MAC LAYER MISBEHAVIORS IN MOBILE AD HOC NETWORKS

LEI GUANG

A THESIS
IN
THE DEPARTMENT
OF
ELECTRICAL AND COMPUTER ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF APPLIED SCIENCE
CONCORDIA UNIVERSITY
MONTRÉAL, QUÉBEC, CANADA

JANUARY 2006

Your file Votre référence
ISBN: 0-494-14258-8
Our file Notre référence
ISBN: 0-494-14258-8

Abstract

MAC Layer Misbehaviors in Mobile Ad Hoc Networks

Lei Guang

MAC misbehavior in mobile ad hoc networks (MANET) is relatively new and unexplored in the literature. Different from traditional wireline networks, the unique nature of MANETs has opened new challenges in the network design. An Mobile *Ad Hoc* Network (MANET) is a collection of wireless mobile hosts forming a temporary network without the aid of any established infrastructure or centralized administration. In such an environment, one mobile host may need the aid of other hosts to communicate with the destination. However, failed data transmission due to selfish or malicious or hybrid node misbehavior and faulty nodes can severely degrade the overall network performance. Hence, a well-behaved node needs to adaptively change its cooperation level to cope with node misbehavior. In this thesis, we first briefly overview the existing security issues in ad hoc network routing and MAC layers. We explain security methods such as economic incentives, secure routing by cryptography and detection system against greedy behavior alleviate some of the problems, but not all. We classify a set of MAC misbehavior and measure their impacts on the ad hoc routing as well as the whole network performance. Based on the analysis of these vulnerabilities, we develop a new detection and reaction system (*DREAM*) to detect one category of node misbehavior *TO* attack and mitigate its negative effects on the network operation through two-stage reactions. Via computer simulation, we have proved the high efficiency of our system.

Acknowledgments

First and foremost, I wish to express my sincere thanks and gratitude to my thesis adviser, Professor Chadi M. Assi, for his expert guidance, and encouragement throughout my research work. Professor Assis gives me the opportunity to work on the amazing topic network security, my favorite since I was a high school student. His boundless enthusiasm and persistent commitment to high quality research has helped me polish every detail of my research papers. His technical advice and infinite patience were essential for the complete of this thesis. I am proud of having the opportunity to study under his supervision.

I must also thank Wei Huo, for his kind help and recommendation during my whole student life in Concordia University. Ahmad Dhaini and Farrukh Mahboob are my student colleagues who have taught me many things during my master study.

I thank all the members of Concordia Institute for Information Systems Engineering (CIISE) and Electrical and Computer Engineering Department (ECE) for creating such a dynamic and collaborative environment for research and study. I thank the office staff and the systems supporting staff in CIISE for all their kind assistance. I also thank the Faculty of Engineering and Computer Science at Concordia University, for their superb teaching and guidance.

Last, but not least, I would like to express my warmest thanks to Liping Xie, who has given me the best support with her endless perseverance, sacrifice and love. My deepest thanks go to my dear parents and sister for their self-giving love and supports throughout my life which make all these possible.

Contents

List of Figures

List of Tables

Abbreviations

ACK - Acknowledgment

AODV - Ad-hoc On-demand Distance Vector

AP - Access Point

ARP - Address Resolution Protocol

CBR - Constant Bit Rate

CGSR - Cluster Gateway Switching Routing

CTS - Clear to Send

CW - Contention Window

DCF - Distributed Coordination Function

DIFS - Distributed Inter Frame Space

DNS - Domain Name Service

DoS - Denial of Service

DSDV - Destination Sequenced Distance Vector

DSR - Dynamic Source Routing

EIFS - Extended Inter Frame Space

ERA - Ensure Randomness Algorithm

FSR - Fisheye State Routing

GPS - Global Position System

KSA - Key Setup Algorithm

MAC - Medium Access Control

MANET - Mobile Ad Hoc Network

MiM - Man in the Middle

MN% - Percentage of Misbehaving Nodes

MPDU - MAC Protocol Data Unit

MSDU - MAC Service Data Unit

NAV - Network Allocation Vector

PCF - Point Coordinator Function

PRGA - Pseudo-Random Generation Algorithm

PSK - Pre-shared key

RERR - Route Error

RREQ - Route Request

RREQ - Route Reply

RTS - Request to Send

SIFS - Short Inter Frame Space

SN - Suspect Node

TO - TimeOut

TORA - Temporally Ordered Routing Algorithm

UN - Untrusted Node

WEP - Wired Equivalent Privacy

WLAN - Wireless Local Area Network

WN - Well-behaved Node

WRP - Wireless Routing Protocol

ZRP - Zone-based Hierarchical Link-state Routing Protocol

Chapter 1

Introduction

In recent years, with the rapid development in wireless communication technologies and proliferation of mobile communication and computing devices, mobile ad hoc networks (MANETs) have emerged as an important part of the envisioned ubiquitous communication. A mobile ad hoc network is a self-configuring network of mobile routers connected by wireless links. The routers are free to move randomly and organize themselves arbitrarily and therefore the wireless network topology may change quickly and unpredictably. Such a network may operate in a standalone fashion, or may be connected to the larger Internet.

