IEEE 802.11a System Simulator

versio 1.6

André Noll Barreto

December 6, 2010

Chapter 1

Introduction

This document describes a discrete-event system level simulator for an IEEE802.11a wireless LAN system.

Chapter 2

Usage

The program can be called by executing the file `SysSim.exe`. The simulation can only be carried out if a configuration file `config.txt` is available. The directory *dir* where this file can be found can be given in the command line by calling: `SysSim -d`*dir*

The results obtained from the simulation will be stored in the same directory. If no working directory is given, the default directory `.\Data` is assumed.

After finishing the simulation, the program pauses and ask the user to press any key before exiting. If the command-line parameter: `-no_pause` is used, this does not occur, which is useful for running batches with different configuration files.

The configuration, results and log files are described below.

2.1 The configuration file

The configuration file `config.txt` contains the simulation parameters. Each line in the file defines one parameter, blank lines are ignored. All the characters in a line following a `%` are not considered either. All the parameters are defined the following way: *parameter_name* = *parameter_value*

If any parameter is not defined in the configuration file, a default value is taken. Parameters can be defined at any particular order.

2.1.1 simulation control parameters

The basic simulation control parameters are described following, with the respective default values:

- `MaxSimTime = 0`. Each simulation run will simulate `MaxSimTime` seconds of system operation.
- `TempOutputInterval = inf`. Partial results will be written on the default output every `TempOutputInterval` seconds of simulation time.
- `TransientTime = 0`. The first `TransientTime` seconds of simulation time are ignored when collecting results, in order to account for transient effects.

- `Confidence = .95`. If more than one seed is given, some results are given including a confidence interval with given confidence level.

- `Log = `. If desired, the main simulation events can be recorded at the file `sim.txt`. The following different categories of event can be logged: `SETUP, MAC, PHY, CHANNEL, ADAPT`(link adaptation) and `TRAFFIC`. A combination of categories can also be chosen, e.g., `Log = SETUP,MAC`.

Apart from the above, all other parameters accept comma-separated multiple values for several simulation iterations, i.e., with a single program execution many simulations can be performed using different parameter values. Simulation iterations are performed for all possible parameter combinations. The outermost iteration loop corresponds to the first defined parameter, the innermost loop is always over the seeds. For instance, if the following parameters (to be described later) are defined in the configuration file:

```
Seed = 1,7
NumberStas = 2,4
Radius = 50,80
```

eight different simulation runs will be performed with the following parameters:

1. `NumberStas = 2; Radius = 50; Seed = 1;`
2. `NumberStas = 2; Radius = 50; Seed = 7;`
3. `NumberStas = 2; Radius = 80; Seed = 1;`
4. `NumberStas = 2; Radius = 80; Seed = 7;`
5. `NumberStas = 4; Radius = 50; Seed = 1;`
6. `NumberStas = 4; Radius = 50; Seed = 7;`
7. `NumberStas = 4; Radius = 80; Seed = 1;`
8. `NumberStas = 4; Radius = 80; Seed = 7;`

The seed parameter means:

- `Seed = 1`. Seed for random number generation. Simulations with the same seed use the same terminals and have the same channel characteristics over time, thus allowing a better comparison among results. The final results are averaged over all different seeds.

Besides the simulation control parameters explained above, the following parameters define the simulation model.

2.1.2 physical configuration parameters

We start by the physical configuration of the communications system:

- `NumberAPs = 1`. Number of access points in the system.

- `NumberStas = 1`. Number of mobile terminals in the system.

- `Radius = 10`. System radius in metres. The mobile terminals are randomly positioned with uniform distribution within a circle of given radius. If only one mobile station is defined, then it is positioned at a fixed location `(Radius,0)` m.
- `APPosition_`n `= (0,0)`. The n-th access point is located at the given position (2-D coordinates) in metres.

The numbering of the stations start with 0, and different counters are used for the Mobile Stations and for the Access Points.

