

Intelligent Jamming Attacks, Counterattacks and (Counter)² Attacks in 802.11b Wireless Networks

Mithun Acharya, David Thuente

Department of Computer Science

North Carolina State University

Raleigh NC USA 27695

E-mail: {mpachary, thuente}@csc.ncsu.edu

Abstract

It has been shown that reservation based MAC protocols such as 802.11b DCF are susceptible to *Intelligent Jamming* wherein a protocol aware jammer can bring down the network throughput essentially to zero by using very limited energy, see [1]. We propose and analyze intelligent jamming techniques based on “misbehaving” MAC level access timings. An additional intelligent jamming technique based on *fake* RTS reservations is proposed in [6] to efficiently disrupt any general reservation based wireless network. [6] also proposes a protocol modification as a counterattack to the above denial of service attack. We instantiate these attacks and counterattacks to an access point based wireless network using 802.11b DCF protocol and perform a simulation in OPNET to study its effect on 802.11b network throughput. Furthermore, we present additional ways to attack the proposed countermeasure in terms of network congestion. In doing so, we compare the efficiency of these various intelligent jamming attacks and their counterattacks. Using OPNET simulations, we discuss the relative merits and demerits of the various denial of service attacks.

Introduction

Wireless networks now enjoy widespread commercial implementation because of their ease of use and setup. Wireless access point based 802.11b hotspots are commonplace with more and more mobile users accessing the Internet wirelessly through various high-end mobile devices already available in the market at affordable prices. But on the flip side, since accessing wireless media is much easier than tapping a wired network, security becomes a serious concern when implementing any wireless network. While [2] and [3] deal mainly with confidentiality and authentication related security attacks, in this paper, we consider a particular class of Denial of Service (DoS) attacks called *jamming*. For our purposes, jamming is any attack to deny service to legitimate users by generating noise or fake protocol packets or legitimate packets but with spurious timing. We also study countermeasures to this particular class of DoS attacks. The jamming in [6] and in this paper is at the MAC layer. The jamming results in [1] in some cases depended on the TCP performance. We will address this again later. We have removed the transport and the network layers from our model for the results in this paper. This is an appropriate way to test MAC layer DoS attacks without having TCP heavily influence the results.

The most trivial way of disrupting a wireless network is by generating a continuous high power noise across the entire bandwidth in the vicinity of transmitting and/or receiving

wireless nodes. The device that generates such a noise is called a *jammer* and the process is called *jamming*. However, jamming can be made much more energy efficient and less obvious if the jammer operates using knowledge of the protocol. Energy efficient stealthy jammers are a goal. As demonstrated in [1], the jammer can choose to disrupt selected control packets for a very short time and bring down the whole network. Such jammers, which jam the network with the knowledge of the protocol, are termed as *protocol aware jammers*. Jamming and its countermeasures have a long history in military applications where the jammer is used to degrade enemy radio communications. Cheap jammer nodes can be scattered across the enemy battlefield with the purpose of generating noise to bring down the enemy network, to disrupt sensor networks, or even to disrupt command and control networks. It becomes very important for such jammer nodes to conserve as much energy as possible to ensure their longevity and lower their probability of detection; and hence, provide effective disruption. On the other hand, jammers can also be used to jam commercial wireless hotspots for “fun” and possibly, profit. These attacks call for efficient counterattacks or prevention techniques. In this paper, we inspect the various protocol aware jamming attacks and some possible countermeasures. Unfortunately, it is much easier to disrupt wireless networks than to defend against such attacks.

The rest of the paper is organized as follows. We review the basic 802.11b DCF access mechanism. Then we present the simulation model used in our OPNET experiments. In the next section, we introduce the different types of jamming as given in [1]. In this paper, we study a protocol aware jammer, which claims channel bandwidth by sending fake RTS control messages to the access point. To introduce the catastrophic effects of protocol aware jamming, we present new results to show how misbehaving nodes, not adhering to the underlying MAC protocol, bring down the network throughput to unacceptable levels. We then present the fake RTS attack [6]. We show the process model in OPNET for a fake RTS jammer and present our results. We implement the corresponding counterattack algorithm, *CTSR*, at the access point and study how effectively the access point can minimize the throughput degradation from the protocol aware jammer. We then present additional jammer attacks that are possible against the access point that uses the *CTSR* algorithm. We outline metrics for evaluating various jamming techniques. Finally, we compare the fake RTS jammer with the jamming techniques proposed in [1] and propose a few hybrid jamming attacks. We complete the paper by drawing conclusions and giving directions for further research.

802.11b MAC Layer

Basic CSMA/CA Mechanism

The basic CSMA/CA mechanism is shown in Figure 1. If the medium is sensed idle for at least the duration of DIFS (with the help of the clear channel signal (CCA) of the physical layer), a node can access the medium at once. If the medium is busy, nodes have to wait for the duration of DIFS before entering a contention phase. Each node now chooses a random backoff time within a contention window and additionally delays the access for this random amount of time. If a station does not get access to the medium in the first cycle, it stops its back off timer, waits for the channel to be idle again for DIFS time and starts the timer again. As soon as the timer expires, the node accesses the medium. Thus, longer waiting stations have the advantage over newly entering stations in that they only have to wait for the remainder of their backoff timer from the previous cycles. If collision occurs, then the station backs off exponentially. ACKs have higher priority over data. So a station wanting to send ACK waits only for SIFS time.

CSMA/CA with RTS/CTS

The basic CSMA/CA mechanism cannot solve the *hidden terminal* problem. The problem occurs if one station can receive two others, but those stations cannot receive each other. If both of these stations sense the channel idle and send the data to the station which can see both, collision occurs at the receiver. Figure 1 illustrates the use of RTS (Request to Send) and CTS (Clear to Send). After waiting for DIFS (plus a random back off time if the medium was busy), the sender can issue a RTS packet. The RTS packet includes the receiver of the anticipated data transmission and the duration of that whole data transmission. This duration specifies the time interval necessary to transmit the whole data frame and the acknowledgement related to it. Every node receiving the RTS now has to set its Net Allocation Vector (NAV) in accordance with the duration field. The NAV specifies then the earliest point in time at which the station can try to access the medium again. Following a successful RTS, CTS is sent after SIFS interval ($SIFS < DIFS$). After a successful reception of CTS, DATA and ACK follow, with the duration of SIFS between the frames. If a RTS frame undergoes collision, the station backs off exponentially.

Simulation Model

In this section, we present the simulation setup, the wireless node model, its attributes, and finally the traffic model used for our simulations.

