Detection and Handling of MAC Layer Misbehavior in Wireless Networks

Technical Report, August 2002

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

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

***Abstract*— Selfish hosts in wireless networks that fail to adhere to the MAC protocol may obtain an unfair share of the channel bandwidth. We present modifications to the IEEE 802.11 backoff mechanism to simplify detection of such selfish hosts. We also present a *correction* scheme for penalizing greedy misbehavior which attempts to restrict the misbehaving nodes to a fair share of the channel bandwidth. Simulation results indicate that our detection and correction schemes are fairly successful in handling MAC layer misbehavior.**

I. Introduction

Wireless MAC protocols such as IEEE 802.11 [1] use cooperative contention resolution mechanisms for sharing the channel. These wireless networks may be built without a central authority and hosts can join or leave the network at any time. In this environment, some *selfish* hosts in the network may misbehave by failing to adhere to the network protocols with the intent of obtaining an unfair share of the channel bandwidth. Our work focuses on detecting and handling MAC layer misbehavior by *selfish* hosts in IEEE 802.11-based networks.

IEEE 802.11 MAC protocol [1] has two mechanisms for contention resolution; a centralized mechanism called PCF (Point Coordination Function) and a distributed mechanism called DCF (Distributed Coordination Function). In DCF mode, a node with a packet to transmit initializes a backoff counter with a random backoff value selected uniformly from range $[0, CW]$, where CW is a variable maintained by each node called the *Contention Window*. While the channel is idle, the backoff counter is decremented by one every slot (the length of the slot is a protocol constant) and the counter is frozen when the channel becomes busy. The node transmits a packet when the backoff counter value reaches zero. If the transmission is successful, CW is reset to an initial value (CW_{min}). If the transmission is unsuccessful (may be due to collision with a transmission from another node), the Contention Window value is set to $2*CW+1$, subject to a maximum of CW_{max}. All the participating nodes are required to adhere to the backoff protocol for ensuring a fair share of bandwidth for each node in the long run.

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

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

- Selecting backoff values from a different distribution with smaller average backoff value than the distribution specified by the backoff protocol (Example: Selecting small constant backoff values or selecting backoff values from a smaller range than [0,CW].)
- Using a different retransmission strategy that does not double the CW value after collision.

In this paper, we consider mechanisms to detect nodes misbehaving with the intent of obtaining an unfair share of bandwidth (greedy misbehavior), in IEEE 802.11-based networks and penalize such misbehavior.

II. Related Work

Considerable research has been done on detecting and handling misbehavior at the network layer [2], [3], [4], [5], [6] in wireless networks. One approach is to identify misbehaving nodes and avoid such nodes in routing [4]. The redundant routes in ad-hoc networks are used to route around misbehaving nodes. This ensures that reasonable throughput is obtained by well-behaved nodes but it does not penalize the misbehaving nodes. Thus, misbehaving nodes may obtain higher throughput and reduced energy consumption.

To discourage misbehavior, another approach is to design protocols that encourage cooperation by penalizing misbehavior [7], [6], [8]. Buttyan *et al.* [7] propose a model where nodes are paid for the services that they offer to other nodes and in turn pay other nodes when they receive any services. To provide a secure payment scheme, they propose the use of a tamper-resistant security module that is present in every node. Buchegger *et al.* [6] propose a mechanism where nodes do not provide services (such as forwarding packets) to nodes which are misbehaving. Every node monitors the behavior of all other nodes in its neighborhood and identifies possible misbehaving nodes. This information about misbehaving nodes is shared among a group of nodes which trust each other ("friends"). Based on this information, nodes refuse to forward packets for nodes that appear to be misbehaving.

Game-theoretic techniques have been used to develop MAC protocols [9], [10], [11] which are resilient to misbehavior. These protocols are designed to reach an optimal operating point called the "Nash equilibrium". At this operating point, no node can get more bandwidth by unilaterally deviating from the protocol (if nodes collude, they might get better bandwidth). These approaches assume that the misbehaving nodes are "*rational*" [12] i.e., misbehaving nodes exhibit "greedy misbehavior" and are interested primarily in improving their share of bandwidth. So, the misbehaving nodes are not expected to attempt to degrade the bandwidth share of other nodes if such an attempt fails to improve the share of bandwidth obtained by the misbehaving node. Further, the nodes continue to use a strategy if that strategy has previously provided them with better performance. For example, a node receiving a higher share of bandwidth by using a strategy is assumed to use that strategy in the future too. Hence, a *rational* node shows some predictability in its behavior. The game-theoretic approach has the potential for designing protocols which are resilient to misbehavior, thereby avoiding the need for explicit detection and correction mechanisms. However, this approach could have several problems:

- The network performance at Nash equilibrium might be far from optimal. Nash equilibrium can only assure that no single node in the network can get better performance than other nodes. However, it does not guarantee that the overall network performance is near the maximum that would be possible by using a different protocol. For example, if minimizing energy consumption is not a requirement, then Nash equilibrium is reached when all nodes use the strategy of transmitting without backing off. In this setting, all nodes obtain low throughput, but no node can do better than other nodes by unilaterally changing its strategy. On the other hand, a cooperative protocol such as IEEE 802.11 can generally provide better performance than that is obtained at the Nash equilibrium. So, for scenarios where node misbehavior is not common, game-theoretic solutions might not be practical.
- Misbehaving nodes may not always behave *rationally*. Using game-theoretic approach does not guarantee that nodes deviating from *rational* behavior will be restricted to a fair share of bandwidth.
- The approach does not work in case of collusion between misbehaving nodes.

The solution that we present in this paper modifies the IEEE 802.11 backoff algorithm for simplified misbehavior detection while retaining its performance characteristics. Our proposed solution assumes that nodes misbehave with the intention of obtaining an unfair share of the bandwidth. However, our solution handles each instance of misbehavior separately and consequently, predictability in behavior of misbehaving node is not necessary for the success of our proposed solution. Our proposed solution cannot detect collusion between misbehaving nodes but we present extensions to the protocol to facilitate collusion detection.

III. Approaches for detecting misbehavior

Greedy misbehaving nodes attempt to obtain more than their fair share of the bandwidth. This suggests the use of traffic analysis to identify such misbehaving nodes by observing the share of bandwidth obtained by each node. The potential problems that may arise with this approach are,

- It is difficult to estimate the *fair share* of bandwidth a well-behaved node will receive. The *fair share* needs to be adaptively estimated based on channel conditions and the number of nodes that are contending for the channel at that time.
- IEEE 802.11 backoff algorithm is unfair in the short term. Thus, the observation interval must be sufficiently large to avoid incorrect designation of well-behaved nodes as misbehaving. Consequently, short-term misbehavior may not be detected.
- Misbehaving nodes may have shorter average delay compared to well-behaved nodes without being detected. For example, a misbehaving node can send the same number of packets as a well-behaved node

on the average, but each packet is sent after backing off for a fraction of the backoff value required by the protocol. Observing the throughput obtained by nodes will not reveal this type of misbehavior. Thus, the average delay for misbehaving nodes will be lesser than that of well-behaved nodes, but traffic analysis may not identify this type of misbehavior.

The DCF function in IEEE 802.11 protocol uses random selection of backoff values. Consequently, a sequence of small backoff values from a misbehaving node cannot be distinguished from a legitimate sequence arising out of random selection. Thus, approaches such as traffic analysis that identify deviations from normal behavior using statistical methods may not be satisfactory.

An alternate approach is to modify the backoff algorithm to simplify misbehavior detection, while providing similar fairness characteristics and throughput as provided by IEEE 802.11 protocol. We use this approach to modify the IEEE 802.11 protocol for simplified misbehavior detection. We first present the modified protocol and then outline the detection scheme.

A. Terminology:

We define the following terminology which we use in presenting the protocol.

- **Sender:** Sender is a node which transmits a packet to a receiver.
- **Receiver:** Receiver is a node which receives a packet from a sender.
- **Monitor:** Monitor is a node which observes the behavior of sender and receiver nodes to identify potential misbehavior.

Sender, *receiver* and *monitor* are the different roles a node can perform. A node could be a sender at some instant of time (when it transmits a packet) and a receiver during a different instant of time (when it receives the transmission from some other node). In case of IEEE 802.11 protocol, the sender node transmits a DATA packet to a receiver node after an optional RTS-CTS exchange. A monitor node could observe the packets sent to it by a sender (the monitor node is also the receiver) or could observe the packets sent to some other node by a sender (a different node is the receiver).

