appeared in Proc. of 2nd Intl. OMNeT++ Workshop, pp. 61-70, Berlin, Germany, January 2002

A HiperLAN/2 simulation model in OMNeT++

D. Hollos[1] and H. Karl[1]

Telecommunication Networks Group
Technical University Berlin
Berlin, Germany
hollos@ee.tu-berlin.de, karl@ee.tu-berlin.de
http://www-tkn.ee.tu-berlin.de

Abstract. This paper describes the implementation of an OMNeT++-based HiperLAN/2 simulation model. The simulation models the HiperLAN/2 standard with a large degree of accuracy, in particular, in the DLC layer. It is designed to be extendable and flexible and incorporates HiperLAN/2 protocol extensions for multihop relaying. It is publicly available.

1 Introduction

Wireless communication has received increasing interest in the recent year. The wireless local area networks (WLANs) market is currently dominated by IEEE 802.11-based products [1]. Advantages of IEEE 802.11 networks are simple setup and reasonable performance in typical network conditions. A possible competitor to IEEE 802.11 is the HiperLAN/2 standard [2]. It promises higher data rates than the current IEEE 802.11 networks, reaching up to 54 MBit/s in optimal circumstances. A number of differences between IEEE 802.11 and H/2 lend credibility to this claim, in particular, the use of a centralized medium access scheduling as opposed to the distributed algorithms used in 802.11 (see Section 2 for more details). This control is performed by a so-called "central controller", which is responsible to orchestrate the communication between a number of terminals, together forming a HiperLAN/2 cell. In an infrastructure-based setup, the central controller would usually be the base station (as it is usually connected to a fixed power supply), in an ad-hoc network, any terminal can be elected to perform these control functions.

The use of such a centrally controlled medium access opens possibilities for additional protocol extensions. One such extension is the introduction of relaying capabilities into HiperLAN/2, using intermediate terminals to communicate between the central controller and another terminal, allowing to reduce the radiated power used by all terminals. The goal of such relaying protocols would be to improve the total capacity of a wireless network by reducing interference [3, 4] as well as to increase the energy efficiency of the communication (which is important as mobile devices usually only have a limited power supply). We are pursuing both these goals in the context of the IBMS[2] project [5].

In order to assess such claims, as well as to experiment with such extensions to and modifications of the H/2 standard, a simulation model of the standard is needed that allows to efficiently evaluate the performance of a system. Such a model must be

extensible, modular, easy to understand, and fast. In the context of the IBMS^2 research project we have developed such a simulation model. The essential characteristics of this model are described in this paper; for more information (and for the source code) please refer to the simulator homepage [6].

The following section gives a brief overview of H/2. In Section 3, the simulation model itself is described. Section 4 summarizes our contribution and discusses possibilities for future work.

2 The HiperLAN/2 System

The HiperLAN/2 (H/2) WLAN system standardized for the 5.2 GHz range with modulation types of up to 54 Mbit/s. The goal of this standard is to maximize the utilization of the radio channel. The main mechanism — and the largest difference to IEEE 802.11 — is the use of a a priori scheduled medium access: Among a set of stations (a so-called cell), one station is declared to be the central controller (CC). Any station that wants to communicate with another station has to announce this to the CC, which will grant time slots in a periodically repeated frame to this station. Hence, H/2 uses a connection-oriented, centrally scheduled TDMA to organize the medium access. Main benefits are collision-free data traffic (100% efficiency, no hidden or exposed terminal problems) and simple support for priorities or QoS requirements.

Scheduling happens on the basis of a frame, which is divided into different phases (compare Figure 1), which are again divided into cells of fixed length. In the first phase, the central controller broadcasts administrative information (BCH and FCH), in particular, which terminal is allowed to transmit to whom at what time and for how long. As this information is available to all terminals, perfect channel access can be organized. While this idea is very simple, it is the source of performance and flexibility of H/2.

Fig. 1. H/2 MAC frame structure

One example of this flexibility is the so-called direct-link traffic: As it is possible for the CC to assign a time slot to one terminal as a sender, to another terminal as a receiver, terminals can communicate directly with each other without having to send the data via the CC. Usually, the sequence of transmissions is organized such that after the frame's initial administration phase, downlink traffic is scheduled, then direct-link traffic, and uplink traffic; there is also a random access time-interval at the end of the MAC frame for associations and other infrequent, unscheduled requests (resulting in five phases in total).