2.1.3 channel parameters

The wireless channel is defined by the following parameters:

- `LossExponent = 3.0`. Path loss exponent.
- `RefLoss_dB = 46.7`. Reference path loss at 1 m (default value according to Friis equation [**?**]).
- `DopplerSpread_Hz = 5.0`. Maximum Doppler spread in Hz.
- `NumberSinus = 20`. Number of sinewaves to emulate Rayleigh fading (see Section **??**). It should be > 10 for good statistical properties.

2.1.4 physical layer parameters

The received signal quality depends on the following physical layer parameters:

- `NoiseVariance_dBm = -95.0`. Noise variance at receiver in dBm.
- `CCASensitivity_dBm = -98.0`. Carrier sensitivity level. Packets with receive power below this level are ignored.

2.1.5 link adaptation parameters

The packet error probability depends also on the transmitted power and on the data rate, which are defined by a link adaptation algorithm. The link adaptation parameters are:

- `TxMode = M6`. This parameter defines the link adaptation approach to be used. It can be any of the following:
  - `M6, M9, M12, M18, M24, M36, M48` or `M54`. No link adaptation is employed, always transmit at fixed rate with maximum power.
  - `OPT`. Genie-aided link adaptation, assume ideal channel knowledge at transmitter.
  - `SUBOPT`. Transmitter-based link adaptation algorithm. Use information inherently available at transmitter MAC layer to adapt link parameters. See description in Section **??**.

- `AdaptMode = RATE`. Defines whether link adaptation algorithm should maximise the transmitted data rate using always the maximum power, or whether it should minimise the transmitted power, transmitting at the lowest rate if necessary. The parameter can be respectively either `RATE` or `POWER`

- `TxPowerMax_dBm = 0`. Maximum transmit power in dBm.

The other link adaptation parameters are only relevant depending on the link adaptation algorithm chosen. If the power should be minimised (`AdaptMode = POWER`) and the link parameters are not fixed (`TxMode` $\neq$ `M*`) then the following power adaptation parameters can be given:

- `TxPowerStepUp_dBm = 1`. Power adaptation is done at this step when increasing the power.

- `TxPowerStepDown_dBm = 1`. Power adaptation is done at this step when decreasing the power.

- `TxPowerMin_dBm = 0`. Minimum transmit power in dBm.

If the genie-aided method was chosen (`TxMode = OPT`), then the following parameter can be given:

- `TargetPER = 0.1`. Highest rate or lowest power are chosen such that the packet error rate is less than or equal to `targetPER`.

If the transmitter-based adaptation algorithm is employed (`TxMode = SUBOPT`), it will use the following parameters:

- `LAMaxSucceedCounter = 0`. The transmitter stores in a counter the number of consecutive transmission successes (ACKs received) using the same data rate or power level. If this counter is equal to the threshold `LAMaxSucceedCounter`, then the transmission data rate is increased (or the power level decreased). If `LAMaxSucceedCounter` is equal to zero then the threshold is chosen automatically (see **??**).

- `LAFailLimit = 1`. The transmitter stores in a counter the number of consecutive transmission failures (ACKs timed out) using the same data rate or power level. If this counter is equal to the threshold `LAFailLimit`, then the transmission data rate is decreased (or the power level increased).

- `UseRxMode = 0`. It is a boolean parameter. If it is set, then then employ same transmission rate (if higher than current rate) as correctly received packets (only if `AdaptMode = RATE`)

2.1.6 traffic parameters

We can define several types of traffic models and associate each model to different flows between terminals. Traffic models are bidirectional. Traffic parameters must be numbered (starting from zero), indicating to which model they refer.

- `TrafficType_`n `= POISSON`. It defines the type of the n-th traffic model. It can be any one of the following:

  - `POISSON`. Packet interarrival time is exponentially distributed.
  - `CBR`. Packet interarrival time is constant.
  - `VOICE`. Models a speech connection, with active and inactive phases. Packets with constant interarrival time are generated in active intervals, no packets are generated otherwise.

