A Reactive Channel Model for Expediting Wireless Network Simulation

Chunyu Hu
Dept. of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Urbana, IL 61801
chunyuhu@uiuc.edu and

Jennifer C. Hou
Dept. of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
jhou@cs.uiuc.edu

ABSTRACT

A major problem with leveraging event-driven, packet-level simulation environments, such as *ns2* [15], *J-Sim* [1], *OpNet* [3]), and *QualNet* [4]), in conducting wireless network simulation is the vast number of events generated, a majority of which are related to signal transmission. Due to the broadcast nature of a wireless channel, transmission of a signal has to be received and processed by all nodes operating on the same channel (and neighboring channels if co-channel interference is taken into account). This implies that one signal transmission event will trigger numerous signal arrival notification events.

In this paper, we investigate the operations of signal transmission in the various stages: *signal propagation*, *signal interference*, and *interaction with the PHY/MAC layers*, and identify where events can be reduced without impairing the accuracy. We observe that for each instance of signal transmission, a large number of signal arrival events are generated to notify nodes in the interference range of the signal. A majority of them are, however, redundant, as only nodes in the *receiving* state or the *idle* state but intend to transmit will be directly affected by the signal and need be informed. We thus propose to leverage the MAC/PHY state information, and devise (from the perspective of network simulation[1]) a reactive channel model (RCM) in which nodes explicitly *register* their interests in receiving certain events according to the MAC/PHY states they are in and the corresponding operations that should be performed. The simulation study indicates that RCM renders an order of magnitude of speed-up without compromising the accuracy of simulation results. The memory required in keeping the extra state information, on the other hand, is minimal. This, coupled with the fact that there is no need to re-design the channel model for each specific MAC layer, and the modification made in the MAC/PHY layer is quite modest (e.g., a few API changes), makes RCM a light-weight candidate mechanism for expediting wireless network simulation.

[1]The channel model discussed in the paper refers to the series of operations carried out in signal transmission/reception, and should not be confused with the channel model that models the physical characteristics in wireless communications.

Categories and Subject Descriptors

I.6.5 [**Simulation and Modeling**]: Model Development; I.6.4 [**Simulation and Modeling**]: Model Validation and Analysis

General Terms

Algorithms, Performance

Keywords

Network Simulation, Channel Model, Scalability, Reactive

1. INTRODUCTION

With the proliferation of portable computing platforms and small wireless devices, wireless networks have received more and more attention as a means of data communication among untethered devices. As the communication activities in the shared wireless medium are quite complex and affected by several factors across the protocol stack (e.g., from signal attenuation and fading in the PHY layer, contention and collision in the MAC layer, up to TCP congestion control in the transport layer), they do not lend well to theoretical analysis. It is not unusual that performance analysis can be rigorously made by focusing only on one protocol function, while leaving out (sometimes subtle) protocol details in the other layers. As a result packet-level, event-driven simulation studies are usually carried out to better study the performance of wireless networks.

Several simulators are currently available in literature that either include wireless extensions from their wired network counterpart (e.g., *ns2* [15], *J-Sim* [1], and *OpNet* [3]) or are built with wireless networks as the major simulation domain (e.g., *QualNet* [4]). A major problem with event-driven, packet-level simulation in such simulators is the vast number of events generated, a majority of which are related to signal transmission. Due to the broadcast nature of a wireless channel, transmission of a signal has to be received and processed by all nodes operating on the same channel (and neighboring channels if co-channel interference is taken into account). This implies that one signal transmission event will trigger numerous signal arrival notification

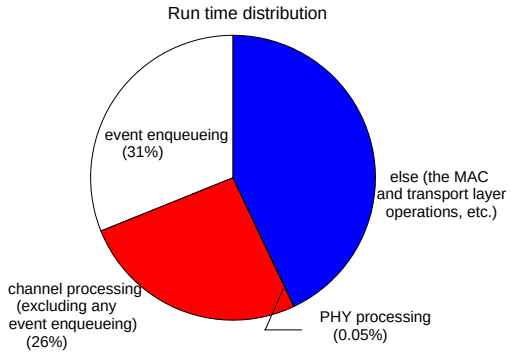


Figure 1: Proportion of the execution time that is spent on event enqueueing in a 100-node ad hoc network over a 1000x1000m^2 field. A total of 40 CBR connections are established in the network (with their source and destination nodes randomly selected) and carry a total of 120 packets/second traffic (where the packet size is 512 bytes).

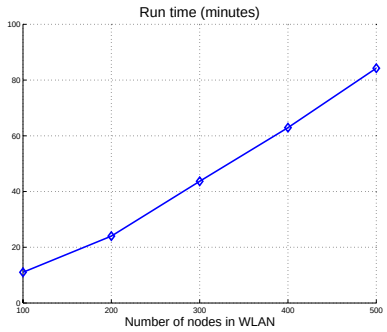


Figure 2: Execution time required to carry out a 60-second simulation of a WLAN with n nodes. $n = 100, \ldots, 500$. Each node generates 0.5Mbps CBR traffic with the packet size set to 25 bytes.

events. It takes non-negligible time to enqueue these events according to their event firing times. Processing each of the signal receipt events takes, in addition to all the operations performed for packet transmission/receipt in wired networks, the following operations: (i) calculation of the signal attenuation (based on the antenna propagation model and terrain/obstacle model used); and (ii) whether or not the received signal is strong enough (with the effects of all the concurrent transmission activities taken into account) so that the corresponding frame can be considered received correctly. As the CPU time required is roughly proportional to the number of events that have to be executed, simulation of wireless networks at packet level easily becomes computationally expensive or even intractable.

To understand quantitatively the gravity of the problem, we consider a WiFi scenario in which n nodes operate in a 802.11-operated wireless LAN, each of which sends CBR traffic at the rate of 0.5 Mbps (with packet size set to 25 bytes). As shown in Fig. 2, it takes 662 seconds (5056 seconds or 1.4 hours) in real time to carry out a 60-second simulation run in the case of $n = 100$ ($n = 500$). This problem is not unique to WLANs of heavy loads. Consider the simulation of an ad hoc network of 100 nodes in a 1000 $\times$ 1000 m^2 field. 40 CBR connections exist in the network (with their source and destination nodes randomly selected) and carry a total of 120 packets/second traffic (where the packet size is 512 bytes). It takes 125 seconds run time to carry out a 60-second simulation on average. Moreover, as shown in Figure 1, approximately 552,600 events per second are generated, 38.4% of which are generated to notify signal arrivals (i.e., signal arrival notification events). Moreover, 31% of the time is spent on enqueueing events alone. In spite of all the limitations of carrying out simulation for wireless networks, the need for large-scale simulation of wireless networks (and hybrid networks where wired and wireless networks coexist) is increasing [9]. How to improve the simulation performance while not compromising the accuracy has become a pressing issue.

