

An Event-Driven Wireless MAC Protocol Simulator

George R.J. Linnenbank, Paul J.M. Havinga

University of Twente, Department of Computer Science
P.O. box 217, 7500 AE Enschede, the Netherlands

{linnenba, havinga}@cs.utwente.nl

ABSTRACT

Recently, many multiple-access (MAC) protocols have been or are being proposed for wireless networks. As most of these multiple-access protocols are designed for specific applications (such as telephony) and analyzed accordingly, the analysis results can not always be adapted to situations where each user has a different behavior. Wireless MAC protocols for data communication are not straightforward to analyse. To quickly make a reliable judgement of the usability of a MAC protocol for specific situations, we designed a simulator that makes it simple to implement the protocol and test it in different configurations and with differently behaving users. Our simulator generates a large amount of quantitative performance information that can be processed with standard graph drawing tools and an integrated trace analyzer.

1 INTRODUCTION

Many multiple-access (MAC) protocols are designed for wireless mobile networks. Most of these MAC protocols are designed for a specific application. The majority of MAC protocols is designed for telephony. Analysis of these protocols is performed with parameters that are specific to telephony, such as the bit rate (usually 9.6, 16 or 64 Kbps), connect time, acceptable error rate, acceptable blocking probability, and so on. For data communication and video communication other parameters, such as peak rate, apply. Such traffic may change the behavior of a protocol completely.

Often protocols are analyzed mathematically to predict the performance. To make a mathematical analysis a mathematical model is needed. Some analysis methods, such as Markov chain analysis [1], are very good for the queuing models for which they were designed. However, Markov chain analysis assumes exponentially distributed packet¹ arrivals while many applications such as video are regular with relatively large gaps between video frames that consist of a large number of bytes. Also, it does not apply to situations where each station behaves differently from another. Furthermore, the mathematical analysis of complex traffic is hard to obtain. Finally, even a small behavioral modification of a protocol renders an invalid mathematical analysis.

This discussion indicates that a simulator is most suited for rapid evaluation of a wireless MAC protocol. Unfortunately, the many simulators that are available do not satisfy our needs. Most available simulators either require the model to be defined as a

queuing model or they are more suited to signal simulation having implemented a wide range of signal propagation properties such as reflections and multipath, fading, various types of noise models, etc. However, we are only interested in the performance of a wireless MAC protocol for mobile networks. Therefore, we have implemented a simulator that is dedicated to wireless mobile MAC protocols only.

1.1 Overview of the Simulator

In order to quickly judge the usability of a protocol for specific situations, we designed a simulator. It allows a simple implementation of the multiple-access protocol and tests it in various configurations where each individual user may behave differently. The simulator generates a large amount of quantitative performance information. Novel characteristics of our simulator are:

- It is dedicated to wireless communication with base stations and mobile stations.
- The ether is modeled modestly. Multipath, fading, etc. are not modeled. Effects are modeled by specifying bit-error rates.
- A new non-trivial protocol can be described and evaluated in one or two days.
- The simulated protocol is described in ANSI C and compiled into the simulator for speed. The evaluation of a wireless communication protocol is fast, often faster than real-time.
- A trace analyzer is integrated in the simulation environment.
- Extending the simulator (e.g. adding energy dissipation characteristics) takes only a few days.

We decided to implement the simulated environment according to the real world behaviour (we do not use abstract models). Since the simulator is completely implemented in ANSI C, the simulator is not only fast but also platform independent.

The simulator generates a vast amount of performance information of a MAC protocol. We can obtain estimates of the throughput, energy dissipation, access latency, etc. The results are text files that can be converted into graphs using a standard graph drawing utility. Furthermore, the simulator can create a trace file that can be visualized using a special displaying program. Using this trace file, the behaviour of a protocol can be analyzed and better understood, and errors can be debugged.

In this paper an overview of the design and implementation of the MAC simulator and analyzer is given. In Section 2, the requirements for a wireless MAC simulator are discussed. The architecture of the wireless MAC simulator is described in Section 3. Section 4 discusses the software environment of the simulator. Section 5 describes the implementation of the wireless MAC simulator and Section 6 gives the current status of the MAC simulator program.

1. In this paper we consider a packet to be a unit of application data and a message to be a unit of protocol information which may include application data (i.e. a data message is a data packet surrounded by protocol information).

