An Integrated OMNeT++Implementation of 802.11

A. Ariza-Quintana E. Casilari J. Hurtado-López
Dpto. Tecnología Electrónica, University of Málaga
Campus de Teatinos, 29071 Málaga (Spain),
Tfno.: 34-952132755; FAX 34-952131447
aarizaq@uma.es ecasilari@uma.es jhurtado@uma.es

ABSTRACT
This work presents a re-implementation of the 802.11 Wi-Fi standard for the OMNeT++ *inet* framework and its forks. This new implementation supports the versions 802.11a/b/g/p of the IEEE 802.11 family of standards, as well as the extensions of 802.11e to provide Quality of Service (QoS). The software is available at: https://github.com/aarizaq/inetmanet-2.0

Categories and Subject Descriptors
D.3.3 [**Programming Languages**]: Frameworks, D.4.8 [**Performance**]: Simulation

General Terms
Algorithms, Experimentation, Standardization, Languages.

Keywords
Wi-Fi, 802.11, *inet* framework

1. INTRODUCTION
Inet- framework (and especially *inetmanet* fork) already includes several transmission modes defined by the 802.11 standard [1]. However, these modes are implemented as separate models. This obliges to replicate a high fraction of the programming code, which is basically common for all the modes. Moreover, the existence of separated modules increases the cost of the upgrading and maintenance of the code, as far as any correction or updating of one model has to be repeated for the rest. This diversity of codes also complicates its use by inexpert or beginning OMNeT++ programmers.

In order to avoid this inefficiency in the way 802.11 is implemented, the corresponding code has to be restructured. We have benefited from this re-organization of the code to add new functionalities and correct some errors existing in the models.

As a prerequisite for the new code, we established that all the different models should be structured in a single Wi-Fi model, so the same code could support the different versions of the standard.

2. IMPLEMENTATION
To develop the new model, we based on the 802.11e/g [1] model present in *inetmanet* and the Wi-Fi module available for NS-3 simulator [2], which in turn derives from the implementation made for YANS [3] simulator. The integration of the new features was performed following the documentation of the standard [1],

SIMUTools 2012 March 19–23, Desenzano, Italy.
.

2.1 Radio Layer
The new model creates two new classes for the Radio Layer. The first one (*ModulationType*) contains a structure which stores a set of parameters describing the employed modulation (see Table 1).

Table 1. Utilized fields for the characterization of a transmission mode

`isMandatory`	It indicates if the mode is designed as mandatory by the standard
`bandwidth`	Bandwidth (in MHz) utilized by each channel.
`codeRate`	Number of encoded bits per symbol
`dataRate`	Binary rate
`phyRate`	Speed of the physical layer, expressed in symbols per second
`constellationSize`	Constellation size of the modulation scheme
`modulationClass`	Type of modulation

All the methods of the second class (*WiFiModulationType*) are static so they can be accessed without creating an object of the corresponding type. This class fills the fields of the *ModulationType* class with the adequate values for the 802.11 modulation scheme and binary rate that are being employed by the Wi-Fi communications. For example, the method *WifiModulationType::GetOfdmRate13_5MbpsBW5MHz* returns an object of the *ModulationType* class whose fields define an OFDM modulation with a bit rate of 5 Mb/s and a 5 MHz channel bandwidth. Aiming at easing the programming, we introduce a set of methods called *getMode80211x*, where *x* indicates the employed version of the standard (a,b,g or p). These methods allow to generate a *ModulationType* object by automatically fulfilling the corresponding fields with the adequate binary rate for the chosen version.

The *WiFiModulationType* class also offers a series of classes that enable the computation of the transmission time of the frames and headers of the Physical Layer. The timing intervals SIFS and DIFS between consecutive frames are also computed as a function of the selected modulation.

The way in which the probability of suffering a bit error is computed has also been modified. Now the occurrence of bit errors can be decided depending on the data of a specific table. This table could be based, for example, on empirical observations of real transmission scenarios or, otherwise, on offline data calculated with complex models, which are hard to implement because of the computation time that they require.

Alternatively the bit errors can be analytically estimated. For this purpose we made use of the modulation error model of YANS [3] simulator, also utilized by NS3 [2].

