Detection and Handling of MAC Layer Misbehavior in Wireless Networks*

Pradeep Kyasanur
Dept. of Computer Science, and
Coordinated Science Laboratory,
University of Illinois at Urbana-Champaign
email: kyasanur@uiuc.edu

Nitin H. Vaidya
Dept. of Electrical and Computer Eng., and
Coordinated Science Laboratory,
University of Illinois at Urbana-Champaign
email: nhv@uiuc.edu

Abstract

Wireless Medium Access Control (MAC) protocols such as IEEE 802.11 use distributed contention resolution mechanisms for sharing the wireless channel. In this environment, selfish hosts that fail to adhere to the MAC protocol may obtain an unfair share of the channel bandwidth. For example, IEEE 802.11 requires nodes competing for access to the channel to wait for a "backoff" interval, randomly selected from a specified range, before initiating a transmission. Selfish nodes may wait for smaller backoff intervals than well-behaved nodes, thereby obtaining an unfair advantage. We present modifications to the IEEE 802.11 protocol to simplify detection of such selfish hosts. We also present a correction scheme for penalizing selfish misbehavior. Simulation results indicate that our detection and correction schemes are successful in handling MAC layer misbehavior.

1 Introduction

Wireless Medium Access Control (MAC) protocols such as IEEE 802.11 [1] use distributed contention resolution mechanisms for sharing the wireless channel. The contention resolution is typically based on cooperative mechanisms (e.g., random backoff before transmission) that ensure a reasonably fair share of the channel for all the participating nodes. In this environment, some *selfish* hosts in the network may misbehave by failing to adhere to the network protocols, with the intent of obtaining an unfair share of the channel. The presence of *selfish* nodes that deviate from the contention resolution protocol can reduce the throughput share received by conforming nodes. Thus, development of mechanisms for detecting and handling *selfish* misbehavior is essential.

*This research was supported in part by UIUC Campus Research Board.

IEEE 802.11 MAC protocol [1] has two mechanisms for contention resolution; a centralized mechanism called PCF (Point Coordination Function), and a fully distributed mechanism called DCF (Distributed Coordination Function). PCF needs a centralized controller (such as a base station) and can only be used in infrastructure-based networks (PCF is also an optional feature in IEEE 802.11). DCF is widely used in both infrastructure-based wireless networks as well as ad hoc wireless networks. In this paper, we identify a misbehavior possible in the DCF mode.

DCF uses CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) for resolving contention among multiple nodes accessing the channel. A node (sender) with data to transmit on the channel selects a random backoff value from range $[0, CW]$, where CW (*Contention Window*) is a variable maintained by each node. While the channel is idle, the backoff counter is decremented by one after every time slot (time slot is a fixed interval of time defined in IEEE 802.11 standard), and the counter is frozen when the channel becomes busy. The node may access the channel when the backoff counter is decremented to zero.

After the backoff counter is decremented to zero, the sender may reserve the channel for the duration of the data transfer by exchanging control packets on the channel. The sender first sends a RTS (Request to Send) packet to the receiver node. The receiver responds with a CTS (Clear to Send) packet and this exchange reserves the channel for the duration of data transmission (RTS-CTS exchange is optional in IEEE 802.11). Both the RTS and the CTS contain the proposed duration of data transmission. Other nodes which overhear either the RTS or the CTS (or both) are required to defer transmissions on the channel for the duration specified in RTS/CTS. After a successful RTS/CTS exchange, the sender transmits a DATA packet. The receiver responds with an ACK packet to acknowledge a successful reception of the DATA packet. If a node's data transmission is successful, the node resets its CW to a minimum value (CW_{min}); otherwise, if a node's data transmission is unsuccessful (detected by the absence of a CTS or the ab-

sence of an ACK), CW is doubled, subject to a maximum of CW_{max}.

A misbehaving node may obtain more than its fair share of the bandwidth by

- Selecting backoff values from a different distribution with smaller average backoff value, than the distribution specified by DCF (e.g., by selecting backoff values from range [0,$\frac{\text{CW}}{4}$] instead of [0,CW]).
- Using a different retransmission strategy that does not double the CW value after collision.

Such selfish misbehavior can seriously degrade the throughput of well-behaved nodes. For example, our simulation results (Section 5) show that for a network containing 8 nodes sending packets to a common receiver, with one of the 8 nodes misbehaving by selecting backoff values from range [0,$\frac{\text{CW}}{4}$], the throughput of the other 7 nodes is degraded by as much as 50%. In this paper we propose modifications to IEEE 802.11, for simplifying the detection of such misbehaving nodes as well as for penalizing nodes detected to be misbehaving.

The rest of the paper is organized as follows. A discussion of related work is presented in Section 2. A brief overview of the proposed scheme is outlined in Section 3, and details are presented in Section 4. The evaluation of the proposed scheme is presented in Section 5, and we conclude in Section 6.

2 Related Work

Recent research has investigated misbehavior at the network layer [20, 8, 18] in wireless networks. One approach is to identify misbehaving nodes and avoid such nodes in routing [13]. Another approach is to design protocols that encourage cooperation by penalizing misbehavior [4, 2]. Network layer mechanisms address network layer misbehavior such as tampering with route discovery/maintenance, dropping, delaying or misrouting packets, etc. The scheme we propose addresses selfish misbehavior at the MAC layer, and can complement network layer mechanisms.

Game-theoretic techniques have also been used to develop protocols [10, 11, 14, 15] which are resilient to misbehavior. Konorski [10, 11] proposes a modified backoff algorithm for the MAC protocol that can tolerate selfish misbehavior. Konorski's work assumes that all hosts can hear transmissions from all other hosts. This and some other assumptions by Konorski are not realizable in practice. Michiardi et al. [14], model the nodes in the network as participants in a non-cooperative game with each node attempting to maximize its own utility. By imposing suitable costs on each network operation such as packet forwarding, the game reaches a stable state (called the "Nash equilibrium") where a selfish node cannot gain an advantage over well-behaved nodes. Although protocols developed with game-theoretic techniques may be resilient to misbehavior, they may not achieve the performance of protocols developed under the assumption that all nodes are well-behaved (e.g., IEEE 802.11). The scheme we propose retains the performance of IEEE 802.11 while ensuring detection of misbehavior.

