Intelligent Jamming in 802.11b Wireless Networks

Mithun Acharya, Tanu Sharma, David Thuente, David Sizemore
Department of Computer Science
North Carolina State University
Raleigh NC USA 27695
E-mail: {mpachary, tsharma, thuente }@csc.ncsu.edu, dwsizemo@ncsu.edu

Abstract

Total physical jamming in wireless networks can be realized by generating continuous high power noise in the vicinity of wireless receiver nodes. The purpose of this paper is to show, using OPNET simulations, that similar jamming effectiveness can be achieved with much lower total energy. In the first half of the paper, we study the effects of periodic jamming with various power levels on the network throughput. IEEE 802.11b has two different DCF modes: basic CSMA/CA and RTS/CTS mechanism. Based on these parameters and the numerical results obtained, we discuss the effectiveness of periodic jamming of these modes. In the second half of the paper, we look at the jamming mechanisms that exploit crucial intervals and control messages in both of these modes and hence add some *intelligence* to the jammer. We use OPNET 10.0 to simulate various types of jamming which we propose. *Intelligent* jamming, which jams with the knowledge of the protocol, is shown to perform significantly better than the trivial continuous high power noise jamming while also retaining its effectiveness. Intelligent jamming appears to be one to two orders of magnitude more efficient than periodic jamming and up to five orders of magnitude more efficient than trivial jamming.

Introduction

The most trivial way of disrupting a wireless network is by generating a continuous high power noise in the vicinity of transmitting and/or receiving wireless nodes. For the rest of the paper, we refer to this kind of jamming as *Trivial Jamming*. The device that generates such a noise is called a *Jammer* and the process is called *jamming.* Jamming finds battlefield applications wherein a jammer can be used to disrupt enemy wireless communication. Cheap jammer nodes could be scattered into the enemy battlefield with the purpose of generating noise to bring down the enemy network, to disrupt sensor networks, or even command and control networks. It becomes very important for such jammer nodes to conserve as much energy as possible to ensure their longevity and hence effective disruption. In this paper, through simulation, we show that jamming effects achieved by continuous high power noise can be achieved by intelligent jamming with much lower energy requirements.

We start with simulating the most trivial case, jamming with high power continuous noise. We then decrease the power level and the frequency of noise bursts with the goal of decreasing the total jamming energy while trying to retain the same effectiveness of high power continuous jamming. For the basic CSMA/CA mode, we conduct various simulations with different power levels and pulse durations and study their effects on the network throughput. Specifically, we look at the continuous low power and bursty high power jamming. After this, we look at the power levels and duration that are required to *fool* the nodes into assuming that the network is *busy* because of some valid packet transfer. The jammer periodically spits out noise packets thus making other nodes in its vicinity sense the medium as busy. The energy requirements for just keeping the network busy this way will be shown to be significantly less than the previous two cases. Jamming by generating periodic noise pulses will be termed as *Simple Periodic Jamming*. We discuss the effectiveness of simple periodic jamming on throughput for the two different IEEE 802.11b modes.

Finally, we realize even lower energy jamming through what we call *Intelligent Jamming*. Corrupting a few crucial control or data messages at the right time is enough to make the network non-operational. Hence, we put the jammer in promiscuous mode and jam only specific control or data packets. We modified the OPNET 10.0 provided jammer, *jam_pulsed_adv*, to add this intelligence. We propose four ways in which the basic jammer can be modified to make the jamming more effective.

The rest of the paper is organized as follows. We start with explaining the simulation model we used for this paper and detail the simulation parameters used in our experiments along with a few assumptions. In the next section, we look at the Simple Periodic Jamming case. In this part, we present the results for three cases: low power continuous noise jamming, high power bursty noise jamming, and finally, the type of jamming which makes the medium look busy all the time. With the results for the Trivial and Simple Periodic Jamming cases as background, in the next section, we present our algorithms and simulation results for the Intelligent Jamming case and show that the total energy required is much less than in the previous two cases. We complete the paper by drawing conclusions and giving direction for further research.

A couple of basic assumptions that are used throughout this study include: no error correction is implemented at the physical layer and the jammer is broadcasting across the entire IEEE 802.11b spectrum. Many of these results could be achieved with significantly less energy by having the jammer use the direct sequence or the frequency hopping sequence of the radio transmitters. For the most part, this is possible for IEEE 802.11b networks since both transmitting sequences can be obtained by listening devices. Here the contributions show which jamming techniques were most effective and show these could be achieved with very low energy. The results in this paper can be done for the direct sequence or the frequency hopping sequence using perhaps as much as two orders of magnitude less energy. The shortest effective jamming period is 1E-006 seconds since that is one bit for the 1Mbps network. We are also assuming that the jammers are able to generate a signal at a very precise time and very quickly in response to a received signal.

Simulation Model

We use OPNET 10.0 to study the effects of jamming on the network throughput. In this section, we present the wireless network model, the wireless node and access point model, the jammer model and finally the traffic model used for our simulations.

