On False Blocking in *RTS*/*CTS*-based Multi-hop Wireless Networks

Saikat Ray and David Starobinski

Abstract

The *RTS*/*CTS* mechanism is widely used in wireless networks in order to avoid packet collisions and, thus, achieve high network throughput. In multi-hop settings, however, current implementations of the *RTS*/*CTS* mechanism may lead to inter-dependencies that unnecessarily prohibit nodes from transmitting over long periods of time. We refer to this problem as "false blocking." In this paper, we describe and analyze the false blocking problem in detail. We show that false blocking can lead to significant performance degradation in a variety of topologies and, possibly also, to network-wide congestion. We propose a backward-compatible solution to the false blocking problem, called *RTS Validation*. We model and analyze the performance of *RTS* Validation under general traffic and topology settings and show that it achieves a considerable reduction in the probability of false blocking. Furthermore, we carry out extensive simulations that validate our analysis and show that *RTS* Validation stabilizes throughput at high load and increases its peak value, sometimes by as much as 50%.

I. Introduction

Multi-hop wireless networks, often referred to as *wireless mesh networks*, have gathered significant interest recently due to their superior coverage, reliability, and performance as compared to simple single-hop (star) wireless networks [1–3]. A crucial aspect in the design of these multi-hop wireless networks lies

Saikat Ray is with the Department of Electrical and Systems Engineering, University of Pennsylvania, and David Starobinski is with the Department of Electrical and Computer Engineering, Boston University. This work was done when Saikat Ray was at Boston University.

This work was supported in part by the National Science Foundation under NSF CAREER grant ANI-0132802, NSF grant ANI-0240333, NSF grant CNS-0435312 and by a SPRInG award from Boston University.

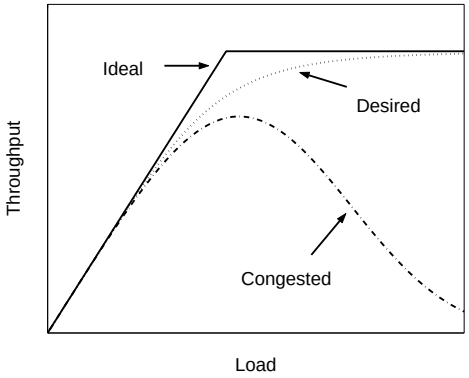


Fig. 1. Typical throughput curves.

in the medium access control (MAC) protocol technology. The Carrier Sense Multiple Access (CSMA) protocol is often chosen in practice because of its inherent simplicity and scalability. However, pure CSMA is susceptible to the well-known hidden node problem [4–6]. Hidden nodes cause costly packet collisions and thus can significantly affect network performance. In order to combat the hidden node problem, a mechanism known as *RTS*/*CTS* handshake is often implemented. For example, the *RTS*/*CTS* mechanism is supported in the IEEE 802.11 family of standards [7], as well as previously proposed MAC protocols, such as MACA [8] and MACAW [9].

From a network point of view, one of the primary reasons for using the *RTS*/*CTS* mechanism is to avoid performance degradation resulting from frequent packet collisions. Figure 1 depicts a set of conceptual "throughput versus load" curves. If the protocol is poorly designed, then the peak throughput remains significantly below the ideal throughput. In some cases, the network may even become congested where the throughput starts decreasing when the offered load exceeds a certain value, as depicted by the "Congested" curve. On the other hand, with a properly designed protocol, the throughput achieves a higher maximum value that is maintained even if the offered load becomes momentarily very high, as depicted by the "Desired" curve.

The *RTS*/*CTS* mechanism generally works well in infrastructure networks, even though it may lead to unfairness in some situations [10]. However, in the general setting of multi-hop wireless networks, current

implementations of the *RTS*/*CTS* mechanism may give rise to situations where a large number of nodes are unnecessarily refrained from transmitting packets for long periods of time. We refer to this problem as “false blocking”. False blocking leads to performance degradation and sometimes even network-wide congestion whereby the network throughput follows the “Congested” curve shown in Figure 1 instead of the “Desired” curve. Therefore, the *RTS*/*CTS* mechanism fails to achieve its goal from a network point of view.

We note that congestion induced by the *RTS*/*CTS* mechanism is different from congestion that arises in the familiar TCP context. The latter occurs due to buffer overflow while the former is related to medium access control, i.e. *RTS*/*CTS*-induced congestion can take place even if an infinite buffer is used in every node.

In this paper, we describe and analyze in detail the false blocking phenomenon in multi-hop wireless LANs. We first explain the cause of false blocking which is due to the fact that any node that receives an *RTS* packet defers its transmission for an entire *DATA* packet transmission period without inquiring whether the *DATA* transmission is actually taking place. We present plausible scenarios where false blocking can affect a large number of nodes in the network and create situations akin to deadlocks, via a cascading effect.

Next, we discuss several approaches for mitigating the false blocking problem including our proposed solution, called *RTS* Validation. A node employing *RTS* Validation uses carrier sensing over a short period of time to decide whether it should defer or not after overhearing an *RTS* packet. This method drastically reduces the average length of a false blocking period, namely the period of time a node is unnecessarily prohibited from transmitting. We note that a method resembling *RTS* Validation is included in the IEEE 802.11 standard, as an optional feature [7, Section 9.2.5.4]. However, as we elaborate in Section IV-C, this optional feature is ineffective in multihop settings, precisely where the false blocking problem is most severe.

To demonstrate the effectiveness of the *RTS* Validation method, we introduce a continuous-time Markov

Chain (CTMC) model to analyze its performance in a general network setting. The analysis reveals that the probability that multiple nodes in the network are simultaneously false blocked decreases at least linearly fast (and in many cases much faster) as the false blocking period is shortened. This result holds for *arbitrary* topology networks and general Markovian traffic patterns.

We perform extensive Matlab simulations of our Markov chain model as well as NS [11] simulations of IEEE 802.11 networks, under various topology and parameter settings. These simulations show that our Markov chain model predicts well the general behavior of IEEE 802.11 networks. More importantly, they confirm the extent of the false blocking problem in these networks. They also show the effectiveness of our *RTS* Validation method in addressing this problem as well as it superiority over alternative mitigating approaches. In particular, our simulations show increase in network throughput of as much as 50% in some cases as well as significant delay reduction, as a result of implementing the *RTS* Validation scheme.

The rest of the paper is organized as follows. We discuss related work Section II. In Section III, the false blocking problem is described in detail. Next, Section IV describes several techniques to mitigate the impact of the false blocking problem. This section includes the description of our proposed solution, the *RTS* Validation mechanism. Section V is devoted to a mathematical analysis of *RTS* Validation. Detailed simulation results studying the effect of *RTS* Validation on network performance are presented in Section VI. Finally, in Section VII, we conclude the paper.

II. Related Work

The Carrier Sense Multiple Access (CSMA) protocol was proposed in [12]. The susceptibility of CSMA to the hidden node problem was noted in [4], where the authors proposed a solution called Busy-tone Multiple Access (BTMA) protocol. To mitigate the hidden node problem with half-duplex radios, the Multiple Access with Collision Avoidance (MACA) protocol was proposed in [8]. MACA uses the *RTS*/*CTS* mechanism to avoid the hidden node problem, but does not include any positive acknowledgment to ensuring the integrity of the *DATA* transmission. A positive acknowledgment scheme was added in the MACAW protocol [9]. The MACAW protocol also requires nodes to send a packet called *DATA*-Send

(DS) to indicate that a *DATA* packet transmission is about to begin, however this mechanism is not part of the IEEE 802.11 standard.

We note that in a general multi-hop network, the *RTS*/*CTS* mechanism cannot completely eliminate *DATA* packet collisions, due to the masked node problem [13]. This is true even under idealized conditions such as negligible control packet size, negligible propagation delay, and identical interference and packet sensing ranges. Nevertheless, the *RTS*/*CTS* mechanism greatly mitigates the hidden node problem and therefore its deployment is generally desirable (this observation is confirmed by our simulations in Section VI-B.2).

Blocking in wireless network (cf. Section III-B) is discussed in [14], but that work does not mention the more severe false blocking problem (cf. Section III-C) which was first described in our preliminary work [15]. The present work significantly adds to [15] by providing (i) a rigorous analysis of the *RTS* Validation mechanism; (ii) detailed description and comparison with other techniques for mitigating the false blocking problem; and (iii) extensive Matlab and NS simulations comparing the performance of various IEEE 802.11 networks with and without use of the *RTS* Validation method.

Several recent publications have built upon [15]. For instance, references [16] and [17] propose new MAC protocols in order to overcome the false blocking problem in sensor networks and wireless LANs, respectively, and the work in [18] describes how false blocking can be used to mount a denial of service attack.

III. The False Blocking Problem

A. Background

We first briefly review salient features of the IEEE 802.11 Wireless LAN protocol that are most relevant to the rest of this paper. The protocol is described in detail in [7].

To mitigate the hidden node problem, the IEEE 802.11 MAC protocol supports the *RTS*/*CTS* access control method, which is illustrated in Fig. 2. When node B wants to send a packet to node C, it initially

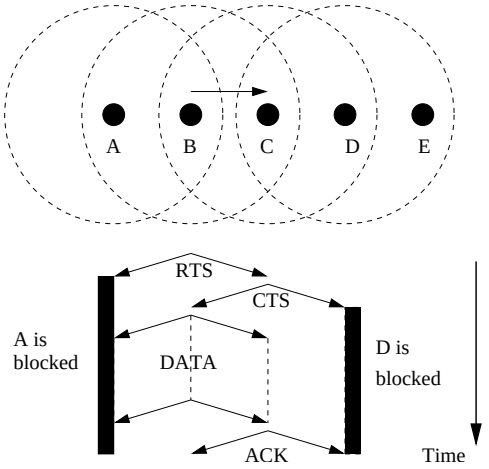


Fig. 2. The *RTS*/*CTS* mechanism. The circles depict the range of the nodes. The lower half depicts the time-line. The dark bars below node C and D indicate their blocking duration.

sends a small packet called *Request-to-Send* (RTS). Upon correctly receiving the *RTS*, node C responds with another small packet called *Clear-to-Send* (*CTS*). After receiving the *CTS*, node B sends its *DATA* packet to node C. If node C receives the *DATA* packet correctly, it sends an *ACK* back to node B. Other than the source and destination, any node that hears an *RTS* or a *CTS* is prohibited from transmitting any signal for a period that is encoded in the *duration* field of the *RTS* and *CTS*. The duration fields in *RTS* and *CTS* packets are set such that nodes B and C will be able to complete their communication within the prohibited period. Each node maintains the deferral periods by an indicator of time called the Network Allocation Vector (NAV). The NAV may be thought of as a counter, which counts down to zero at a uniform rate with nonzero value of NAV indicating prohibited period [7, Section 9.2.1]. Assuming control packets are correctly received by all the nodes, the *RTS*/*CTS* mechanism solves the hidden node problem in the topology of Fig. 2, since node D is notified by a *CTS* when node B initiates a transmission.

