# On Intrusion Detection and Response for Mobile Ad Hoc Networks

James Parker, Jeffrey Undercoffer, John Pinkston, and Anupam Joshi
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
1000 Hilltop Circle, Baltimore, MD 21250

## Abstract

*We present network intrusion detection (ID) mechanisms that rely upon packet snooping to detect aberrant behavior in mobile ad hoc networks. Our extensions, which are applicable to several mobile ad hoc routing protocols, offer two response mechanisms, passive – to singularly determine if a node is intrusive and act to protect itself from attacks, or active – to collaboratively determine if a node is intrusive and act to protect all of the nodes of an ad-hoc cluster. We have implemented our extensions using the GloMoSim simulator and detail their efficacy under a variety of operational conditions.*

## 1 Introduction

Mobile ad hoc networks (MANETs) are fundamentally different from their wired-side counterparts. MANETs provide no fixed infrastructure, base stations or switching centers. Moreover, the nodes of a MANET are computationally constrained and have limited power. The routing protocols utilized in MANETs are dependent on each node serving as a router. Examples of these routing protocols include: *AODV* [12], *DSR* [7], *ZRP* [4], and *TORA* [11] as well as cluster based optimizations as described in [6], [8] and [9].

The nature of MANETs not only introduces new security concerns but also exacerbates the problem of detecting and preventing aberrant behavior. Whereas in a wired network an intruder could be a host that is either inside or outside of the network and could be subjected to varying degrees of access control and authentication, in a MANET, an intruder is part of the network infrastructure. Moreover, at the outset, an intruder in a MANET could be a trusted and integral component of the network infrastructure and only later exhibit aberrant behavior. Message mis-routing and message modification are the primary concerns in MANETs [13].

Existing ID for MANETs capitalize on the collaborate nature of mobile ad-hoc routing. These mechanisms rely upon promiscuous packet snooping to detect the mishandling of data in mobile ad-hoc networks. Our work improves and enhances existing mechanisms. Our research also revealed that the routing protocols typically employed by mobile ad-hoc networks lack sufficient functionality to enable robust ID, hence we have added modules that provide the necessary functionality. These modules are applicable to all of the routing protocols used in MANETs, not just *DSR*.

Snooping protocols leverage two properties inherent in most mobile ad hoc protocols. The first property is that each node in the network maintains a list containing the addresses of those nodes with which it is in immediate proximity or on the path from a source to a destination. The second property, as is the case in the 802.11 [5] and *MACAW* [2] link layer protocols, is that a node is able to "hear" the *RTS/CTS* negotiation of its neighbors. Accordingly, each node that participates in the intrusion detection process *"snoops"* on its neighbor's transmissions in order to ensure that they have not been modified or mis-routed. The notion of "snooping" is also employed in *DSR*, which is used for "reflecting shorter routes" as an optimization of the route maintenance process.

In our extension, which is viable for DSR and other ad hoc routing protocols, the snooping nodes listen to all other nodes in their proximity. This is in sharp contrast to both *Watchdog* [10] and *Neighborhood Watch* [3], which only work with *DSR*, watching the forward node on the patch from source to destination. We have experimented with, and provide detailed results for, two response mechanisms. The *passive* response mode, where a node, upon determining that another node is aberrant, will unilaterally cease interaction with that node. Although each node acts independently, eventually the intrusive node will be blocked from using all network resources. In the *active* response mode, each node relies upon a *Cluster Based* hierarchy. When a node detects an aberrant neighbor it informs its *Cluster Head*, who in turn initiates a voting procedure. If the majority determines that the suspected node is in fact intrusive, an alert will be broadcast throughout the network and the intrusive node will be denied network resources.

The remainder of this paper is organized as follows: Section 2 details related work. Section 3 details our extension and implementation of the ID Protocol for MANETs. Section 4 presents the intrusion response mechanism. Section

5 details our addition to *AODV*'s *neighbor table* so that it would support our needs. In Section 6 we detail our experiments and their results and we conclude in Section 7 with future work.