2.2 MAC Layer

By means of new channel access methods, such as the Enhanced Distributed Channel Access (EDCA), IEEE 802.11e allows to establish different traffic categories. These traffic categories are intended to provide Quality of Service (QoS) by prioritizing certain flows. To characterize this multi-category operation of the MAC layer in OMNeT++, a *std::vector* template is employed for every category. The structure stores all the parameters related to a particular traffic category. Thus, the code can work with an arbitrary number of categories.

When employing EDCA, the categorization of the traffic at the MAC layer module requires to cancel the queues existing at the management module of the Link Layer. Consequently, all the frames received at this management module are immediately sent to the MAC module, where packets will be classified and queued.

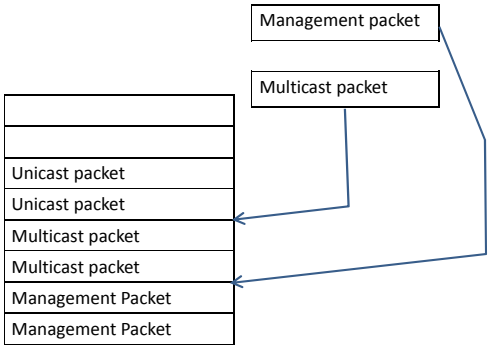

Figure 1. Insertion of the packets in the queue using priority criteria

As soon as a 802.11 frame arrives at the MAC layer, a new categorization module decides which category it belongs to.

A second classifier module is in charge of queuing the packet within the corresponding queue of the selected category. In this sense, every category can be assigned just a simple queue or a set of queues (according to multi-queue priority system). For the first case, to guarantee the priority of the management information, the queues at the MAC layers are ordered so that the management packets are located in the first positions of the queues. Figure 1 shows an example of the ordering criteria in the queues: management packets are prioritized while multicast data frames are served before any unicast data packet.

In addition, the performed implementation also presents the possibility of utilizing a multi-queue classifier module (sketched at figure 2) for each category. For every category, this module creates an arbitrary number of sub-queues with priority management. These sub-queues are divided into two subgroups. The first subgroup is governed by a strict priority policy. Under this policy, packets of a queue of a higher priority are always transmitted before any packet present in a queue with a lower precedence. By default, three strict-priority queues are created: one (with the highest priority) to handle the management packets, one for the multicast/broadcast frames and, finally, a low priority queue for the rest.

Additionally, we have also implemented a second subgroup which utilizes Weighted Round Robin to schedule the packet service of the different queues.

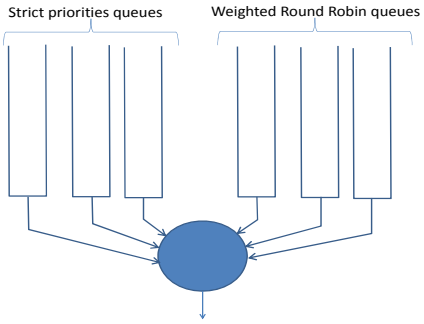

Figure 2. Multi queue priority system

Both the categorizing and the classifier modules are programmed independently of the MAC layer, so they can be adapted to a particular QoS policy without altering the source code of the MAC layer. For this purpose, both the categorizer and the classifier modules inherit from a new virtual pure class (*IQoSClassifier*). Thus, different classifier/categorizer modules can be programmed. So, the criteria to categorize/classify traffic can be easily modified in the input configuration parameters of the MAC layer, just indicating the particular categorizer/classifier that is going to be employed.

Another feature that has been incorporated is that user can optionally configure the values of certain timing parameters (e.g.: the SIFS interval, the Slot Time and the waiting time of the Acknowledgment packets). If they are not user defined, they are set to the default values defined by the standard for the particular modulation that is being simulated.

3. FUTURE EXTENSIONS

The code is presently being extended to incorporate different features. For example, the next version will incorporate the Block-ACK mechanism. We also contemplate to include Weighted Round Robin (WRR) as a new scheduling policy for the management of queue. This will enable to prioritize a certain traffic type within a particular class.

4. ACKNOWLEDGMENTS

This work was supported by Project No. TEC2009-13763-C02-01/TCM

5. REFERENCES

[1] IEEE 802.11-2007 IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY)

[2] Ns3. Network Simulator 3 http://www.nsnam.org/

[3] Yans - Yet Another Network Simulator http://sourceforge.net/projects/yans-netsim/

[4] S. Keshav, Srinivasan, *An Engineering Approach to Computer Networking*, Reading (USA), Addison-Wesley, 1997.