2 SIMULATOR PROPERTIES

In this section we discuss the properties of our simulator for wireless MAC protocols. First we give the general properties of a simulator. Then we describe what properties are added by a MAC protocol simulator.

2.1 General Properties

To be generally useful, some properties are highly desirable for a simulator. These properties concern the level of detail, the reproducibility of the simulation results and the portability of the simulator program.

When simulating high level processes it is time consuming to simulate all the effects down to the lowest level in the system. Lower-level physical events that influence the performance of a MAC protocol but do not appear as such at the MAC protocol level must be modelled such that the effect of the event appears at the MAC level. For example, when two messages collide, the MAC protocol does not detect a collision but detects the reception of a corrupted message (i.e. the result of the collision is modelled and not the collision itself). Furthermore, since we evaluate MAC protocols, it is sufficient to consider the transfer of messages between stations instead of dealing with each bit separately.

When during a simulation a rare event happens, it must be possible to reproduce the simulation using the same values. We chose to have an option that makes the simulator read random numbers from a file. The file needs to be sufficiently large and a new file with random numbers must be made to perform a new reproducible simulation.

Since the simulator is written in ANSI C [2], the program works both on a Unix workstation and a Windows95 PC. It should be easy to port the program to other architectures.

2.2 Properties for a MAC Simulator

In this section we discuss the properties that must be included in the simulator to be able to effectively simulate MAC protocols. The main issue in designing a simulator is to implement it with sufficient detail, and neither less nor more.

Detail

For MAC protocol evaluation simulating messages instead of signals is sufficient. Therefore, the ether (or the medium) model does not simulate medium dependent properties that are related to, for example, the use of infrared or radio signals, such as blocking of signals (infrared) and fading (radio). Most properties of the physical medium can be translated into bit-error rates that can easily be simulated by the ether. For example, we do not model noise itself but the effect of noise by setting a bit-error rate property which allows the ether to randomly corrupts messages.

Examples of transmission effects are interference and collisions. These effects result from the implementation of the wireless MAC protocol and determine the performance of the protocol. Therefore, these effects must be included in the simulator. Modeling signal interference or collisions with message corruption is enough to obtain realistic results.

Examples of protocol properties are access rate, back-off procedure, etc. These properties have their effect on the performance of a wireless MAC protocol but they depend on the

protocol implementation. Therefore, these properties must be included in the model of the wireless MAC protocol.

Furthermore, in a wireless environment, multiple receivers may receive a transmitted message as long as these receivers are in the receiving range of the transmitter which depends on the signal strength. We assume that all antennas are omnidirectional and that signals have a predefined reach. Omnidirectional communication makes the reception depend on the distance only and the distance between two points is determined easily.

Finally, MAC protocols support the communication of user and operating system applications. It is the performance that these applications experience that is decisive as a MAC protocol performance measure. Therefore, applications must be implemented as realistic as possible.

Flexible Setups

In our simulator environment, it is possible to create certain predefined setups. A setup is a representation of the real world with mobile stations, optional base stations and the description of the wired network with fixed computers and servers. It is possible to model mobility to examine events such as the appearance of a new mobile station in a stable situation, the disappearing of a mobile station or a mobile station moving from one cell to another cell while maintaining the communication (i.e. hand over).

3 SIMULATOR ARCHITECTURE

The architecture of the wireless MAC protocol simulator is shown in Figure 1. The simulator incorporates several objects. First, there is an *ether* object which represents the communication medium. It performs the task of delivering messages to those stations that are in the position to receive a transmitted message (i.e. in range of the transmission). It also simulates medium effects such as noise by corrupting messages with a probability determined by the bit-error rate.