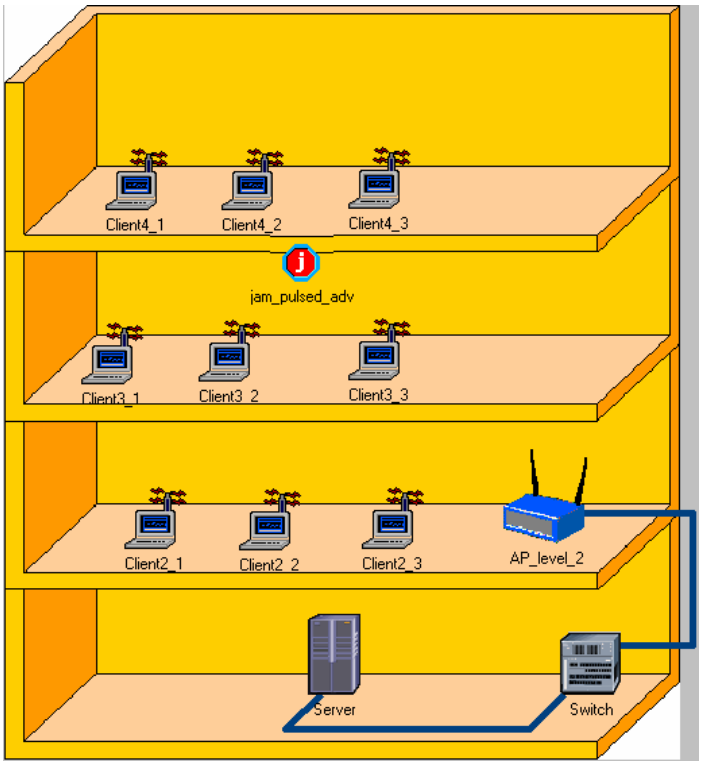

Figure 1: Infrastructure BSS with a Jammer

Wireless 802.11 LAN Model

We use the 802.11 Wireless LAN model from [1] by installing the lab files for [1] over OPNET 10.0. Figure 1 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 a 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 workstations 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 to simplify analysis. This simplification only reduces the total network throughput for every case we consider and does not skew the experimental results.

Wireless Node and Access Point Model

The nine similar nodes are spread across three floors. As can be seen from Figure 2, all the wireless workstation 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 RTS/CTS mechanism is not switched on for the Simple Periodic Jamming case. We also analyze the effect of switching on the RTS/CTS mode for the Simple Periodic Jamming case. 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 packet if the propagation distance between the source and destination is less than 300 meters as required by the IEEE 802.11b WLAN standard [2]. The farthest distance between any two nodes, including the access point, in our simulation scenario is less than 50 meters.

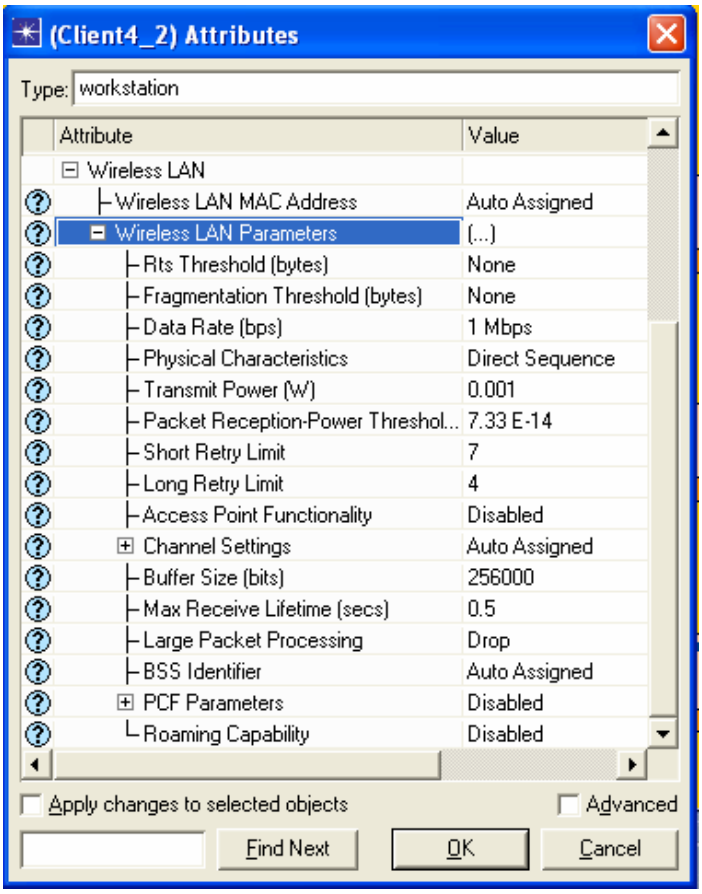

Figure 2: Wireless Workstation Node Model

Jammer Model

We use the *jam_pulsed_adv* jammer module available in OPNET 10.0. It is shown as a red node in Figure 1. The *jam_pulsed_adv* node model represents a pulsed jammer which can be deployed as a fixed, mobile, or satellite node. The jammer provides transmission on a single fixed frequency band which is masked by a periodic pulse train in time. The source creates and transmits packets for the duration of a pulse. It is important to place the jammer and access point at a small

positive altitude since if all the nodes are on the surface of the earth, the curvature of the earth will prevent line of sight communication between them. In our simulation the jammer is placed at an altitude of 10 meters. Pulse width specifies the length of time (in seconds) a pulse is transmitted. Silence width specifies the interval (in seconds) between pulses. Jammer bandwidth specifies the bandwidth (in kHz) of the transmitting channel. Jammer band base frequency specifies the base frequency (in MHz) of the transmitting channel. Finally, jammer transmitter power specifies the transmission power (in Watts) allocated to packets transmitted through the channel. The jammer attributes are shown in Figure 3.

