12-15-2006

Protecting Mobile Ad Hoc Networks from Spurious CTS Attacks 'Carrier Sensing based Deferral Mechanism'

Sreekanth Pagadala
University of New Orleans

Protecting Mobile Ad Hoc Networks from Spurious CTS Attacks
‘Carrier Sensing based Deferral Mechanism’

A Thesis

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Master of Science
in
Computer Science

by

Sreekanth Pagadala

University Of New Orleans, 2006

December, 2006

Acknowledgements

I would like to express my appreciation to my advisor Dr. Jing Deng. I want to sincerely thank him for his support, constant availability and pertinent advices. He, the first, introduced me to Ad Hoc wireless Networks and encouraged me in pursuing this thesis research and I am deeply grateful for that.

I would also like to thank the two members of my committee: Dr Shengru Tu and Dr Adlai Depano. I must mention their tremendous kindness, their availability and constant support with their students in general, and myself in particular.

In addition, I would like to thank the rest of the CS faculty members, since I have always been very satisfied with the level of knowledge and pedagogy offered by the teachers of our department. A special thanks to Venkata for his eternal availability and smile.

On a personal note, I would like to thank my dad for always being my model and source of inspiration and my sister and brother-in-law for their unconditional love and support.

I shall cloture these acknowledgements by thanking the most important person in my life: my mother. This work as well as any other success in my life is dedicated to you.

Table of Contents

List of Figures

List of Tables

Abstract

The Request-To-Send (RTS)/Clear-To-Send (CTS) mechanism is used to combat Hidden/Exposed node problems in Mobile Ad Hoc Networks (MANETs). The popular implementation of the RTS/CTS mechanism may lead to virtual blocking of the wireless medium. Problem arises if a malicious node randomly sends out a large number of spurious CTS frames in order to falsely block regular nodes in its neighborhood. Such an attack is termed as spurious CTS attack.

In this thesis, we investigate the effects of such spurious CTS attacks and propose a solution to mitigate the problem. Our proposed solution, Carrier Sensing based Deferral (CSD) mechanism, asks nodes to perform carrier sensing in their deferral state. Therefore, when deferring nodes do not sense any carrier (of data packet transmission) after CTS packets, the CTS packet will be treated as spurious and they will not defer further. Simulation results are presented in the thesis.

1 Introduction

Mobile Ad hoc Networks (MANETs) are autonomous systems of mobile nodes connected by wireless links. Each node operates not only as an end-system, but also as a router to forward packets. The network topology is in general dynamic in nature. In MANETs the nodes operate in peer-to-peer fashion with no centralized base station, which makes the connectivity between the nodes quick and spontaneous. Typical application examples include a disaster recovery or a military operation.

Since in a network the nodes may reside in the transmission range of each other, the so-called hidden/exposed terminal problems may appear. The Request-To-Send (RTS)/Clear-To-Send (CTS) mechanism has been introduced in mobile ad hoc networks to combat hidden/exposed node problems. However, such a technique may suffer from the spurious CTS attack, as we discuss in Chapter 3.

Our project deals with the issue that arises when malicious nodes send out randomly a large number spurious CTS frames in order to falsely block regular nodes in its neighborhood. We term such an attack as *spurious CTS attack*, as these CTS packets are unsolicited. In this thesis, we investigate the effect of such spurious CTS attacks through the means of simulations. We further propose a technique to mitigate the effect of such attacks. Our solution *Carrier Sensing based Deferral (CSD)* mechanism makes use of the random channel assessment technique and resets the NAV value based on the status of the channel to guard MANETs from spurious CTS attacks.

1.1 Background

A wireless ad hoc network is a collection of several nodes/devices which have wireless communication capability. They operate in peer-to-peer fashion without the use of any centralized base station. Each node in a wireless ad hoc network functions as both a host and a router. The network topology is in general dynamic in nature. The connectivity among the nodes varies with time as new nodes join the network and some existing nodes depart from the network.

There are two major types of wireless ad hoc networks: Mobile Ad Hoc Networks (MANET) and wireless sensor networks. While our technique and results may apply to wireless sensor networks, we restrict our discussions to MANETs.

There are a number of characteristics that are unique to wireless ad hoc networks compared to wired networks. These include range limitation of each node, unreliable media on which data transmission occurs, interference from outside nodes, lack of ability for every node to hear every other node within WLAN, and dynamic topologies where nodes move around.

IEEE 802.11 is the standard for wireless LANs [5]. This standard is designed for local area networks and specifies how the connected devices communicate to wireless devices using radio frequency waves which are in close proximity to each other. This standard defines Media Access Control (MAC) and three different Physical (PHY) layers: frequency hopping spread spectrum in 2.4GHz band, direct sequence spread spectrum in 2.4 GHz band, and infrared.

This standard defines the transmissions at 1 or 2 Mbps speed. The IEEE 802.11 standard not only specifies how 802.11 device should operate in peer-to-peer fashion or integrate with an existing wired LAN, but also provides services such as support of asynchronous and time-bounded (time-critical) delivery services as voice or video transmission, accommodation of transmission rates, multicast (including broadcast) services, network management services and registration and authentication services.

MAC layer provides various services which include association, disassociation, authentication, de-authentication, privacy, data transfer, distribution, integration and power management.

The primary service of MAC layer is to provide data transfer/frame exchange between MAC layers. The Carrier Sense Multiple Access (CSMA) scheme may be used by wireless clients. In this mechanism, the client willing to transmit assesses the state of the carrier. If the channel is busy, it defers its transmission. If the channel is free, it transmits the data packet. The receiving client checks the checksum and responds with acknowledgment packet. The receipt of acknowledgment by transmitting client indicates no collision occurred. If transmitting client does not receive the acknowledgment packet, indicates a collision and it either retransmits the data packet till the acknowledgment is received or gives up after certain number of retransmissions.

When the basic CSMA scheme is used in WLANs, a node transmits a data packet if the channel is idle and the receiver node upon receiving the data packet replies back with an acknowledgment packet. This, however, leads to the hidden/exposed terminal problems. Hidden node problem arises when a node cannot hear from a sender that is outside its communication range. Exposed node problem arises when a node is unnecessarily prevented from transmitting due to its neighboring transmitting node. We explain the details of these two problems in the following subsections.

1.2 Hidden Nodes

In a wireless network, every node has a signal receiving range and a carrier sensing range. A node cannot receive any packet from beyond its receiving range and cannot sense any neighboring transmission beyond the carrier sensing range.

In the figure below node A and node C cannot hear each other; we can have a scenario below where nodes are hidden from each other i.e. the nodes exist outside each others range and cause costly data packet collision when both nodes transmit at the same time.

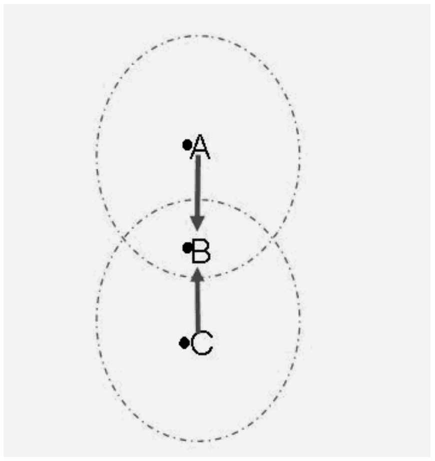

Figure 1: Hidden node problem.

In the figure above, we can see that node B is within the communication range of both nodes A and C but node A is hidden from node C and vice versa. In this case, if both nodes A and C transmit to node B simultaneously, it would cause a data packet collision at node B. Since node A and C can not sense the carrier around the node B, CSMA does not work.

1.3 Exposed Nodes

In the figure below node B and node C are in the range of each other. Node D and node A cannot hear each other; we can have a scenario below where nodes are exposed to each other i.e. node B and node C prevent each from transmitting though no interference would actually occur.

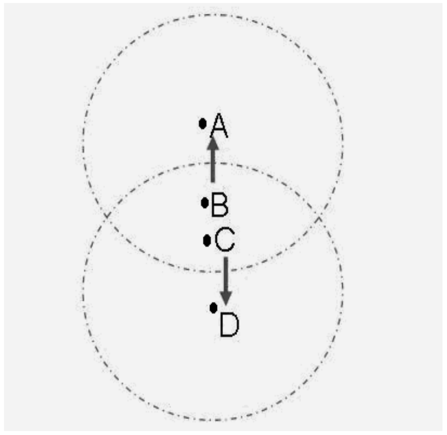

Figure 2: Exposed node problem.

Here, if a transmission between B and A is taking place, node C is prevented from transmitting to D as it concludes that it will interfere with the transmission by its neighbor B.

This problem in wireless networks affects throughput in a significant way, In order to increase the throughput, exposed node C should be allowed to transmit in a controlled fashion without interfering with the on-going transmission between B and A. To avoid this case IEEE 802.11 standard for wireless MAC protocol proposed the RTS- CTS handshake before any transmission.

When a node hears an RTS from a neighboring node, but not the corresponding CTS, that node can deduce that it is an exposed node and is permitted to transmit to other neighboring nodes.

1.4 RTS-CTS Mechanism in IEEE 802.11

To avoid such expensive collisions caused by the hidden/exposed terminal problems, IEEE 802.11 standard use virtual carrier sense mechanism, RTS-CTS handshake. A client willing to transmit will send RTS, which will include source address, destination address and duration for the transmission. The receiver responds with CTS if the carrier is free, with duration information in it. All the clients which hear RTS/CTS set their virtual carrier sense indicator (NAV, Network Allocation Vector) to the value in the duration field of RTS/CTS. The transmitting client then responds with data packets and receiver sends acknowledgment packet. Virtual carrier sense indicator, NAV along with physical carrier sense is used to assess the carrier.

The mechanism is shown in the figure below.

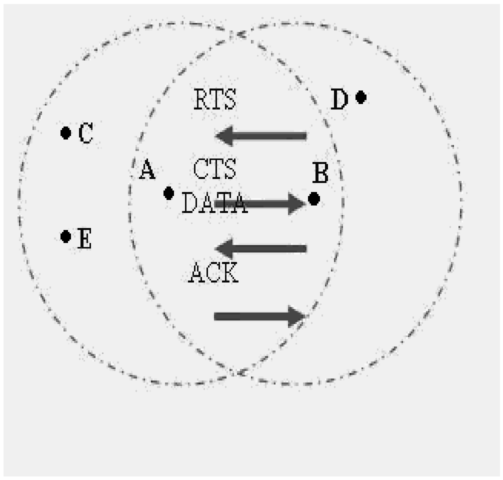

Figure 3: RTS-CTS Mechanism.

When node B wants to transmit a data packet to node A, it first sends a small control packet called Request-To-Send (RTS). Node A upon receiving the RTS replies back with another control packet called Clear-To-Send (CTS). After receiving CTS, Node B transmits the data packet to Node A which replies back with acknowledgment.

