Diagnosing and Penalizing MAC Layer Misbehavior in Wireless Networks*

Technical Report, December 2002

Pradeep Kyasanur
Dept. of Computer Science, and
Coordinated Science Laboratory,
University of Illinois at Urbana-Champaign
1308 West Main St. Urbana, IL 61801
email: kyasanur@uiuc.edu

Nitin H. Vaidya
Dept. of Electrical and Computer Eng., and
Coordinated Science Laboratory,
University of Illinois at Urbana-Champaign
1308 West Main St. Urbana, IL 61801
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. Such selfish misbehavior may arise in those wireless networks where it is not feasible to validate the MAC protocol being used by individual users. 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.

I. 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. The rest of the paper is organized as follows. Section II presents a brief overview of IEEE 802.11, and identifies a potential node misbehavior possible with IEEE 802.11. A discussion of related work is presented in Section III. A brief

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

overview of the proposed scheme is outlined in Section IV, and details are presented in Section V. The evaluation of the proposed scheme is presented in Section VI, and we conclude in Section VII.

II. Description of IEEE 802.11 protocol and identification of possible misbehavior

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). DCF is widely used in both infrastructure-based wireless networks as well as ad hoc wireless networks.

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 is a variable called the *Contention Window* maintained by each node. While the channel is idle, the backoff counter is decremented by one after every time slot[1,2] 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 reserves 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[3]. Both the RTS and the CTS contain the proposed duration of data transmission. Other nodes (in the vicinity of the sender or the receiver) 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 absence 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 VI) show that for a network containing 8 nodes sending packets to a common receiver, with one

[1] A time slot is a fixed interval of time defined in IEEE 802.11 standard.

[2] When the channel state changes from busy to idle, nodes are required to wait for a interval of time, called the DIFS, before counting down of the backoff counter may begin.

[3] RTS-CTS exchange is optional in IEEE 802.11. The scheme we propose can be used with or without an RTS/CTS exchange

of the 8 nodes misbehaving by selecting backoff values from range $[0, \frac{CW}{4}]$, 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.

III. Related Work

Recent research has investigated misbehavior at the network layer [2], [3], [4], [5], [6], [7], [8] in wireless networks. One approach is to identify misbehaving nodes and avoid such nodes in routing [9], [10], [11]. Another approach is to design protocols that encourage cooperation by penalizing misbehavior [12], [13]. A general discussion on security issues in ad hoc networks including secure routing, authentication, key management and mechanisms for encouraging co-operation is available in a working report on security [14]. 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.

A related approach to encouraging co-operation is to design protocols which are resilient to misbehavior [15], [16], [17]. In the context of TCP, Savage et al. [15] 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. Similarly, Anderson et al. [17] propose a set of design guidelines for improving the robustness of Internet protocols against arbitrary failures. The guidelines advocate a defensive design philosophy that takes into account the possibility of malicious attacks, misimplementation, and misconfiguration at every step of the design process. 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.

Protocol verification is an approach used to formally verify the correctness of protocols [18], [19]. Protocol verification by itself does not provide guidelines for designing *secure* protocols *resilient* to misbehavior. However, protocol verification can be used to validate security protocols [18] or identify potential problems in protocol specification [19]. We do not consider verification mechanisms in our work.

Nisan et al. [20], [21] apply game-theoretic techniques for algorithm design. The participants (nodes) are assumed to possibly deviate from the algorithm (i.e., protocol). The goal of these papers is to ensure in advance that the participants' interests are best served by behaving correctly. These papers and other work using a similar approach focus on formally modeling each participant (and the associated utility function), and the algorithm (e.g. routing), and obtaining bounds on complexity of computing the payment schemes. In contrast, our work focuses on designing implementable protocols that are resilient to misbehavior.

Game-theoretic techniques have also been used to develop protocols [22], [23], [24], [25], [26] which are resilient to misbehavior. Konorski [22], [23] 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. Mackenzie et al. [24], [27] consider *random access games* using game-theoretic strategies for developing medium access control protocols. The objective of the game is to reach a stable state where a greedy node cannot gain an advantage over well-behaved nodes. In [25], [26], the nodes in the network are modeled as participants in a non-cooperative game having different utility functions. Each node attempts to maximize its 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. While the game-theoretic approach has the potential for designing protocols which are resilient to misbehavior, the network performance at Nash equilibrium might be far from optimal (i.e., Nash equilibrium does not guarantee that the network performance obtained is close to that possible with a different protocol such as IEEE 802.11). However, the scheme we propose achieves performance comparable to that of IEEE 802.11.

Intrusion detection techniques are used as a tool for diagnosing misbehavior [28], [29], [30]. These 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).