If no *CTS* arrives in response to a *RTS* or no *ACK* arrives in response to a *DATA* packet, a sender enters an exponential backoff and retransmits after a backoff interval. The backoff interval after the j-th attempt ($j = 0, 1, \ldots$) is set to $k \times 20\mu s$, where the integer k is chosen uniformly at random from the values $(0, 1, \ldots, \min\{2^{j+5}, \text{CW\_MAX}\})$. The constant CW_MAX is set to 1024. The parameters SHORT RETRY LIMIT and LONG RETRY LIMIT limit the maximum number of successive *RTS* and *DATA* retransmissions, respectively (assuming that the length of *DATA* packets exceeds the value of the *RTS*

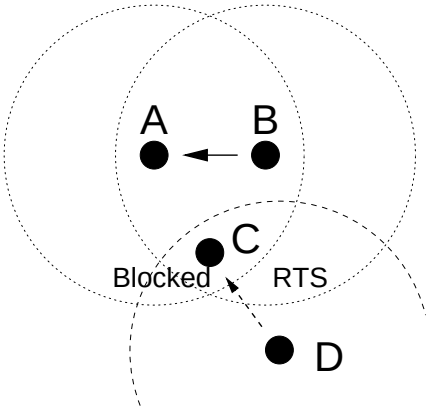


Fig. 3. Blocking Problem. Node C is blocked due to the communication between node A and node B. Therefore, node D does not get any response to the *RTS* packets it sends and enters backoff.

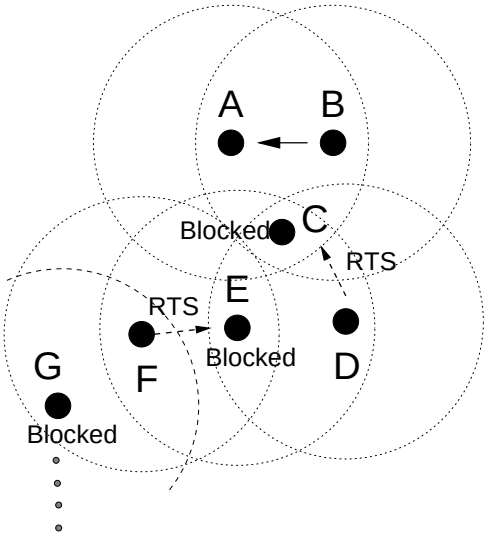


Fig. 4. False blocking problem. Node E is unnecessarily blocked because of node D's *RTS*. Therefore, node F does not get any response to its *RTS*, which in turn blocks node G and so on.

Threshold parameter [7]).

B. Blocking

We define a node to be *blocked* if it is prohibited from transmitting at a given instant. In particular, any node that hears an *RTS* or a *CTS* packet, but is not the destination of that packet, becomes blocked. A blocked node cannot reply to *RTS* packets sent to it. The *RTS* sender, however, interprets the lack of response as a channel contention and enters backoff. We refer to this problem as the *blocking problem*.

Fig. 3 describes the blocking problem. In this figure, node B transmits a packet to node A. Node C receives both *RTS* and *CTS* packets and therefore remains prohibited from transmitting. When the communication between node B and node A is going on, node D sends an *RTS* to node C. Since node C is blocked, it cannot respond with a *CTS*. Therefore, node D does not get any response and enters backoff.

In [14], the author briefly describes this problem where it is termed as *hidden receiver* and *exposed receiver* problem. However, the blocked node need not be a hidden or an exposed node. For example, in Fig. 4, node C receives both *RTS* and *CTS* and is therefore neither a hidden nor an exposed node. Therefore, we prefer to call this problem the blocking problem.

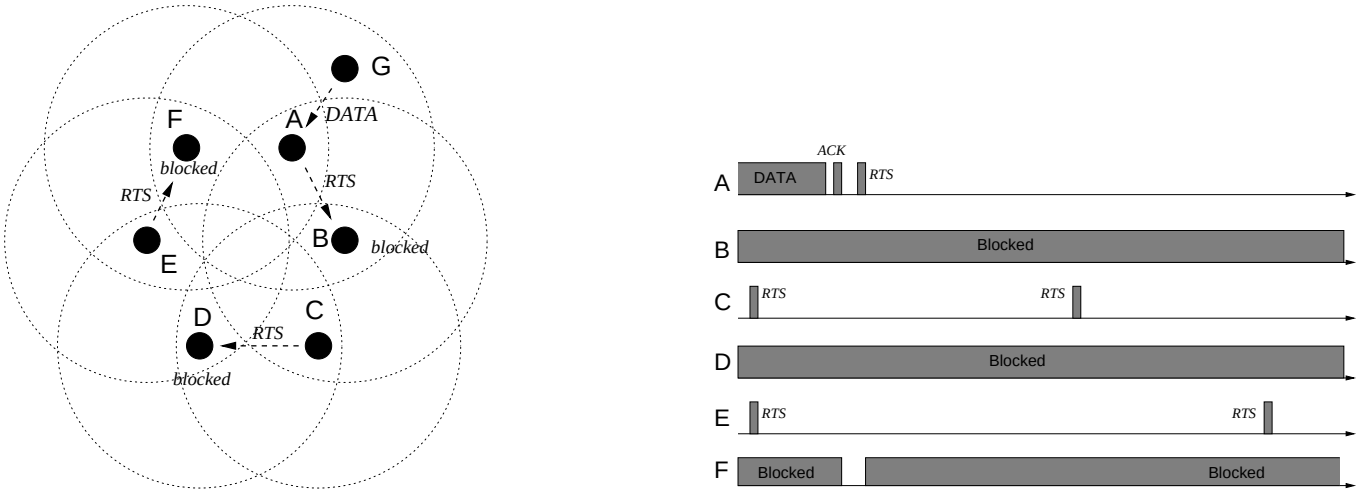
C. False Blocking

In IEEE 802.11, any node that receives an *RTS* packet becomes blocked in order to protect the *ACK* packet that the *RTS*-sender is supposed to receive. However, due to this rule, a nearby node may get false blocked, i.e. it may become prohibited from transmitting even if no other node is actually transmitting. Specifically, an *RTS* packet destined to a blocked node forces every other node that receives the *RTS* to inhibit itself even though the blocked destination does not respond and thus no *DATA* packet transmission takes place. We call this problem the *false blocking problem*. Fig. 4 illustrates the problem. In this figure, node B is sending a packet to node A and therefore node C is blocked. While node C is blocked, node D sends *RTS* packets to node C, but node C does not respond. However, node E receives the *RTS* packets and prohibits itself from transmitting. Node E is therefore false blocked.

The simple blocking problem is localized to the neighbors of the blocked node and thus has a limited impact on the network performance. False blocking, however, may *propagate* through the network, i.e. one node may become false blocked due to a node that itself is false blocked. Therefore, false blocking may affect network performance severely.

Fig. 4 shows a scenario of the propagation of false blocking. As in the previous example, node E is false blocked. Now, node F sends an *RTS* packet to node E and does not get any response. Therefore, node F goes into backoff. However, the *RTS* packet sent by node F blocks node G and so on. Note that a blocked node remains blocked for slightly longer than a *DATA* packet transmission time (see Fig. 2). Therefore, the likelihood that a node tries to communicate with a blocked node is non-negligible, especially at high network load.

In the rest of this paper, we will use the terms *false RTS* to specify that the *RTS* was sent to a blocked node and *false blocking period* to denote the duration of time a node blocks itself after overhearing a false *RTS*.



(a) Topology. (b) Timing.

Fig. 5. Pseudo-Deadlock. Initially node A receives a packet from node G. This transmission creates the sequence of blocked and deferring nodes $\{F, E, D, C\}$. After G's transmission is over, node A transmits an *RTS* packet to node B. However, node C's *RTS* sent to node D blocks node B while node A receives packet from node G, therefore, node A does not get response to its *RTS*. Now, every node tries to transmit to a blocked node and a deadlock occurs.

D. Pseudo-Deadlock

The false blocking problem may not only propagate through a network, but also give rise to deadlock situations, at least temporarily. Once such a deadlock takes place, the throughput of the nodes involved in the deadlock goes down to zero. This deadlock, however, may eventually be broken if packets are dropped after a certain number of attempts. Therefore, we refer to this situation as a *pseudo-deadlock*.

A pseudo-deadlock occurs when the propagation of false blocking takes place along a "circular" path. Fig. 5(a) depicts such a situation. In this figure, node A initially receives a packet from node G. Node F is blocked during this time because node A is receiving. So, node E does not get any response to the *RTS* packets it sends to node F. Node E's *RTS* packets, however, force node D into false blocking. Subsequently, node C does not get any response to the *RTS* packets sent to node D and, therefore, goes into a deferral period. Node B, however, receives node C's *RTS* packets and therefore gets blocked.

Now, node A, after receiving the packet from node G, wishes to communicate to node B. But, when node A sends an *RTS* to node B, node B is already blocked and so node A does not get any response. Node A's *RTS* blocks node F, though. So, when node E sends its next *RTS*, it again receives no reply. At this moment, in the cycle $\{A, B, C, D, E, F, A\}$, every second node $\{A, C, E\}$ is sending *RTS* to the next

node $\{B, D, F\}$, which is already blocked and due to this *RTS*, the previous node gets blocked. As long as the blocking period for a node that receives an *RTS* is greater than the maximum time gap between two *RTS* packets during retries, the nodes cannot come out of this situation and, therefore, this is a deadlock. Fig. 5(b) shows the timings of each packet.

In the usual situation, one of the nodes will drop its packet after a certain number of retries and the deadlock will be broken. However, if each node needs to send several packets to the same destination (that may originate from a higher layer protocol), then the deadlock may persist for a long period of time.

IV. Mitigation Techniques

In this section, we present several approaches for mitigating the false blocking problem. The first approach, described in section IV-A, makes use of explicit notifications. Unfortunately, this approach is not backward-compatible with the current IEEE 802.11 standard. The next approach, described in Section IV-B, relies on increasing the length of backoff intervals. This approach helps somewhat in mitigating the problem, but is not fully effective. Our proposed solution is presented in section IV-C, where an *RTS* is validated using carrier sensing, making it both effective and backward compatible.