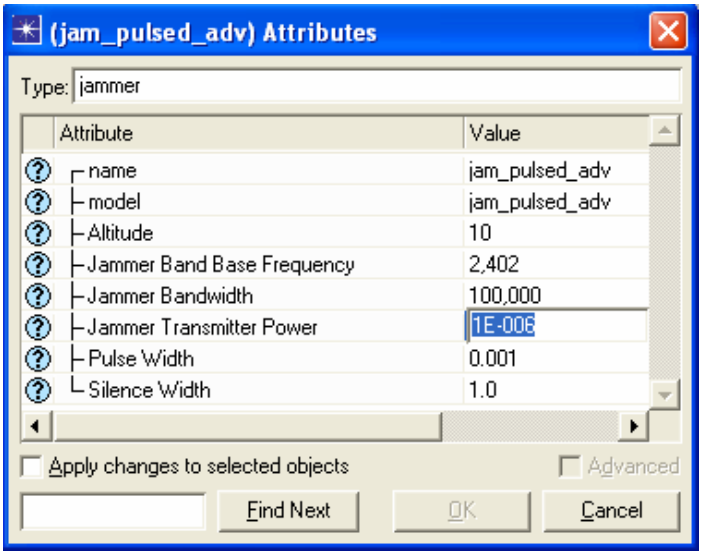

Figure 3: Jammer Attributes

Traffic Model

We model the usual network traffic by profiling email, web browsing, file transfer and telnet traffic. Figure 4 shows some details for these traffic profiles. In this figure, Repeatability attribute is expanded only for web browsing traffic since it is the same for all traffic profiles. The traffic model is essentially lifted from the lab exercises of the Wireless tutorial at OPNETWORKS 03 [1].

The mean data packet size is critical in the effectiveness of jamming. If we consider a Simple Periodic Jammer trying to jam by sending a periodic noise pulse, it is easier to be effective if the mean data packet size is larger. If the mean data packet size increases, then the chance that the data packet transmission and noise pulse intersect in time is higher. If they intersect, the data packet is corrupted and the packet needs to be resent. Similarly, if smaller data packets are sent, it is more likely that some of them will be received. This is just an application of the standard network rule that, as there are more errors on the network, smaller packets are more efficient.

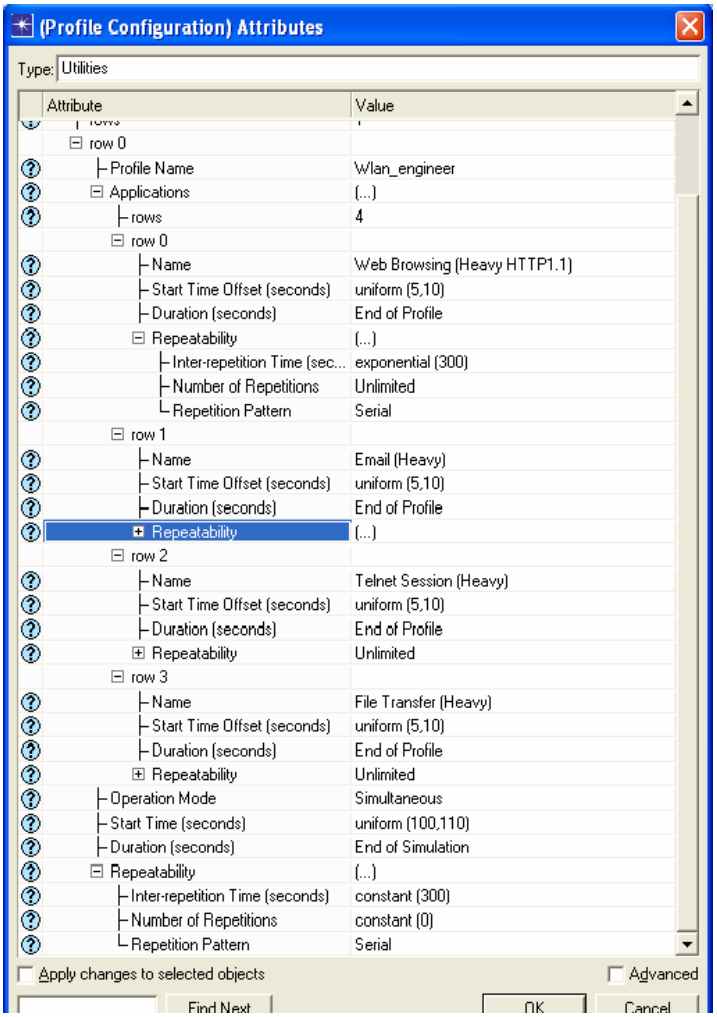

Figure 4: Network Traffic Profiles

In the simulation model considered here, all the nodes are static and always transmit with a power level of .001 Watts. The 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 [1] and the modulation technique is Binary Phase Shift Keying. The relative merit of the jamming techniques may vary somewhat with different modulation techniques but the general ranking will remain the same. 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. In this paper we are mainly interested in showing that Simple Periodic Jamming performs better than Trivial Jamming and that Intelligent Jamming is even more

efficient. In fact, Intelligent Jamming is sufficiently effective to possibly cause security concerns. Our numerical values, presented later, demonstrate this. If attributes change, the numerical values will likely change but we will still be in a position to make a relative comparison compellingly in favor of Intelligent Jamming over Trivial Jamming and strongly in favor of Intelligent Jamming over Simple Periodic Jamming.

Simulation Parameters
We have the following variables for simulation. All the simulations are run for 600 seconds.

Jammer Transmit Power
The workstation nodes in the simulation setup transmit with a power of 1E-003 Watts. Based on this, we define two jammer power levels, high power and low power. 1E-005 Watts and above will be termed as high power and power levels below 1E-005 will be termed low power. 1E-005 is no magic figure. It is just that we recorded contrasting results approximately on either side of 1E-005. Similarly, we say that the network throughput is significantly reduced, or there is a considerable decrease in the throughput if the throughput falls below the 50% mark of the no-jammer, ideal case throughput. Finally, we say that the throughput is destroyed if no data gets transferred whatsoever.