Intrusion tolerance is another approach to security that allows continued operation of networks and systems in spite of malicious attacks [31], [32]. Sanders et al. [31] present a probabilistic validation approach that uses a model of the system and a set of measurements for estimating values of model parameters. This enables them to compare different intrusion tolerant systems based on their probability of resisting malicious attacks. Similarly, Goseva-Popstojanova et al. [32] present a state transition model to characterize the properties of intrusion tolerant systems. Intrusion tolerance approaches focus primarily on designing mechanisms into protocols for *tolerating* misbehavior. On the other hand, our work focuses on designing mechanisms into protocols that simplifies *detection* of misbehavior and enables *penalizing* misbehavior.

IV. PRELIMINARIES

In this section we define the terminology used in the rest of the paper. We then present a brief overview of the proposed scheme and the underlying assumptions.

A. Terminology:

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.

B. 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. One goal of the proposed scheme is to simplify misbehavior detection. In IEEE 802.11 protocol, a sender transmits a RTS 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 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 details are in Section V.

Another goal of the proposed scheme is to negate any throughput advantage that the misbehaving nodes may obtain. To achieve this, we penalize the deviating senders 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.

The proposed scheme is designed to require minimal modifications to IEEE 802.11 DCF, and allows a receiver to detect sender misbehavior. As noted before, DCF is widely used in practice. We assume that the receivers are well-behaved. We also assume that there is no collusion between the sender and the receiver. For example, these assumptions are valid in infrastructure-based wireless networks. 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. Hence, when the base station is the receiver, it can use the proposed scheme to detect misbehavior by the senders (wireless hosts). The base station being well-behaved, there is no misbehavior when it is sending. The proposed scheme can also be applied to ad hoc networks to detect misbehavior by a node when it is sending (described in another technical report [33]). We also discuss some enhancements for handling receiver misbehavior and collusion, as well as alternative approaches to our proposed scheme, in [33].

V. Proposed Scheme

In this section, we first present our modifications to IEEE 802.11 protocol for identifying *deviations* from the protocol. When the receiver receives a packet, if it perceives the sender to have not backed off "sufficiently" (details are presented below), it designates the sender to have *deviated* from the protocol. We then present a *correction* scheme that enables a receiver to penalize a deviating sender, based on the magnitude of the perceived deviation. A deviation does not always indicate that the sender is misbehaving (as explained later), and we present a *diagnosis* scheme to identify, with high probability, senders that are indeed misbehaving. We conclude this section with a discussion on other aspects of the proposed scheme.

A. 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 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[4] from the sender S, R assigns a backoff

[4]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.

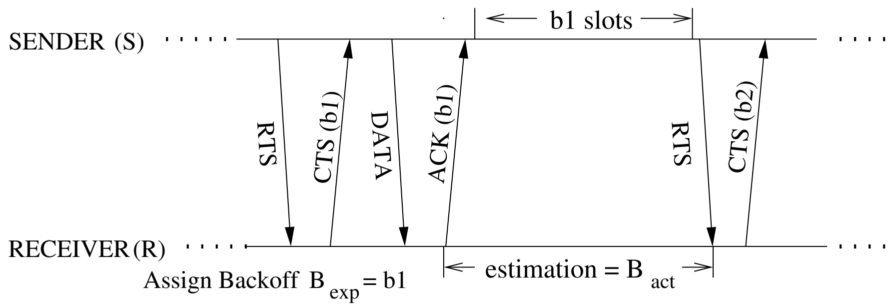


Fig. 1. Timeline depicting the Receiver - Sender interaction

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.,

$$B_{act} < \alpha * B_{exp} \quad , \quad 0 < \alpha \leq 1 \tag{1}$$

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}. Thus, observed B_{act} values in the range of 90% to 100% of B_{exp}, will not be designated as deviations. However, selecting α to be too small may enable misbehaving senders to elude detection. Hence, we select α to be reasonably high and use a *diagnosis* scheme, presented in Section V-C, that reduces the probability of falsely accusing well-behaved senders to be misbehaving, while retaining a high probability of detecting misbehaving senders.

We now describe the extensions to the proposed scheme for handling packet losses (possibly due to collisions).

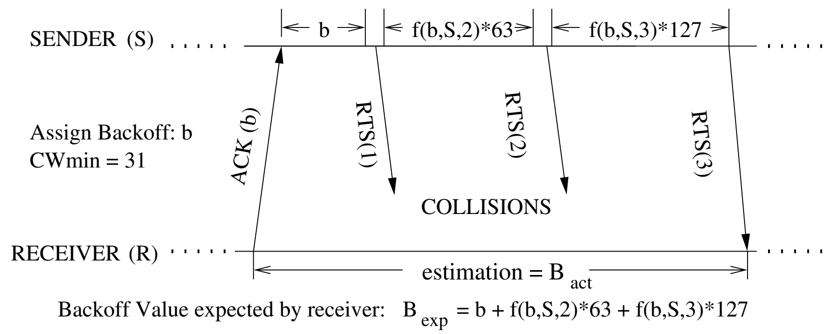


Fig. 2. Timeline depicting protocol behavior during retransmissions

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 an unsuccessful transmission. An unsuccessful transmission is 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:

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

In Figure 2, *backoff*=b, *nodeId*=S, and the *attempt* numbers are 1, 2 and 3. 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. The deterministic function f that we use (described in Appendix I) has been carefully chosen to ensure that after collisions, the colliding senders will select different backoff values with high probability.