The earliest MANETs were called "packet radio" networks, and were sponsored by DARPA in the early 1970s. BBN(Bolt Beranek and Newman) and SRI(Stanford Research Institute) designed, built, and experimented with these earliest systems. It is interesting to note that these early packet radio systems predated the Internet, and indeed were part of the motivation of the original Internet Protocol suite. Later DARPA experiments included the Survivable Radio Network (SURAN) project, which took place in the 1980s. Another third wave of academic activity started in the mid 1990s with the advent of inexpensive 802.11 radio cards for personal computers. Current MANETs are designed primarily for military utility; examples include JTRS(Joint Tactical Radio System) and NTDR(Near-Term Digital Radio).

The popular IEEE 802.11 ("Wi-Fi") wireless protocol incorporates an ad-hoc networking system when no wireless access points are present, although it would be considered a very low-grade ad-hoc protocol by specialists in the field. The IEEE 802.11 [1] system only handles traffic within a local "cloud" of wireless devices. Each node transmits and receives data, but does not route anything between the network's systems. However, higher-level protocols can be used to aggregate various IEEE 802.11 ad-hoc networks into MANETs.

One of the most used protocols in IEEE 802.11 family is 802.11b, which operates at the 2.4GHz unlicensed frequency band and has a maximum bandwidth of 11Mbit/s. 802.11g is a recent and rapidly spreading development that is backwards-compatible with 802.11b, but has a headline data rate of 54Mbit/s. In order for multiple networks to share the same medium, thus having more than one wireless network in the same physical place, there are different communication channels that may be used, each with a different frequency band. Channels in 802.11b/g vary from 1 to 14 (2,412-2,484GHz) but have legal constraints on which subset of channels may be used. A final, though less prevalent variant is 802.11a, which operates around 5GHz and also has a maximum bandwidth of 54Mb/s.

1.1 Problem Statement

The increasingly widely deployed wireless networks have offered great convenience and commercial potential for our future life. However, like traditional networks, the spread of this kind of network may have been much faster than the diffusion of security knowledge.

It is foreseeable that wireless network will become a new playground for hackers who are attracted by the *open* nature of wireless networks: access from *anywhere* at *anytime*. Hence, security has become a primary concern to provide reliable communication between mobile stations in such an untrusted environment. Unlike wired networks, the unique characteristics of mobile ad hoc networks have posed a non-trivial challenges to the security design.

Ad hoc routing security and MAC layer security have attracted increasing research focus in recent years. People have proposed secure on-demand routing protocols, secure link-state routing protocols and reputation-based systems. In addition, several detection methods against MAC selfish misbehavior have also been developed. In this thesis, we will present some of the MAC layer vulnerabilities and study their impact on ad hoc network performance. We classify MAC misbehaviors as selfish and malicious and we present a new vulnerability that could be exploited by a host to either obtain unfair share of the network resources or todisrupt the network service; we present a detection tool and corresponding reaction mechanisms to mitigate its impact.

1.2 The Approach

The primary approach for this study was computer simulations. We used the network simulator NS-2 [17] developed by the VINT research group at University of California at Berkeley. The Monarch research group at Carnegie Mellon University extended the NS-2 simulator to include wireless scenarios with mobile nodes [2]. The most popular ad-hoc routing protocols were implemented in the CMU extension. Subsequent versions of the

CMU wireless extension also included energy models for the mobile nodes. From our own observations and comments of others in the ns-2 news groups, we learned that the AODV simulation goes unpredictably into infinite loops for certain scenarios. To cope with these bugs, we have added proper modifications to the original NS-2 models. We also developed our own models to simulate different attacking scenarios and corresponding detection and reaction methods. In addition, a set of tools have been made for statistical analysis.

1.3 Contributions of Thesis

The main contribution of the thesis are the following:

- We give a complete survey on the ad hoc networks MAC layer misbehavior. Implementation techniques of these misbehaviors as well as detection and handling schemes are presented.

- Via network simulation, we evaluate the ad hoc routing performance in the presence of different types of MAC misbehaviors. Consequently, non of these tested routing protocols are robust to MAC misbehaviors. Hence, rather than design separate detection systems for single network or MAC layer, it is necessary to develop secure routing protocols robust to MAC misbehaviors to provide cross-layer cooperation. These results are partially published in [19,21,22]

- We provide one prevention and detection method against MAC selfish misbehavior. We also design a detection and reaction system towards MAC malicious misbehavior. Our simulation results have proven that our approaches are efficient in detecting and handling MAC misbehaviors. These results are shown in [20]

1.4 Organization of Thesis

The rest of the thesis is organized as follows. In Chapter 2, we overview and compare two on-demand routing protocols AODV (Ad hoc On-demand Distance Vector) and DSR (Dynamic Source Routing) which are used for performance evaluation; we also present secure routing techniques in the literature. In Chapter 3, we first describe the IEEE 802.11 standard, then we focus on the illustration of our MAC layer misbehavior models. In Chapter 4, we evaluate the ad hoc routing performance in the presence of MAC misbehavior. Our proposed detection and reaction scheme against MAC misbehavior is described in Chapter 5. Finally, we summarize this thesis in Chapter 6.