The neighboring nodes, in this case, node C, node D and node E hear either RTS or CTS and set their Network Allocation Vector (NAV) to the value in duration field of RTS/CTS frame respectively and defer any of their own transmission till the data transmission is over. If a node does not get a reply for its RTS or data packet, it enters into an exponential back off mode before retransmitting.

In CSMA/CA method, the virtual carrier sensing is performed with the Network Allocation Vector or NAV, which is a count down timer. NAV along with physical carrier sensing indicates when the channel will be available.

This approach, however, suffers the spurious CTS attack problem. Malicious nodes may send out unsolicited CTS packets just to block regular nodes to use the channel. We investigate the effect of spurious CTS attack and propose our solution to mitigate such adverse effects in this thesis.

1.5 Thesis Organization

The remainder of this thesis is structured as follows:

In Chapter 2, we review the related work: The different approaches by which selfish/malicious nodes try to access unfair share of channel, denying the neighboring nodes access to the channel and their proposed solutions in general to counter them.

In Chapter 3, we present the spurious CTS attack problem in IEEE 802.11 networks where malicious nodes try to block neighboring nodes to access the channel. We further propose a technique to combat this problem.

In Chapter 4, performance evaluations are presented. These include throughput performance of regular network, and networks under spurious CTS attacks, and networks implemented with our proposed technique. We further present our simulation results on more realistic networks that model errors of carrier sensing.

In Chapter 5 includes our concluding remarks.

2 Related Work

Most research related to misbehaving nodes in the wireless network addresses *selfish* and *malicious misbehavior*. Selfish misbehavior implies that the selfish nodes misbehave with the intention to improve its own performance in terms of throughput, latency, energy etc. Malicious misbehavior intends to disrupt normal network operation with no performance gain to the misbehaving node. Here we studied different methods by which misbehaving nodes either selfishly or maliciously affect the well-behaved nodes in the network.

Selfish nodes trying to access unfair share of the channel bandwidth by choosing to back off to a minimum value is studied in [4, 10]. In contention systems the nodes competing for access to the channel, back off for random period of time from a specified range before starting a transmission. Selfish node may wait for shorter period of time than well-behaved node and gain unfair advantage. In Research [4, 10] they proposed modifications to the IEEE 802.11 protocol thereby simplifying detection of such selfish nodes. In the research they consider that the receiver is trustworthy. Each trustworthy receiver maintains a counter (which counts down the back off value) and gives a back off value to a sender for its next transmission in its previous CTS or ACK packet. When the next transmission from sender is received, the receiver compares the idle slots it waited to its counter value and penalizes selfish misbehavior node if they do not match. It penalizes the node by adding the same amount of time it made it early, in addition to new back off time. Research [17] also deals with the focus on the prevention and detection of the manipulation of the back off mechanism by selfish nodes in 802.11 networks. They proposed a detection algorithm to ensure honest back off when at least one, either the receiver or the sender is honest.

Research [15] the authors studied the problems in IEEE 802.11 MAC protocol in TCP wireless ad hoc networks. They dealt with neighboring node, one-hop unfairness problem where a single 1-hop transmission and 2-hop transmissions are considered and found that 1-hop transmission was gaining unfair channel access. This is because of the binary exponential back off scheme favoring last winner amongst the contending nodes and also because nodes that are heavily loaded tend to capture the channel continuously thereby transmitting data. This causing lightly loaded neighbor to back off repeatedly.

In Research [7] the authors analyzed the attacks which deny channel access causing congestion in mobile ad hoc networks. They showed that Mac layer fairness is necessary to reduce the various types of DoS attacks. The factors which decide the efficiency of attack are traffic patterns generated by an attacking node, its location in the network, location of other compromised nodes in the network. [6] Also deals with the DoS in the 802.11 Mac protocol. They describe the vulnerabilities in 802.11 network and show ways of exploiting them. Research [16] deals with security problems on the basic mechanisms of the ad hoc networks such as routing protocols and key management mechanism.

In general through spurious RTS/CTS frames and by NAV value tampering malicious nodes try to access unfair share of channel. Spurious RTS/CTS leads to blocking of the nodes in its neighborhood. Blocked nodes propagate false blocking throughout the network, and degrade the network performance. Tampering includes manipulating the duration value in their control packets, thereby setting the NAV high.

Research [13] deals with RTS fake attack where the RTS fake intelligent jammer promiscuously listens to the medium, if it listens to an RTS packet being sent it destroys it by jamming and sends its own RTS and reserves the channel. Jammer upon on receiving the CTS packet does not respond with data. The neighboring nodes of receiver of fake RTS update their NAV and are blocked. The authors proposed CTSR protocol to over come the problem. It is based on assessing the channel status periodically and resetting NAV value if found idle. Our work is similar to the work done by them in terms of malicious behavior, but the malicious attack generated in our work is through spurious CTS packets. Our solution is better than the solution proposed in the paper [13] as we assess the channel randomly instead of periodically. Random channel assessment helps to overcome intelligent spurious CTS attacks, discussed in chapter 3.3.

Research [1] discusses about RTS/CTS attacks and NAV attack. For RTS/CTS attacks they suggested that IEEE 802.11 standard should provide a provision for the nodes to reset the NAV value if there is no transmission within a specified period of time. NAV attacks arise due to the fact that the nodes do not check the validity of correction of duration field in the RTS/CTS frames. Their approach to overcome such an attack is by introducing 2 timers. Every node at least has one timer which checks if the data/ack is received within a specified time as in the timer. If data/ack is not received then the nodes will reset its NAV value. The proposed technique, NAV validation, requires the overhearing nodes to overhear all packets in the RTS/CTS/DATA/ACK sequence. Therefore, it only works in fully-connected networks where all senders can hear each other. In contrast, the scheme that we propose in this work functions well in multi-hop networks.

Research [14] discusses about the various jamming attacks ranging from trivial jamming, periodic jamming and intelligent jamming. The authors showed through their simulations that misbehaving nodes do not follow the MAC layer protocol and the network throughput suffers drastically with two misbehaving nodes than one misbehaving node.

In [18], the authors show that consideration of the physical layer is necessary to determine ad hoc wireless network performance. Their study shows that slight inaccuracy at physical layer can magnify inaccuracy at the higher layer protocols. Their research focuses on the set of factors at the physical layer such as signal reception, path loss, fading, interference and noise computation, and preamble length. These factors are relevant to the performance evaluations of higher layer protocols. Their research includes studying the impact and comparison of the above mentioned factors through two commonly used simulators NS-2 and GloMoSim. The research [18] in a way was helpful for us when we implemented fixed noise in the network for our simulations.

The Paper [3] discusses about the genuine channel blocking problem that arises in the networks. Blocked nodes in the network can propagate false blocking which degrades the network throughput. Their proposed scheme RTS validation includes doing the channel assessment to check the status of medium and reset the virtual carrier sense indicator, NAV, if the channel is idle. Their approach includes the periodic channel assessment.

Our proposed technique *CSD* mechanism counters the spurious CTS attack generated by a malicious node. The malicious node transmits spurious CTS packet periodically to any node in the network thereby blocking its neighboring nodes. Our approach to counter the attack is to assess the state of channel randomly instead of periodically and reset the virtual carrier sense indicator, NAV, if the channel is idle. Our approach is a better solution than the solutions proposed for fake RTS attacks [13] or Channel Blocking Problem [3]. Our scheme does not assess the channel periodically instead assesses the channel randomly. Random channel assessment also helps to avoid intelligent spurious CTS attack, discussed in Chapter 3.3.

3 Spurious CTS Attack and Our Solution

In this chapter, we present the spurious CTS attacks and our solution to the attacks.

3.1 Spurious CTS Attacks

Deferral mechanism of neighboring node when they hear RTS/CTS packets during RTS-CTS handshake may be exploited to generate the spurious CTS attack, thereby blocking the deferred nodes from transmission. The scenario of spurious CTS attack is shown in the figure below.

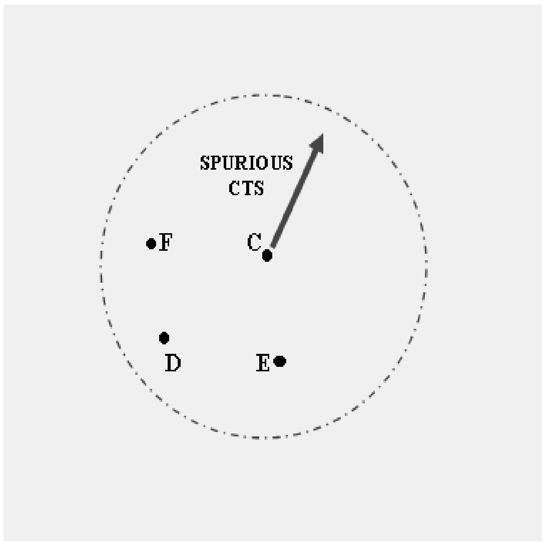

Figure 4: Spurious CTS attack.

Node C transmits spurious CTS packet. No one will respond to the spurious CTS packet because none has sent any RTS packet to node C. Neighboring nodes of C, node D, node E and node F however set their NAV value to the duration value in the CTS frame of node C as they lie within the range of node C. So these neighboring nodes are blocked till the NAV timer counts down to zero.

Figure below shows how the neighboring nodes, node D, node E and Node F are blocked by malicious node C after it sends a spurious CTS packet.

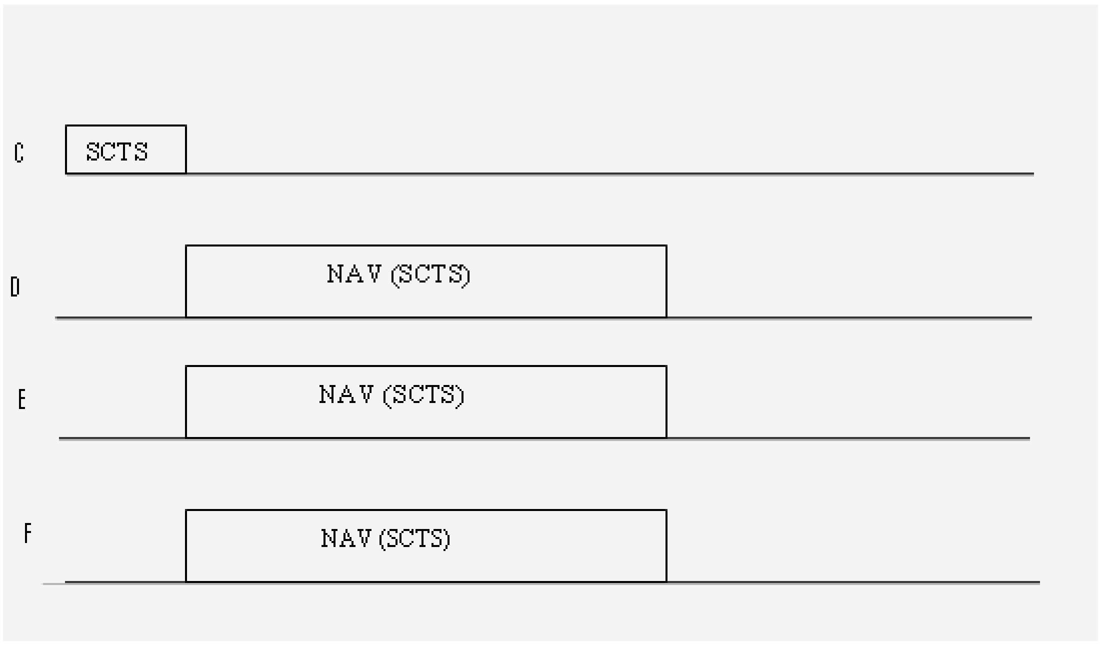

Figure 5: Spurious CTS attack by malicious node C and neighboring nodes D, E, F are blocked for time of NAV (SCTS).