- `PacketLength_`n `= 1000`. Number of data bytes in a packet. Variable-length packets can be considered by giving several lengths with the corresponding probabilities, e.g., `250(.4);1000(.3);2000(.3)`.

- `DataRate_`n `= 0.5`. Offered load in Mbps per link. In case of `VOICE` traffic, this is the data rate in active mode.

- `DownlinkFactor_`n `= 1`. The offered downstream load is `DataRate * DownlinkFactor`. The term downstream refers to the direction from the first and the second terminal in a flow (see definition of `Flows` below). This and the following parameter allow for bidirectional data streams.

- `UplinkFactor_`n `= 1`. The offered upstream load is `DataRate * UplinkFactor`

- `TID_`n `= 0`. Traffic ID.

- `Flows_`n `= AP0-MS0`. It contains a list of flows between different terminals, for which this traffic model is activated. a flow can be defined as in the following example: `AP0-MS1`, which means that this flow is activated from Access Point `AP0` to Mobile Station `MS1` and vice versa (duplex). If a terminal does not exist, this is ignored. Several traffic flows may be associated to a traffic model, and they are separated by "/", e.g., `AP0-MS1/AP0-MS2/AP1-MS3`.

In a centralised network having at least one Access Point all flows to a given Mobile Station must belong to the same connection. In an ad-hoc network, each Mobile Stations can be connected to several others.

If no flows are defined for a given traffic model, then this flow applies to all MSs, using a standard configuration, which depends on the number of Access Points, which is defined by parameter `NumberAPs` in Section **??**. If there are no of Access Points, then mobile stations form an ad-hoc network, where each station is connected to its nearest neighbour. Connections are independent and no routing is simulated. If there is at least one Access Point, each Mobile Station is associated to the nearest Access Point only.

2.1.7 MAC layer parameters

The MAC protocol to be used by all the terminals in simulation run is selected by:

- `WhichMAC = DCF`. Type of MAC protocol. It can be any of the following:
  - `DCF`. Distributed coordination function from IEEE 802.11 standards.
  - `EDCF`. Extended DCF from IEEE 802.11e draft.

For the DCF protocol the following parameters are relevant:

- `RTSThreshold = 10`. Packets with more than `RTSThreshold` bytes employ RTS/CTS.
- `FragmentationThreshold = 1500`. Maximum MPDU size, larger packets are fragmented.
- `QueueSize = 1000`. Size of MAC waiting queue.
- `LongRetryLimit = 10`. If packets are retransmitted `RetryLimit` times without success they are discarded. The `LongRetryLimit` applies only to long packets containing at least `RTSThreshold` bytes.
- `ShortRetryLimit = 10`. Same as above, but applies for short packets with less than `RTSThreshold` bytes.

If the EDCF protocol is selected,...

2.2 The results file

The simulation results are stored in the file `results.txt` in the same directory as the configuration file. Results are given for each simulation iteration, for each terminal and for each connection. Several result parameters are displayed for each connection, as in the example below:

```
Mobile Station 0( 50;  0): P(mW)=  4.2
    to Access Point 0, TID = 0, tp(Mbps)= 5.955, trt(ms)= 1318.74, s_trt(ms)=   57.80,
                       txt(ms)= 1.22, s_txt(ms)= 0.01, npck= 37143, PLR=0.0000, QOR=0.0000
```

This represents the performance results for the 0-th Mobile Station, located at position (50m,0m) in (x, y)-coordinates, on its connection to Access Point 0 with traffic identifier `TID = 0`. `tp` is the link throughput in Mbps. `trt` is the average transfer delay[1] and `txt` the average transmission delay[2]; `s_trt` and `s_txt` their standard deviations respectively, all in milliseconds. `npck` is the number of successfully transmitted, `PLR` the packet loss rate caused by the retry limit being reached, and `QOR` the queue overflow rate.