Chapter 2

Routing in Mobile Ad Hoc Networks

2.1 Introduction

In MANETs, nodes establish communication outside of the transmission range of each other through multi-hop wireless links. The mobility and limited device resources, together with wireless transmission effects, such as attenuation, multipath propagation and interference have introduced unique routing operation issues in MANET in recent years. Extensive research work has been done on the design of ad hoc routing protocols, however, security in ad hoc routing is still an interesting topic. In this chapter, we first survey the current routing protocols in MANET in Section 2.2. Second, we describe the operation of two on-demand ad hoc routing protocols AODV and DSR that we will use for future evaluation in Section 2.3 and Section 2.4 respectively. Then we briefly compare these two protocols in Section 2.5. Section 2.6 reviews the attacks towards ad hoc routing mechanism and the present state of research in secure ad hoc routing protocols.

2.2 Review on Ad Hoc Routing Protocols

Ad hoc routing protocols are typically divided into two classes: *proactive* (table-driven) and *reactive* (on-demand).

Proactive routing protocols are derived from the traditional Internet distance vector and link state protocols. Each node maintains routing table to store routing information and any change in the network will trigger the propagation of update information throughout the whole network. This can cause substantial control traffic overhead, ultimately wasting the bandwidth and the limited device power of mobile nodes. In addition, in networks like MANET, these updated information might be already out of date when received by the nodes. The advantage, however, is that routes to any destination are always available without the need of route discovery; however, the performance of the proactive routing protocols will be severely affected when the mobility is high and the network size is large. Protocols in this school are different from the type of routing tables and the way of maintenance.

- **DSDV.** Destination Sequenced Distance Vector [39] routing protocol maintains a routing table that records the routes to every node in the network and simple change of network topology will require all the nodes to update their tables.
- **WRP.** Wireless Routing Protocol [34] triggers the updates of routing information upon the change of neighbors. Upon the change of its neighbor, the node will change the information of its routing table.
- **CGSR.** Cluster Gateway Switching Routing [14] aims at reducing the size of routing

table and the frequency to update routing information by dividing the network into clusters. Each cluster has a cluster head node to manage the exchange of routing information with other clusters through a gateway node. However, this method introduces a centralized control node which disobeys the distributed nature of ad hoc networks and additionally may cause great security flaws.

- **FSR.** Fisheye State Routing Protocol [38] differs from the previous protocols in that the updated frequency is inversely related to the distance between any two nodes.

On-demand routing protocols, as indicated by its name, initialize routing discovery only when a source needs to transmit data packets to a previously unknown destination. Basically the operations of on-demand routing protocols are divided into *route discovery* process and *route maintenance* process. Variants of on-demand routing protocols differ from the usage of the above two operations. This category of protocols outperforms the proactive ones in terms of their low routing overhead and quick response to the changes of network topology. However, it introduces relatively high overhead and latency during the routing discovery.

- **AODV.** Ad-hoc On-demand Distance Vector [41] is based on the change of DSDV with on-demand features.

- **DSR.** Dynamic Source Routing [10, 30] depends on the use of source routing and route cache. Each transmitted packet carries the whole route path to the destination and route maintenance is basically the maintenance of route cache.

- **TORA.** Temporally Ordered Routing Algorithm [37] discovers multiple paths from a source to a destination and re-initiates discovery only when all of them have failed.

In addition to proactive and active routing protocols, the ZRP (Zone-based Hierarchical Link-state Routing Protocol) [24] is an example of hybrid routing protocols. It tries to combine the advantage of both proactive and reactive routing protocols. Each node defines a zone that contains the number of nodes within certain hops. Proactive routing is used by a node to maintain routes within the zone, while active routing is triggered to discover routes outside the zone.

AODV, DSR, TORA and ZRP are currently the four ad hoc routing protocols that are under review by the IETF MANET workgroup as candidates for standardization. In this thesis, we will focus on the evaluation of two popular on-demand routing protocols: AODV and DSR. The operations of these two protocols are described in details in Section 2.3 and Section 2.4.

2.3 Ad Hoc On-demand Distance Vector Routing

2.3.1 Overview of AODV

The Ad hoc On-Demand Distance (AODV) routing algorithm enables dynamic, self starting, multi hop routing between participating mobile nodes wishing to establish and maintain an ad hoc network. AODV allows mobile nodes to quickly obtain routes for the new destinations, and does not require the nodes to maintain routes that are not active in communication. AODV allows the mobile nodes to respond to link breakage and changes in topology in a timely manner. AODV is loop-free, and offers quick convergence in the presence of network topology changes by avoiding the Bellman-Ford “counting to infinity”problem.

One distinguishing feature of AODV is its use of a destination sequence number for each route entry. The destination sequence number is created by the destination to be included along with any route information it sends to requesting nodes. Using destination sequence numbers ensures loop freed operation and is simple to program. Given the choice between two routes to a destination, a requesting node is required to select the one with the greatest sequence number.