In the figure it shows that neighboring nodes D, E and F are blocked for duration value in spurious CTS packet sent by malicious node C. This duration value gets updated as NAV value for nodes D, E and F.

When a malicious node randomly sends out a large number spurious CTS frames periodically in order to falsely block other well-behaved nodes in its neighborhood, such an attack is termed as *spurious CTS Attack*. If Node C sends spurious CTS frames periodically neighboring nodes, node D, node E and node F are continuously blocked.

3.2 Carrier Sensing based Deferral - Proposed Solution

In IEEE 802.11 network when a channel is reserved by a node for transmission, neighboring nodes cannot access the channel though it is idle, until the reserved time expires. Spurious CTS attack exploits the above vulnerability denying the channel accessibility to neighboring nodes though the channel is idle.

IEEE 802.11 Mac protocol uses both physical carrier sense function and virtual carrier sense function to determine the availability of the channel. The channel is idle when both the functions indicate the channel is idle. Physical carrier sense function uses physical layer to sense the carrier and virtual carrier sense function is based on NAV. NAV is a timer that indicates the amount of time the channel is reserved.

Transmitting node reserves the channel, by setting its NAV value to the value in the duration field of IEEE 802.11 frame. The neighboring nodes update their NAV value to this and count it down till it becomes zero. The channel is found idle if only both physical carrier sense function and virtual carrier sense function indicate so. The channel is found busy if NAV is greater than zero though physically carrier sense function indicates it as idle.

Our approach *CSD* mechanism to mitigate spurious CTS attack is based on physical carrier sense and virtual carrier sense functions. As spurious CTS attack leads to virtual blocking, the physical carrier is idle and so does the physical carrier sense function indicate. But virtual carrier sense function parameter NAV should be updated on the basis of channel assessment.

The neighboring nodes that overhear a CTS packet defer until the corresponding data packet transmission is expected to begin in the channel and then assesses the state of the channel randomly. If the channel is found busy then it continues to defer, otherwise it resumes its transmission schedule.

(a) Specifically when a node receives CTS that is not intended for it, it defers for random time between [CTS_Defer_Time, Total_Deferral_Time]. The CTS_Defer_Time is set as small as possible so that the data packet transmission is expected to begin at the end of this deferral period. After the deferral period the node assesses the state of the channel while continuing deferring. Total_Deferral_Time is the time duration shown in CTS frame.
(b) If the channel is assessed to be busy then node defers for an additional period so that the total deferral time equals to Request_Defer_Time equivalent to time duration shown

in the CTS frame, the duration of deferral requested by the CTS; otherwise it defers no longer.

Instead of all the neighboring nodes accessing the channel at the same time and resetting their respective NAV value, each neighboring node randomly assesses the state of the channel and resets its NAV value. This would counter intelligent spurious CTS attack, discussed in Chapter 3.3.

The figure below shows how CSD works. Neighboring node of malicious node C, D' does not follow CSD mechanism so waits for the total time for which channel is reserved by it, whereas neighboring nodes D, E and F follows CSD mechanism and assess the channel randomly between the time [CTS_Defer_Time, Total_Deferral_Time] and resets its NAV to zero when the channel is found idle.

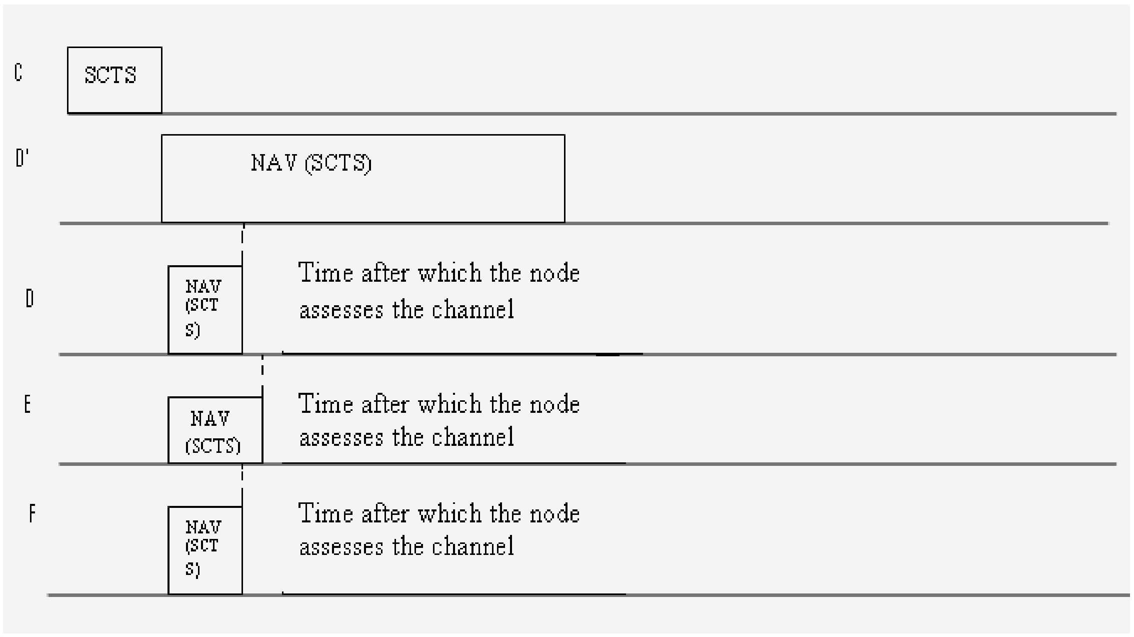
C
SCTS
D'
NAV (SCTS)
D
NAV (SCTS)
Time after which the node assesses the channel
E
NAV (SCTS)
Time after which the node assesses the channel
F
NAV (SCTS)
Time after which the node assesses the channel

Figure 6: Random channel assessment is done by nodes D, E and F, no channel assessment done by node D' when malicious node C generates spurious CTS attack.

CSD mechanism has carrier sensing range over 2.2 times the transmission range. Having carrier sensing range more than twice the receiving range helps to avoid, hidden node problem.

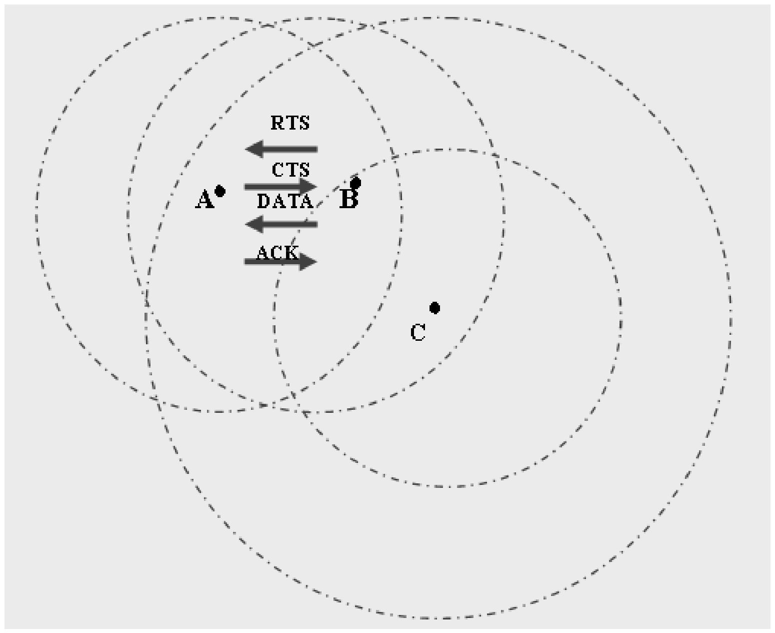

Figure 7: Carrier sensing range 2.2 times the transmission range

In the figure above nodes A and C are hidden from each other, since the carrier sensing range is 2.2 times of transmission range for each node, node C defers it transmission when it hears the start of a legitimate transmission from node A, thereby avoiding collision at node B.

3.3 Intelligent Spurious CTS Attack and Counter Attacks

In spurious CTS attack the malicious node sends the spurious CTS packets periodically to any node in the network. For time being we assume that in CSD mechanism, we assess the state of the channel periodically instead of randomly.

In periodic channel assessment, we assume the channel is assessed every 33 microseconds to check if it is idle; to reset the NAV of the neighboring nodes. But an intelligent malicious node can send spurious CTS frames every 33 microseconds and prove that the channel is busy and continue blocking. Such an attack where a malicious node sends spurious CTS packets at the same time as the channel assessment is termed as *intelligent spurious CTS attack.*

Intelligent spurious CTS attack can be over come by random channel assessment. In random channel assessment each node randomly assesses the state of the channel and reset its NAV accordingly. *CSD* mechanism is based on random channel assessment.

Virtual blocking of the channel can also be done using a spurious RTS frame. Research [13] is related to jamming of the channel using a fake RTS frame. But we implemented the attack using spurious CTS frame as the attack is more effective and realistic. Though each node has carrier sensing range 2.2 times transmitting range, when the spurious CTS packet is sent to non existing node or node outside its range, then malicious node behaves as legitimate hidden node. And the neighboring nodes assume that the malicious node is responding to a legitimate RTS request that is beyond their receiving range.

4 NS Simulations Results

The effect of the spurious CTS attack and CSD mechanism on a wireless network is studied through NS simulations using the simulator's default parameters and remodeling few to more realistic parameters.

NS2 is network simulator developed to simulate the variety of IP networks. It implements network protocols such as UDP and TCP, routing algorithms such as DSR, DSDV, traffic source behavior such as FTP, Telnet, Web, CBR router queue management mechanism such as Drop Tail, RED and CBQ, and more. NS2 have tools for simulation results display, analysis. NS is written in C++ and OTcl (Tcl script language with Object-oriented extensions developed at MIT).