In this paper, we investigate the operations of signal transmission in the various stages: *signal propagation*, *signal interference*, and *interaction with the PHY/MAC layers*, and identify where events can be reduced without impairing the accuracy. We observe that for each instance of signal transmission, a large number of signal arrival events are generated to notify nodes in the interference range of the signal. A majority of them are, however, redundant, as only nodes in the *receiving* state or the *idle* state but intend to transmit will be directly affected by the signal and should be informed. The former nodes need to be informed, as they have to calculate whether or not the signal (taken as part of the noise) is strong enough to induce collision. The latter nodes need to be informed, as they need to monitor the channel status if they are in the back-off stage. With this key observation, we propose a *reactive channel model* (RCM) in which nodes *register* their interests in receiving certain events according to the MAC/PHY states they are in. With the judicious use of the MAC/PHY state information, only nodes whose activities will be affected by a signal transmission event will be notified of (and hence process) the event. Several issues have to be adequately addressed in order to realize the notion of RCM. For example, when a node intends to transmit (and hence has to sense the channel) at time t, it has to know not only all the signal transmission events in its vicinity from the time instant t on, but also *on-going* transmission events that start prior to time t. We address each of the issues and devise light-weight solution methods. We also evaluate analytically and via simulation the performance gain of the proposed channel model in terms of the number of events thus reduced and the memory usage thus incurred. We observe one order of magnitude of improvement in the number of events (and hence in the execution time required to carry out simulation). The memory usage incurred in keeping the extra state, on the other hand, is minimal.

How to expedite wireless network simulation has not been extensively addressed in literature. In spite of its success in memory systems simulation, parallel simulation with the use of conservative methods [5, 10] has been shown to be rather ineffective in expediting network simulation [8]. This is due to the fact that a simulation engine has to synchronize with others and hence cannot advance or lag behind by more than the so-called *lookahead* time. This lookahead time is con-

strained by the delay between two subsets of wireless entities whose events are executed on two simulation engines. Unfortunately as wireless network simulation is usually carried out at the granularity of signal levels (in order to capture all the wireless physical characteristics), the lookahead time is usually in the microsecond or even smaller scale. This, coupled with the fact that any two subsets of wireless entities cannot usually be "loosely" decoupled due to the broadcast nature of wireless media, diminishes the benefit of parallel simulation. The only advantage of leveraging parallel simulation for wireless network simulation is perhaps the memory space made available on multiple machines.

Efforts have also been made on using the caching technique to improve computation efficiency. For example, in J-Sim [1] the path losses that have been previously calculated are cached and indexed for future reuse. The staged simulation [16] further explores the possibility of reusing previous computation results in the same run and across different runs of simulation. Another effort made is to exploit the notion of *fluid model-based simulation* [7] in which a large number of packets are abstracted as a single fluid chunk, and analytical models are developed to fully characterize their operational behaviors *at the MAC layer* in IEEE 802.11-operated WLANs. Fluid model-based simulation is then realized with the use of the time stepped simulation technique [17]. Two orders of magnitude improvement have been reported in incorporating fluid models to expedite simulation (at the cost of losing packet-level dynamics). All the aforementioned research complements RCM and can be integrated to improve simulation efficiency. The work that comes closest to our is that by Ji *et al.* [6]. They proposed a mechanism, called *Lazy event Scheduling and Corrective Retrospection* (*LSCR*), to reduce the number of events generated in signal transmission. (We will give a detailed description of LSCR in the taxonomy given in Section 2.) Although LSCR has been shown to significantly reduce the number of events, the corrective retrospection phase has to be tightly coupled with the MAC layer. As a result, the CR phase has to be redesigned for *each* specific MAC protocol. Also, as will be shown in Section 4, the extra state that has to be kept to facilitate retrospective correction introduces a 25% increase in memory usage at times.

The rest of the paper is organized as follows. In Section 2, we investigate the various stages of signal transmission in the conventional channel model and identify the surplus events that can be eliminated without compromising the accuracy. A taxonomy of related work is also given there. In Section 3, we present the proposed reactive channel model, and elaborate on its detailed operations. This is followed by a performance study in Section 4. Finally, we conclude the paper in Section 5 with a list of research avenues for future work.

2. AN INVESTIGATION OF THE WIRELESS CHANNEL MODEL

The channel model is an important component in a wireless network simulation environment. It is a crucial element that decides *validity*, *accuracy* and *speed* of the simulation. To facilitate our investigation on where events can be further reduced, we have "decomposed" (from the perspective of network simulation) the operation of signal transmission/reception in the channel model into three stages: *signal propagation*, *signal interference* and *interaction with the PHY/MAC layers*. (The "decomposition" may, however, differ slightly from one simulation environment to another, depending on the level of details that a simulation environment emulates and the architectural layout of the simulator.) In what follows, we elaborate on the events generated/processed in each stage.

Signal propagation
In this stage, a propagation model is used to describe how a radio signal is attenuated and distorted along the path from a transmitter to a receiver. Several commonly used propagation models are the free space model, two-ray ground model [11], and irregular terrain model [12]. They capture the statistical wireless characteristics of the field. The complexity involved in computing the path loss and the transmission/interference range (as determined by the propagation model) have a direct effect on the simulation efficiency. However, the choice on the propagation model cannot be made simply for the sake of reducing the execution time of simulation runs, but rather depends on the physical characteristics of the field to be simulated.