After the results for each simulation run is displayed, some performance results are averaged over all the terminal and all the different seeds, for each different simulation parameter set. If more than one seed is given, then the confidence interval is calculated as well. One example can be seen below:

```
%%%% Final results %%%%


cell radius    =        10         10         20         20         50         50 meters
transmit mode  =    6 Mbps    24 Mbps     6 Mbps    24 Mbps     6 Mbps    24 Mbps


Throughput (Mbps)
mean           =     5.246     16.696      5.206     15.826      4.606      6.939
conf. interval =     0.009      0.129      0.038      0.151      0.163      1.011
```

This means that the simulation was run for three different cell radii (10, 20 and 50m) and two different transmit modes (6 and 24 Mbps), whereas all the other parameters were kept fixed. For instance, with a cell radius of 20m and transmission at 24 Mbps the total throughput is 15.826 ± 0.151 Mbps with the given confidence (see Section**??**).

Besides the throughput, several other parameters are also given at the end.

[1] Transfer delay is the time it takes since the packet is generated until it is correctly received.

[2] Transmission delay is the transfer delay less the time spent in the MAC waiting queue.

2.3 The log file

Depending on the parameter `Log` in Section **??**, the main simulation events are logged into the file `log.txt`, located in the same directory as the configuration and the results file.

Chapter 3

Simulation Model

3.1 System configuration

The simulator consists of single-frequency wireless network of stationary nodes. Even though the terminals do not move, the channel is time-variant to account for changes in the propagation environment, as described in the next Section.

3.2 Channel

A frequency-flat channel is assumed, and the received power in dBm at time t is given by

$$P_{\mathrm{Rx}}(t) = P_{\mathrm{Tx}}(t) - R_{1\mathrm{m}} - 10L \log_{10} d + G(t),$$

where $P_{Tx}(t)$ is the transmit power in dBm, $R_{1\mathrm{m}}$ the reference path loss at 1 m distance in dB, d the distance in metres and L the path loss exponent.

G(t) is a time-variant Rayleigh-distributed fading gain for which the well-known Jakes' Doppler spectrum [**?**] is assumed. The fading value is updated at the beginning of each packet transmission, and it is assumed that the channel remains constant during any single packet transmission. The fading is implemented using Jakes' simulator model [**?**], in which the complex channel gain is obtained by the sum of a finite number of oscillators N_{OSC}. It was shown in [**?**] that this method provides a good approximation to the desired fading process if the number of oscillators $N_{\mathrm{OSC}} > 10$.

Each receiver is affected by white Gaussian noise with a given variance. Besides this, packets transmitted simultaneously interfere with each other, but in order to reduce the simulation complexity only the maximum interference level during the packet transmission is considered for the calculation of the error probability. The interference is also only considered at the terminal to which the packet is targeted.

3.3 Receiver

IEEE 802.11a employs a convolutional code with constraint length $K = 7$, rate $R = 1/2$ and different puncturing rates. Due to coding, bit error events are not independent, and we must consider that errors occur in bursts. If the bit error

Table 3.1: Modelling of the bit error rate

Mbps	T_1	$\log_{10}(P_e)$ if $T_1 \leq \gamma < T_2$	T_2	$\log_{10}(P_e)$ if $T_2 \leq \gamma$
6	−2.51	$-2.235 - 1.072\gamma - 0.171\gamma^2 + 0.024\gamma^3 + 0.00967\gamma^4$	1.99	$-2.397 - 1.158\gamma$
9	0.75	$-0.362 - 0.294\gamma - 0.0011\gamma^2 - 0.041\gamma^3 + 0.0038\gamma^4$	5.3	$2.825 - 1.482\gamma$
12	0.5	$-0.452 - 0.356\gamma + 0.0628\gamma^2 - 0.0651\gamma^3 + 0.00648\gamma^4$	5.0	$2.114 - 1.374\gamma$
18	3.8	$-0.308 - 0.206\gamma + 0.155\gamma^2 - 0.0390\gamma^3 + 0.00182\gamma^4$	8.3	$7.708 - 1.535\gamma$
24	5.5	$2.697 - 1.935\gamma + 0.474\gamma^2 - 0.0509\gamma^3 + 0.00162\gamma^4$	10.5	$9.258 - 1.324\gamma$
36	9.3	$34.8 - 13.91\gamma + 2.033\gamma^2 - 0.128\gamma^3 + 0.00285\gamma^4$	14.8	$11.38 - 1.10\gamma$
48	12.5	$93.9 - 26.7\gamma + 2.811\gamma^2 - 0.129\gamma^3 + 0.00214\gamma^4$	18.5	$14.65 - 1.045\gamma$
54	14.5	$-120.2 + 26.38\gamma - 2.156\gamma^2 + 0.0787\gamma^3 - 0.00112\gamma^4$	20.0	$20.07 - 1.228\gamma$