In NS2 the parameters which define the carrier sensing range and receiving range are given as the Carrier Sensing Threshold (CSThresh_) and the Receiving Threshold (RXThresh_) are given, in terms of power level, as CSThresh_ = -78 dBm and RXThresh_ = -64 dBm. The ranges can be calculated in terms of distance using the "Two-Ray Ground Reflection Radio Propagation" model. The carrier sensing range is calculated as 550m which is double that of receiving range of 250 m using this model. Having carrier sensing range more than twice the receiving range helps to avoid, hidden node problem.

The NS default parameters can be found in *".../ns-allinone-2.27/ns-2.27/tcl/lib/ns-default.tcl*".

```
# Unity gain, Omni-directional antennas
# set up the antennas to be centered in the node and 1.5 meters above it
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5
Antenna/OmniAntenna set Gt_ 1.0
Antenna/OmniAntenna set Gr_ 1.0
# Initialize the SharedMedia interface with parameters to make
# it works similar to 914MHz Lucent WaveLAN DSSS radio interface
Phy/WirelessPhy set CPThresh_ 10.0
Phy/WirelessPhy set CSThresh_ 1.559e-11
Phy/WirelessPhy set RXThresh_ 3.652e-10
Phy/WirelessPhy set bandwidth_ 2e6
Phy/WirelessPhy set Pt_ 0.28183815
Phy/WirelessPhy set freq_ 914e+6
Phy/WirelessPhy set L_ 1.0
```

NS has implemented RTS-CTS handshake for all transmissions. However the handshake can be turned on or off in the TCL script. It can be turned off by setting the parameter *MAC_RTSThreshold* to high value, e.g. 3000 in the TCL script, which means that no RTS will be send for any data packet transmission of size below 3000 bytes. Then can simulate packet transmission with lower packet size for having no RTS-CTS handshake.

Mac/802_11 set MAC_RTSThreshold 3000.

By default RTS-CTS handshake is turned on in NS, which we utilize to generate the spurious CTS attack. We introduced 2 new timers

(i) Retransmit timer which generates periodic “spurious CTS attack” and

(ii)Channel assessment timer to implement random channel assessment in “*CSD*” mechanism.

4.1 Assumptions for Simulation

Every node in the network transmits Omni directionally with the same power and receiver sensitivity. Nodes within transmission range of each other communicate without any error; nodes outside transmission range do not interfere. The network is of fixed topology, we used 1000X1000 grid topology**.** The number of nodes in the network is predetermined. We set a flag to decide a well behaving node and a malicious node. We can randomly select any node to misbehave in the network. The nodes are fixed i.e. there is no mobility for the nodes. In addition, we assume that a successful *RTS*/*CTS* exchange guarantees a collision free data transmission. The destination of each packet is one hop away, randomly in any direction. The circular region around each node which defines the communication range, called footprint, is 250m.

4.2 Spurious CTS Attack Simulation

We modified the Mac 802_11 code to simulate a malicious attack; the malicious node promiscuously captures the address of the node from a real frame transmission. Spurious CTS are sent at periodic intervals to the captured address, succeeding blocking the channel. Retransmit timer was introduced to generate the spurious CTS attack periodically. Retransmit timer details can be seen in the Appendix B.

To generate the best spurious CTS attack different values for the parameters CBP and ASI are tested.

CBP = Channel Blocking Period/Duration value for SCTS.

ASI = Average Spurious CTS Packet Interval.

CBP is the duration for which channel is reserved by the node and ASI is the frequency at which spurious CTS packets are generated. The Best attack conditions are determined on the basis of metric "Max Misbehavior Gain" for a given CBP.

PSNA = Number of packets received under no spurious CTS attack.

PSA = Number of packets received under spurious CTS attack.

CBRI =Constant bit rate interval (CBR Interval).

ST = Total simulation time.

CBRI is the frequency at which the data packets are transmitted.

MMG = Maximum misbehavior gain

The results obtained here are the experimental results.

For CBP = 0.065535 seconds the simulations are generated and the results are displayed in the table.

This table is to find Best ASI for CBP = 0.065535.

SNO	CBP	ASI	CBRI	PNSA	PSA	ST
1	0.065535	1.0	0.0035	2828	2808	10
2	0.065535	0.9	0.0035	2828	2811	10
3	0.065535	0.065535	0.0035	2828	222	10
4	0.065535	0.06540	0.0035	2828	203	10
5	0.065535	0.06534	0.0035	2828	258	10
6	0.065535	0.06533	0.0035	2828	424	10
7	**0.065535**	**0.06530**	**0.0035**	**2828**	**166**	**10**

Table 1

In the table CBRI shows that the data packets are transmitted at a frequency of 0.0035 sec. In the normal network scenario channel is reserved based on the duration field of IEEE 802.11 frame. In case of network suffering from spurious CTS attacks, the channel is blocked for a relatively maximum time by setting the duration field to maximum value in spurious CTS packet. In our study we set the CBP to 0.065535 seconds and tried to generate spurious CTS packets at different frequencies to find the best ASI.

The above table shows the non-linear dependency of ASI for a given CBP. As the frequency of the spurious CTS packet transmission increases, the data packets received at the destination node decreases. The ASI value when least number of packets are received at the destination is termed best ASI. In the above table for CBP 0.065535 seconds, best ASI is at 0.06530 seconds.

We tested for different CBP values and tried to find the best ASI for their respective CBP's. We used the metric "Max Misbehavior Gain" to compare the results for different pairs of CBP and best ASI. MMG for a given CBP is given as

Max Misbehavior Gain = {(PSNA – PSA) * Best ASI}/Total_Simulation_Time

The comparison metric MMG helps to decide which pair of CBP and best ASI will lead to best spurious CTS attack.

The table displays the results obtained for the simulations generated for CBP's and their respective best ASI.

SNO	CBP	Best ASI	CBRI	PNSA	PSA	ST	MMG
1	8191	0.00800	0.0035	2828	6	10	2.25
2	16383	0.01613	0.0035	2828	22	10	4.58
3	32767	0.03259	0.0035	2828	45	10	9.99
4	65535	0.065530	0.0035	2828	222	10	17.38

Table 2

From the table we can conclude that as CBP increases, best ASI value increases. The above table shows the linear dependency of best ASI on CBP.

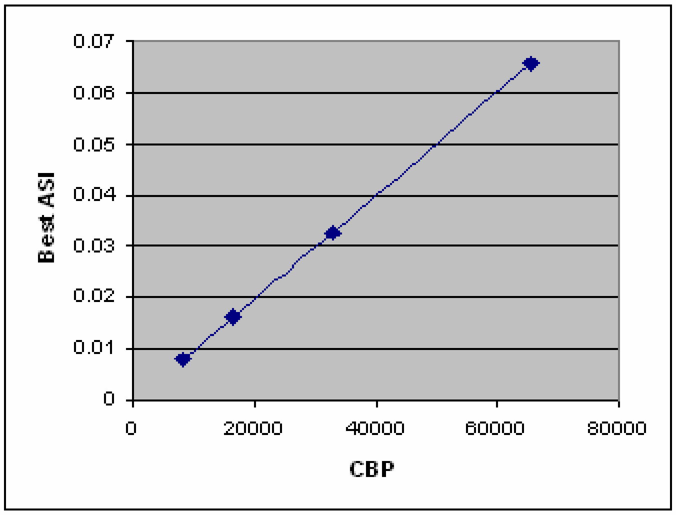

Figure 8: Dependency of Best ASI on CBP.

"Max Misbehavior Gain" from the tables above suggest that CBP = 0.065535 seconds and ASI = 0.06530 seconds form the best pair, to simulate the spurious CTS attack.

The simulations for the spurious CTS attacks are generated using CBP is 0.065535 seconds and ASI is 0.06530 seconds. We modified few other parameters in addition to adding retransmit timer. The network topology for simulations was chosen to mimic many existing 802.11 ad-hoc networks. Simulations are generated in 1-dimensional 1000X1000 and 2-dimensional 1000X1000 topologies varying number of nodes, malicious nodes and the simulation environment.

4.3 Carrier Sensing based Deferral Mechanism Simulation

Our approach to mitigate the effect of spurious CTS attack is based on random channel assessment by the neighboring nodes. In IEEE 802.11 network when a channel is blocked from transmission by a malicious node, neighboring nodes reset its NAV. In our spurious CTS attack scenario, the channel is blocked for the maximum time i.e. 0.065535 seconds. The neighboring nodes of the malicious node update their NAV and count it down till it becomes zero.

Since spurious CTS attack leads to virtual blocking, the physical carrier sense function detects it to be idle, but the virtual carrier sense function indicator, NAV shows the channel busy. We introduced a new timer channel assessment timer to assess the state of the channel, whether channel is busy or idle. We used the same timer to find CTS_Defer_Time, the minimum time after which data packet transmission is expected to begin. Experimentally CTS_Defer_Time was found 33 microseconds.

Instead of all the neighboring nodes of malicious node assessing the state of channel at same time i.e. 33 microseconds, neighboring nodes in our approach assess the channel randomly in the range of [CTS_Defer_Time, CBP]. Here CBP is equivalent to total deferral time, the time for which malicious node reserved the channel. CBP in our simulations is 0.065535 seconds.

Channel assessment timer checks the channel and updates NAV accordingly. If the channel is found idle, neighboring nodes defer no longer and reset their NAV value to zero; otherwise nodes continue deferral for the total duration time. We modified NS-2, MAC802_11 code to add these conditions and to assess the state of the channel randomly in the time range of [0.000033, 0.065535] seconds. Details of channel assessment can be seen in the Appendix C.

We generated the simulations for 1-dimensional 1000X1000 and 2-dimensional 1000X1000 topologies under ideal and real conditions. Our simulations results are presented as graphs and are comparative study of the simulations generated in 3 different networks.
(I) Normal network
(II) Network suffering from spurious CTS attacks
(III) CSD network

We later introduced some fixed noise in the network to study the effect noise on the network. We introduced fixed negative noise in the network. We also introduced error in carrier sensing in the network to make it look more realistic. Error was introduced through missed detection and false alarm artificially. Missed detection is the case where the node fails to detect the packet transmission on the channel though there is some transmission. And false alarm is the case where the nodes assess the channel and assume that there is some transmission on the channel though there is no transmission. We analyzed this scenario in our CSD based network and presented the results.

Below are the simulation results generated under these conditions.

1) **Simulation results for 1-dimensional network with fixed positioned nodes**: A network of 1000 X 1000 grid topology consisting fixed number of static nodes. The transmission range of each node is fixed at 250m.