A related approach is to design protocols which are resilient to misbehavior [17, 5]. In the context of TCP, Savage et al. [17] identify certain receiver misbehavior that may allow a misbehaving receiver to gain an throughput advantage over other well-behaved receivers, by exploiting weaknesses in the TCP congestion control algorithm used by a sender. Savage et al. also propose simple modifications to TCP, which prevent a misbehaving receiver from gaining significant throughput advantage. The modifications we propose to IEEE 802.11 protocol are based on a similar design philosophy of incorporating features in a protocol that help detect or discourage misbehavior.

Intrusion detection and tolerance techniques are used as a tool for diagnosing and tolerating misbehavior [19, 3, 16, 7]. Intrusion detection approaches are based on developing a long-term profile of "normal" activities, and identify intrusion by observing deviations from the measured profile. On the other hand, our proposed modifications are not dependent on the availability of a long-term profile of "normal" behavior (when the topology, channel conditions and traffic patterns are dynamic, such a profile may not be accurate).

3 Preliminaries

We define the following terminology used in presenting the proposed scheme.

Sender: Sender is a node which wants to transmit a data packet to a receiver node.

Receiver: Receiver is a node which receives a data packet from a sender node. The receiver monitors the sender node to detect sender's misbehavior.

Sender and *receiver* are the different roles a node can perform. A node may assume the roles of a sender and a receiver at different times. Recall that, in the case of IEEE 802.11 DCF, the sender node transmits a DATA packet to a receiver node after an optional RTS-CTS exchange.

3.1 Motivation and assumptions

The proposed scheme is designed to require minimal modifications to IEEE 802.11 DCF, and allows a receiver to detect sender misbehavior identified earlier. Detecting sender misbehavior is important, for example, in infrastructure-based public wireless networks (e.g., public

wireless networks in airports). Infrastructure-based wireless networks are a popular network architecture used in practice, and consist of base stations that provide connectivity to wireless hosts. The base stations are maintained by the network service providers, and can be trusted. Since the base station is well-behaved, there is no misbehavior when it is sending. On the other hand, wireless hosts sending data to the base station using the DCF mode[1] are untrusted, and may misbehave to gain higher throughput share than competing nodes. Hence, the base station (receiver) is required to detect misbehavior of wireless hosts (senders).

We assume that the receivers are well-behaved while presenting the proposed scheme. We briefly discuss mechanisms to address receiver misbehavior in Section 4.4. We also assume that there is no collusion between the sender and the receiver. For example, these assumptions are valid in the infrastructure-based wireless networks with a trusted base station. The proposed scheme can also be applied to ad hoc networks (self organized networks without a central authority) to detect sender misbehavior as discussed later. The proposed scheme addresses selfish misbehavior (nodes intending to obtain higher throughput or lower delay), and does not consider malicious attacks such as jamming the channel.

3.2 Brief overview of the proposed scheme

The proposed scheme is designed to handle *selfish* MAC layer misbehavior in nodes using IEEE 802.11 DCF mode. A goal of the proposed scheme is to simplify misbehavior detection. In IEEE 802.11 protocol, a sender transmits a RTS (Request to Send) after waiting for a randomly selected number of slots in the range [0,CW]. Consequently, the time interval between consecutive transmissions by the sender can be any value within the above range. Hence, a receiver that observes the time interval between consecutive transmissions from the sender cannot distinguish a well-behaved sender that legitimately selected a small random backoff, from a misbehaving sender that maliciously selected a non-random small backoff. It may be possible to detect sender misbehavior by observing the behavior of senders over a large sequence of transmissions, but this may introduce a large delay in detecting misbehavior. In addition, it may not be feasible to monitor the behavior of senders over a large sequence of transmissions when the node mobility is high.

Hence, we propose modifications to the IEEE 802.11 protocol that enables a receiver to identify sender misbehavior within a small observation interval. Instead of the sender selecting random backoff values to initialize the backoff counter, the receiver selects a random backoff value and sends it in the CTS (Clear to Send) and ACK packets to the sender. The sender uses this assigned backoff value in the next transmission to the receiver. With these modifications, a receiver can identify senders deviating from the protocol by observing the number of idle slots between consecutive transmissions from the sender. If this observed number of idle slots is less than the assigned backoff, then the sender may have *deviated* from the protocol. The magnitude of observed deviations over a small history of received packets is used to diagnose sender misbehavior with high probability.

The proposed scheme also attempts to negate any throughput advantage that the misbehaving nodes may obtain. To achieve this, deviating senders are penalized to discourage misbehavior. When the receiver perceives a sender to have waited for less than the assigned backoff, it adds a penalty to the next backoff assigned to that sender. If the sender does not backoff for the duration specified by the penalty (or backs off for a small fraction of the duration), it significantly increases the probability of detecting misbehavior reliably (as explained later). On the other hand, a misbehaving sender which backs off for the duration specified by the penalty (or a large fraction of it) does not obtain significant throughput advantage over other well-behaved nodes. Hence, with the proposed scheme, it is difficult for a misbehaving node to obtain an unfair share of the channel bandwidth while eluding detection.

4 Proposed Scheme

The proposed scheme has three components. Firstly, the receiver decides at the end of a transmission from the sender, whether the sender deviated from the protocol for that *particular* transmission. A deviation does not always indicate that the sender is misbehaving (as explained later). Next, if the sender has identified a deviation for a transmission from the sender, it penalizes the sender, based on the magnitude of the perceived deviation for that *particular* transmission (*correction scheme*). Lastly, based on the magnitude of the perceived deviation over *multiple* transmissions from the sender, the receiver identifies senders that are indeed misbehaving (*diagnosis scheme*).