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,

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

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. In Section V-D we propose a mechanism to identify such misbehavior.

B. Correction Scheme

Nodes deviating from the protocol may obtain a larger throughput share than conforming nodes. Hence, our *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 [33] and simulations, we identified the need for additional penalty to effectively penalize the misbehaving nodes. So, the total penalty P is given by,

$$P = D + \text{Additional Penalty}$$

where D is the measured deviation. The justification for the additional penalty and its computation is described in Appendix II. 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 required to back off for a longer interval (before initiating the next transmission) than it would have needed to (without the penalty).

The *correction* scheme adds a penalty for every perceived deviation. A consequence of using this approach is that a well-behaved sender may be penalized if the receiver 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 this approach to prevent a misbehaving node from trying to adapt to any protocol parameters, and thereby obtain a throughput advantage over well-behaved nodes. In addition, 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. The simulation results presented later 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.

C. Diagnosis Scheme

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. After the *diagnosis* scheme identifies a node to be misbehaving, higher layers can be informed of the misbehavior. Using this information, the higher layers or the system administrator may take suitable action. For example, the network layer may use such information to route around misbehaving nodes. The network layer can also refuse to forward packets originating from misbehaving nodes.

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, and hence, a sender adapting to these values will not obtain any throughput advantage (because a penalty is added for every perceived deviation, even if a node is not diagnosed to be misbehaving). The parameter *THRESH* used in the protocol may be adaptively selected, based on the channel conditions (we defer adaptive selection to future work).

The combination of *correction* and *diagnosis* schemes is effective in handling misbehavior:

- Nodes that ignore the assigned backoff or back off for a small fraction of the assigned backoff are quickly detected to be misbehaving, since the measured deviation is large, allowing detection of misbehavior with high probability.
- Nodes that adhere to a significant portion of the assigned backoff (and thereby a significant portion of the penalty) are restricted to a bandwidth share comparable to that of well-behaved nodes.

D. Other issues

The scheme used for detecting deviations requires the senders to correctly set the *attempt* number when a RTS is retransmitted. To ensure that senders follow this procedure, 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. (We have not enabled this feature in our simulations.)

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.

VI. Simulation Results

We have performed simulations to evaluate the proposed scheme. We present the simulation model and simulation results in the following sections.

A. Simulation Model

We use the ns-2 [34] 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.

The *shadowing* channel model [34] 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 [34]. 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 simulation 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.

B. 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.

We evaluate our protocol under three 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) ONE-FLOW: In this scenario, only traffic flow C-D is turned on. In this scenario, receiver R and nodes 1 through 5 sense the transmission from C to D with high probability, while nodes 6, 7, and 8 do not.

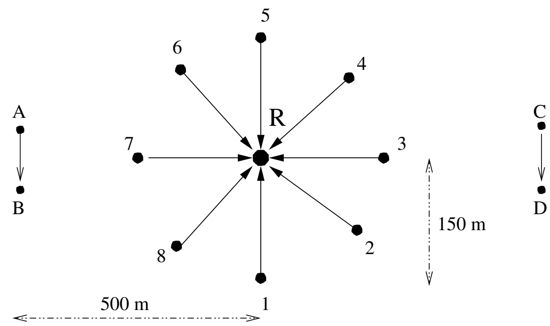


Fig. 3. Simulation setup for a network with the well-behaved receiver R and 8 senders

Consequently, nodes 6, 7, and 8 may occasionally appear to be deviating from the protocol (as they sense the channel to be idle while receiver R senses the channel to be busy). We select node 3 to be the misbehaving node. Hence, this scenario tests our protocol performance when node 3 is actually misbehaving, while nodes 6-8 may falsely appear to be misbehaving.

3) 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).

C. Simulation Metrics

The metrics we use to evaluate the protocol are:

- *Correct Diagnosis:* This is computed as the percentage of packets transmitted by misbehaving senders, which are correctly diagnosed by the receiver 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 V-C.
- *Misdiagnosis:* This is computed as the percentage of packets sent by well-behaved senders which are wrongly diagnosed by the receiver to be packets from misbehaving senders.
- *Average throughput of well-behaved nodes:* This is the average throughput per well-behaved sender (designated as "AVG TPUT" in the simulation results presented later).
- *Misbehaving node throughput:* This is the average throughput per misbehaving sender (designated as "MSB TPUT" in the simulation results presented later).

