SELFISH MISBEHAVIOR AT MEDIUM ACCESS CONTROL LAYER IN WIRELESS NETWORKS

BY

PRADEEP NARAYANASWAMY KYASANUR

B.E., Mangalore University, 2001

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2003

Urbana, Illinois

Selfish Misbehavior at Medium Access Control Layer in Wireless Networks

Approved by
Dr. N. H. Vaidya

____

____

ABSTRACT

Wireless medium access control (MAC) protocols, such as IEEE 802.11, use distributed contention resolution mechanisms for sharing the wireless channel. In public area wireless networks, such as those used at airports, it is possible that the participating hosts may deviate from the specified MAC protocol. Selfish hosts that fail to conform to the MAC protocol may obtain an unfair throughput share. For example, IEEE 802.11 requires hosts competing for access to the channel to wait for a "backoff" interval, randomly selected from a specified range, before initiating a transmission. Selfish hosts may wait for smaller backoff intervals than well-behaved hosts, thereby obtaining an unfair advantage. Thus, with the increasing popularity of public area wireless networks, there is a need for mechanisms that detect selfish users attempting to obtain an unfair advantage. In this thesis, we identify certain selfish misbehavior possible in IEEE 802.11 networks. We present modifications to the IEEE 802.11 protocol to simplify detection of such selfish hosts, and analyze the optimality of the detection strategy. We also present a penalty scheme for punishing selfish misbehavior, to act as a disincentive for future misbehavior. We develop two misbehavior models to capture the behavior of misbehaving hosts. Simulation results under these misbehavior models indicate that our detection and penalty schemes are successful in handling MAC layer misbehavior.

ACKNOWLEDGMENTS

I would like to thank my advisor, Prof. Nitin Vaidya, for his guidance and support. I am indebted to my family members and friends for the encouragement and help they have provided.

I also acknowledge UIUC Campus Research Board for financially supporting my thesis research.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

CHAPTER 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 mechanism is typically based on cooperative protocols (e.g., random backoff before transmission) that attempt to ensure a reasonably fair throughput share for all the participating hosts. In environments where hosts in the network are untrusted, some hosts 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* hosts that deviate from the contention resolution protocol can reduce the throughput share received by conforming hosts. Thus, development of mechanisms for detecting and handling *selfish* misbehavior is essential.

Wireless networks can be classified into infrastructure-based networks and ad hoc networks. Infrastructure-based networks have a centralized base station. Hosts in the wireless network communicate with each other, and with other hosts on the wired network, through the base station. Infrastructure-based networks are commonly used to provide wireless network services in numerous environments such as college campuses, airports, homes, etc. Ad hoc networks are characterized by the absence of any infrastructure support. Hosts in the network are self-organized, and forward packets on behalf of each other, enabling communication over multi-hop routes. Ad hoc networks are envisaged for use in battlefield communication, sensor communication, etc. IEEE 802.11 is a MAC layer protocol that can be used in infrastructure-based networks as well as in ad hoc networks.

IEEE 802.11 has two mechanisms for contention resolution; a centralized mechanism called point coordination function (PCF), and a fully distributed mechanism called distributed coordination function (DCF). PCF needs a centralized controller (located at the base station), and can be used only in infrastructure-based networks. PCF is an optional feature in IEEE 802.11, and is not supported by all IEEE 802.11 implementations. DCF provides distributed access, and is the only contention resolution mechanism that can be used for ad hoc networks. DCF is also suitable when the number of hosts and load in the network is not fixed. Consequently, DCF is widely used in practice for infrastructure-based networks as well. In this thesis, we address a misbehavior possible in the DCF mode. Using PCF, instead of DCF, in infrastructure-based networks may alleviate the selfish misbehavior that we identify, but this results in degraded performance when the incidence of misbehavior is low.

1.1 Details of DCF and illustration of possible misbehavior

DCF uses CSMA/CA (carrier sense multiple access/collision avoidance) for resolving contention among multiple hosts accessing the channel. A host (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 host. 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 host 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 host. 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 hosts 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 (acknowledgment) packet to acknowledge successful reception of the data packet. If a host's data transmission is successful, the host resets its CW to a minimum value (CW_{min}); otherwise, if a host'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 host 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], or by always selecting a fixed backoff of 1 slot).
- 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 hosts. For example, our simulation results (Section 6) show that for a network containing 8 hosts sending packets to a common receiver, with one of the 8 hosts misbehaving by selecting backoff values from range $[0,\frac{\text{CW}}{4}]$, the throughput of the other 7 hosts is degraded by as much as 50%. In this thesis, we propose modifications to IEEE 802.11 for simplifying the detection of such misbehaving hosts as well as for penalizing hosts detected to be misbehaving.

The rest of the thesis is organized as follows. A discussion of related work is presented in Chapter 2. The motivation for using MAC layer mechanisms to detect misbehavior, and a brief overview of the proposed protocol is outlined in Chapter 3. Details of the proposed protocol for detecting and penalizing misbehavior are presented in Chapter 4. Extensions to the proposed protocol for detecting receiver misbehavior, and improving diagnosis accuracy are in Chapter 5. Two models for misbehavior, and evaluation of the proposed protocol under the two models is presented in Chapter 6. Our conclusions and a discussion of future work is in Chapter 7.

CHAPTER 2

RELATED WORK

Most research addressing *selfish misbehavior* assume that selfish hosts misbehave primarily to improve their own performance (throughput, latency, energy, etc.). Selfish hosts are assumed to desist from attempting to degrade the performance of other hosts, when such an attempt does not improve their own performance. In contrast, research addressing wireless security are primarily focused on addressing *malicious misbehavior*, which is misbehavior aimed at disrupting normal network operation, possibly with no performance gain to the misbehaving host. Selfish misbehavior includes hosts that refuse to forward packets on behalf of other hosts to conserve energy, hosts that select small backoff values to obtain larger throughput share (the misbehavior that we address), etc. Malicious misbehavior includes denial of service attacks that disrupt routing operation, jamming the wireless channel to prevent communication, etc. Good discussions on various security issues in wireless networks are in [2, 3, 4].

Several approaches have been proposed for addressing selfish misbehavior at the network layer in wireless networks. One approach, proposed by Marti et al. [5], is to identify misbehaving hosts, and avoid using misbehaving hosts for network operations such as routing. Another approach, proposed by Buchegger et al. [6, 7], is to design protocols that encourage cooperation by penalizing misbehavior. A complementary approach, proposed by Buttyan et al. [8, 9], is to provide incentives for hosts to cooperate by using payment mechanisms. Network layer mechanisms address network layer misbehavior such as dropping, delaying or mis-routing packets. The scheme we propose addresses selfish misbehavior at the MAC layer, and can complement network layer mechanisms. However, we adopt techniques from network layer mechanisms, such as the approach of first detecting misbehavior, and then using suitable mechanisms for penalizing misbehavior. We incorporate these features into IEEE 802.11, to improve its resilience against misbehavior.

A related approach is to design protocols which are resilient to misbehavior. In the context of TCP, Savage et al. [10, 11] identify certain receiver misbehavior that may allow a misbehaving receiver to gain a throughput advantage over other well-behaved receivers, by exploiting weaknesses in the TCP congestion control algorithm used by the sender. Savage et al. 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.

Game-theoretic techniques have been used to develop protocols which are resilient to misbehavior. Game-theoretic approach assumes that all users are selfish and rational. Rational hosts always select a strategy that maximizes their utility (utility is a measure of the benefit obtained by a host). Protocols are designed that reach a equilibrium state called the "Nash equilibrium", where a selfish host cannot gain any advantage over well-behaved hosts. Game-theoretic approaches are well suited for designing protocols resilient to selfish misbehavior, and we discuss in detail below, some representative work.

Michiardi et al. [12] study mechanisms to address selfish misbehavior at the routing layer. They model the hosts in the network as participants in a non-cooperative game with each host attempting to maximize its own utility. By imposing suitable costs on each network operation such as packet forwarding, the game reaches a Nash equilibrium. In practice, selecting the right cost for each operation is hard. In addition, a pricing infrastructure must be available to ensure hosts pay for the services that they obtain.

Mackenzie et al. [13, 14] consider selfish misbehavior in Aloha protocol. Hosts are assumed to incur a cost for each transmission (e.g., energy required for the transmission), and each host is assumed to have perfect knowledge of channel conditions and backlogged hosts (in practice, this knowledge may not be available to hosts in the network). Under this setting, it is shown that the protocol has a Nash equilibrium. When all hosts follow the strategy proposed by MacKenzie et al., there is no scope for selfish misbehavior.

Konorski [15, 16] studies selfish MAC layer misbehavior, where hosts deviate from the specified backoff strategy. Konorski proposes a modified backoff algorithm using black-bursts, and with a game-theoretic analysis shows that the protocol is resilient to selfish misbehavior. Konorski's work assumes that all hosts can accurately measure the duration and originator of each black-burst, which is hard to guarantee in a wireless network.