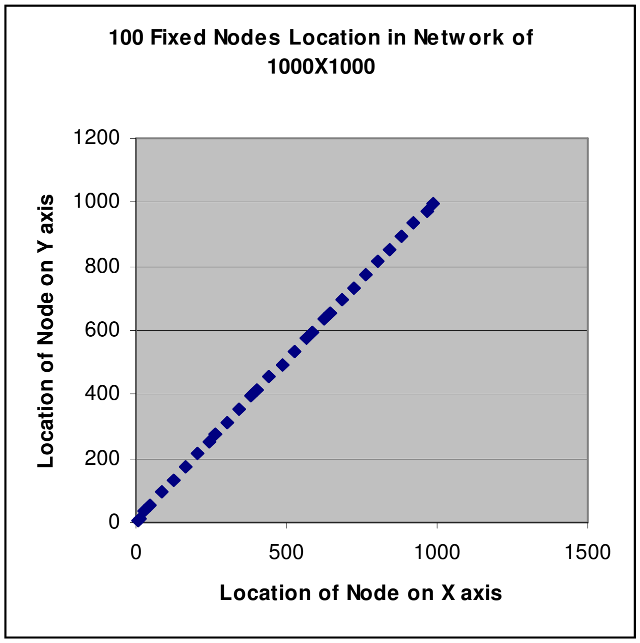

Figure 9: 100 fixed position nodes in a 1000X1000 1- dimensional network.

The Figure 9 displays the placement of nodes in the network. The nodes are placed along a straight line in a 1-dimensional network, each node separated from its neighboring node by approx (10, 10, 0). There is a *single node* generating CBR traffic, of 500 byte size packet and are transmitted at a 2-Mbps rate to its neighboring node every 0.0035 seconds. The number of nodes, in the network is increased gradually starting initially with 3 nodes then to 10, 20, 40, 60 and 100. Each simulation is run for a time period of 10 seconds. There is only *one malicious node* in the network and is within the range of the traffic generating node. This basic simulation is generated to test the working of spurious CTS attack and the CSD mechanism.

The throughput results obtained below are by varying number of nodes in the network.

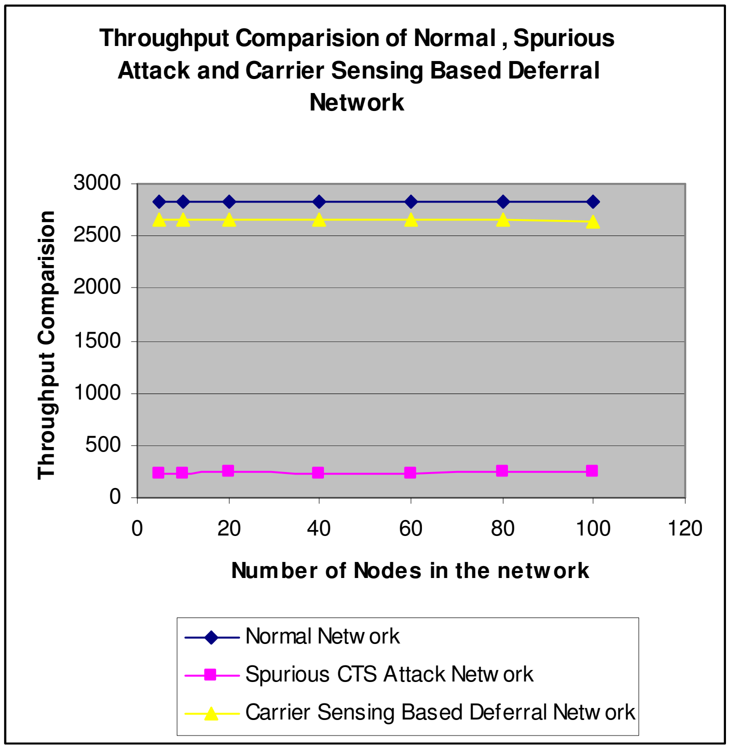

Figure 10: Throughput comparison of 1-dimensional fixed positioned nodes of 3 different networks varying number of nodes in network.

The throughput obtained in case of normal network when there is no spurious CTS attack or implementation of CSD is around 2827 data packets at the destination node/data sink. The throughput obtained after generating spurious CTS attack is around 200 data packets at the destination node. Since there is only one CBR traffic generating node the malicious node which lies in the range of the traffic generator promiscuously captures the destination address and sends out spurious CTS packets, so it could almost block the channel denying the access to the traffic generating node, thus bringing down the throughput to 10% of normal network. With the introduction of our technique, the neighboring nodes of the malicious node .i.e. traffic generating node assesses the channel, if found idle it resets NAV to zero and starts transmitting the traffic. In CSD network the throughput is around 2650 data packets at the destination node which is approximately 93% of the normal network throughput.

2) Simulation results for 2 dimensional network with fixed positioned nodes: A network of 1000 X 1000 grid topology consisting fixed number of static nodes. The transmission range of each node is fixed at 250m.

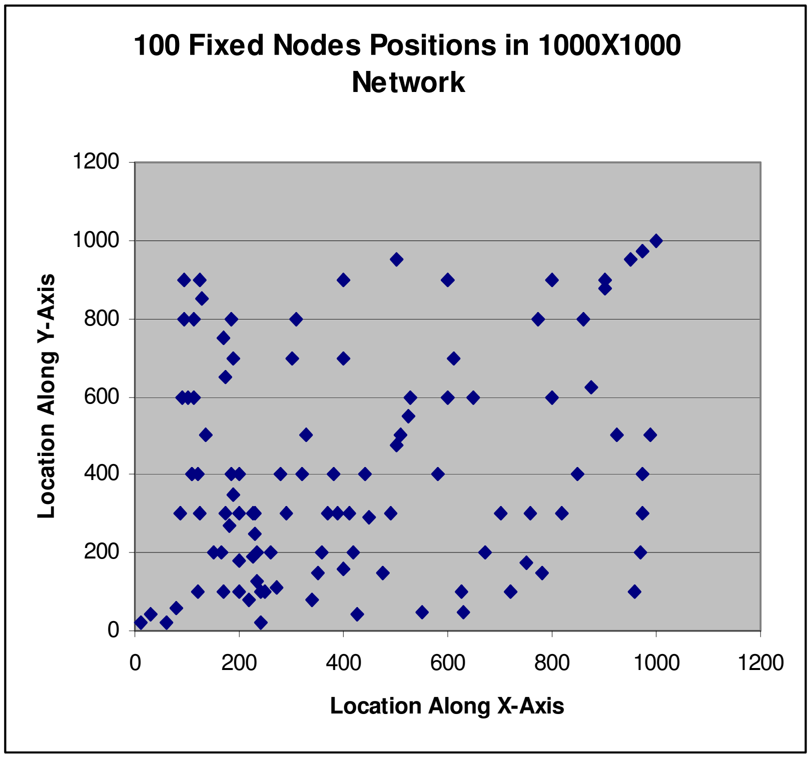

Figure 11: 100 fixed position nodes in 1000X1000 2- dimensional network.

The Figure 11 displays the placement of nodes in the network. The nodes are placed in a 2-dimensional network. There are 5 nodes generating CBR traffic, of packet size 500 byte transmitted at a 2-Mbps rate, independently generated at each node every 0.0035 seconds. The number of malicious nodes in the network is increased. Each simulation is run for a time period of 12 seconds.

The throughput results obtained below are by varying number of malicious nodes in the network.

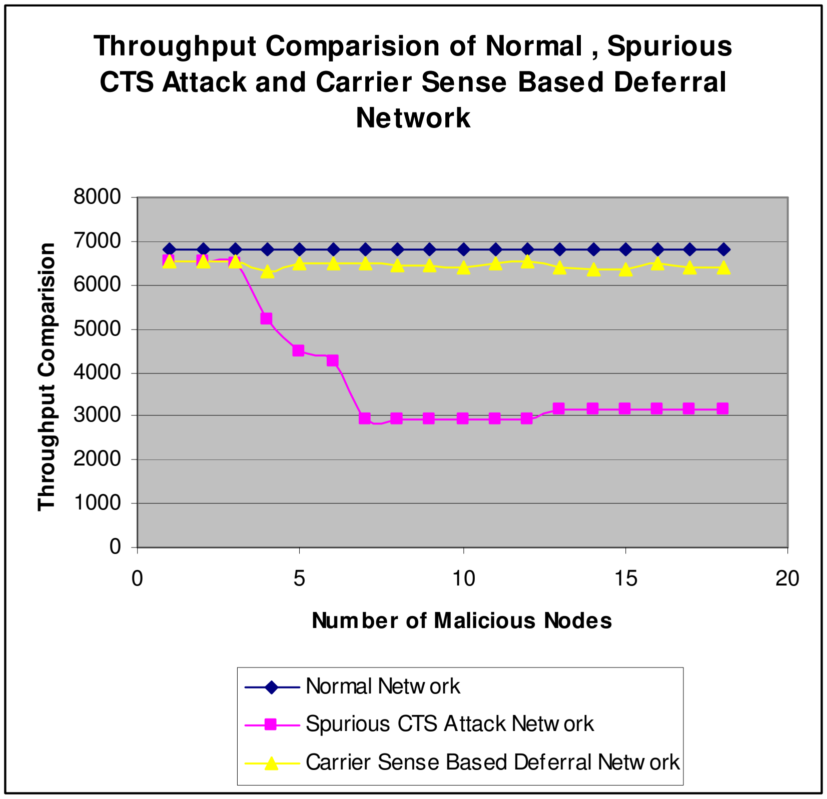

Figure 12: Throughput comparison of 100 fixed position nodes in 3 different networks varying number of malicious nodes.

The throughput in case of normal network is around 6824 data packets at the data sink, since there are 5 CBR traffic generators the throughput is relatively higher than the previous setup. Introducing spurious CTS attack in the network, initially the attack is not effective for few malicious nodes say 3 or 4 and this is because there are 5 traffic generators in the network. As the number of malicious nodes is increased in the network, the attack is effective. Spurious CTS attack was able to block the traffic generators when the malicious nodes lie within the range of traffic generator and capture the address and generate the spurious CTS packets. The attack remains same after a certain number of malicious nodes, because whether there is one malicious node or say 5 malicious nodes within the range of traffic generator, the impact is same as only one of them will generate spurious CTS packets and others have to defer their transmission. The spurious CTS attack could bring down the throughput to 3100 data packets at the data sink which is almost 45% of normal network throughput. In CSD introduced network the neighboring nodes of malicious nodes, i.e. traffic generators assess the channel randomly. When ever the 5 traffic generators assess the channel to be idle, it resets its NAV and starts transmission. CSD mechanism could bring the back the throughput to approximately 6400 data packets, which is 93 % of normal network throughput.

3) Simulation results for 2 dimensional network with randomly placed nodes: A network of 1000X1000 grid topology, consisting of 100 nodes, each node placed randomly .The transmission range of each node is 250m.There are 5 nodes generating CBR traffic, 500 byte size packets are transmitted at a 2-Mbps rate independently generated at each node every 0.0035 seconds. The number of malicious nodes in the network is increased. Each simulation is run for a time period of 10 seconds and an average of 20 simulations is taken.