D. Simulation Results

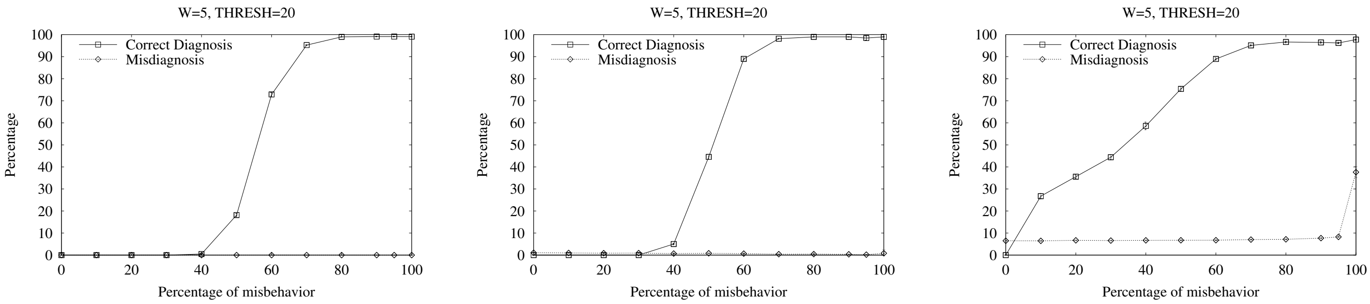
In this section, we evaluate our proposed scheme. The protocol parameters *W*, *THRESH* and α are set to 5, 20 and 0.9 respectively. Recall that *W* and *THRESH* are used by the *diagnosis* scheme to identify misbehaving nodes, and α is used by the *correction* scheme for computing the penalty.

Figure 4(a) plots the correct diagnosis percentage and misdiagnosis percentage for the ZERO-FLOW scenario. We can see from the figure that misdiagnosis percentage is close to 0. The correct diagnosis percentage is quite high once the extent of misbehavior becomes high. We can observe a sharp increase in the correct diagnosis percentage when PM increases above 50%. This is because, 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, as seen later in Figure 6(a), the throughput of well-behaved nodes is not adversely affected in this case. As the misbehavior increases, the observed deviation rises above THRESH=20 slots, and there is a rapid increase in the correct diagnosis percentage.

Figure 4(b) plots the correct diagnosis percentage and misdiagnosis percentage for the ONE-FLOW scenario and the results are similar to that of ZERO-FLOW scenario. For the ONE-FLOW scenario, nodes 6, 7 and, 8 do not sense the channel to be busy when the receiver does, and hence, there is a slight increase in the misdiagnosis percentage. However, the increase is not significant as our choice of *THRESH*=20 slots is sufficiently conservative to keep the misdiagnosis percentage low.

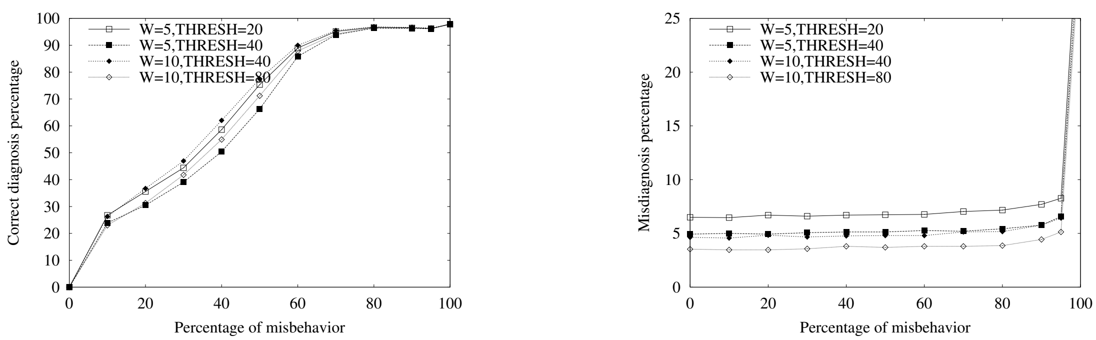
Figure 4(c) plots the correct diagnosis percentage and misdiagnosis percentage for the TWO-FLOW scenario. In this case, correct diagnosis percentage is fairly high even when the extent of misbehavior is small, but at the price of a higher misdiagnosis percentage. In this scenario, both traffic flows 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 ONE-FLOW scenario, and now the *THRESH* value is not sufficiently high to prevent misdiagnosis. Hence, we can observe a higher misdiagnosis percentage as well as a higher correct diagnosis percentage. Thus, there is a tradeoff involved in achieving low misdiagnosis percentage versus achieving high correct diagnosis percentage. We will be looking at adaptively selecting *W* and *THRESH* parameters as part of future work.