A. Auxiliary Control Packets

False blocking is a consequence of the fact that every node that receives an *RTS* inhibits itself from transmitting, even if the *RTS* was false. Therefore, the most obvious way of mitigating the false blocking problem is to explicitly notify the neighbors whether the *RTS* was false or not. This way, the neighbors can safely ignore a false *RTS*. This can be done by sending additional packets. For example, in the MACAW [9] protocol, a packet called *DATA-Send* (DS) is sent when a node receives *CTS* in response to an *RTS*. Thus, the neighboring nodes block themselves only when they hear the DS packet. In a complementary approach, a *Negative CTS* (NCTS) packet is sent by the *RTS* sender when it does not receive a *CTS* reply [14]. The main drawback of such approaches is that they break compatibility with legacy protocols and require additional system resources for transmitting auxiliary packets.

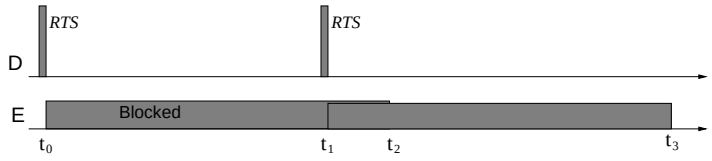


Fig. 6. Continuous blocking due to closely spaced *RTS* packets.

B. Increasing Backoff Intervals

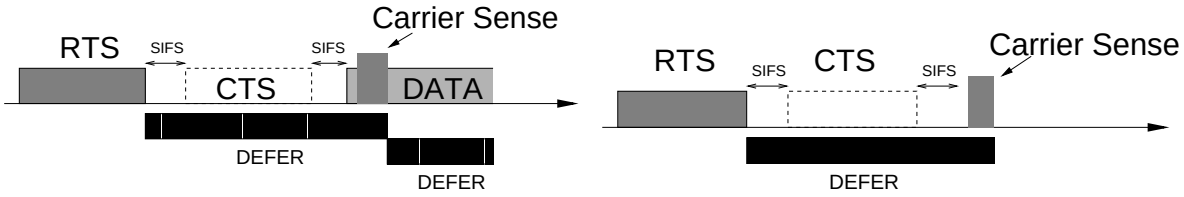
As observed earlier, the false blocking problem is exacerbated when the gap between successive RTS retransmissions is too short. For example, consider nodes D and E in the scenario of Fig. 4 and suppose node E receives an *RTS* at time $t = t_0$ and blocks itself until time $t = t_2$, as shown in Fig. 6. If the next *RTS* sent by node D reaches node E at time $t = t_1 < t_2$, then node E blocks itself until $t = t_3$. Thus, in this example node E remains continuously blocked for a period that is considerably larger than a *DATA* packet transmission time.

This problem can be mitigated by making sure that gaps between *RTS* retries are longer than a *DATA* packet transmission. In IEEE 802.11 protocol, this can be achieved by setting the SHORT RETRY LIMIT parameter to a large value, which increases the maximum allowable number of *RTS* retries and backoff times between them. However, the false blocking period is still at least as long as the transmission time of a *DATA* packet. Thus the extent of recovery achievable with this approach is limited, as pointed out by our simulation results presented in the sequel.

C. RTS Validation

When a node receives a false *RTS*, then the corresponding *DATA* packet transmission does not take place while the node defers. Therefore, if a node assesses the channel to be idle during the expected *DATA* packet transmission period following an *RTS*, then the node must be false blocked.

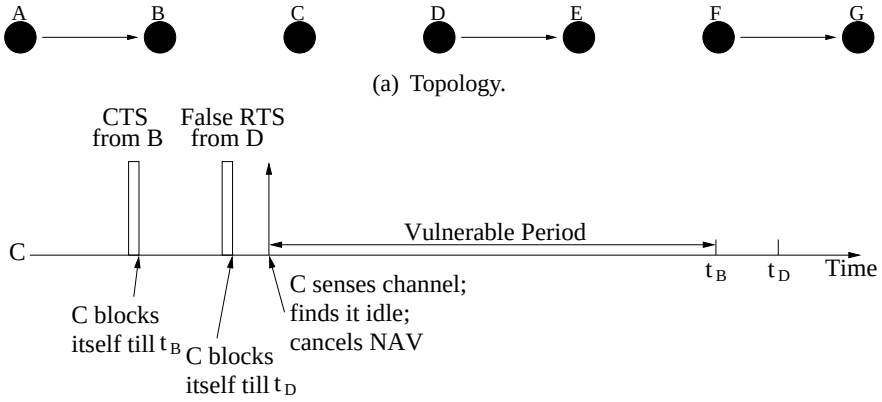
Our proposed solution of the false blocking problem, called *RTS Validation*, is based on this observation. A node that uses *RTS* Validation, upon overhearing an *RTS* packet, defers until the corresponding *DATA* packet transmission is expected to begin and then assesses the state of the channel. If the channel is found idle, then it defers no longer, otherwise it continues deferral. Specifically, when a node receives



(a) The node senses busy channel following *RTS*_Defer_Time and therefore continues deferral.

(b) The node assesses idle channel following *RTS*_Defer_Time and therefore defers no longer.

Fig. 7. *RTS* Validation Mechanism.



(a) Topology.

(b) Timing diagram of events at node C.

Fig. 8. Example scenario where the optional feature of the IEEE 802.11 standard may lead to a packet collision.

an *RTS* that is not destined for itself, it defers for next *RTS_Defer_Time*. The *RTS_Defer_Time* is set as small as possible so that the *DATA* packet transmission is expected to begin at the end of this period, with allowances for various propagation delays. After this deferral period, the node assesses the channel for next *Clear Channel Assessment Time* (CCA_TIME) while continuing deferring (The CCA_TIME is the time required to assess the state of the channel [7]). If the channel is assessed to be busy, the node defers for an additional period so that the total deferral time equals to *Requested_Defer_Time*, the duration of deferral requested by the *RTS*; otherwise it defers no longer. Fig. 7 explains the proposed rule.

With *RTS* Validation, the nodes that receive an *RTS* destined to a blocked node ignore the *RTS* when the channel is assessed idle. Since *RTS_Defer_Time* and CCA_Time are generally much smaller than the *Requested_Defer_Time*, the likelihood of propagation of false blocking greatly reduces when *RTS* Validation is used. Note, however, that the *RTS* Validation mechanism may decide to defer even if the corresponding *DATA* packet transmission did not begin if it assesses the channel to be busy because of other transmissions.

The IEEE 802.11 standard includes an *optional* feature that resembles *RTS* Validation, but does not effectively address the false blocking problem. This feature permits a node to *reset* its NAV if an *RTS* was *the most recent basis* for updating its NAV and no packet reception starts at (2 $\times$ SIFS_TIME) + (CTS_TIME) + (2 $\times$ SLOT_TIME) after the reception of the *RTS* (i.e., when the corresponding *DATA* packet was supposed to start) [7, Section 9.2.5.4]. This mechanism works well in single-hop topologies. For instance, it prevents nodes from being unnecessarily blocked in cases where an *RTS* or a *CTS* is dropped by the intended recipient, e.g., due to bit errors. In multihop settings, however, this optional feature may erroneously allow nodes to transmit and cause packet collisions. For a simple example, consider the linear topology of seven nodes shown in Fig. 8(a). Suppose node A starts transmitting an *RTS* to node B at time $t_0 = 0$. Node B successfully receives the *RTS* and replies with a *CTS*. At time $t_1 =$ RTS_TIME + SIFS_TIME + CTS_TIME, node C hears the *CTS* sent by node B and advances its NAV to time t_B (cf. Fig. 8(b)). Now suppose that at time t_2, node F starts sending a *DATA* packet to node G (following a successful *RTS*/*CTS* handshake). At this point, node E is blocked. Now, suppose node D starts sending an *RTS* to node E at time t_3, for which it receives no reply from node E. This *RTS* however, reaches node C at time $t_4 = t_3 +$ RTS_TIME and node C updates its NAV to $t_D > t_B$. Then, according to the optional feature, node C checks the medium at time $t_4 + 2 \times$ SIFS_TIME + CTS_TIME + (2 $\times$ SLOT_TIME) and finds the medium idle. Since at this time, the *RTS* sent by node D was the most recent basis for updating the NAV, it resets the NAV and becomes free to transmit. However, node A may still be transmitting at this time, and if node C proceeds to transmit any packet, it will cause a collision at node B.

The reason why the optional feature fails in multihop topologies is that the NAV does not contain enough state information to avoid false blocking without increasing the chance of packet collisions. *RTS* Validation effectively addresses this deficiency by keeping appropriate state information for each received *RTS*, and thus is able to avoid false blocking without increasing packet collisions.

RTS Validation is a backward-compatible approach in the sense that a node that uses *RTS* Validation (an "intelligent" node) may communicate with a node that does not use *RTS* Validation (standard node),

since *RTS* Validation does not require any change in the packet format or the packet exchange protocol. However, in a network mixed with intelligent and standard nodes, the intelligent nodes may be able to transmit more packets since they defer for a much smaller time in the case of false blocking. Note that this is actually an excellent incentive for nodes to implement *RTS* Validation.

V. Evaluation of the *RTS* Validation Mechanism

A. Motivation

We presented the *RTS* Validation mechanism in Section IV-C for mitigating the false blocking problem described in Section III. When *RTS* Validation is in use, a false *RTS* blocks the neighboring nodes for a time that is typically much shorter than the requested deferral time. Although it is intuitive that *RTS* Validation helps mitigating the false blocking problem, the extent of its effectiveness with regard to reducing the probability of false blocking is not evident. Thus, our goal this section is to develop an analytical model of the *RTS* Validation mechanism, applicable to networks of general topology, and derive upper bounds on the probability that one or more nodes in the network are simultaneously false blocked.

In order to achieve the above goal, we model the network dynamics (e.g. exogenous arrivals, *RTS* attempts, etc.) so that it evolves as a continuous-time Markov chain. We assume that the exogenous traffic load on the network is low enough so that the Markov chain is ergodic (i.e., there exists a stationary distribution for all of the states). In the sequel, we derive bounds on the stationary probabilities of states where nodes are false blocked and investigate the behavior of these bounds as the false blocking period is reduced. Specifically, put $1/\gamma$ to be the average false blocking period. Then, we show that the probability that any node is false blocked decreases at least as fast as $1/\gamma$, as $\gamma \to \infty$. Furthermore, consider a state with $n \geq 1$ false blocked nodes in the network, each blocked by a different false *RTS*. Then, we show that the probability that the system is in such a state decreases at least as fast as $1/\gamma^n$, as $\gamma \to \infty$. This result shows that the *RTS* Validation mechanism drastically reduces the probability of having multiple false blocked nodes, and thereby pseudo-deadlocks, in the network.