Figure 1 illustrates a simple scenario having three nodes A, B and C. Node A sends a DATA packet to node B after exchanging RTS-CTS packets and the transmission is overheard by node C. In this example, node A acts as the sender, node B is the receiver and node C is an observer which can overhear the transmissions between A and B. The monitor node could be any or all of nodes A, B and C. Node A can monitor the receiver B to identify potential receiver misbehavior (we present the possible receiver misbehavior later). Node B can monitor the sender A to identify potential sender misbehavior. Node C can monitor both the sender A and receiver B to identify sender misbehavior, or receiver misbehavior, or collusion between the sender and the receiver.

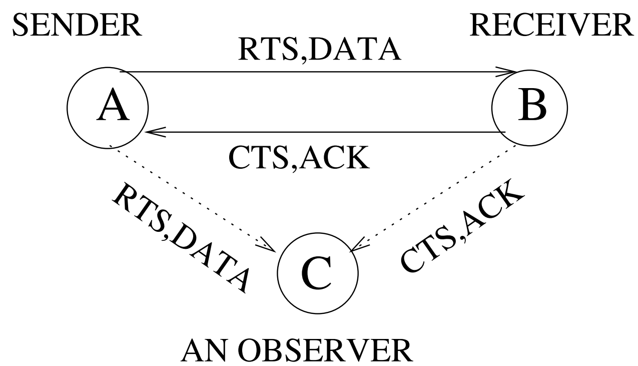

Fig. 1. Scenario depicting different roles performed by a node

In our proposed solution, we assume that a monitor node monitors only those packets that are sent to it by a sender. Hence, our proposed protocol allows a receiver to detect sender misbehavior. Later in the paper, we consider the use of sender nodes as monitors to detect potential receiver misbehavior.

The first step in the monitoring process is for the monitor to predict the sequence of backoff values that will be selected by a sender that it is monitoring. Next, the monitor counts the number of idle slots I between two successive transmissions from that sender, and verifies that I is at least as large as the backoff value that the sender was predicted to select. If the verification fails, then the sender is deemed to have deviated from the backoff protocol. The actual details of the scheme are presented later in Section IV.

For the monitor node to accurately predict the sequence of backoff values that will be used by the sender node, the following schemes can be used:

1) The monitor can provide an initial seed to the sender and the sender uses a function of its unique node identifier and the initial seed to select successive backoff values. As the monitor knows both the initial seed that it has provided as well as the node identifier of the sender, the sequence of backoff val-

ues that the sender will use can be accurately determined.

2) The sender initially selects a random seed and sends this seed to the monitor. The sequence of backoff values selected are a function of the node identifier of the sender and the selected seed. As in the previous scheme, the monitor can compute the sequence of backoff values that the sender is going to use.

The sharing of the initial seed of the pseudo-random generator for predicting the successive sequence of backoff values is similar to the approach used in SEEDEX protocol [13]. However, SEEDEX protocol uses the knowledge of backoff sequence for scheduling packet transmissions.

A problem with the two schemes described above is that the monitor cannot predict the backoff sequence correctly if it does not receive a packet transmitted by the sender. This can happen when a packet sent by the sender collides at the monitor or the packet is received with errors and is dropped. To overcome this problem, the above two schemes can be extended to periodically synchronize the monitor with the backoff value being used by the sender. Using these ideas, we present the algorithm for misbehavior detection in the next section.

IV. Misbehavior Detection using Modified Backoff Protocol

Nodes follow the IEEE 802.11 protocol [1] except for the modifications to the backoff scheme that we propose. To simplify the description, we assume that RTS-CTS packets are always exchanged to reserve the channel before the actual data packets are exchanged. So the sequence of packets transmitted for one data transmission is RTS-CTS-DATA-ACK. The receiver provides the backoff value to be used by a sender for the next $[(i+1)^{th}]$ transmission to the receiver in the CTS packet of the current $[i^{th}]$ transmission (this value can be included in the ACK if RTS-CTS exchange is not used).

A. Modified Backoff Mechanism

We present the modifications to the backoff mechanism by specifying the steps taken by the sender and the receiver during the transmission of a packet from the sender to the receiver. The RTS packet has a field called *attemptNumber* used in the detection procedure in addition to the fields of 802.11 protocol. The CTS packet has an additional field called *nextBackoff* used in the modified backoff protocol. In the absence of RTS-CTS exchange, the field in the RTS could be incorporated into the header of the DATA packet and the field in the CTS could be incorporated into the ACK packet. The rules used for backing off are the same as that used in 802.11 protocol. When the channel status changes from busy to idle, a node waits for DIFS time and after that, while the channel continues to be idle, the backoff counter is decremented by 1 every slot time. If the backoff counter is decremented to zero, a packet is transmitted. If the channel becomes busy before the backoff counter is decremented to zero, the counter value is frozen till the beginning of the next idle interval, when the above procedure is repeated.

We designate the sender as ‘S’ and receiver as ‘R’.

1) When S wants to send a packet to a R for the first time (S and R have not communicated in recent past), S picks a random backoff value, and transmits the RTS after that backoff value expires as in IEEE 802.11 protocol. The *attemptNumber* field is set to 1 for this transmission.
2) Whenever R receives an RTS from S, it generates a random backoff value X uniformly from a fixed range [0,CW_{min}] (CW_{min} is the minimum CW value of 802.11 protocol), and sends a CTS with the *nextBackoff* field set to X. R stores X as the expected backoff value *nextBackoff*$_S$, for the next transmission from node S.
3) When S receives the CTS, it stores the value of *nextBackoff* field as the value to be used for backing off on the next transmission to R.
4) Whenever S needs to send a packet to R after the first time (S and R have communicated with each other in the recent past), it uses the stored backoff value *nextBackoff* for backing off before transmitting the RTS. The RTS *attemptNumber* is initially set to 1 and CW (Contention Window) value is set to CW_{min}.
5) On sending an RTS to R, if no CTS is received in a specified time, the *attemptNumber* value is incremented and CW is doubled, subject to a maximum of CW_{max}. A new backoff value Y is computed by selecting a number in range 0 to 1 using a deterministic function f and scaling it by CW.

$$Y = f(\textit{nextBackoff},\textit{nodeID},\textit{attemptNum}) * CW$$

where *nextBackoff* is the stored value of backoff, *nodeID* is the unique node identifier (MAC address can be used for this) of S and *attemptNum* is the number of attempted transmissions, which is reset to 1 after a successful transmission and incremented by 1 after each unsuccessful transmission. The RTS is then retransmitted after backing off for Y slots.

The details of the function f and an analysis of its properties are provided in appendix II. The node identifier used as a parameter to function f ensures that two colliding nodes will use different values of backoff for retransmission after collision, even if their *nextBackoff* values are the same. The retransmission attempt number is used as a parameter to f for generating different numbers between 0 and 1 for different retransmission attempts. The function f is chosen such that the expected value of f is 0.5 [1] when the *nextBackoff* value used as a parameter to f is selected using an uniform distribution. Hence, the expected value of backoff for each retransmission is $\frac{CW}{2}$ (as Y is computed by scaling the output of f by CW). In IEEE 802.11 protocol, the backoff values for a transmission are selected randomly from range [0,CW] and the average value of backoff is $\frac{CW}{2}$. Thus, the backoff mechanism used in our modified protocol is *equivalent* to IEEE 802.11 protocol in terms of the average backoff. The difference from IEEE 802.11 protocol is that the backoff value used for the first transmission attempt (of i^{th} transmission from S to R) for which $CW = CW_{min}$, is provided by the receiver, and subsequent values are computed at the sender using the value provided by receiver. Thus, the receiver can predict the sequence of backoff values that the sender uses simplifying the process of detecting sender misbehavior.

B. Procedure to detect Sender Misbehavior:

We designate the receiver node as M and the sender as S. The receiver node is also the monitor node and observes the transmissions it receives from any sender S to detect possible sender misbehavior.

1) M has an *IdleCounter* which counts the idle slots, starting from the time when the monitor is enabled (possibly when the receiver node is turned on). The *IdleCounter* follows the rules of 802.11 in counting the idle slots. Specifically, when the channel goes idle after a busy interval, it waits for DIFS interval and counts idle slots from then on while the channel continues to be idle. The counter value is frozen when the channel becomes busy.