## 2 Related Work

*Watchdog*, introduced by Marti et al. was the first snooping ID protocol for MANETs . *Watchdog* relies upon *DSR* and each node participates by "watching" its downstream node, on the route from source to destination, to ensure that it has re-transmitted the packet without modification. Marti et al. hold that if source routing is not used then a misbehaving node could simply broadcast to a non-existent node to fool the watchdog. While this is true, packet modification is not covered up by simply broadcasting to a non-existent node. To mitigate the effects of a misbehaving node, Marti et al. introduce *Pathrater*, which selects a path from source to destination based upon a "reliability" metric, instead of the shortest path. This approach, as observed in [3] relieves the malicious node from the requirement of participating in the routing process, which may be construed as a reward.

Buchegger and Le Boudec [10] build upon Marti et al.'s work by replacing *Watchdog* with *Neighborhood Watch*, which is also dependent upon *DSR*, and snoops its downstream neighbor. They introduce a *Trust Manager*, *Reputation System*, and a *Path Manager*. Essentially each node is required to run a finite state machine to calculate trust, which in turn is used to rank the other node's reputation and then determine routes with the highest security metric. Buchegger and Le Boudec seemingly did not consider the resource constraints imposed upon most mobile ad hoc devices, nor did they provide analysis of their protocol with respect to network performance.

We believe our work extends both of these efforts by expanding the malicious detection to collaborate with routing protocols other than DSR, and offering a more robust identification procedure of malicious activity in our cluster voting scheme.

## 3 Snooping Protocol Extensions

We assume the presence of symmetric omni-directional links within the ad hoc network. When a node that is not on the path from source to destination is able to hear transmissions of two intermediate nodes, $A$ and $B$, that are on the *source route*, it becomes a snooping or *monitor* node for node $B$. Its ID function is to ensure that node $B$ does not alter the contents of the packet or misroute the packet. It accomplishes this by comparing certain information contained in the packet $p$ as it is inbound to intermediate node $B$ with the same information as contained in packet $p'$ as it is outbound from node $B$.

This section details the data structures and algorithms maintained and executed at each node to facilitate our Intrusion Detection and Response Protocol.

Figure 1 illustrates the scenario where *node B* snoops on *Node 3* by examining packet $p$ as it is inbound to *node 3* from *node 2* and packet $p'$ as it is outbound from *node 3* to *node 4*.

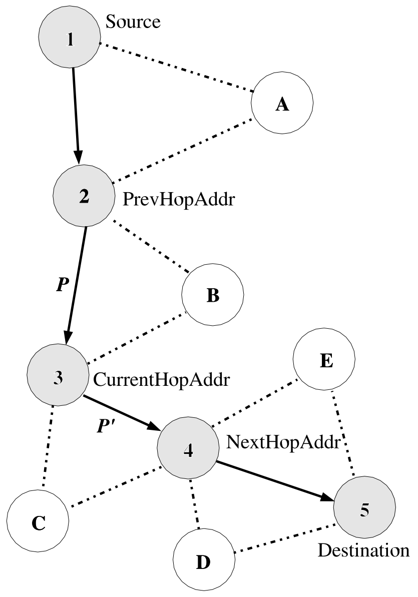

**Figure 1. Node A snoops on Node 2, Node B snoops on Node 3, etc.**

### 3.1 Data Structures

Each node employing our ID Protocol maintains four data structures: *ID Snoop Table*, *IDStatus Table*, *BadNode Table* and *Threshold Table*. For every packet snooped the monitoring node makes an entry in its $ID\ Snoop\ Table$, recording information which will be used to detect intrusions. An entry in the $ID\ Snoop\ Table$ is uniquely identified by the $SrcAddr$, $DstAdrr$ and the $PacketSeqNumber$. Referring to Figure 1, when node $B$ is snooping on node 3, node 2 is the $PrevHopAddr$, node 3 is the $CurrentHopAddr$ and node 4 is the $NextHopAddr$.

The monitoring node creates an entry in the *IDStatus Table* for every node that it is snooping upon. This table contains the total number of times that the monitoring node has detected an intrusion of a particular class for a particular node.

