The 11th International Symposium on Wireless Personal Multimedia Communications (WPMC'08)

CROSS-LAYER MULTI-HOP SIMULATOR FOR AD-HOC IEEE 802.11E

João M. Ferro, Fernando J. Velez
Instituto de Telecomunicações
Covilhã, Portugal

ABSTRACT

The purpose of this paper is to present a custom-made cross-layer network simulator for wireless IEEE 802.11e in a multi-hop environment. A previous work of our research team created a simulator just for the physical plus MAC layers, which in this work was adapted to model the transmission of a packet from a source to a destination using intermediate nodes. Some initial tests were performed using a simple routing algorithm, and the results obtained from these tests indicate that the system is capable of deliver the packets regardless of the source/destination, and successfully calculates several metrics such as end-to-end delay and number of packets lost.

I. INTRODUCTION

The world is going wireless. In today's world, wireless networks are getting an increased importance, antennas for mobile phones, and antennas for Wi-Fi are present almost everywhere. WiMAX networks are also beginning to be deployed worldwide. Besides working in a wireless environment, these networks share also another feature: they require some sort of backbone infrastructure in order to maintain the connectivity of the network. For example, if someone makes a phone call, the conversation will always pass from the cell phone to the operator's antenna, and then to the receiver's phone, even if they are both in the same building. Resources would certainly be saved if somehow the cell phones could connect directly to each other.

However, there is a type of network in which all the participants (generally called nodes) can communicate directly with the others, as long as they are in the range of its communication device. A network of this type is called an ad-hoc network. In this network the nodes collaborate among them to bring the messages from the source to the destination. If the nodes are directly connected the message will be simply delivered, but if they are too far, intermediate nodes are necessary. The choice of which intermediate nodes to use is made in each node, i.e., each node has a sufficient knowledge of the remaining network to know where to forward its data. Thus, a centralised infrastructure to establish the connectivity is not required, making it more tolerant to individual systems failures. In Wi-Fi, the specification IEEE 802.11 refers to this type of network as Independent Basic Service Set (IBSS).

The paper is organized as follows. Section II. presents some background information about the previous simulator and about IEEE 802.11e. Section III. describes the features of the new simulator highlighting the modifications performed. Section IV. discusses the results of the initial simulations, while Section V. presents conclusions and suggestions for future work.

II. BACKGROUND

The simulator is based on the one previously developed at the Instituto de Telecomunicações, Laboratório da Covilhã, as part of the European IST-UNITE project. One objective of this project is to "implement an efficient, accurate and scalable Virtual Distributed Testbed (VDT) to support cross-system, and cross-layer optimisation of heterogeneous systems in unified manner". To achieve this, a time-driven HSDPA simulator and a event-driven Wi-Fi simulator were created, which can run separately or together, in tight cooperation for a coexistence scenario. The Wi-Fi simulator is built for the infrastructure mode, i.e., an architecture with one access point (AP) and several wireless stations (STAs) connected to it. As so, it can only simulate traffic from one STA to the AP and vice-versa. The simulator automatically calculates interesting metrics such as the throughput, packet delay, number of lost packets, number of retransmissions, and number of collisions. This computation is presented in total per simulation, and in average per unit of time. Since it is oriented to the support of the Quality of Service (it implements IEEE 802.11e with four access categories), the output metrics are calculated for each traffic type: voice (VO), video (VI), background (BK) and best-effort (BE).

Quality of Service (QoS) refers to the reservation of resources for certain applications, guaranteeing that most important data packets (or more delay-sensitive packets) will have higher priority levels, waiting less time in the buffer before transmission. This is mostly done at the Medium Access Control (MAC) layer. In the standard 802.11, the functions that coordinate the MAC are called the Distributed Coordination Function (DCF) and the Point Coordination Function (PCF). They establish the timing and sequence for each station to access the shared medium. The PCF is only used in the infrastructure mode, with the AP acting as the coordinator. It provides some basic QoS support, but since it is defined as optional and is more complex and costly to implement, only a few APs support it. The DCF does not present QoS guarantees at all. The amendment 802.11e was introduced to enhance the support of QoS. This one introduces the Hybrid Coordination Function (HCF), which defines the HCF Controlled Channel Access (HCCA) and Enhanced Distributed Channel Access (EDCA). In EDCA, a Transmission Opportunity (TXOP) is won according to the traffic type, scheduling firstly higher priority traffic, like voice and video. The HCCA is more complex, but allows greater precision in terms of QoS. A more detailed explanation of these functions and the standard itself can be found in [1].