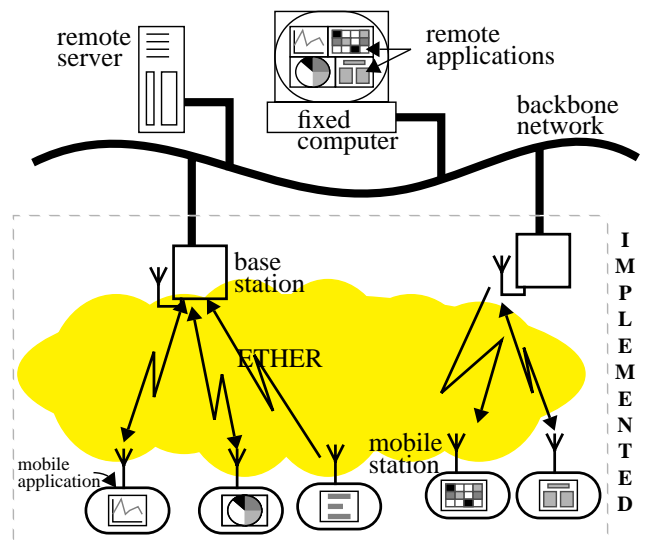


Figure 1. MAC Simulator Architecture

Then there are *mobile stations* that communicate through the ether. A mobile station is represented as an object with a protocol independent part and a protocol dependent part. To simulate a new protocol only the protocol dependent part needs to be changed.

Base stations also consists of a protocol independent and a protocol dependent part. Thus, for each protocol that makes use of base stations, a base protocol and a mobile protocol must be defined. Additionally, a base station connects to a fixed network that allows communication between base stations, and between base stations and fixed computers.

Since communication is not created by itself, we need *application models* that generate data traffic. Currently, we have implemented several models that can be simulated on the mobile stations. These models are: CBR and VBR audio, CBR video and Poisson traffic. The poisson model allows us to verify some simulation results using standard analysis methods such as Markov chain analysis.

Base stations can connect to a *fixed backbone network*, usually a high capacity local area network (LAN). This backbone network allows mobile to mobile communication between different communication cells. Fixed network hosts can be connected to the LAN. They can perform heavy duty functions for the mobile stations to save energy and reduce network traffic. Currently, the backbone network and the fixed network hosts are not implemented. Therefore, communication between base stations is not yet possible and no downlink data traffic is present. Only uplink data messages and two way control messages are exchanged. However, as uplink data communication is the main bottleneck in wireless communication, the simulator already provides very useful results.

4 MAC SIMULATOR ENVIRONMENT

The simulator is embedded in a system, shown in Figure 2. The simulator takes a number of call arguments, reads a setup file and creates two types of output files: a single *trace file* and a number of *results files*. The number of arguments is large (see example below). We shielded off the extensive argument exchange using shell programs. Using the shells it is easy to simulate over a range of values for a specific argument.

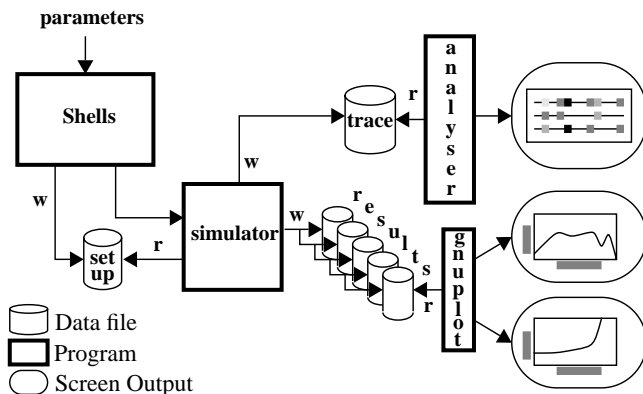


Figure 2. MAC Protocol Simulator Environment

Example of a session:

```
simulate_all -PROT ALOHA -MOB 2 -TIME 250000 -BER 0.01
-OUT /usr/tmp/loha -LOADS 0 1000 10 -RT None -NRT Poisson
-SETUP singlecell
```

This line describes a session to evaluate the Aloha protocol with two mobile stations in a single-cell setup. Each mobile station has a Poisson application running over a non-real-time (NRT) connection and does not use a real-time (RT) connection. Simulations are performed for connection loads 0.00, 0.01, 0.02,..., 1.00

and for a virtual time of 250000 time units. The results obtained by the simulations are stored in files with the prefix `/usr/tmp/loha_2`. Finally, the bit-error rate (BER) of the channel is set to 10^{-2} .