Signal interference
In this stage, an interference model is used to determine whether the signal can be correctly received even in the presence of other signals and noise. A commonly used criterion is the *Signal-Interference-Noise-Ratio* (SINR) threshold [14]. As SINR considers the effect of *accumulated interference*, a signal of weak signal strength can not be ignored. Takai *et al.* [13] have shown that simulation that fails to consider the effect of accumulated interference can produce dramatically different results with respect to system throughput, packet delay and fairness. This implies that in the worst case all the nodes operating on the same channel as the sender node have to be notified of the signal transmission event. The number of events generated is even larger in mobile ad hoc networks, which employ fairly complicated MAC mechanisms to avoid collision. The cost incurred in processing each event includes event enqueueing (i.e., the event has to be inserted into a proper position of the event queue), calculation of the propagation delay and the ultimately received signal strength, and possibly state transition of the PHY/MAC layers. As shown in Figure 1, although event enqueueing is purely simulation overhead, it constitutes a non-negligible portion of the computation time.

To reduce the number of events in this stage, one effective method is to limit the distance that a signal propagates. This limit is called *the propagation limit*. The larger the propagation limit, the more accurate the simulation results will be but the longer it takes to run the simulation. Clearly, there is a tradeoff between the simulation accuracy and the speed-up. The impact of the propagation limit on the inaccuracy has not been explored till [6] in which an upper bound on the inaccuracy introduced by the propagation distance is derived assuming a CSMA-based MAC protocol.

Another promising technique is *location management* and it aims to expedite the process of searching for nodes within the propagation limit. Grid-based location management is perhaps the most frequently used and has been deployed

in several popular network simulation environments such as *ns2*, *J-Sim* and *QualNet*. The field is divided into grids, and the search for nodes within the propagation limit is limited within several adjacent grids. (Note that the *node aggregation* approach proposed in [8] achieved similar objectives.) JiST [2] further refined the grid location management with a hierarchical grid structure, which divides the network field recursively both in the horizontal and vertical directions, and uses the tree structure to store grid coordinates. Nodes reside in the leaves of the tree. In this manner, location updates can be performed in constant amortized time, and the time incurred in searching a list of neighbor nodes is proportional to the list size. Naoumov *et al.* [9], on the other hand, proposed a list structure to keep track of nodes in the ascending order of their X-coordinates. The extra cost incurred in all the location management methods is the events needed to update the grid/list-location of a node when it moves.

Interaction with the PHY/MAC layers

In this stage, a mechanism has to be devised to instrument how the PHY/MAC layers of a wireless node retrieves from the channel the information of a signal. The information includes the arrival time, the duration, and the signal strength. Conceptually, signal transmission is composed of two steps: the transmitter node injects the signal to the channel, and the channel notifies the receiver nodes within the propagation limit (as determined in the second stage). The first step is simply a function call. How to realize the second step is, however, more involved, and in our opinion, is predominant in reducing the simulation overhead. The simplest method is for the channel model to schedule and enqueue, for each receiver node (identified in the second stage) an event that will be triggered at the respective arrival time. It is then up to each receiver node to determine, *after* processing the event, whether or not it will be affected by the event. This is the conventional method used by most, if not all, of the simulators. However, as will become clearer in Section 3, this method generates a large amount of surplus and unnecessary events.

In spite of the potential for significant improvement in simulation efficiency, there has not been much research along this direction. The only known work is perhaps *LSCR* [6]. In the LSCR mechanism, the channel model only schedules and enqueues signal arrival notification events for nodes within the *transmission* range. For nodes that are within the propagation limit but outside the transmission range, the channel model merely records the signal information in the signal history at each receiver node. The signal history provides sufficient information for a node to look up whenever necessary. As such, the number of events generated in each signal transmission is reduced.

As no signal arrival notification events are scheduled for in a node outside the transmission range, the node may fail to operate correctly. For example, in the case that IEEE 802.11 DCF is used as the MAC protocol, a node is allowed to transmit, only if the medium is sensed idle for a specified time interval, called the *distributed inter-frame space* (DIFS). If the medium is sensed busy, a random backoff interval value is uniformly chosen in $[0, \widehat{CW} - 1]$ and used to initialize the backoff timer, where $\widehat{CW}$ is the current contention window. The backoff timer is decreased as long as the channel is sensed idle, stopped when data transmission is in progress, and reactivated when the channel is sensed idle again for more than DIFS. As the node outside the transmission range of a signal is not notified of the signal arrival, it cannot infer the channel status correctly. A second technique, called *Corrective Retrospection*, is then used to retrospectively correct premature backoff timeouts. For corrective retrospection to operate correctly, it requires a complete knowledge of the MAC protocol operations in order to parse the signal history. This implies, for each specific MAC protocol, the corrective retrospection phase has to be re-designed. It is also quite difficult to determine when to clear the signal history recorded at each node.

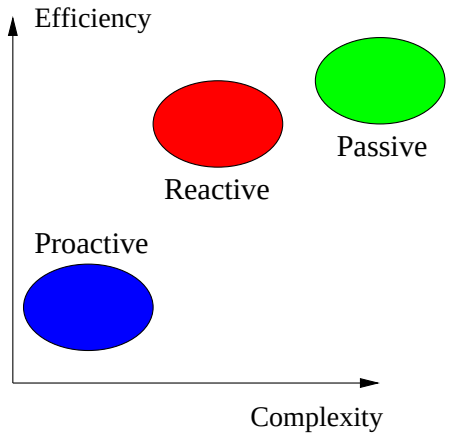


Figure 3: The design space for the wireless channel model (from the perspective of simulation).

As compared to the conventional channel model that schedules and enqueues signal arrival notification events for *all* the nodes within the propagation limit, LSCR is on the other extreme — it schedules events only for nodes within the transmission range, and takes complicated retrospective, corrective measures for nodes which can be potentially affected within the propagation limit. We therefore characterize them as *proactive* and *passive*, respectively. As shown in Fig. 3, while the complexity of the conventional channel model is low, it suffers from high simulation overhead (i.e., the number of surplus events scheduled). LSCR, on the other hand, generates a significantly less number of events, but the retrospective correction required to ensure accuracy is quite complex. Now the question is, can we devise a mechanism that schedules signal arrival notification events for all the nodes that will be truly affected, thus eliminating any corrective measure that entangles with specific MAC operations. The answer is affirmative, and we will discuss in Section 3 how we achieve this by exploiting the MAC information (i.e., the MAC/PHY state each node is in) to determine whether or not a node should receive certain signal arrival notification events.