The *Threshold Table* holds threshold values for the attack classes. When a node exceeds the threshold value for a particular attack class, the protocol assumes that the anomalous behavior displayed by the node is in fact malicious and that link errors are not the cause of anomalies.

The *BadNode Table* holds the address of nodes that have been deemed to be intrusive. Whenever a node receives any packet or request from a node that is listed in the *BadNode Table* that request or packet is ignored. This effectively denies the intrusive node access to any resources in the MANET.

## 3.2 Algorithms

The *Snoop* (*Packet p*) method forms the core of the intrusion detection protocol. All packets that the monitoring node receives (that are not explicitly addressed to the monitor) are passed to this method. Accordingly, the *Snoop* algorithm does one of the following:

1. Ignores non-data carrying packets.
2. If a Route Error Packet is detected, **AND** the *ID Snoop Table* contains an entry (or entries) for the node being reported as *unreachable*, the entry (or entries) are removed from the *ID Snoop Table*.
3. If there is an entry for the packet in the *ID Snoop Table* (implying that the packet was snooped during its previous hop), **AND** the *CurrentHopAddr* is also in the node's *Neighbor Table* then run $PerformID$ (*Packet p*) and $MakeEntry$ (*Packet p*) on the packet.
4. If there is an entry for the packet in the *ID Snoop Table*, implying that the packet was snooped during its previous hop, run $PerformID$ (*Packet p*) on the packet.
5. If there is no entry for the packet in the *ID Snoop Table* **AND** if the monitor node has an entry for the next hop recipient in its neighbor table (implying that it will be able to hear the next hop relay the packet) make an entry in the *ID Snoop Table* by running $MakeEntry$ (*Packet p*), on the packet.
6. If there is no entry for the packet in the *ID Snoop Table* **AND** if the monitor node does not have an entry for the next hop recipient in its neighbor table, drop the packet.

As stated, *Snoop*() calls the $MakeEntry()$ and $PerformID()$. $MakeEntry()$ creates an entry in the *ID Snoop Table*, storing the relevant header and routing information. The $TIMEtamp$ field in the *ID Snoop Table* is used to detect message mis-routing attacks where the node fails to forward the packet and to clear the table of "old " entries.

The $PerformID(p')$ method tests for message modification attacks and message mis-routing attacks. It does so by comparing the entry in the *ID Snoop Table* derived from the inbound packet $p$ to information derived from the outbound packet, $p'$. To test for a suspected message modification attack, $PerformID(p')$ compares the checksum in the packet $p'$ to that which was in packet $p$, as is recorded in the checksum field of the corresponding entry in the *ID Snoop Table*. To test for the altered route mis-routing attack it ensures the route specified in the route path was followed. $PerformID()$ makes entries into the $IDStatusTable$ when it detects a node exhibiting anomalous behavior.

To test for message mis-routing attacks, for each time period $T$ the monitor node calls the $PerformMisRoute()$ method to test for entries in the *ID Snoop Table* that have exceeded the *TimeOutPeriod* specified in the *Threshold Table*. Whenever a node displays anomalous behavior by dropping or mis-routing a packet the *MisRouteCount* entry in that node's entry in the $ID\ Status\ Table$ is incremented.

In addition to testing for message mis-routing attacks, $PerformMisRoute()$ also clears old entries in the *ID Snoop Table*. If, in the event that a node moves out of range of its "monitoring" node after it has received a packet but before it forwards the packet it could appear to the monitoring node that the packet was maliciously dropped. $PerformMisRoute()$ tests and corrects for this condition.

Depending on which approach is being used (*passive* or *active*), the $RAISEAlarm()$ method in the above algorithm results in one of two different responses. We detail the responses in the following section.

## 4 Response to Intrusions

Our ID protocol allows for either an active or passive response to intrusions. With either response, the outcome is the isolation of the offending node from the network. In the passive mode a node makes a unilateral decision based on its own observations of anomalous behavior. In the passive mode the more frequent and aberrant the behavior on the part of an intrusive node, the sooner the intrusive node will be isolated and denied access to the underlying network infrastructure.