Using this basic mechanism, a number of additional capabilities are included in the H/2 standard (e.g., automatic frequency selection, multicast, automatic CC selection, etc.). Also, the CC can request channel measurements from any terminal, describing

the channel characteristics between any two terminals (which is important for multi-hop extensions of H/2, see Section 4, and for the parallel transmission extension of the CC scheduler, see Section 3.2). Other possible add-ons are sleep modes for terminals or the interconnection of separate H/2 cells [7].

3 Features and Possibilities

Main goals of the simulator design were flexibility and extensibility as well as a correct capturing of reality: No "incorrect" information transfer or any other methods which are possible in a simulation environment, but not possible according to real life or the standard, are used. The following Subsection 3.1 describes the simulation of standard H/2 functionality, Subsection 3.2 exemplifies the simulator's extensibility by outlining some functions we have introduced in the course of our multi-hop research.

3.1 Simulator Structure

This simulator was made for research purposes; we payed attention to separate different parts, and tried to keep it open to future developments. Therefore, all major functionalities have their own separate modules; these modules communicate with each other using OMNeT++ messages.

The main configuration file (omnetpp.ini) defines the number of cells, number of terminals, the data source description file and the initial location file.

The initial location file contains the identifiers (MAC_IDs) and initial positions of all terminals. Note that some of them can be turned off and later on, so this number should be the maximum number of the terminals. The positions are "initial" as it is possible that a terminal moves during a simulation; however, mobility is currently not yet supported.

In the following subsections we give a brief description of all modules included in the simulator, its structure is outlined in Figure 2. As the primary focus of our own research is on the link and routing layer, these functions are modeled with the largest degree of detail; lower layers are comparably simple, but it is easy to replace these layers by more sophisticated modules.

The lowest layer in our simulator is an abstraction of the radio channel. The basic idea is to represent the physical layer along with its error behavior in a single module. All terminals are grouped into cells (compound modules), which are in turn connected to the channel module. An example of such a layout can be seen in Figure 3; Figure 4 shows how a radio cell consists of a central controller and a number of terminals.

A terminal is turn again structured as a compound module, consisting of a load generation module (also used as a data sink), the network stack as such and a control module (Figure 5). The network stack is the last compound module used in the simulator, it contains simple modules for the datalink, network and transport protocol layers (Figure 6).

The Configuration Distributor This module is always called first when starting a simulation, and later any time when the configuration changes. For example, a terminal can