3. THE REACTIVE CHANNEL MODEL

3.1 Overview of the Model

We design the reactive channel model with the following objectives. First, the model should not require alteration of the MAC/PHY operations or a tight, intrinsic coupling with the latter. For example, the backoff procedure (in terms of when the back-off timer should start, pause, resume and expire) as stipulated by the IEEE 802.11 MAC protocol will remain intact as it should be. Second, the details of how interference takes place and its effect on the transmission activities of each node are faithfully captured.

The reactive channel model receives the signal transmission event injected by a node and dispatches the signal arrival notification events to nodes in a recipient set V, where $V \subseteq V_L$ and V_L is the set of nodes located within the propagation limit L of the signal (denote the area by A_L). The set of V is composed of two subsets: (i) V_1 is the set of nodes chosen by the channel; and (ii) V_2 is the set of nodes that *explicitly* register to receive signal arrival notification events.

To ensure that V contains only nodes whose activities will be affected by the signal transmission, we take into account of both physical and MAC characteristics. First, nodes that are within the transmission range of a signal (denoted by A_R) will almost surely be affected. They may either start to receive this signal, or are themselves in the receiving state (of some other signal) and have to check whether or not this signal causes any collision. The only exception is when the node operates in the half-duplex mode and is in the transmitting state. The exception, however, does not occur frequently, as most MAC protocols operate on the principle of *listen-before-talk*, i.e., the sender node has to sense the channel and ensure it is idle before it initiates a transmission. As a result, unless two signal transmissions take place within the time interval of the order of end-to-end propagation delay, the case in which a node that is currently in the transmission state is also within the transmission range of another signal does not occur frequently. Due to this reason, all the nodes in A_R are enclosed in V_1.

Second, not all the nodes within the propagation limit but outside the transmission range of a signal need to be notified of the arrival of the signal. Rather, this is dictated by the MAC/PHY state the node is in and the specific MAC operations in each state. Specifically, a wireless node may be in one of the following states: *transmit*, *receive*, *idle*, *sleep* and *power off*. Obviously a node in the sleep or power off state is "disconnected" from the network. A node that is in the transmitting state can not receive at the same time (under the half-duplex mode). Thus there is no need to notify nodes in these states of the signal arrival. On the other hand, a node in the receive or idle states has or may have to sense the wireless medium for the following reasons. A node in the idle state has to carrier-sense the channel according to certain contention-avoidance procedure, *if* it intends to transmit a frame. For example, as mentioned in Section 2, a node that operates under IEEE 802.11 MAC DCF has to ensure the medium has been sensed idle for a DIFS, before it can transmit. That is, the node has to obtain the information on all the *on-going* transmissions. If the medium is sensed busy, the node sets up a backoff timer with the backoff interval drawn from a uniform distribution $[0, \widehat{CW} - 1]$. The backoff timer is decreased as long as the channel is sensed idle, stopped when data transmission is in progress, and reactivated when the channel is sensed idle again for more than a DIFS. In this case, the node has to continuously receive signal arrival notification events (if any). A node in the receive state has to *continuously* monitor the wireless medium to keep track of the interference level throughout the duration of the signal reception. In this way, the receiver node is able to detect if a collision occurs and/or if the signal being received is corrupted because the accumulative interference exceeds certain threshold. In this case, the node has to continuously receive signal arrival notification events.

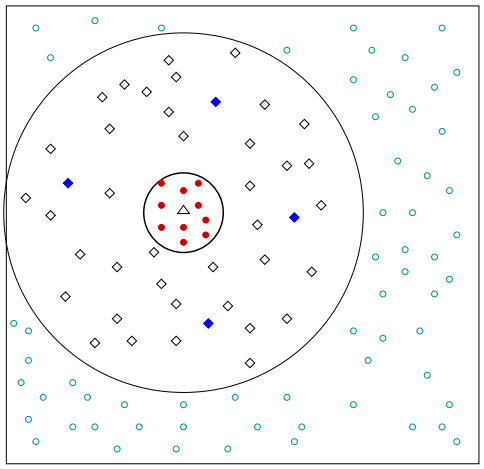
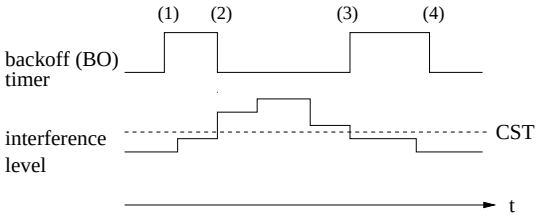


Figure 4: A snapshot of a network upon a signal transmission under the reactive channel model

As the MAC layer of a wireless node knows best which MAC/PHY state the node is in and what operations have to be performed in certain state, it is a natural candidate for determining whether or not the node should be notified of signal arrivals. Thus, we argue that nodes should leverage the MAC information and *explicitly* register and de-register for receiving signal arrival notification. That is, V_2 contains nodes that are within the propagation limit but outside the transmission range, *and* explicitly register to receive signal arrival notification events. Figure 4 depicts such a scenario. In what follows, we discuss the several APIs that the channel model exports to facilitate registration of notification requests.

3.2 Interfaces between the Channel Model and the MAC/PHY Layers

We instrument the channel model to provide the following APIs, *transmit*, *schedule*, *register/deregister* and *inquiry*:



(1) inquire the medium status: IDLE; start the BO timer; register
(2) freeze the BO timer; deregister
(3) resume the BO timer, register
(4) the BO timer times out; deregister

Figure 5: An example that illustrates when the MAC layer registers/deregisters during a backoff procedure under the reactive channel model

- *transmit* is the function call provided for nodes to inject signals into the channel.
- *schedule* is the call taken by the channel to schedule and enqueue the signal arrival notification events in the event queue for nodes in V.
- *register* and *deregister* are the calls that a node invokes to add/remove itself into/from the signal recipient set V.
- *inquiry* is the call that a node invokes in order to obtain a list of *on-going* signal transmissions.

The reason for providing the *inquiry* interface is as follows. When a node becomes interested in sensing the wireless medium, it may have already missed several notification events that contain information of signal transmissions that started earlier and go beyond the current time. With the *inquiry* call, a node can acquire a complete list of current signals-on-air.