Jammer Pulse Width and Silence Width
The jammer transmits for the pulse width time and does not transmit for the silence width time. These two variables together determine the total time the jammer transmits.

RTS/CTS Mode
This mode can either be on or switched off. We discuss the effects of using the RTS/CTS on the effectiveness of the different types of jamming considered.

Before looking at the different kinds of jamming we propose, we summarize them below. All of them will be defined when they are considered in the different sections.

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

Simple Periodic Jamming with CSMA/CA

The most trivial way of jamming a wireless network is by generating a continuous high power noise. By simulation, we conclude that periodic noise pulses at lower power levels are enough to make a wireless network non-operational. We term this kind of jamming as Simple Periodic Jamming since all we do is to generate a periodic noise irrespective of the packets that are put on the network. We conduct various simulation experiments to determine the power levels and pulse durations that most efficiently disrupt the wireless network throughput. The three jamming parameters that are varied in these experiments are power level, pulse width and silence width. We represent this as a triplet, *(P, p, s)*, where *P* is the jamming power in Watts, *p* is the pulse width in seconds, and *s* the silence width also in seconds. We use *T* to represent the total energy, in Joules, used by the jammer to gauge the effectiveness of a specific kind of jamming. Total power can be calculated as the product of jamming power, pulse width and number of such pulses. Finally we use *t* to represent the total simulation time. Accordingly,

$$T = P.\ p.\ (\ t\ /\ (\ p + s\ )\ )\ Joules$$

Unless otherwise mentioned, we assume that the units are in metric system. The power levels are always in Watts, the time in seconds and the total energy in Joules. Generally p << s, so the p term may be dropped from the denominator. Before conducting the simulations with the jammer, we graph the network throughput against time for the Infrastructure BSS scenario shown in Figure 1 without a jammer. Figure 5 shows that in the absence of jammer, the network throughput stabilizes at about 0.8 Mbps, with a time average of around 0.66 Mbps taken from time = 0. There is no traffic generated in the early part of the simulation run so the network gets stabilized. Consequently, the time average performance is unduly pessimistic.

There is a slight decrease in throughput when the nodes are clustered and when there is no jammer. This may reflect some internodal interference that is not present when the nodes are more dispersed. Since the farthest distance between any two nodes in our simulation scenario is not more than 50 meters, this is justifiable. Since the jammer transmit power is .001 W, all the nodes hear each other. Clustering of nodes should not vary the network throughput much and that is what we observe.

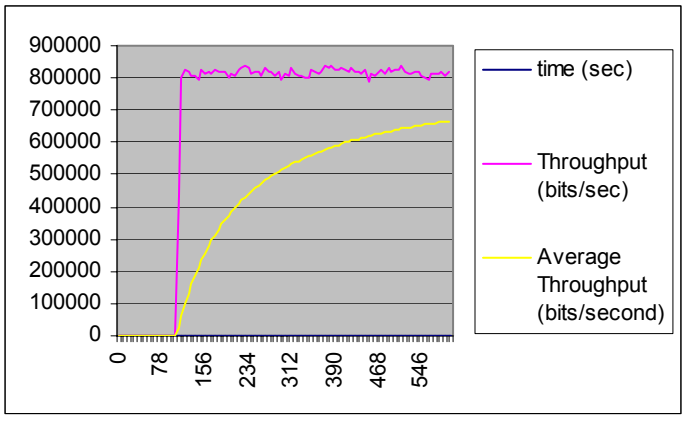

Figure 5: Network Throughput without a Jammer

High Power Continuous Noise Jamming
This is nothing but Trivial Jamming and we present it here just to serve as a benchmark for comparisons. In this least power efficient form of jamming, the jammer is run continuously at the same or ten times the normal transmit power of the other stations (1E-003 W in our case). At this power level, all the packets that are ever put out on the medium are destroyed and there can be no communication whatsoever. We record a zero throughput graph.

Total Energy Estimation
If we assume that the Trivial Jammer is run at 1E-003 W, which is the transmit power for the other stations, and continuously throughout the simulation, the total energy required will be

(1E-003) . 600 = 6E-001 Joules

The purpose of this paper is to show that we can do a lot better. We will present various methods in increasing order of jammer power efficiency.

Low Power Long Duration Noise
We are interested in knowing the lowest continuous power that is required to keep the network down. We start with a power of 1E-007 with jammer operating throughout the length of the simulation. To be precise, we start with (1E-007, 600, 1). The throughput graph we get is similar to that in Figure 5. We repeated the experiment with jammer placed in different positions (sometimes close to the access point). There was no change in the throughput. Experiments were carried out with multiple jammers each with (1E-007, 600, 1); and finally with the nodes clustered all around the jammer. There was no change in the throughput. We conclude that 1E-007 W of power is not sufficient to vary the network throughput in wireless networks under the given simulation settings.

An important point to note here is regarding the jammer altitude. For all our experiments, we fixed the jammer at an altitude of 10 meters from all the other wireless nodes, for the reason we mentioned earlier. If we place the jammer at an altitude of say 2 meters, then a power level of 1E-007 might be sufficient to disrupt the network (according to the power law, the received power at any given point is inversely proportional to the logarithm of the square of its distance from the power source). Since the jammer altitude was fixed at the same altitude of 10 meters for all the experiments, our experimental results are not skewed. However, if we place the jammer closer to the wireless nodes, the energy requirements may be lower than what we estimate here.