2.3.2 Route Discovery

As long as the endpoints of a communication connection have valid routes to each other, AODV does not play any role. When a route to a new destination is needed, the source node broadcasts a RREQ to find a route to the destination. A route can be determined when the RREQ reaches either the destination itself, or an intermediate node with a 'fresh enough' route to the destination. A ‘fresh enough’ route is a valid route entry for the destination whose associated sequence number is at least as great as that contained in the RREQ. The route is made available by broadcast or unicasting a RREP back to the source originating the RREQ. Each node receiving the request caches a route back to the originator of the request, so that the RREP can be unicast from the destination along a path to that originator, or likewise from any intermediate node that is able to satisfy the request.

Nodes monitor the link status of next hops in active routes. When a link break on an active route is detected, a RERR message is used to notify other nodes that the loss of that link has occurred. The RERR message indicates those destinations which are no longer reachable by way of the broken link. In order to enable this reporting mechanism, each node keeps a "precursor list", containing the IP address for each of its neighbors that are likely to use it as a next hop towards each destination. The information in the precursor lists is most easily acquired during the processing for generation of a RREP message, which by definition has to be sent to a node in a precursor list.

AODV deals with route information by managing routing tables. Route table information must be kept even for short-lived routes, such as those created to temporarily store reverse paths towards nodes originating RREQs. Managing the sequence number is crucial to avoid routing loops, even when links break and a node is no longer reachable to supply its own information about its sequence number. A destination becomes unreachable when a link breaks or is deactivated. When these conditions occur, the route is invalidated by operations involving the sequence number and marking the route table entry state as invalid.

2.3.3 Route Maintenance

A node initiates processing for a Route Error (RERR) message in three situations:

1. if it detects a link break for the next hop of an active route in its routing table while transmitting data (and route repair, if attempted, was unsuccessful), or;
2. if it gets a data packet destined to a node for which it does not have an active route and is not repairing (if using local repair), or
3. if it receives a RERR from a neighbor for one or more active routes.

A RERR message may be either broadcast (if there are many precursors), unicast (if there is only 1 precursor), or iteratively unicast to all precursors (if broadcast is inappropriate). Even when the RERR message is iteratively unicast to several precursors, it is considered to be a single control message. The node receives a REER message can try to repair the broken link locally by initiating new RREQ to the destination. If there is no RREP received within certain period, the REER will be propagated to the upstream nodes until finally it reaches to the original source node of the data packet.

2.4 Dynamic Source Routing

2.4.1 Overview of DSR

The basic version of DSR uses explicit *source routing*, in which each data packet sent carries in its header the complete, ordered list of nodes through which the packet will pass. This use of explicit source routing allows the sender to select and control the routes used for its own packets, supports the use of multiple routes to any destination (for example, for load balancing), and allows a simple guarantee that the routes used are loop-free; by including this source route in the header of each data packet, other nodes forwarding or overhearing any of these packets can also easily cache this routing information for future use.

2.4.2 Route Discovery

When some source node originates a new packet addressed to some destination node, the source node places in the header of the packet a "source route" giving the sequence of hops that the packet is to follow on its way to the destination. Normally, the sender will obtain a suitable source route by searching its *Route Cache* of routes previously learned; if no route is found in its cache, it will initiate the Route Discovery protocol to dynamically find a new route to this destination node.

When another node receives this Route Request, if it is the target of the Route Discovery, it returns a *Route Reply* to the initiator of the Route Discovery, giving a copy of the accumulated route record from the Route Request. When the initiator receives this Route Reply, it caches this route in its Route Cache for use in sending subsequent packets to this destination.

Otherwise, if this node receiving the Route Request has recently seen another Route Request message from this initiator bearing this same request identification and target address, or if this node's own address is already listed in the route record in the Route Request, this node discards the Request. A node considers a Request recently seen if it still has information about that Request in its Route Request Table. Otherwise, this node appends its own address to the route record in the Route Request and propagates it by transmitting it as a local broadcast packet (with the same request identification).

In returning the Route Reply to the initiator of the Route Discovery, the destination will typically examine its own Route Cache for a route back to the source, and if found, will use it for the source route for delivery of the packet containing the Route Reply. Otherwise, the destination should perform its own Route Discovery for the source node, but to avoid possible infinite recursion of Route Discoveries, it must piggyback this Route Reply on the packet containing its own Route Request for the destination.

The destination could instead simply reverse the sequence of hops in the route record that it is trying to send in the Route Reply, and use this as the route on the packet carrying the Route Reply itself. For MAC protocols, such as IEEE 802.11, that require a bidirectional frame exchange as part of the MAC protocol [1], the discovered source route must be reversed in this way to return the Route Reply since it tests the discovered route to ensure it is bidirectional before the Route Discovery initiator begins using the route. This route reversal also avoids the overhead of a possible second Route Discovery.