To illustrate how these APIs are used, we continue our IEEE 802.11 MAC example. When a node is in the receiving state, it expresses its interest in receiving signal notification events by (i) acquiring a list of current, on-going signal transmissions to calculate the current interference level; and (ii) registering itself to *continuously* receiving event notification. By the end of the duration of the received signal, the node deregisters itself. Similarly, when a node is in the idle state and intends to transmit, it continuously senses the channel by first *inquiring* the channel to calculate if the current interference level is below the sensibility threshold (denoted by CST). If yes (i.e., the channel is sensed idle), the node starts its backoff timer and continuously senses the channel by registering itself with the channel until the backoff timer freezes, i.e., when the channel is sensed busy. When the backoff timer resumes (i.e., the channel has been idle for more than a DIFS), the node registers itself again. (Note that when the channel will become idle can be calculated with the information on the all the signal durations.) The process repeats until the backoff timer expires. The time instants when a node in a backoff procedure registers and deregisters are depicted in Fig. 5.

3.3 Implementation

In this subsection, we elaborate on several implementation issues. Although the reactive channel model can be implemented in any discrete-event driven wireless network simulators, we use, wherever needed for clarity of presentation, the wireless network extension in *ns-2* (that comes along with the latest ns-2 release, ns-2.27) distribution as the reference architecture.

Recall that a major difference between the RCM and the conventional channel model lies in the signal recipient set. Under the RCM, each wireless node is attributed with a new state variable that indicates whether it registers to receive signal arrival notification events. When a signal is transmitted, the channel *schedules* such events only for (i) nodes located in A_R and (ii) nodes located in $A_L \setminus A_R$ *and* currently registered.

A key decision has to be made on where to implement the *inquiry* service. A closely related question is where to store the list of signals on-air at any time instant. The most straightforward (and centralized) method is for the channel to store and maintain a list of signals on-air and to respond to inquiries. In spite of its simplicity, this method suffers from the problem of effectively calculating propagation delays. Note that propagation delays vary with the distance between transmitter-receiver pairs. Every time the channel receives an inquiry from a node, it has to calculate, for each signal on the list, the propagation delay and the perceived signal strength. Signals that have not arrived, have ended, or originate outside the propagation limit will not be counted for. In the case that inquiries are frequently made by a node, the channel may have to repeat the same calculation multiple times. To the end, we decide to take a distributed approach and enable the channel to *silently* store *in the PHY layer of each node* the information of signals that originate within the propagation limit. The information includes the arrival time, the duration, and the received signal strength, all from the perspective of each wireless node. Note that the distributive method does not consume more memory because under the conventional channel model, all the signals that are received have to be stored in the MAC layer anyway (so as to facilitate the calculation of interference level).

The procedure that a channel takes to process signal transmission is as follows. When the channel receives a signal transmission event injected by a node, it schedules, for each node in the signal recipient set $V \supseteq V_L$. For the rest of nodes in $V_L \setminus V$, the channel silently stores the signal information in a local list in the PHY layers of these nodes. Note that there is no need to store a copy of the packet carried in the signal, as the signal is not strong enough to be received as a packet. As no events are scheduled (no timers are set up to announce the arrival and ending of the signal) for nodes in $V_L \setminus V$, these signals are invisible to their MAC layer (unless they are later requested explicitly). Signals that have ended will be removed from the local list in the PHY layer of a node whenever a new signal is inserted and/or an inquiry is made, thus keeping the list "slim." With the signal information stored in the PHY layer of each node, the *inquiry* interface is actually implemented between the PHY layer and the MAC layer.

One caveat of storing the signal information in the PHY layer of a node is that as a signal is inserted into the local list in the PHY layer earlier than the time instant when the signal actually arrives at the node. When a node registers its interest of receiving signal arrival notification events, the PHY layer has to check if there is any signal that has not yet arrived and schedules a notification event on behalf of the channel.

3.4 Discussion

Usually the number of events generated in, and the execution time required to carry out, a simulation run are used to evaluate mechanisms with respect to their scalability. To perform a rough analysis on the number of events reduced with the use of the RCM, we consider the case in which N wireless nodes are distributed uniformly in an area of size $F \times F$. The area size is large enough so that the edge effect can be ignored. The node density is then $\alpha \triangleq N/F^2$. Under the symmetric propagation assumption (i.e., both A_R and A_L are circles with radius R and L, respectively), the number of nodes for which the channel schedules signal arrival notification events is $\Theta(\alpha\pi L^2)$ under the conventional channel model, and $\Theta(\alpha\pi R^2 + \beta\alpha\pi(L^2 - R^2))$ under the reactive channel model, where β is the average percentage of nodes that are in the idle states but intend to transmit or in the receive states. Thus a total of $\Theta(\alpha\pi(1-\beta)(L^2 - R^2))$ events is saved for each signal propagation. The saving could be considerable if β is much smaller than 1 and/or L is much larger than R. The latter is true since to account for the accumulative interference effect, L has to be much larger than distCST and distCST is larger than R. Table 1 lists the specifications of several typical wireless cards and confirms the fact. The value of β depends on the traffic load and distribution. As will be seen in Section 4, the number of events reduced is in the range of one order of magnitude.

	Tx Range	Receiver Sensibility	distCST*	L†
ORiNOCO	160m	-82dBm	400m	1200m
Aironet 350	244m	-85dBm	632m	1896m
SMC2336W-AG	300m	-82dBm	400m	1200m

* calculated using two-ray model
† L is set to $3 \times distCST$
(All data are obtained when the cards operate at 2.4GHz, IEEE802.11b, 11Mbps)

Table 1: Specification of Several Wireless Cards

Although the RCM reduces the number of events and hence the execution time incurred in enqueueing the events, it also incurs computation costs in registration/deregistration, maintenance of the signal lists in the PHY layers of wireless nodes, and calculation of of the interference levels. The RCM is scalable only if the saving in event scheduling outweighs the additional computation costs. As it is difficult, if not impossible, to analytically derive these costs, we will carry out simulation studies in Section 4 to quantify the performance of the RCM.

4. PERFORMANCE EVALUATION

In this section, we conduct *ns-2* simulation to evaluate the efficiency of RCM and compare it against the conventional channel model and LSCR. In the results illustrated below, the three channel models are referred to as *Conventional*, *RCM* and *LSCR*, respectively. As the current release of *ns-2* does not take into account of the effect of accumulative interference, we first discuss how we modify *ns-2* to include that effect so as to improve the simulation fidelity. Then we present the simulation study. Several system parameters, i.e., the propagation limit, the node density, the field size and the traffic load, are varied to study the performance under different scenarios. The following two metrics are used as the performance indices: (i) the average execution time required to carry out 1-second simulation; and (ii) memory usage (in units of MB).