The active response mode offers a higher level of assurance than does the passive mode. The increased assurance level is due to a majority voting scheme and consequently the flooding of the intrusive node's identity throughout the network. The active mode, however, is more complex to implement.

### 4.1 Passive Response

Once the *threshold value*, which mitigates the effects of link error, for message mis-routing or message modification has been exceeded, an alarm is raised. In the passive mode, the node that raised the alarm removes the intrusive node from its *Neighbor Table* and will no longer participate in *route discoveries*, *Hello Messages* or collaborative routing with the intrusive node. Additionally, the intrusive node's address is recorded in the *BadNode Table*. As we will show in the section detailing our experiments, the more dense the network, the more nodes that simultaneously declare a node intrusive and prevent the malicious node from utilizing network resources. If the node in question continues to act intrusively each node in the network will eventually make a unilateral decision to disassociate itself with the intruder.

### 4.2 Active Response

Tay et al. [6] propose the *Cluster Based Routing Protocol* (CBRP) where nodes form clusters, each with an elected cluster head. The role of the cluster head is to optimize the route discovery process. We utilize the cluster heads to enable a voting protocol and active responses to intrusions.

When a node raises an alarm it forwards that alarm to all of its cluster heads. In turn, the cluster heads initiate the voting scheme described below. It is important that no node be able to spoof identities of other nodes, as this will enable it to foil the voting scheme by generating spurious votes. Accordingly, we assume that some kind of mechanism to authenticate each node is available. Secondly the voting scheme may fail if the majority of the cluster heads are in fact malicious nodes. If this were to be the situation, the malicious cluster heads could vote in an incorrect manner and foil the protocol. However, we feel that the likelihood of malicious nodes being elected as cluster heads to the majority of the clusters is relatively small.

**4.2.1 Data Structures.** Each cluster head participating in the voting scheme is required to maintain the following four data structures: Neighbor Cluster Head Information, Two-hop Neighbor Information, Suspect Table, and Voting History Table. The first two data structures are available from the underlying *Cluster-Based Protocol* while the remaining two are exclusively used for the voting process.

The last two data structures are required to avoid multiple instances of the same voting process from being initiated for a single suspected node. These tables also prevent a single monitoring node from raising an alarm at different cluster heads and all of them voting positively based on the information obtained exclusively from a single monitoring node.

**4.2.2 The Voting Protocol.** The voting protocol employs two key strategies: Distributed Voting and Majority Voting. They are detailed as follows:

1. Distributed Voting: Whenever the voting process is initiated, all of the participating nodes send their votes to all other participating nodes. Each node, on receipt of the votes, decides locally the outcome of the vote. This avoids the need for a voting coordinator.
2. Majority Voting: Any vote is successful if a majority of the participating nodes vote positively.