Figure 5 compares the correct diagnosis and misprediction percentage for varying window and threshold values. The results are for the TWO-FLOW scenario. As discussed above, increasing the *THRESH* value decreases the misdiagnosis percentage with a marginal reduction in the correct diagnosis percentage as well. Alternatively, increasing the window (*W*) while retaining the same ratio between the window size and the threshold (e.g., using $W = 10$, *THRESH*$= 40$ instead of using $W = 5$, *THRESH*$= 20$) reduces the misdiagnosis percentage. This is because, increasing the window value allows for larger observation intervals before a diagnosis is made, and consequently false deviations arising out of short-term variations in the channel conditions are better masked. However, increasing the window value will result in larger delays before a misbehaving node can be diagnosed



(a) ZERO-FLOW: Zero surrounding flows (b) ONE-FLOW: One Surrounding flow (c) TWO-FLOW: Two Surrounding flows

Fig. 4. Correct diagnosis and misdiagnosis percentages for varying Percentage of Misbehavior

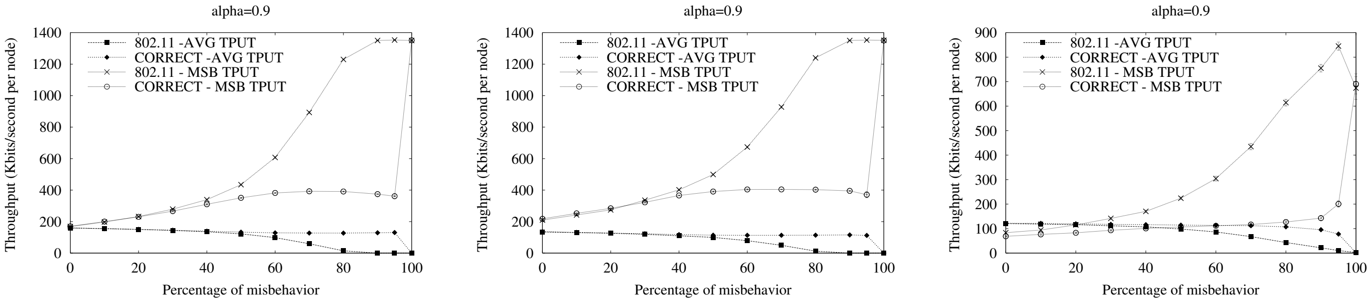


(a) Correct Diagnosis Percentage (b) Misdiagnosis Percentage

Fig. 5. Correct diagnosis and misdiagnosis percentages for different window and threshold values

as such. Thus, there is a tradeoff involved in reducing the misdiagnosis percentage versus reducing the delay in diagnosing misbehavior.

Figure 6 compares the throughput obtained by a misbehaving node (designated as MSB TPUT) using our *correction 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 TPUT). We define the 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 TPUT" curve) is restricted to its fair share (except when PM is close to 100%), while for 802.11 ("802.11 - MSB TPUT" 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 TPUT" 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 TPUT"



(a) ZERO-FLOW: Zero surrounding flows (b) ONE-FLOW: One Surrounding flow (c) TWO-FLOW: Two Surrounding flows

Fig. 6. Throughput comparison between IEEE 802.11 and proposed scheme for varying Percentage of Misbehavior

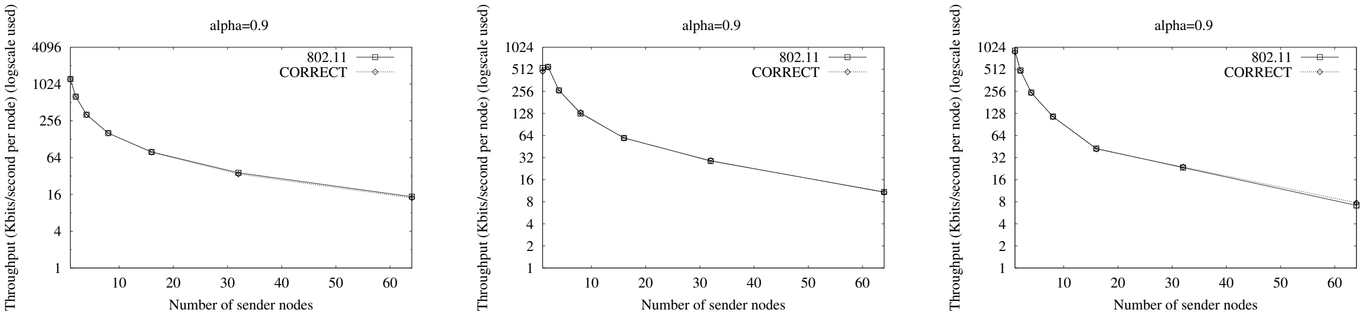
curve) starts degrading even when extent of misbehavior is not too high. Hence, the proposed correction scheme is fairly successful in ensuring reasonable throughput for well-behaved nodes, even in the presence of misbehaving nodes. When PM is close to 100%, the misbehaving node backs off for a small fraction of the assigned backoff, and consequently the *correction* 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 V-C).