The default physical layer parameters used in *ns-2* are used in the simulation and listed in in Table 2. Under this setup, the nominal transmission range R is approximately 250m and the sensibility range $distCST$ is 550m. Unless stated otherwise, the propagation limit is set to $3 \times distCST$, i.e., 1650m. IEEE 802.11 DCF with the RTS-CTS floor acquisition mechanism is used as the MAC protocol, and DSR is chosen as the underlying ad-hoc routing protocol. Grid-based location management is used with the grid size set to 200 m. Both TCP/FTP and CBR traffic is used, but due to the fact that results exhibit similar trends under both types of traffic, we report only results under CBR traffic (with packet sizes set to 512 bytes). For each connection, the source and destination nodes are randomly selected. Mobility may slow down simulation, but the effect is uniform to all the three channel models since the same location management is assumed. Thus, the results with mobility exhibit similar trends and we only present results conducted under static scenarios.

Propagation model	Two-Ray	Radio frequency	914 MHz
Data rate	1Mbps	SNRT	10 dB
RXT	-64 dBm	CST	-78 dBm
Antenna height	1.5m	transmission power	24.5 dBm

Table 2: Parameters for the physical layer

All simulation are carried out on a Dell Dimension 4600 computer with the following configuration: Dual CPU 3.0GHz, 1G RAM and Linux Fedora Core (kernel: 2.6.5-1.358smp). Each run of simulation lasts 300 seconds.

4.1 Modification Made in ns-2

The current release of ns2 (*ns-2.27*) employs a simple collision detection mechanism. The physical layer is a very thin layer and its only function is to inject the signal into the channel, pass the signals to the MAC layer at receivers, and update the energy. Signals with strength below the receipt threshold *RXThresh* are marked as corrupted. The MAC layer keeps the signal currently being "received", if any, and implements the collision detection mechanism. The effect of accumulative interference is not taken into account at all. A timer is scheduled for each signal passed to the MAC layer to notify the end of the signal if no other signal is being received. The MAC layer switches to the *MAC_RECV* state even the signal strength is below *RXThresh*. If another signal arrives in the interval during which a signal is being received, its strength will be compared to that of the signal

being received. If the new signal is strong enough, the timer is rescheduled for the signal that lasts longer and the MAC layer enters the *MAC_COLL* state; otherwise, the signal is simply discarded (without considering the accumulative interference effect).

To incorporate the effect of additive interference and use SINR as the criterion for determining whether or not a signal can be correctly received, we add in the MAC layer data structures that record the level of accumulated interference and the list of signals which are considered as interference. A signal whose SINR is below a pre-determined threshold or which arrives when the MAC layer is currently receiving some other signal will be considered as interference and is inserted into the list in the order of signal ending times. Its strength will be added to the interference level. A dedicated timer will be scheduled that triggers at the signal ending time. The signal at the head of the list will be removed at the time the corresponding timer expires and the interference level will be decreased accordingly.

4.2 Simulation Results

Figure 6 (a)–(d) depict the the average execution time required to carry out a 1-second simulation under three channel models as the node density ((a)), the traffic load ((b)), the field size ((c)), and the propagation limit ((d)) increases, respectively. The field size in both Figure 6 (a) and (b) is set to $2000 \times 2000m^2$. In the first scenario (Fig. 6 (a)), the number of nodes varies from 150 to 450 nodes. A total of 40 CBR connections are created, each carrying 3 packets/second traffic. In the second scenario (Fig. 6 (b)), 300 nodes carry a total of 50 CBR connections. The traffic rate of each CBR connection varies from 0.5 to 4 packets/second, and the throughput is maintained above 98%. In the third scenario (Fig. 6 (c)), the node density is fixed at $1/(100\text{x}100m^2)$, and the field size varies from $500 \times 500m^2$ to $3000 \times 3000m^2$ in steps of $500 \times 500m^2$. To keep the traffic load increases proportionally to the field size, 20% pairs of neighboring nodes are chosen randomly as the CBR sources and destinations, each pair carrying 2 packets/second traffic. In the fourth scenario (Fig. 6 (d)), 1000 nodes are randomly placed in a $5000 \times 5000m^2$ area. A total of 100 CBR connections are established, each of which carries 2 packets/second traffic. The propagation limit varies from $1 \times distCST$ to $6 \times distCST$.

As shown in Figure 6, the execution time under the conventional channel model increases dramatically as either of the system parameters (node density, traffic load, field size, and propagation limit) increases. The RCM achieves approximately an order of magnitude improvement in expediting the simulation over the conventional channel model. In the scenarios depicted by Figure 6 (a-b), the RCM performs slightly worse than LSCR; while in the scenarios depicted by Figure 6 (c-d) the RCM performs slightly better. This is attributed to two counteracting factors. On the one hand, LSCR only dispatches signal arrival notification events to nodes within the transmission ranges and thus achieves the most saving in the number of events. On the other hand, the corrective retrospection phase taken by LSCR has to parse the signal history, which involves sorting of signal starting/ending times. When the signal history is large, the corrective retrospection operation becomes expensive. Besides, the memory usage, if large to certain degree, will incur more page-in/out operations in the kernel. (We will further discuss this problem in Figure 8.) This is the case when the field size and/or the propagation limit are large. Several interesting findings are in order: first, the node density seems to have a larger impact on the performance than the traffic load. This observation is corroborated by the respective weights of these parameters in $\Theta(\alpha\pi(1-\beta)(L^2-R^2))$ (Section 3.4). Second, the improvement (in terms of the order of magnitude) exhibits a decreasing trend in Fig. 6 (b). This is because the increase in the traffic load implies more simultaneous transmission attempts are made, and hence more signal arrival notification events have to be scheduled under the RCM. Third, as shown in Fig. 6 (d), L plays perhaps the most significant role in the number of events generated. When L is equal to $6\times$ *distCST*, it takes 30 seconds to carry out one-second simulation under the conventional channel model, while it takes only approximately 5 seconds under RCM and LSCR.

To further understand where the performance gain results from, we measure the number of events generated per second, and the execution time spent on enqueueing all the events in the same scenario of Figure 6(a). As shown in Figure 7, both the number of events generated per second and the event enqueueing time are significantly reduced under RCM and LSCR.