Most of the protocols using game-theoretic techniques are based on the assumption of "Perfect Information", i.e., every host can observe all the actions of other hosts in the network. This assumption is hard to realize in practice, especially in the context of a wireless network (with fading channels, hidden terminals, etc.). In addition, protocols developed with game-theoretic techniques may not achieve the performance of protocols developed under the assumption that all hosts are well-behaved and cooperate with each other (e.g., IEEE 802.11). The scheme we propose retains the performance of IEEE 802.11 (a protocol based on cooperation among hosts), while ensuring detection of misbehavior.

Intrusion detection and tolerance techniques are used as tools for diagnosing and tolerating misbehavior [17, 18, 19, 20]. Intrusion detection approaches are based on developing a long-term profile of *normal* activities, and identify intrusion by observing deviations from the long-term 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).

This thesis is based on an extension of our earlier conference paper [21] on detecting and handling MAC layer misbehavior.

CHAPTER 3

OVERVIEW OF THE PROPOSED PROTOCOL

In this chapter we define the terminology used in the rest of the thesis. We then discuss the motivation for using MAC layer detection and penalty schemes, and state our assumptions in developing the proposed protocol. This is followed by a brief overview of the proposed protocol.

3.1 Terminology used in the thesis

The following terminology used in presenting the proposed protocol.

- **Sender:** Sender is a host which wants to transmit a data packet to another host.
- **Receiver:** Receiver is a host which receives a data packet from a sender host. The receiver monitors the sender host to detect sender's misbehavior.

Sender and *receiver* are the different roles a host can perform. A host 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 host transmits a data packet to a receiver host after an optional RTS-CTS exchange. In Chapter 5, we define extensions to the protocol, where the sender and receiver may be monitored by other nodes which are not directly involved in the communication between the sender and the receiver. We designate these nodes as *observers*.

3.2 Motivation and assumptions

The proposed protocol 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). In public wireless networks, 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 are untrusted, and may misbehave to gain higher throughput share than competing hosts. 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 protocol. We discuss mechanisms to address receiver misbehavior in Chapter 5. 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 protocol can also be applied to ad hoc networks (self organized networks without a central authority) to detect misbehavior as discussed later. The proposed protocol addresses selfish misbehavior (hosts intending to obtain higher throughput or lower delay), and does not consider malicious attacks such as jamming the channel.

3.3 Brief overview of the proposed protocol

The proposed protocol is designed to handle selfish MAC layer misbehavior in hosts using IEEE 802.11 DCF mode. A goal of the proposed protocol 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 between a well-behaved sender that legitimately selected a small random backoff, and 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 host 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 knows the exact backoff value sender is expected to use. Hence, the receiver can identify a sender 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 protocol also attempts to negate any throughput advantage that the misbehaving hosts may obtain. To achieve this, deviating senders are penalized thereby discouraging 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 hosts. Hence, with the proposed protocol, it is difficult for a misbehaving host to obtain an unfair share of the channel bandwidth while eluding detection.

CHAPTER 4

PROPOSED PROTOCOL

The proposed protocol has three components. First, 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 (*penalty scheme*). Last, based on the magnitude of the perceived deviation over *multiple* transmissions from the sender, the receiver identifies senders that are indeed misbehaving (*diagnosis scheme*). Extensions to the protocol for detecting receiver misbehavior, and improving detection accuracy are in Chapter 5.

4.1 Identifying deviations from the protocol

In the proposed protocol, hosts 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 S sends a packet to 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 4.1 depicts the receiver-sender interaction in the modified protocol. When the receiver R receives a

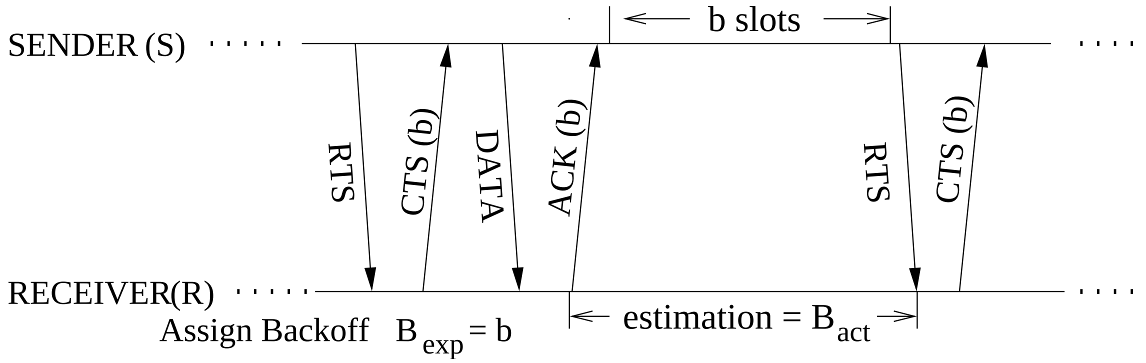


Figure 4.1 Receiver - Sender interaction in modified IEEE 802.11

RTS[1] from the sender S, R assigns a backoff value $B_{exp} = b$ to S in the CTS packet as well as the subsequent ACK packet as shown in Figure 4.1 (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 b 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 observes the channel status during the interval between the sending of an ACK by R, and the reception of the next RTS from S. The receiver notes down the length of this interval in slots, K, as well as the number of slots that were idle B_{act} during this interval. 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.,

$$B_{act} < \alpha * B_{exp} \quad , \quad 0 < \alpha \leq 1 \tag{4.1}$$

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

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

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 4.1 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 Chapter 4.3, for accurately diagnosing misbehaving hosts.

A threshold scheme, which compares the observed backoff B_{act} with a threshold (based on the assigned backoff B and total slots K) is optimal in maximizing detection percentage, subject to a maximum allowed misdiagnosis percentage (proof in Chapter 4.4). In this thesis, we use a constant fraction α of B as the threshold to simplify the protocol. Accurate selection of the threshold may require more information about channel parameters, which may not be available.

We now describe the extensions to IEEE 802.11 for handling sender misbehavior during packet retransmissions. Every RTS sent by the sender has an *attempt* number included in a new field in the RTS header. Sender sets the attempt number to 1 after a successful transmission, and increments it 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 4.2 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

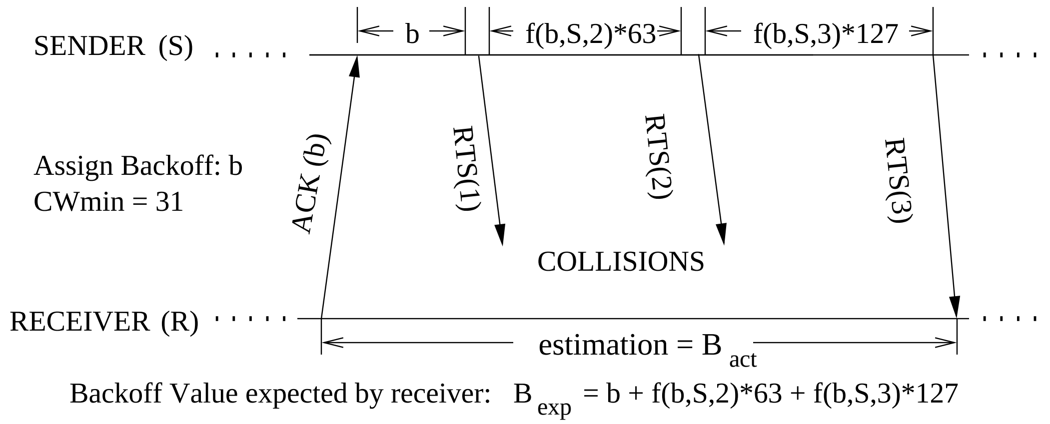


Figure 4.2 Protocol for retransmissions

transmission is unsuccessful, sender increments the attempt number, and chooses a new backoff value using a deterministic function f as follows:

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

where *backoff* is the backoff previously assigned by the receiver, *senderId* is the unique sender identifier, and *attempt* is the attempt number maintained by the sender. In Figure 4.2, *backoff*=b, *senderId*=S, and the *attempt* numbers are 1, 2 and 3.

The function f used by the sender for computing backoff values for retransmission attempt is given by :-

$$f(\textit{backoff, senderId, attempt}) = (aX + c)\ mod\ (CW_{min} + 1)$$

where $a = 5$, $c = 2 * attempt + 1$ and $X = (backoff + senderId)\ mod\ (CW_{min} + 1)$.
The function f generates a uniform random number between $[0, CW_{min}]$ and dividing the number by CW_{min} gives the required fraction between 0 and 1. The deterministic function f that we use has been carefully chosen (more details about function f is in a related technical report [22], and a good discussion on pseudo-random generation is in [23]) 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, applying the same deterministic function f used by the sender as,

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

where *attempt* is the attempt number in the received RTS, *backoff* is the backoff assigned to the sender by the receiver, *senderId* 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 4.1 as explained before. Note that if a deterministic function is not used by the sender, then the receiver cannot easily estimate the backoff value used by the sender after a collision.