[1] As the function f is deterministic, each value of *nextBackoff* (for constant *nodeId* and *attemptNum*) maps to a specific number between 0 and 1. As the range of values for *nextBackoff* is fixed between 0 and CW_{min} the range of values of f are also fixed independent of the *attemptNumber*. We discuss alternative functions to increase the range of f for larger *attemptNumber* in Appendix II.

2) M stores the *nextBackoff* values assigned by it to all the senders (and, in general, M may store the *nextBackoff* value of any sender that M wants to monitor including those sending packets to a different receiver).
3) When M sends an ACK for acknowledging the data received from S, M stores the value of *IdleCounter* at that time as the earliest time $eTime_S$, when S can start counting down the given backoff value.
4) When M receives an RTS from a node S (except when they are communicating with each other for the first time), M can verify that node S has waited for the required number of idle slots. The number of slots B_{exp} that S is expected to wait is given by,

$$B_{exp} = nextBackoff_S + Y$$

where Y is computed as,

$$Y = \sum_{i=2}^{attemptNum} f(nextBackoff_S, nodeID, i) * CW_i$$

attemptNum is the transmission attempt number sent in RTS, $nextBackoff_S$ is the backoff assigned by receiver to sender S for the current transmission, and *nodeID* is the unique node identifier of the sender S. CW_i is the Contention Window value for the i^{th} attempted transmission and is given by,

$$CW_i = min(\ CW_{min} * 2^{i-1}\ ,\ CW_{max}\ )$$

5) The node S is said to be conforming to the protocol if the actual number of idle slots elapsed B_{obsr}, computed as the difference of current value of *IdleCounter* and $eTime_S$ satisfies the condition,

$$B_{obsr} \geq \alpha * B_{exp}\ ,\ 0 < \alpha \leq 1$$

where α is a protocol constant. α specifies the minimum fraction of B_{exp} a node should actually wait to be designated as conforming to the protocol. If the above condition is not satisfied, then the sender S is designated to have *deviated* from the protocol.
6) When the monitor identifies more than a threshold *THRESH* number of deviations by S for the last K packets from S (*THRESH* and K are protocol constants), it designates the node S to be *misbehaving* and takes suitable action (such as informing upper layers or taking some corrective action at the MAC layer).

The accuracy of the detection process depends on the differences in channel conditions seen by the monitor and sender. Due to variations in channel conditions, the sender might sense the channel to be idle while the monitor might sense the channel to be busy. In this scenario, the monitor can falsely conclude that the sender is misbehaving. The parameters α, *THRESH* and K can be set, perhaps adaptively, to ensure that the number of wrong predictions is reduced. For example, when we reduce α, the sender is required to wait for a smaller fraction of B_{exp} and thus false predictions are likely to be reduced. However, care must be taken to ensure that the chosen values of protocol parameters do not allow a misbehaving node to obtain a significant throughput advantage while escaping detection.

The protocol requires the senders to correctly set the *attemptNum* when a RTS is retransmitted. To ensure that senders follow this procedure, the monitor can sense the channel to identify high collision intervals (when the channel is mostly busy but few transmissions are successful) and analyze the traffic during such intervals to see if any node B is having more successful transmissions on an average than other nodes or has smaller average *attemptNumber* than other nodes. If such a node B exists, then the monitor can instruct the receiver to intentionally drop RTS packets from B occasionally and check whether B increments the *attemptNumber* in the retransmission of RTS. Even a single failure by B to increment the *attemptNumber* in the retransmission is an immediate proof of misbehavior by B. As B does not know which RTS packets are lost due to collision and which are intentionally dropped by the receiver, it will be harder for the node to misbehave always by sending incorrect *attemptNumber*. Dropping RTS packets occasionally will not affect the throughput of B significantly. Since B is getting a higher share of throughput a slight decrease in its throughput may be tolerable.

V. Handling misbehavior at MAC layer

Misbehaving nodes obtain more than their fair share of throughput. Therefore, one approach in handling node misbehavior is to penalize such nodes by assigning larger backoff values to them than those assigned to conforming nodes. We designate this approach as the *correction* scheme. When an RTS is received at the receiver from a sender, the monitor at the receiver can compare the expected value of backoff with the actual value of backoff to identify deviations. If the actual value is lesser than α times the expected value (α is a protocol parameter), then the difference D, computed according to the equation given below, is added to the randomly generated *nextBackoff* sent in the CTS.

$$D = max(\ \alpha * B_{exp} - B_{obsr}\ ,\ 0\ )$$

The sender node is expected to wait for the assigned *nextBackoff* value (*nextBackoff* could now be greater than CW_{min}). On collision, the rules presented earlier in section IV-A are used. This additional penalty is intended to compensate for misbehavior and thereby we are attempting to ensure that the average actual backoff of misbehaving nodes is close to that of conforming nodes.

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

Let the misbehaving node suffer E_{diff} fewer collisions per successful transmission on an average in comparison to those suffered by a well-behaved node. We estimate E_{diff} by applying the approach of Cali *et al.* in [14]. The theoretical analysis for estimating E_{diff} is presented in Appendix I. Once we have estimated the benefit obtained by the misbehaving node (designated as MB) in terms of reduced collisions, we can penalize it for that benefit by computing the additional slots (AS) the node would have backed off for, if it had actually suffered the estimated additional collisions.

$$AS = \sum_{i=aNum_{MB}+1}^{aNum_{MB}+E_{diff}} f(nextBackoff_{MB}, nodeId, i) * CW_i$$

where $aNum_{MB}$ is the actual number of collisions suffered by the misbehaving node MB and CW_i is the contention window size for the i^{th} transmission attempt, *nodeId* is the node identifier of misbehaving node. The characteristics of function f were provided earlier while describing the modified backoff protocol in section IV-A.

So, the total penalty added for the misbehaving node is given by

$$Penalty = (\alpha * B_{exp} - B_{obsr}) + AS$$

where B_{exp} is the expected backoff (algorithm for computing it was presented earlier in section IV-B) and B_{obsr} is the observed number of idle slots between transmissions from the node.

VI. Types of misbehavior handled by the Proposed protocol

In this section we analyze the types of MAC layer misbehavior handled by our protocol. We can broadly classify the types of greedy node misbehavior into three categories:

Case 1: All the receivers are well-behaved and only the senders are possibly misbehaving. There is no collusion between the senders and the receivers. Our protocol is effective in detecting and correcting misbehavior in this setting. An example of this setting is an infrastructure-based network having a trusted base station and possibly misbehaving senders. Since the base station is well-behaved and is the sole receiver, there is no collusion between the senders and the receivers.

Case 2: Both the senders and the receivers are misbehaving but there is no collusion between the senders and the receivers. This type of misbehavior is possible in any ad hoc network or wireless LAN, having multiple senders and receivers. Our protocol can partially handle this type of misbehavior. We can detect sender misbehavior but cannot detect receiver misbehavior. A possible scenario is a misbehaving node that adds arbitrarily large penalty to nodes sending packets to it (to possibly increase its own chances of accessing the channel). This attack was not possible in IEEE 802.11 protocol as the receiver had no control over sender behavior. However, this attack can be mitigated in our protocol by turning off the *correction* scheme. We plan to add additional safeguards to the correction scheme to prevent such attacks.

Case 3: There is collusion between misbehaving senders and receivers. Our protocol may not identify this type of misbehavior. An example of such misbehavior is for the receiver to assign small backoff values to a preferred sender and larger backoff values to other senders.

Our approach detects deviations from the correct behavior with fairly high accuracy and handles each deviation separately. The *correction* scheme penalizes each deviation of a node based on the value of that deviation independent of the prior behavior of that node. The *correction* scheme can be considered to be "*memoryless*" as the penalty for each deviation is calculated independent of the penalty for prior deviations. Hence, our approach is effective against an adaptive strategy adopted by a misbehaving node. For example, our protocol is not subverted by on-off type of misbehavior or bursty misbehavior. With the use of the *memoryless* approach in our protocol, deviations of well-behaved nodes from the protocol (arising out of varying channel conditions at the receiver and sender) may lead to wrongly penalizing such nodes. However, the penalty added is usually small and simulation results indicate that in the absence of misbehavior, the throughput of well-behaved nodes when using our proposed protocol is comparable to that obtained using IEEE 802.11 protocol.