The *correction* 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 7 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 each of the three scenarios ZERO-FLOW, ONE-FLOW and TWO-FLOW. As we can see from the figure, the average throughput obtained when using the *correction* scheme is comparable with IEEE 802.11 across different network sizes (the two curves almost overlap in Figure 7). 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 [35] defined as,

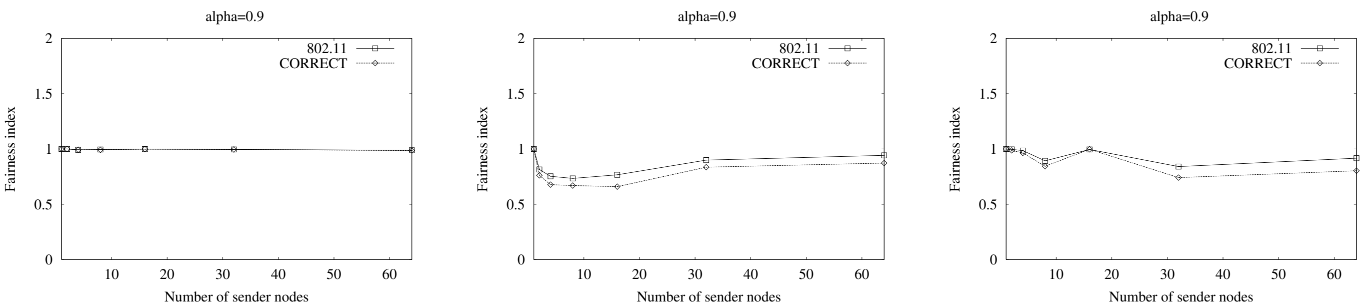
$$\text{fairness index} = \frac{(\sum_f T_f)^2}{N * \sum_f T_f^2}$$

where T_f represents the throughput of a flow f (between a sender node and receiver R), and N is total the number



(a) ZERO-FLOW: Zero surrounding flows (b) ONE-FLOW: One Surrounding flow (c) TWO-FLOW: Two Surrounding flows

Fig. 7. Throughput comparison without misbehavior for varying network sizes (curves for 802.11 and CORRECT almost overlap)



(a) ZERO-FLOW: Zero surrounding flows (b) ONE-FLOW: One Surrounding flow (c) TWO-FLOW: Two Surrounding flows

Fig. 8. Comparison of fairness index between IEEE 802.11 and proposed scheme for varying network sizes

of flows. Fairness index values closer to 1 indicate better fairness.

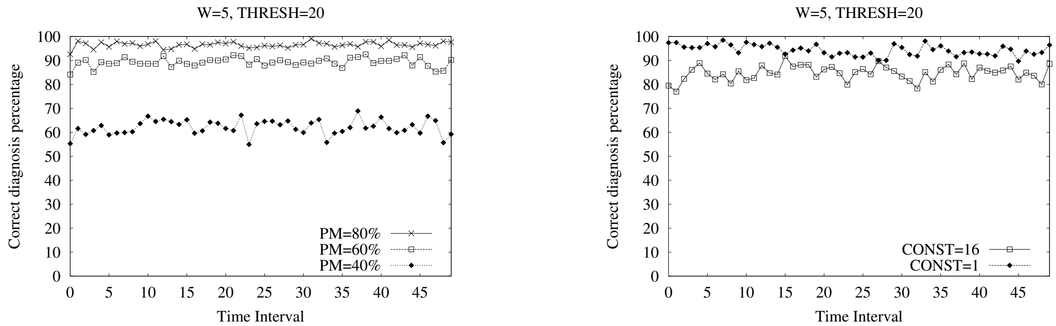
Figure 8 compares the fairness index of IEEE 802.11 and the correction scheme for varying network sizes under each of the three scenarios ZERO-FLOW, ONE-FLOW and TWO-FLOW. For the ZERO-FLOW scenario (Figure 8(a)) the fairness index of correction scheme is comparable to that of IEEE 802.11. For the ONE-FLOW scenario (Figure 8(b)), the fairness index of our scheme is only slightly smaller that that of 802.11. This indicates that our scheme slightly degrades the throughput of some senders, and increases the throughput of some other senders, since the average throughput (Figure 7(b)) is the same as in IEEE 802.11. This is because, in the ONE-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. Hence, as seen from Figure 8(b), the fairness index of the proposed scheme is close to that of 802.11. For the TWO-FLOW scenario (Figure 8(c)), we see similar results as in the ONE-FLOW scenario, as the receiver occasionally adds a penalty for well-behaved nodes.