Wireless 802.11 LAN Model

We use the MAC and below of the 802.11 Wireless LAN model from [8] by installing the lab files for [8] over OPNET. We do not use the network layer or the transport layers since this interferes with the experiments we are doing with the MAC protocols and the disruption of the network due to MAC level attacks. TCP can strongly decrease the performance of the

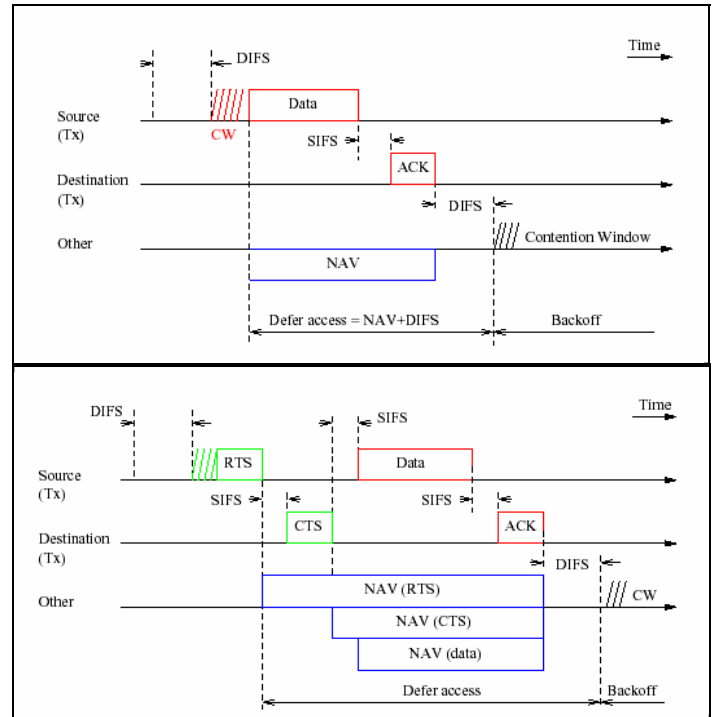


Figure 1: The basic CSMA/CA and RTS/CTS mode in 802.11b networks

network due to MAC layer attacks. This is a very interesting study and one we have looked at but it is not the subject of this paper. Figure 2 shows the Infrastructure Basic Service Set scenario that is used in this paper to study the effects of jamming on network throughput. In this scenario, we have an 802.11b wireless network that spans a building with three floors. We have an access point that is connected to a server through a switch. The access point resides on the second floor. There are nine similar stations evenly spread out among the top three floors. Even though it is customary to have at least one access point per floor, we have only one access point for the whole wireless network so that there are nine nodes accessing the access point. We could have put all nine nodes on one floor and the results would not have changed at all. We chose to do an extension of the model from [8] which does not skew the experimental results. The figure also shows an intelligent RTS fake jammer node placed near the center of the setup.

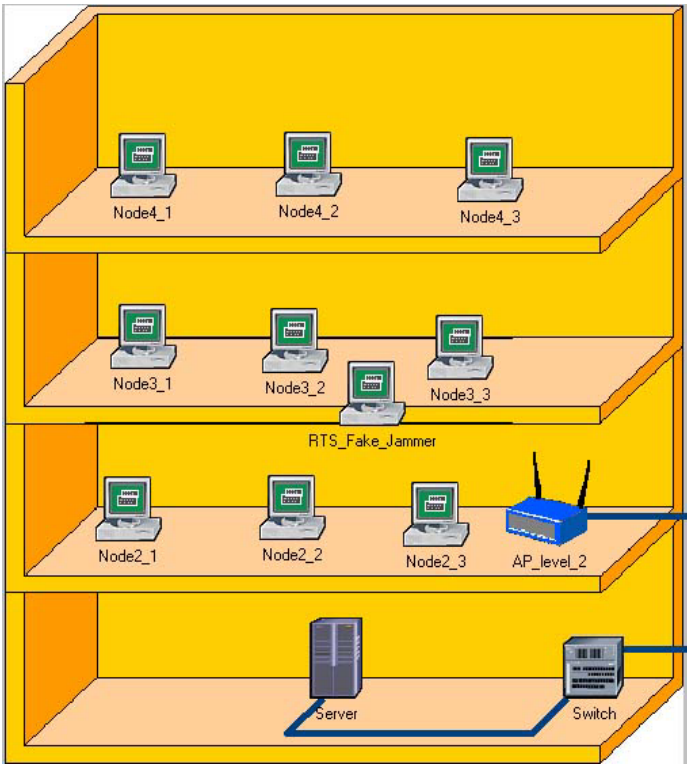


Figure 2: Infrastructure BSS with intelligent RTS fake jammer

Wireless Node and Traffic Model

The nine similar nodes are spread across three floors. *wlan_station_adv* is used as the node model for all the stations. The node model is shown in Figure 3. The node model consists of a wireless LAN MAC layer (*wireless_lan_mac*) interfaced with the higher layer (*source* and *sink*) by a MAC interface (*wlan_mac_intf*). The interface receives packets from the higher layer and generates random addresses for them to the other eight nodes. The wireless LAN receiver (*wlan_port_rx0*) accepts any incoming packets from the physical layer and passes them onto the wireless LAN MAC layer. The wireless transmitter (*wlan_port_tx0*) receives the packet from the MAC layer and passes it on to the physical medium. This node model uses the source and sink module to simulate the higher layers (IP, TCP, Application, etc.). The source model generates packets sent to random destination addresses. The packets received at the destination nodes are discarded at the sink module. The traffic model used by the station nodes is shown in Figure 4.

The packet size distribution is exponential with a mean of 1000 bytes. The interarrival time is $\exp(.1)$ for all the nodes unless otherwise specified.

The wireless attributes of a station node are shown in Figure 5. The RTS threshold is 128 bytes. This means that data packets, whose size together with the WLAN MAC header of 28 bytes, exceed this threshold, require completion of a successful RTS/CTS frame exchange before the transmission of the actual

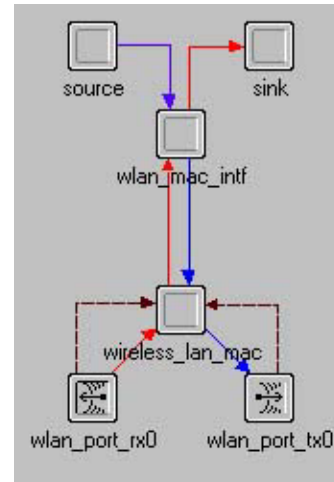


Figure 3: Node model

?	name	Node3_2
?	model	wlan_station_adv
?	Destination Address	Random
?	Traffic Generation Parameters	(...)
?	Start Time (seconds)	constant (5)
?	ON State Time (seconds)	constant (1200)
?	OFF State Time (seconds)	constant (0)
?	Packet Generation Arguments	(...)
?	Interarrival Time (seconds)	exponential (.1)
?	Packet Size (bytes)	exponential (1000)
?	Segmentation Size (bytes)	No Segmentation
?	Stop Time (seconds)	Never
	Wireless LAN	

Figure 4: Traffic model

packet over the radio channel. Since the packet size is exponentially distributed with mean of 1000 bytes, RTS/CTS exchange is required for most of the packets. As can be seen from Figure 5, all the wireless station nodes and the access point use Direct Sequence Spread Spectrum at the physical layer. All the nodes employ the DCF basic CSMA/CA access mechanism. The nodes transmit with a power level of .001 Watts and at a maximum data rate of 1 Mbps. Packets received at a node with power less than $7.33E-14$ Watts will be considered noise and will not change the status of a receiver to busy. The packet transmissions with a power higher than this threshold are considered as valid. Unless the default transmission power is changed, all the WLAN packets should reach their destinations with sufficient power to be valid packets if the propagation distance between the source and destination is less than 300 meters as required by the IEEE 802.11b WLAN standard [9]. The farthest distance between any two nodes, including the access point, in our simulation scenario is less than 50 meters.