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 hosts, or having smaller average *attempt* values than other hosts. 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. 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 Penalty Scheme

Hosts deviating from the protocol may obtain a larger throughput share than conforming hosts. The *penalty* scheme penalizes deviating hosts by assigning larger backoff values to them than those assigned to conforming hosts. We use the principle that hosts deviating more should be assigned larger penalties. Hence, when the receiver detects a deviation (using equation 4.1), 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 and simulations (details are in Section 4.5), we identified the need for additional penalty to effectively penalize the misbehaving hosts. 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 penalty 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 host from trying to adapt to any protocol parameters, and thereby obtain a throughput advantage over well-behaved hosts. 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 host. Our simulation results show that the average throughput obtained by well-behaved hosts when the *penalty* scheme is enabled 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) since a well-behaved host perceived as deviating for a packet may be perceived to backoff for larger than the expected backoff for some other packet. However, a persistently misbehaving host will have positive differences for most packets and is likely to be diagnosed. The choice of W and *Thresh* does not affect the *penalty* scheme. Hence, a sender adapting to W and *Thresh* will still have a penalty added for every perceived deviation, even if the host is not immediately 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 misdiagnosis (we defer adaptive selection to future work).

The *penalty* scheme is used to penalize potentially misbehaving hosts. However, the penalty scheme is not effective if a misbehaving host 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 host that backs off for a small fraction of the assigned penalty will be large, and the *diagnosis* scheme can identify such hosts with high probability. Thus, penalty and diagnosis schemes together ensure that a misbehaving host cannot obtain a larger than fair share of the bandwidth without being diagnosed as misbehaving.

After the *diagnosis* scheme identifies a host to be misbehaving, MAC layer may refuse to accept packets from the misbehaving host (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, hosts forward packets on behalf of each other. When misbehavior is diagnosed, network layer protocols may use the diagnosis information to route around misbehaving hosts. Network layer protocols can also refuse to forward packets originating from misbehaving hosts.

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

4.4 Proof of existence of optimal threshold

In this section, we show that comparing the observed backoff against a threshold, as proposed in the diagnosis scheme, is indeed optimal. The aim of the diagnosis scheme is to maximize the correct diagnosis probability, while restricting the misdiagnosis (i.e., false alarm) percentage below a specified threshold β. The diagnosis problem can be formulated under Neyman-Pearson hypothesis testing framework [24].

On receiving a packet from the sender, the receiver has to decide if the sender misbehaved. The information available at the receiver in diagnosing sender behavior is the status of the channel (idle or busy), in all the slots between the previous transmission from the sender and the current transmission, and characteristics of the channel. Using this information, the receiver has to choose one of the following two hypotheses:

- *Hypothesis H0*: Sender is well-behaved.
- *Hypothesis H1*: Sender is misbehaving.

We first prove under a general model for the channel that a scheme that compares the likelihood ratio (defined later) to a threshold is optimal for maximizing correct diagnosis probability. We then present a simple channel model, and demonstrate that for this model it is sufficient to compare the observed backoff (instead of the likelihood ratio) against a threshold.

Both the sender and the receiver observe the channel. The channel in each slot may either be *idle* (I) or *busy* (B). The sender is assigned a backoff value B by the receiver. Say the sender waits for K slots on the channel (both idle and busy slots), before transmitting a packet. The channel conditions observed by the sender can be represented by a string s of length K, derived from an alphabet {I,B}, with I (B) at position i in the string indicating slot i was observed idle (busy) by the sender. The receiver observes a string s_o (possibly different from s because of varying channel conditions), also of length K. The receiver has to select, based on the assigned backoff B, and the observed string s_o, the more likely hypothesis. This can be extended to a more general model which uses the past history of observed strings in addition to the newly observed string.

If hypothesis H0 is true, sender will have selected the string s from a set S_G (set of good strings) that have at least B idle slots. Otherwise, the string s will be selected from a set S_B (set of bad strings) that have less than B idle slots. The channel is specified using a conditional probability distribution (P_C) of observing a string s at the sender when string s_o is observed at the receiver.

Under the above model, we can derive the probabilities for hypothesis H0 and H1 being true, given the observation s_o as

$$\text{P(H0 is true given } s_o \text{ was observed)} \quad P_{H0} = \sum_{s \in S_G} P_C(s/s_o)$$

$$\text{P(H1 is true given } s_o \text{ was observed)} \quad P_{H1} = \sum_{s \in S_B} P_C(s/s_o)$$

The likelihood-ratio L is defined as:

$$L = \frac{P_{H1}}{P_{H0}}$$

Applying the Neyman-Pearson theorem [24], the optimal decision rule for maximizing the probability of diagnosis, while restricting the misdiagnosis percentage below a specified value β, is of the form:

- Choose hypothesis H1 if $L > T$
- Choose hypothesis H1 with probability γ if $L = T$
- Choose hypothesis H0 if $L < T$

where the threshold T, and probability γ can be computed from β and distributions of P_{H1} and P_{H0} as shown in [24].

Next, we use a simple channel model and prove that for this model, comparing observed backoff against a threshold is optimal. We assume that the receiver has a probability p of sensing a slot, observed by the sender as idle, to be busy. On the other hand, when the sender senses a slot to be busy, the receiver too senses the slot to be busy. Such a model approximates a scenario in practice, when the carrier sensing threshold of IEEE 802.11 at the receiver is chosen to be significantly lower[2] than the receive threshold. In this scenario, receiver senses most of the transmissions sensed by the sender, but may also sense some other transmissions that are not sensed by the sender (as is the case in the model).

In the proposed protocol, the receiver counts the total number of slots K between two transmissions by the sender, as well as the number of idle slots B_o observed on the channel in that interval. The receiver is also aware of the backoff B that it had assigned to the sender.

We assume that a well-behaved sender waits for at least B idle slots, before transmitting a packet (alternatively, we could model a well-behaved sender to wait for exactly B idle slots, and the same analysis as below applies for that scenario also). Under the assumed channel model, the number of idle slots observed by the receiver is less than or equal to that observed by the sender. Thus, when $B_o \geq B$, the sender has waited for at least B slots and is well-behaved. On the other hand, if $B_o < B$, we can apply the general formula derived earlier (replacing the observed event as number of idle slots, instead of a specific string s_o). As a result P_{H0} can be computed as,

$$
\begin{aligned}
P_{H0} &= 1, \quad B_o \geq B \\
P_{H0} &= \sum_{i=B}^{K} P_C(i/B_o), \quad B_o < B
\end{aligned}
$$

[2]Lower value of threshold implies that signals received with lower power are detected, enabling the sensing of further away transmissions

$P_C(i/B_o)$ is the probability that the sender waited for exactly i idle slots (out of K total slots) given that the receiver observed B_o idle slots. Similarly, we can compute P_{H1} as,

$$\begin{aligned} P_{H1} &= 0, \quad B_o \geq B \\ P_{H1} &= \sum_{i=B_o}^{B-1} P_C(i/B_o), \quad B_o < B \end{aligned}$$

The likelihood ratio L is 0 for $B_o \geq B$, and for $B_o < B$ is given by

$$\begin{aligned} L &= \frac{P_{H1}}{P_{H0}} \\ &= \frac{1}{P_{H0}} - 1, \quad \text{(because } P_{H0} + P_{H1} = 1\text{)} \\ &= \frac{1}{\sum_{i=B}^{K} P_C(i/B_o)} - 1 \end{aligned}$$

When B_o increases, $P_C(i/B_o)$ increases if the probability p of receiver sensing a slot, observed to be idle by the sender, to be busy is less than 0.5. When B_o increases, we are conditioning on receiver making fewer mistakes ($i - B_o$ is the number of mistakes), and this probability increases whenever $p < 0.5$. Under this condition, the likelihood ratio L is monotonically decreasing with increasing B_o (until L reaches 0). Therefore, in the Neyman-Pearson rule, instead of comparing L against a threshold T, we can compare B_o with a new threshold T', suitably derived from T.

Hence, the optimal decision rule is,

1. The sender is misbehaving if $B_o < T'$
2. The sender is misbehaving with some probability γ, if $B_o = T'$
3. The sender is well-behaved if $B_o > T'$

Note that the threshold itself is some function of B and K, though in the proposed protocol we choose a constant fraction (α) of B as the threshold for simplicity.

