is aware of the MAC address of source and sink and it injects the generated packets accordingly into the protocol stack.
- Network Layer (IP). openWNS uses IP as its network layer. The IP binding is similar to the DLL binding but uses IP-Addresses instead of MAC addresses.
- Transport Layer (TCP, UDP). The UDP binding additionally is aware of the destination port. The TCP binding is responsible to open and close a connection before transmitting any packets.

This structure of the traffic generator module makes its usage very simple. The traffic source characteristics are configured completely separate from the deployment within the simulation scenario. Sources can be plugged on any layer and traffic routing can be decided individually per generator instance.

VI. Cluster Computing Support

One of the most advanced features of the openWNS simulation platform is its support for cluster computing. During the

development phase, compilation cycles can be significantly accelerated by employing a compile cluster. openWNS supports *icecc* out of the box.

Even more important is the support during the *simulation phase*, parallelizing whole simulation campaigns, which consists of multiple simulation runs, each simulation run with different parameter sets is performed on a single processor. Many simulation tools do not offer support for this and leave the implementation of collecting results, extraction of measurements and parameter plots to the user. openWNS offers the *Wrowser* (an acronym for *W*ireless network simulator *R*esult B*rowser*) which solves this problem and lets users focus on the research rather than on the scripts that collect their measurements.

The approach taken by *Wrowser* is illustrated in Figure 4. *Wrowser* supports Sun Grid Engine and Postgresql databases as cluster and database backends. The starting point for running a simulation campaign (i.e. parameter sweeps) and analyzing the results is a scenario configuration file. This file is augmented by the user with definitions of the parameters that should be altered between different parallel simulation runs on the cluster, e.g. one could define to increase the offered traffic from 0 to 30 Mbit/s in steps of 1 Mbit/s and for all of these load settings set the packet sizes to 80 byte and 1480 byte.

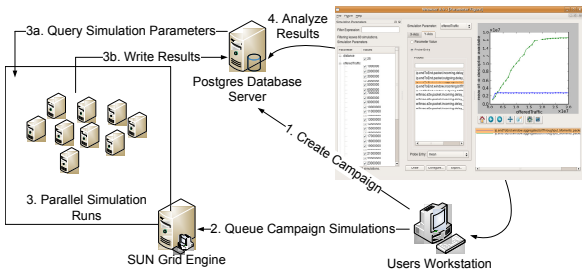

Fig. 4. Wrowser

Once this is done settings are written to the database and simulation directories are prepared. Users queue simulation runs and wait for the them to finish. Once a job executes simulation parameters are retrieved from the database and results are written back for further study.

As soon as the first results have been written to the database the graphical frontend of Wrowser can be used to access the results. Wrowser is aware of all the simulation parameters and parameter plots can be generated within a few steps. Figure 4 shows a plot of the carried traffic over the offered traffic for different packet sizes within a WiFi system.

VII. Conclusion

A new open source wireless network simulator was presented. Simulation modules for the physical, data link, network and transport layer have been released and are available to the public. The physical layer provides detailed channel models with support for directional and beamforming antennas, as well as frequency selective and time correlated channels. Furthermore, models for IEEE 802.16m and IEEE 802.11 with draft-n and mesh support have been released. An extensible TCP/IP suite and the traffic generator close the list of available modules.

The whole simulation platform is highly modular and offers users different extension points. The graphical support to view simulator configurations and results from multiple parallel simulation runs significantly simplifies the data evaluation process.

Acknowledgment

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 Ralf Pabst, Arif Otyakmaz, Klaus Sambale, Sebastian Max, Rainer Schoenen, Ralf Jennen 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.

References

[1] openWNS - open Wireless Network Simulator. Web Page. Department of Communication Networks, RWTH Aachen University. [Online]. Available: http://www.openwns.org

[2] 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*, Los Angeles, U.S., Nov 2005, p. 6. [Online]. Available: http://www.comnets.rwth-aachen.de/typo3conf/ext/cn_download/pi1/passdownload.php?downloaddata=824|1

[3] M. Schinnenburg, R. Pabst, K. Klagges, and B. Walke, "A Software Architecture for Modular Implementation of Adaptive Protocol Stacks," in *MMBnet Workshop*, Hamburg, Germany, Sep 2007, pp. 94–103. [Online]. Available: http://www.informatik.uni-hamburg.de/bib/medoc/B-281-07.pdf

[4] ns-2 web presence. Web Page. [Online]. Available: http://www.isi.edu/nsnam/ns/

[5] ns-3 web presence. Web Page. [Online]. Available: http://www.nsnam.org/

[6] OMNeT++ web presence. Web Page. [Online]. Available: http://www.omnetpp.org/

[7] OPNET web presence. Web Page. [Online]. Available: http://www.opnet.com/

[8] M. Lbbers and D. Willkomm. A mobility framework for omnet++. User Manual (online). [Online]. Available: http://mobility-fw.sourceforge.net/manual/index.html

[9] Mixim framework. Website. [Online]. Available: http://mixim.sourceforge.net/

[10] WiMAX (IEEE 802.16) Specialized Model. OPNET Technologies, Inc. [Online]. Available: http://www.opnet.com/solutions/brochures/wimax_model.pdf

[11] Boost C++ Libraries. web page. [Online]. Available: http://www.boost.org/

[12] *TR19768 Technical Report on C++ Library Extensions*, ISO/IEC Std.

[13] B. Karlsson, *Beyond the C++ Standard Library - An Introduction to Boost*. Addison Wesley, 2005.

[14] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Trans. Model. Comput. Simul.*, vol. 8, no. 1, pp. 3–30, 1998.

[15] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-oriented Software Architecture Volume 1*. John Wiley & Sons, 1996.

[16] L. Godara, "Application of antenna arrays to mobile communications. II. Beam-forming and direction-of-arrival considerations," in *Proceedings of the IEEE*, vol. 85, no. 8, 1997, pp. 1195–1245.

[17] ITU-R, "ITU-R M.2135 : Guidelines for evaluation of radio interface technologies for IMT-Advanced," ITU, Tech. Rep., 2008.

[18] Z. Wang, E. Tameh, and A. Nix, "Joint Shadowing Process in Urban Peer-to-Peer Radio Channels," *Vehicular Technology, IEEE Transactions on*, vol. 57, no. 1, pp. 52–64, Jan 2008.

[19] B. Walke, *Mobile Radio Networks*. John Wiley & Sons, November 2001.

[20] W. C. Jakes, *Microwave Mobile Communications*, W. C. Jakes, Ed. Wiley & Sons, 1975.

[21] B. Sklar, "Rayleigh fading channels in mobile digital communication systems. I. Characterization," *IEEE Communications Magazine*, vol. 35, no. 9, pp. 136–146, Sept. 1997.

[22] 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*, pp. 159–, 2002.

[23] S. Max, E. Weiss, G. R. Hiertz, and B. Walke, "Capacity bounds of deployment concepts for wireless mesh networks," *Performance Evaluation*, vol. 66, no. 3-5, pp. 272 – 286, 2009, modeling and Analysis of Wireless Networks: Selected Papers from MSWiM 2007. [Online]. Available: http://www.sciencedirect.com/science/article/B6V13-4TT30PH-1/2/3f79cd7cc92acbd90e83cd0fdb273c53

[24] Wireshark. Web Page. [Online]. Available: http://www.wireshark.org/