Implementation of an IEEE 802.11 Wireless LAN Model using OPNET™

Rusty O. Baldwin, Nathaniel J. Davis IV, Scott F. Midkiff
Bradley Department of Electrical and Computer Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061-0111

ABSTRACT

Implementing a model of a multiple access protocol such as IEEE 802.11 can be a highly error-prone task. Standards are typically textual and can, however precise the wording, have several plausible interpretations. In this paper we describe how the formal specification of the IEEE 802.11 standard [IEE97] was used to overcome this ambiguity and produce an accurate OPNET simulation model. We found that the objects used in the formal specification language SDL-92 (Specification and Description Language) mapped to OPNET objects quite well. The IEEE 802.11 simulation model is described and validation of the model is discussed. Research into real-time issues in wireless LAN using the IEEE 802.11 model is briefly presented.

INTRODUCTION

Building a simulation model from a specification is a frustrating process. The specification is invariably large, seemingly unorganized, and details about important subsystem interactions can be scattered throughout the document. The document may also contain apparent (or actual) contradictions and ambiguities. Furthermore, in group projects, different interpretations of the specification will occur and significant time will be wasted reworking the model.

In this paper, we discuss the modeling of a communications protocol where many of the problems cited above were avoided by using a formal specification of the system. In a very straight-forward manner we mapped the objects in the formal specification to objects in MIL3's communication simulation system, OPNET. In this paper we describe the communication protocol being modeled, IEEE 802.11, and how it differs from more familiar multiple access protocols such as Ethernet. Next, we discuss the mapping between the formal specification of the protocol and OPNET objects. The model itself is then described and research issues in real-time wireless LANs is presented. Finally, validation of the model is discussed.

IEEE 802.11

IEEE 802.11 is a recent (1997) standard developed for wireless local area networks (WLANs). It is a multiple access protocol in which stations in the network must "compete" for access to the shared communications medium to transmit data. It shares some characteristics of more familiar multiple access networks such as Ethernet (IEEE 802.3) but also has significant differences. IEEE 802.11 uses, as does Ethernet, a carrier sensing capability to determine if the communications medium is currently being used. If two or more stations in the network transmit at the same time (i.e., a collision occurs), stations retransmit their data after random periods of time as in Ethernet.

IEEE 802.11 differs from many multiple access protocols in three ways: (1) the transmission medium, (2) collision detection, and (3) the backoff algorithm. In most networks, the data is transmitted on a piece of wire or fiber-optics. In wireless networks, radios transmit the data over the air. This so-called "air-interface," in contrast to wires or fiber-optics, is prone to induce bit errors into the data being transmitted. Typical bit-error-rates (BER) for wires and fiber-optics are 10^{-10} and are fairly static. The wireless BER can be very dynamic and can be as poor as 10^{-2} or worse. This highly dynamic environment obviously presents unique challenges to the implementation of WLANs.

A second area where IEEE 802.11 differs from many multiple access protocols is in collision detection. In wired networks, stations can listen to their own transmissions. If another station is transmitting at the same time, stations will detect that a collision has occurred. When using radios to transmit data, stations cannot listen to their own transmissions. They therefore must rely on an acknowledgment from the receiving station to determine if the transmission was successful. If no acknowledgment is received, the transmission is attempted again.

A third area where IEEE 802.11 differs is in its backoff algorithm. In many backoff algorithms, a timer is set and decrements until it reaches zero whereupon the station will transmit the data (assuming an idle medium). In IEEE 802.11, the timer counts idle slots and only decrements when the medium has been detected as being idle for the entire slot. The backoff algorithm works basically as follows: a backoff timer is set by selecting a random integer from a uniform distribution over the interval [0, CW], where CW is the width (in slots) of the contention window (the contention window is the range of integers from which stations in the network choose their backoff timer values). For every idle slot that is detected, the timer is decremented by one. If the medium becomes busy prior to the timer expiring, the timer is frozen until another idle period is detected—when the timer is decremented again. When the timer reaches zero the station transmits its packet. If there is a collision, the CW is doubled until it reaches a predefined maximum value, CWmax. The CW is reset to a default minimum value of CWmin after a packet is successfully transmitted.