4.5 Analysis of required additional penalty

In this section, we identify the need for additional penalty in the penalty scheme to effectively penalize misbehaving hosts, and describe a mechanism for estimating the additional penalty. On simulating the approach of adding measured deviation as penalty, we found that the misbehaving hosts still obtain a bigger throughput share than conforming hosts, although the share obtained by misbehaving hosts is far lesser than what they would have obtained without any correction. Misbehaving hosts obtain a higher throughput share because they suffer lesser number of collisions per successful transmission compared to well behaved hosts. Note that the expected backoff B_{exp} is computed based on the number of attempts made by the sender host before successfully transmitting. However, the misbehaving hosts by backing off for lesser duration on an average, suffer fewer collisions per successful transmission. To illustrate this, consider a network containing only two hosts A and B. Host A misbehaves by transmitting packets at a higher rate than the well-behaved host B (assume that the channel capacity is greater than the transmission rate of either of them). Then, the total collisions suffered by both the hosts is the same, but the number of collisions per successful transmission is less for the misbehaving host as it transmits more often. So, the correction scheme should also penalize the misbehaving hosts for the benefit gained from having fewer collisions per successful transmission.

We analytically estimate (and thereby demonstrate) the fewer number of collisions that a misbehaving host suffers in comparison to a well-behaved host. We model the channel access behavior of hosts by *access probabilities*. We follow the approach used by Cali *et al.* [25] in our analysis. As mentioned while presenting our backoff protocol, the average expected backoff for our modified scheme is the same as that of IEEE 802.11 and hence we can apply the approach of Cali *et al.* We assume that at any instant there are N *active* hosts in the network (N varies with time). Active hosts are those that have a packet to send to the base station at the instant under consideration. Each conforming active host accesses the channel every idle slot with a probability p (p is the parameter of a geometric distribution). Thus, in any idle slot a conforming active host has a probability p of transmitting a packet. In reality, the access probabilities depend on the *Contention Window* value which in turn depends on the number of collisions that a host suffers while transmitting a packet. In [25] the access probabilities are assumed to be inversely proportional to the average contention window size. These assumptions are reasonably accurate for sufficiently large number of hosts in the network. It has been shown in [25] that the results are fairly accurate even for small networks containing around 10 hosts.

We analyze the behavior of a misbehaving host, designated as MB, that accesses the channel with probability p_{mb} such that $p_{mb} > p$. For the sake of this analysis, we assume that the rest of the hosts in the network are well-behaved. When MB transmits a packet, the transmission is successful if no other host attempts to transmit a packet in that slot. Thus, the probability of an attempted transmission being successful is given by

$$P(\text{attempted transmission successful})\ ,\ P_{succ} = (1-p)^{N-1}$$

The probability that an attempted transmission is unsuccessful is given by

$$P(\text{attempted transmission fails})\ ,\ P_{fail} = 1 - P_{succ}$$

Assuming that each retransmission attempt is independent, the probability that exactly i^{th} attempted transmission is the successful is given by

$$P(\text{exactly } i^{th} \text{ attempt successful }) = P_{fail}^{i-1}\ P_{succ}$$

Hence the expected number of attempts needed for a successful transmission for misbehaving host MB is (using standard algebraic identities)

$$E(\text{attempts for MB})\ ,\ E_{mb} = \frac{1}{P_{succ}} = \frac{1}{(1-p)^{N-1}}$$

Using a similar analysis, the expected number of attempts needed for a successful transmission for a conforming host is

$$
\begin{aligned}
E_{conf} &= \frac{1}{P(\text{success for conforming host})} \\
&= \frac{1}{(1-p_{mb})(1-p)^{N-2}}
\end{aligned}
$$

Since the misbehaving host backs off for some fraction of its assigned backoff value, its access probability is higher than that of the conforming hosts. When $p_{mb} > p$, $\frac{1}{1-p_{mb}} > \frac{1}{1-p}$. Hence, using this in the equations above we see that the conforming hosts suffer more collisions per successful transmission than the misbehaving host. The difference in number of collisions must be accounted in the misbehavior correction mechanism. This is given by

$$
\begin{aligned}
E_{diff} &= E_{conf} - E_{mb} && (4.2) \\
&= \frac{1}{(1-p_{mb})(1-p)^{N-2}} - \frac{1}{(1-p)^{N-1}} && (4.3) \\
&= \frac{1}{(1-p_{mb})(1-p)^{N-1}} * (p_{mb} - p) && (4.4)
\end{aligned}
$$

The next step is to compute this value. In practice, when we are computing the penalty to be added, we know that B_{act} (the actual time the host under observation has backed off for) is smaller than $\alpha * B_{exp}$ (B_{exp} is the expected backoff). By our model, the access probability of a host is inversely proportional to its average contention window value. When the misbehaving host waits for a fraction of its assigned backoff value, this behavior is equivalent to the misbehaving host waiting completely for a backoff value which is selected from a smaller range (in comparison with other conforming hosts). The average contention window value (denoted as ACW) can be computed as the ratio of the average total backoff per successful transmission (denoted as ATB) to the average number of attempts per successful transmission. Hence,

$$p \propto \frac{1}{ACW}$$

where average contention window ACW is computed as,

$$ACW = \frac{ATB}{\text{Average number of attempts}}$$

We expect the average number of attempts for a successful transmission (in the long run) to be the same for misbehaving hosts and well-behaved hosts when the *correction* scheme is enabled. Our aim is to compute the benefit gained by the misbehaving host for the observed deviation from protocol. If the misbehaving host persists with misbehavior, then its average total backoff will be proportional to B_{act} while the average total backoff (ATB) of well-behaved hosts will be proportional to B_{exp}. Assuming persistent misbehavior, the ratio of access probabilities of well-behaved host to misbehaving host is given by,

$$\frac{p}{p_{mb}} = \frac{B_{act}}{B_{exp}}$$

We denote this ratio of observed backoff interval to expected backoff interval by β. Hence $p = \beta\, p_{mb}$. For accurate computation of the ratio of access probabilities we need to use the average total backoff. However, we make the above approximations to obtain an approximate estimate of the ratio of access probabilities using the observed host behavior. Using this substitution, we can rewrite equation 4.4 as,

$$E_{diff} = \frac{p_{mb}}{(1-p_{mb})(1-p)^{N-1}} * (1-\beta)$$

The term $(1-p_{mb})(1-p)^{N-1}$ is the probability that a slot is idle. By using the correction scheme, we expect the access probability of the misbehaving host in the long run to be roughly equal to the access probability of conforming hosts, thus $p_{mb} \approx p$. Using this approximation, the number of collisions that the misbehaving host has saved by backing off for less than the assigned value is given by,

$$E_{diff} \approx \frac{p * (1-\beta)}{P(\text{Slot is idle})} \tag{4.5}$$

This is one of the several possible ways of estimating E_{diff} starting from equation 4.4. We can directly estimate p and N using the approach of Cali *et al.* in [25] and compute E_{diff} from that. However, our approach is simpler to implement.

We use a *idleCounter* (informally described in section 4.1) for estimating the idle probability. The *idleCounter* is incremented by 1 for every idle slot and is frozen when the channel becomes busy. The number of idle slots between two successive transmissions i and $(i+1)$ on the channel is calculated as the difference X in *idleCounter* values at the end of the i^{th} transmission and at the end of $(i+1)^{th}$ transmission. In our analysis, each transmission is assumed to be 1 slot in length (the idle slot in which a host accesses the channel and begins a transmission) irrespective of the length of the transmission (a single transmission may involve exchange of RTS-CTS-DATA-ACK packets). Hence, the total number of slots between i^{th} and $(i+1)^{th}$ transmission is $X+1$.

Idle probability can be approximated to be equal to the ratio of number of *idle slots* between two successive transmissions to the total number of slots between two successive transmissions. Hence, the idle probability is given by,

$$P(\text{Channel idle}) = \frac{X}{X+1}$$

where X is the number of idle slots between successive transmissions.

The average idle probability P_{idle} is estimated using the idle probability computed for each transmission and a smoothening factor δ as follows

$$P_{idle} = \delta * P_{idle} + (1-\delta) * P(\text{Channel idle})$$

where $0 < \delta < 1$ and P_{idle} is initialized to 1.

As discussed earlier, the average access probability p is estimated as the reciprocal of the average number of slots a host waits per transmission attempt ACW (including one slot for the attempted transmission itself). Thus,

$$p = \frac{1}{\text{ACW}+1}$$

where the average contention window ACW is calculated as,

$$ACW = \frac{\text{Average expected backoff}}{\text{Attempts per successful transmission}}$$

The average expected backoff is calculated by the receiver based on the backoff values that the receiver has assigned. The average number of attempts per successful transmission is calculated from the *attemptNumber* field of RTS packets received from the senders.

Using the calculated values of p, β and P_{idle}, E_{diff} is computed by substituting these values in equation 4.5. Once we have estimated E_{diff}, we compute the additional penalty as the number of slots the misbehaving node would have additionally backed off for, if it had actually suffered E_{diff} additional collisions.

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

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 *senderId* is the identifier of the misbehaving node.

CHAPTER 5

EXTENSIONS TO THE PROTOCOL

In this chapter, we propose extensions to the base protocol described in Chapter 4, for handling receiver misbehavior, improving diagnosis accuracy, and utilizing multiple observers.