For all continuous jammer power levels above 1E-005, no data gets transferred whatsoever. The throughput dies off to zero. This happens since the jammer is now able to corrupt almost all the packets that are thrown into the network or the network appears busy to all transmitting nodes. Since the jammer persistently corrupts the packet, TCP reaches the limit on retransmission attempts and an abort will be issued to the application. After this abort, packets are sent out attempting to reopen the connection with the server. The jammer corrupts even these packets and the server connection is never reopened. In summary, no data gets transferred. Since there is no effect on the throughput with continuous noise with power level 1E-007 and since no data gets transferred when we increase the power level to 1E-005, the least power continuous noise that is required to bring down the throughput significantly (say by more than 50%) lies between 1E-007 and 1E-005. Further experiments confirm that the throughput dies off close to zero near power levels of 1E-005, while at power levels near 1E-007, there is no noticeable decrease in the throughput.

So we assume that the lowest continuous power required to considerably decrease the throughput is in the order of 1E-006 under the given simulation settings. We call this type of periodic jamming as *Continuous Low Power Jamming.*

Total Energy Estimation
The total jammer energy required for the low power continuous noise case is

1E-006 . 600 = 6E-004 Joules

which is three orders of magnitude better than the Trivial Jamming.

High Power Short Duration Noise
We ran several experiments to study the effect of high power noise for a very short duration. The throughput graph was very similar to that of no-jammer case with (1E-006, 1E-007, 5). With (1E-005, 1E-007, 5) we get the graph as shown in Figure 6.

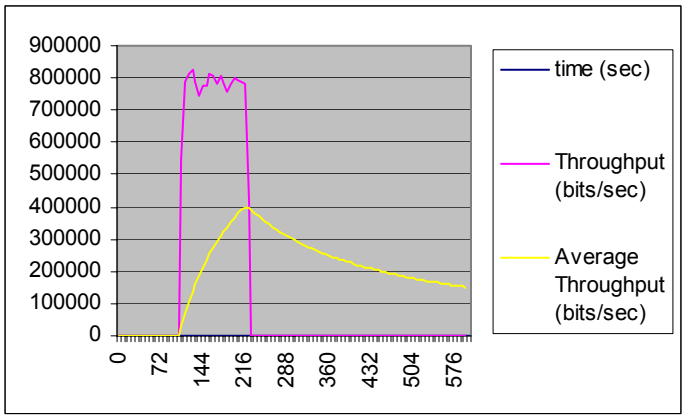

Figure 6: Network Throughput for (1E-005, 1E-007, 5)

The throughput graph has one spike. The spike becomes narrower when the power and the jammer pulse width are increased. We show the results for (1, 1E-005, 5) in Figure 7. For higher power levels, we obtain an initial spike after which the throughput returns to zero. When we inspected the DES log, we found that this was again because of TCP reaching the retransmission limit and an abort being issued to the application.

When we ran several experiments, we found that a random number of spikes show up in arbitrary places in the throughput graph. Since the noise pulses are bursty, which is short duration time, some packets manage to pass uncorrupted at arbitrary times. Hence we observe a spiky throughput graph. Figure 8 shows the spiky throughput graph for the case (4E-006, 1E-007, 5), wherein the high power bursty noise just starts affecting the throughput (the change in the average throughput is not noticeable in the figure, but the throughput graph becomes spiky). On increasing the power level and the pulse duration, the number of spikes decreases and the spikes become narrower.

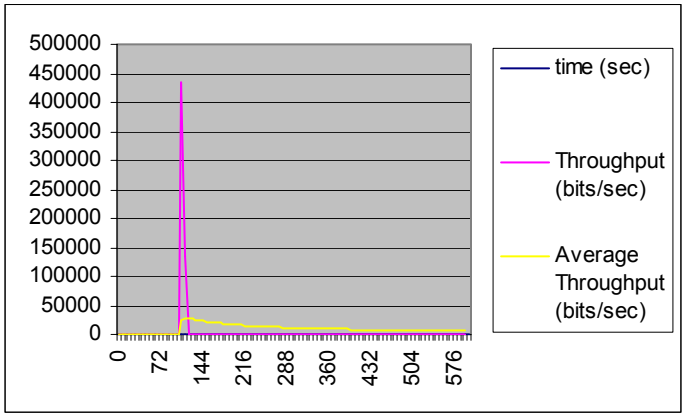

Figure 7: Network Throughput for (1, 1E-005, 5)

On taking the average from several experiments, it appears that the high power level and the bursty pulse width required for a drastic change in the throughput is (1E-002, 1E-004, 5). We term this type of periodic jamming as *Bursty High Power Jamming*. This result is probably particularly sensitive to the modulation techniques used.

Total Energy Estimation

From the previous estimates, the total energy required for the high power short duration noise case is

1E-002 . 1E-004 . (600 / 5) = 1E-004 Joules

The total energy required is comparable to that of Continuous Low Power Jamming. Hence the energy requirement for both Continuous Low Power Jamming and High Power Bursty Jamming is almost a thousand times less than in the Trivial Jamming case.

Keep the Network Busy

From the experiments we conducted for the Continuous Low Power Jammer, we determined that the lowest continuous power that is required to keep the network busy is 1E-006 W under the given simulation settings. In the 802.11b MAC layer, a station would not send the data if it does not see the medium idle for at least DIFS time (See Figure 9 for the two 802.11b modes). So if we create a very short pulse of noise for every interval that is less than DIFS, we should manage to fool the nodes into thinking that the medium is busy. The IEEE 802.11b standard [2] specifies the DIFS time to be 50E-006. We ran experiments with a noise pulse of 1E-006 seconds and at power level 1E-006 for various silence width intervals ranging from 25E-006 to 75E-006 seconds. For all the silence intervals less than 50E-006, very little data got transferred. For time intervals above 50E-006, there was a considerable increase in the network throughput. We call this type of jamming *Busy Jamming*.