The simulator environment includes an extensible set of setup generation utilities. Each setup generator makes a setup file with specific properties for the stations that will be simulated. An example of a setup file is shown in Listing 1. The file defines two mobile stations with the x, y coordinates of the location of the mobile station, the reach of its transmitter, and the name and traffic rate of the applications that run on each of the two connections of the mobile stations. Also a base station is defined with a location given by x and y coordinates, and a transmitter reach. The number of mobile stations and base stations is not limited by the simulator.

```
mobiles: 2
134.5, 208.0, 75, None, 300.0, Poisson, 300.0
98.1, 180.0, 75, None, 300.0, Poisson, 300.0

bases: 1
150, 150, 75
```

Listing 1. Example Setup File

5 SIMULATOR IMPLEMENTATION

In this section we describe the implementation of the main objects in the simulator program. The focus is on the functionality and structuring of the objects.

5.1 Action scheduling

The beating hart of the simulator is the action scheduler. The scheduler maintains an action list with actions that have to be performed at some defined time. Actions represent events that will happen in the future. An action scheduler calls the first action and receives new actions that are returned by the called object method. Actions are references to a method of an simulated object which can be a mobile station, a base station, an application, etc.

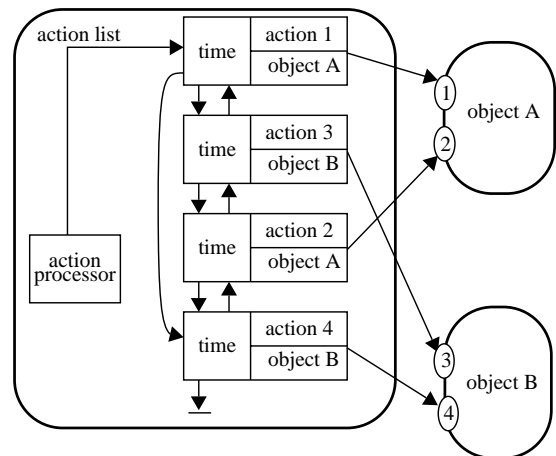


Figure 3. The Action Scheduler

The action list consists of a chain of actions that are sorted in time. Therefore, a time is associated to each action. Executing the actions, the simulator steps through time in large steps when events are widely spaced in time. Therefore the simulator is event-driven.

Every action is associated to a specific method of a specific object. Performing an action is done by calling the given method of the given object and passing the current simulation

time as a parameter. Since the only parameter given to a method is the current simulation time, the required state information must be maintained by the objects themselves.

An executed action can return zero, one or more new actions. These actions are inserted into the action list at the proper location such that the list remains sorted in time. For optimization, actions that are scheduled for the early future are inserted from the start of the list and actions that are executed in the future are inserted from the end of the list. Finally, the execution of an action completes by changing the current simulation time to the time of the next action in the list.

5.2 Base Station

Base stations are modelled as objects that can communicate through the ether and through a backbone network. Each base station consists of a common part that implements the basic behaviour of the base station that is independent of the MAC protocol and a protocol part that performs the MAC protocol functions.

The location of a base station is stored in a location object. A receive buffer receives messages that are transmitted within the reception range of the base station and a transmit buffer performs the transmission of messages through the ether. Two data queues buffer the data communication to and from the fixed data communication network. The fixed network protocol delivers data messages to hosts that are connected to the fixed network and receives new messages from these hosts.

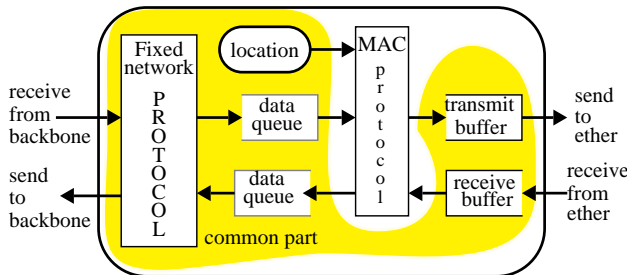


Figure 4. Base Station Implementation

5.3 Mobile Station

A mobile station is a stand-alone device that communicates over the wireless channel. The data transmission and reception operations of the mobile MAC protocol are often controlled by a base station (except for fully random access protocols like Aloha[3] or protocols that do not require base stations such as MACA[8]). Every mobile station has two main parts (see Figure 5). A common part that is protocol independent and a protocol-dependent part.