5.1 Handling receiver misbehavior

In the proposed scheme, there exists a possibility that the receiver may misbehave in assigning backoff values. As discussed before, in the case of infrastructure-based networks, base station is trusted, and is not expected to misbehave. However, in the case of ad hoc networks, receivers cannot be trusted, and may misbehave.

One receiver misbehavior is assigning small backoff values to a preferred sender to receive data from that sender at a higher rate. This type of attack is possible, say, when the receiver is expecting some data from a particular sender, and seeks to obtain that data with minimal latency.

This 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 assigning small backoff values. Senders can choose to use larger of the backoff assigned by the receiver, and the backoff expected by the sender. This solution is not sufficient in case the sender and the receiver collude. Mechanisms to address collusion are discussed later.

An alternate approach is based on the observation that receiver assigns the backoff values to the sender to enable the receiver to know the exact backoff sender is expected to count down. The same goal can be achieved by requiring the sender to publish its backoff for next transmission, with the constraint that these values have to be picked using a well-known deterministic function g. This is similar to the approach used in SEEDEX protocol [26], where senders inform receivers the transmission schedule by publishing the seed of the random number generator. When the receiver gets the schedule, it has to first verify that the published schedule has been legitimately chosen, and then has to verify for each transmission whether the sender counted down the required backoff using the approach described earlier. The drawback of using this approach is that receiver can no longer punish misbehaving senders using a simple penalty mechanism as we proposed earlier. Receivers can just drop packets from potentially misbehaving senders, but if packets are dropped from well-behaved senders, it may drastically degrade their throughput (e.g., if TCP is being used, TCP may timeout to recover from lost packets, leading to severe degradation in throughput). Hence, dropping packets, in contrast to adding additional backoff as penalty, may have a detrimental effect on throughput of misdiagnosed well-behaved hosts.

Another receiver 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.

5.2 Reducing misdiagnosis

In IEEE 802.11, the channel is said to be busy in a slot in the following cases:

- When that slot has been reserved by a RTS or a CTS, or a host is receiving a packet[1].
- When the strength of the received signal on the channel (including noise and interference) is above a threshold called the "Carrier Sense threshold" (even if the packet is not decoded correctly). Carrier sense threshold is often chosen such that the maximum distance from which a transmission can be sensed, called the carrier sense range, is approximately twice the distance from which a packet can be correctly received. This enables a host to sense transmissions originating from its two-hop neighbors, and avoid colliding with them. (More details are in IEEE standard [1].)

We identify a scenario where misdiagnosis may occur, and propose a solution. In Figure 5.1, bold lines connecting two hosts indicate that they can receive packets from each other. Dashed lines indicate that the two hosts can sense each other's transmission, but cannot successfully receive packets. In Figure 5.1, sender S attempts to communicate with the receiver R. Transmissions from host 1 in the vicinity of R can be sensed by S, but packets from host 1 cannot be received at S. R can only sense transmissions from host 2, while S cannot even sense transmissions from host 2. Hence, host 2 is hidden from S, but not from R.

In this scenario, when host 2 starts a transmission, receiver R senses the channel to be busy, but the sender S does not. Later, when a packet is received from S, R may decide that S did not backoff for sufficient number of idle slots (this is a problem arising out of "hidden terminals" [27]), and may conclude that S is misbehaving. This leads to misdiagnosis. The protocol conservatively chooses parameters to reduce the incidence of

[1] A packet is being received on the channel, when the preamble sent by the sender has been correctly decoded.

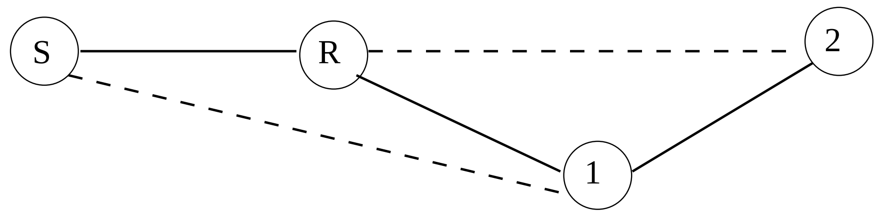


Figure 5.1 Scenario where misdiagnosis occurs

misdiagnosis, but the misdiagnosis percentage may be high if the above scenario persists.

We propose an optional modification to the proposed scheme to address this problem. Observe that the problem illustrated above would not have arisen, if R had counted the slots in which host 2 was transmitting to be idle. So, if the receiver classifies a slot to be busy only when an overheard RTS/CTS has reserved the slot, or a packet is being received (and not when only sensing some transmission but not receiving anything correctly), then the receiver in most cases will identify a slot to be idle whenever the sender senses a slot as idle. This will reduce the incidence of misdiagnosis. Note that senders are still required to count a slot as busy if they sense a transmission, as in IEEE 802.11. Furthermore, this modification is used to decide if a sender is misbehaving. If the sender is classified as misbehaving, then the penalty is added based on a counter that counts busy slots as before (even slots where transmissions are just sensed are counted as busy). This ensures that appropriate penalty will be added once a sender is diagnosed to be misbehaving.

With this modification, the receiver may now classify some slots to be idle, when the sender actually senses the slots as busy (e.g., when the sender and receiver are reversed in Figure 5.1). Misbehaving hosts aware of this modification may try to intelligently misbehave, and leverage the conservative behavior of the receiver. However, a misbehaving sender cannot decide with certainty, whether the receiver has classified a busy slot as idle, and thus, does not have a guaranteed strategy for obtaining better performance.

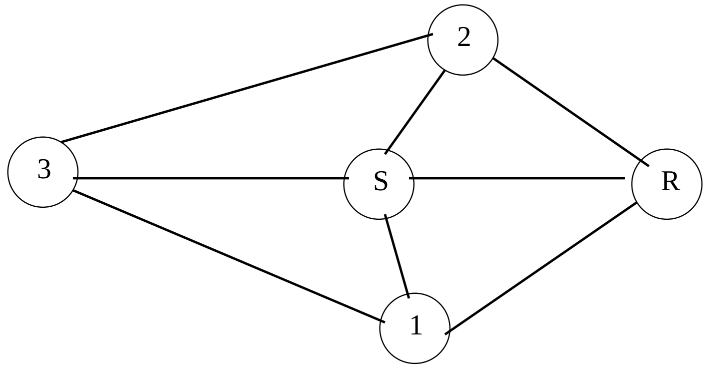


Figure 5.2 Using multiple observers to improve diagnosis accuracy and to detect collusion

We evaluate the impact of this modification in Chapter 6.6.

5.3 Using multiple observers

In the discussions so far, we have required the receiver to monitor sender behavior. We can easily extend the protocol to allow other hosts in the vicinity of the sender to monitor sender behavior. For example, in Figure 5.2, host 1 can receive packets from both the sender S and the receiver R. When host 1 receives a CTS from R intended for S, host 1 knows the backoff assigned to S. Later, when S sends a packet, host 1 can decide if the sender S waited for the assigned number of idle slots. These additional observers can also be used to detect collusion. For example, host 1 can monitor the backoff values assigned by the receiver R, and decide if R is assigning small backoff values to any particular sender. In addition, host 1 can also monitor the sender S, and verify if a receiver is correctly punishing sender misbehavior (otherwise, a receiver may intentionally ignore misbehavior of a preferred sender).

Multiple observers can also be used to improve diagnosis accuracy. In an infrastructure-based network, the observers may be neighboring base stations, or other specially installed monitoring hosts. In ad hoc networks, observers may belong to a common trust group, and can share diagnosis information (e.g., hosts 1, 2, 3 in Figure 5.2). As discussed before, misdiagnosis incidence may be high in certain scenarios. But, when multiple observers are used, not all observers incorrectly diagnose misbehavior. So, intelligently combining information from multiple observers can reduce misdiagnosis percentage. Similarly, we can improve the correct diagnosis percentage as well. We defer for future work analysis of scenarios where multiple observers are beneficial, and strategies to be used for combining information from multiple observers (a simple strategy is to take a majority vote, but more sophisticated strategies may be developed using signal processing techniques). Another question of interest for infrastructure-based networks is the locations where observers need to be placed to maximize diagnosis accuracy.

CHAPTER 6

SIMULATION RESULTS

We use the ns-2 [28] 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 [28]. 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 [28]. 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. For example, the carrier sense (receive) threshold is set to the received signal strength at 550m (250m) when X_{db} is set to 0 in the above equation (we assume all senders use the same transmit power).

In our simulations, all the sender hosts 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 for each run 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. Hosts are stationary in all simulations.

6.1 Simulation topology and simulation metrics

We first simulate our proposed protocol for a network having a well-behaved receiver R, and multiple senders transmitting to R. We use this simple network setting to simplify the evaluation of the proposed protocol's effectiveness in handling sender misbehavior, and identify the various trade-offs 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 chapter, simulation results for multiple senders and receivers randomly placed in the network.