?	name	wireless_lan_mac
?	process model	wlan_mac
?	icon name	processor
	Wireless LAN	
?	Address	promoted
?	Wireless LAN Parameters	(...)
?	Rts Threshold (bytes)	128
?	Fragmentation Threshold (bytes)	None
?	Data Rate (bps)	1 Mbps
?	Physical Characteristics	Direct Sequence
?	Transmit Power (W)	0.001
?	Packet Reception-Power Threshol...	7.33 E-14
?	Short Retry Limit	7
?	Long Retry Limit	4
?	Access Point Functionality	Disabled
?	Channel Settings	(...)
?	Buffer Size (bits)	1024000
?	Max Receive Lifetime (secs)	0.5
?	Large Packet Processing	Drop
?	BSS Identifier	Auto Assigned
?	PCF Parameters	Disabled
?	Roaming Capability	Disabled

Figure 5: Wireless attributes of a station node

In the simulation model considered here, all the nodes are static and always transmit with a power level of .001 Watts. The intelligent jammer node is placed at the approximate center of the network and is again static. The numerical values, which we obtained in this paper, are based on these particular parameter settings. In particular they are also based on the modulation techniques used at the physical layer. These affect the SNR needed for successful receptions and hence the numerical results presented in this paper. The physical layer characteristics are the same as given in [8] and the modulation technique is Binary Phase Shift Keying. The DIFS, SIFS, and Slot times are set to 50, 10, and 20 μ seconds respectively as specified in the 802.11b model. The relative merit of the jamming techniques will vary somewhat with different modulation techniques but the general ranking will likely remain the same. Error correction techniques will affect the effectiveness of most of the jamming techniques discussed here and in [1]. [6] gives a very limited introduction to error correction for one particular case. Error correction will be considered in a future paper. The values may vary if some or all of the above attributes are changed. For example, we can have a mobile jammer instead of a static one and that would change the numerical values. The numerical values might also change if we have a jammer close to the access point (and possibly affect more packets) or have all the nodes clustered around the jammer or the access point. We retain the same setting for all the simulations and hence we can make valid comparisons.

Intelligent Jamming In 802.11b Networks

In [1], we introduced different kinds of possible jamming in 802.11b networks. We list the jamming attacks here:

1. Trivial Jamming
2. Simple Periodic Jamming
 - a. Continuous Low Power Jamming
 - b. Bursty High Power Jamming
 - c. Busy Jamming
3. Intelligent Jamming
 - a. CTS Corruption Jamming
 - b. ACK Corruption Jamming
 - c. DATA Corruption Jamming
 - d. DIFS Wait Jamming

We categorized various types of jamming that were proposed in [1] into three categories: trivial jamming, simple periodic jamming and intelligent jamming. In trivial jamming, continuous high power noise is generated to bring down the network. This is the least energy efficient of all jamming methods and is demonstrated for benchmarking purposes only. In simple periodic jamming, a periodic noise pulse is generated irrespective of the packets that are put on the network. The power levels and the pulse duration are parameters of simple periodic jamming that can be modified. In continuous low power jamming, a low powered continuous pulse is sent out by the jammer. In bursty high power jamming, short pulses of higher power are sent out at regular intervals to disrupt the medium. In busy jamming, a short pulse is generated every DIFS interval so that the stations always find the network busy. The results on bursty jamming in [1] depend heavily on the TCP layer.

In [1], we showed that simple periodic jamming can be three to four orders of magnitude more effective than trivial jamming while intelligent jamming can be as much as five orders of magnitude more effective than the trivial jamming. In intelligent jamming, the jammer jams with the knowledge of the protocol. The jammer is put in promiscuous mode and it continuously listens to the network. The jammer can distinguish between the control packets and the data packets by analyzing the packet headers. We identified four types of intelligent jamming. In CTS corrupt jamming, the jammer listens or waits for any arbitrary RTS packet promiscuously. For every RTS packet the jammer sees, it starts counting down for a period of SIFS from the end of RTS packet and then issues a short jamming pulse that disrupts the CTS packet that would follow the RTS packet. Since the CTS packet does not get through, no data is ever transferred. This attack is among the most efficient (energy use) of the methods presented here as the jammer is active for only the short interval of time necessary to disrupt the CTS packet. The process model for the CTS corrupt jammer is shown in Figure 6. We modified the process model of the *jam_pulsed_adv* model available with the OPNET package to construct the CTS corrupt jammer. The other types of jamming presented in [1] include ACK corruption jamming wherein an ACK packet is destroyed after DATA is sent, DATA corruption jamming wherein a DATA packet is destroyed after CTS is received, and finally DIFS wait jamming, wherein a short pulse is sent by the jammer after sensing the medium idle for DIFS time to disrupt either a RTS packet or a DATA packet. In a network with high traffic, it is very likely that if the medium is idle for DIFS time, a transmission occurs immediately after that. We showed that intelligent jamming is five orders more effective than trivial

jamming. These results, however, do depend heavily on the TCP characteristics.

Node Misbehavior Jamming

For our considerations, a misbehaving node is a network node that unilaterally has modified its MAC protocol so as to attempt to gain an unfair advantage in transmitting its messages or transmitting messages so as to deny others the legitimate use of the network. The interesting and somewhat surprising result is that if the misbehaving node wants to gain an extra share of the network bandwidth then it is possible that its behavior will result in lesser throughput for itself as well as all of the other nodes in the network.

One could easily imagine a network that has not been as responsive to command and control functions and a node manager on the network reconfiguring its MAC to improve its chances to gain access. A supervisor might even suggest such a change for a heavily loaded network. What we will see is that the performance suffers severe deterioration for all the nodes on the network as well as the node that was intending to gain an advantage.

A user in an access point based 802.11b hotspot might selfishly choose not to adhere to the MAC protocol to get an unfair share of the bandwidth. An easy way to accomplish this is by not adhering to the random exponential backoff-algorithm. The user might either decide to use smaller backoff or choose no backoff at all. [4] presents the results of the effect of misbehaving nodes that select smaller backoff times, on the total network throughput and proposes a solution to solve the misbehavior attack. This solution requires a significant modification to the 802.11b protocol (in short, instead of the nodes selecting the random back off times, the access point selects the *next* random backoff time and conveys it to the sending nodes via CTS packets) and hence is not realistically viable as a countermeasure. In this section, we use different and more effective misbehaving patterns than those proposed in [4] and we show that two misbehaving nodes is more effective than just one. We consider the effect of misbehaving node(s) that chooses a small backoff with and without the DIFS wait period. In the first case, the misbehaving node waits for DIFS time and just a single slot time before entering the contention phase. In the second case the DIFS time is ignored and its backoff time is just a single slot. The random exponential backoff is not used in either case. In both cases, if these nodes find the medium busy, they wait till the medium becomes free. In the first case, the node enters the contention phase only after waiting for a period of DIFS while in the second case it immediately enters the contention phase. But in the contention phase the misbehaving node chooses a single slot time for its backoff.