B. Network Model and Statistical Assumptions

We consider a network of *arbitrary topology* with N nodes. Each node maintains an infinite buffer. The exogenous traffic arriving to node i, where $1 \leq i \leq N$, follows an independent Poisson process with rate λ_i. We assume that λ_i's are small enough so that the network is stable (that is, queues do not grow indefinitely). The *RTS*/*CTS* access method is used for each packet. The transmission times of the control packets and propagation delay for all packets are assumed to be negligible. The transmission time for each *DATA* packet is an independent exponential random variable with rate μ. The channel is assumed to be noise-free so that a packet transmission always succeeds unless it collides with another packet (because of a masked node). Each new packet sent by node i is destined to one of its neighboring node j with probability p_{ij}; packet retransmissions are destined to the same node as the original packet. Note that the results derived in the sequel can easily be generalized to the case where the service times and inter-arrival times follow phase-type distributions (which in turn can be used to approximate arbitrarily closely any general distributions with $[0, \infty)$ support [19]).

The *RTS* Validation is modeled in the following way: an *RTS* destined for a free node (defined to be a genuine *RTS*) blocks the neighboring nodes for the duration of the corresponding *DATA* transmission time, but an *RTS* destined for a blocked node (defined to be a false *RTS*) blocks the neighbors for a random time that is exponentially distributed with rate γ. The larger the γ, the shorter the false blocking period. Note that in the standard case (when *RTS* Validation is not used) we have $\gamma = \mu$, while if *RTS* Validation is used, we have $\gamma \gg \mu$.

Finally, we model the backoff mechanism in the following way. Let the transmission queue at node i be nonempty, its backoff stage be denoted by w_i and $t = \tau_0$ be the time when one of these following events occurs: (i) the channel becomes idle, (ii) an *RTS* attempt is unsuccessful; i.e., no *CTS* is sent for that *RTS*, (iii) a *DATA* transmission ends. Then node i schedules an *RTS* transmission after a random time τ that is independent and exponentially distributed with rate σ_{w_i}. At time $t = \tau_0 + \tau$, if node i senses the channel idle, it proceeds with the *RTS* attempt; otherwise it repeats the process. The backoff stage w_i is

reset to 0 after a successful *DATA* transmission and incremented by 1 after an unsuccessful *RTS* or *DATA* transmission. The constant σ_{w_i} depends on the backoff stage w_i and can be set to match average backoff times of the IEEE 802.11 protocol. The maximum backoff rate is denoted by σ^*, i.e. $\sigma^* = \max_{w_i} \sigma_{w_i}$. It is worth mentioning that backoff times being exponentially distributed can be arbitrarily long (although with very small probability). Thus, pseudo-deadlocks as described in Section III will always be broken eventually.

C. The Markovian Structure

In order to show that the network dynamics can be modeled using a Markov chain, we first characterize the state-space $\mathscr{S}$ and define the variables that collectively determine each state $s \in \mathscr{S}$. In state s, for each node i, $q_i(s)$ is the queue-length and $D_i(s)$ the ordered array of the destination nodes for the packets in the queue; $w_i(s)$ is the backoff stage; $t_i(s)$ is the transmission indicator, i.e. $t_i(s) = 1$ if node i transmits in state s, $t_i(s) = 0$ otherwise; u_i is the success indicator, i.e. if $u_i(s) = 1$, then the current transmission did not undergo any collision, and if $u_i(s) = 0$, the transmission collided.

Now, define an *RTS* (a *CTS*) to be active at node i in state s if node i is blocked in state s due to this *RTS* (*CTS*, respectively). Then, $rts_{ij}(s)$ is the genuine *RTS* indicator, i.e. $rts_{ij}(s) = 1$ if a genuine *RTS* sent by node j is active at node i and $rts_{ij}(s) = 0$ otherwise; $cts_{ij}(s)$ is the *CTS* indicator, i.e. $cts_{ij}(s) = 1$ if a *CTS* sent by node j is active at node i, and $cts_{ij}(s) = 0$ otherwise.

The false *RTS* packets are counted network-wide. We define a false *RTS* to be active in state s if one or more nodes are false blocked because of this *RTS*. We let $f(s)$ denote the total number of active false *RTS* in the network in state s, and B_j $(j = 1, \ldots, f(s))$ denote the set of nodes that are blocked by the j-th false *RTS* (the ordering of false *RTS* is arbitrary). For convenience, denote the collection of the sets B_j by $\mathbb{B} = \{B_j : j = 1, \ldots, f(s)\}$. By convention, $\mathbb{B}$ is empty for $f(s) = 0$.

Using the above variables, we can then define a state as follows $s = (q_i, D_i, w_i, t_i, u_i, rts_{ij}, cts_{ij}, f, \mathbb{B} : i, j = 0, 1, \ldots, N-1)$. Now, let $\mathbf{X}(t)$ denote the state of the network at time t. Then, one can prove that the stochastic process $\{\mathbf{X}(t), t \geq 0\}$ evolves as a continuous-time Markov chain over the state-space $\mathscr{S}$

with stationary probabilities $\pi(s)$, where $s \in \mathscr{S}$. This is because the time spent by $\{\mathbf{X}(t)\}$ in each state s is exponentially distributed (as the minimum between various independent and exponentially distributed random variables) and at a transition instant, the transition probability from state s to any state s' is a fixed probability $p_{ss'}$, where $\sum_{s' \neq s} p_{ss'} = 1$. The probability $p_{ss'}$ is given by the ratio of the transition rate from state s to state s' to the total rate out of state s. A detailed description of the state transitions and their rate is deferred to the Appendix. The most important point for the following is that, after each transition, the active false *RTS* count f can increase by at most 1, if there is an unsuccessful *RTS* attempt, or decrease by at most 1, if the blocking period associated with a false *RTS* expires.

D. Effectiveness of the RTS Validation Mechanism

We now quantify the reduction in the false blocking probability resulting from the use of the *RTS* Validation mechanism. Assume that there exists at least one node in the network that can be false blocked; otherwise the false blocking probability is trivially zero. Our main result then is as follows:

Theorem V.1 *The probability that there are at least $m \geq 1$ concurrently active false RTS in the network is $O(1/\gamma^m)$, as $\gamma \to \infty$.*

To make the connection with the probability of having false blocked nodes in the network, we define the notion of *separated* nodes. Two nodes in the network are separated if no other node can block both of them by sending one false *RTS*. Similarly, a set of nodes are separated if they are mutually separated. By convention, a single node is considered separated. It is a sufficient condition for two nodes to be separated that they do not share any common neighbor, although it is not a necessary condition (see, for instance, the example of Fig. 5(a) where nodes B, D, and F are all separated). Having n separated false blocked nodes implies that the number of concurrently active false *RTS*, m, must satisfy $m \geq n$ since no two separated nodes can be blocked by the same false *RTS*. We therefore have the following corollary.

Corollary V.2 *The probability that there are $n \geq 1$ separated false blocked nodes in the network is $O(1/\gamma^n)$ as $\gamma \to \infty$.*

In general, a set of separated false blocked nodes can be chosen in many ways, e.g., we can choose a maximal set to find the greatest decay exponent n. Note that the presence of false blocked nodes in a network always implies $m \geq 1$ since, in the worst case, a single false *RTS* blocks all the nodes. Thus, another corollary of Theorem V.1 is the following:

Corollary V.3 *The probability that there are one or more false blocked nodes in the network is* $O(1/\gamma)$*, as* $\gamma \to \infty$.

From the above results, we see that as γ is increased, the probability that there are false blocked nodes in the network goes to zero. Moreover, the higher the number of false blocked nodes, the faster the decay if the nodes are separated. Thus increasing γ as much as possible is a very effective way of mitigating the false blocking problem. The *RTS* Validation algorithm proposed in Section IV-C precisely achieves this goal.

The key for proving Theorem V.1 is Proposition 1 stated below. In preparation for this result, define class C_k to be the class of states where k false *RTS* are concurrently active. More precisely, $C_k = \{s \in \mathscr{S} : f(s) = k\}$. Then, the following holds:

Proposition 1 *The stationary distribution* π *for the Markov chain* $\mathbf{X}(t)$ *satisfies*

$$\sum_{s \in C_k} \pi(s) \leq A(\gamma) \frac{(N\sigma^*)^k}{k!\gamma^k}. \tag{1}$$

Here, $1 \geq A(\gamma) \geq 0$ *denotes the probability that there are no false blocked nodes in the network.*

Proof: We use induction on k. For $k = 0$, the L.H.S of (1) is equal to sum of the probabilities of the states where $f = 0$. Since $f = 0$ implies that there are no false blocked nodes and vice versa, $\sum_{s \in C_0} \pi(s) = A(\gamma)$, which establishes the base case.

We now show that if the proposition is true for $k = K$, then it is also true for $k = K + 1$. The proposition then follows for all k by induction. Towards this end, recall that in the Markov chain $\mathbf{X}(t)$, the active false *RTS* count can change only by at most 1. Therefore, for any transition from state s to

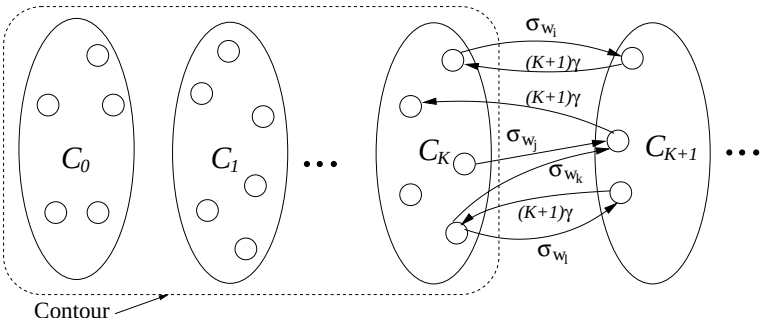


Fig. 9. Conceptual Diagram Showing the State Space structure of the Markov Chain $\mathbf{X}(t)$.

state s', we have $|f(s) - f(s')| \leq 1$. Thus, if $s \in C_i, s' \in C_j$ and $|i - j| \geq 2$, then there is no transition from state s to state s'. So the state-space can be viewed as a graph shown in Fig. 9, where there are transitions only between the states that belong to same or adjacent classes.

Now we observe that from any state in C_{K+1}, a transition to a state in C_K occurs due to the expiration of one of the $K+1$ false *RTS*. Thus, such transition occurs with rate $(K+1)\gamma$. On the other hand, any transition from a state $s' \in C_K$ to a state $s \in C_{K+1}$ increases the active false *RTS* count and occurs with rate σ_{w_j} for some node j that is at backoff stage w_j. We let $\tilde{C}_K (\subseteq C_K)$ denote the set of states from which a transition occurs to a state in C_{K+1}. The set $\tilde{C}_K$ is nonempty because of our assumption that at least one node in the network can be false blocked.