VII. Simulation Results

We have performed a preliminary evaluation of the proposed scheme. We present the simulation model and simulation results in the following sections.

A. Simulation Model

We use the ns-2 [15] simulator for our simulations. The simulator has been modified to accurately implement the counting down of backoff values. In the original ns-2 implementation, channel idle status is not verified during each slot while counting down the backoff values. In our modification, for each slot while the backoff counting is in progress, we check the channel status to confirm it is idle, and the backoff counter is frozen if the channel is found to be busy. This modification is needed because varying channel conditions may result in channel status being different in successive slots.

We use the *shadowing* model [15] of the channel in the simulation. This models the variations in channel conditions with time. The shadowing model has two components. The first component is the *path-loss model* which computes the mean power received at a distance d, denoted by $P_r(d)$ with respect to the power $P_r(d_0)$ received at a reference distance d_0 as follows (the measurement is in decibels)

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

β is called the Path Loss Exponent and is usually empirically measured. For free space propagation β is 2. We use this value of β in our simulations.

The second part of the shadowing model reflects the variation of received power on account of variation in channel conditions. It is represented by a Gaussian random variable X_{dB} with zero mean and σ_{dB} standard deviation. σ_{dB} is called the *shadowing deviation* and is obtained by measurements. For outdoor environments the typical range for σ_{dB} is from 4 to 12 [15] and we have simulated for this range of values. The overall shadowing model is represented as

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

With the use of shadowing model, the transmission range of a node is no longer a constant. Instead, we can only specify the probability of receiving a transmission at a certain distance from the transmitter.

In addition to the *shadowing* channel model, we apply an error model to the received transmission to model other losses that might occur in the channel on account of random noise, etc. The errors are modeled by a constant bit-error probability, which is the probability that a bit in a transmitted packet will be corrupted. Using this error model, longer packets have higher probability of being corrupted due to errors. The simulation model used here will be improved in future simulations of the proposed approach to model signal and noise variations using a single channel model.

B. Simulation Metrics

The metrics we use to evaluate the protocol are:

- *Correct Prediction:* This is computed as the percentage of packets transmitted by misbehaving nodes, which are correctly detected by the monitor as packets from a misbehaving node. This metric measures the effectiveness of the protocol in detecting misbehavior.

- *Misprediction:* This is computed as the percentage of packets sent by conforming nodes which are wrongly identified by the monitor to be packets from misbehaving nodes. This metric measures the error in the misbehavior detection process.

- *Average throughput:* This is the average number of data packets successfully sent by each sender node, including the misbehaving nodes, for the duration of simulation. It is computed as the ratio of total data packets successfully sent by all the sender nodes to the total number of sender nodes in the network. This metric is an indication of the fair share of throughput which all sender nodes in the network should ideally receive.

- *Misbehaving node throughput:* This is the average number of data packets successfully sent by misbehaving sender nodes for the duration of simulation. This is measured as the ratio of the total number of data packets successfully sent by misbehaving sender nodes to the number of misbehaving sender nodes in the network. This metric is useful for comparing the throughput obtained by misbehaving nodes with respect to their fair share.

To model various types of misbehavior, we define a parameter called "Percentage of conformance". A misbehaving node with percentage of conformance X always transmits a packet after counting down to X percent of the assigned backoff value. When the percentage of conformance is 100, the node is not misbehaving and when the percentage of conformance is close to 0, the node misbehaves by picking very small backoff values.

C. Simulation results for Infrastructure-Based Network

We simulate our proposed protocol for an infrastructure-based network containing a trusted base station (receiver) and multiple potentially misbehaving nodes (senders) transmitting packets to the base station. We use this network setup to evaluate the effectiveness of our protocol in detecting and handling sender misbehavior. The trusted base station is responsible for monitoring the behavior of all sender nodes transmitting packets to the base station. Both the *detection* scheme and the *correction* scheme are enabled unless mentioned otherwise. In the next section, we evaluate our protocol for an ad hoc network.

All the sender nodes in the network are backlogged. The traffic from the nodes to the base station is modeled as CBR traffic with rate of 2 Mbps and size of CBR packets is 512 bytes. The channel bit rate is 2 Mbps. The simulation time is 20 seconds. The simulation results are averaged over 30 runs of the simulation. Each run is seeded by a different seed and the set of seeds used for different data points is the same to allow for direct comparison among them.

Figure 2 shows the simulated network. The number of nodes around the base station is 8 (numbered 1-8 in the figure) and one of the eight nodes (selected randomly) is misbehaving. The 8 nodes are placed in a circle of radius 200 meters around the base station, equidistant from each other. The simulation area is a rectangular grid of length 1500 meters and width 750 meters. The base station is placed at the center of the grid.

There are 24 other nodes in the network placed in 6 chains with 4 nodes per chain. The chains are at distances of 300 meters, 500 meters and 700 meters on either side of the base station as shown in Figure 2. Each chain includes 3 CBR connections as shown in the figure. The transmissions along these chains may affect the carrier sensing at the base station and at the nodes that communicate with it. For example, in Figure 2, when node A sends a packet to node B, node 7 may sense the channel to be busy while the base station may sense the channel to be idle.

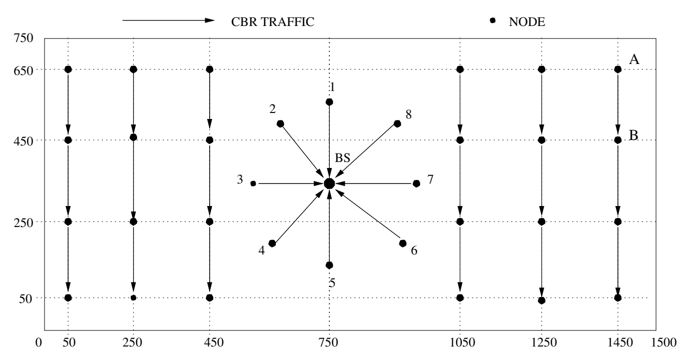

Fig. 2. Simulation setup for an infrastructure-based network with the base station at the center of the 1500m X 750m grid

1) Evaluation of effectiveness of misbehavior detection: Figure 3 shows the variation of 'Correct Prediction' and 'Misprediction' with variation in 'Percentage of conformance' of misbehaving nodes. Bit error rate is 0.001 and $\sigma_{dB} = 8$. The parameters of the protocol α, *THRESH* and K are set to 1, 10 and 6 respectively. We observe from the figure that the accuracy of detection increases rapidly as the conformance to the protocol by the misbehaving node reduces (decreasing percentage of conformance). So, we can detect node misbehavior with reasonable accuracy even when the deviation of misbehaving nodes from the protocol is not very large. The misprediction percentage on the other hand is reasonably small and is independent of the behavior of misbehaving node.

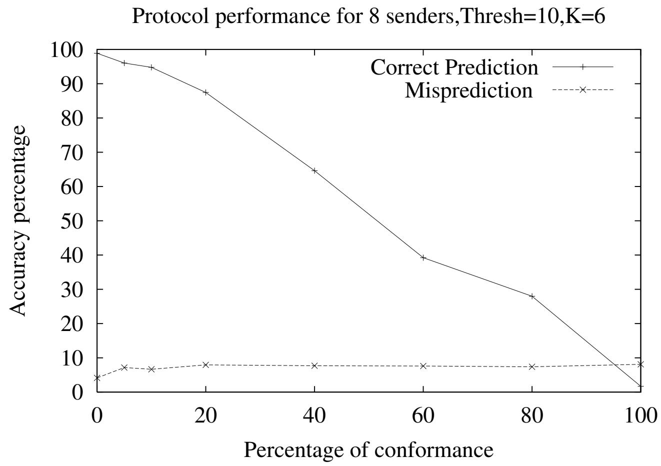

Fig. 3. Variation in correct prediction and misprediction percentage with percentage of conformance. THRESH and K are protocol parameters

Figure 4 shows the variation of 'Correct Prediction' with time. We measure the Correct Prediction' percentage for 1 second intervals starting from time 0 and the results are averaged over 30 runs. For example, the 'Correct Prediction' percentage plotted at 1 second is computed based on the packets received in the interval [1,2] seconds. The 3 curves in the figure plot the variation of 'Correct Prediction' for misbehaving sender nodes having Percentage of Conformance (denoted as PC in the figure) of 0%, 40% and 80%. We see from the figure that a misbehaving node not following the backoff protocol at all (modeled by PC = 0%) is correctly detected to be misbehaving with high accuracy. In addition, the detection accuracy reaches a maximum in a short interval after the node misbehavior begins (at time 0). For example, the detection percentage is close to 100% within a second for PC = 0%. The penalty added by the *correction* scheme increases with time for a misbehaving node that waits for a very small fraction of the assigned backoff. Consequently, the expected backoff for such nodes rapidly reaches a large value which increases the prediction accuracy. Hence, our protocol is quite successful in detecting misbehaving nodes which deviate significantly from the backoff protocol. When PC = 40%, the misbehaving node is backing off for 40% of the assigned backoff value. In this case, the accuracy percentage reaches close to 100% after a slightly longer duration of around 4 seconds (in comparison with around 1 second for PC = 0%). When

PC = 80%, the misbehaving node is backing off for almost all of the assigned backoff (80% of the assigned backoff). For this scenario, the detection accuracy reaches a maximum threshold of 60% and higher accuracy is not obtained as some deviations are not detected by the monitor.