4.1 Identifying deviations from the protocol

In the proposed scheme, nodes follow the rules of IEEE 802.11 DCF except for some suitable modifications to the backoff scheme, as explained below. Proposed modifications to the backoff scheme enable a receiver R, to dictate the backoff values to be used by a sender S that is sending packets to R. The first time a sender S sends a packet to a receiver R, S may use an arbitrarily selected backoff value. For all subsequent transmissions, the sender has to

[1]DCF mode is often used as it provides distributed access, and is suitable when the number of wireless hosts and load in the network is not fixed.

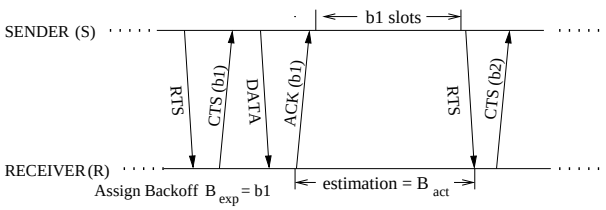


Figure 1. Receiver - Sender interaction

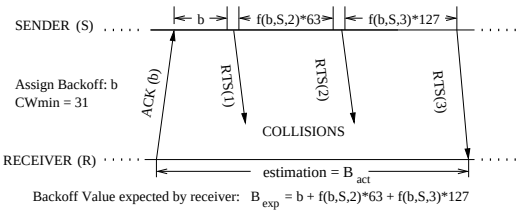


Figure 2. Protocol for retransmissions

use the backoff values provided by the receiver. For example, Figure 1 depicts the receiver-sender interaction in the modified protocol. When the receiver R receives a RTS[2] from the sender S, R assigns a backoff value $B_{exp} = b1$ to S in the CTS packet as well as the subsequent ACK packet as shown in Figure 1 (the assigned backoff may be included in either of CTS or ACK when RTS/CTS exchange precedes data transfer). S is required to use this backoff value $b1$ for sending the next packet to R.

The receiver selects the backoff values B_{exp} assigned to the sender, from the range $[0, CW_{min}]$ (CW_{min} is the minimum contention window value used by IEEE 802.11). The sender may misbehave by backing off for a smaller duration than B_{exp}. The receiver R counts the number of idle slots (B_{act}) observed on the channel, during the interval between the sending of an ACK by R, and the reception of the next RTS from the S. The sender is designated as deviating from the protocol if the observed number of idle slots B_{act} is smaller than a specified fraction α of the assigned backoff B_{exp}, i.e.,

$$B_{act} < \alpha * B_{exp} \ , \ \ 0 < \alpha \leq 1 \qquad (1)$$

A deviation does not necessarily indicate that the sender is misbehaving as the channel conditions seen by the sender and receiver may be different. For example, if the sender senses the channel to be idle and counts down its backoff timer, while the receiver senses the channel to be busy and does not count down its timer, then the transmission from the sender may be falsely designated as a deviation. The parameter α in equation 1 can be suitably chosen, based on the channel conditions, to reduce the incidence of false deviations. For example, if α is chosen to be 0.9, a sender is designated as deviating only when the observed backoff B_{act} is less than 90% of the assigned backoff B_{exp}. However, selecting α to be too small may enable misbehaving senders to elude detection. Hence, we select α to be reasonably high and use the *diagnosis* scheme, presented in Section 4.3, for accurately diagnosing misbehaving nodes.

[2]We assume RTS/CTS exchange is used before data transmission. However, the proposed scheme can be applied even when RTS/CTS exchange is not used.

We now describe the extensions to the proposed scheme for handling packet retransmissions by the sender. Every RTS sent by the sender has an *attempt* number included in a new field in the RTS header. Attempt number is set to 1 after a successful transmission, and is incremented by 1 after every unsuccessful transmission (indicated by the absence of a CTS following a RTS, or the absence of an ACK following a DATA packet). The contention window CW maintained by the sender, is set to CW_{min} after a successful transmission, and after an unsuccessful transmission, CW is set to $min(\ (\ CW_{min} + 1\ ) * 2^{i-1} - 1\ ,\ CW_{max}\ )$ for the i^{th} transmission attempt, as in IEEE 802.11.

Figure 2 demonstrates the working of the protocol after a collision. In the figure, the number in parenthesis next to the RTS is the value of the *attempt* number. When a RTS from the sender is unsuccessful, it selects a new backoff using a deterministic function f, of the backoff previously assigned by the receiver (*backoff*), the unique node identifier (*nodeId*), the attempt number (*attempt*) and contention window (*CW*) as follows (in Figure 2, *backoff*=b, *nodeId*=S, and the *attempt* numbers are 1, 2 and 3.):

$$\text{New Backoff} = f(\textit{backoff}, \textit{nodeId}, \textit{attempt}) * CW$$

The function f used by the sender for computing backoff values for retransmission attempt is given by :- $f(\textit{backoff}, \textit{nodeId}, \textit{attempt}) = (aX + c)\ mod\ (CW_{min} + 1)$, where $a = 5$, $c = 2 * \textit{attempt} + 1$ and $X = (\textit{backoff} + \textit{nodeId})\ mod\ (CW_{min} + 1)$.

The function f generates a uniform random number between [0,CW_{min}] and dividing the number by CW_{min} gives the required number between 0 and 1. The deterministic function f that we use has been carefully chosen to ensure that after collisions, the colliding senders will select different backoff values with high probability [12].

After a collision, the sender has to compute a new backoff value, from a larger range, to reduce the probability of colliding again. When the sender uses a deterministic function f to compute the backoff value x after a collision, the receiver on reception of a packet from the sender can calculate this backoff value x used by the sender, by applying the same deterministic function f, as described below. If such a

deterministic function is not used by the sender, the receiver cannot easily estimate the backoff value used by the sender after a collision.