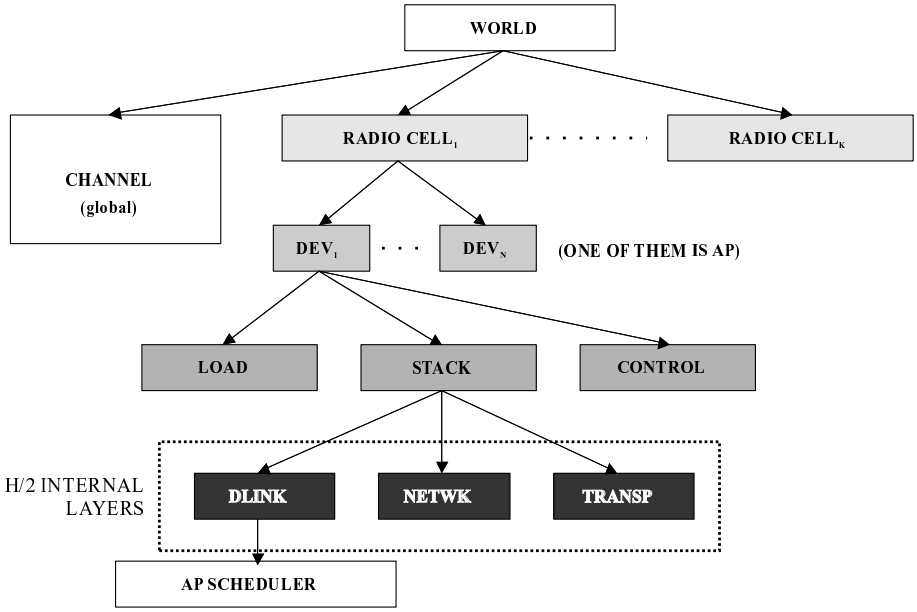

Fig. 2. Simulator structure

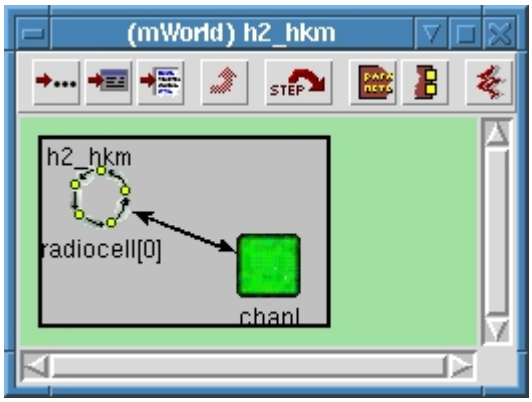

Fig. 3. Screenshot of the physical channel and a single radio cell

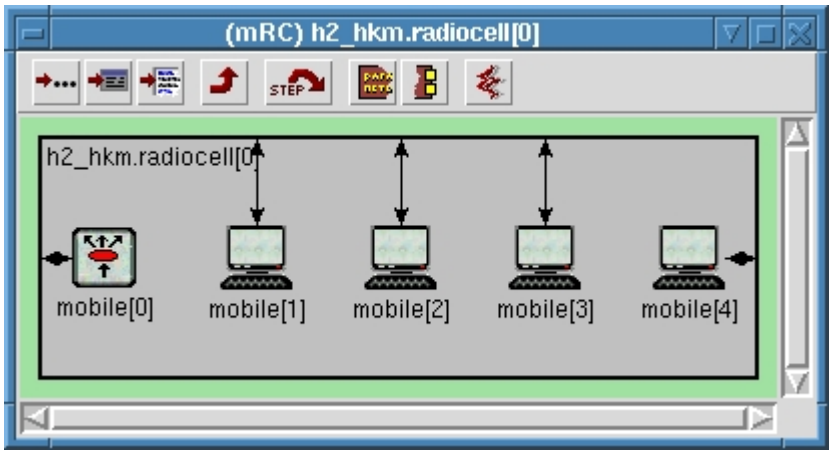

Fig. 4. Screenshot of a radio cell compound module, containing a single controller (mobile[0]) and four terminals

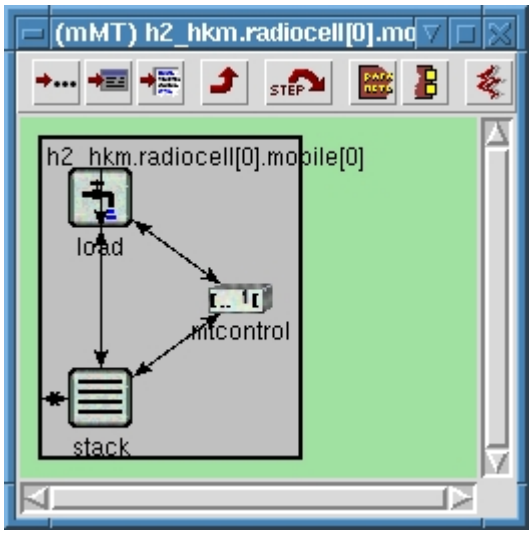

Fig. 5. Screenshot of a terminal compound module

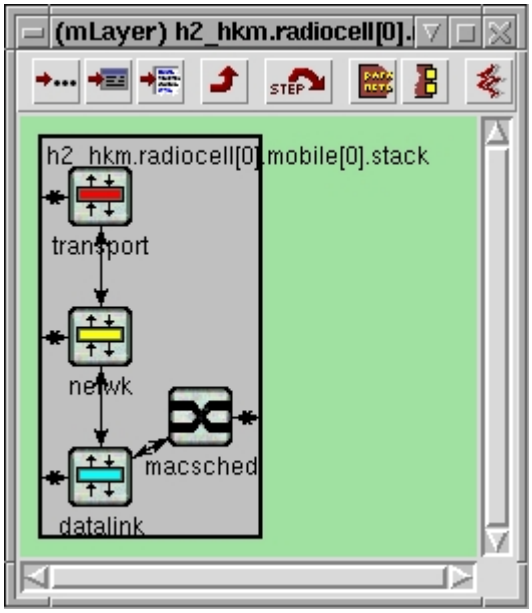

Fig. 6. Screenshot of a stack compound module, showing simple modules for the individual network layers.

be turned into a central controller by calling a function in this module. It is connected to all layers and distributes the new state to them.

The Physical Medium Simulation Module The terminals send their data to the Physical Medium Simulator Module called “channel” (common for the entire scenario). This module is responsible for determining whether or not a transmission succeeds or a packet is lost. To do so, this module collects packet transmissions and calculates the delay to all other terminals. For each destination terminal, the channel gain and the interference level is then calculated. Based on this signal to interference/noise ratio, a packet error rate (PER) can be calculated [2]. The PER is used to decide (with a simple Bernoulli random variable) whether the packet is correctly received by the terminal; if not, an error indication is sent to the terminal (akin to a failed CRC check). In case the received power is below the receiver sensitivity, the packet will not be delivered at all.