Now consider a contour that encloses the states belonging to the classes $C_0, C_1, \ldots, C_K$, as shown in Fig. 9. We remind that the stationary distribution π satisfies the balance equation across any closed contour. In this case, the only transitions across this contour are from a state in C_K to a state in C_{K+1}, and vice versa. Thus, the balance equation across the contour takes the form

$$\sum_{s' \in \tilde{C}_K} \left(\sum_w \sigma_w n_w(s') \right) \pi(s') = \sum_{s \in C_{K+1}} (K+1)\gamma \pi(s) \tag{2}$$

The constant $n_w(s')$ denotes the number of nodes in state s' that are in backoff stage w and transmits false *RTS* packets. Note that, for any state s', we have

$$\sum_w \sigma_w n_w(s') \leq N\sigma^*. \tag{3}$$

Now, since $\sum_{s\in C_{K+1}} \pi(s) \leq 1$, the R.H.S, and hence the L.H.S of Eq. 2 is bounded by $(K+1)\gamma$. Thus, the sums involved in Eq. 2 are absolutely convergent (since all the terms are positive), and therefore we can perform term-by-term operations. Then,

$$(K+1)\gamma \sum_{s\in C_{K+1}} \pi(s) = \sum_{s\in C_{K+1}} (K+1)\gamma\pi(s) = \sum_{s'\in \tilde{C}_K} \left(\sum_w \sigma_w n_w(s')\right) \pi(s') \quad (4)$$

$$\leq N\sigma^* \sum_{s'\in C_K} \pi(s') \leq N\sigma^* A(\gamma) \frac{(N\sigma^*)^K}{K!\gamma^K} \quad (5)$$

Here, the second equality in Eq. 4 is due to the balance equation (Eq. 2), the first inequality in Eq. 5 is due to Eq. 3 and the fact that $\tilde{C}_K \subseteq C_K$, and the second inequality in Eq. 5 is due to the induction hypothesis. From (5), we get

$$\sum_{s\in C_{K+1}} \pi(s) \leq A(\gamma) \frac{(N\sigma^*)^{K+1}}{(K+1)!\gamma^{K+1}}. \quad (6)$$

Therefore, the induction hypothesis is true for $k = K+1$, which proves the proposition. ■

We complete this section by proving Theorem V.1.

Proof: [Theorem V.1] From Proposition 1, the probability of having at least m active false *RTS* is

$$\sum_{k\geq m} \sum_{s\in C_k} \pi(s) \leq A(\gamma) \sum_{k\geq m} \frac{(N\sigma^*)^k}{k!\gamma^k}. \quad (7)$$

Note that the sum is always convergent. A simple upper bound for the R.H.S of Eq. 7 is given by $\frac{(N\sigma^*)^m}{m!\gamma^m} e^{N\sigma^*/\gamma} = O(1/\gamma^m)$. ■

VI. Simulation

Our analysis presented in Section V showed that the *RTS* Validation mechanism effectively reduces the probability of having false blocked nodes in the network. As a result, we expect significant increase in network throughput and reduction in delay. In this section, we perform detailed simulations to quantify these performance gains. We first present Matlab simulations, based on the Markov chain model of Section V-B, and evaluate the impact of parameters λ and γ on the network throughput for a ring topology. We then carry out simulations using *NS* [11] for IEEE 802.11 wireless networks, for a ring and a general large random topology. We evaluate the performance of these networks with and without use

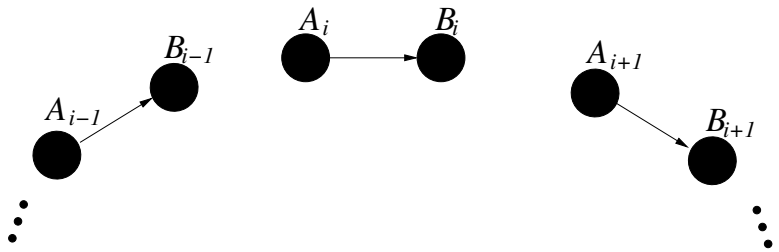


Fig. 10. Ring topology.

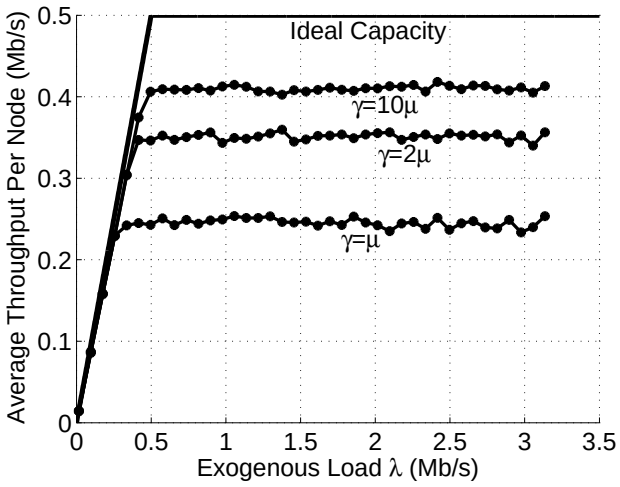


Fig. 11. Simulation of the Markov Chain model: throughput versus exogenous traffic load λ.

of *RTS* Validation. We also evaluate the effectiveness of increasing the time between *RTS* attempts (by modifying the value of the SHORT RETRY LIMIT parameter) versus using *RTS* Validation.

A. Simulation of the Markov Chain Model

In this section, we present simulations results for the general Markov chain model model described in Section V. We assume that packet lengths are exponentially distributed with parameter $\mu = 62.5$ so that the mean packet transmission time is 16 ms. This value corresponds to the transmission time of a 2000 bytes packet at 1 Mb/s rate. Backoff intervals are exponentially distributed with rate σ_i, for any backoff stage $i \leq 6$, and $\sigma_i = \sigma_6$, for any backoff stage $i > 6$. We set $\sigma_1 = 3125\ s^{-1}$ so that the mean backoff time in the first backoff stage is $16 \times 20\mu s = 320\mu s$ and $\sigma_{i+1} = \sigma_i/2$ for the subsequent stages, where $1 \leq i < 6$.

The simulation uses the ring topology shown in Fig. 10 that consists of 10 pairs of nodes. Each node

A_i transmits packets to node B_i using the *RTS*/*CTS* access method. This topology allows us to isolate the effect of the *RTS* Validation mechanism from other factors that can affect the throughput. In particular, simulation results are not affected by unfairness issues that can arise in asymmetric topologies. In addition, there is no masked node in this topology, and therefore *DATA* packets do not collide [13]. Also, it is easy to calculate the ideal network throughput for a ring topology, which can serve as a reference for our results.

The simulations are carried out for various values of γ. For each given value of γ, we let λ increase, starting from a small value until the network saturates. The simulation outcomes are shown in Fig. 11. We define the throughput per node to be the average number of successfully transmitted *DATA* bits per second per node. Note that, in the ring topology, the throughput per node is upper bounded by half the channel rate (500 Kb/s for 1 Mb/s channel rate), since both nodes A_{i+1} and A_i cannot transmit simultaneously, and thus they must share the channel. The figure plots the throughput versus the offered load λ at each node for $\gamma = \mu$, $\gamma = 2\mu$ and $\gamma = 10\mu$. For each value of γ, the simulation is run for sufficient time so that 100,000 packets are generated in aggregate.

We now discuss the results. When $\gamma = \mu$, which can be considered to be the case when *RTS* Validation is not used, we see that the network can at most deliver a throughput of only about 250 Kb/s per node. However, when we increase γ to 2μ by using *RTS* Validation, the maximum throughput per node increases to about 350 Kb/s. When γ is further increased to 10μ, the maximum throughput per node approaches 410 Kb/s. We thus conclude that even a moderate increase in γ, resulting from the use of *RTS* Validation, can result in a very substantial increase in throughput. This result confirms that *RTS* Validation is a very efficient means of recovering the network throughput lost due to the false blocking problem. Note that according to the IEEE 802.11 specification, the implementation of *RTS* Validation as proposed in Section IV-C is equivalent to using $\gamma \approx 50\mu$ (the result obtained for this case is very similar to that obtained with $\gamma = 10\mu$).

TABLE I
KEY NS PARAMETERS.

Routing protocol	DumbAgent
Propagation model	TwoRayGround
Data rate	1 Mb/s
Carrier sense threshold	2.5×10^{-10}
Receiving range Threshold	2.5×10^{-10}
Packet size	2000 Byte/500 Byte
RTS Threshold	0
LONG RETRY LIMIT	4

B. Simulation of IEEE 802.11 Networks

Our next goal is to determine the effectiveness of *RTS* Validation in IEEE 802.11 wireless networks. For this purpose, we have created a simulation environment using the *NS* simulator [11], which simulates wireless networks based on the IEEE 802.11 standard. Table I summarizes the main parameter settings. In particular, the data rate is set to 1 Mb/s and the *RTS* Threshold is set to 0 so that the *RTS*/*CTS* access control method is always used. Since the standard distribution of NS does not implement *RTS* Validation, we have added this capability to NS.

In order to uncouple the impact of *RTS* Validation from other effects, such as *DATA* packet collisions, we will first consider the ring topology of Fig. 10. The effectiveness of *RTS* Validation in a large random network is explored afterwards.

1) Ring Topology:

a) Throughput and Delay: In this set of simulations, we used the default NS settings. In particular, the SHORT RETRY LIMIT parameter is set to 7. The arrival process at each node A_i (cf. Fig. 10) follows an independent Poisson process with rate λ.