When a RTS is successfully received at the receiver (after possibly multiple transmission attempts by the sender), the receiver can estimate the number of retransmission attempts by using the *attempt* number field included in the RTS. An *attempt* number value greater than 1 indicates that there was at least 1 unsuccessful transmission attempt by the sender. The receiver can then estimate the total time, B_{exp}, for which the sender was expected to backoff for, using the above deterministic function f as,

$$B_{exp} = backoff + \sum_{i=2}^{attempt} f(backoff, nodeId, i) * CW_i$$

where *attempt* is the attempt number in the received RTS, *backoff* is the backoff assigned to the sender by the receiver, *nodeId* is the sender's identifier and CW_i is the contention window for the i^{th} transmission attempt (computed as in IEEE 802.11) given by $CW_i = min(\ (\ CW_{min} + 1\ ) * 2^{i-1} - 1\ ,\ CW_{max}\ )$. This estimated backoff is then used in checking for possible deviation, by applying equation 1 as explained before.

It may be possible for the sender to provide incorrect *attempt* number values in the RTS. To ensure that senders provide correct attempt numbers, the receiver can sense the channel to identify high collision intervals (when the channel is mostly busy but few transmissions are successful). During these intervals, the receiver can analyze the traffic to identify any sender S achieving larger number of successful transmissions than other nodes, or having smaller average *attempt* values than other nodes. If such a sender S exists, the receiver can intentionally drop RTS packets from S occasionally, and verify that S increments the *attempt* number in the retransmission of RTS. Even a single failure by S to increment the *attempt* number in the retransmission is an immediate proof of misbehavior by S. As S does not know which RTS packets are lost due to collisions and which are intentionally dropped by the receiver, it will be harder for such misbehaving senders to persistently send incorrect *attempt* numbers without being detected. Dropping RTS packets occasionally will not significantly affect the throughput of S.

4.2 Correction Scheme

Nodes deviating from the protocol may obtain a larger throughput share than conforming nodes. The *correction* scheme penalizes deviating nodes by assigning larger backoff values to them than those assigned to conforming nodes. We use the principle that nodes deviating more should be assigned larger penalties. Hence, when the receiver detects a deviation (using equation 1), it measures the deviation $D = max(\ \alpha * B_{exp} - B_{act}\ ,\ 0\ )$, and assigns this measured deviation as a penalty to the sender. From analysis [12] and simulations, we identified the need for additional penalty to effectively penalize the misbehaving nodes. So, the total penalty P is equal to the sum of D and the additional penalty. The next backoff value assigned to the deviating sender is the sum of a random value, selected as in IEEE 802.11 from range $[0, CW_{min}]$, and the computed penalty P. Thus, the deviating sender is dictated to back off for a longer interval, before initiating the next transmission, than it would have needed to without the penalty.

Since the correction scheme adds a penalty for every perceived deviation, a well-behaved sender may be penalized if the receiver incorrectly identifies the sender as deviating from the protocol. As described earlier, this scenario may arise when the channel conditions at a well-behaved sender differs significantly from the channel conditions at the receiver. However, we decided to use the approach of adding a penalty for every perceived deviation to prevent a misbehaving node from trying to adapt to any protocol parameters, and thereby obtain a throughput advantage over well-behaved nodes. Furthermore, in most cases the magnitude of deviation for well-behaved senders is very small. As the penalty added is proportional to the magnitude of deviation, this penalty will be small in most cases for a well-behaved node. Our simulation results show that the average throughput obtained by well-behaved nodes using the *correction* scheme is comparable to that obtained when using IEEE 802.11 protocol.

4.3 Diagnosis Scheme

The *diagnosis* scheme uses two protocol parameters *W* and *THRESH*. The receiver maintains a moving window containing information about the last *W* packets received from each sender. When a new packet is received, the difference $B_{exp} - B_{act}$ is stored in the moving window (B_{exp} is the expected backoff and B_{act} is the observed backoff). A positive (negative) difference indicates that the sender waited for less (more) than the backoff duration expected by the receiver. If the sum of these differences in the previous *W* packets from the sender is greater than a threshold *THRESH*, then the sender is designated as "Misbehaving".

We add both positive differences (sender has waited for less than the required duration, i.e., a "deviation") and negative differences (the sender has waited for more than the required duration) because a well-behaved node perceived as deviating for a packet may appear to backoff for larger than the expected backoff for some other packet. However, a persistently misbehaving node will have positive differences for most packets and is more likely to be diagnosed. The choice of *W* and *THRESH* does not affect the *correction*

scheme. Hence, a sender adapting to these values will still have a penalty added for every perceived deviation, even if the node is not diagnosed to be misbehaving. The parameter *THRESH* used in the protocol may be adaptively selected, based on the channel conditions, to maximize the probability of correct diagnosis of misbehavior, while minimizing the probability of false diagnosis (we defer adaptive selection to future work).

The *correction* scheme is used to penalize potentially misbehaving nodes. However, the correction scheme is not effective if a misbehaving node does not backoff for at least a significant fraction of the assigned penalty when it transmits its next packet. On the other hand, the magnitude of the observed deviation for a sender node that backs off for a small fraction of the assigned penalty will be large, and the *diagnosis* scheme can identify such nodes with high probability. Thus, correction and diagnosis schemes together ensure that a misbehaving node cannot obtain a larger than fair share of the bandwidth without being diagnosed as misbehaving.

After the *diagnosis* scheme identifies a node to be misbehaving, MAC layer may refuse to accept packets from the misbehaving node (by not responding with a CTS). Alternatively, higher layers can be informed of the misbehavior. Using this information, the higher layers or the system administrator may take suitable action. For example, in ad hoc networks, nodes forward packets on behalf of each other. If misbehavior is diagnosed, the network layer may use the diagnosis information to route around misbehaving nodes. The network layer can also refuse to forward packets originating from misbehaving nodes.