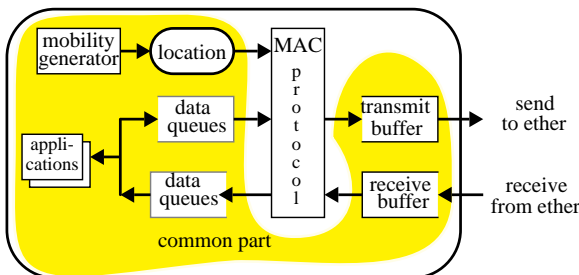


Figure 5. Mobile Station Implementation

The common part contains entities and state that are always needed and are independent of the MAC protocol that

is used. Among these entities and state are: applications that generate data packets, mobility generator that changes the location of a mobile station, mobile station name, a data queue for transmission, a data queue for reception, transmit and receive buffers, etc.

The protocol dependent part contains the MAC protocol implementation that is simulated. It contains information such as frame size (when the protocol uses frames), access probability (when the protocol uses random access), allocated slots (when the protocol allocates slots to mobile stations), etc., depending on the protocol being used.

5.4 Ether

The object ether handles all message transmissions for all transmitters. The ether administrates all communication traffic in the air on a per receiver basis. For each receiver it maintains a separate message list that contains a copy of each ongoing transmission. Two methods are used to implement a start transmission event and a stop transmission event.

When a mobile or base station starts a transmission, it calls the start transmission method of the ether object. The ether checks which receivers are in the range of the transmitter. For those receivers an *in-range* copy of the message is inserted in the message list. For the other receivers an *out-of-range* copy is inserted. When for a receiver more than one message transmission is in-range, this results in a collision. The ether replaces these colliding message copies by corrupted copies.

When the mobile or base station stops its the transmission, it calls the stop transmission method of the ether object. Before returning the copies of the message to the receive buffers that are associated with the messages lists, the ether simulates noise and other signal effects (e.g. fading) by randomly corrupting messages based on the bit-error rate and the message size.

The ether does not generate actions by itself. However, when after completion of a transmission the ether writes a message into the receive buffer of a station, this receive buffer may return an action to be scheduled. This action can serve as an interrupt to processes the received messages.

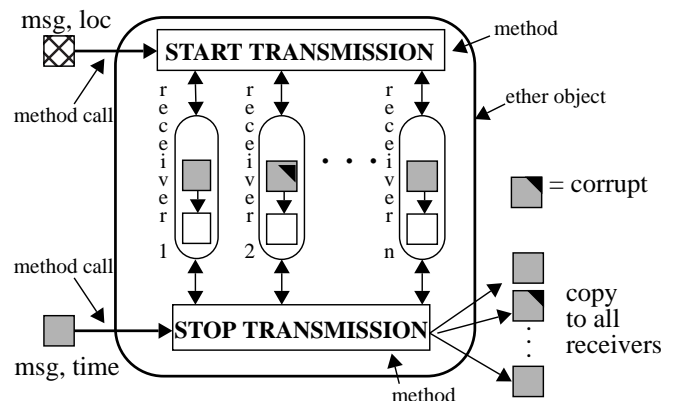


Figure 6. Ether Implementation

5.5 Administration

The objective of the simulator is to obtain and evaluate results generated by simulation of a protocol in a certain setup. Two types of results are important to evaluate the behavior of a protocol: measurements and protocol behavior.

Measurements consist of information concerning throughput, delay and, in our simulator, also energy consumption. These measurements can be used to estimate the performance of the simulated protocol.

Currently, the simulator can evaluate a certain configuration for a variety of loads. The simulator stores the results obtained by each simulation in delay, throughput and energy results files. These files can be transformed directly into a graph by using a graph drawing tool like **gnuplot**.

Measurements that the simulator currently generates are:

1	Throughput	(utilisation per connection)
2	Access Delay	(time to access the channel)
3	Queuing Delay	(time spent in the data queue)
4	Transfer Delay	(total transmission time = 2+3)
5	Connection Setup Delay	(time to setup a connection)
6	SendTime	(time transmitter was sending)
7	ReceiveTime	(time receiver was receiving)
8	SendOverhead	(sendtime and throughput ratio)
9	Energy	(energy dissipated: Tx+Rx)

These results are grouped in four ways. When there are m mobile stations and t connection types (e.g. real-time and non-real-time connections), the grouping is as follows:

1	All connections	(1 file)
2	Per type of connection	(t files)
3	Per mobile station	(m files)
4	Per individual connection	($m*t$ files)

This gives a total of $1+t+m+m*t$ groups for each of the 8 measurements.