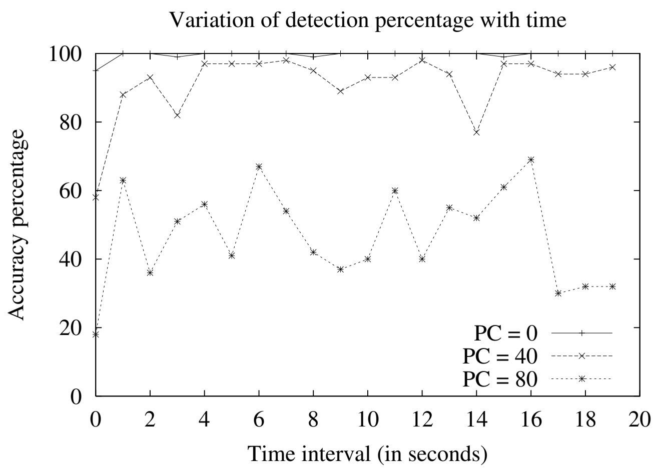

Fig. 4. Variation in correct prediction percentage with time for different Percentage of Conformance

Figure 5 shows the variation of misprediction percentage with varying channel conditions. We simulate bad channel conditions with high bit error rate of 0.01 and large standard deviation value of $\sigma_{dB} = 12$, good channel conditions with low bit error rate of 0.0001 and small standard deviation value of $\sigma_{dB} = 4$, and moderate channel conditions with bit error rate of 0.001 and standard deviation value of $\sigma_{dB} = 8$. From the figure, we see that the misprediction percentage is roughly similar but slightly *higher* for better channel conditions. However, with better channel conditions, the average throughput obtained by sender nodes is significantly higher than the throughput with worser channel conditions. This higher traffic on the channel could result in different observed channel conditions at the sender and receiver leading to higher misprediction rate with better channel conditions.. In addition, the "other" nodes placed in the vicinity of base station also exhibit higher traffic with better channel conditions and this might result in higher misprediction rate.

Figure 6 shows the variation of misprediction percentage with different protocol parameters. We observe from the figure that higher value of K for same *THRESH* value decreases the misprediction percentage, while same ratios of *THRESH* to K with smaller value of *THRESH*

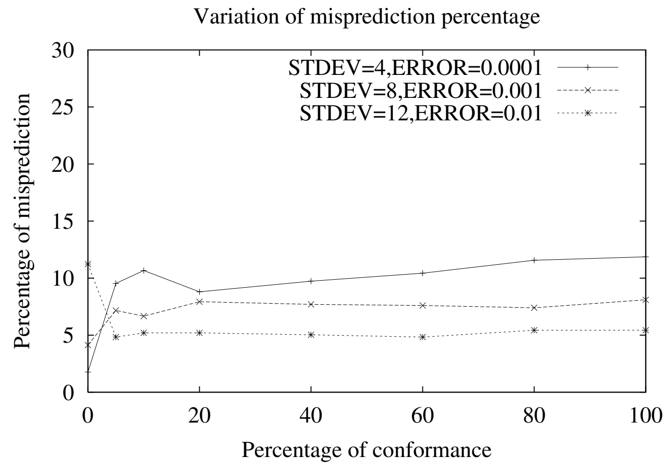

Fig. 5. Variation in misprediction percentage with percentage of conformance for varying channel conditions. STDDEV is the standard deviation value for shadowing model. ERROR is the packet error probability

increases the misprediction percentage. This suggests that suitable selection of protocol parameters can help in reducing inaccurate prediction. But the detection procedure using these protocol parameters cannot detect misbehavior by nodes adapting to the protocol parameters. To handle this, the *correction* scheme treats every deviation from the protocol as an indication of misbehavior and performs correction by adding a penalty (for the correction scheme, the protocol parameters are effectively $\alpha = 1$, *THRESH=1*, and $K = 1$ irrespective of the values used). So, the protocol parameters could be adaptively set by the monitor or statically by an administrator based on channel conditions to reduce inaccurate predictions. The detection scheme merely serves as a mechanism to inform higher layers of potential misbehavior.

Figure 7 shows the variation of correct prediction percentage with different protocol parameters. From the figure, we see that correct prediction is less sensitive to change in protocol parameters. However, the correct prediction percentage does depend on the type of misbehavior. The protocol may provide different prediction percentages for different types of misbehavior. In particular, if the misbehaving node adapts to the protocol parameters, the correct prediction percentage could be lower. To safeguard against these attacks, the *correction* scheme adds a penalty for every deviation.

2) Evaluation of effectiveness of misbehavior correction: To analyze the effectiveness of our proposed protocol in correcting misbehavior we run the simulations under the following settings (each simulation run consists

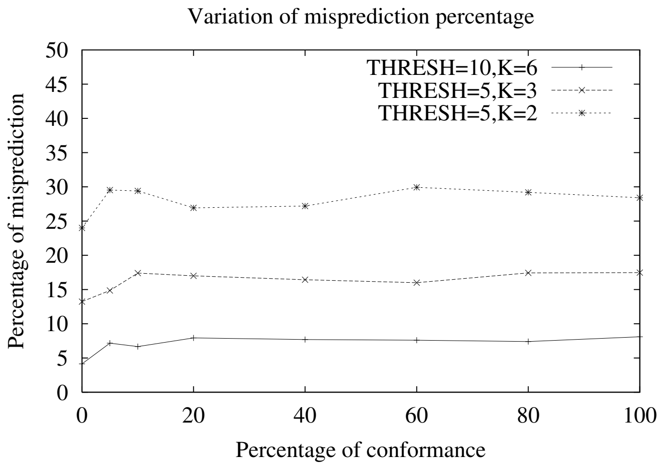

Fig. 6. Variation in misprediction percentage with percentage of conformance for different protocol parameter values

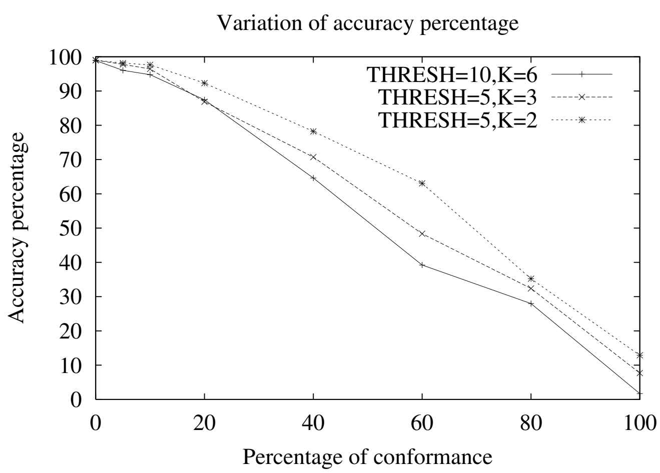

Fig. 7. Variation in correct prediction percentage with percentage of conformance for different protocol parameter values

of a randomly selected misbehaving node and a particular percentage of conformance):

- BASE : This setting simulates the IEEE 802.11 protocol. No misbehavior detection or correction is done. All nodes are well-behaved. This setting is used to compare the performance obtained using our modified protocol with that obtained using IEEE 802.11 in the absence of any misbehavior.
- DETECT : In this setting, only *detection* scheme is enabled. The percentage of conformance of misbehaving node is varied from 0% to 100%.
- CORRECT: In this setting, both the *detection* and *correction* schemes are enabled. The percentage of conformance of misbehaving node is varied from 0% to 100%.