MAPPING BETWEEN SPECIFICATION AND DESCRIPTION LANGUAGE (SDL) OBJECTS AND OPNET OBJECTS

The IEEE 802.11 specification has both a textual description of the standard and a formal description written in SDL-92 [EHS97]. Both the textual and the formal description are normative, that is, if a system correctly implements the formal (and/or the textual) description, it is, by definition, an IEEE 802.11 implementation. This presents an obvious advantage to someone modeling the system since the model (correctly implemented) inherently conforms to the standard. Additionally, the formal description contains all of the subsystem interactions explicitly identified in the location where they occur and all subsystem interfaces are identified. Obviously, we cannot present a complete description of SDL, however, the major components of the SDL language are quite similar to OPNET objects and so should be easily followed by anyone familiar with OPNET.

Three fundamental objects in SDL are blocks, processes, and signals. Blocks determine lexical scope and structural hierarchy while processes specify behavior using finite state machines. Processes operate concurrently and independently and they communicate using signals. Each block may contain other blocks and/or processes. These fundamental SDL objects are shown in Figure 1. Note that this figure does not contain all (or even most) of the objects available in SDL. Figure 2 is the SDL diagram for the subset of IEEE 802.11 functionality we implemented. The model does not include encryption, authentication, power-save mode functions, fragmentation/de-fragmentation, and the point coordination function (PCF). These functions were not included since they were not relevant to the real-time issues we were investigating. The request-to-send (RTS) and clear-to-send (CTS) capability will be added at a later date. The solid line that encloses the figure denotes the logical boundary of the object.

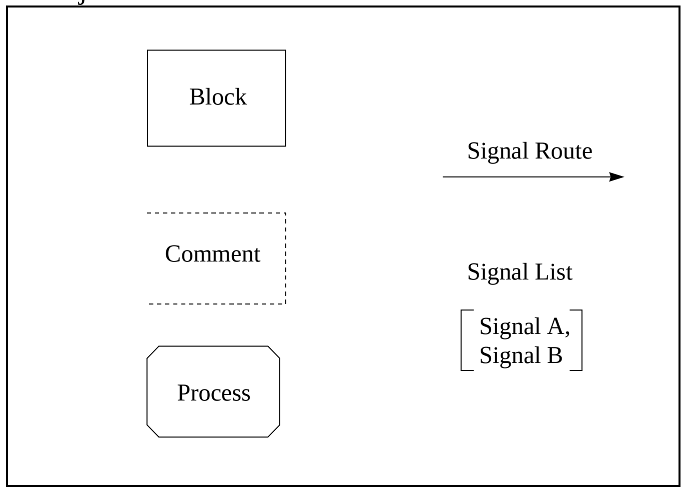

Figure 1: SDL Legend

At some point in a hierarchy of SDL blocks behavior is specified by including process objects. Focusing on the Transmission block in Figure 2, we will describe the process objects and the symbols used. Figure 3 shows the view inside of the block Transmission. Note how the input and output signals in Figure 3 correspond to those in Figure 2, as one would expect. More detail about which processes these signals go to or are received from is included at this level.

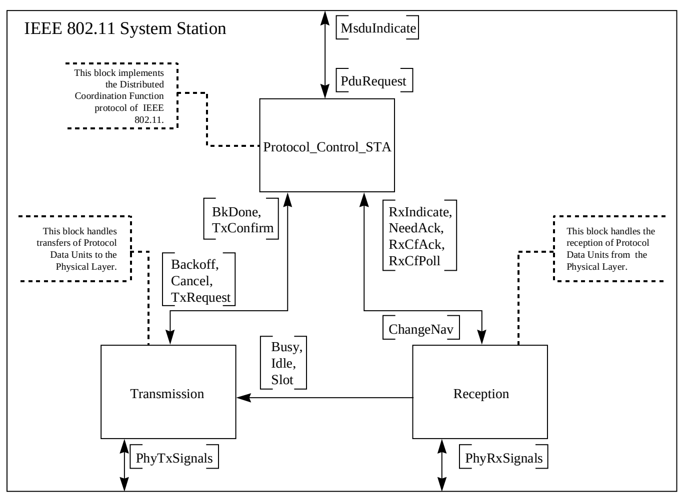

Figure 2: IEEE 802.11 System Station

The process objects have their own symbols, some of which include: start, state, input signal, output signal, and task. These process object symbols are shown in Figure 4. They are self-explanatory if one keeps in mind that the process object is essentially a finite state machine. One exception is the task symbol which indicates algorithmic steps that need to be accomplished within the process.