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.

Figure 9 shows the variation of correct diagnosis percentage with time, measured for the TWO-FLOW scenario. We 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. Figure 9(a) plots the variation of correct diagnosis percentage for partially compliant misbehaving sender nodes with PM% misbehavior. In this case, the correct diagnosis percentage rapidly reaches a upper threshold, the value of the threshold depending 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 a mild misbehavior, the diagnosis scheme cannot always detect the 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.

Figure 9(b) plots the variation of correct diagnosis percentage for a misbehaving sender (namely sender 3) that picks a fixed backoff interval for all packets, irrespective of the backoff assigned by the receiver. In the figure, CONST=1 represents that the sender 3 is waiting for exactly 1 idle slot before transmitting a RTS (irrespective of the backoff assigned by the sender), and similarly CONST=16 represents that the sender 3 is waiting for exactly 16 idle slots before transmitting a RTS. We can see from the figure that the correct diagnosis percentage reaches a high value at the end of the first 1 second interval itself, implying misbehavior is quickly identified. The penalty added by the *correction* scheme increases when a misbehaving node waits for a very small fraction of the previously assigned backoff. A node continuing to misbehave can be easily diagnosed as misbehaving, since the measured deviations become large. Thus, the prediction accuracy rapidly reaches a high value.

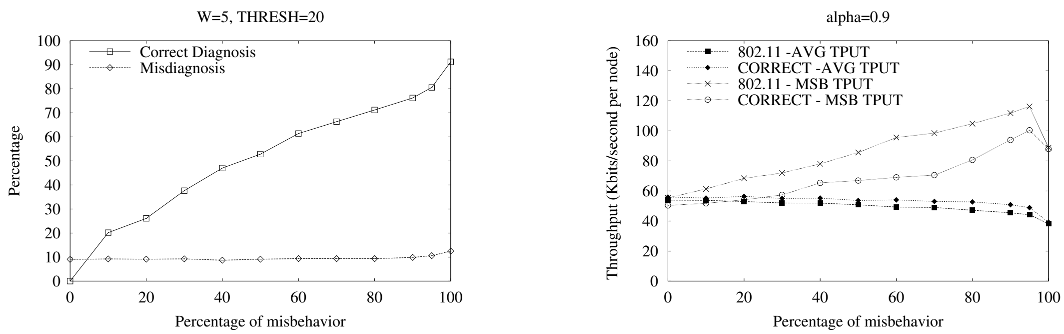
Figure 10 compares the protocol performance for random topologies. 40 nodes are placed at random locations in a 1500m by 700m area. We run the simulations for 30 different topologies. 5 nodes, selected at random, are misbehaving. Each node sets up a CBR connection with one of its neighbors, and the connections are always backlogged. Figure 10(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



(a) Partially conformant sender

(b) Non-conformant sender

Fig. 9. Evaluation of responsiveness of misbehavior diagnosis scheme



(a) Diagnosis Accuracy

(b) Throughput comparison

Fig. 10. Comparison of protocol performance for random topology with 40 nodes in 1500m X 700m area

extent of misbehavior is large, and the misdiagnosis percentage is reasonably small across all values of PM. Figure 10(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 6). As we can see from the Figure 10(b), 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 10(a)).

VII. 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 [33]. 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] L. Zhou and Z. J. Haas, "Securing Ad Hoc Networks," *IEEE Network*, vol. 13, no. 6, pp. 24–30, 1999.

[3] 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*, September 2002, pp. 12–23.

[4] 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.

[5] P. Papadimitratos and Z.J. Haas, "Secure Routing for Mobile Ad Hoc Networks.," in *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002), San Antonio, TX*, January 2002.

[6] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding-Royer, "A Secure Routing Protocol for Ad Hoc Networks," in *Proceedings of 2002 IEEE International Conference on Network Protocols (ICNP)*, November 2002.

[7] Y. Hu, D. B. Johnson, and A. Perrig, "SEAD: Secure Efficient Distance Vector Routing in Mobile Wireless Ad Hoc Networks," in *Fourth IEEE Workshop on Mobile Computing Systems and Applications* (WMCSA '02), June 2002, pp. 3–13.

[8] S. Yi, P. Naldurg, and R. Kravets, "Security-Aware Ad hoc Routing for Wireless Networks," in *Second ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'01)*, Long Beach, CA, USA, October 2001.

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

[10] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens, "An On-Demand Secure Routing Protocol Resilent to Byzantine Failures," in *ACM MOBICOM Wireless Security Workshop (WiSe'02)*, September 2002.