We also show that two misbehaving nodes are able to cause a more severe degradation of network performance than just one such node. Moreover, the energy required for two such nodes is only very slightly increased from just one such node. In addition, two misbehaving nodes are able to accomplish this in a stealthier manner as will be explained later. To our knowledge no one has previously proposed this two node misbehavior in

this fashion. All of these result hold whether the misbehavior is generated by greed for bandwidth or by trying to create DoS attacks.

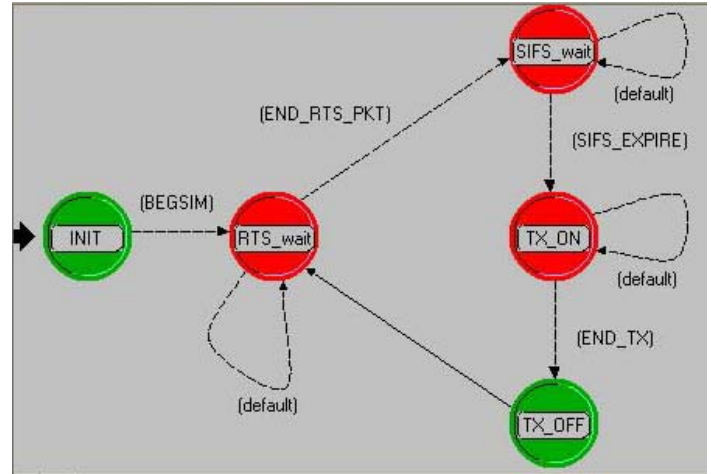


Figure 6: Process model for CTS corrupt jammer

The node misbehavior attack has some similarities to the RTS faking attack [6] discussed in the next section. In the RTS faking attack, a jammer sends out fake RTS packets, whenever it finds the medium idle and reserves the channel for maximum time duration possible thereby degrading the network throughput with very little effort. The misbehaving attack we discuss in this section, on the other hand, captures the medium only when it has data to send by not adhering to the DIFS wait and/or backoff protocol (values here are actually DIFS = 0, 50 and Slot = 20 microseconds). Hence other nodes will get a chance to transmit and we expect the misbehavior attack to be less effective in terms of degrading the network throughput as compared to the intelligent jammer that uses fake RTS attack. This misbehavior can be implemented unilaterally on a node and the other nodes in the network will not be equipped to even detect the misbehavior, much less to offer any countermeasures. In this way this DoS attack is far superior to many other attacks including the fake RTS attack that can be instantly recognized by all nodes that are listening to the network. It is useful to study the presence of a few misbehaving nodes in access point based wireless networks and demonstrate that just by not adhering to the underlying MAC protocol, a malicious node can substantially disrupt the network throughput. For a heavily loaded network, the extent of the decrease in network throughput is severe. This attack creates some serious questions about the stability of the 802.11b networks to withstand even minimal attacks.

At this point, we present the MAC layer process model in detail for the 802.11b wireless LAN shown in Figure 7. After initialization in the INIT and BSS_INIT state, the process model waits in the IDLE state. Whenever it has data to send and the medium is not idle, it enters the DEFER state waiting for the medium to become idle. Once the medium becomes idle, it waits for DIFS time and enters the contention phase in the BACKOFF state. The node backs off for the random back off duration and enters the TRANSMIT state if the medium is idle. Once in the TRANSMIT state, it transmits the packet after which it reaches the FRM_END state and waits for the corresponding response in

the WAITS_FOR state if it expects a response packet (e.g., it expects a CTS after SIFS time, if the last packet sent was RTS). The process model with all the states and transitions is clearly shown in Figure 7.

To establish a benchmark, we measure the total throughput for the access point based wireless network shown in Figure 2 without any jammers and without any misbehaving nodes. The result is shown in Figure 9. We use the same traffic profile as shown in Figure 4 for all our experiments unless otherwise specified. Without any disruption, the average throughput is about 0.44 Mbps at the end of 600 seconds.

For the misbehaving node that chooses the wrong backoff time, we retained the process model as shown in Figure 7 but the backoff was selected in the BKOFF_NEEDED state as opposed to randomly set in BKOFF_NEEDED. For the results presented here we set backoff = 1 slot. To model the MAC layer for misbehaving node that adheres neither to the DIFS wait nor the backoff protocol, we removed the DEFER, BACKOFF and BKOFF_NEEDED states from the process model showed in Figure 7 and route the transitions to the correct states; we also made a few changes to the process header and function block. We get the process model as shown in Figure 8. The single or double node(s) to misbehave are chosen randomly and the simulation is run for 600 seconds. For the misbehaving nodes, we choose a packet size of uniform (140-160) with an interarrival time of $\exp(.01)$.

The offered load for the network with one misbehaving node is then:

Misbehaving node:

$$(150 \text{ bytes} + 28 \text{ byte header}) * 100 \text{ pkts/sec.} * 8 \text{ bits/pkt} \\ = 142,400 \text{ bps.}$$

Sum of regular nodes:

$$(1000 \text{ bytes} + 28 \text{ byte header}) * 10 \text{ pkts/sec.} * 8 \text{ bits/pkt} * 8 \\ \text{nodes} = 657,920 \text{ bps.}$$

The offered load for the misbehaving node is larger than that of the other nodes and its packets are only 15% of that of the regular packets. As is shown in Figure 10, the throughput decreases by about 40% to 0.26 Mbps in the presence of a single misbehaving node that adheres to the DIFS wait protocol (DIFS = 50 μ seconds) but not to the backoff protocol. The backoff is set to 1 slot or 20 μ secs. The graph in Figure 10 also shows the throughput results for a single misbehaving node that adheres neither to the DIFS wait protocol nor the backoff protocol (DIFS = 0 and Slot = 20 μ seconds.) and the result is a network

throughput of 0.28 Mbps. One might expect that much of the throughput for this case would be from the misbehaving node since it transmits in the DIFS time and before any regular node gets a chance to transmit. If that were the case it could be a better DoS attack since less “good” data might be transmitted even though it might be more costly in terms of energy required for transmissions.

In order to determine whether the DIFS = 50 or 0 μ seconds is better we needed node throughput information. In terms of the DoS attack the fundamental questions are: how much of the good network traffic actually gets through (and to some extent how much “bad” traffic gets through.) We also need to know how much traffic the misbehaving node sends.

Figures 11 and 12 provide the answers to all of these questions. From Figure 11 we can see that more good and more bad traffic are received by network nodes for the DIFS = 0 case than for the DIFS = 50 case. Consequently, the DIFS = 0 is less effective for a DoS attack. From Figure 12, we see that more traffic is sent by the misbehaving node for the DIFS = 0 case than for the DIFS = 50 case. We see that the DIFS = 0 case takes more energy and is less effective in preventing good traffic from being received and in reducing overall throughput. Hence we can conclude that the DIFS = 0 case is inferior to the DIFS = 50 case for this scenario. Moreover the DIFS = 0 violates the MAC protocol in a very obvious way and hence is much more easily detected.