4.4 Other issues

In the proposed scheme, there exists a possibility that the receiver may misbehave in assigning backoff values (e.g., untrusted receivers in ad hoc networks). The receiver may assign small backoff values to a sender to obtain data from the sender at a higher rate. This type of misbehavior can be detected using an approach similar to that used for detecting sender misbehavior. For example, the receiver can be required to select the initial backoff values (i.e., backoff value before penalty is added) using some well-known deterministic function g, which the sender is aware of. Hence, the sender can detect a receiver sending small backoff values, and choose to wait for longer interval between transmissions when such misbehavior is detected. An alternate misbehavior is for the receiver to assign large backoff values to a sender. We do not address this misbehavior in our scheme, as this misbehavior is equivalent to the receiver refusing to accept packets from the sender. To encourage the receiver to accept packets from the sender, higher level solutions (e.g., incentive based mechanisms) may be used. The proposed scheme also does not address collusion between a sender and a receiver. Collusion detection will require a third party observer to monitor the behavior of both the sender and the receiver. With the above extensions, the proposed scheme can be used in ad hoc networks as well (details in [12]).

The proposed scheme can be used in conjunction with the upper layers to detect other types of MAC layer misbehavior as well. For example, a misbehaving node may use different MAC addresses for different packet transmissions. A receiver monitoring such a sender cannot effectively penalize the misbehaving node, as the receiver associates different MAC addresses with different nodes. The proposed scheme can be augmented with authentication mechanisms provided by higher layers to identify such misbehaving nodes.

5 Simulation Results

We use the ns-2 [6] simulator for our simulations. The simulator has been extended with modifications needed for our protocol. We have also incorporated modifications to the physical carrier sensing to account for variations in channel conditions at the granularity of a slot. We use the *shadowing* channel model [6]. The shadowing channel model captures the variations in channel conditions over time and space by using a Gaussian random variable, X_{dB}, with zero mean and σ_{dB} standard deviation. The model is represented as

$$\Big[\frac{P_r(d)}{P_r(d_0)}\Big]_{dB} = -10\,\beta\,log\Big(\frac{d}{d_0}\Big) + X_{dB}$$

β is called the Path Loss Exponent, d is the distance between the sender and receiver, $P_r(d)$ is the received power and $P_r(d_0)$ is the power at some reference distance d_0 [6]. For free space propagation β is 2 and we use this value in our simulations. We set σ_{dB} to 1 and the Carrier Sense and Receive Thresholds are selected such that a transmission is received with 50% probability at a distance of 250m, and sensed with 50% probability at a distance of 550m.

In our simulations, all the sender nodes in the network are backlogged. The traffic from the senders to the receivers is a CBR (Constant Bit Rate) flow with rate 2 Mbps and size of CBR packets is 512 bytes. The channel bit rate is 2 Mbps. The simulation time is 50 seconds. The results are averaged over 30 runs of the simulation. Each run is seeded by a different seed and the set of seeds used for different data points is the same.

To model various levels of misbehavior, we define a parameter called "Percentage of Misbehavior" (PM). A misbehaving node with percentage of misbehavior x% transmits a packet after counting down to $(100 - x)$% of the assigned backoff value. We use this parameter to quantify

the *magnitude* of misbehavior, with larger values of PM indicating greater misbehavior. Hence, a node with PM=0% fully counts down the assigned backoff and is well-behaved, whereas a node with PM=100% transmits a packet without counting down any backoff at all.

Simulation topology: We first simulate our proposed protocol for a network having a well-behaved receiver R, and multiple senders transmitting to R. All the nodes in the network are stationary. We use this simple network setting to simplify the evaluation of the proposed protocol's effectiveness in handling sender misbehavior, and identify the various tradeoffs involved. However, the simulation setup includes other traffic in the vicinity of the receiver that can affect the carrier sensing at the receiver R and the senders that communicate with it. We also present later in this section, simulation results for multiple senders and receivers randomly placed in the network.

Figure 3 shows the simulated network. The number of sender nodes around the receiver R is 8 (numbered 1 through 8 in the figure) and node 3 is misbehaving. The 8 sender nodes are placed in a circle of radius 150 meters around R, equidistant from each other. There are 4 other nodes A, B, C, and D in the network, with constant bit rate (CBR) flows of rate 500 Kbps from A to B, and from C to D. The flows A-B and C-D are at a distance of 500 meters on either side of the receiver R as shown in Figure 3. The flows are positioned such that the transmissions on these flows A-B and C-D are sensed with high probability by the receiver R, while farther away sender nodes do *not* sense these transmissions with high probability. For example, in Figure 3, when A sends a packet to B, node 3 may not sense the transmission, while R may sense the transmission.

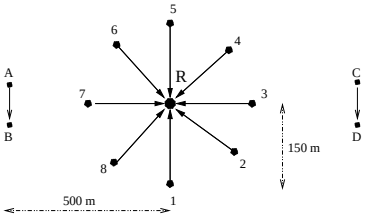


Figure 3. Simulation setup

We evaluate our protocol under two different scenarios by enabling or disabling traffic on flows A-B and C-D:

1. ZERO-FLOW: In this scenario, both traffic flows A-B and C-D are turned off. This gives a symmetric topology with 8 senders sending to a common receiver R.

2. TWO-FLOW: In this scenario, both traffic flows A-B and C-D are turned on. Now, all the senders occasionally appear to be deviating from the protocol (as they sense the channel to be idle when the flow farthest from them is transmitting, while the receiver senses the channel to be busy).

Simulation Metrics: The metrics used in the protocol evaluation are:

1. *Correct Diagnosis:* This is computed as the percentage of packets transmitted by misbehaving senders, which are correctly diagnosed by the receiver (i.e., by the *diagnosis* scheme) as packets from a misbehaving sender. A packet received at the receiver R from a sender S is classified to be from a misbehaving sender only if the measured deviation over the previous *W* packets from S is greater than *THRESH*, as explained in Section 4.3.