Figure 8 shows the memory usage under the three channel models. We use the same scenario of Figure 6 (d) except that in Figure 8 (b) the propagation limit is set to $3 \times distCST$. As shown in Figure 8, LSCR consumes more memory than the other channel models, and the difference becomes more pronounced as L or the simulation time increases. The RCM, on the other hand, consumes slightly less memory than the conventional channel model. This is because in the RCM less signal arrival notification events are generated and consequently less packet copies are made. The signal information written into the local list at each node takes less space than the packet information. Signals that have ended are purged whenever a new signal is inserted or an inquiry is performed. Recall that in LSCR, the signal history is maintained at each node. A signal that is not received but originates within the propagation limit will be written into the signal history at a receiver node. The signal history will be used by a node which encounters a false backoff time-out. It is suggested in [6] that the signal history be cleared every time corrective retrospection is performed — the signals that ended before the time of corrective retrospection are removed. In our implementation, we follow the guideline, and in addition, add in the function *checkBackoffTimer()* the operation of purging the signal history as long as the backoff timer is not running. This function is called whenever the MAC state changes. Even with all the above refinements, LSCR incurs quite significant memory consumption. This, on the one hand, increases the time to perform the corrective retrospection; and on the other hand, may cause frequent page-in/out operations in the kernel and even memory thrashing.

In summary, we list the features of the three channel models in Table 3.

	Efficiency	*Complexity (Design & Implementation)*	*Reflect exact MAC behavior*	*Memory consumption*
Conventional	low	simple	yes	low
RCM	high	moderate	yes	lowest
LSCR	highest	high*	no	high

*Has to take corrective measures and adapt to different MAC protocols.

Table 3: Features of the three channel models

5. CONCLUSION

In this paper, we investigate the operations of signal transmission in the various stages: *signal propagation*, *signal interference*, and *interaction with the PHY/MAC layers*, and identify where events can be reduced without impairing the accuracy. We observe that for each instance of signal transmission, a large number of signal arrival notification events are generated to notify nodes in the interference range of the signal. A majority of them are, however, redundant, as only nodes in the *receiving* state or the *idle* state but intend to transmit will be directly affected by the signal and need be informed. We thus propose to leverage the MAC/PHY state information, and devise a reactive channel model (RCM) in which nodes *register* their interests in receiving certain events according to the MAC/PHY states they are in and the corresponding operations that should be performed. With the judicious use of the MAC/PHY state information, only nodes whose activities will be affected by signal transmission will be notified of (and hence process) the signal arrival event. This effectively reduces the number of events generated by $\Theta(\alpha\pi(1-\beta)(L^2-R^2))$ (ref. Section 3.4) and hence the execution time spent on enqueueing these events. The simulation study indicates an order of magnitude of speed-up without compromising the accuracy of simulation results. The memory required in keeping the extra state information, on the other hand, is minimal. Moreover, as the channel and the MAC/PHY layers communicate with well-defined APIs (*transmit*, *schedule*, *register/deregister*, *inquiry*), there is no need to re-design the channel model for each specific MAC protocol, and the modification made in the MAC/PHY layer is modest (i.e., enabling the MAC layers to register/de-register and inquire the list of on-going signal transmissions).

Although the performance improvement under the RCM is quite encouraging, it is by no means the end of this research avenue. As part of our on-going research work, we are investigating how to effectively combine hierarchical grid-based location management with RCM to expedite search of nodes which should be notified of certain signal arrival events, while leaving out nodes which should not.

6. REFERENCES

[1] Network simulator – J-Sim. http://www.j-sim.org/.

[2] Network simulator – JiST. http://jist.ece.cornell.edu/swans-user/node5.html.

[3] Network simulator – OpNet. http://www.opnet.com/.

[4] Network simulator – QualNet. http://www.scalable-networks.com/.

[5] A. Ferscha and S. K. Tripathi. *Parallel and Distributed Computing Handbook*, chapter Parallel and distributed simulation of discrete event systems. McGraw-Hill, 1995.

[6] Z. Ji, J. Zhou, M. Takai, and R. Bagrodia. Scalable simulation of large-scale wireless networks with bounded inaccuracies. In *Proc. of the 11th international workshop on Modeling analysis and simulation of wireless and mobile systems (MSWiM'04)*, 2004.

[7] H. Kim and J. C. Hou. A fast simulation framework for ieee 802.11-operated wireless lans. In *Proc. of ACM Sigmetrics Conf. on Measurement and Modeling of Computer Systems (Sigmetrics'04)*, May 2004.

[8] M. Takai and R. Bagrodia and A. Lee and M. Gerla. Impact of Channel Models on Simulation of Large Scale Wireless Networks. In *Proc. of the 6th international workshop on Modeling analysis and simulation of wireless and mobile systems (MSWiM'99)*, 1999.

[9] V. Naoumov and T. Gross. Simulation of large ad hoc networks. In *Proc. of the 6th international workshop on Modeling analysis and simulation of wireless and mobile systems (MSWiM'03)*, pages 50–57, 2003.

[10] A. L. Poplawski and D. M. Nicol. Nops: A conservative parallel simulation engine. In *Proc. of the 1998 International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, 1998.

[11] T. S. Rappaport. *Wireless Communications, Principles and Practice*. Prentice Hall, 2002.

[12] K. Sarabandi, I. Koh, G. Liang, and H. Bertoni. Propagation modeling for FCS. In *Proc. of the IEEE Military Communications Conference (IEEE MILCOM'01)*, October 2001.