As mentioned in the previous section, the correction scheme designates every deviation from protocol as possible misbehavior and performs correction by adding the computed penalty. High misprediction rate could result in degradation of throughput obtained by conforming nodes in the absence of node misbehavior. We compare the throughput obtained by the nodes in a network having well-behaved nodes only using the *correction scheme* (CORRECT setting) with the throughput obtained using IEEE 802.11 (BASE setting). The bit error rate is 0.001 and $\sigma_{dB} = 8$. The number of nodes "around" the base station is varied from 2 to 64 and there are 24 "other" nodes in the network (as described earlier). The nodes around the base station are placed in a a circle of radius 200 meters *around* the base station, equidistant from each other. Figure 8 compares the average number of packets sent by the senders using IEEE 802.11 protocol (BASE curve) with the average number of packets sent by senders using the *correction* scheme (CORRECT curve) for varying number of nodes in the network. From Figure 8 we can see that the correction scheme does not degrade the throughput of conforming nodes in comparison with the throughput obtained using IEEE 802.11 protocol. The correction scheme does occasionally add penalty to well-behaved nodes. But for well-behaved nodes, the computed penalty is usually small. In the long run, adding small penalty occasionally does not affect the throughput significantly.

In Figure 8, the average throughput (measured as the average number of packets sent by nodes placed around the base station) is less for smaller number of nodes and peaks at around 32 nodes in the network. In our simulation setup, we have 6 chains of "other" nodes in the vicinity of the base station. When the number of nodes around the base station is small, in many simulation runs, most of the channel bandwidth is being obtained by the other nodes in the vicinity of the base station. Consequently, the average throughput is less for smaller number of nodes. For larger number of nodes ($>$ 32 nodes) the average throughput decreases because of increase in overhead due to contention among nodes around the base station.

We evaluate the effectiveness of the correction scheme by comparing the throughput obtained by misbehaving nodes when using only the *detection* scheme and when using both the *detection* and *correction* schemes (i.e., DETECT and CORRECT respectively). We compare the CORRECT scheme with DETECT scheme instead of the BASE scheme to evaluate the effectiveness of using *correction* scheme in addition to the *detection* scheme. The evaluation network has 8 nodes and a base station

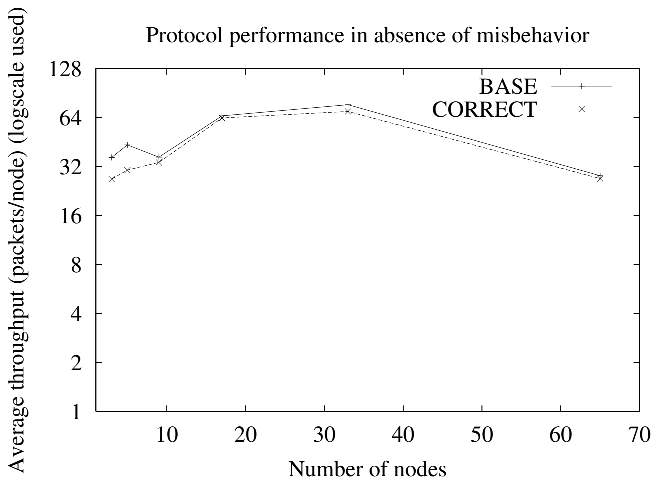

Fig. 8. Comparison of throughput with and without correction scheme for different network sizes

and 24 "other" nodes as described before. There is one misbehaving node (and it is randomly selected). The error rate is 0.001 and $\sigma_{dB} = 8$.

Figure 9 plots the throughput obtained by the misbehaving node and the average throughput per node with varying percentage of conformance of the misbehaving node for DETECT and CORRECT settings. In the figure, "AVG TPUT" represents the average throughput obtained by all the sender nodes in the network (including misbehaving nodes) and "MSB TPUT" represents the average throughput obtained by the misbehaving sender nodes in the network. From the figure we observe that when the percentage of conformance of the misbehaving node is small, it obtains a high share of network bandwidth when correction scheme is not enabled. On the other hand, when using the correction scheme, the throughput obtained by misbehaving node is close to the average throughput. Average throughput provides an estimate of the *fair* share of throughput a well-behaved node should receive. So the correction scheme is fairly effective in restricting the misbehaving node to its fair share. The average throughput increases with increasing misbehavior as the misbehaving node obtains most of the channel bandwidth and transmits packets after waiting for a fraction of the assigned backoff value. Consequently, the overhead of backing off is reduced and the average network throughput increases (as the average throughput includes the throughput obtained by misbehaving node).

The correction scheme is not effective when percentage of conformance tends to zero. Percentage of conformance tending to zero implies that the misbehaving node always picks small backoff values irrespective of the assigned backoff value. Hence, adding penalty to the assigned backoff value is not effective. But, this type of misbehavior can be detected with high accuracy and higher layers can be informed of the misbehavior. This situation is analogous to the *jamming* of channel as the misbehaving node transmits packets without following the backoff protocol.

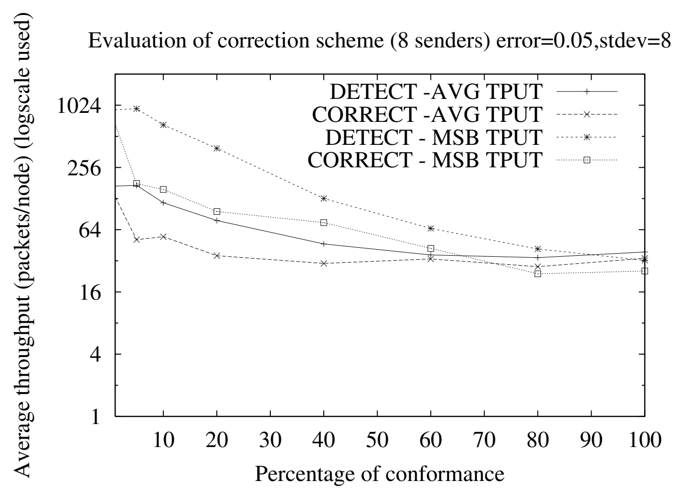

Fig. 9. Comparison of throughput obtained by misbehaving nodes with and without correction scheme for varying percentage of conformance

D. Simulation results for ad hoc networks

The simulation setup consists of 20 nodes randomly placed in a 700m X 700m square. Every node sets up CBR connections with all its neighbors. For a node X, all nodes within a circle of radius 250 meters around X are designated as its neighbors. With the shadowing model, neighbors within the 250 meters of node X may occasionally fail to receive a transmission while nodes further away might receive the transmission. We randomly select one node among the 20 nodes as the misbehaving node. The bit error rate is 0.001 and standard deviation of shadowing model is $\sigma_{dB} = 8$. Other parameters of the simulation are the same as that of the infrastructure-based network.

In an ad hoc network, it is difficult to estimate the *fair* share which the misbehaving node should receive after correction. In the case of infrastructure-based networks, we approximated this to the average throughput obtained per sender. However, in ad hoc networks, different nodes in the network have a different number of neighboring nodes contending for the channel and so the average throughput is no longer an indicator of the *fair* share. Therefore, we also run each simulation in the BASE setting (described in the previous section) with the designated misbehaving node (the randomly selected misbehaving node) conforming completely to the protocol. The throughput obtained by the misbehaving node in this setting is an estimate of the throughput that the node would obtain using IEEE 802.11 protocol, if it were well-behaved.

Figure 10 compares the average throughput obtained by all the nodes in the BASE, CORRECT and DETECT settings for varying Percentage of Conformance. We can see from the figure that the average throughput obtained when using the *correction* scheme (CORRECT curve) is close to that obtained by IEEE 802.11 protocol (BASE curve). When the Percentage of Conformance is 100% (i.e., the designated misbehaving node fully conforms to the backoff protocol), the throughput obtained by all the nodes with the CORRECT setting is slightly higher than that obtained using the BASE setting. So, the *correction* scheme does not degrade the network performance.