**The Protocol:**

1. When the threshold is reached at a node, the node sends an alarm to all of its cluster heads. This alarm contains the identity of the monitoring node and the identity of the suspected node. If a node suspects its cluster head of being an intruder it will only send the alarm information to its alternative cluster head, if one exists. If the node does not have an alternative cluster head it will forward the alarm information to a cluster head that is two-hops away. This two-hop information is contained in its *Cluster Adjacency Table* as described in [6].
2. When a cluster head receives intrusion information it adds this information to its *Suspect Table*. This information is used to respond to voting requests from other cluster heads.
3. The cluster head checks the *Voting History* to ascertain if a vote is currently in process for this suspect node:
   (a) If the cluster head finds that a vote is in progress it does not initiate a new round of voting.
   (b) If no vote is currently in process for the suspect node the cluster head initiates the voting process. It sends a `VOTE-REQ` packet to all its neighboring cluster heads. The `VOTE-REQ` includes a list of cluster heads that are to participate in the vote, the identity of the suspect node, and the identity of the monitoring node which raised the alarm.
4. When a cluster head receives a `VOTE-REQ` containing the same suspect node for which it has just initiated a vote process, it resolves the conflict by giving preference to the initiator with the higher address. The non-initiating cluster heads vote in the following manner:
   (a) Vote positive if it finds an entry in the suspect table for the same suspected node but reported by a different monitoring node from that included in the `VOTE-REQ`.
   (b) Vote neutral if the suspected node is not in it's two hop neighborhood. This means that the suspected node is not a neighbor of any adjacent cluster. Hence this cluster head cannot judge the behavior of the suspected node.
   (c) Vote negative if the suspected node is in its two hop neighborhood but does not find an entry in the suspect table for that suspected node. This indicates the members of this cluster have not noticed anything malicious about the suspected node even though the suspected node is a neighbor to some of the members.
5. Every participating cluster head decides the outcome of the voting independently. The vote is positive if it has received a majority of votes in the affirmative, where a majority is calculated from the number of participating cluster heads listed in the original `VOTE-REQ`. Otherwise the vote is deemed to be negative.
6. If the vote is deemed positive at a cluster head it sends out a `FINAL-RESPONSE` packet which is flooded throughout the network. This `FINAL-RESPONSE` is to instruct all the nodes in the network to stop communicating with the malicious node. It includes the identity of the malicious node and a list of cluster heads that voted positive in the voting process.
7. A node in the network that is unaware of this process cannot arbitrarily trust a single `FINAL-RESPONSE` message because the message could have been sent by a malicious node as a denial-of-service attack. Hence a node, upon receiving the `FINAL-RESPONSE`, waits to receive the `FINAL-RESPONSE` from enough participating cluster heads to conclusively verify positive results.
8. Upon receiving `FINAL-RESPONSE` from all of the required cluster heades, a node enters the malicious node in its *BadNode Table.*

## 5 Protocol Modules

We have extended our base snooping algorithm to work with other routing schemes such as AODV. While each packet does not carry the route from source to destination, a snooping node can determine if the current hop is the final destination. This allows the snooping node to listen for the packet to be forwarded without modification.

Obviously, a mis-route can not be determined, but any modification to the packet, or packet dropping, can easily be determined and logged.

In order to implement the algorithms, two additional pieces of support code needed to be in place. It is important to recognize that performance of the ID algorithm is only as good as the underlying protocol that keeps track of the nodes current one hop neighbors. The only routing protocol in GloMoSim having a *Neighbor Table* was AODV. Unfortunately, the table was only updated when nodes are expected to route traffic. The fundamental basis for our algorithms is knowing current one hop neighbors in order to determine correct packet handling. The implementation of AODV's neighbor table was determined to be woefully inadequate for our purposes.

We chose to implement a Neighbor function that periodically sends `Hello` messages to announce its presence. The messages are received and tracked in a one hop *Neighbor Table*. If a node does not receive a `Hello` packet from one of its neighbors for three consecutive `Hello` periods, then the neighbor is assumed to have moved out of range and is removed from the *Neighbor Table*.

The second piece of code added was a dynamic clustering scheme based on the Distributed and Mobility-Adaptive Clustering (DMAC) algorithm as described in [1]. The algorithm was slightly modified to use the Neighbor function to determine changes in Clusters, and initiate the appropriate actions (i.e. new Cluster Head elections). It should be noted that we are using DMAC to maintain a cluster hierarchy for voting, and not as a routing protocol.

## 6 Experiments

The algorithms were simulated using GloMoSim version 2.03. We used the simulation environment detailed in [10] as a starting point. The following subsection details our simulation environment, metrics, and experimental results.

### 6.1 Simulation Environment

1. Grid Size: 2,000 by 2,000 meters.
2. Number of Nodes: 50 (16 nodes involved in constant bit rate (CBR) connections, and we varied the number of bad nodes).
3. Packet Traffic: 10 CBR connections are generated simultaneously, where 4 nodes are source for two streams each, and 2 nodes were the source for a single stream each; destination nodes only receive one CBR stream.
4. Mobility: Random Waypoint Model (max speed 20 meters per sec., pause time 15 sec.).
5. Routing Protocol: AODV or DSR.
6. MAC Layer: 802.11, peer-to-peer mode.
7. Radio: "no fading" radio model, with range of 376 meters.
8. Simulation Time: 200 sec.
9. Dropped Packet Time Out: 10 sec.
10. Dropped Packet Threshold: 10 packets.
11. Clear Delay (event expiration timer): 100 sec. (e.g.: the amount of time that a node considers an event without coming to a final determination).
12. Misroute Threshold: 5 events. Detectable only in routing protocols using Source Routes such as DSR.
13. Modification Threshold: 5 events.
14. Neighbor `Hello` period: 30 seconds.