Figure 6.1 shows the simulated network. The number of sender hosts around the receiver R is 8 (numbered 1 through 8 in the figure) with host 3 misbehaving. The 8 sender hosts are placed in a circle of radius 150 meters around R, equidistant from each other. There are 4 other hosts 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 6.1. 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 hosts do *not* sense these transmissions with high probability. For example, in Figure 6.1, when A sends a packet to B, host 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:

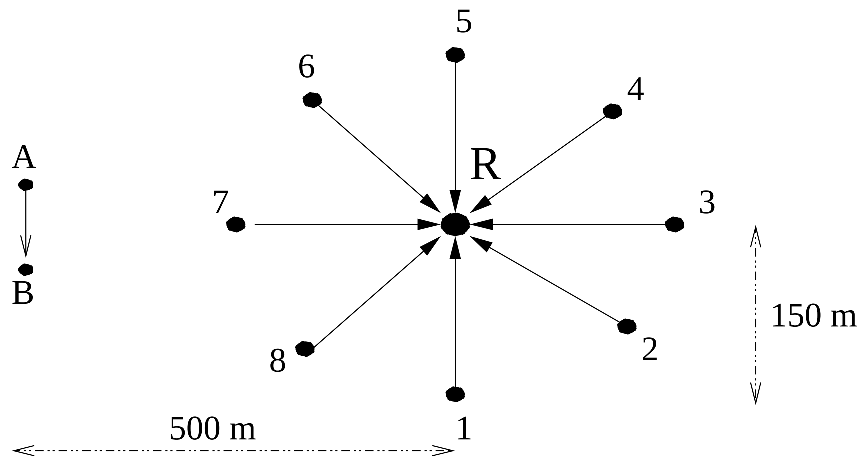


Figure 6.1 Simulation setup

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. This models the case when background traffic is small.

2. ONE-FLOW: In this scenario, only traffic flow C-D is turned on. In this scenario, receiver R and hosts 1 through 5 sense the transmission from C to D with high probability, while hosts 6, 7, and 8 do not. Consequently, hosts 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 host 3 to be the misbehaving host. Hence, this scenario tests our protocol performance when host 3 is actually misbehaving, while hosts 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).

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 Chapter 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 hosts:* This is the average throughput per well-behaved sender.

4. *Misbehaving host throughput:* This is the average throughput per misbehaving sender.

6.2 Misbehavior Models

We evaluate our protocols under two misbehavior models that capture the behavior of a misbehaving host. The first model, termed the "*Persistent Misbehavior Model*", captures the behavior of a misbehaving host that always misbehaves using a fixed strategy. In this model, we characterize various levels of misbehavior, with a parameter called "Percentage of Misbehavior" (PM). A misbehaving host with PM=x% transmits a packet after counting down to (100-x)% of the assigned backoff value. The PM parameter is used to quantify the *magnitude* of misbehavior, with larger values of PM indicating greater misbehavior. Hence, a host with PM=0% fully counts down the assigned backoff and is well-behaved, whereas a host with PM=100% transmits a packet without counting down any backoff at all. Misbehaving hosts persist with this behavior irrespective of the magnitude of assigned backoff. This captures the behavior of a host which misbehaves without adapting to the assigned penalties. Although, this is a simple misbehavior model, we use this model for for most simulations to simplify the evaluation.

The second model we use, termed as "*Adaptive Misbehavior Model*", captures the behavior of a misbehaving host which changes the magnitude of misbehavior based on the magnitude of penalty assigned by the receiver. This is intended to model a misbehaving host which tries to obtain a higher throughput share without getting caught. This model also uses a variable called "Percentage of Misbehavior" (PM) that is used to decide what fraction of the assigned backoff a host will wait for, as described for the persistent misbehavior model. However, the value of this variable is changed based on the penalty assigned by the receiver. If the receiver assigns a penalty, misbehaving sender decreases its PM, and in the absence of a penalty, PM is increased. Intuitively, the misbehaving host tries to misbehave more when the receiver cannot diagnose the misbehavior (and therefore does not assign a penalty). In this model, PM is initially set to 0. For every packet transmission not assigned a penalty by the receiver (penalty is said to be not assigned, if assigned backoff is not greater than CW_{min}), the value of PM is increased by a additive factor called "Additive Increase Percentage". If a penalty is assigned, then the value of PM is reduced by half (multiplicative decrease). In our evaluations, we vary the "Additive Increase Percentage" from 1% to 20% to model varying aggressiveness of misbehavior.

6.3 Results for protocol performance in the absence of misbehavior

We first evaluate the performance of our protocol (without the extensions proposed in Chapter 5) in the absence of 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, to characterize the effect of occasionally penalizing well-behaved hosts. Our evaluation indicates that the average throughput, as well as the fairness of the proposed scheme, is comparable to that of IEEE 802.11 in the absence of misbehavior.

The number of senders communicating with the receiver R is varied from 1 to 64 (replacing the 8 senders in Figure 6.1). All senders are well-behaved. Figure 6.2 compares the average throughput obtained by hosts when using IEEE 802.11 (curve "802.11") with that obtained when using the proposed scheme (curve "CORRECT") for varying network sizes under ZERO-FLOW, ONE-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 6.2). Hence, the penalty scheme does not degrade the aggregate throughput of the network.

We are also interested in comparing the fairness properties of the penalty scheme with that obtained using IEEE 802.11. We use Jain's Fairness Index [29] as a fairness metric, 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 host and receiver R), and N is total the number of flows. Fairness index values closer to 1 indicate better fairness. Figure 6.3 compares the fairness index of IEEE 802.11 and the penalty scheme for varying network sizes under ZERO-FLOW, ONE-FLOW and TWO-FLOW scenarios. For the ZERO-FLOW scenario, the fairness index of penalty scheme is comparable to that of IEEE 802.11. For the ONE-FLOW and TWO-FLOW scenarios, the fairness index of our scheme is slightly lesser that that of 802.11. This indicates that our scheme degrades the throughput of some senders minimally, while increasing the throughput of some other senders, since the average throughput (Figure 6.2) is the same as in IEEE 802.11. This is because, in ONE-FLOW and TWO-FLOW scenarios, a few sender hosts

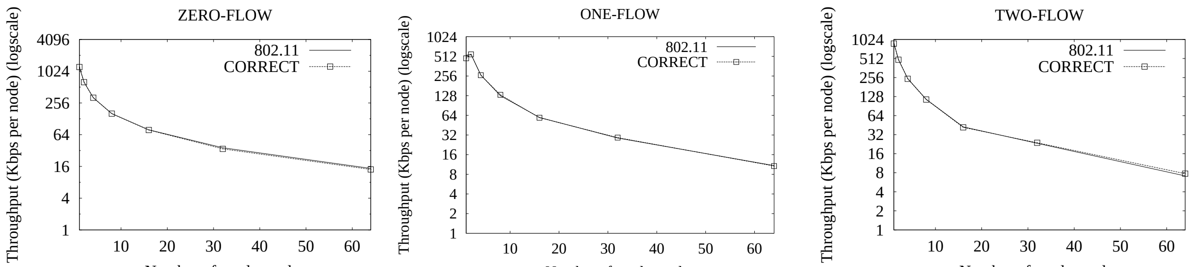


Figure 6.2 Throughput comparison without misbehavior for varying network sizes

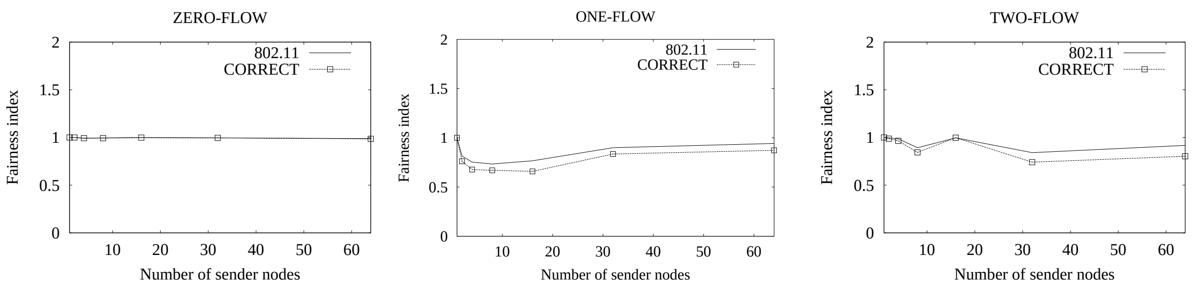


Figure 6.3 Comparison of fairness index between IEEE 802.11 and proposed scheme

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 that is still close to that of 802.11.

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

6.4 Results for persistent misbehavior model

In this section, we evaluate our proposed scheme (without the extensions of Chapter 5), under the persistent misbehavior model. The protocol parameters *W* and *Thresh* are used by the *diagnosis* scheme to identify misbehaving hosts, and α is used by the *penalty* 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 so as to reduce misdiagnosis. 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.