When initiating a Route Discovery, the sending node saves a copy of the original packet (that triggered the Discovery) in a local buffer called the *Send Buffer*. The Send Buffer contains a copy of each packet that cannot be transmitted by this node because it does not yet have a source route to the packet's destination. Each packet in the Send Buffer is logically associated with the time that it was placed into the Send Buffer and is discarded after residing in the Send Buffer for some timeout period SendBufferTimeout; if necessary for preventing the Send Buffer from overflowing, a FIFO or other replacement strategy may also be used to evict packets even before they expire.

While a packet remains in the Send Buffer, the node should occasionally initiate a new Route Discovery for the packet's destination address. However, the node must limit the rate at which such new Route Discoveries for the same address are initiated, since it is possible that the destination node is not currently reachable. In particular, due to the limited wireless transmission range and the movement of the nodes in the network, the network may at times become partitioned, meaning that there is currently no sequence of nodes through which a packet could be forwarded to reach the destination. Depending on the movement pattern and the density of nodes in the network, such network partitions may be rare or may be common.

If a new Route Discovery was initiated for each packet sent by a node in such a partitioned network, a large number of unproductive Route Request packets would be propagated throughout the subset of the ad hoc network reachable from this node. In order to reduce the overhead from such Route Discoveries, a node should use an exponential back-off algorithm to limit the rate at which it initiates new Route Discoveries for the same target, doubling the timeout between each successive Discovery initiated for the same target. If the node attempts to send additional data packets to this same destination node more frequently than this limit, the subsequent packets should be buffered in the Send Buffer until a Route Reply is received giving a route to this destination, but the node must NOT initiate a new Route Discovery until the minimum allowable interval between new Route Discoveries for this target has been reached.

2.4.3 Route Maintenance

When originating or forwarding a packet using a source route, each node transmitting the packet is responsible for confirming that data can flow over the link from that node to the next hop.

An acknowledgment can provide confirmation that a link is capable of carrying data; In wireless networks, acknowledgments are often provided at no cost, either as an existing standard part of the MAC protocol in use (such as the link-layer acknowledgment frame defined by IEEE 802.11), or by a *passive acknowledgment* (in which, for example, node B confirms receipt at node C by overhearing C transmit the packet when forwarding it on to D).

If a built-in acknowledgment mechanism is not available, the node transmitting the packet can explicitly request a DSR-specific software acknowledgment be returned by the next node along the route. This software acknowledgment will normally be transmitted directly to the sending node, but if the link between these two nodes is unidirectional, this software acknowledgment could travel over a different, multi-hop path.

After an acknowledgment has been received from some neighbor, a node may choose to not require acknowledgments from that neighbor for a brief period of time, unless the network interface connecting a node to that neighbor always receives an acknowledgment in response to unicast traffic.

When a software acknowledgment is used, the acknowledgment request should be retransmitted up to a maximum number of times. A retransmission of the acknowledgment request can be sent as a separate packet, piggybacked on a retransmission of the original data packet, or piggybacked on any packet with the same next-hop destination that does not also contain a software acknowledgment.

After the acknowledgment request has been retransmitted the maximum number of times, if no acknowledgment has been received, then the sender treats the link to this next-hop destination as currently *broken*. It should remove this link from its Route Cache and should return a *Route Error* to each node that has sent a packet routed over that link since an acknowledgment was last received.

For sending such a retransmission or other packets to this same destination, if source has in its Route Cache another route to destination (for example, from additional Route Replies from its earlier Route Discovery, or from having overheard sufficient routing information from other packets), it can send the packet using the new route immediately. Otherwise, it should perform a new Route Discovery for this target.

2.5 Comparison between AODV and DSR

The obvious difference between AODV and DSR is that AODV uses hop-by-hop table-driven routing and sequenced number while DSR makes aggressive uses of source routing and route cache. AODV maintains routing tables per entry for each destination. In the absence of route cache and promiscuous mode [11], AODV tends to flood the network with more RREQ and other broadcast packets than DSR. Once the destination receives the RREQ, it will discard the rest of the RREQs and only reply with the first one received (the one with minimal delays). But in DSR, the destination will reply to every RREQ it receives which may possibly cause a RREP flood in the network rather than RREQ. In this way, DSR can also maintain more than one route to the destination, a clear advantage over AODV (only one entry is maintained).

Each route in AODV has its own life time; if a table entry has not been used for a certain life time, it will expire and no longer be used. DSR on the other hand does not have proper scheme to scan stale routes in its route cache. A stale route may still be used even when it is not available and the associated RERR has not reached all nodes. Furthermore, because of promiscuous listening and node mobility, it is possible that more caches get polluted by stale entries than are removed by error packets. In AODV, it is also possible to expire valid routes if unused beyond an expiry time. Moreover, determining a suitable expiry time is also a difficult task.

Finally, route deletion activity is more conservative in AODV; RERR packets reach all nodes using a failed link on its route to any destination. However, in DSR when the route of a data packet breaks, only upstream nodes of the failed link are notified through RERR messages. A performance comparison of these two protocols is presented in [11]. It has been shown that the packet delivery ratio, the network average latency of DSR performs better than AODV in a small sized network with low traffic load while AODV performs better in large sized network with high traffic load. However, because DSR makes use of aggressive source routing, route caching and other optimizations, it always has lower routing overhead.