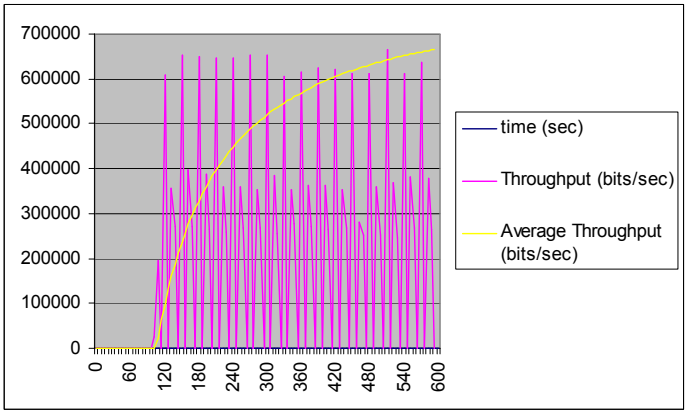

Figure 8: The Spiky Throughput Graph for High Power Bursty Jamming, (4E-006, 1E-007, 5)

Total Energy Estimation

So the amount of jammer energy that is required just to make the network appear busy by short pulses is

1E-006 . 1E-006 . 600 / 50E-006 = 12E-006 Joules.

This is close to 1E-005 and hence approximately ten times more energy efficient than the Continuous Low Power Jamming or High Power Bursty Jamming.

On Using RTS/CTS Mode with Simple Periodic Jamming

We ran experiments for all of the above cases in Simple Periodic Jamming with RTS/CTS mechanism switched on. We noticed that there was a decrease in throughput for all the experiments. This is mainly because if a RTS or a CTS packet is damaged, then no data gets transferred whatsoever. Hence, we record lower network throughput. Simple Periodic Jamming is nothing but generating a pulse train without any knowledge of the protocol, hoping to kill most of the control/data packets. Hence the throughput levels drop drastically. But when periodic noise pulses are generated, there is a chance that a packet might get through if the packet gets transmitted during the jammer silence time. If the packet is smaller, this chance is higher. If RTS/CTS mode is used, then the packets in the network are most likely above the RTS threshold size. Since the packets are larger, it is very unlikely that a packet gets transmitted without corruption.

So a possible countermeasure to Simple Periodic Jamming would be to use smaller packet sizes and employ the basic CSMA/CA mechanism instead of RTS/CTS mechanism.

Intelligent Jamming

There are many control packets that are absolutely necessary for communication in 802.11b networks. Just by corrupting these packets, we can reduce the throughput to zero levels since no data gets transferred if control packets are destroyed. This forms the basis of Intelligent Jamming. In the Simple Periodic Jamming case, the jammer just generated periodic noise hoping to corrupt the control/data packets. In Intelligent Jamming, we put the jammer in promiscuous mode and destroy primarily the control packets. Hence this form of jamming is more power efficient. We provide a brief overview of 802.11b network from [3] before presenting our experimental results.

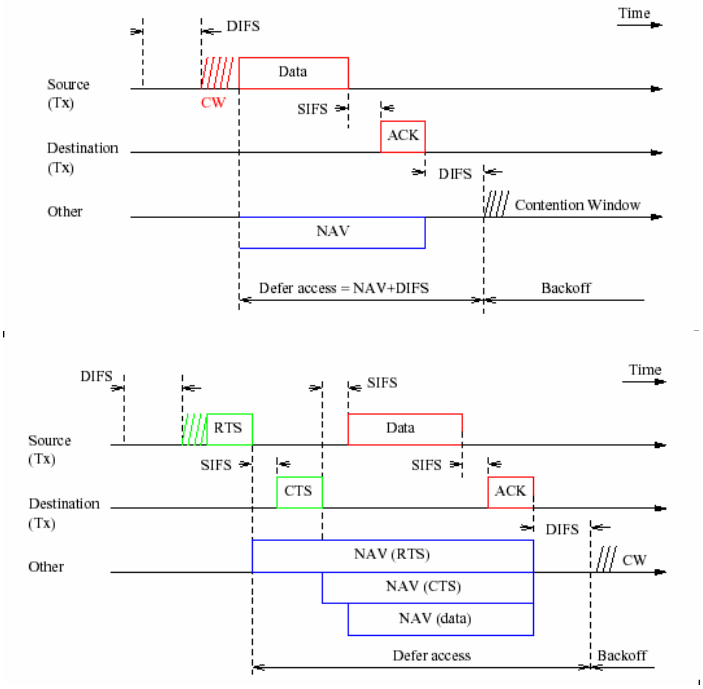

Figure 9: The Basic CSMA/CA and RTS/CTS Mode

Basic CSMA/CA Mechanism

The basic CSMA/CA mechanism is shown in Figure 9. If the medium is sensed idle for at least the duration of DIFS (with the help of 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 back off 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 back off timer from the previous cycles. 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 9 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.

Algorithms for Intelligent Jamming

OPNET 10.0 uses the jam_pulsed_adv jammer. This is a very simple jammer that transmits for the duration specified by the jammer parameters. The Finite State Machine for the jam_pulsed_adv process model is given in Figure 10.