The throughput results obtained below are by varying number of malicious nodes in the network.

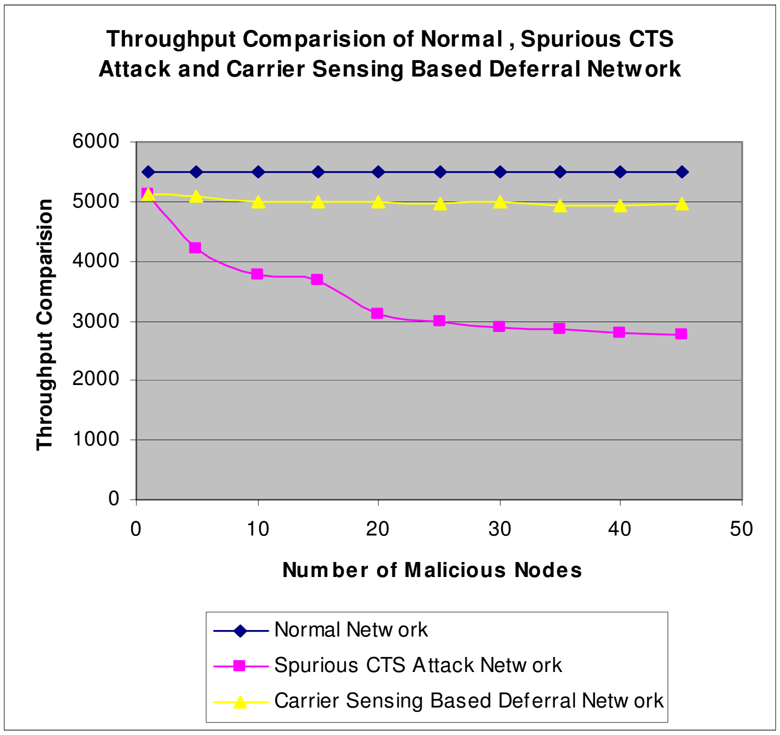

Figure 13: Throughput comparison of 100 randomly placed nodes in 3 different networks varying number of malicious nodes.

The throughput in case of normal network is around 5511 data packets at data sink which is generated by 5 CBR traffic generators. The nodes in the network get randomly placed for every simulation generated. In case of spurious CTS attack, the malicious nodes change their position for every simulation generated. We generated the simulations for each set of malicious nodes for 20 times and average of them is taken. The throughput obtained due to spurious CTS attack gradually reduces to 2800 data packets which are approximately 50% of normal network throughput. As number of malicious nodes is increased in the network, these malicious nodes lie within the range of all 5 traffic generating nodes and promiscuously capture the destination address and start transmitting the spurious CTS packets. These spurious CTS packets reserve the channel for long period of time thereby blocking the traffic generating nodes from transmission. After certain number of malicious nodes, say 30 malicious nodes the throughput remains same, as the effect of one or more than one malicious node in the range of traffic generator is same, because one produces spurious CTS packets and others defer their transmission. The introduction of the CSD enables the neighboring nodes of malicious nodes to access the channel randomly, especially the traffic generating nodes to reset its NAV if channel is blocked falsely. This brings back the throughput to approximately 5000 data packets which are 90% of normal network throughput.

4) Simulation results for 2 dimensional network with randomly placed nodes and noise introduced in the network: A network of 1000X1000 grid topology, consisting of 100 nodes, each node placed randomly .The transmission range of each node is 250m. There are 5 nodes generating CBR traffic, 500 byte size packets are transmitted at a 2-Mbps rate independently generated at each node every 0.0035 seconds. The number of malicious nodes in the network is increased. Each simulation is run for a time period of 10 seconds. The simulations are generated with and without fixed noise introduced in the network. We introduced a negative fixed noise in the network called Noise Factor equal to -2E (8).

The throughput results obtained below are by varying number of malicious nodes in the network. The graph shows the comparison between noise introduced and no noise introduced network. The dotted lines in the graph show the network with noise introduced and the bold lines show the network with no noise.

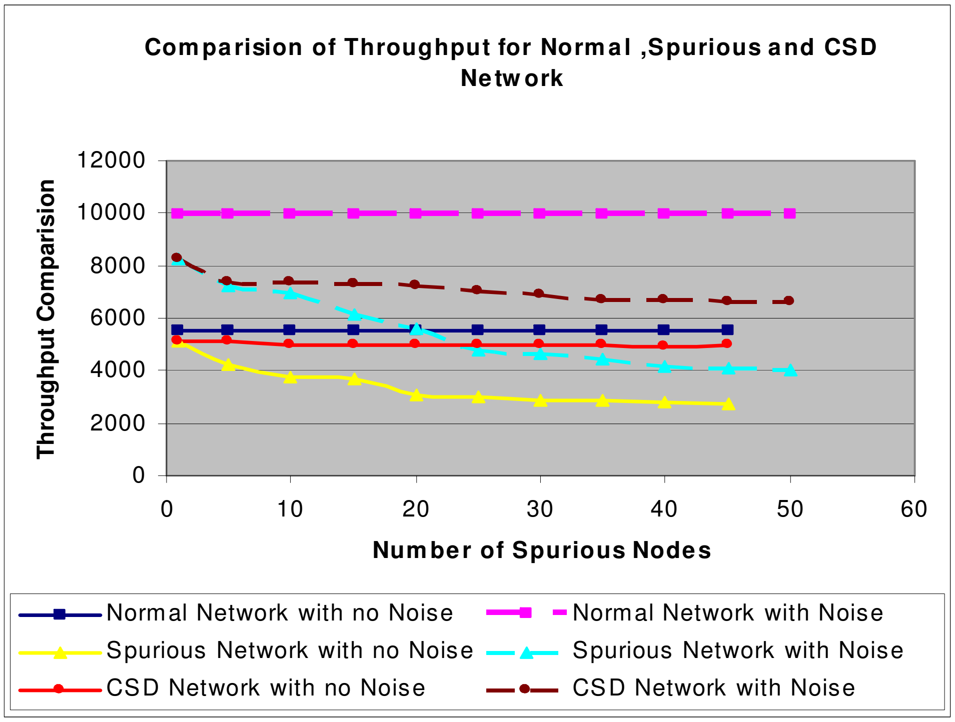

Figure 14: Throughput comparison of 100 randomly placed nodes in 3 different networks.

Without noise: The throughput in case of normal network is around 5511 data packets received and the throughput obtained due to spurious CTS attack gradually reduces to 2800 data packets which is 50% of the normal network throughput. With the introduction of CSD, brings back the throughput to approximately 5000 data packets which are 90% of normal network throughput.

With noise: After introducing fixed negative noise, the signal to interference ratio decreases, so the throughput has comparatively increased in all the 3 networks. Signal to interference ratio is defined as the signal strength the node receives from the transmitter to the signal strengths of other transmitting neighboring nodes. When the negative noise factor is introduced, the signal to interference ratio at the receiver decreased. The throughput in case of noise introduced normal network is around 9970 data packets and the throughput obtained due to spurious CTS attack reduces to 4950 data packets which is nearly 50% of the normal network throughput. The malicious nodes lying within the range of the traffic generating nodes promiscuously capture the destination address and generate the spurious CTS packets to that node, thereby blocking the neighboring nodes.

After introducing CSD, the throughput is approximately 7500 data packets which are 75% of normal network throughput. In CSD network, the traffic generating nodes which are the neighboring of the malicious nodes assess the channel and reset their NAV if they are blocked.

5) Simulation results for 2 dimensional network with randomly placed nodes and error introduced in carrier sensing: A network of 1000X1000 grid topology, consisting of 100 nodes. And each node placed randomly in the network .The transmission range of each node is 250m. There are 5 nodes generating CBR traffic, 500 byte size packets are transmitted at a 2-Mbps rate independently generated at each node every 0.0035 seconds. The number of malicious nodes, in the network is increased. Each simulation is run for a time period of 10 seconds and an average of 20 simulations is taken.

Assumption for the Simulation: To introduce error in carrier sensing we introduced missed detection threshold (T_md) and the false alarm threshold (T_fa) (generally represented in terms of power in units of dB). Simulations are generated with probability of false alarm (p_fa) and probability of missed detection (p_md) in the range of (0, 1) and varying T_fa, T_md.

a) 2-D network with false alarms and missed detections: The simulations are generated with p_md, p_fa in the range (0, 1) and T_fa = 0.1, T_md = 0.1. The throughput results obtained below are by varying number of malicious nodes in the network.

The graph shows the comparison of CSD under ideal conditions and the CSD with error introduced in carrier sensing.

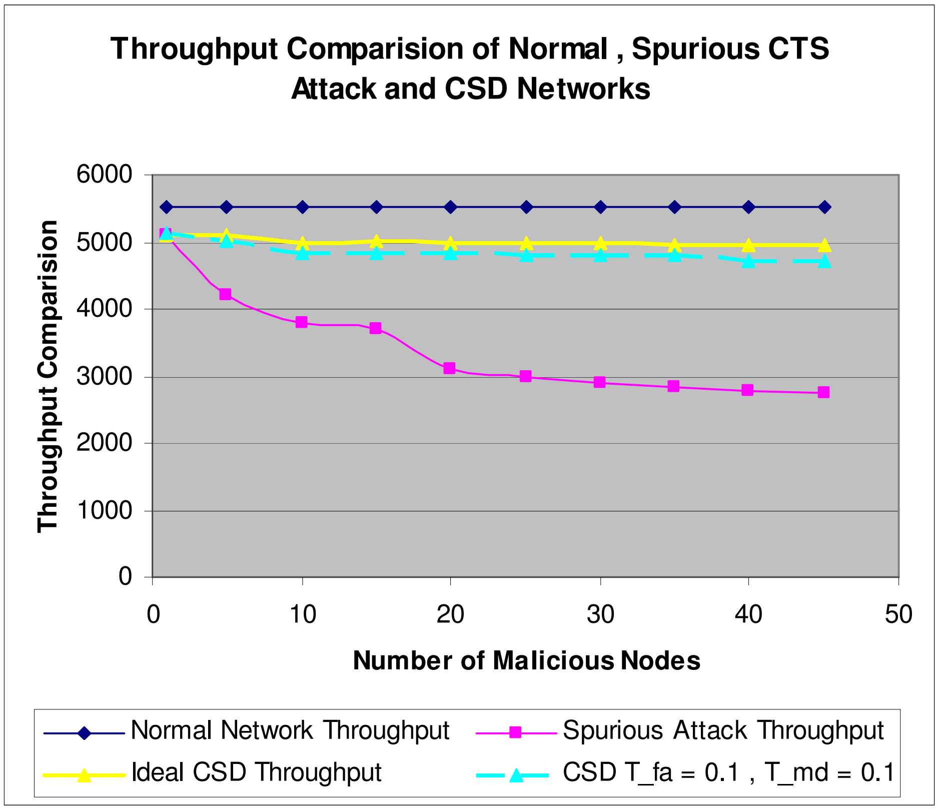

Figure 15: Throughput comparison of 100 randomly placed nodes in different networks

When missed detection and false alarm are introduced in the network, the curve for CSD with T_fa = 0.1, T_md = 0.1 is slightly lower than the curve of CSD with no error introduced. The number of false positive is slightly greater than the number of missed detections after introducing error making the CSD throughput with error less than CSD without error.