In the simulator each station (and AP) has four buffers, one for each traffic type. In order to allow the simulations to run faster, the simulator is built with an event-driven approach. Each new packet is associated with an event, the engine uses

an internal clock to pick up the next event from the event list, and passes the corresponding packet to the developed MAC and Physical layers. These model the transmission of the packet through the wireless channel, for a detailed explanation of the features of these lower layers one can consult [2].

III. The Simulator

The current simulator uses the engine and most of the features previously developed. To provide support for multi-hop environment it was necessary to implement a routing algorithm above the already existing Medium Access and Control (MAC) plus physical (PHY) layers, as it is presented in Fig. 1.

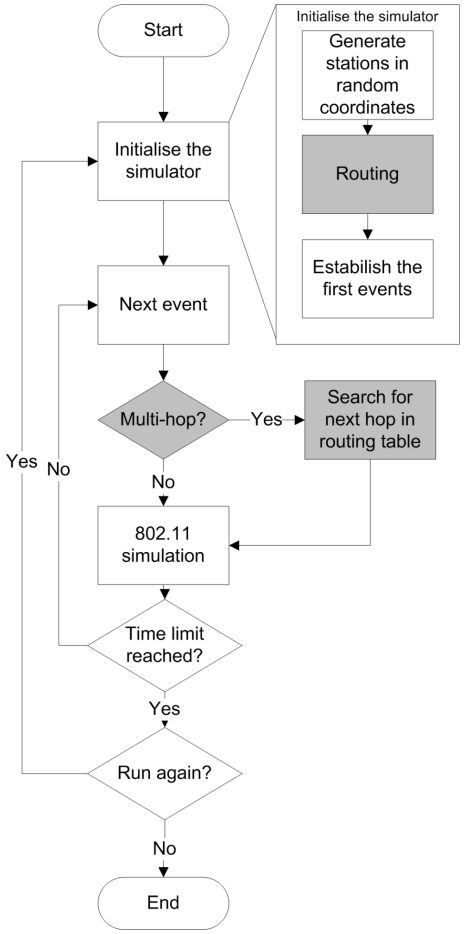

Figure 1: Simulator flowchart, modules added to the previous simulator are displayed darkened

The simulator initialises by spreading randomly the stations on the field, and with their coordinates determines which can communicate directly. With this data, the selected routing algorithm determines the shortest path from a station to all the others. For these initial tests we chose the well-known Dijkstra's algorithm [3], which calculates the least cost path from one node to all the remaining nodes. This calculation takes place before the simulation starts, making the routing table available to all nodes from the start. The routing table is available for changes during the entire simulation, allowing the use of cross-layer design (i.e., gather information from physical and MAC layers and use them into the routing algorithm) and the adaptation to topology changes. In this paper, however, we show only the initial tests, the routing table is not modified during the simulation time, and we present no cross-layer at all.

After the initialisation is complete, the engine collects the first event and the simulation starts. Whenever a packet arrives at a station, a new module verifies if the packet has another destination by checking the two additional fields now included in the packet header: besides the "payload", "origin station", and "destination station", among others, each packet now includes the "initial origin" and the "final destination" fields. If found that the destination is another node, the routing table is consulted to determine the next hop, and the packet is added to the machine buffer with the new destination stamped.

The simulation ends when the time limit is reached, the system gathers the data collected and performs the calculations to obtain the simulation output. It will then repeat the simulation up to a number of times defined by the user, and in the end it will present an average output for all simulations.

For the end-user, this simulator is capable of simulating a network with a unrestricted number of nodes, in random configuration type. The use of the current routing algorithm allow the connection of every reachable station in the field, allowing for a message to reach any destination. Almost all of the simulation parameters can be modified by the user, some of the most useful are referred in Table 1.

Table 1: Some values modifiable by the end-user

Parameter	Default value	Description
Side	60m	Side of the square where the stations are placed
Range	35m	Transmission range of each station
Simulation Time	10000ms	Period of time to be simulated
Payload XX	- bits	Length of each packet of the type XX
Inter Arrival XX	- ms	Interval of time to generate a new packet of the type XX
Degrees Freedom	5	Number of times the simulation will be repeated

IV. Results

To test the simulator we first tried to verify how the presence of multiple streams in parallel would affect the end-to-end delay and packet losses. For this test, and the subsequent ones, we used the deployment of stations presented in Fig. 2 (notice that to achieve this the random placement of the stations was frozen).