2. *Misdiagnosis:* This is computed as the percentage of packets sent by well-behaved senders which are wrongly diagnosed by the receiver as packets from misbehaving senders.

3. *Average throughput of well-behaved nodes:* This is the average throughput per well-behaved sender (designated as "AVG" in the simulation results presented later).

4. *Misbehaving node throughput:* This is the average throughput per misbehaving sender (designated as "MSB" in the simulation results presented later).

5.1 Results

In this section, we evaluate our proposed scheme. The protocol parameters *W* and *THRESH* are used by the *diagnosis* scheme to identify misbehaving nodes, and α is used by the *correction* scheme for computing the penalty. *W* has to be chosen to be a small value to allow reasonably fast misbehavior diagnosis. *THRESH* also has to be reasonably small (a few slots per packet) for diagnosing most misbehavior, but not too small to avoid false diagnosis. Similarly, α has to be close to one to penalize most misbehavior. Based on this intuition, *W*, *THRESH* and α are set to 5 packets, 20 slots (i.e., 4 slots per packet) and 0.9 respectively (simulation results are similar with other reasonable values of *W*, *THRESH* and α as well). Adaptive selection of protocol parameters based on channel conditions has been deferred to future work.

Diagnosis Accuracy: Figure 4 plots the correct diagnosis percentage and misdiagnosis percentage for the ZERO-FLOW and TWO-FLOW scenarios. In the ZERO-FLOW scenario, misdiagnosis percentage is close to 0, and the correct diagnosis percentage is quite high once the extent of misbehavior becomes high. We observe a sharp increase in the correct diagnosis percentage when PM increases above 50% as we have conservatively selected the *THRESH* parameter (node is designated as misbehaving only when the total deviation for previous *W*=5 packets is greater than *THRESH*=20 slots), resulting in small correct diagnosis percentage when the extent of misbehavior is less (however, with the benefit of low misdiagnosis percentage). As the misbehavior increases, the observed deviation rises above

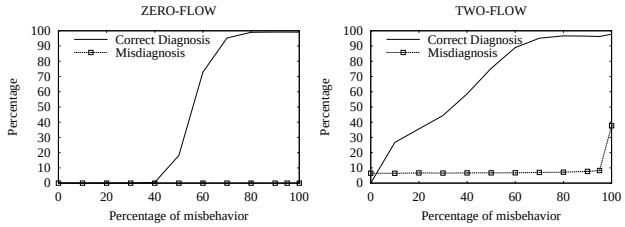


Figure 4. Diagnosis accuracy for varying magnitude of misbehavior

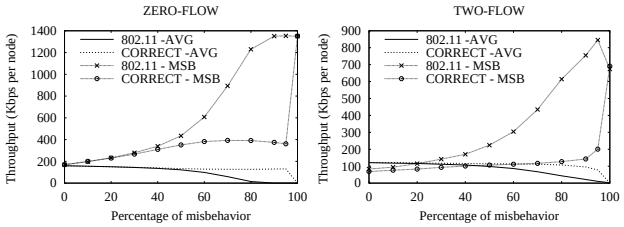


Figure 5. Throughput comparison between IEEE 802.11 and proposed scheme

THRESH=20 slots, and there is a rapid increase in the correct diagnosis percentage.

In the TWO-FLOW scenario, correct diagnosis percentage is fairly high even when the extent of misbehavior is small, but at the price of a higher misdiagnosis percentage. In this scenario, both traffic flows A-B and C-D are on leading to higher interference levels at the senders. This results in an increase in the magnitude of the observed deviations over the ZERO-FLOW scenario, and now the *THRESH* value is not sufficiently high to prevent misdiagnosis. Hence, we observe higher misdiagnosis percentage as well as higher correct diagnosis percentage. Thus, there is a tradeoff involved in achieving low misdiagnosis percentage versus achieving high correct diagnosis percentage.

Throughput in the presence of misbehavior: Figure 5 compares the throughput obtained by a misbehaving node (designated as MSB) using the proposed scheme (designated as CORRECT) with that obtained using IEEE 802.11 protocol (designated as 802.11). The figure also plots the average throughput obtained by the 7 well-behaved senders (1, 2, and 4 through 8) when using both the schemes (designated as AVG). We define fair share as the throughput obtained by a node when it is using IEEE 802.11 protocol and fully conforming to the protocol (i.e., PM=0%). As seen from the figure, throughput of the misbehaving node ("CORRECT - MSB" curve) is restricted to its fair share (except when PM is close to 100%), while for 802.11 ("802.11 - MSB" curve), the misbehaving node obtains a large throughput share even when extent of misbehavior is not too high. In addition, the throughput of well-behaved nodes using the proposed scheme ("CORRECT - AVG" curve) is not affected, except when PM is close to 100%. On the other hand, the throughput of well-behaved nodes using IEEE 802.11 ("802.11 - AVG" curve) starts degrading even when extent of misbehavior is not too high. Hence, the proposed correction scheme is fairly successful in ensuring reasonable throughput for well-behaved nodes, in the presence of misbehaving nodes. When PM is close to 100%, the misbehaving node backs off for a small fraction of the assigned backoff, and consequently the proposed scheme cannot restrict the throughput of the misbehaving node. However, we can see from Figure 4 that the correct diagnosis percentage is significantly high when PM is close to 100%, and in this case, higher layers can be informed of the node misbehavior (as discussed in section 4.3).