Missed detection refers to situation in the network where a node fails to detect the transmission on the channel though there is some real transmission. And false alarm refers to the situation in the network, where the node detects some transmission on the channel though there is no real transmission. The normal network throughput is 5511 data packets and network suffering from spurious CTS attacks throughput is 2800 data packets which are 50% of normal network throughput and the CSD with no error is 5000 data packets which are 90% of normal network throughput and CSD with error introduced 4880 data packets which is 88 % of normal network throughput.

When T_fa and T_md = 0.1 then the number of false positives are slightly greater than missed detections making CSD curve with error in carrier sensing lower than CSD with no error.

(b)CSD networks with different false alarm and missed detection thresholds: The graph shows the comparison between various curves for different values of T_fa, T_md after introducing error in carrier sensing.

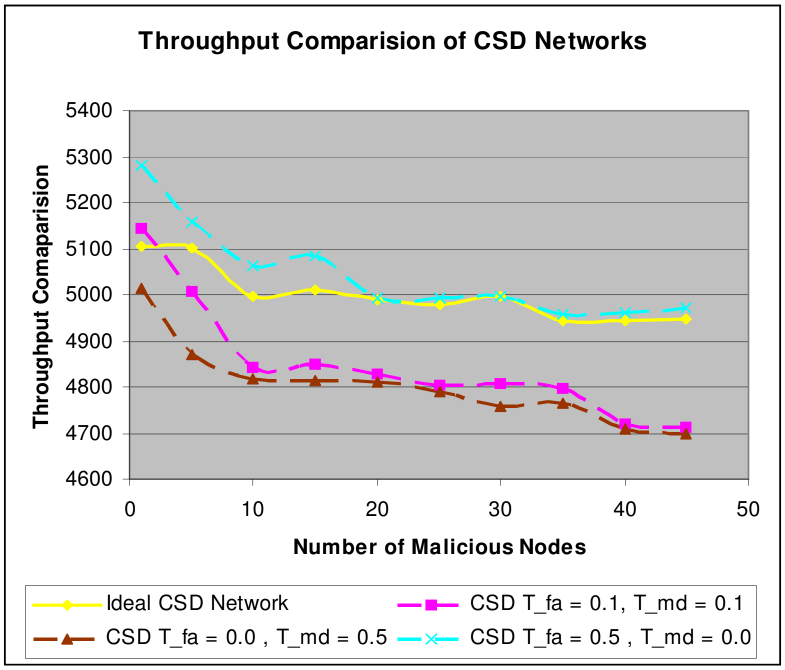

Figure 16: Throughput comparison of 100 randomly placed nodes in CSD network introducing error in carrier sensing varying thresholds.

The bold curve in the graph shows the CSD network performance with no error introduced in carrier sensing. The dotted lines in the graph represent the throughput performance of CSD networks with error introduced. When the false positive occurs in the network, it reduces the throughput; this is because though there is no transmission on the channel, due to a false positive the physical carrier sense function responds as the channel as busy. The network performance with T_fa = 0.0 and T_md = 0.5 is the lowest curve. This is because when carrier sensing is done by node to detect the channel, it has more false positives compared to missed detections on the network. And the curve with T_fa = 0.5 and T_md = 0.0 is the curve higher than the ideal CSD network curve. This is because when the nodes detect the channel it has least false positive in the network, making its throughput slightly higher than ideal CSD network throughput. When T_fa = 0.1 and T_md = 0.1 the throughput is slightly higher than the throughput of CSD network with T_fa = 0.0 and T_md = 0.5 due to lesser false positives, but lower than the throughput of ideal CSD network due to more false positives.

The proposed solution CSD mechanism was able to counter the spurious CTS attacks under different network conditions. The throughput obtained after implementing the solution varied depending upon the simulation setup chosen. On the whole the proposed technique could obtain 88 to 93% of normal network throughput in our simulations.

5 Conclusions

The RTS/CTS mechanism is used to avoid collisions in ad-hoc networks due to hidden nodes. In the current implementation of RTS/CTS transmissions, any node that receives a RTS or a CTS packet is forbidden to transmit. This becomes a vulnerability when malicious nodes may send out spurious CTS packets to block neighboring nodes. Our investigations have shown that such a spurious CTS attack may lower the throughput to 40-50% of the normal value.

In this thesis, we have proposed a technique termed the Carrier Sensing based Deferral (CSD) to combat such an attack. We have implemented the CSD scheme and studied its performance in various networks. It is shown that the CSD scheme protects the network from spurious CTS attacks. For example, in 1000X1000 grid 1-dimensional network, the CSD scheme retains 93% throughput of normal network. The throughput of network suffering from spurious CTS attacks was 10% of normal network throughput. In 2-dimensional network with 1000X1000 topology and nodes randomly placed, the throughput with CSD is 90% of normal network throughout compared to network suffering from spurious CTS attack which is 50% of normal network throughput.

We have also investigated the performance for CSD networks with potential false alarms and missed detections of carrier sensing. Our results show that the performance of our CSD scheme remains rather constant even if some false alarms and missed detections take place.

Our simulations show that our proposed solution was able to combat the spurious CTS attack. By means of simulation, we have shown that the use of *CSD* mechanism improves the network performance.

References

1. D. Chen, J. Deng , and P. K. Varshney, "Protecting Wireless Networks against a Denial of Service Attack Based on Virtual Jamming," *ACM MobiCom '03,* Poster, San Diego, CA, USA, September 14-19, 2003.

2. Pradeep Kyasanur and Nitin Vaidya, "Detection and Handling of MAC Layer Misbehavior in Wireless Networks ", Dependable Computing and Communications Symposium (DCC) at the International Conference on Dependable Systems and Networks (DSN), June 2003.

3. Saikat Ray, Jeffrey B. Carruthers and David Starobinski, "RTS/CTS-Induced Congestion in Ad Hoc Wireless LANs," WCNC 2003: Wireless Communications and Networking Conference: New Orleans March 16 - 20, 2003.

4. Pradeep Kyasanur and Nitin Vaidya "Selfish MAC Layer Misbehavior in Wireless Networks " , accepted for publication in IEEE Transactions on Mobile Computing (April 2004).

5. IEEE Computer Society, "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," IEEE standard 802.11, 1999.

6. John Bellardo and Stefan Savage, "802.11 Denial-of-Service Attacks: Real Vulnerabilities and Practical Solutions", In Proceedings of the USENIX Security Symposium, Washington D.C., August 2003.

7. Gupta, V., Krishnamurthy, S.V., and Faloutsos, M., "Denial of Service Attacks at the MAC Layer in Wireless Ad Hoc Networks", Milcom 2002, Anaheim.

8. S.-R.Ye, Y.-C.Wang and Y.-C.Tseng, "A Jamming-Based MAC Protocol to Improve the Performance of Wireless Multihop Ad Hoc Networks", Wireless Communications and Mobile Computing, Vol. 4, No. 1, Feb. 2004, pp. 75-84 (SCIE).

9. J Parker, A Patwardhan, A Joshi, "Detecting wireless misbehavior through cross-layer analysis", Proceedings, IEEE Consumer Communications and Networking Conference Special Sessions, January 2006.

10. Pradeep Kyasanur and Nitin Vaidya "Detection and Handling of MAC Layer Misbehavior in Wireless Networks", Dependable Computing and Communications Symposium (DCC) at the International Conference on Dependable Systems and Networks (DSN), June 2003.

11. Mithun Acharya, David Thuente "Intelligent Jamming Attacks, Counterattacks and (Counter) ^2 attacks in 802.11b Wireless Networks", OPNETWORK-2005 Conference, Washington DC, August 2005.

12. S. Radosavac, J.S. Baras and I. Koutsopoulos, "A framework for MAC layer misbehavior detection in wireless networks", Proceedings of ACM Workshop on Wireless Security (WiSe) 2005, Cologne, Germany.

13. R. Negi and A. Rajeswaran, "DoS analysis of reservation based MAC protocols ", IEEE International Conference on Communications, 2005.

14.David Thuente and Mithun Acharya, "Intelligent Jamming in Wireless Networks with Applications to 802.11b and other Networks" , to appear in Proceedings of the 25th IEEE Communication Society Military Communications Conference (MILCOM 2006).

15. S.Xu and T.Saadawi," Does the IEEE 802.11 MAC protocol work well in multihop wireless adhoc networks", IEEE Communications Magazine, Volume. 39, Issue 6, Jun 2001.

16. Jean-Pierre Hubaux, Levente Buttyán and Srdan Capkun, “The quest for security in mobile ad hoc networks", Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing, Long Beach, CA, USA.

17. Alvaro Cardenas, Svetlana Radosavac and John S. Baras, *"*Detection and Prevention of MAC Layer Misbehavior for Ad Hoc Networks*",* 2004 ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN 2004), pages 17-22, Washington DC, USA.

18. M. Takai, J. Martin, and R. Bagrodia, "Effects of Wireless Physical Layer Modeling in Mobile Ad Hoc Networks", In Proceedings of MobiHoc 2001, pages 87-94, October 2001.

19. M.Cagalj, S.Ganeriwal, I. Aad, J. P.hubaux, "On cheating in CSMA/CA ad-hoc networks,” IEEE INFOCOM 2005.

20. The *ns* Manual -- collaboration between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC. Editor Kevin Fall, Editor Kannan Varadhan.

21. Computer Networks (Fourth Edition), Author Andrew S.Tanenbaum, Publisher Prentice Hall PTR, ISBN 0130661023.

Appendices

Appendix A

NS2, Simulation Model

Ad Hoc wireless scenario is generated taking into consideration various factors. The topology of the simulation is the boundary where the nodes, node_(0), node (1) ....nose_(n) are placed. The connection either UDP/TCP is established for the packets to be transmitted between the nodes as they are within the range of each other else the packet gets dropped.