The inner circle has a radius of 15 meters, the next one 30m, and the outer one 45m. The square field has $120\text{x}120\text{m}^2$. In the first set of tests we added sequentially video streams as follows:

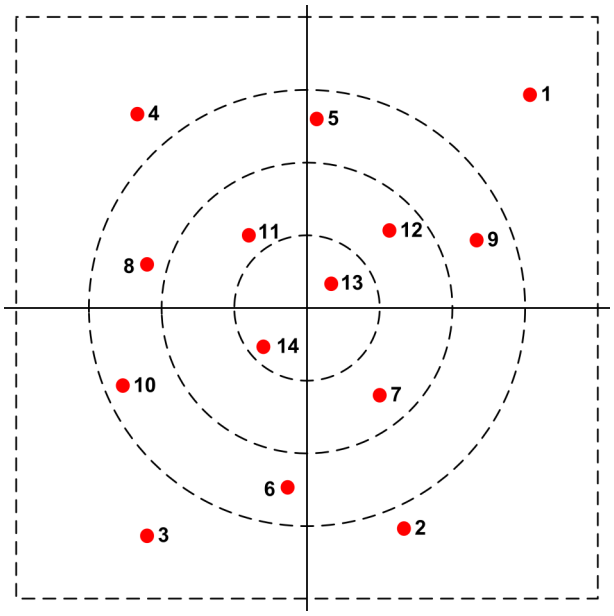

Figure 2: Station deployment for the tests

1. From stations 1 to 3;
2. From stations 4 to 2;
3. From stations 5 to 6;
4. From stations 8 to 9.

For each of these streams the routing algorithm selected the following paths:

1. $1 \rightarrow 9 \rightarrow 13 \rightarrow 14 \rightarrow 10 \rightarrow 3$;
2. $4 \rightarrow 11 \rightarrow 14 \rightarrow 7 \rightarrow 2$;
3. $5 \rightarrow 13 \rightarrow 14 \rightarrow 6$;
4. $8 \rightarrow 14 \rightarrow 13 \rightarrow 9$.

Fig. 3 presents the end-to-end delay for the video packets flowing from station 1 to station 3 (stream 1). By increasing the number of stations that generate packets, the end-to-end delay is increased, like one should expect.

Another interesting metric is the number of lost packets, which is presented in Fig. 4. When the number of packets to be transmitted increases, there are more collisions and packet losses. This is due to the fact that more packets are being pushed to the shared medium in the same time period.

Being intrinsically built for the QoS support, our simulator can work with four different traffic types. By using this feature, we produced a new set of tests keeping the same topology but modifying the type of packets in each stream:

1. Video stream from stations 1 to 3;
2. Background traffic from stations 4 to 2;
3. Voice traffic from stations 5 to 6.

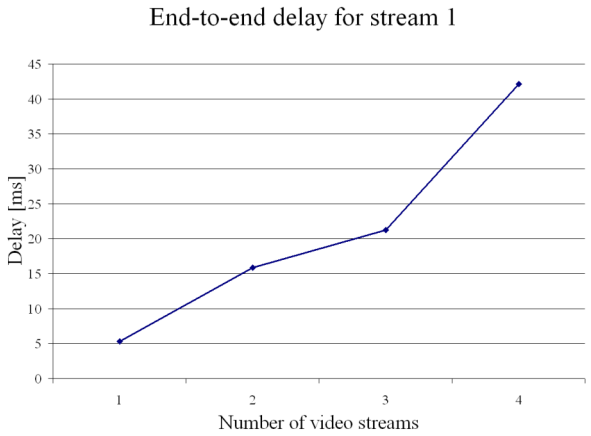

Figure 3: End-to-end delay for stream 1

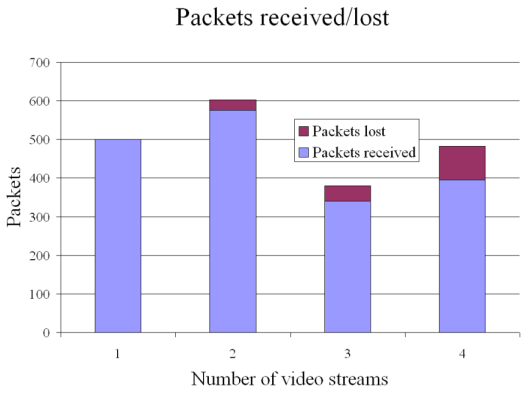

Figure 4: Total of packets received/lost

We modelled the traffic according to the following criteria: for the video, a new packet is generated every 10ms; for the voice, packets are generated in both directions, the receiver generates a packet 5ms after the sender has generated its own, this is repeated every 20ms; for the background, a new packet is queued every 12.5ms; for the best-effort traffic, a new packet is placed in the buffer every 2ms, however this traffic type is not considered in this paper. These values were taken from [4] and are summarised in Table 2, and the results for the number of packets successfully delivered and lost are presented in Fig. 5 and Fig. 6.