### 6.2 Metrics

We measure False positives, True positives, and packet throughput each as a function of the percentage of bad nodes in the network. False positives and True positives are counted as a single tally for each node making the identification. By using this method there may be greater than 50 total False or True positives counted. All results are averaged over a number of simulation runs.

### 6.3 Results and Discussion

Results were obtained by averaging 100 simulation runs for 200 seconds each. The plots in Figure 2 show the true positives, false positives, and successfully delivered packets as a percentage of the number of bad nodes in the network for DSR (top row) and AODV (bottom row) respectively. Node density of both malicious and normal nodes is a significant factor in rates of true positives. For a malicious node to be detected, it must both act maliciously and be in proximity to a good node in order to be detected.

As expected, the performance of the both the Passive and Active response protocols improved, in respect to both True Positives and False Positives, as the density of the malicious nodes increased. Likewise, and as expected, the number of successfully delivered packets decreased as the density of malicious nodes increases. For the case of 0 bad nodes, this is attributable to the increased bandwidth for the current implementation of the voting mechanism.

According to [10] there are two contributing factors that influence the rate of false positives — speed and collisions. The node's speed can cause monitoring nodes to believe packets have been dropped, when the mobiles move out of range prior to packet relaying. Collisions at the monitoring node may also lead to a nodes failure to detect a packet relay.

## 7 Conclusions and Future Work

We have extended our base snooping algorithm to work with not only DSR, but also AODV. While each packet does not carry the route from source to destination in AODV, a snooping node can determine whether the current hop is the final destination. This allows the snooping node to listen for the packet to be forwarded without modification. Obviously, a mis-route can not be determined, but any modification to the packet, or dropping of the packet can easily be recognized and logged.

The implementation of both the Passive and Active ID algorithms in GloMoSim led to a number of parameters that can be adjusted. The data obtained for this paper was with our best guess at realistic values for these parameters. As future work, we will optimize these parameters to effect better performance as well as varying node density.