Wireless simulations are generated using a tcl script. A mobile node consists of network components, Link Layer (LL), Routing Protocol (RP), Interface Queue (IFQ), MAC layer, Network Interface Type (NETIF).At the beginning of the simulation these components have to be defined. In addition other components are antenna model, number of mobile nodes, dimensions of the topology, simulation etc has to be defined.

```
#=====================================================================
# Define options
#=====================================================================
set val(chan) Channel/WirelessChannel     ;# channel type
set val(prop) Propagation/TwoRayGround    ;# radio-propagation model
set val(netif) Phy/WirelessPhy            ;# network interface type
set val(mac)   Mac/802_11                 ;# MAC type
set val(ifq)   CMUPriQueue                ;# interface queue type
set val(ll)    LL                         ;# link layer type
set val(ant)   Antenna/OmniAntenna        ;# antenna model
set val(x)     1000                       ;# X dimension of topology
set val(y)     1000                       ;# Y dimension of topology
set val(cp)    ""                         ;# node movement model file
set val(sc)    ""                         ;# traffic model file
set val(ifqlen) 50                        ;# max packet in ifq
set val(nn)     100                       ;# number of mobilenodes
set val(seed)   0.1
set val(stop)   12.0                      ;# simulation time
set val(tr)     exp.tr                    ;# trace file name
set val(rp)     DSR                       ;# routing protocol
set AgentTrace  ON
set RouterTrace OFF
set MacTrace     ON
#=====================================================================
```

In this example we have the number of nodes as 100, but for large networks we can set it to higher numbers.

```
set val(nn) 500 ;# number of nodes
```

After declaring these parameters,
(i) The instance of the Simulator,
set ns_ [new Simulator]

(ii) Trace files
set tracefd [open $val (tr) w]
$ns_ trace-all $tracefd

(iii) The topology is defined.

```
# set up topography object
set topo [new Topography]
# topo scale 1000m X 1000m
$topo load_flatgrid $val(x) $val(y)
```

The nodes within the 1000x1000m boundary are configured according to the parameters above with only MacTrace ON. The configuration API is given as:

```
#  configure nodes
#-----------------------------------------------
$ns_ node-config -adhocRouting $val(rp) \
                 -llType $val(ll) \
                 -macType $val(mac) \
                 -ifqType $val(ifq) \
                 -ifqLen $val(ifqlen) \
                 -antType $val(ant) \
                 -propType $val(prop) \
                 -phyType $val(netif) \
                 -channelType $val(chan) \
                 -topoInstance $topo \
                 -agentTrace OFF \
                 -routerTrace OFF \
                 -macTrace ON \
                 -movementTrace OFF
```

In our simulations, the nodes exhibit no mobility so that we turn off any random motion or any kind of other motion. The nodes are given a random initial position and they maintain that throughout the simulation.

For the Four-node scenario the nodes are set as

```
for {set i 0} {$i < $val(nn) } {incr i} {
        set node_($i) [$ns_ node]
        $node_($i) random-motion 0          ;# disable random motion
       set range 1000
       set X($i) [expr {(int(rand()*$range))*1.0}]
       set Y($i) [expr {(int(rand()*$range))*1.0}]
 }
```

```
# Provide Random (X,Y, for now Z=0) co-ordinates for nodes
#-------------------------------------------------------------
$node_(0) set X_ $X(0)  ;# node0
$node_(0) set Y_ $Y(0)
$node_(0) set Z_ 0.0
```

```
$node_(1) set X_ $X(1) ;# node1
$node_(1) set Y_ $Y(1)
$node_(1) set Z_ 0.0

$node_(2) set X_ $X(2) ;# node2
$node_(2) set Y_ $Y(2)
$node_(2) set Z_ 0.0

$node_(3) set X_ $X(3) ;# node3
$node_(3) set Y_ $Y(3)
$node_(3) set Z_ 0.0

$node_(4) set X_ $X(4) ;# node4
$node_(4) set Y_ $Y(4)
$node_(4) set Z_ 0.0
```

Now the traffic flows between the nodes are set with sources as UDP Agents with CBR traffic. One such traffic flow between node0 and node1 looks as shown below.

```
# Setup traffic flow between nodes 0 connecting to 1 at time 2.0
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(1) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 500
$cbr_(0) set interval_ 0.0035
$cbr_(0) set random_ 0
$cbr_(0) set maxpkts_ 100000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 2.0 "$cbr_(0) start"
```

At the end, the stop time is defined when the nodes reset themselves. The procedure stop{} is called to flush out traces and close the trace file. And finally the simulation is run.

```
$ns_ at $val(stop)  "stop"
proc stop {} {
global ns_ tracefd
$ns_ flush-trace
close $tracefd
exit 0
}
puts "Starting Simulation..."
$ns_ run
```

Appendix B

/NS2.27/ns-allinone-2.27/ns-2.27/mac/ mac-timers.cc

```
/*
==============================================================
  Retransmit Timer

==============================================================*/
void
RetransmitTimer::start(double time)
{
      Scheduler &s = Scheduler::instance();
      assert(busy_ == 0);
       busy_ = 1;
       paused_ = 0;
       stime = s.clock();
       rtime = time;
#ifdef USE_SLOT_TIME
       ROUND_TIME();
#endif
       assert(rtime >= 0.0);
       s.schedule(this, &intr, rtime);
}


void
RetransmitTimer::handle(Event *)
{
       busy_ = 0;
       paused_ = 0;
       stime = 0.0;
       rtime = 0.0;
       mac->RetransmitHandler();
}
```

This piece of code below is to identify the nodes which behave maliciously in the network and those malicious nodes calling the Retransmit Timer.

Ns2.27/ns-allinone-2.27/ns-2.27/mac/mac-802_11.cc

```
void
Mac802_11::recv_timer()
 {  ......
    ......
    .....
 /*
        * Address Filtering
        */
       if(dst != (u_int32_t)index_ && dst != MAC_BROADCAST) {
               /*
               *  We don't want to log this event, so we just free
               *  the packet instead of calling the drop routine.
               */

     if((malicious_[nodeid]==1)&&(spurious_cts_scheduled==0))  {
                                    spurious_cts_dst=dst;
                           spurious_cts_scheduled=1;
                                 srand((unsigned int)time((time_t *)NULL));

   for(i=0;i<100;i++)  {
randnumretransmit =(int)(80.0*rand()/(RAND_MAX+1.0));
                     randnumretransmit++;
          retransmittime=0.06530 + randnumretransmit*0.0000025;

               if(retransmittime < lowerlimitretransmit)
                       retransmittime = lowerlimitretransmit;
                 if(retransmittime > upperlimitretransmit)
                         retransmittime=upperlimitretransmit;
                               }
                 mhRetransmit_.start(retransmittime);

 }

              discard(pktRx_, "---");
              goto done;
       }
................
.........
...............
}
```

This piece of code is to generate the Spurious CTS packets periodically.

Ns2.27/ns-allinone-2.27/ns-2.27/mac/mac-802_11.cc

```
void
Mac802_11::RetransmitHandler()
{

 send_spuriousCTS(spurious_cts_dst,65535);
   if (mhDefer_.busy()) mhDefer_.stop();tx_resume();
 if(mhRetransmit_.busy()) mhRetransmit_.stop();

          mhRetransmit_.start(0.06530);
}
```

Appendix C

/NS2.27/ns-allinone-2.27/ns-2.27/mac/ mac-timers.cc

```
/*=================================================================
   Channel Assessment Timer

=================================================================*/

void
ChannelAssessmentTimer::start(double time)
{
        Scheduler &s = Scheduler::instance();
        assert(busy_ == 0);
        busy_ = 1;
        paused_ = 0;
        stime = s.clock();
        rtime = time;
#ifdef USE_SLOT_TIME
        ROUND_TIME();
#endif
        assert(rtime >= 0.0);
       s.schedule(this, &intr, rtime);
}


void
ChannelAssessmentTimer::handle(Event *)
{
        busy_ = 0;
        paused_ = 0;
        stime = 0.0;
        rtime = 0.0;
        mac->ChannelAssessmentHandler();
}
```

This piece of code is to identify the well-behaving nodes in the network and invoke the Channel Assessment timer.

Ns2.27/ns-allinone-2.27/ns-2.27/mac/mac-802_11.cc

```
void
Mac802_11::recv_timer()
{   ......
   ......
   ......
   if(dst != (u_int32_t)index_) {

         srand((unsigned)time(NULL));
              for(i=0;i<100; i++) {
                     randnum = 0.5 * upperlimit * rand() /(RAND_MAX + 100.0);
                          if( randnum  < lowerlimit ) {
                                  randnum = lowerlimit;
                                                    }
                                 }
            if((malicious_[nodeid]==0)&&(spurious_cts_scheduled==0))
                          {
                   if(mhChannelAssessment_.busy()) mhChannelAssessment_.stop();
                     mhChannelAssessment_.start(randnum);
                          }
            set_nav(mh->dh_duration);
        }
    ........
    ........
    ........
}
```

Ns2.27/ns-allinone-2.27/ns-2.27/mac/mac-802_11.cc

This piece of code is to do the Carrier sensing and identify whether channel is idle and there is no false alarm of carrier sense, NAV should be reset. Similarly, if the channel isn't idle but there is a missed detection of carrier sense, NAV should be reset.

```
void
Mac802_11::ChannelAssessmentHandler()
  {
 srand((unsigned)time(NULL));

for(i=0;i<100; i++) {
                 p_fa = rand() /(RAND_MAX + 1.0);
               }

for(i=0;i<100; i++) {
                p_md = rand() /(RAND_MAX + 1.0);
                  }

         if((is_idlephy()&&(p_fa > Thresh_fa))||(!is_idlephy()&& (p_md < Thresh_md)) )
                          {
tx_state_ = MAC_IDLE ;
nav_ = 0.0;
set_nav(0.0);
if(mhDefer_.busy()) mhDefer_.stop();
}

 }
```

This piece of code is to introduce noise in the network.

/Ns2.27/ns-allinone-2.27/ns-2.27/mac/ wireless-phy.cc

```
int
WirelessPhy::sendUp(Packet *p)
{ ....
 .....
 ....

 for(i=0;i<100; i++) {
                        x1 = rand() /(RAND_MAX + 1000.0);
                  if( x1  < lowerlimit1 ) {
x1 = lowerlimit1;                                                                    }
                                  }
                                          }
           for(i=0;i<100; i++) {
                        x2 = rand() /(RAND_MAX + 1000.0);
                  if( x2  < lowerlimit1 ) {
                               x2 = lowerlimit1;
                                                 }
                            }

                 noisefactor  = sqrt(-2*log(x1)/0.434294482)*cos(2*3.14159*x2)*(0.10);

              if ((Pr + noisefactor  )  < CSThresh_) {
pkt_recvd = 0;
                   goto DONE;
                    }

......
.....
....

}
```

Vita

Sreekanth Pagadala was born in Hyderabad, India in 1980. He grew up and studied in Hyderabad until passing his Baccalaureate (B-Tech) in June 2002. After graduating from JNT University, Sreekanth has been accepted in the Master program in Computer Science of UNO in May 2003. He defended his thesis in September 2006 and decided to apply for the Master program in Math at University of New Orleans.