$P_e = 0.5$ for all modes if $\gamma < T_1$

rate BER is known, a good approximation to the packet error rate PER can be obtained as:

$$PER = 1 - (1 - \frac{BER}{b})^N, (2)$$

where N is the number of bits in the packet and b the expected error-burst length. The error probability is calculated assuming a flat-fading channel. The bit error rate as a function of the signal-to-noise-plus-interference ratio (SNIR) for all eight transmission modes was obtained through link-level simulation and approximated by a polynomial of 4th degree. The polynomials are given in Table **??** as a function of the SNIR value in dB γ. The value of b depends on the SNIR, but it has been observed through simulation that a value of $b = 3.3$ provides a good approximation over a large range of SNRs.

Packets received with a power below a given carrier sensitivity level are ignored.

3.4 MAC Layer

Only the distributed coordination function (DCF) is considered, as the point coordination function (PCF) is typically not implemented in commercial devices. A description of the DCF is beyond the scope of this manual and can be found for instance in [**?**]. Only data packets and some control frames (RTS, CTS, ACK) necessary for the DCF are considered.

The propagation and signal-processing delays are ignored.

3.5 Link Adaptation

In this simulator there are two link adaptation modes, consisting on the choice of either the optimum transmission data rate or the optimum transmit power. If the rate is to be maximised, then the transmit power is constant and equal to the maximum power defined in the configuration. If the power is to be minimised, then all the packets are transmitted at the lowest possible data rate of 6 Mbps, except if the minimum transmit power is reached, in which case the data rate is maximised for this power. The two following link-adaptation schemes are supported by the simulator.

3.5.1 Genie-aided

In order to assess the potential gains of link adaptation, a genie-aided scheme was implemented, i.e., one in that the transmitter knows perfectly the current channel conditions, and hence the error rate of any transmission. Based on this knowledge

3.5.2 ACK-based

If the transmitter does not receive an ACK for a data frame sent to a certain receiver, the link adaptation algorithm concludes that the quality of the link to that receiver has deteriorated and that therefore a lower transmission rate should be used for future transmissions to that receiver. On the other hand, if the transmitter succeeds to send multiple data frames to a certain receiver, it assumes that the quality of the link has improved and therefore a higher rate should be used for future transmissions.

3.6 Scheduler

The discrete-event scheduler operates with a resolution of 1μs. It is implemented as a 64-bit unsigned integer, such that a simulation time of up to $1.8447e^{19}\mu\text{s} \simeq$ 584942 years can be supported.

Nothing can be guaranteed about the processing order of discrete-events scheduled for the same time.

Chapter 4

Implementation

4.1 Overview

4.2 Tools used

Bloodshed freeware Dev-C++ (http://www.bloodshed.net) with Gnu compiler for windows Mingw (http://www.mingw.org) was used. The latest Dev-C++ version (5.0 beta5) is based upon gcc2.95.3 compiler. The source code should be portable without major problems to other C++ compilers.

Chapter 5

Classes

5.1 class PHY

This class implements the physical layer of IEEE 802.11a.

5.1.1 Usage

A **PHY** object must belong to a **Terminal** object. A **MAC** object must be assigned to the **PHY** through function *connect*.

A **MAC** object forwards a packet to the **PHY** using the function *send*. The packet is sent by **PHY** to the wireless channel.