The data traffic sent by the misbehaving and a good node are shown in Figure 12. The effect of the misbehaving node that uses the standard DIFS = 50 and only chooses a backoff of one slot is more severe on the throughput. There are several reasons for this. First, the misbehaving node manages to “waste” the DIFS time and transmit after the end of DIFS time and second, by transmitting in the usual window after the end of the DIFS time, the misbehaving node introduces more contention with the RTS transmissions of the good nodes. This is shown by the reduced throughput of both good and bad nodes for the DIFS = 50 case. From this point forward, we consider misbehaving nodes that adhere to the DIFS wait but not to the backoff protocol.

We now introduce two misbehaving nodes instead of one and run the experiment to record a throughput decrease of about 60% to 0.18 Mbps (Figure 13). The data traffic sent by misbehaving and good nodes is shown in Figure 14. The traffic sent for the two misbehaving nodes is just slightly over the traffic sent for the single misbehaving node and yet they provide an additional 36% decrease in network throughput over the single misbehaving node case.

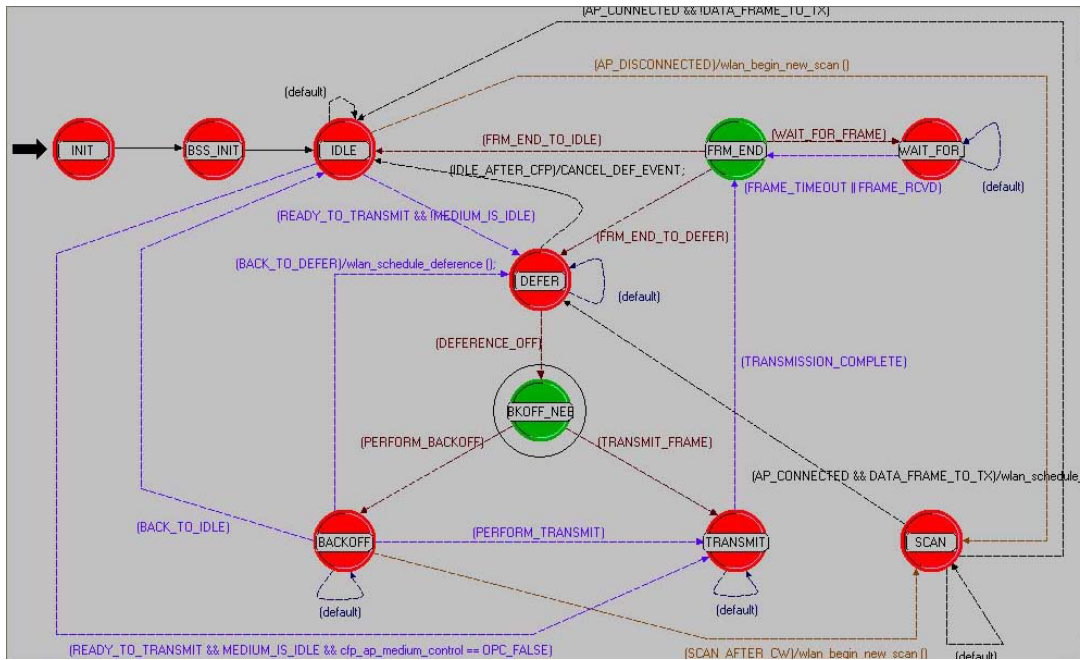


Figure 7: 802.11b wireless LAN MAC process model

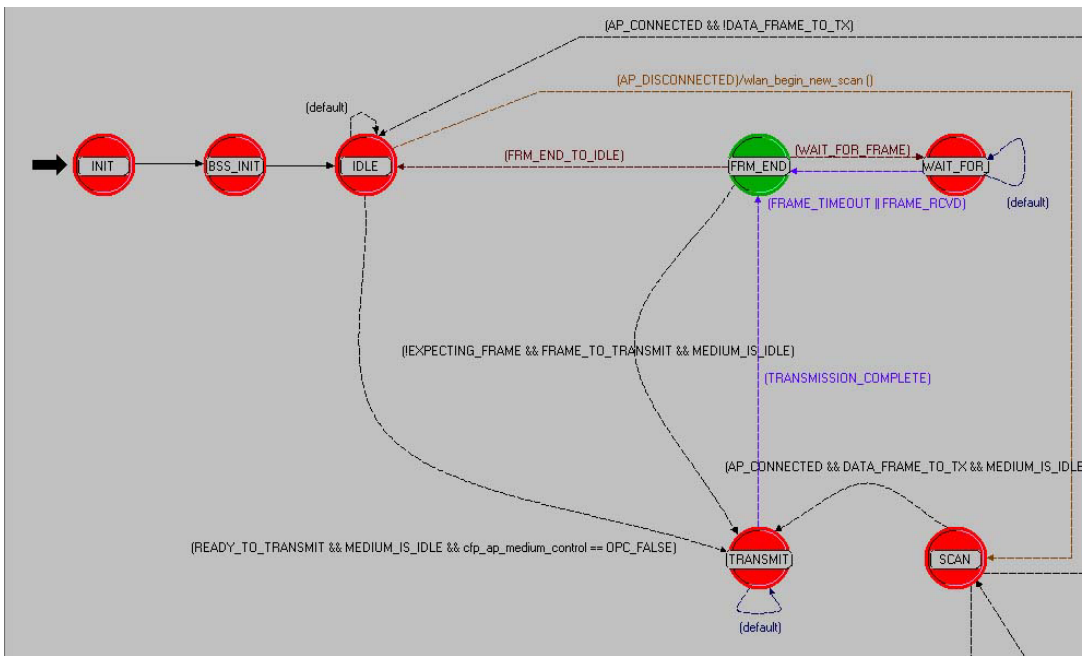


Figure 8: MAC process model of a misbehaving node that does not adhere to DIFS Wait and backoff protocol

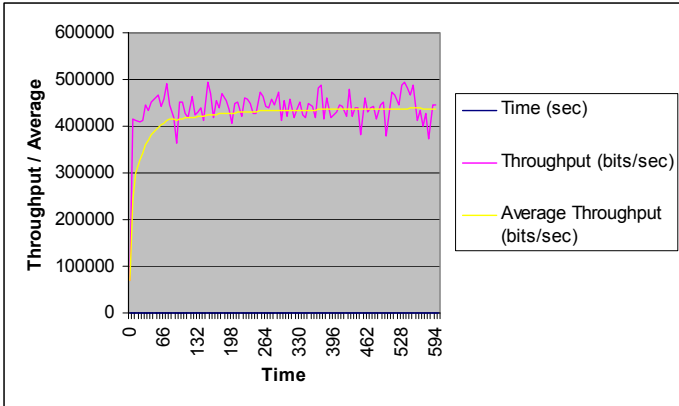


Figure 9: Network throughput without a jammer

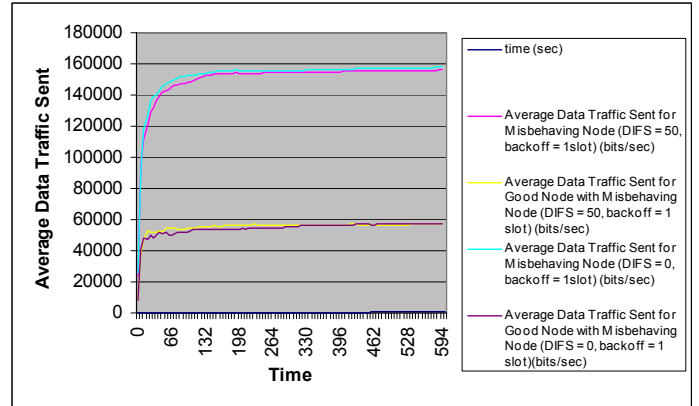


Figure 12: Average data traffic sent by a misbehaving and good node with a single misbehaving node