Fig. 12 shows the simulation outcome when the packet size is fixed at 2000 byte. We find that with the standard protocol, the maximum throughput per node is only about 260 Kb/s. On the other hand, when we use *RTS* Validation as described in Section IV-C, the maximum throughput per node becomes about 400 Kb/s. Thus, *RTS* Validation is able to increase the maximum throughput by about 50%. It is important to point out that the increase in throughput occurs at all load values as we expect from the analysis presented in Section V. Indeed, the difference starts becoming visible from $\lambda = 0.2$ Mb/s onwards. Also note the

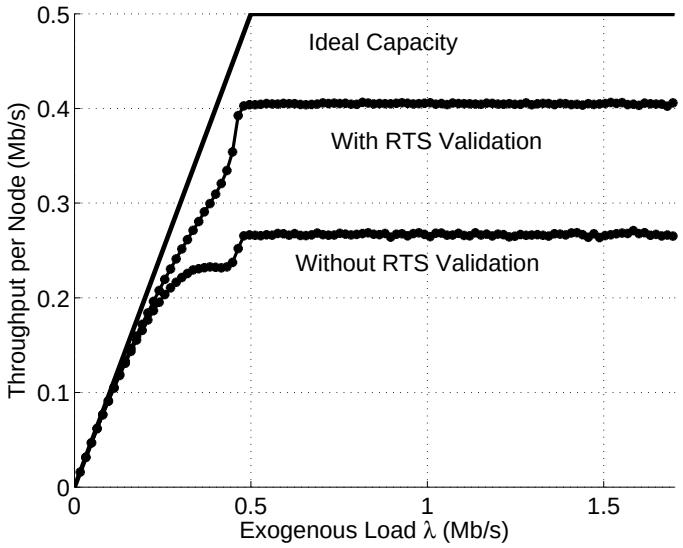


Fig. 12. Ring topology: throughput vs. λ.

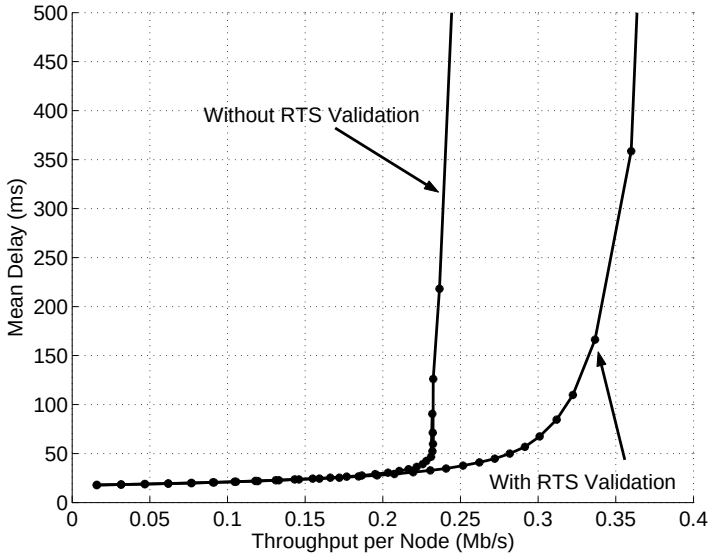
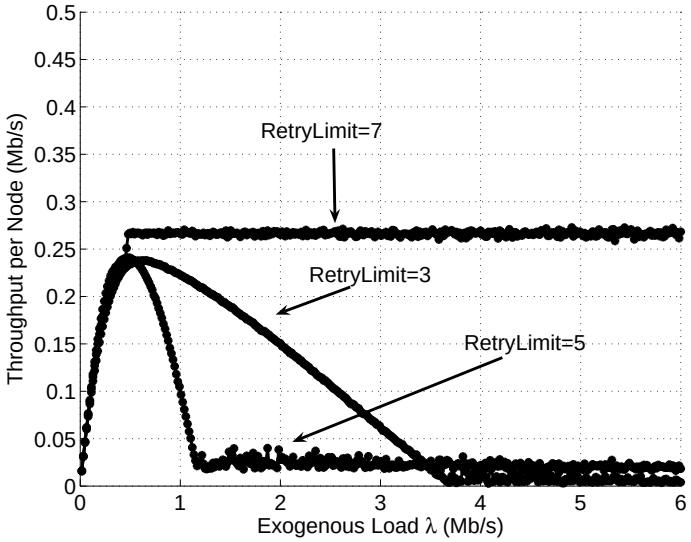


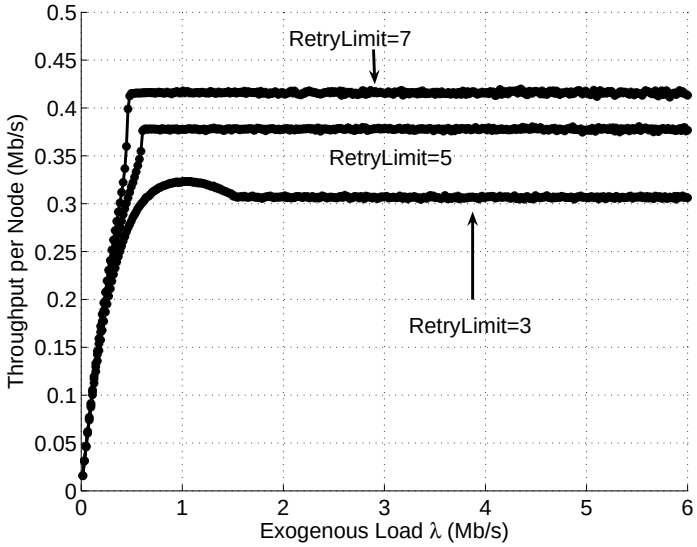
Fig. 13. Ring topology: throughput vs. delay.

similarity between Fig. 12 and Fig. 11. This results confirms that our analytical model accurately captures the behavior of the IEEE 802.11 protocol.

We next evaluate another important performance metric, the packet delivery time, or simply delay. Suppose a packet enters its transmission queue at time t_1 and let the packet be received successfully at time t_2. Then, the delay for this packet is defined to be $(t_2 - t_1)$. The delay calculation includes only successfully transmitted packets. In Fig. 13, we plot the throughput versus mean delay, which is also



(a) Without *RTS*_Validation



(b) With *RTS*_Validation

Fig. 14. Effect of changing SHORT RETRY LIMIT.

known as the *operating curve* [20, Page 366]. The operating curve provides a visual depiction of the efficiency of a protocol: the lower the curve, the better the protocol. It is clear from Fig. 13 that the protocol with *RTS* Validation is superior to the standard protocol. Note that the offset of about 18 ms at the bottom of the figure is due to the minimum time needed to perform *RTS*/*CTS* handshake and the transmission of a *DATA* packet.

b) Effect of modifying the backoff interval: Our next goal is to study the effect of increasing the backoff interval between successive *RTS* attempts on the network throughput. For doing so, we vary the

SHORT RETRY LIMIT parameter. By increasing the values of this parameter, we increase the maximum allowable number of *RTS* transmission attempts as well as the average backoff interval.

Fig. 14 shows the behavior of the network throughput for SHORT RETRY LIMIT set to 3, 5 and 7. For each value of SHORT RETRY LIMIT, we carry out simulation both with and without *RTS* Validation. The packet size is fixed to 2000 bytes. The most interesting point to note is that if *RTS* Validation is not used, then the throughput tends to zero at high load, for small values of SHORT RETRY LIMIT. Thus, the network behaves as a congested network. An intuitive explanation is that for a small value of SHORT RETRY LIMIT, the network rapidly enters the pseudo-deadlock situation described in Section III-D. We remind that to avoid such pseudo-deadlocks, the backoff interval must be larger than the deferral time requested by the *RTS* packets, which is about 16 ms in this case. This condition is satisfied for SHORT RETRY LIMIT=7, for which the maximum size of the congestion window becomes approximately 20 ms, but not for SHORT RETRY LIMIT equal to 3 or 5, where the maximum congestion window is about 2.5 ms and 10 ms, respectively.

On the other hand, when we use *RTS* Validation, we see that the effect of SHORT RETRY LIMIT is much less pronounced. This is because when *RTS* Validation is in use, the false blocking period is very small, and any deadlock situation cannot persist for long. Thus, the throughput maintains its peak value, even for small values of SHORT RETRY LIMIT. However, increasing the length of backoff intervals does improve the peak throughput that the network can achieve with *RTS* Validation. Thus, *RTS* Validation and increasing backoff intervals should be considered as complementary approaches. Note that increasing SHORT RETRY LIMIT beyond a value of about 10 does not appreciably improve the achievable peak throughput, as shown by the simulation results presented in Table II. Note that setting too high a value for SHORT RETRY LIMIT is undesirable as it could lead to head-of-the-line (HOL) blocking problems.

c) Effect of changing the packet size: When channel noise is negligible, larger packet size generally leads to higher throughput since the packet overhead is lower. However, our simulations show that this may not always be the case when false blocking is present. They also illustrate that the solution of increasing

TABLE II

PEAK THROUGHPUT WHEN INCREASING SHORT RETRY LIMIT.

SHORT RETRY LIMIT	7	9	11	13	15
Peak Throughput With *RTS* Validation (Mb/s)	0.41	0.42	0.43	0.43	0.43
Peak Throughput Without *RTS* Validation (Mb/s)	0.27	0.31	0.33	0.34	0.34

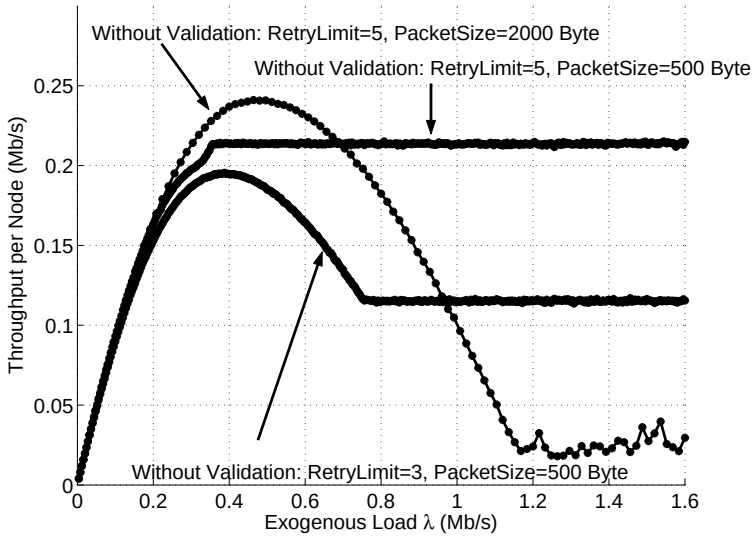


Fig. 15. Without *RTS* Validation: increasing packet size decreases throughput.