Finally, the information is presented against the load of a connection and against the aggregated system load giving two files per results group. In total each simulation run creates $9*(1+t+m+m*t)*2$ files containing the results of the simulation. The simulation of a setup with 1 mobile station having 2 connection types creates 108 results files. For each additional mobile station, an additional 54 files are created. Since the number of files is huge, we added the option to obtain overall results only (groups 1 and 2) giving 54 files for 2 connection types independent of the number of mobile stations.

5.6 MAC Protocol Analyzer

The simulator also writes events to a trace file. This information can be used to analyse unexpected or erroneous behaviour. The information in the trace file includes event descriptions such as when a wireless station sends a message, when a wireless station receives a message, when an application creates data, etc. The information in the trace file is sufficient for finding a problem in the protocol implementation and to understand why certain (unexpected) events occur. However, the amount of information generated can be extensive. A 5 second (virtual time) simulation of a set-up with 3 mobile stations and a single base station can generate a trace file containing in the order of 100.000 events. To obtain efficient access to this information, an analyzer is implemented that presents the information in a clear and easy way.

6 STATUS AND FUTURE WORK

Currently, a virtually unlimited number of mobile stations

and base stations and the communication between them can be simulated. The mobile stations can run a number of different applications that can be configured to resemble real-life applications such as real telephony or real video.

The simulator extracts a large amount of information from a simulation. This information currently consists of the data throughput, the latency and its distribution, and the energy consumption.

The simulator has proven to be very flexible. It was not difficult to implement a variety of MAC protocols. Currently, we have implemented, Aloha and Slotted Aloha[3], ISMA[4], R-ISMA[5], TDMA[6], R-TDMA[7], MACA[8], and MACAW[9]. The results that we obtained with the simulator compare very well to the analytical results (as far as the analytical results are known in the literature).

A measurement that should be added to the simulator is the number of dropped packets. This is useful to evaluate time sensitive applications such as real-time audio and video. By adding a time limit to the data packets, packets can be dropped when that time has passed without having sent the packet.

7 CONCLUSIONS

We have implemented a simulator that can be used to simulate wireless MAC protocols. The simulator proved to be very flexible as it required little effort to implement several wireless MAC protocols that are described in literature.

The simulator can give a good indication of the performance of a wireless MAC protocol. The performance measurements compare well with the analytical results that are described in literature.

The simulator runs on a Unix Workstation and on a PC. Since the code of the simulator is in ANSI C, we expect that it is no problem to port the program to other platforms.

8 REFERENCES

- [1] Hammond, J.L., O'Reilly, P.J.P., Performance Analysis of Local Area Networks, Addison-Wesley Publishing Company, pp. 125 - 128, 1986.
- [2] Kernighan, B.W., Ritchie, D.M., The C Programming Language, Second Edition, Prentice Hall PTR, Englewood Cliffs, NJ, USA, 1988.
- [3] Abramson, N., Development of the ALOHANET, IEEE Transactions on Information Theory, vol. IT-31, pp. 119-123, March 1995.
- [4] Mukumoto, K., Fukuda, A., Idle-Signal Multiple-Access (ISMA) Scheme for Terrestrial Packet Radio Networks, Electronics and Communications in Japan, Vol. 64-B, No. 10, pp. 66 - 74, 1981.
- [5] Wu, G., et. al., Performance Evaluation of Reserved Idle Signal Multiple-Access Scheme for Wireless Communication Networks, IEEE Transactions on Vehicular Technology, Vol. 43, No. 3, pp. 653 - 658.
- [6] Tanenbaum, A.S., Computer Networks, Second Edition, Prentice Hall, Englewood Cliffs, New Jersey, 1989.
- [7] Linnenbank, G.R.J., et al., A Request-TDMA Multiple-Access Scheme for Wireless Multimedia Networks, Proceedings 3rd International Workshop on Mobile Multimedia Communications, Princeton NJ, Sept. 1996.
- [8] Demers, A., et. al., A Nano-Cellular Local Area Network Using Near-Field RF Coupling, Virginia Tech's Fourth Symposium on Wireless Personal Communications, 1994.
- [9] Bharghavan, V., et al., MACAW: A Media Access Protocol for Wireless LAN's, Proceedings of SigComm'94, 1994.