Table 2: Configuration for each traffic type

Type	**Packet size [bits]**	**Period [ms]**
Video	10240	10
Voice	1280	20
Background	18430	12.5

While Fig. 5 presents the results for each packet, Fig. 6 addresses each stream, i.e, the latter only counts a successful packet when it arrives at the final destination, while the former counts whenever a packet arrives at any station. For this reason,

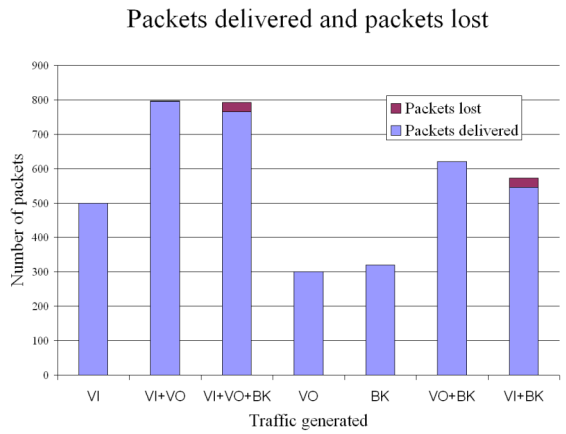

Figure 5: Total of packets received/lost

and looking at the results for the video stream (VI) only, in the former case, 500 packets were successfully delivered, while in the latter only 100 packets arrived at the final destination. Remember that this stream is being transmitted in a 5-hop path, and that the most relevant result is the latter.

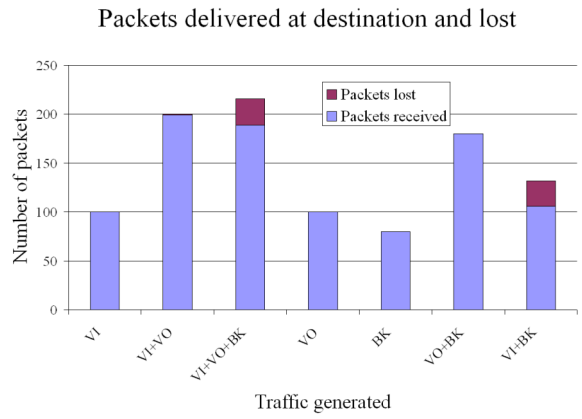

Figure 6: Total of packets lost/received at destination

With only one stream, the system behaves perfectly, all of the packets are delivered at destination. Since the rate that packets are generated has a period longer than the end-to-end delay, there are no collisions. However, when more than one type of traffic is being generated simultaneously, the packets start to collide and some of them are lost (current policy establishes that a station will drop a packet after it experiences 8 collisions). By analysing the data, one can conclude that the "background" traffic is the one that causes more problems, which can be explained by the larger size of each packet. Due to the limitation of the maximum packet size which can be transmitted at a time, in this traffic type each packet will be fragmented into four smaller packets, increasing the number of packets in circulation. For voice traffic, the delay is very low and it seems that it does not affect the remaining traffic at all. This is essentially due to the small packets generated by this traffic type (no fragmentation is required).

This behaviour can also be noticed by looking at the end-to-end delay, Fig. 7. When only one stream is flowing the delay is minimum for each traffic, when other streams are added the delay starts to increase. The background traffic seems to affect more negatively the other streams. For example, with only one video stream (VI) the end-to-end delay is 5.259ms, adding a voice stream (VI+VO) will increase this delay (i.e., end-to-end delay for video stream) to 7.867ms, while adding just a background stream (VI+BK) will increase it to 15.482ms. Of course, this delay is maximum when the 3 streams (VI+VO+BK) are flowing (18.490ms). Notice that in order to keep the important part of the chart visible, the delay for the background traffic is not entirely shown in the VI+VO+BK and VI+BK bars.