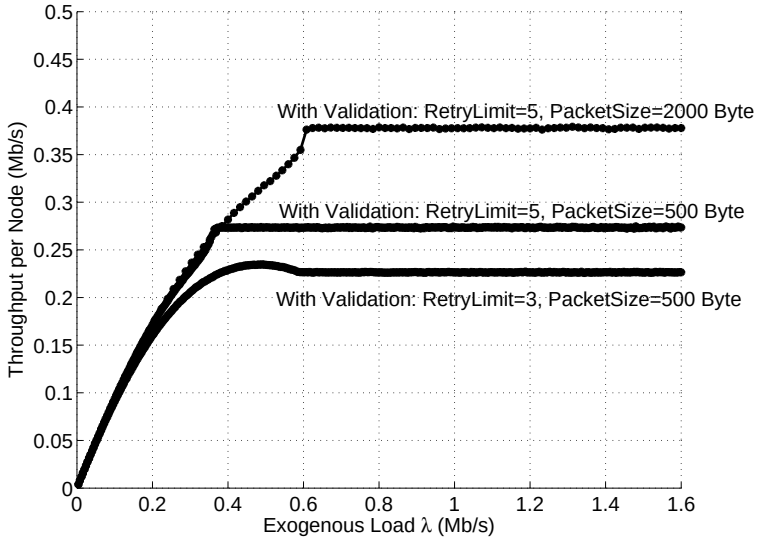


Fig. 16. With *RTS* Validation: increasing packet size increases throughput.

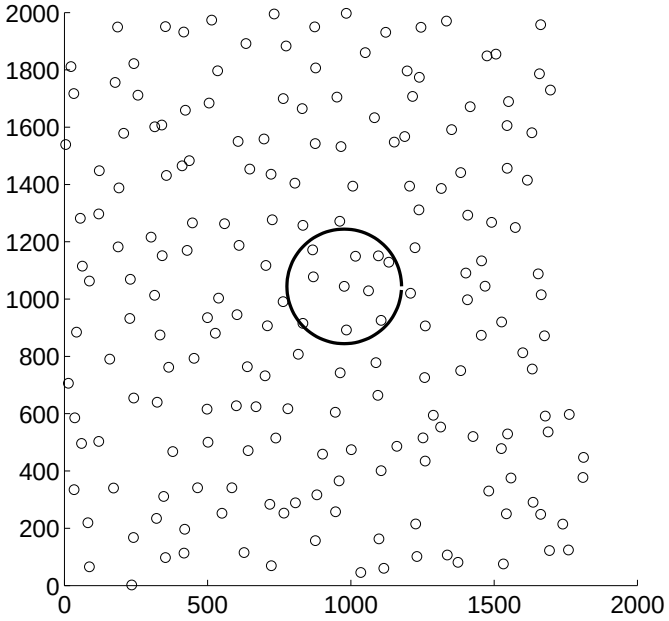


Fig. 17. A random network. The circle represents the transmission range of the node at the center.

backoff intervals alone is not robust.

Fig. 15 plots the throughput as a function of the traffic load for three cases, each without use of *RTS* Validation. The first case is for 500 byte packets and SHORT RETRY LIMIT=3. The saturation throughput (i.e., the throughput at very high load) is only about 60% of the peak throughput in this case. Next, for the same packet size, we set SHORT RETRY LIMIT=5, and find that the saturation throughput and peak throughput become equal. This could lead to the belief that increasing the SHORT RETRY LIMIT parameter is enough to avoid network congestion. However, if we now increase the packet size to 2000 bytes, we find that although the peak throughput increases, the saturation throughput vanishes. We thus observe that, due to false blocking, a network can become congested when the size of packets is increased.

When *RTS* Validation is used, the above situation does not arise, as shown in Fig. 16. Here, as SHORT RETRY LIMIT increased from 3 to 5 (with packet size of 500 bytes), both the peak throughput and saturation throughput increase (note that the axes in Fig. 16 are different from those of Fig. 15). Moreover, the performance improves as the packet size is increased to 2000 bytes, which is the desirable behavior.

2) Random Network: Our analysis presented in Section V is valid for arbitrary networks and predicts that *RTS* Validation reduces the probability that the network is in a state with false blocked nodes. Therefore, we expect the number of *DATA* packet transmission to increase when *RTS* Validation is used. However, in a general network, not all these transmissions are successful due to masked nodes [13], thereby limiting somewhat the gain in throughput.

We simulated a network with 200 nodes spread randomly over a $2000 \times 2000\ m^2$ area. The nodes are initially distributed on an uniform grid and then the coordinates of each node is perturbed by a Gaussian distributed random number with zero mean and 37.5 variance. Fig. 17 shows the resultant network. The transmission range of each node is 200 m. The channel is assumed to be noiseless so that a receiver can always decode a packet unless the packet overlaps with another transmission by a node within its range. The propagation delay is assumed to be negligible. The nodes remain static. These simulation settings allow us to separate the effects of false blocking from other causes of performance degradation, such as channel fading, propagation delay, and mobility.

We used the same network configuration in all the simulations to avoid fluctuations in the simulation outcomes resulting from topology changes. Each node in this network independently generates a traffic of 2000 byte packets. Packets at each node are generated independently according to a Poisson process with average rate λ. For each new packet, one of the neighbors of the source node is selected at random (uniformly) to be the destination. In order to isolate the effects of routing mechanisms from medium access issues, the destination of each packet is only one hop away. Each node uses a single First-In First-Out (FIFO) queue of infinite size. Therefore, the simulation results are not affected by the issues of finite buffer size. For each value of λ, the simulation is run for a sufficient amount time so that the network generates 200,000 packets on aggregate. We used default parameters of the *NS*. In particular, we have SHORT RETRY LIMIT=7. We simulated the network both with and without *RTS* Validation.

The simulation outcomes are shown in Figs. 18 and 19. In Fig. 18 we plot the throughput, counting only the successfully transmitted *DATA* packets. The *RTS* Validation mechanism is again successful in

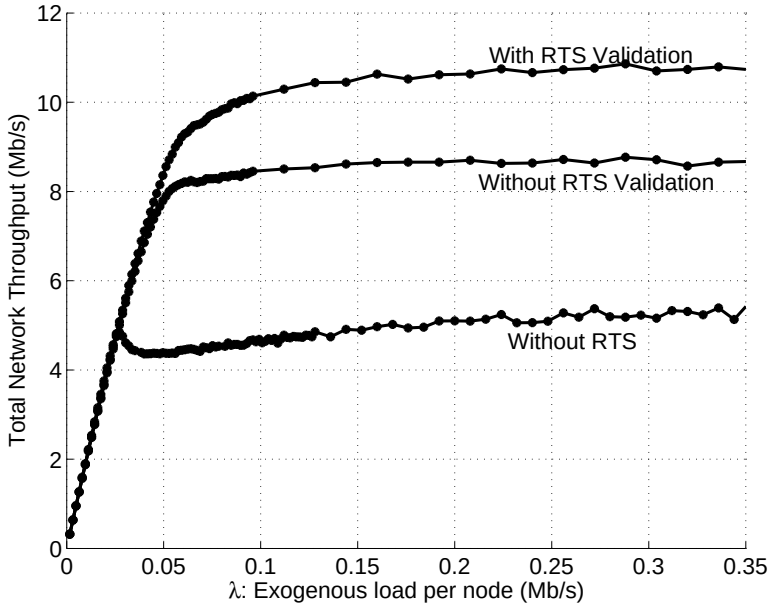


Fig. 18. Random topology: throughput vs. λ.

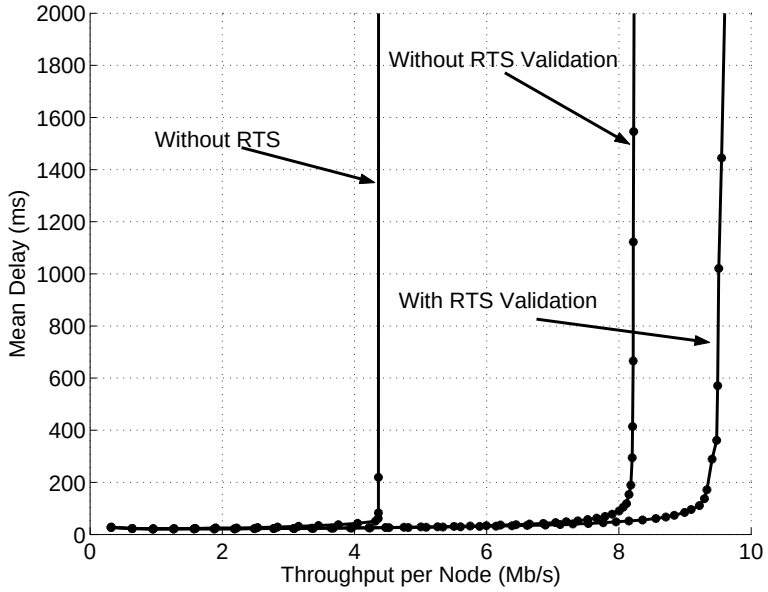


Fig. 19. Random topology: delay vs. throughput.

increasing the peak throughput considerably: from about 8.7 Mb/s to about 10.8 Mb/s, a 25% increase. As a reference, we also show the network throughput if *RTS*/*CTS* handshake is not employed at all. In this case, we see that the throughput is only about 5 Mb/s. As a matter of fact, about 80% of the transmitted *DATA* packets collide when we do not use *RTS*/*CTS* handshake. This demonstrates the need for *RTS*/*CTS* handshake in multi-hop wireless networks.

The operating curves for this network is shown in Fig. 19. As we expect, this figure clearly shows that

the *RTS* Validation is much more efficient than the standard *RTS*/*CTS* access, and the basic access mode performs the worst.

In summary, our simulations verify that the *RTS* Validation mechanism is a very efficient solution to the false blocking problem that improves both the peak and saturation throughput, as well as decreases delay under diverse situations. Moreover, we found that the solution of increasing backoff intervals is neither robust, nor efficient; however, when used in conjunction with *RTS* Validation, it reinforces the effectiveness of *RTS* Validation.

VII. Conclusion

The *RTS*/*CTS* mechanism is widely used in multi-hop networks to avoid collisions caused by hidden nodes. In current implementations of this mechanism, any node that receives an *RTS* or a *CTS* packet inhibits itself from transmitting without using any further information. In this paper, we have shown that this approach leads to false blocking, where a node may become prohibited from transmitting even if no nearby node transmits. Moreover, false blocking can propagate throughout a network resulting in a large number of false blocked nodes in the network and even pseudo-deadlocks. Through extensive simulations on various network topologies, we have shown that false blocking can significantly impact network performance, e.g., by considerably reducing the peak throughput. Furthermore, in some cases, the throughput may start decreasing as the offered load exceeds a certain value. Therefore, the *RTS*/*CTS* mechanism may congest a network instead of stabilizing it.

We have discussed multiple approaches for mitigating the false blocking problem including the solution of increasing backoff intervals, and our proposed solution, *RTS* Validation. The *RTS* Validation is a simple backward-compatible solution: a node that uses *RTS* Validation defers for an entire packet transmission period if *DATA* packet transfer begins, but defers only for a short time if no transmission takes place when *DATA* packet transmission is expected. We have modeled and analyzed the reduction in false blocking probability resulting from the use of *RTS* Validation in networks of arbitrary topology. The analysis demonstrates that *RTS* Validation is an efficient solution that sharply reduces the probability of false

blocking.