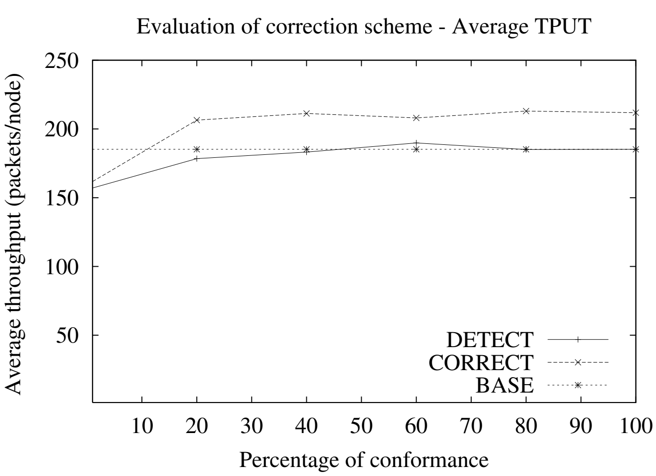

Fig. 10. Comparison of average throughput obtained over all nodes with and without correction scheme for varying percentage of conformance

Figure 11 compares the throughput obtained by the misbehaving node in BASE setting with the DETECT (only detection scheme is enabled) and CORRECT (both detection and correction are performed) settings. We observe from the figure that when the *correction* scheme is not used, the misbehaving node obtains a significant throughput advantage. When the *correction* scheme is used, the throughput obtained by the misbehaving node is restricted to less than that it would have obtained using IEEE 802.11 protocol, so long as the "Percentage of Conformance" is not very low. Thus, node misbehavior is effectively penalized.

As shown in Figure 11, the throughput obtained by the designated misbehaving node while fully conforming (i.e., Percentage of Conformance is 100%) is lower when using the *correction* scheme (CORRECT setting) than when using the IEEE 802.11 protocol (BASE setting). On the other hand, the average throughput over all the sender nodes (as seen in Figure 10) when using the *correction* scheme, is comparable with that obtained using the IEEE 802.11 protocol. Preliminary analysis of the trace data suggests that the reduction in the throughput of the designated misbehaving node, when fully conforming (when using the *correction* scheme), may be due to a different distribution of throughput[2] (among the nodes in the network) in comparison to the distribution of throughput when using the IEEE 802.11 protocol. We plan to investigate further the possible reasons for the reduction in the throughput obtained by a misbehaving node using the *correction* scheme.

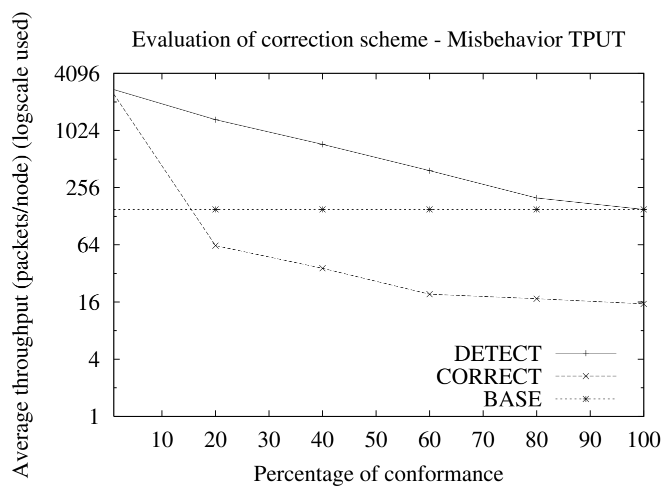

Fig. 11. Comparison of throughput obtained by the misbehaving node with and without correction scheme for varying percentage of conformance

VIII. Extensions for handling Receiver Misbehavior

In this section we present extensions to our protocol to handle receiver misbehavior. We leave the analysis of the extensions by simulation for future work.

The types of receiver misbehavior that might arise are:

[2]The correction scheme adds a penalty for every deviation identified by the monitor. With varying channel conditions, some transmissions from the conforming nodes may be wrongly designated as deviations, and the nodes may be penalized. Thus, a conforming node may obtain different throughput when using the *correction* scheme than when using IEEE 802.11 protocol.

Case 1: The receiver does not respond to a sender's RTS with a CTS even when the channel is idle. The receiver might indulge in such misbehavior to avoid receiving packets from certain nodes. For example, a receiver may avoid receiving packets that need to be forwarded to other nodes as part of the routing service. However, a receiver misbehaving in this fashion will have to decide whether a received RTS is for a DATA packet that has to be forwarded or for a DATA packet meant for the receiver itself. Deciding the type of the DATA packet based on the received RTS is expected to be non-trivial and therefore the above mentioned misbehavior may not arise often in practice. This type of misbehavior may be handled with payment schemes that provide incentives to nodes to receive and forward packets. Such an approach could be implemented at the network layer as proposed by Buttyan *et al.* in [7].

Case 2: The receiver might misbehave by adding large penalty to some nodes when *correction* scheme is enabled. This misbehavior is a consequence of using our *correction* scheme. One possible solution is to use the *correction* scheme only in scenarios where receivers are trusted (Example: Infrastructure-based networks with trusted base stations). In the absence of trusted receivers, we can use our protocol with *correction* scheme turned off to provide *detection* of misbehavior and rely on higher layer protocols to handle the misbehavior.

Case 3: The receiver might misbehave by assigning small backoffs to preferred nodes and larger backoffs to other nodes. Even with the *correction* scheme turned off, the receiver might misbehave in assigning the initial backoff values. For example, a receiver might assign backoff values from range [0,CW_{min}] with mean less than $\frac{CW_{min}}{2}$ for preferred nodes and with mean greater than $\frac{CW_{min}}{2}$ for other nodes.

To handle misbehavior mentioned above in case 3, we can apply the approach used in our protocol for detecting sender misbehavior. Specifically, we can use a deterministic function g such that the backoff value assigned by the receiver for the $(i+1)^{th}$ transmission from the sender is a function g of the backoff value assigned by the receiver for the i^{th} transmission from the sender. The function g can be selected such that the mean value of backoffs generated is $\frac{CW_{min}}{2}$. Using this approach, the sender or any other node which can monitor the values assigned by the receiver can validate those assigned values. A potential problem with this approach is that if two different senders are assigned the same backoff values at some point of time, then the subsequent sequence of backoff values assigned to them will be the same, possibly increasing the chances of collision.

An alternate approach for handling misbehavior mentioned above in case 3 is to bound the variation in the sum of backoffs assigned by the receiver by a constant δ. The receiver maintains a variable S to count the sum of assigned backoffs to a sender starting from the first transmission. When the backoff value for transmission i is assigned, the sum S is required to be within δ of the mean value of the sum of backoff values ($i * \frac{CW_{min}}{2}$) i.e.,

$$\frac{CW_{min}}{2} * i - \delta \leq S \leq \frac{CW_{min}}{2} * i + \delta \ , \ \forall i$$

The receiver is free to use any random number generator and generate backoff values subject to the above restriction. Using this scheme, the sender can easily verify that the backoff values assigned by the receiver satisfy the above condition. An useful property of this approach is that the sum of the backoff values assigned by the receiver for a sequence of successive transmissions (i to j, $j > i$) is within $2\,\delta$ of the mean for those transmissions ($(j-i+1) * \frac{CW_{min}}{2}$). Thus, the receiver can deviate by at most $2\,\delta$ from the mean while assigning backoff values. This property can be used by any third-party node monitoring the receiver to validate the behavior of the receiver by observing the assigned backoffs in a successive sequence of transmissions.

We can use a monitor in a third-party node to monitor the behavior of both the sender and receiver. The scheme presented above can be used to detect receiver misbehavior and the *detection* scheme presented in section IV-B can be used to detect sender misbehavior. Together, these two schemes provide a mechanism for a third-party node to detect collusion between a sender and a receiver.

IX. Conclusion and Future Work

Handling MAC layer misbehavior is an important requirement in guaranteeing service availability. In this paper, we have presented a MAC protocol which simplifies misbehavior detection. Simulation results have indicated that our detection scheme provides fairly accurate misbehavior detection. The correction scheme we have proposed is effective in restricting the throughput of greedy misbehaving nodes to a fair share.