6.4.1 Diagnosis Accuracy

Figure 6.4 plots the correct diagnosis percentage and misdiagnosis percentage for the ZERO-FLOW, ONE-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%. Since we have conservatively selected the *Thresh* parameter (host is designated as misbehaving only when the total deviation for previous *W*=5 packets is greater than *Thresh*=20 slots), the correct diagnosis percentage is small when the extent of misbehavior is less (however, with the benefit of low misdiagnosis percentage). As the misbehavior increases, the observed deviation rises above *Thresh*=20 slots, and there is a rapid increase in the correct diagnosis percentage.

Similarly for the ONE-FLOW scenario, the correct misbehavior percentage is quite high. In the ONE-FLOW scenario, flow C-D is on, resulting in misdiagnosis of some hosts,

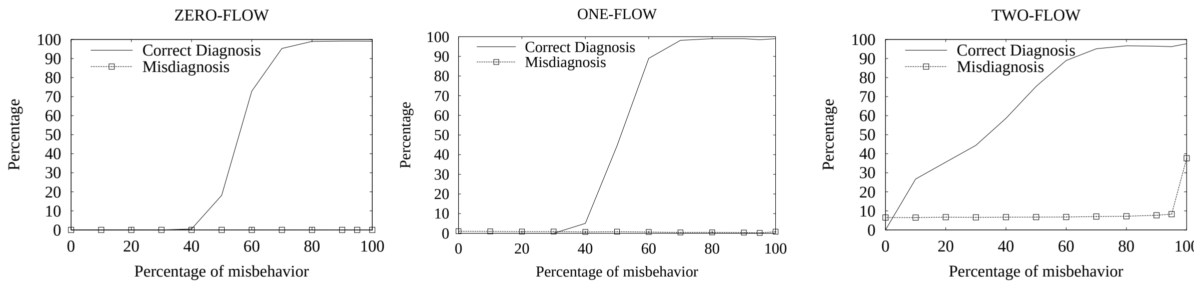


Figure 6.4 Diagnosis accuracy for varying magnitude of misbehavior

but the misdiagnosis percentage is small because our choice of protocol parameters is conservative.

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 an increase in the magnitude of the observed deviations over the ZERO-FLOW scenario. As a result, 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 trade-off involved in achieving low misdiagnosis percentage versus achieving high correct diagnosis percentage.

6.4.2 Throughput in the presence of misbehavior

Figure 6.5 compares the throughput obtained by a misbehaving host (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 host 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 host ("CORRECT - MSB" curve) is restricted to its fair share (except when PM is close to 100%), while for

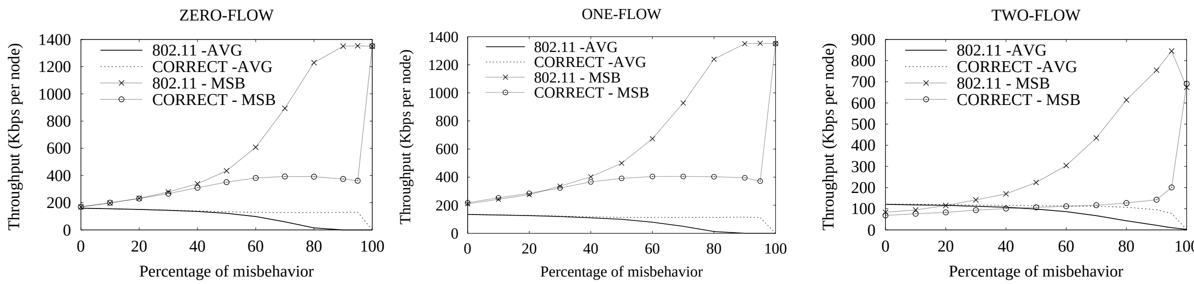


Figure 6.5 Throughput comparison between IEEE 802.11 and proposed scheme

802.11 ("802.11 - MSB" curve), the misbehaving host obtains a large throughput share even when extent of misbehavior is not too high. In addition, the throughput of well-behaved hosts 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 hosts using IEEE 802.11 ("802.11 - AVG" curve) starts degrading even when extent of misbehavior is not too high. Hence, the proposed penalty scheme is fairly successful in ensuring reasonable throughput for well-behaved hosts, in the presence of misbehaving hosts. When PM is close to 100%, the misbehaving host backs off for a small fraction of the assigned backoff, and consequently the proposed scheme cannot restrict the throughput of the misbehaving host. However, we can see from Figure 6.4 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 host misbehavior (as discussed in Chapter 4.3).

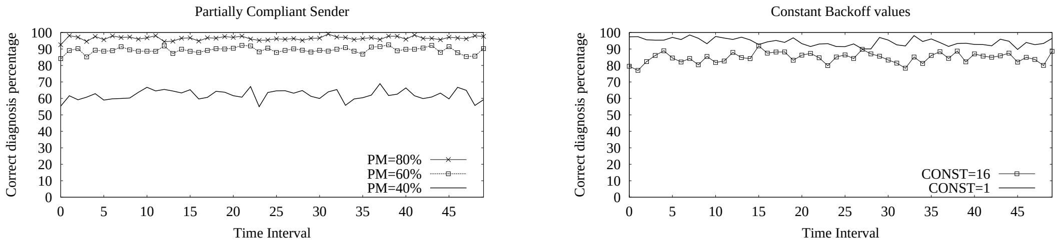
6.4.3 Responsiveness of Diagnosis Scheme

Figure 6.6(a) shows the variation of correct diagnosis percentage with time, measured using the TWO-FLOW scenario, for a persistent misbehaving sender with PM% misbehavior. 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. As seen from Figure 6.6(a) 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.

Figure 6.6(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 models a misbehaving sender waiting for exactly 1 idle slot before transmitting a RTS (irrespective of the backoff assigned by the sender), and similarly CONST=16 models a misbehaving sender 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 penalty scheme increases when a misbehaving host waits for a very small fraction of the previously assigned backoff. A host continuing to misbehave can be easily diagnosed as misbehaving, since the measured deviations become large. Thus, the prediction accuracy rapidly reaches a high value.

6.4.4 Protocol performance with random topologies

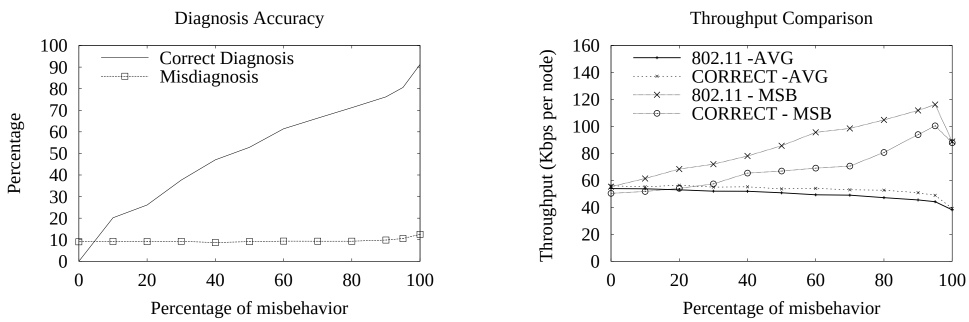
Figure 6.7 compares the protocol performance for 30 different random topologies. 40 hosts are placed at random locations in a 1500m by 700m area. 5 hosts, selected at random, are misbehaving. Each host sets up a CBR connection with one of its neighbors, and the connections are always backlogged. Figure 6.7(a) plots the correct diagnosis percentage and misdiagnosis percentage for different PM (Percentage of Misbehavior)



(a) Partially compliant sender

(b) Non-conformant sender

Figure 6.6 Evaluation of responsiveness of misbehavior diagnosis scheme



(a) Diagnosis Accuracy

(b) Throughput comparison

Figure 6.7 Protocol performance for random topology with 40 hosts in 1500m X 700m area

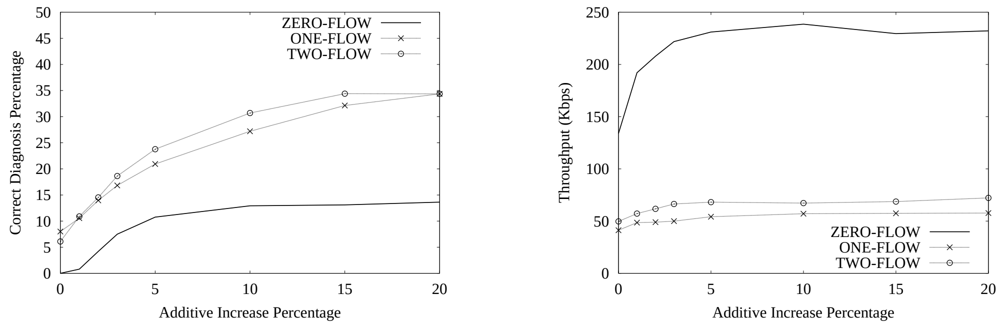
values. As we can see from the figure, the correct diagnosis percentage is high when extent of misbehavior is large, and the misdiagnosis percentage is reasonably small across all values of PM. Figure 6.7(b) compares the throughput obtained by the misbehaving hosts and the average throughput, for IEEE 802.11 and the proposed penalty scheme (the notations used are those described earlier for Figure 6.5). When the extent of misbehavior is small, the penalty scheme is fairly successful in restricting the misbehaving hosts to a fair share, thereby ensuring that the throughput of well-behaved hosts are not affected. When the extent of misbehavior is large, the penalty scheme is not as successful, but the misbehavior is diagnosed with high probability (as seen from Figure 6.7(a)).