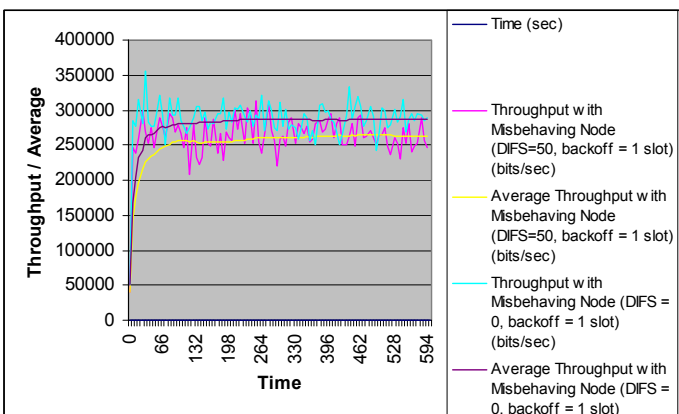


Figure 10: Total Network throughput with a single misbehaving node with uniform (140-160) packet size and exp(.01) packet interarrival

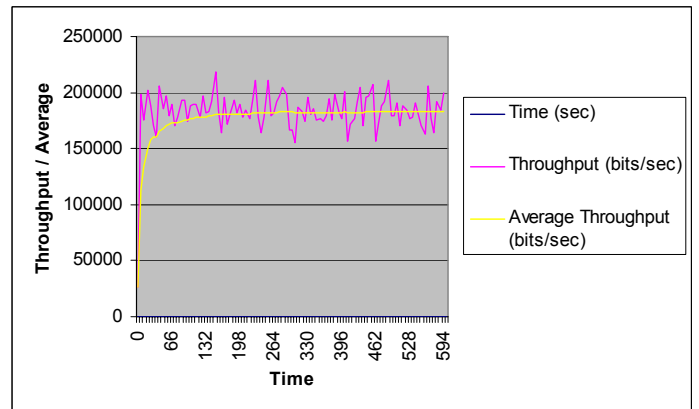


Figure 13: Network throughput with two misbehaving nodes with uniform (140-160) packet size and exp(.01) packet interarrival

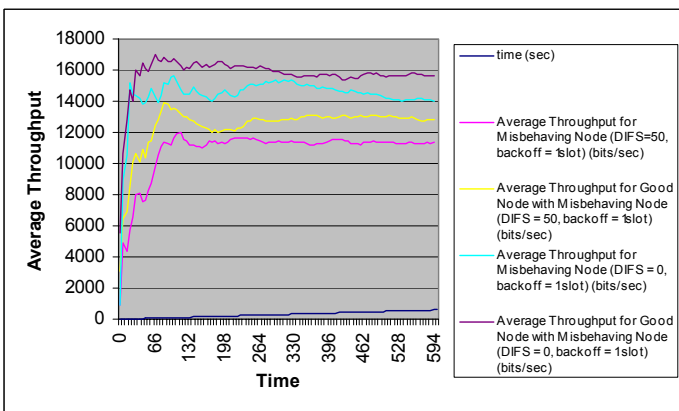


Figure 11: Average throughput for the misbehaving node and a good node in presence of a single misbehaving node

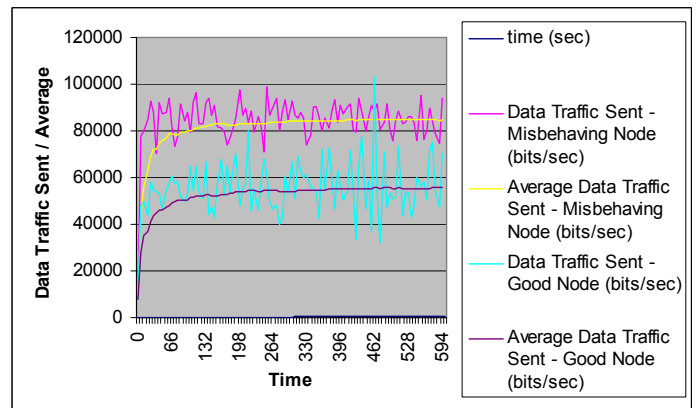


Figure 14: Data traffic sent by a misbehaving node and good node with the two misbehaving nodes having uniform (140-160) packet size and exp(.01) packet interarrival

RTS Fake Jamming

The misbehavior attack is an effective DoS attack and/or the malicious user enjoys higher bandwidth at the expense of other nodes that see the network busy much of the time. However, if the sole purpose of the attacker is to launch a denial of service attack then the misbehaving attack is not among the best in terms of energy efficiency. It may be among the best in terms of stealth since its communication load is nearly the same as everyone else's but it gets access a little sooner and more frequently. The attack relies on idle medium capture whenever the misbehaving station has data to send. An attacker might disrupt the network to the same extent or more so by launching an attack called as RTS fake attack [6]. The RTS fake attack is launched in the following manner. The RTS fake intelligent jammer promiscuously listens to the medium. If it sees an RTS packet being sent on the medium, it destroys it by jamming with probability q . If the jammer finds the medium idle, then it sends its own RTS packet reserving the medium for $M \cdot \text{RTS_PACKET_SIZE}$ data duration with probability p . Here RTS_PACKET_SIZE is the size of the RTS control packet and M is a simulation parameter that controls the length of the data reservation time (NAV). However, on receiving a CTS from the access point, the jammer does not send the data. In the RTS packet, the NAV_DURATION is set to $M \cdot \text{RTS_PACKET_SIZE}$ duration and hence all the stations hearing the fake RTS will set their NAV vector to this value and postpone medium sensing until that time in the future. The duration field in the CTS packet sent to the jammer also reflects this future time and the nodes hearing the CTS also delay their medium sensing until that time. The jammer then awakes after the expiration of NAV_DURATION and repeats the process. During this period, other stations assume that the medium is

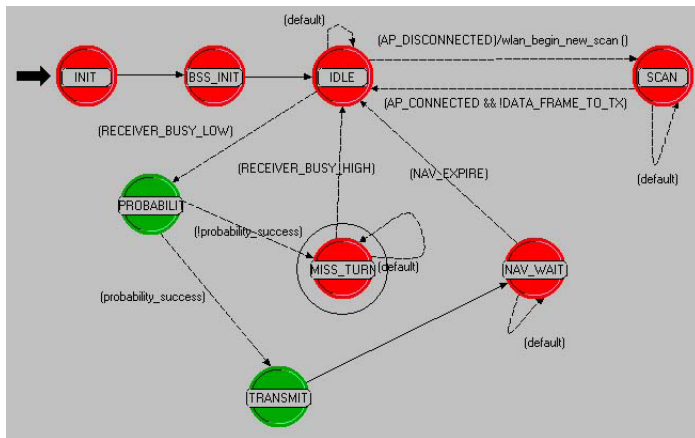


Figure 15: MAC process model for RTS fake intelligent jammer

busy and will not send any data. Effectively the network appears jammed. In the jammer we designed, we set $q = 0$ as it might not be possible for the jammer to simultaneously detect a RTS transmission in the medium and also destroy it. Even if it is possible it might not be efficient since it is more difficult to disrupt an established transmission to a node than to hinder the initial establishment of that transmission. This is particularly true for FHSS. The modified MAC process, which achieves this type of jamming, is shown in Figure 15.