[11] Y. Xue and K. Nahrstedt, "Bypassing Misbehaving Nodes in Ad Hoc Routing," Tech. Rep. UIUCDCS-R-2002-2306, UILU-ENG-2002-1749, Department of Computer Science, University of Illinois at Urbana-Champaign, November 2002.

[12] L. Buttyan and J.-P. Hubaux, "Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks," Tech. Rep. DSC/2001/046, EPFL-DI-ICA, August 2001.

[13] 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*, Canary Islands, Spain, January 2002, pp. 403 – 410, IEEE Computer Society.

[14] L. Buttyan and J.-P. Hubaux (eds), "Report on a Working Session on Security in Wireless Ad Hoc Networks," *Mobile Computing and Communications Review*, vol. 6, no. 4, November 2002.

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

[16] 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.

[17] T. Anderson, S. Shenker, I. Stoica, and D. Wetherall, "Design Guidelines for Robust Internet Protocols," in *First Workshop on Hot Topics in Networks (HOTNET-I)*, Princeton, NJ, USA, October 2002.

[18] W. Marrero, E. Clarke, and S. Jha, "A Model Checker for Authentication Protocols," in *Proceedings of DIMACS Workshop on Design and Formal Verification of Crypto Protocols*, September 1997.

[19] I. Schieferdecker, "Abruptly Terminated Connections in TCP - A Verification Example," in *Proceedings of COST 247 International Workshop on Applied Formal Methods in System Design*, June 1996.

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

[21] N. Nisan and A. Ronen, "Algorithmic Mechanism Design," in *ACM Symposium on Theory of Computing*, 1999, pp. 129–140.

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

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

[24] A. B. Mackenzie and S. B. Wicker, "Game Theory and the Design of Self-Configuring, Adaptive Wireless Networks," *IEEE Communications Magazine*, vol. 39, no. 11, pp. 126– 131, 2000.

[25] P. Michiardi and R. Molva, "Game theoretic analysis of security in mobile ad hoc networks," Tech. Rep. RR-02-070, Institut Eurecom, April 2002.

[26] P. Michiardi and R. Molva, "Making greed work in mobile ad hoc networks," Tech. Rep. RR-02-069, Institut Eurecom, March 2002.

[27] A. B. Mackenzie and S. B. Wicker, "Selfish Users in Aloha: A Game-Theoretic Approach," in *Proceedings of IEEE Vehicular Technology Conference*, October 2001.

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

[29] N. Ye, M. Xu, and S. M. Emran, "Probabilistic Networks with Undirected Links for Anomaly Detection," in *Proceedings of IEEE Workshop on Information Assurance and Security*, June 2000.

[30] 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.

[31] 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.

[32] 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.

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

[34] K. Fall and K. Varadhan, "ns notes and documentation," Tech. Rep., UC Berkley,LBL,USC/ISI,Xerox PARC, 2002.

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

APPENDIX I: DESCRIPTION OF FUNCTION USED IN DETECTION SCHEME

In our simulations, the function f used by the sender for computing backoff values for retransmission attempt is given by :- $f(\textit{backoff, nodeID, attempt}) = (aX + c)\ mod\ (CW_{min} + 1)$, 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 detailed discussion of choice of parameters for f, and a evaluation of its properties is described in our technical report [33].

APPENDIX II: COMPUTATION OF PENALTY FOR MISBEHAVING NODES

The amount of deviation measured by the receiver is given by, $D = max(\ \alpha * B_{exp} - B_{act}\ ,\ 0\ )$, where B_{exp} is the number of slots the sender was expected to backoff, B_{act} is the actual number of slots that the sender was measured to have backed off for and α is a protocol parameter in range [0,1]. On simulating this approach by adding D as the penalty, we found that the misbehaving nodes still obtain a larger share of the bandwidth than conforming nodes, although the share obtained by misbehaving nodes is lesser than what they would have obtained without any correction. Misbehaving nodes obtain a higher share of bandwidth as they suffer lesser number of collisions per successful transmission in comparison to well behaved nodes. A detailed explanation is in [33].

Let the misbehaving node suffer $E_{\textit{diff}}$ fewer collisions per successful transmission on an average in comparison to those suffered by a well-behaved node. The theoretical analysis for estimating $E_{\textit{diff}}$ is presented in our technical report [33]. Once we have estimated $E_{\textit{diff}}$, we compute the additional penalty as the number of slots the node would have additionally backed off for, if it had actually suffered $E_{\textit{diff}}$ additional collisions.

$$AdditionalPenalty = \sum_{i=attempt+1}^{attempt+E_{\textit{diff}}} f(\textit{backoff,nodeId,i}) * CW_i$$

where *attempt* is the attempt number included in the RTS, *backoff* is the backoff assigned by the receiver, CW_i is the contention window size for the i^{th} transmission attempt, and *nodeId* is the identifier of the misbehaving node.