Protocol performance without misbehavior: The proposed scheme adds a penalty for every observed deviation from the protocol. With varying channel conditions between the sender and the receiver, well-behaved senders may be incorrectly designated as deviating, and some penalty may be added, possibly degrading their throughput. Hence, we evaluate our protocol in the *absence* of misbehavior as well, to characterize the effect of occasionally penalizing well-behaved nodes. The number of senders communicating with the receiver R is varied from 1 to 64 (replacing the 8 senders in Figure 3). All senders are well-behaved. Figure 6 compares the average throughput obtained by nodes when using IEEE 802.11 (curve "802.11") with that obtained when using the proposed scheme (curve "CORRECT") for varying network sizes under both ZERO-FLOW and TWO-FLOW scenarios. As we can see from the figure, the average throughput obtained when using the proposed scheme is comparable with IEEE 802.11 across different network sizes (the two curves almost overlap in Figure 6). Hence, the correction scheme does not degrade the total network capacity.

We are also interested in comparing the fairness properties of the correction scheme in comparison with that obtained using IEEE 802.11. We use Jain's Fairness Index [9] defined as,

$$\text{fairness index} = \frac{(\sum_f T_f)^2}{N * \sum_f T_f^2}$$

where T_f represents the throughput of a flow f (between a sender node and receiver R), and N is total the number of flows. Fairness index values closer to 1 indicate better fairness. Figure 7 compares the fairness index of IEEE 802.11 and the correction scheme for varying network sizes under both ZERO-FLOW and TWO-FLOW scenarios. For the ZERO-FLOW scenario the fairness index of correction scheme is comparable to that of IEEE 802.11.

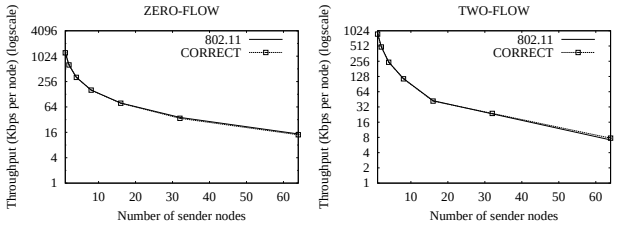


Figure 6. Throughput comparison without misbehavior for varying network sizes

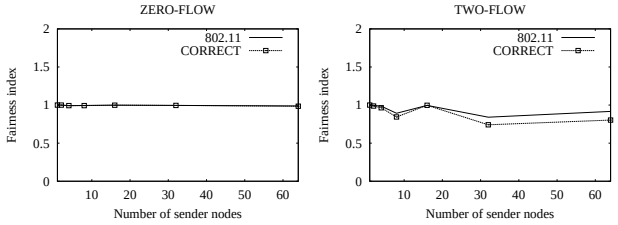


Figure 7. Comparison of fairness index between IEEE 802.11 and proposed scheme

For the TWO-FLOW scenario, the fairness index of our scheme is slightly lesser that that of 802.11. This indicates that our scheme minimally degrades the throughput of some senders, while increasing the throughput of some other senders, since the average throughput (Figure 6) is the same as in IEEE 802.11. This is because, in the TWO-FLOW scenario, a few sender nodes occasionally appear to be deviating from the protocol (from the perspective of the receiver), leading to the addition of a penalty, and thereby resulting in a slight degradation in their throughput. However, the penalty added in those cases is small, resulting in fairness index close to that of 802.11.

There is a tradeoff involved between penalizing misbehaving nodes versus ensuring the fairness of well-behaved nodes. If we use a conservative approach of adding smaller penalty, then misbehaving nodes may obtain a higher throughput share. On the other hand, an aggressive strategy of adding larger penalty may unnecessarily penalize some well-behaved nodes, degrading fairness. We balance this to an extent by penalizing nodes in proportion to their measured deviation. Thus, large penalties are assigned to misbehaving nodes with significant levels of misbehavior, while minimal penalty is assigned for well-behaved nodes falsely designated as deviating.

Responsiveness of Diagnosis Scheme: Figure 8 shows the variation of correct diagnosis percentage with time, measured using the TWO-FLOW scenario, for a partially compliant misbehaving sender with PM% misbehavior. We

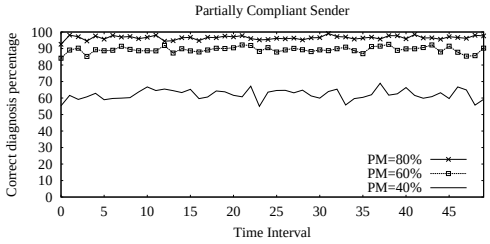
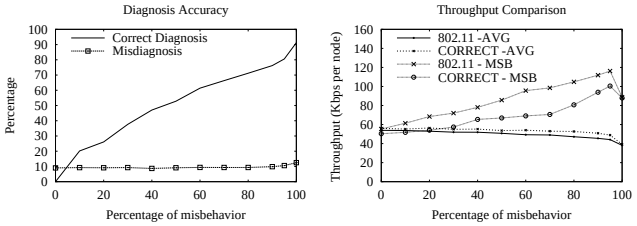


Figure 8. Evaluation of responsiveness of misbehavior diagnosis scheme

measure the correct diagnosis percentage over 1 second intervals starting from time 0, and the results are averaged over 30 runs. For example, the correct diagnosis percentage plotted at 1 second is computed based on the packets received in the interval [1,2] seconds. As seen from Figure 8 the correct diagnosis percentage rapidly reaches a upper threshold, with the value of the threshold dependent on the extent of misbehavior. For example, when the extent of misbehavior is large (PM=80%), the correct diagnosis percentage is consistently above 90%, while it is around 60% when extent of misbehavior is small (PM=40%). With mild misbehavior, the diagnosis scheme cannot always diagnose misbehavior, and thus the correct diagnosis percentage stabilizes at a lower level. We can increase the correct diagnosis percentage by modifying the *W* and *THRESH* parameters of the diagnosis scheme, but that may increase the misdiagnosis percentage.