In the IDLE state, whenever the medium becomes idle, the jammer transmits the RTS packet with probability $\text{probability_success}$ (p). After transmitting the fake RTS and reserving the medium for $M \cdot \text{RTS_PACKET_SIZE}$ duration, the jammer waits for that much time in the NAV_WAIT state. With probability $(1 - p)$, the jammer does not send a RTS and enters MISS_TURN state and waits till the medium becomes busy again. Then the process is repeated. We insert an intelligent RTS fake jammer node with the process model shown in Figure 15 into the 802.11b network and run the simulation for 10 minutes. The throughput drops by about 85%. The results are shown in the graph of Figure 16. The jammer only uses a small RTS packet to reserve very large data duration (M times the size of RTS packet). Hence the attack is effective and also energy efficient. The average throughput drops as the time progresses because of the increased contention among nodes that hardly get to transmit. This in turn increases the network congestion leading to throughput deterioration. In the next section, we present a countermeasure discussed in [6] known as the Reservation Revoke Protocol or the CTSR protocol.

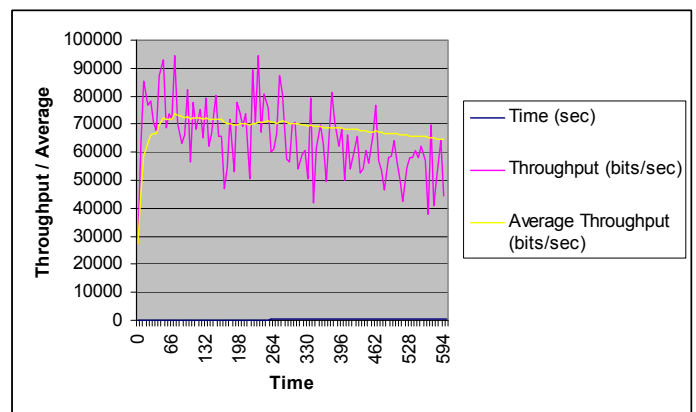


Figure 16: Network throughput with intelligent jammer faking RTS packets reserving for $M = 25 \cdot \text{RTS_PACKET_SIZE}$ data with $p = 0.5$ probability

Counterattacks – The CTSR Protocol

The RTS fake attack can be countered by a modification to the 802.11b protocol. The RTS fake jammer sends fake RTS packets reserving the medium but does not send any data in the reserved period. To counter this malicious behavior, once

the access point sends out the CTS in response to the jammer's fake RTS, it starts monitoring the channel for actual data transmission. The access point checks the medium every $M' \cdot \text{RTS_PACKET_SIZE}$ slots ($M' \ll M$) to see if the medium is busy (due to data transmission or packet collision). If the medium is not busy, the access point revokes the rest of the reserved bandwidth by sending out a CTSR packet (Reservation Revoke) which is essentially a CTS packet with $\text{NAV_DURATION} = 0$. As other nodes in the network receive this CTSR packet, they reset their NAV vector to zero and hence the medium appears to be free again; the nodes enter into contention phase. The CTSR protocol should be implemented at the access point. However, we simplify the process model by implementing the CTSR protocol in the RTS fake jammer itself. This is possible since in our simulation setup, all nodes can hear all other nodes and the jammer node can easily fake the access point. The process model becomes simplified if the jammer fakes the access point and sends the CTSR packet on behalf of the access point. The jammer, after sending the RTS packet, starts counting down for $M' \cdot \text{RTS_PACKET_SIZE}$ slots. During this period, the jammer gets the CTS packet in response to its RTS packet if the RTS packet is delivered successfully. If the RTS packet is indeed successfully received at the access point, the jammer transmits a CTSR packet (with $\text{NAV_DURATION} = 0$) revoking the rest of the reserved bandwidth. It is easy to see that the above implementation is equivalent to the CTSR protocol implementation at the access point. The process model for such a jammer is shown in Figure 17. Higher M/M' value improves the throughput since the fake reservation is cleared earlier. We will present an effective counter to this counter attack in the next section.

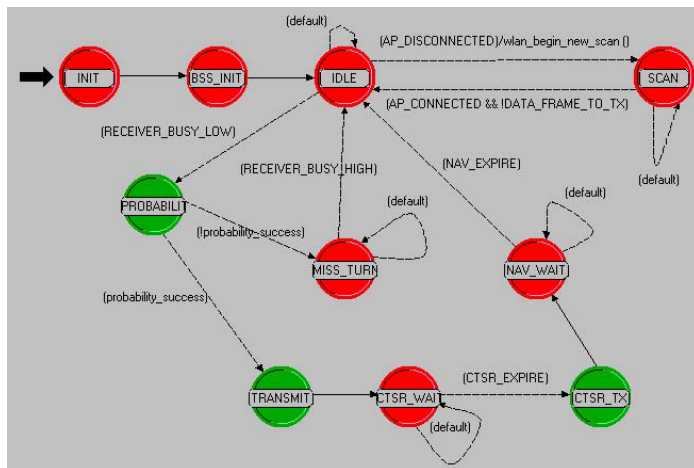


Figure 17: Modified MAC process model for the RTS fake jammer to demonstrate the CTSR protocol

Attacks on CTSR Protocol

CTSR Jamming Attack

An obvious way for the intelligent jammer to attack the CTSR protocol would be to jam the CTSR packets. But this can be

countered by making M' smaller. We can make the jammer expend more energy by making the ratio M/M' larger because the access point will retransmit the CTSR and each should be jammed. This jamming can be done though since the timing of the CTSR packets is determined by M' . This avoids the problem that can be encountered in jamming the original RTS packets. However, the CTSR protocol does limit the bandwidth reservation capability of the jammer.