Each packet will also increase the interference of the non-destination terminals as determined by the respective channel gain. As a simplification, when calculating the PER (Packet Error Rate) we use the highest interference level determined during that particular packet.

The data collection, parallel interference calculation and all other physical-layer related calculations are put to a subclass of the channel module. Therefore, applying a more sophisticated physical layer model is a matter of replacing the channel subclass of this module.

The Physical Layer This is a very simple layer, one instance per H/2 device, which adds the basic H/2 preambles to the MAC packets.

The channel module always needs some parameters which are given in real life, but not in the simulation: for example, the transmission power, the modulation type to use, terminal ID (for simulation administrative reasons), etc. These parameters are also set by this layer; they have a separate, unique structure in each message type.

The DLC Layer The data link control (DLC) layer contains the basic data sending and reception functionalities, like handling data initiated by the load generator, constructing resource requests, decoding the administrative information received at the beginning of a frame (BCH and FCH cells), sending the data at the scheduled time, etc.

The DLC layer (outlined in Figure 7) basically contains two major subclasses: APControl and MTControl. They are for the different AP and MT functionality, but as a subclass they contain MTInstance class(es). An MTControl has only one MT subclass, the APControl has one for each associated MTs and one for itself. Changing a common MT functionality is a matter of changing a function in the MTInstance class. The MTInstance classes contain several DLCC queues: send, received, sending_in_this_frame, received_in_this_frame. When an MTInstance receives data from the FCH which is valid for it, allocates space in one of its _in_this_frame DLCC queue immediately. This structure reflects reality, since MTs have time after the FCH to do administrative procedures, but not later when receiving by 54Mbit/s rate. The DLC layer also has the CC scheduler as a subclass, which is activated only when the node is indeed a CC (see Section 3.2).

The CC scheduler proceeds in two steps, implemented in two separate classes: one for handling priorities, QoS demands, parallel transmission support, etc., computing a schedule describing when each terminal is allowed to transmit (`Compute_Schedule`). The second module constructs the MAC frame based on this schedule, keeping the H/2 rules defined in the standard (`Build_FCH`). Thanks to this structure, new scheduling algorithms (priorization, parallel transmission, or any other optimization method) can be applied without having to worry about complex H/2 MAC frame structure rules.

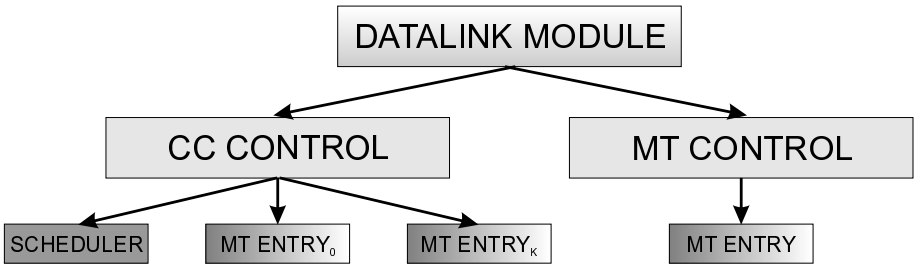

Fig. 7. DLC layer structure

All nodes have the same DLC, RLC, etc. modules.

The Network Layer This is an internal network layer, part of the H/2 RLC; it has the database with DLCC connections for that terminal. When a data packet comes from the load generator, this layer translates source- and destination MAC_IDs to streams with DLCC_IDs; when data has been received by the DLC layer, it translates DLCC_IDs to source- and destination MAC_IDs.

Generating Load Currently, two simple load generation methods are implemented: Constant bitrate (CBR), and Poisson process (PPS). The load generation parameters are contained in a file (which is specified in omnetpp.ini). In CBR mode the start- and end time, the bitrate of the data flow, and the DLCC_ID can be specified. In PPS the starting- and ending time, and the parameter λ of the arrival process can be specified. The system generates as many load modules per terminals as were described in the data sources description file. Other load generators can be easily integrated.

3.2 Extra Features

Here we describe the features we added to the simulation and are not present in the standard.

The Parallel Transmission CC Scheduler Since the H/2 provides direct, scheduled data transmissions between terminals in a cell, and the CC has all information to decide whether a parallel transmission is possible or not (based on channel measurements provided by the terminals, see Section 2), we developed a scheduler class (extending the one described in Section 3.1) which calculates possible parallel transmission pairs based on a graph coloring algorithm. Using this class the `Build_FCH` class is able to schedule multiple transmissions at the same time.