The wireless channel delivers packets to **PHY** using function *receive*. Each one of these packets has a path loss and an interference level determined by the channel. **PHY** decides then if the packet is received correctly, and if this is the case forwards it to the **MAC** layer. The error model is based on bit error rate results obtained through simulations for an AWGN channel. The packet error rate is obtained by

$$PER = 1 - (1 - \frac{BER}{L})^N,$$

where N is the number of bits in a packet and L is the expected error-burst length, which is employed to approximate the coding effects.

5.1.2 Constructor

PHY(**Terminal*** t, **Position** p, **Channel*** c, **random*** r, **Scheduler*** s, **PHY_struct** ps)

- t: pointer to owner terminal.
- p: terminal position.
- c: pointer to wireless channel used.
- r: pointer to random number generator.
- s: pointer to simulation scheduler.
- ps: structure with physical layer parameters.

5.1.3 Member Functions

- **bool** *carrier_sensing*(): returns **true** if channel is sensed to be busy by **PHY**, i.e., if interference level is higher than the carrier sensitivity level; returns false otherwise.
- **void** *channeLoccupied*(**double** *interf*): receives message (from channel) that a new packet is being transmitted. Informs MAC if new interference is greater than sensitivity level.
- **void** *channeLreleased*(**double** *interf*): receives message (from channel) that a packet stopped being transmitted. Informs MAC if new interference is less than sensitivity level.
- **void** *connect*(**MAC*** *m*): associate **MAC** element *m* to this object.
- **unsigned** *getLid*() **const**: returns unique **PHY** identification number.
- **Position** *getLpos*() **const**: returns position.
- **void** *notify_busy_channel*(): MAC requests notification when channel becomes free.
- **void** *cancelLnotify_busy_channel()*: MAC cancels notification request.
- **void** *notify_free_channel*(): MAC request notification when channel becomes busy.
- **void** *cancelLnotify_free_channel*(): MAC cancels notification request.
- **void** *receive*(**Packet** *p*, **double** *pl*, **double** *interf* = 0): a packet *p* is received with path loss *pl* dBs and interference level *interf* mW. If packet is received correctly, forward it to MAC layer.
- **void** *send*(Packet p, bool to_all);
- transmission_mode opt_mode(Terminal* t1, unsigned pack_len, double per _target,double power);

5.2 Scheduler

The scheduler manages the discrete-event simulation.

The declarations are in "Scheduler.h" and the source code in "Scheduler.cpp".

5.2.1 class Event

It represents a basic discrete-time event. Each event consists of a time stamp, a function pointer, an optional object pointer in case of a member function, and an optional function parameter, which can be a pointer or a long_integer. Events can be active or inactive. Each event has a unique number.

Constructors

Member functions

- **deactivate** makes event inactive, i.e., action will no longer be performed.
- **get_id** and **get_time** return event number and time stamp respectively.
- **go** performs action defined in event, if active.
- **operators** <, >, <=, >=: events are compared based on their time stamp.

5.2.2 class Scheduler

A container of events. It is derived from a **priority_queue** using a **vector**. The events are sorted according to their time stamp.

Constructors

Member functions

5.3 Auxiliary libraries

5.3.1 Math

In header "math.h" one can find definitions for π, $\sqrt{2}$ and $1/\sqrt{2}$, as *M_PI*, *M_SQRT2* and *M_1_SQRT2*.

The *Bessel* function of 0-th order, which is needed for channel fading, is also defined in "math.h".

5.3.2 Random Number Generator

5.3.3

Bibliography

[1] T. Rappaport, *Wireless Communications: Principles and Practice*, Prentice-Hall, Upper Saddle River, NJ, USA, 1996.

[2] W. C. Jakes, Jr, *Microwave Mobile Communications*, Wiley, New York, 1974.

[3] M. Pätzold and F. Laue, "Statistical properties of Jakes' fading channel simulator," in *Proc. Vehicular Technology Conference (VTC)*, Ottawa, Canada, May 1998, pp. 712–718.

[4] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 3, pp. 535–547, Mar. 2000.