Congestion Inducing Attacks

We could have implemented the fake CTSR attack proposed in [6] but have not for reasons given below. Since there is no authentication of the reservation channel in 802.11b, the jammer could send fake CTSR packets whenever a legitimate user is transmitting a data packet. The users that do not hear the data transmission but hear the fake CTSR see the medium as idle. Following the reception of CTSR packet, the nodes start sending out RTS packets that collide with the ongoing data transmission. The access point fails in decoding the data packets due to excessive interference by these RTS packets. This counter attack is not viable in our scenario since we assume all nodes hear all other nodes. Hence, the jammer broadcasting the CTSR packet when there is an ongoing valid transmission would make it obvious the jammer is a malicious node. Even in a network that does not assume that all nodes are within a single broadcast link, nodes that hear the ongoing broadcast and the spurious CTSR would know the true character of the jammer. For these reasons we do not think that the secondary attack proposed in [6] merits further consideration.

A hybrid attack that combines the fake RTS reservations just discussed and the ACK jamming attack presented in [1] would make an effective DoS attack. The attack would go as discussed for the fake RTS attack. If CTSR is used to limit the effectiveness of the fake RTS attack, the jammer should just ignore the CTSR packet. However, when any data gets sent, the jammer can jam the subsequent ACK and thereby requiring retransmissions and increased congestion. If it is important to do nearly complete DoS, then the jammer should jam the end of the data packet as determined by the NAV vector. Clearly this attack is not stealthy since the fake RTS is not stealthy but it is nearly as energy efficient as the fake RTS attack and more efficient than jamming the CTSR messages.

Comparison of Various Jamming Attacks

There is a variety of metrics that one can use to compare various jamming attacks. Clearly the following are all relevant:

1. Energy efficient
2. Low probability of detection
3. Stealthy
4. Strong DoS, complete if so desired
5. Maintain behavior consistent with or close to the protocol standard

6. Authenticated or unauthenticated users
7. Strength against error correction algorithms
8. Strength against physical layer techniques such as FHSS, DSSS, CDMA.

Which of these are most important will depend greatly on the application that is being addressed. Energy efficient may be the most important for sensor networks that are expected to last a long time. Strong DoS may be the key component if even a few successful messages will compromise your situation such as behind enemy line. Low probability of detection is crucial if you need to maintain a modestly long-term presence in the area. It would take many additional pages to rate each of the techniques on all of these metrics. We will instead make some comments on and summaries of various techniques and their hybrids.

We have presented different types of jamming attacks from trivial jamming to RTS fake jamming and simulated them in OPNET. Protocol aware jamming is possible in 802.11b because of the lack of authentication of the control and data packets. If the underlying MAC supports authentication, then the access point would discard any RTS packet from a node that is not in the network. Though RTS fake attack might fail in an authenticated network, it is still possible to launch simple periodic jamming and the intelligent jamming attacks proposed in [1]. Simple periodic jamming relies on sending periodic short pulses while CTS corrupt, ACK corrupt, DATA corrupt jamming rely on disrupting select control packets. Since we know the protocol timing for all parts of the RTS/CTS protocol, it is possible to do CTS corrupt, ACK corrupt, DATA corrupt jamming without being part of the network. The efficiency of these techniques for DSSS with no error correction was discussed in [1].

Hybrid attacks for misbehaving nodes:

We can do the misbehaving node attacks with two nodes and always use the standard DIFS period but instead of using a single backoff time vary the backoff between the two misbehaving nodes. Formally:

- Node 1: with probability p use backoff = 1 slot
with probability $1-p$ use backoff in $[cw/2, cw]$
Node 2: with probability $1-p$ use backoff = 1 slot
with probability p use backoff in $[cw/2, cw]$.

The effect will be to make the nodes appear closer to normal and to still maintain nearly the same level of effectiveness for DoS.

Hybrid attacks for intelligent jamming:

There are many ways to combine the intelligent jamming that was presented in [1] with the methods present here. All of the simple periodic jamming attacks, the CTS corrupt, ACK corrupt, DATA corrupt jamming should be done with different probabilities assigned to each.

Attacks by unauthenticated users:

We should note that attacks that do not depend on being an authenticated user have a significant advantage. Most of the techniques presented in [1] do not require that the user be a member of the network. In particular we look at the ACK jamming attack. We know the data packet is longer (28 byte header) than the control packets and hence we can recognize it and the end of the packet transmitted. The jammer waits for SIFS time and then jams the network and the corresponding ACK.

Conclusions and Future Work

We studied various protocol aware jamming attacks that can be launched in an access point based 802.11b network. We started by presenting the various low power jamming attacks ranging from trivial jamming to intelligent jamming attacks such as CTS corrupt jamming. We then presented our simulation results showing the effect of misbehaving nodes that do not adhere to the underlying MAC protocol. The network throughput suffered drastically even in the presence of a single misbehaving node and more so with two misbehaving nodes. We then instantiated the RTS fake attack proposed in [6] to a standard 802.11b network and studied the effect of a single RTS fake jammer on the network throughput. We then presented our process model to demonstrate the CTSR protocol counterattack for the RTS fake jamming attack. We proposed a congestion based attack on the CTSR protocol counterattack. Finally, we made a qualitative comparison of different types of jamming that were presented in this paper.

All of the above results should be repeated for 2, 5, and 11 Mbps implementations of IEEE 802.11b. In addition, many of these results can be repeated directly for IEEE 802.11a/e/g. As was stated earlier, these results hold only for networks with no error correction at the physical layer. We need to investigate the error correction algorithms used for the various implementations and incorporate these into the OPNET model. In all the experiments here we assumed a base station oriented network. We would like to conduct experiments wherein the IEEE 802.11b is functioning in the ad hoc mode. The results for ad hoc networks will definitely be different than what we saw in the base station oriented network. There is scope for more forms of intelligent jamming in ad hoc networks (maybe we could exploit the network layer protocols also) and we reserve this exploration for future work. Apart from applying the attacks presented in this paper to different 802.11 networks, we intend to explore the possibilities for more sophisticated jamming and counter-jamming attacks. We expect to extend the results here to other wireless protocols such as MIL-STD-188-220C. We believe that the jamming attacks and counterattacks presented in this paper are a solid first step.

References

- [1] Acharya, M., T. Sharma, D. Thunte, D. Sizemore, “Intelligent Jamming in 802.11b Wireless Networks”, OPNETWORK 2004, August 2004.
- [2] Bellardo, J., S. Savage, “802.11 Denial-of-Service Attacks: Real Vulnerabilities and Practical Solutions”, USENIX 2003.
- [3] Fluhrer, S., I. Martin, I. A. Shamir, “Weakness in the key scheduling Algorithm of RC4”, LNCS, 2259, 2001.
- [4] Kyasanur, P., N. Vaidya, “Detection and Handling of MAC Layer Misbehavior in Wireless Networks”, DSN 2003.
- [5] Leon-Garcia, A., I. Widjaja, “Communication Networks,” McGraw Hill, Boston, 2000.
- [6] Negi, R., A. Rajeswaran, “DoS Analysis of Reservation Based MAC Protocols”, ICC 2005.
- [7] Schiller, J., “Mobile Communications”, Addison-Wesley Longman Publishing, Boston, 1999.
- [8] Lab for Session 1332: Planning and Analyzing Wireless LANs and Mobile IP Networks, OPNETWORK 2003.
- [9] IEEE Std 802.11b-1999/Cor 1-2001 Standard for wireless LAN medium access control (MAC) and physical layer (PHY) specifications, 2001.