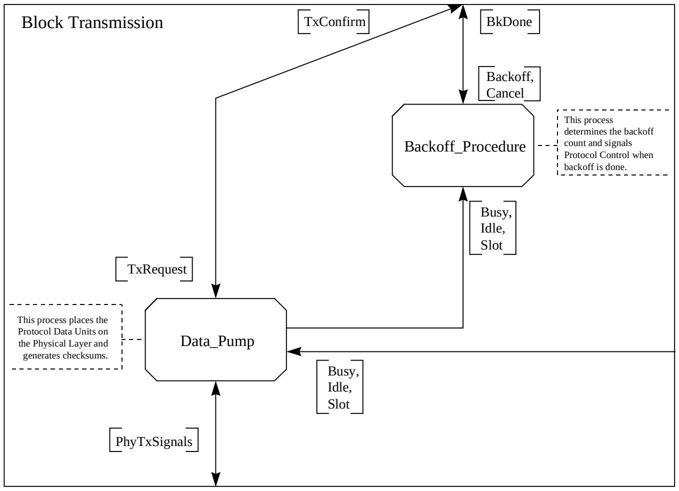

Figure 3: Block Transmission

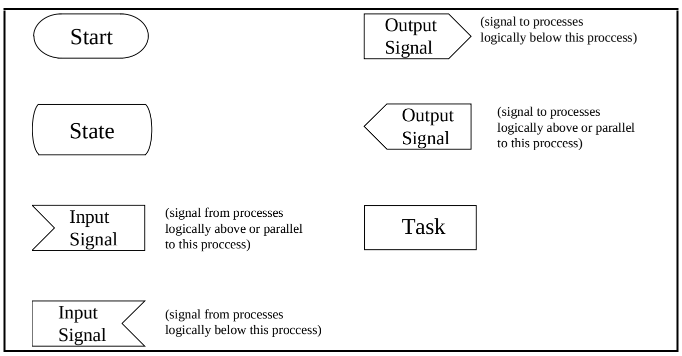

Figure 4: Process Legend

Figure 5 shows an extract of the process block for Data_Pump in Figure 3. In this process, a computational task block is encountered first. Next, the process enters the Tx_Idle state where it remains until it receives one of the signals TxRequest, Busy, Idle, or Slot. If Data_Pump receives TxRequest, the process transmits a packet via other processing not shown in the figure. If it receives Busy, Idle, or Slot, Data_Pump sends the same signal to Backoff_Procedure and returns to state Tx_Idle.

These SDL objects map quite nicely to OPNET objects. The IEEE 802.11 System Station block shown in Figure 2 corresponds to a node. The processes within lower level blocks (i.e., Data_Pump and Backoff_Procedure within Transmission) map to objects available within the OPNET Node Editor such as processors, queues, generators, etc.

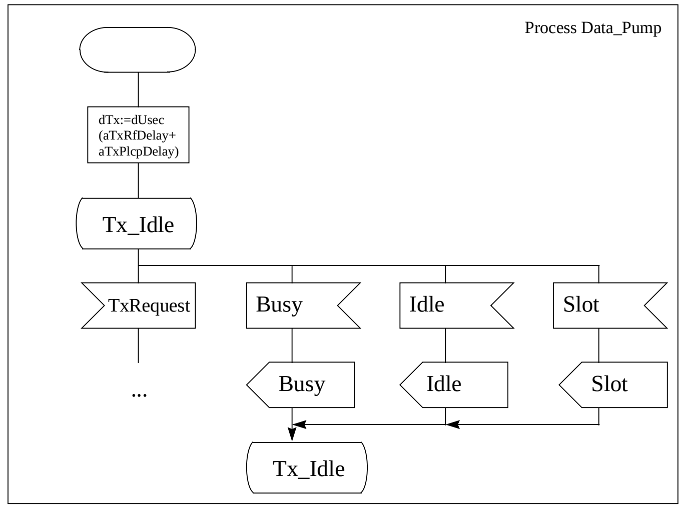

Figure 5: Process Block

Referring to Figure 5, the implementation of a process block corresponds to objects available within the OPNET Process Editor such as initial states, states, and transitions. SDL signals would obviously be implemented using OPNET interrupts combined with state transition conditions. A task would be implemented using Proto-C code in state enter/exit executives or transition executives. Figure 6 summarizes these mappings. SDL has other objects to model more complex behavior but we have found that they can all be implemented with ease in a manner similar to the objects discussed.