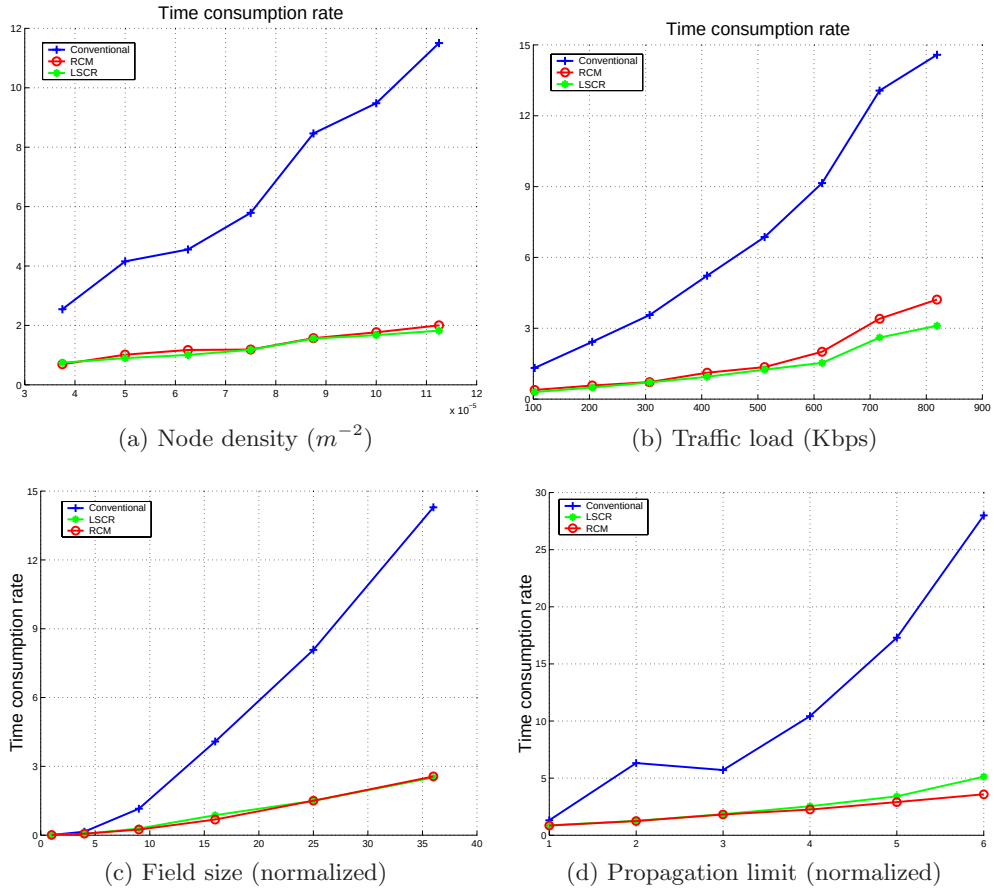
[13] M. Takai, J. Martin, and R. Bagrodia. Effects of wireless physical layer modeling in mobile ad hoc networks. In *MobiHOC 2001, Long Beach, CA, USA*, 2001.

[14] D. Tse and P. Viswanath. *Fundamentals on Wireless Communication*. Cambridge University Press, 2004.

[15] UCB/LBNL/VINT. Network simulator - ns (version 2). http://www-mash.cs.berkeley.edu/ns/, January 2000.

[16] K. Walsh and E. G. Sirer. Staged simulation for improving scale and performance of wireless network simulations. In *Proc. of the 2003 Winter Simulation Conference*, 2003.

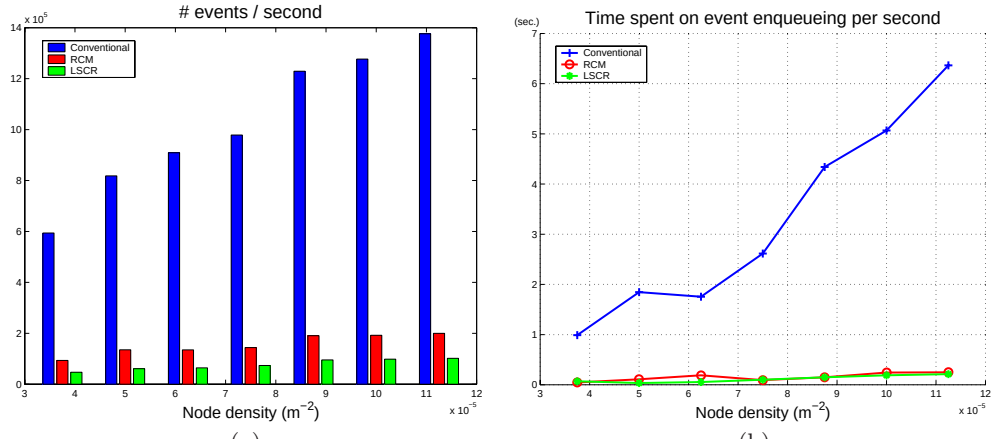
[17] Y. Wu and W. Gong. Time stepped simulation of queuing systems. In *Technical Report, Department of Electrical and Computer Engineering, University of Massachusetts*, 2001.



(a) Node density (m^{-2}) (b) Traffic load (Kbps)

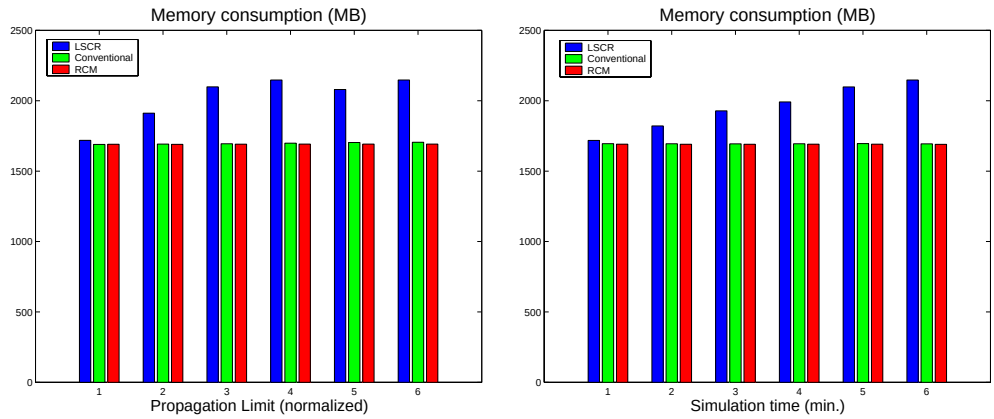
(c) Field size (normalized) (d) Propagation limit (normalized)

Figure 6: Performance comparison of three channel models with respect to node density, traffic load and field size. The field size is normalized to $500 \times 500m^2$ and the propagation limit is normalized to distCST.



(a) (b)

Figure 7: # of events generated per second and the time spent in enqueueing events under the three channel models.



(a) As propagation limit increases (b) As the simulation time increases

Figure 8: Memory usage under the three channel models. Note that the propagation limit is normalized to *distCST*.