By means of simulation, we have evaluated the effect of *RTS* Validation on network throughput and delay. The simulations show that the use of *RTS* Validation improves the network performance in at least three aspects: it stabilizes the network throughput at high load; it increases the peak throughput by as much as 50%, and it significantly reduces the average delay. We have also found from simulations that the solution of increasing backoff interval alone is neither efficient nor robust. However, when implemented in conjunction with *RTS* Validation, network performance is further improved.

ACKNOWLEDGMENT

The authors would like to thank Dr. Jeffrey B. Carruthers for fruitful discussions on the *RTS* Validation mechanism and Soma Ghosh for her help in setting up the NS simulations.

APPENDIX

In Section V, we introduced a continuous-time Markov chain framework that allows to model the qualitative behavior of general IEEE 802.11 wireless networks. In this appendix, we describe in detail the state transitions of this Markov chain.

It is convenient for this purpose to define some auxiliary variables. Let $\mathscr{B}_i$ be the set of neighbors of node i. We let $b_i(s) = 1$ be the blocking indicator of node i, i.e. $b_i(s) = 1$ if node i is blocked in state s, $b_i(s) = 0$ otherwise. Clearly, node i is blocked either because of a genuine or false *RTS* or a *CTS* or if one of the nodes in its neighborhood is transmitting (node i may not have heard the *RTS* or *CTS* if it were masked). Mathematically, $b_i(s) = 1$ if and only if either $i \in B_j$ for some $B_j \in \mathbb{B}$, or $rts_{ij} = 1$ (for some j), or $cts_{ij} = 1$ (for some j), or $t_j(s) = 1$ for some $j \in \mathscr{B}_i$.

We define $N_i(s)$ ($\subseteq \mathscr{B}_i$) to be the set of nodes that can hear a control packet sent by node i in state s. In other words, these are the neighboring nodes of node i that sense the channel idle in state s, i.e. $j \in N_i(s)$ if and only if $\forall\ k \in \mathscr{B}_j$, $t_k(s) = 0$.

Type	From State s	To State s'	Rate	Event
1.	q_i	$q_i + 1$	$p_{ij}\lambda_i$	An exogenous arrival at node i, with node j as the destination.
	D_i	$D_i \overset{j}{\Leftarrow},\ \ j \in \mathscr{B}_i$		
2.	q_i	$q_i - 1$	μ	Completion of a successful *DATA* transmission by node i.
	$u_i = 1$	$u_i = 1$		
	w_i	$w_i = 0$		
	$t_i = 1$	$t_i = 0$		
	$rts_{ji},\ j \in \mathscr{B}_i$	$(rts_{ji} - 1)^+$		
	$cts_{jd_i},\ j \in \mathscr{B}_{d_i}$	$(cts_{jd_i} - 1)^+$		
	D_i	$D_i \overset{d_j}{\Rightarrow}$		
3.	q_i	q_i	μ	Completion of an unsuccessful *DATA* transmission by node i.
	$u_i = 0$	$u_i = 1$		
	w_i	$w_i + 1$		
	$t_i = 1$	$t_i = 0$		
	$rts_{ji},\ j \in \mathscr{B}_i$	$(rts_{ji} - 1)^+$		
	$cts_{jd_i},\ j \in \mathscr{B}_{d_i}$	$(cts_{jd_i} - 1)^+$		
4.	$q_i > 0$	q_i	σ_{w_i}	An unsuccessful *RTS* attempt to node d_i by node i.
	w_i	$w_i + 1$		
	$t_i = 0$	$t_i = 0$		
	$b_i = 0$	$b_i = 0$		
	$b_{d_i} = 1$	$b_{d_i} = 1$		
	f	$f + 1$		
	$\mathbb{B}$	$\mathbb{B} \overset{N_i(s)}{\Leftarrow},$		
	$u_k,\ k \in V_i$	$u_k = 0$		
5.	$q_i > 0$	q_i	σ_{w_i}	A successful *RTS* attempt to node d_i by node i.
	$t_i = 0$	$t_i = 1$		
	u_i	$u_i = 1$		
	$b_i = 0$	$b_i = 0$		
	$b_{d_i} = 0$	$b_{d_i} = 1$		
	$rts_{ji},\ j \in N_i$	$rts_{ji} + 1$		
	$cts_{jd_i},\ j \in N_{d_i}$	$cts_{jd_i} + 1$		
	$u_k,\ k \in V_i \cup V_{d_i}$	$u_k = 0$		
6.	$f > 0$	$f - 1$	$f \cdot \gamma$	Completion of the j-th false blocking period.
	$\mathbb{B}$	$\mathbb{B} \overset{B_j}{\Rightarrow}$		

TABLE III

THE TRANSITIONS OF THE MARKOV CHAIN $\mathbf{X}(t)$.

Next, we denote $d_i(s)$ to be the destination for the packet at the head of the queue at node i. Thus, $d_i(s)$ is the first element in the array $D_i(s)$.

Finally, let $V_i(s)$ denote the set of vulnerable nodes around node i in state s. These nodes are receiving a packet in state s and any transmission from node i will destroy this packet. Clearly, $j \in V_i(s)$ if and only if node j is in the neighborhood of node i and node j is receiving a packet from another node k. Mathematically, these conditions translate into $j \in \mathscr{B}_i$, $j = d_k(s)$ for some k and $t_k(s) = 1$.

With this notation, Table III show all the transitions possible out of any state. There are four events that cause a state transition: (i) an exogenous arrival at a node; (ii) an *RTS* transmission attempt, which may or may not be successful; (iii) completion of a *DATA* packet transmission, which may or may not

have collided; and (iv) completion of a false blocking period. So, there are six types of transitions for each node i, i.e. at most $6N$ transitions can occur out of a given state. However, not all transitions are possible out of each state. The variable values in the second column defines the conditions which must be satisfied for that transition. For example, a node can send an *RTS* only if it is not blocked in that state. So, we must have $b_i = 0$ for a Type 4 or Type 5 transition. The third column defines the new state. The rate of transition is shown in the fourth column, and the fifth column describes the event that causes this transition. We use the notation $(x)^+$ to denote $\max\{0, x\}$. The insertion of the integer j at the end of the array D_i is denoted by $D_i \stackrel{j}{\Leftarrow}$, and the deletion of the first element from the array D_i is denoted by $D_i \stackrel{d_i}{\Rightarrow}$. Similarly, $\mathbb{B} \stackrel{N_i(s)}{\Leftarrow}$ denotes the insertion of the set $N_i(s)$ in the collection $\mathbb{B}$, and $\mathbb{B} \stackrel{B_j}{\Rightarrow}$ denotes the deletion of the j-th set from $\mathbb{B}$. The destination of each new packet sent by a node is chosen with a fixed probability among its neighbors, but other Markovian policies can be accommodated similarly.

REFERENCES

[1] Mesh Networks corp. (now part of Motorola corp.). [Online]. Available: http://www.meshnetworks.com/

[2] Tropos corp. [Online]. Available: http://www.tropos.com/

[3] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. IT-46, no. 2, pp. 388–404, March 2000.

[4] L. Kleinrock and F. A. Tobagi, "Packet switching in radio channels: Part 2 - the hidden node problem in carrier sense multiple access modes and the busy tone solution," *IEEE Transactions on Communications*, vol. COM-23, no. 12, pp. 1417–1433, 1975.

[5] Z. J. Haas, J. Deng, P. Papadimitratos, and S. Sajama, "Wireless ad hoc networks," in *Wiley Encyclopedia of Telecommunications*, J. G. Proakis, Ed. Wiley, December 2002.

[6] S. Ray, D. Starobinski, and J. B. Carruthers, "Performance of wireless networks with hidden nodes: A queueing-theoretic analysis," *Elsevier Journal of Computer Communications (Special Issue on the Performance of Wireless LANs, PANs, and Ad-Hoc Networks)*, vol. 28, pp. 1179–1192, June 2005.

[7] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, ANSI/IEEE Std. 802.11, 1999.

[8] P. Karn, "MACA - a new channel access method for packet radio," in *ARRL/CRRL Amature Radio 9th Computer Networking Conference*, September 22 1990, pp. 134–140.

[9] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A media access protocol for wireless LANs," in *Proceedings of ACM SIGCOMM '94*. ACM, 1994, pp. 212–225.

[10] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. Knightly, “Ordered packet scheduling in wireless ad hoc networks: Mechanism and performance analysis,” in *Proceedings of MOBIHOC’02*. EPFL Lausanne, Switzerland: ACM, 2002.

[11] [Online]. Available: http://www.isi.edu/nsnam

[12] L. Kleinrock and F. A. Tobagi, “Packet switching in radio channels: Part 1 - carrier sense multiple access modes and their throughput-delay characteristics,” *IEEE Transactions on Communications*, vol. COM-23, no. 12, pp. 1400–1416, 1975.

[13] S. Ray, J. B. Carruthers, and D. Starobinski, “Evaluation of the masked node problem in ad-hoc wireless LANs,” *IEEE Transactions on Mobile Computing*, vol. 4, no. 5, pp. 430–442, September/October 2005.

[14] V. Bharghavan, “Performance evaluation of algorithms for wireless medium access,” in *IEEE Performance and Dependability Symposium ’98*. Raleigh, NC.: IEEE, 1998.

[15] S. Ray, J. B. Carruthers, and D. Starobinski, “RTS/CTS-induced congestion in ad-hoc wireless LANs,” in *IEEE Wireless Communication and Networking Conference (WCNC)*, March 2003, pp. 1516–1521.

[16] H. Li, P. Shenoy, and K. Ramamritham, “Scheduling communication in real-time sensor applications,” in *Proceedings of the Tenth IEEE Real-Time/Embedded Technology and Applications Symposium (RTAS04)*, Toronto, Canada, May 2004.

[17] T. Shigeyasu, T. Hirakawa, H. Matsuno, and N. Morinaga, “Two simple modifications for improving IEEE802.11DCF throughput performance,” in *IEEE Wireless Communication and Networking Conference (WCNC)*, 2004.

[18] D. Chen, J. Deng, and P. K. Varshney, “Protecting wireless networks against a denial of service attack based on virtual jamming,” MOBICOM 2003, Student Poster.

[19] S. Asmussen, *Phase-type distributions and related point processes: Fitting and recent advances*. in Matrix-analytic Methods in Stochastic Models, S.R. Chakravarthy and A.S. Alfa, Marcel Deekker, New York, 1996.

[20] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. Prentice Hall, 1991.