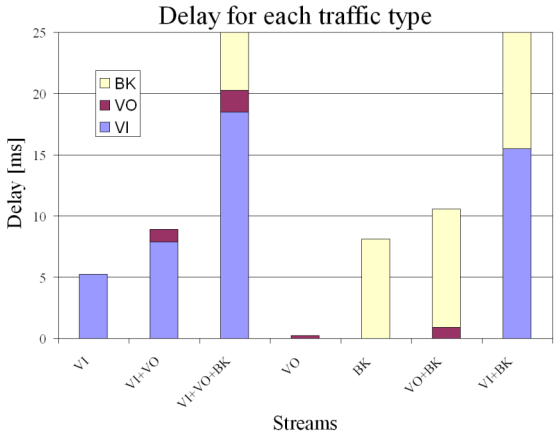

Figure 7: End-to-end delay according to the traffic type

End-to-end delay and packet loss are very important metrics in terms of traffic quality measurement, to get the best performance it is necessary a minimal end-to-end delay and a null packet loss. The most delay sensitive packets are voice and video, since an exaggerated delay can cause these real-time services to become incomprehensible to the end-user. With the increasing of the number of stations, packets start to collide and increase the queue size, making packets spend more time in the buffers and thus increasing the end-to-end delay. The mechanism that prevents the queue to become too large is very simple: if a packet experiences more than a certain number of collision, the machine will simply discard the packet. This number, called L_COL in the simulator, can be modified by the user of this one, and we tried several values to determine which is the best. The setup for Fig. 8 was using video traffic as stream 1, while stream 2 and 3 where using background traffic, the results are for the video stream only.

Although it seems evident that the larger the collision limit the better, one should also look at Fig. 9 which represents the end-to-end delay for the same test.

Reducing the number of collisions a packet can experience will decrease the end-to-end delay, however it will increase the packet loss. One should note that for delay-sensitive traffic, a packet that arrives out of time is useless and will be discarded by the higher layers, however for other traffic is more impor-

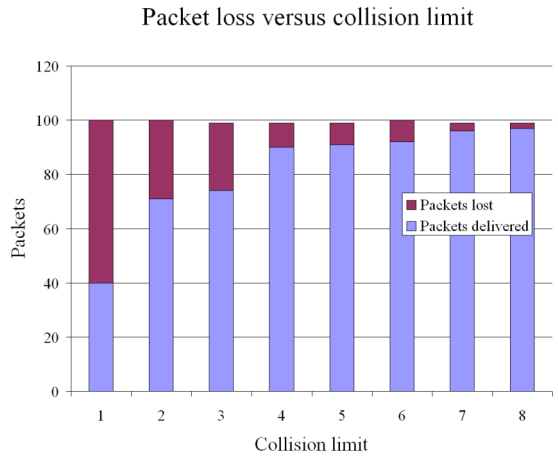

Figure 8: Packets lost and received for each configuration of the packet collision limit

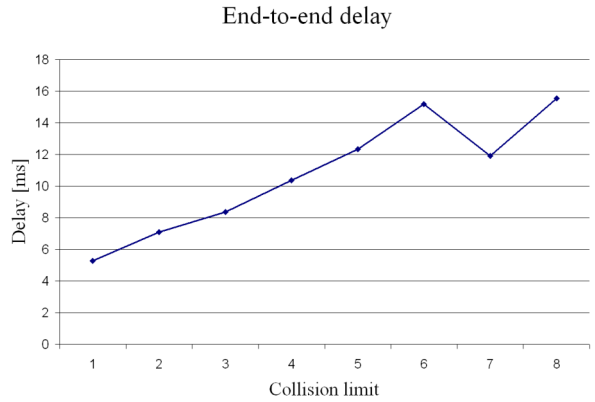

Figure 9: End-to-end delay for the video stream

tant to arrive at destiny than to arrive "on time". Thus, a compromise must be done when choosing the number of collisions allowed for a packet.

V. Conclusions

In this paper we presented the initial results of our custom-made IEEE 802.11e simulator. Using a previous work of our team, which developed the MAC and Physical layer for a simulator in the context of the project IST-UNITE, we added above these a routing algorithm to allow the simulation of a network in the IBSS (ad-hoc) mode. After this modification we performed some tests to evaluate if the simulator was behaving like one should expect, which were very positive. A few more validation tests are planned for the simulator, and if these reveal to be successful the release of the source code and/or the simulator itself to the scientific community for research purposes is being considered. If this becomes a reality, adding a visual interface would also be considered.

References

[1] Anand R. Prasad and Neeli R. Prasad. *802.11 WLANs and IP Networking*. Artech House, Boston, 2005.

[2] O. Cabral, A. Segarra, and F.J. Velez. Event-driven simulation for ieee 802.11e optimization. *IAENG International Journal of Computer Science*, 35(1), 2008.

[3] E.W Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[4] Qiang Ni. Performance analysis and enhancements for ieee 802.11e wireless networks. *IEEE Network*, 19(4):21–27, 2005.