The above approach cannot detect a misbehaving node that gains more bandwidth by using multiple MAC addresses. Detection of such misbehavior may require the support of higher layers and we intend to explore approaches for handling such misbehavior. We have also presented preliminary solutions for handling receiver misbehavior and collusion between senders and receivers. We plan to further develop these solutions and evaluate them in scenarios having misbehaving receivers and collusion between senders and receivers. We will also explore other approaches for detecting and correcting misbehavior.

ACKNOWLEDGEMENTS

We thank Xue Yang and Romit Roy Choudhury for their useful comments and suggestions.

REFERENCES

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

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

[3] Lidong Zhou and Zygmunt J. Haas, "Securing Ad Hoc Networks," *IEEE Network*, vol. 13, no. 6, pp. 24–30, 1999.

[4] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker, "Mitigating Routing Misbehavior in Mobile Ad hoc Networks," in *Mobile Computing and Networking*, 2000, pp. 255–265.

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

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

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

[8] Sonja Buchegger and Jean-Yves 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, June 2002, IEEE.

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

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

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

[12] L. Buttyan and J. P. Hubaux, "Rational Exchange – A Formal Model Based on Game Theory," in *Proceedings of 2nd International Workshop on Electronic Commerce (WELCOM 2001)*, Heidelberg, Germany, November 2001.

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

[14] Federico Cali, Marco Conti, and Enrico 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.

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

[16] L. Buttyan and J. P. 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.

[17] J. P. 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.

[18] Frank Stajano and Ross Anderson, "The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks," pp. 172–194.

[19] Donald E. Knuth, *The Art of Computer Programming*, vol. 2, chapter 3, pp. 10–17, Addison-Wesley, third edition, 2000.

APPENDIX I
ANALYSIS OF COLLISIONS SUFFERED BY MISBEHAVING NODES

In this appendix, we estimate the fewer number of collisions that a misbehaving node suffers in comparison to a well-behaved node. We model the channel access behavior of nodes by *access probabilities*. We follow the approach used by Cali *et al.* [14] 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* nodes in the network (N varies with time). Active nodes are those that have a packet to send to the base station at the instant under consideration. Each conforming active node 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 node 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 node suffers while transmitting a packet. In [14] 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 nodes in the network. It has been shown in [14] that the results are fairly accurate even for small networks containing around 10 nodes.

We analyze the behavior of a misbehaving node, 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 nodes in the network are well-behaved. When MB transmits a packet, the transmission is successful if no other node 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 node 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 node is

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

Since the misbehaving node backs off for some fraction of its assigned backoff value, its access probability is higher than that of the conforming nodes. 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 nodes suffer more collisions per successful transmission than the misbehaving node. The difference in number of collisions must be accounted in the misbehavior correction mechanism. This is given by

$$\begin{aligned} E_{diff} &= E_{conf} - E_{mb} && (1) \\ &= \frac{1}{(1-p_{mb})(1-p)^{N-2}} - \frac{1}{(1-p)^{N-1}} && (2) \\ &= \frac{1}{(1-p_{mb})(1-p)^{N-1}} * (p_{mb} - p) && (3) \end{aligned}$$

The next step is to compute this value. In practice, when we are computing the penalty to be added, we know that B_{obsr} (the actual time the node under observation has backed off for) is smaller than $B_{exp} * \alpha$ (B_{exp} is the expected backoff). By our model, the access probability of a node is inversely proportional to its average contention window value. When the misbehaving node waits for a fraction of its assigned backoff value, this behavior is equivalent to the misbehaving node waiting completely for a backoff value which is selected from a smaller range (in comparison with other conforming nodes). 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 nodes and well-behaved nodes when the *correction* scheme is enabled. Our aim is to compute the benefit gained by the misbehaving node for the observed deviation from protocol. If the misbehaving node persists with misbehavior, then its average total backoff will be proportional to B_{obsr} while the average total backoff (ATB) of well-behaved nodes will be proportional to B_{exp}. Assuming persistent misbehavior, the ratio of access probabilities of well-behaved node to misbehaving node is given by,

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

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 node behavior. Using this substitution, we can rewrite equation 3 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 node in the long run to be roughly equal to the access probability of conforming nodes, thus $p_{mb} \approx p$. Using this approximation, the number of collisions that the misbehaving node has saved by backing off for less than the assigned value is given by,

$$E_{diff} \approx \frac{p * (1-\beta)}{P(\text{Slot is idle})} \tag{4}$$

This is one of the several possible ways of estimating E_{diff} starting from equation 3. We can directly estimate p and N using the approach of Cali *et al.* in [14] and compute E_{diff} from that. However, our approach is simpler to implement.

We use the *idleCounter* described in section IV-B 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 node 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

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

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 node 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 monitor 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 of p, β and P_{idle}, E_{diff} is computed by substituting these values in equation 4.

Appendix II
Function for computing Backoff values after Collision

In our simulations, the function f used by the sender for computing backoff values for retransmission attempt is given by,

$$f(\textit{nextBackoff,nodeID,attemptNum}) = \\ (aX + c)\ mod\ (CW_{min}+1)$$

where $a = 5$, $c = 2 * \textit{attemptNum} + 1$ and X is given by

$$X = (\textit{nextBackoff} + \textit{nodeId})\ mod\ (CW_{min}+1)$$

The function f generates a number between [0,CW_{min}] and dividing the number by CW_{min} gives the required number between 0 and 1.

Knuth [19] discusses the various issues in generating uniform random numbers. A sequence of random numbers $< X_n >$ can be generated as follows,

$$X_{n+1} = (a\,X_n + c) mod\, m \ \ , \ \ n \geq 0$$

Such a sequence has a period of length m if and only if (details are in [19])

- c is relatively prime to m
- $a-1$ is a multiple of p for every prime dividing m
- $a-1$ is a multiple of 4 if m is a multiple of 4.

We have defined $m = CW_{min}+1$ in our function f. According to IEEE 802.11 standard, $CW_{min} = 2^k - 1$ for some k, depending on the physical layer characteristics. Hence, m is a power of 2. Consequently, using $a = 5$ and setting c to an odd number, we satisfy the conditions for generating a sequence of length m.

When the generated sequence is of length m, we are assured that if different X_n values chosen (between [0,$m-1$]), different X_{n+1} values are obtained. In addition, if the set of X_n values includes all numbers in the range [0,$m-1$], then the set of X_{n+1} values also includes all numbers in the range [0,$m-1$]. In our function f, we set X_n to the sum of *nextBackoff* and *nodeId* modulo $CW_{min}+1$. Thus, for a single node (having a constant node identifier), when the *nextBackoff* values are selected uniformly from the range [0,CW_{min}], the output values of f are also uniformly distributed in the range [0,CW_{min}]. Dividing this by CW_{min}, we get uniform random numbers between 0 and 1.

Other properties of function f are :

- If two nodes have different node identifiers (or more specifically, *nodeId mod* ($CW_{min}+1$) is different) then, the output of f is different for the same input *nextBackoff* values. Hence, after a collision, the probability of the new backoff values (used for retransmission) being different is high even if the initially assigned backoff values are the same.
- For a single node, different retransmission attempts will use different backoff values as the parameter c used by function f in generating the backoff values is dependent on the *attemptNumber* of retransmission.
- When the *nextBackoff* values are uniformly selected from range [0,CW_{min}], the output of f is also uniformly distributed in the same range with mean of $\frac{CW_{min}}{2}$. Thus, when the output of function f is divided by CW_{min} to obtain random numbers in range 0 to 1, the mean value of the generated random numbers will be 0.5.

A drawback of the function f used above is that it can only generate CW_{min} distinct random values irrespective of the transmission attempt number used as a parameter to the function. However, in IEEE 802.11 protocol, the number of distinct random values is a function of CW (CW is increased for successive retransmissions subject to a maximum of CW_{max}). Consequently, the backoff values selected using the function f for retransmission in our proposed protocol are not as "*random*" as the values selected in IEEE 802.11 protocol.

One way to overcome this is to set $m = CW_{max}+1$ and generate the initially assigned *nextBackoff* values from the range [0,CW_{max}] instead of [0,CW_{min}] (This approach was not used in the simulations presented in this paper.) The sender node can use [*nextBackoff mod* CW_{max}] for first retransmission instead of using *nextBackoff*, and subsequent retransmissions can continue to follow the rules specified in our proposed protocol. With this modification, the function f can generate CW_{max} distinct random values.