6.5 Results for adaptive misbehavior model

We have proposed the penalty scheme as a mechanism to discourage misbehavior. By adding a penalty, hosts that ignore the penalty have higher probability of being detected. In this section, we evaluate the benefit a misbehaving host can obtain, while not ignoring the assigned penalty. We assume that the misbehaving host follows the adaptive misbehavior strategy defined earlier.

Figure 6.8(a) plots the correct diagnosis percentage when the misbehaving host uses an adaptive misbehavior strategy. As we can see from the figure, the misbehaving host can evade diagnosis most of the times (low diagnosis accuracy) in all three flow scenarios, but the accuracy improves when the misbehaving host uses an aggressive misbehavior policy (large additive increase percentages).

The low diagnosis accuracy has to be compared with the throughput obtained by the misbehaving host, shown in Figure 6.8(b). In the figure, for the three curves, the value when Additive Increase Percentage is equal to 0 is the throughput the misbehaving host would have obtained if it was well-behaved. The ZERO-FLOW curve starts at a higher value than the other two curves, because in the ZERO-FLOW scenario there is no background traffic, allowing all hosts to obtain higher throughput. For each of the curves, with increasing additive increase percentage, there is an increase in the throughput, till a saturation level is reached. This indicates that the misbehaving host gains some throughput advantage, but the benefit is not significant, as can be seen from the figure. This throughput gain is obtained because the penalty added by the penalty scheme is conservatively chosen to ensure large penalties are not assigned to well-behaved hosts. Although, adaptive misbehaving hosts may gain some throughput advantage by exploiting the conservative penalty scheme, in the absence of the penalty scheme, misbehaving hosts can obtain significantly larger benefits (e.g., comparing with the "802.11 - MSB"



(a) Diagnosis Accuracy (b) Throughput comparison

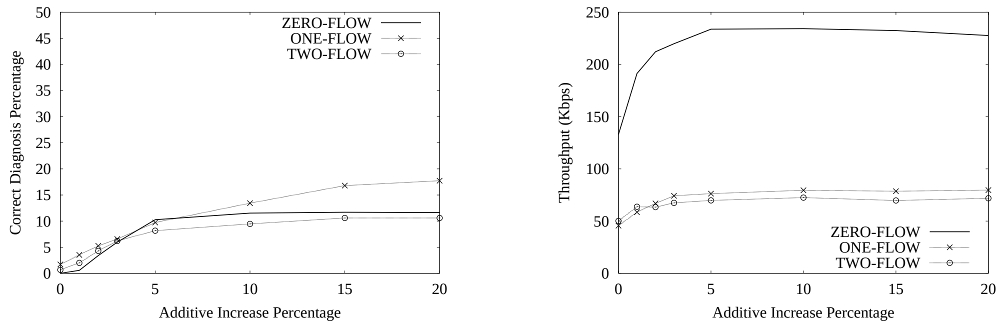
Figure 6.8 Protocol performance with adaptive misbehavior

curve in Figure 6.5).

6.6 Results for extensions to the protocol

In this section, we evaluate the optional extensions to the protocol for reducing misdiagnosis, discussed in Chapter 5.2. As we noted there, by modifying the receiver to conservatively classify slots as idle, we can reduce misdiagnosis, but allow a misbehaving host to gain larger throughput. To ascertain the impact of the extension, we evaluate the extended protocol under adaptive misbehavior model.

With the modified protocol, the misdiagnosis percentage is zero for all three scenarios (we have not included that figure here as the curves overlap with x-axis). Figure 6.9(a) plots the correct diagnosis percentage for the modified protocol. As we can see from the figure, the diagnosis accuracy is lower than that obtained with the base protocol under adaptive misbehavior model (Figure 6.8(a)). The low diagnosis accuracy is a consequence of conservatively classifying some busy slots to be idle, thereby not diagnosing some misbehavior instances. However, the diagnosis accuracy is high under persistent misbehavior model (those results are similar to the results for the base protocol and are



(a) Diagnosis Accuracy

(b) Throughput comparison

Figure 6.9 Performance with extensions to protocol

omitted here). The throughput curves for the extended protocol in Figure 6.9(b) are also similar to that of the base protocol (Figure 6.8(b)), with the misbehaving host obtaining only a small increase in throughput. Hence, the extension to the protocol does not allow a misbehaving host to gain significant benefits. We postpone for future work evaluation of the extended protocol under alternate misbehavior strategies that may be designed to specifically exploit these protocol extensions.

CHAPTER 7

CONCLUSION AND FUTURE WORK

In this thesis, we identified certain selfish misbehavior possible in IEEE 802.11 networks. With the growing popularity of IEEE 802.11 based wireless networks, handling selfish misbehavior is necessary to ensure a reasonable throughput share for well-behaved hosts in the presence of misbehaving hosts. We presented simple modifications to IEEE 802.11 that preserve most of the features of IEEE 802.11 MAC protocol, while enabling detection of misbehavior. A key contribution of the thesis was to demonstrate that even simple modifications to a protocol can greatly improve its resilience to misbehavior. We also proposed a scheme to penalize misbehaving hosts. The combination of diagnosis and penalty schemes ensures that selfish hosts are detected or prevented from misbehaving. We proposed extensions to the base protocol to enable detection of more sophisticated receiver misbehavior, and outlined techniques to reduce misdiagnosis.

Future work includes detecting other types of host misbehavior, such as a host using multiple MAC addresses for obtaining higher bandwidth share, with the support of higher layers. Designing strategies for combining information from multiple observers, as well as optimally placing the observers are also part of future work. Another area of research is adaptive selection of protocol parameters.

REFERENCES

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

[2] F. Stajano and R. Anderson, "The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks," in *Security Protocols, 7th International Workshop Proceedings*, vol. 2133 of *LNCS*, Springer, April 1999, pp. 172–194.

[3] J. Hubaux, L. Buttyan, and S. Capkun, "The Quest for Security in Mobile Ad Hoc Networks," in *Proceedings of ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Long Beach, CA, October 2001.

[4] L. Buttyan and J. H. (eds), "Report on a Working Session on Security in Wireless Ad Hoc Networks," *Mobile Computing and Communications Review*, vol. 6, November 2002.

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

[6] 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, IEEE Computer Society, January 2002, pp. 403 – 410.

[7] S. Buchegger and J. Le Boudec, "Performance Analysis of the CONFIDANT Protocol: Cooperation Of Nodes — Fairness In Dynamic Ad-hoc NeTworks," in *Proceedings of IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Lausanne, CH, IEEE, June 2002.

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

[9] L. Buttyan and J. Hubaux, "Enforcing Service Availability in Mobile Ad-Hoc WANs," in *Proceedings of IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Boston, MA, USA, August 2000.

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

[11] 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.

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

[13] A. B. MacKenzie and S. B. Wicker, "Stability of Multipacket Slotted Aloha with Selfish Users and Perfect Information," in *Proceedings of Infocom 2003*, San Francisco, CA, IEEE, April 2003.

[14] 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.

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

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

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

[18] 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.

[19] 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.

[20] 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.

[21] P. Kyasanur and N. H. Vaidya, "Detection and Handling of MAC Layer Misbehavior in Wireless Networks," in *Proceedings of the 2003 International Conference on Dependable Systems and Networks*, San Francisco, CA, June 2003, pp. 173–182.

[22] 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.

[23] D. E. Knuth, *The Art of Computer Programming*, vol. 2, ch. 3, pp. 10–17. Addison-Wesley, 3 ed., 2000.

[24] V. Poor, *An Introduction to Signal Detection and Estimation*, vol. 1, ch. 2, pp. 22–29. Springer-Verlag, 2 ed., 1994.

[25] F. Cali, M. Conti, and E. Gregori, "IEEE 802.11 Protocol: Design and Performance Evaluation of an Adaptive Backoff Mechanism," *IEEE Journal on Selected Areas in Communication*, vol. 18, no. 9, 2000.

[26] R. Rozovsky and P. R. Kumar, "SEEDEX: A MAC protocol for ad hoc networks.," in *Proceedings of ACM Symposium on Mobile Ad Hoc Networking and Computing*, October 2001, pp. 67–75.

[27] P. Karn, "MACA - A New Channel Access Method for Packet Radio," in *Proceedings of 9th Annual ARRL Networking Conference*, London, Ontario, Cananda, 1990.

[28] K. Fall and K. Varadhan, "ns notes and documentation," tech. rep., UC Berkley, LBL, USC/ISI, Xerox PARC, 2002.

[29] 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.