SDL Objects	OPNET Objects	
Block	Node	ND
Process	Processor, Generator, Queue, Receivers, Transmitters, Antenna	
Start	Initial State	st_0
State	State	st_0
Input Signal, Input Signal, Output Signal, Output Signal	Interrupt, Transition Condition	
Task	Proto-C code in Entry/Exit Executive, Transition Executive	st_0 Entry Executive Exit Executive

Figure 6: SDL to OPNET Object Mapping

While metrics of programming errors were not collected (this was not the focus of the research effort) mapping the objects in the formal specification to OPNET objects seemed to greatly reduce the number of logical errors as well as reduce development time. Development of this complex system from initial design through debugging and model validation took approximately 250 man-hours. This time included learning OPNET. In addition, since we used the formal specification of IEEE 802.11, we had the added assurance that the model would indeed be a valid IEEE 802.11 implementation.

THE MODEL

The IEEE 802.11 model is to be used to conduct research into the real-time capabilities of IEEE 802.11. The model implements the distributed coordination function (DCF) of IEEE 802.11. Stations within model form an ad-hoc network (i.e., an independent basic service set (IBSS) in IEEE 802.11 terms).

Each station can generate three classes of data packets: hard real-time (i.e., packets with deadlines that cannot be missed), soft real-time (i.e., packets with deadlines that can be met within a certain tolerance), and normal data packets with no deadlines. As currently implemented, each class of data packets has up to three independent input streams that can be specified to simulate different applications running on the station. The arrival, service, and packet deadline distributions can be any of the predefined distributions supported by OPNET. Hard real-time packets are transmitted until the hard real-time queue is empty, then soft real-time packets are transmitted. Finally, normal data packets are sent. The queueing discipline is currently first-come-first-served (FCFS), although we plan to enhance the model to support several other disciplines.

The model is highly parameterized; in an individual station, over 20 different IEEE 802.11 parameters can be varied. It would be impossible to discuss each parameter in detail here but examples include the transmission rate, the idle period slot time, the packet length (which can be fixed or described stochastically), and the radio transmit to receive mode switch time. All parameters are set to the default values (where applicable) specified in the IEEE 802.11 standard. If one parameter is changed and that change affects the value of another parameter, this change is automatically calculated and dynamically updated in the model prior to simulation execution.

We believe that one way to improve the real-time performance of IEEE 802.11 is to change the backoff algorithm. We are in the process of modifying the model to allow the user to select from a variety of different backoff algorithms.

One of the characteristics of wireless LANs is the large number of errors introduced in the channel. We are incorporating a Gilbert model [Gil60] into the radio transceiver pipeline to induce bursty bit errors in the transmitted packets. This type of bit error model allows us to directly specify the BER of the channel as well as the burstiness of the errors. This model is, conceptually, a higher-level model than that which is supported in the default radio transceiver pipeline, but it can very easily and seamlessly be incorporated into it.

The model collects several specialized statistics including the mean number of transmission attempts until successful transmission, the mean access delay until a waiting packet accesses the channel, and percent of packets that meet/miss their deadlines. Statistics that are automatically supported by OPNET such as throughput, queue size, etc. are available as well.

MODEL VALIDATION

The model was validated by comparing the performance metrics of our model against those obtained in [BiF96]. Because [BiF96] was based on an earlier draft, several of that paper’s parameters did not match those in the latest IEEE 802.11 standard. For the validation, we changed the values of these parameters in our model to match [BiF96].

Figure 7 shows throughput for 5, 10 and 20 station networks. Figure 8 shows the average transmission attempts per packet for various values of the contention window, CWmin, and CWmax. Figure 9 shows saturation throughput versus number of stations for various values of the contention window, CWmin, and CWmax.

As can be seen from these three figures, the agreement between the two models is quite good. In fact, the data obtained in the two uppermost plots in Figure 8 were so close to [BiF96] that our data and their data virtually overlap. The agreement of other performance metrics is similar.

CONCLUSION

In this paper, we demonstrated the advantages of implementing a simulation model using a formal specification of the system mapped to OPNET objects. The validation data showed excellent agreement between our model and other published results. In addition, using this approach we were able to construct this model in a short time with no prior experience using

OPNET. Our model is highly parameterized and can model a wide range of input traffic. We expect that by modifying the default backoff algorithm in IEEE 802.11 we can significantly enhance its real-time performance.