2.6 Secure Routing in MANET

In many ad hoc networking applications, security in the routing protocols is necessary to guard against attacks such as malicious routing misdirection. Attacks on ad hoc networks generally fall into two classes [26]:

- **Routing Disruption**: this attack attempts to force data packets routed in a dysfunctional manner;
- **Resource Consumption**: this attack targets the consumption of valuable network resources, such as bandwidth, limited device energy, etc.

In both cases, they will cause DoS (Denial of Service) in the upper application layer. Various techniques can be applied to disrupt the normal routing mechanisms:

- *Simple Dropping [33]:* This is a simple but common attack when a node agrees to forward the packets but fails to do so, e.g., drop the received data packets.
- *Black Hole [26]:* An attacker can create a black hole by distributing forged routing packets, e.g., by announcing itself to be on the shortest path, which will attract the traffic to traverse through itself. In this way, it can discard these data packets to degrade the network performance. A variant of black hole is the *gray hole*, where a node selectively drops packets. On the other hand, a node can also inject forged routing information to cause other nodes choose longer routes than optimal one (shortest path).
- *Gratuitous Detour [26]:* Here, an attacker will attempt to make itself appear on the longer route by adding virtual nodes to the route thus avoiding being selected by the route discovery process.
- *Rushing Attack [28]:* This attack targets on-demand routing protocols that use duplicate suppression at each node. An attacker can distribute its RREQs faster than other nodes, suppressing any later legitimate RREQs to be dropped by well-behaved nodes due to duplicate suppression.
- *Wormhole [27]:* A pair of colluding attackers try to build a fast transmission tunnel via the use of private network connection, e.g., a wired link which is faster than wireless link. In this way, routing discovery packets traversing through this tunnel

will first reach the destination and when the source send packets using the discovered route will be failed. Wormhole can successfully prevent the discovery of routes that are longer than two hops.

Significant research efforts have been made for securing routing protocols, we will discuss the state of research in the following sections.

2.6.1 Watchdog

In [33], people have described two techniques to mitigate the *simple dropping* attack. They use a *watchdog* that identifies misbehaving nodes and a *pathrater* that helps routing protocols avoid these nodes.

The watchdog is implemented by maintaining a buffer of recently sent packets and comparing each overheard packet with the packet in the buffer to see if there is a match. If so, the packet in the buffer is removed and forgotten by the watchdog, since it has been forwarded on. If a packet remained in the buffer for longer than a certain timeout, the watchdog increments a failure tally for the node responsible for forwarding on that packet. If the tally exceeds a certain threshold bandwidth, it determines that the node is misbehaving and sends a message to the source notifying it of the misbehaving node.

The pathrater run by each node maintains a rating for every other node it knows about in the network. It calculates a path metric by averaging the node ratings in the path. Based on this metric, it can give a comparison of the overall reliability of different paths and allows pathrater to emulate the shortest path algorithm when no reliability information has been collected. If there are multiple paths to the same destination, the path with highest metric, i.e., most reliable, will be selected. Thus it is different from routing protocols, e.g. DSR, which select the shortest path. As we can see, pathrater needs to know the exact paths to the destination, it has to be built on top of source routing protocols.

The obvious weakness with watchdog is that it will mistakenly identify a node as misbehaving in the presence of: (1) ambiguous collisions, (2) receiver collisions, (3) limited transmission power, (4) false misbehavior, (5) collusion and (6) partial dropping. However, it is the first published work that focuses on detection and handling ad hoc routing misbehavior.

2.6.2 Packet Leashes

As mentioned earlier, the worm hole attack will prevent a node to discover routes more than two hops longer. For example, when on-demand routing protocols such as AODV and DSR are used, the worm hole can transmit all the routing packets, e.g. RREQs, directly to the destination through the tunnel. Consequently, the source will use the route through the worm hole nodes as forwarding nodes. If these worm hole nodes also tunnel the data packets, there will be no harm; however, if they do not tunnel the data packets, these data packets cannot route through these worm hole nodes and will be dropped because the route they try to use never exists. Periodic protocols are particularly vulnerable to this kind of attack. In OLSR [15] and TBRPF [35], nodes use HELLO messages to ensure the connectivity of neighborhood. However, worm hole can be realized by tunneling the HELLO messages.

One solution to solve the worm hole attack is by using of packet leashes [27]. Two types are considered: *geographical* and *temporal*. The main idea is that by authenticating either an extremely precise timestamp or location information combined with a loose timestamp, a receiver can determine if the packet has traversed an unrealistic distance for the specific network technology used. *Temporal* leashes rely on extremely precise time synchronization and timestamps in each packet. A node can approximate a packet's travel time as the difference between the receive time and the timestamp. To be more conservative, however, a node can choose to add the maximum time synchronization error, assuming that the senders clock might be faster than the receivers. Conversely, to allow all direct communication between legitimate nodes, a node can substract the maximum time synchronization error, assuming that the sender's clock might be slower than the receiver's. The *geographical* leash requires the receiver to compute an upper bound on the distance between the sender and itself d_{sr} based on the information of timestamp and velocity and other relative errors.