Protocol performance with random topologies: Figure 9 compares the protocol performance for 30 different random topologies. 40 nodes are placed at random locations in a 1500m by 700m area. 5 nodes, selected at random, are misbehaving. Each node sets up a CBR connection with one of its neighbors, and the connections are always backlogged. Figure 9(a) plots the correct diagnosis percentage and misdiagnosis percentage for different PM (Percentage of Misbehavior) values. As we can see from the figure, the correct diagnosis percentage is high when extent of misbehavior is large, and the misdiagnosis percentage is reasonably small across all values of PM. Figure 9(b) compares the throughput obtained by the misbehaving nodes and the average throughput, for IEEE 802.11 and the proposed *correction* scheme (the notations used are those described earlier for Figure 5). When the extent of misbehavior is small, the correction scheme is fairly successful in restricting the misbehaving nodes to a fair share, thereby ensuring that the throughput of well-behaved nodes are not affected. When the extent of misbehavior is large, the correction scheme is not as successful, but the misbehavior is diagnosed with high probability (as seen from Figure 9(a)).



(a) Diagnosis Accuracy (b) Throughput comparison

Figure 9. Protocol performance for random topology with 40 nodes in 1500m X 700m area

6 Conclusion and Future Work

Handling MAC layer misbehavior is an important requirement in ensuring a reasonable throughput share for well-behaved nodes in the presence of misbehaving nodes. In this paper, we have presented modifications to IEEE 802.11 MAC protocol that simplifies misbehavior detection. Simulation results have indicated that our scheme provides fairly accurate misbehavior diagnosis. The correction scheme we have proposed is effective in restricting the throughput of selfish nodes to a fair share.

We plan to extend the proposed scheme for detecting other types of node misbehavior, such as a node using multiple MAC addresses for obtaining higher bandwidth share, with the support of higher layers. We have already explored preliminary solutions for handling receiver misbehavior and collusion between senders and receivers, which are presented in a related technical report [12]. We plan to further develop these solutions, and evaluate them in scenarios having misbehaving receivers, and collusion between senders and receivers. We also plan to incorporate adaptive selection of protocol parameters into the proposed scheme.

References

[1] IEEE Standard for Wireless LAN-Medium Access Control and Physical Layer Specification, P802.11, 1999.

[2] S. Buchegger and J. Le Boudec. Nodes Bearing Grudges: Towards Routing Security, Fairness, and Robustness in Mobile Ad Hoc Networks. In *Proceedings of the Tenth Euromicro Workshop on Parallel, Distributed and Network-based Processing*, pages 403 – 410, Canary Islands, Spain, January 2002. IEEE Computer Society.

[3] D. J. Burroughs, L. F. Wilson, and G. V. Cybenko. Analysis of Distributed Intrusion Detection Systems Using Bayesian Methods. In *Proceedings of IEEE International Performance Computing and Communication Conference*, April 2002.

[4] L. Buttyan and J. Hubaux. Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks. Technical Report DSC/2001/046, EPFL-DI-ICA, August 2001.

[5] D. Ely, N. Spring, D. Wetherall, S. Savage, and T. Anderson. Robust Congestion Signaling. In *Proceedings of the 2001 International Conference on Network Protocols,Riverside, CA*, November 2001.

[6] K. Fall and K. Varadhan. ns notes and documentation. Technical report, UC Berkley,LBL,USC/ISI,Xerox PARC, 2002.

[7] K. Goseva-Popstojanova, F. Wang, R. Wang, F. Gong, K. Vaidyanathan, K. Trivedi, and B.Muthusamy. Characterizing Intrusion Tolerant Systems using a State Transition Model. In *Proceedings of DARPA Information Survivability Conference and Exposition II (DISCEX'01)*, 2001.

[8] Y. Hu, A. Perrig, and D. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *The 8th ACM International Conference on Mobile Computing and Networking, MobiCom 2002*, pages 12–23, September 2002.

[9] R. Jain, G. Babic, B. Nagendra, and C. Lam. Fairness, call establishment latency and other performance metrics. Technical Report ATM_Forum/96-1173, ATM Forum Document, August 1996.

[10] J. Konorski. Protection of Fairness for Multimedia Traffic Streams in a Non-cooperative Wireless LAN Setting. In *PROMS*, volume 2213 of *LNCS*. Springer, 2001.

[11] J. Konorski. Multiple Access in Ad-Hoc Wireless LANs with Noncooperative Stations. In *NETWORKING*, volume 2345 of *LNCS*. Springer, 2002.

[12] P. Kyasanur and N. H. Vaidya. Detection and Handling of MAC Layer Misbehavior in Wireless Networks. Technical report, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, August 2002.

[13] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad hoc Networks. In *Mobile Computing and Networking*, pages 255–265, 2000.

[14] P. Michiardi and R. Molva. Game theoretic analysis of security in mobile ad hoc networks. Technical Report RR-02-070, Institut Eurecom, April 2002.

[15] N. Nisan. Algorithms for Selfish Agents. In *Proceedings of the Symposium on Theoretical Aspects of Computer Science, LNCS 1563,*, pages 1–17, Berlin, March 1999. Springer.

[16] W. H. Sanders, M. Cukier, F. Webber, P. Pal, and R. Watro. Probabilistic Validation of Intrusion Tolerance. In *Digest of Fast Abstracts: The International Conference on Dependable Systems and Networks, Bethesda, Maryland*, June 2002.

[17] S. Savage, N. Cardwell, D. Wetherall, and T. Anderson. TCP Congestion Control with a Misbehaving Receiver. In *ACM Computer Communications Review,*, pages 71–78, October 1999.

[18] H. Yang, X. Meng, and S. Lu. Self-organized Network Layer Security in Mobile Ad Hoc Networks. In *ACM MOBICOM Wireless Security Workshop (WiSe'02)*, September 2002.

[19] Y. Zhang and W. Lee. Intrusion detection in wireless ad-hoc networks. In *Mobile Computing and Networking*, pages 275–283, 2000.

[20] L. Zhou and Z. J. Haas. Securing Ad Hoc Networks. *IEEE Network*, 13(6):24–30, 1999.