The Single Relay Protocol When a terminal is located on the edge of the cell, it has to use its highest transmission power to reach the CC. The single relay protocol enables the system to use one relay station for a transmission between the CC and a "far" terminal. The protocol fits into the H/2 standard and allows dynamic route path change even in every MAC frame without additional protocol overhead. In this case, a terminal located far away will use less transmission power (exploiting non-linear channel characteristics); it also dramatically reduces the inter-cell interference. The protocol realization itself is included in the CC scheduler `Build_FCH` class; the path calculation has to be done in the class `Compute_Schedule`. This work is part of our ongoing research in the context of the IBMS^2 project [5].

4 Conclusions

This paper has described an H/2 simulator, capturing essential parts of H/2's medium access and link layer functionalities. Its design lends itself to simple modifications and to introduction of new functionalities, which has been demonstrated by introducing a

single-relay protocol and non-standard cell schedulers. We believe that this simulator could be useful to a broader community of researchers.

Currently, we are developing an IP convergence layer on top of the actual H/2 standard in order to be able to use realistic traffic models based on IP traffic for performance evaluation. This convergence layer will be integrated with the TCP/IP suite available for OMNeT++ [8]. In the long run, we are interested in incorporating true ad-hoc functionality such as dynamic CC selection, terminal handover between different cells and inter-cell connection support (H/2 multihop extension). Also, suitable mobility models should be integrated in the simulator. It will be a challenging task to take into account mobility-induced handovers between different CCs, as these are represented by compound modules. It would imply to remove a compound module representing a node from a cell compound module and reinserting it into another cell module in order to maintain the general structure of the simulation setup. In addition, this process has to closely coordinated with the protocol-specific processing of handover. It would be an interesting question to see if such functionalities are generally useful to the modeling of mobile ad-hoc networks; examples for such a generalization would be cluster-based routing protocols for ad-hoc networks, which also superimpose a cell-like structure upon an ad-hoc network. A convenient handling of such problems would be useful to OMNeT++-based simulations of ad-hoc networks.

More specifically to the OMNeT++ tool, a particular problem that we encountered was the simulation of the physical layer. Taking into account interference generated by other terminals as well as noise to compute packet error rates proved to be feasible only by delegating these tasks to a separate module which is responsible for the entire simulation setup; using OMNeT++ channel abstraction did not turn out to be useful. In general, the development of more sophisticated and customizable channel abstraction would prove rather useful in many respects. Besides the notion of simulation-wide (or cell-wide) computation of interference discussed here, also bit error models might prove to be useful, e.g., models which can express autocorrelation within the bit error process [9].

References

1. : Ieee 802.11 (iso/iec 8802-11:1999). IEEE Standards for Information Technology (1999) Available via `http://standards.ieee.org/getieee802/`.
2. Khun-Jush, J., Malmgren, G., Schramm, P., Torsner, J.: HIPERLAN type 2 for broadband wireless communication. Ericsson Review **2** (2000) 108–119 `http://www.ericsson.com/review/2000_02/files/2000026.pdf`.
3. Karl, H., Mengesha, S.: Analyzing capacity improvements in wireless networks by relaying. In: Proc. IEEE Intl. Conf. Wireless LANs and Home Networks, Singapore (2001) 339–348
4. Mengesha, S., Karl, H., Wolisz, A.: Improving goodput by relaying in transmission-power-limited wireless systems. In: Proc. of Informatik 2001-31. Jahrestagung der GI, OCG, Austria, Vienna (2001)
5. : Webpage of the IBMS^2 project. (http://www.ibms-2.de)
6. Hollos, D., Karl, H., Kubisch, M., Mengesha, S.: Hiperlan/2 simulator homepage. `http://www-tkn.ee.tu-berlin.de/research/H2Simulator/` (2001)
7. Peetz, J.: HiperLAN/2 multihop ad hoc communication by multiple-frequency forwarding. In: Proc. of Vehicular Technology Conference (VTC) Spring 2001, Rhodes, Greece (2001)

8. : Omnet++ internet protocol suite. (Webpage at http://cvs-int.etec.uni-karlsruhe.de/omnetpp/model-doc/ip-suite.html)
9. Willig, A., Kubisch, M., Wolisz, A.: Measurements and stochastic modeling of a wireless link in an industrial environment. Technical Report TKN-01-001, Telecommunication Networks Group, Technische Universität Berlin (2001)