2.6.3 SEAD

SEAD [25] is robust against multiple uncoordinated attackers creating incorrect routing state in any other node, in spite of active attackers or compromised nodes in the network. The authors based SEADs design in part on the DSDV [39]. To support use of SEAD with nodes of limited CPU processing capability, and to guard against DoS attacks in which an attacker attempts to cause other nodes to consume excess network bandwidth or processing time, they use efficient one-way hash functions and do not use asymmetric cryptographic operations in the protocol. Each node in SEAD uses a specific single next element from its

$S:$	$h_0 = MAC_{K_{SD}}(REQUEST,S,D,id,ti)$
$S \rightarrow *:$	$REQUEST,S,D,id,ti,h0,(),()\rangle$
$A:$	$h_1 = H[A,h_0]$
..	$M_A = MAC_{K_{A_{ti}}}$
..	$REQUEST,S,D,id,ti,h_1,(A),())$
$A \rightarrow *:$	$REQUEST,S,D,id,ti,h_1,(A),M_A)$
$B:$	$h_2 = H[B,h_1]$
..	$MB = MAC_{K_{B_{ti}}}(REQUEST,S,D,id,ti,h_2,(A,B),(M_A))$
$B \rightarrow *:$	$REQUEST,S,D,id,ti,h_2,(A,B),(M_A,M_B)\rangle$
$C:$	$h_3 = H[C,h_2]$
..	$M_C = MAC_{K_{C_{ti}}}(REQUEST,S,D,id,ti,h_3,(A,B,C),(M_A,M_B))$
$C \rightarrow *:$	$REQUEST,S,D,id,ti,h_3,(A,B,C),(M_A,M_B,M_C)$
$D:$	$M_D = MAC_{K_{DS}}(REPLY,D,S,ti,(A,B,C),(M_A,M_B,M_C))$
$D \rightarrow C:$	$REPLY,D,S,ti,(A,B,C),(M_A,M_B,M_C),M_D,()$
$C \rightarrow B:$	$REPLY,D,S,ti,(A,B,C),(M_A,M_B,M_C),M_D,(K_{C_{ti}})$
$B \rightarrow A:$	$REPLY,D,S,ti,(A,B,C),(M_A,M_B,M_C),M_D,(K_{C_{ti}},K_{B_{ti}})$
$A \rightarrow S:$	$REPLY,D,S,ti,(A,B,C),(M_A,M_B,M_C),M_D,(K_{C_{ti}},K_{B_{ti}},K_{A_{ti}})$

Figure 2.1: Route Discovery Example in Ariadne

hash chain in each routing update that it sends about itself (metric 0). Based on this initial element, the one-way hash chain conceptually provides authentication for the metrics lower bound in other routing updates for this destination. The authentication provides only a lower bound on the metric, that is, an attacker can increase the metric or claim the same metric, but cannot decrease the metric.

2.6.4 Secure On-demand Routing

Ariadne

Ariadne [29] is a secure on-demand routing protocol that withstands node compromise and relies only on highly efficient symmetric cryptography. Ariadne can authenticate routing messages using one of three schemes:

$$\begin{array}{ll}
S \rightarrow *: & (ROUTEREQUEST, D, cert_S, N, t)_{K_S^-} \\
A \rightarrow *: & ((ROUTEREQUEST, D, cert_S, N, t)_{K_S^-})_{K_A^-}, cert_A \\
B \rightarrow *: & ((ROUTEREQUEST, D, certS, N, t)_{K_S^-})_{K_B^-}, cert_B \\
C \rightarrow *: & ((ROUTEREQUEST, D, certS, N, t)_{K_S^-})_{K_C^-}, cert_C \\
D \rightarrow C: & ((ROUTEREPLY, S, cert_D, N, t)_{K_D^-} \\
C \rightarrow B: & ((ROUTEREPLY, S, cert_D, N, t)_{K_D^-})_{K_C^-}, cert_C \\
B \rightarrow A: & ((ROUTEREPLY, S, cert_D, N, t)_{K_D^-})_{K_B^-}, cert_B \\
A \rightarrow S: & ((ROUTEREPLY, S, cert_D, N, t)_{K_D^-})_{K_A^-}, cert_A
\end{array}$$

Figure 2.2: Route Discovery Example in ARAN

$$\begin{array}{ll}
B \rightarrow A: & \langle((ROUTEERROR, S, D, cert_B, N, t)_{K_B^-}\rangle \\
A \rightarrow S: & \langle((ROUTEERROR, S, D, cert_B, N, t)_{K_B^-}\rangle
\end{array}$$

Figure 2.3: Route Maintenance Example in ARAN

- shared secrets between each pair of nodes;
- shared secrets between communicating nodes combined with broadcast authentication or;
- digital signatures.