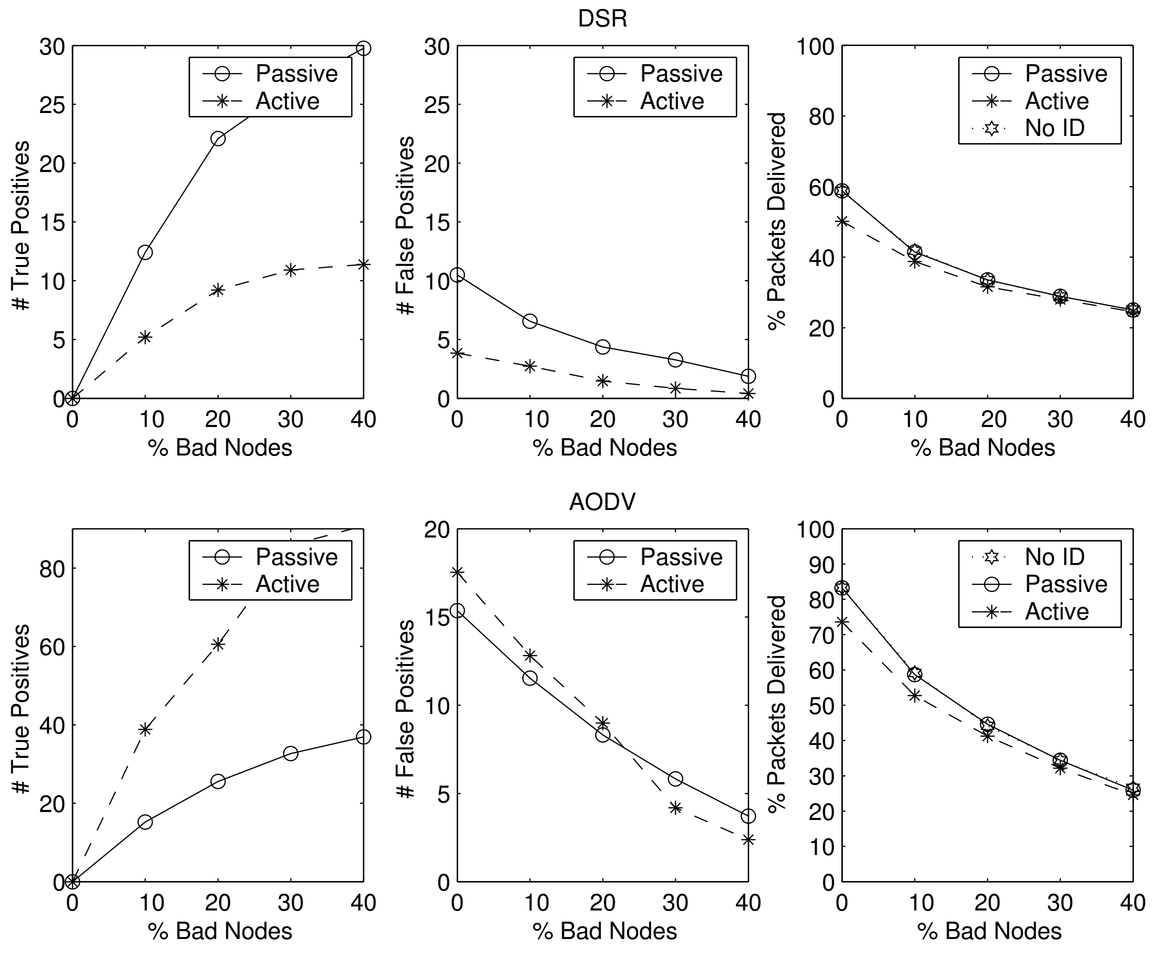

**Figure 2. Simulation Results of the Active and Passive Response Protocols for DSR (top) and AODV (bottom)**

## References

[1] Stefano Basagni. Distributed clustering for ad hoc networks. In *Parallel Architectures, Algorithms, and Networks*, pages 310–315. IEEE, June 1999.

[2] Vaduvur Bharghavan, Alan Demers, Scott Shenker, and Lixia Zhang. Macaw: a media access protocol for wireless lan's. In *Proceedings of the conference on Communications architectures, protocols and applications*, pages 212–225. ACM Press, 1994.

[3] Sonja Buchegger and Jean-Yves Le Boudex. Nodes bearing grudges: Towards routing security, fairness, and robustness in mobile ad hoc networks. In *Proceedings of the 1oth Euromicor Workshop on Parallel, Distributed and Network-based Processing*. IEEE Computer Society, 2002.

[4] Z. J. Haas and M. Perlman. The performance of query control schemes for zone routing protocol. In *SIGCOMM'98*, 1998.

[5] IEEE. *IEEE Std 802.11, 1999 Edition*, r2003 edition, 1999.

[6] Mingliang Jiang, Jinyang Li, and Y.C. Tay. *INTERNET-DRAFT : Cluster Based Routing Protocol (CBRP)*. IETF.

[7] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[8] P. Krishna and D.K Pradhan N.H. Vaidya, M. Chatterji. A cluster-based approach for routing in dynamic networks. Technical report, Department of Computer Science, TAMU, 2000.

[9] Ben Liang and Zygmunt J. Haas. Virtual backbone generation and maintenance in ad-hoc network mobility management. Technical report, School of Electrical Engineering, Cornell University, Ithaca, NY 14850, 2000.

[10] Sergio Marti, Thomas J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of MOBICOM*, pages 255 – 265, 2000.

[11] Vincent D. Park and M. Scott Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proceedings of IEEE INFOCOM '97*, 1997.

[12] Charles E. Perkins and E. M. Royer. Ad hoc on demand distance vector routing. In *IEEE WMCSA'99*, 1999.

[13] Lidong Zhou and Zygmunt J. Hass. Securing ad hoc networks.