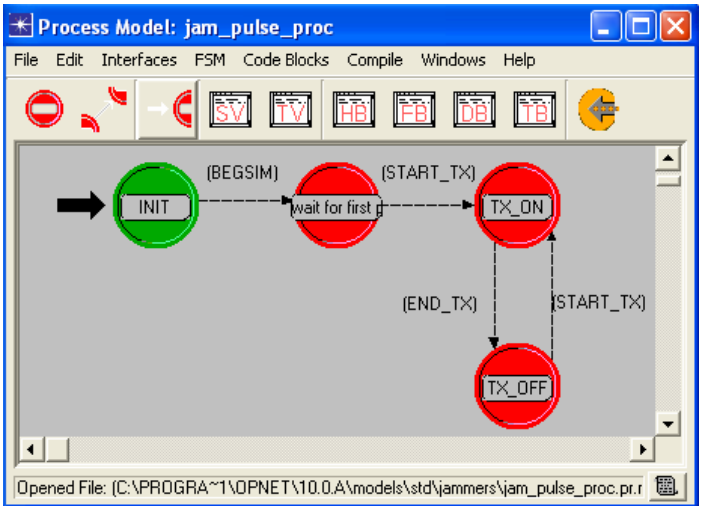

Figure 10: The Finite State Machine for the Basic OPNET Jammer

For these experiments we modify the basic jammer provided in OPNET 10.0, *jam_pulsed_adv* and put the jammer in promiscuous mode. Our jammer listens to the network and has the ability to distinguish between the DATA packets, the RTS and the CTS packets. We present here the results for four types of Intelligent Jammers. For each of the four jammers, we add the intelligence by modifying our promiscuous jammer.

CTS Corruption Jamming

Here the jammer listens or waits for any arbitrary RTS packet promiscuously. For every RTS packet the jammer sees, the

jammer then starts counting down for a period of SIFS from the end of RTS packet and then issues a short jamming pulse which 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 the most power efficient of the methods presented here as the jammer is active for only the short interval of time necessary to disrupt the CTS packet. This type of jamming works only for the RTS/CTS scheme. We changed the jammer module *jam_pulsed_adv* as shown in Figure 11. We have the following states in the CTS Corrupt Jammer process model:

Init State: In this state, the state variables used in the entire process are initialized.

RTS_wait State: . On the start of the simulation, the jammer gets into the RTS_wait state in promiscuous mode. It listens for all the physical layer packets and neglects them as long as they are not RTS packets. On receiving any RTS packet, it waits until it reads the end of that RTS packet and then goes into the SIFS_wait state.

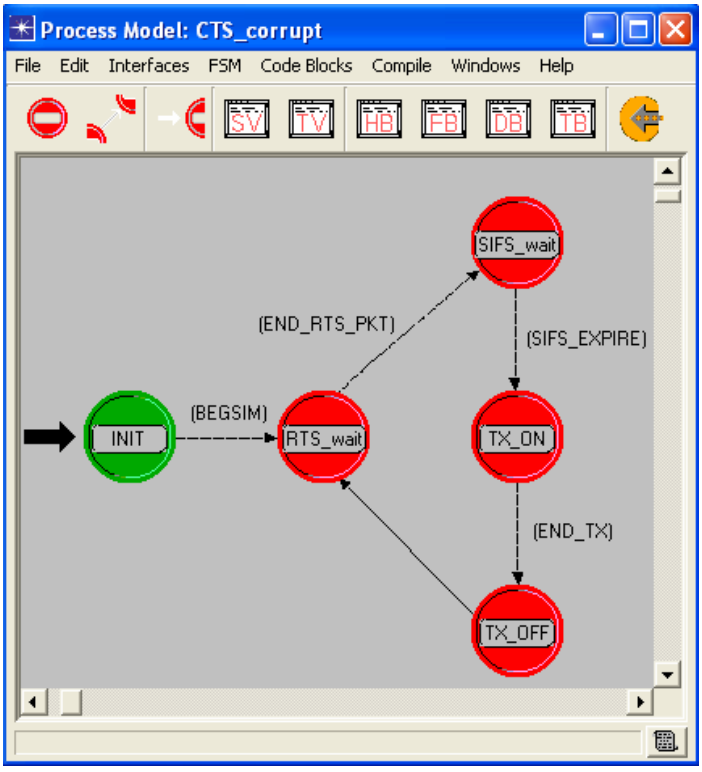

Figure 11: The Finite State Machine for the CTS Corrupt Jammer

SIFS_wait state: We know that, if a node sends out a RTS packet, it most likely gets a CTS packet after a time period of SIFS. In the SIFS_wait state, we wait for a period of SIFS and then get into the TX_ON state.

TX_ON state: We create a short noise pulse (the power and pulse width are configurable) that disrupts the CTS packet. Once this is done, we go into the TX_OFF state.

TX_OFF state: This is just an intermediate state. We immediately go back to the RTS_wait state where we start waiting for RTS packets again.

CTS Corrupt Jamming fails if the RTS/CTS mode is not switched on. For packets that are below the RTS threshold size, RTS/CTS mechanism is not used and these packets can get through.

Total Energy Estimation

For the experiments we conducted, the jammer could see about 100 RTS packets. For each of the RTS packets seen, the jammer corrupted the CTS packet. In the Continuous Low Power Jamming we saw that a jamming power of 1E-005 is enough to kill all the packets. For the high power bursty case, we saw that a power level of 1E-002 was required for a duration of 1E-004. For CTS Corruption Jamming, on average, a jamming pulse of 1E-005 power level is generated for a period of 1E-003 seconds which was enough to reduce the throughput for packets above the RTS threshold size to essentially zero. The total energy required for this kind of jamming is

1E-005 . 1E-003 . 100 = 1E-006 Joules.

For total energy, the CTS Corruption Jamming is seen to be an order of magnitude more effective than the Busy Jamming and about five orders of magnitude times more effective than that of Trivial Jamming.