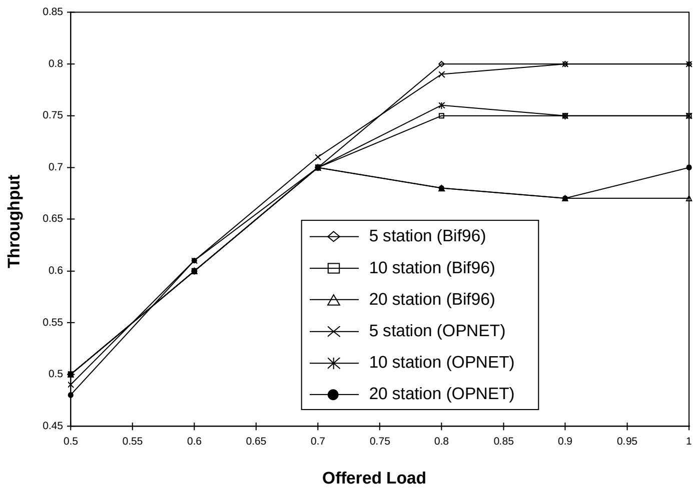

Figure 7: Throughput versus Offered Load

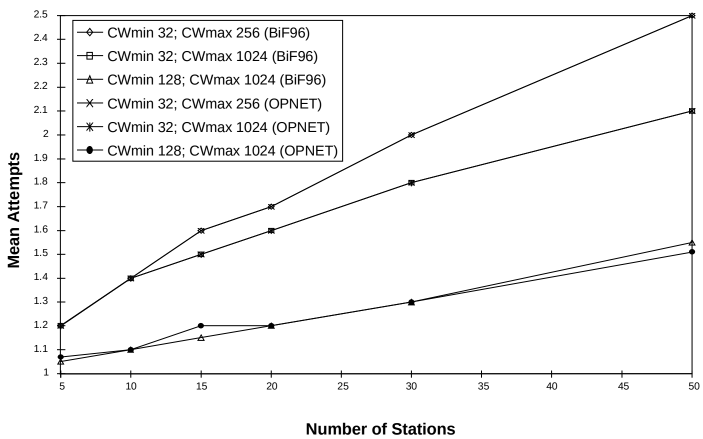

Figure 8: Mean Attempts per Packet

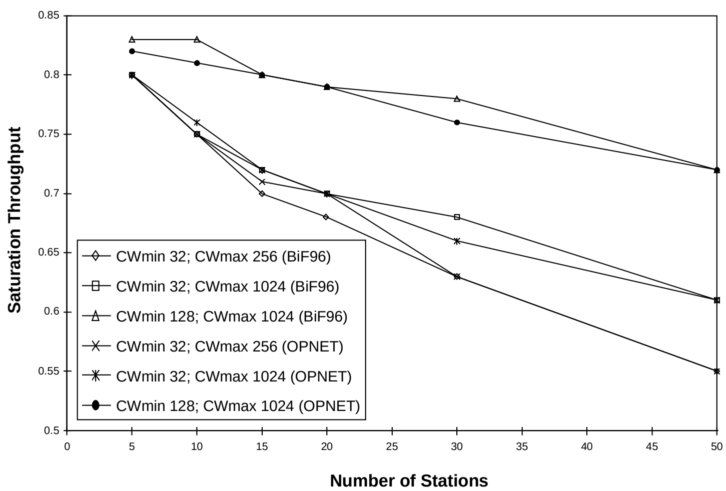

Figure 9: Saturation Throughput

REFERENCES

BiF96 G. Bianchi, L. Fratta, M. Oliveri, “Performance Evaluation and Enhancement of the CSMA/CA MAC Protocol for 802.11 Wireless LANs,” *The Seventh IEEE International Symposium on Personal, Indoor and Mobile Radio Communications PIMRC ’96*, pp. 392-396, Oct. 1996.

EHS97 Jan Ellsberger, D. Hogrefe, A. Sarma, *SDL, Formal Object-Oriented Language for Communicating Systems,* Prentice-Hall Europe, Hertfordshire, UK, 1997.

Gil60 E. N. Gilbert, “Capacity of a Burst-Noise Channel,” *Bell Systems Technical Journal*, Vol. 39, pp. 1253-1265, 1960.

IEE97 Editors of IEEE 802.11, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, Draft Standard 802.11*, *P802.11/D6.1*, Institute of Electrical and Electronics Engineers, Inc., New York, May 9, 1997.