ACK Corruption Jamming

This is very similar to the CTS corruption jamming. For this type of jamming, the jammer simply waits for a DATA packet. Once it hears the DATA packet on the medium, it starts counting down from the end of the DATA packet for a period of SIFS. At the expiration of SIFS time, it sends a small duration jamming pulse to disrupt the ACK packet that would follow the DATA packet. This is done for every DATA packet that is heard on the network.

Since the ACK does not get back to the sender, the sender keeps retransmitting until a limit on TCP retransmission is reached. After this point, the sender gives up. No further data ever gets transmitted to the receiver. This type of jamming is effective since the jammer is active only for a short interval of time to disrupt the ACK packet.

This type of intelligent jamming works for both of the basic scheme and the RTS/CTS scheme. Sometimes a large amount of data gets discarded if the sender buffer becomes full. Here what happens is that the TCP layer keeps on trying to retransmit the data corresponding to the ACK packet that the jammer destroys each time. The application is unaware of this and thus it sends

more data to the TCP layer to send it out. Eventually the TCP send buffer overflows and TCP stops accepting data from the application layer.

The Finite State Machine for ACK Corruption Jammer is almost the same as that of CTS Corruption Jammer. In fact, the Finite State Machine is similar for all types of Intelligent Jamming methods proposed in this paper. The total energy required for ACK Corruption Jamming was also found to be the same order of CTS Corruption Jamming.

DATA Corruption Jamming
This is very much similar to the ACK corruption jamming. Here the jammer promiscuously waits for the CTS packet. On getting a CTS packet, it waits for SIFS time and issues a short pulse of noise to disrupt the DATA packet.

This type of jamming works only when RTS/CTS mode is switched on. Again the energy requirements were comparable to those of CTS or ACK Corruption Jamming.

DIFS Wait Jamming
In a network with high traffic, it is very likely that if the network is idle for DIFS time, then some transmission takes place after this time. In the basic CSMA/CA case, it could be ordinary data or with the RTS/CTS case, it could be the RTS packet. The jammer can make use of this fact. The jammer monitors the transmission on the medium and if the medium is idle for DIFS time, then the jammer issues a very short pulse just to disrupt the transmission after the DIFS idle time. It could be either the DATA packet or the RTS packet depending on which mode is switched on.

This type of jamming is not as effective as the previous two Intelligent Jamming cases since after the expiration of DIFS time, there might not be a data packet or a RTS packet and it could be that the jammer is trying to jam an idle network. In the previous two cases, an ACK packet would definitely follow a DATA packet and a CTS packet would definitely follow a RTS packet. Then the jamming pulse and hence the jammer power would be effective. Hence DIFS Wait Jamming becomes most effective when the network traffic is high.

Note that if the jammer sends the jamming noise before the expiration of DIFS time, then this type of jamming reduces to the one which we have already considered, i.e., if we generate a short pulse periodically, with a period less than DIFS, then the other nodes are fooled into thinking that the medium is busy all the time. We had termed this type of jamming as Busy Jamming. The energy requirements for DIFS Wait Jamming were comparable to those of Busy Jamming.

Total Energy Comparison
We compare the total energy requirement for the different types of jamming and summarize it in Table 1.

Type of Jamming	Total Energy (Joules)
Trivial Jamming	6E-001
Continuous Low Power Jamming	6E-004
Bursty High Power Jamming	1E-004
Busy Jamming	12E-006
CTS Corruption Jamming	1E-006
ACK Corruption Jamming	Same order as above
DATA Corruption Jamming	Same order as above
DIFS Wait Jamming	Same order as Busy Jamming

Table 1: Total Energy Comparison Table for Different Types of Jamming

Conclusions and Future Work
With the most trivial form of jamming as our benchmark, we showed and simulated different types of jamming with increasing energy efficiency. We started with periodic jamming, which we termed as Simple Periodic Jamming and showed that noise pulses with certain power levels and pulse width can disrupt the network equally well compared to the Trivial Jamming. We then showed that just by sending a short pulse for every time interval that is less than DIFS, we can fool the nodes into thinking that the network is busy. The power requirement for this kind of jamming was reduced by an order of magnitude from Trivial Jamming. We then showed four different types of intelligent jamming that consume very little power. The experiments were based on the observation that if we corrupt the crucial control packets, no data can get through. Specifically, for our simulation model, 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.

The numerical values in the experiments conducted above are specific to the WLAN scenario shown in the simulation model. The main purpose of the paper was to show that we can disable the networks much more efficiently than by using Trivial Jamming and the numerical values are provided to make relative comparisons. Even though the results may vary for different simulation models, relatively speaking, it is our strong conclusion that Intelligent Jamming will be more efficient than Simple Periodic Jamming where no protocol dependent analysis is made. Future work should include generalizing our results as well as considering the effects of PCF modes on the different types of jamming we have considered. We did not consider PCF mode here because the traffic profiles for which PCF mode is typically used are different than the traffic profiles that we have considered in these experiments. PCF mode is generally suitable and switched on for real time traffic. It will also be interesting to see the results for cases where all the nodes including the jamming nodes are mobile.

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. This will be a significant challenge as one author has already done that for a SINCGARS implementation of MIL-STD-188-220C and knows the level of difficulty. Also validation of the model in actual IEEE 802.11b networks should be done.

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.

References

[1] Lab for Session 1332: Planning and Analyzing Wireless LANs and Mobile IP Networks, OPNETWORK 2003.

[2] IEEE Std 802.11b-1999/Cor 1-2001 Standard for wireless LAN medium access control (MAC) and physical layer (PHY) specifications, 2001.

[3] J. Schiller, “Mobile Communications”, Addison-Wesley Longman Publishing, Boston, 1999.

[4] A. Leon-Garcia and I. Widjaja, “Communication Networks,” McGraw Hill, Boston, 2000.