Ariadne has a mechanism to let the target verify the authenticity of the RREQ and then present an efficient per-hop hashing technique to verify that no node is missing from the node list in the RREQ (see Fig. 2.1, S attempts to discover a route to D). Route maintenance in Ariadne is based on DSR. A node forwarding a packet to the next hop along the source route returns a RERR to the packets original sender if it is unable to deliver the packet to the next hop after a limited number of retransmission attempts. To prevent unauthorized nodes from sending REERs, the sender will authenticate the REERs.

ARAN

People have developed authenticated routing for ad hoc networks (ARAN) [44] based on AODV. ARAN provides authentication and non-repudiation services using pre-determined cryptographic certificates that guarantees end-to-end authentication. In doing so, ARAN limits or prevents attacks that can afflict other insecure protocols. In ARAN, each node has a certificate signed by a trusted authority, which associates its IP address with a public key. ARAN can be divided into two processes: secure route discovery (see Fig. 2.2, S attempts to discover a route to D) and secure route maintenance (see Fig. 2.3, node B determines that the next hop D is broken by broadcasting a signed route error messages indicating D is unreachable. Each node using B as the next hop will re-broadcast this message but will not re-sign it.). However, since ARAN uses public-key cryptography for authentication, it is particularly vulnerable to DoS attacks based on flooding the network with bogus control packets for which signature verifications are required. As long as a node cannot verify signatures at line speed, an attacker can force that node to discard some fraction of the control packets it receives.

SAODV

Secure AODV (SAODV) [49] is a secure protocol based on AODV. The idea behind SAODV is to use a signature to authenticate most fields of a RREQ and RREP and to use hash chains to authenticate the hop count. SAODV designs signature extensions to AODV. Network nodes authenticate AODV routing packets with an SAODV signature extension, which prevents certain impersonation attacks. In SAODV, a RREQ packet includes a route request single signature extension (RREQ-SSE). The initiator chooses a maximum hop count, based on the expected network diameter, and generates a one-way hash chain of length equal to the maximum hop count plus one. This one-way hash chain is used as a metric authenticator, much like the hash chain within SEAD. The initiator signs the RREQ and the anchor of this hash chain; both this signature and the anchor are included in the RREQ-SSE. In addition, the RREQ-SSE includes an element of the hash chain based on the actual hop count in the RREQ header. This value is called as the hop-count authenticator. For example, if the hash chain values $h_0, h_1, ..., h_N$ are generated such that $h_i = H[h_{i+1}]$, then the hop-count authenticator h_i corresponds to a hop count of Ni.

2.6.5 Secure Link-state Routing

Secure Link-State Protocol (SLSP) [36] uses digital signatures and one-way hash chains to ensure security of link-state updates. SLSP provides secure proactive topology discovery, which can be very beneficial to the network operation. SLSP link-state updates are signed and propagated to a limited number of hops. In ZRP [24], SLSP link-state updates would have a maximum hop count equal to a zone radius. To ensure that an SLSP update does not travel too many hops, each update includes a hop count representing the number of hops traveled by the SLSP update. As in SEAD and SAODV, a hash chain is used to authenticate the hop count, and the hash chain values are authenticated using the hash chains anchor, which is included in the signed portion of the SLSP link-state update. SLSP uses the same lightweight flooding prevention mechanism as SRP, wherein nodes that relay or generate fewer link-state updates are given priority over any node that sends more link-state updates.

As in ZRP, an attacker can masquerade as a victim node and flood the victims neighbors with link-state updates that appear to originate at the victim. Although the victim might be able to detect the attack, due to NLP's (normal link pulses) duplicate MAC address detection functionality, the victim will have no way to protest.

2.6.6 Reputation-based System

Reputation-based systems *Confidant* [12], based on DSR, consists of four components: the monitor, the trust monitor, the reputation system, and the path manager. For each packet a node forwards, the monitor on that node attempts to ensure that the next-hop node also forwarded the packet correctly. When the monitor detects an anomaly, it triggers action by the reputation system, which maintains a local ratings list. These lists are potentially exchanged with other nodes; the trust monitor handles input from other nodes. If a list is received from a highly trusted node, the receiver can directly place information from the list into its local ratings list. On the other hand, if a list is received from an untrusted source, the receiver can completely ignore it or give it substantially less weight than a list received from a more trusted node. Finally, the path manager chooses paths from the nodes route cache based on a blacklist and the local ratings list. The path manager also specifies the reaction to a RREQ from a node on the blacklist or to a RREQ that has traversed a node on the blacklist.

The authors of [5] propose an on-demand routing protocol for ad hoc wireless networks that provides resilience to byzantine failures caused by individual or colluding nodes. Their adaptive probing technique detects a malicious link after $\log n$ faults have occurred, where

n is the length of the path. These links are then avoided by multiplicatively increasing their weights and by using an on-demand route discovery protocol that finds a least weight path to the destination which is similar to watchdog and pathrater. They also propose a technique for performing route maintenance in cases where an attacker is already on the path. Their approach is to define an acceptable level of performance for example, based on a packet delivery ratio within a latency limit. When the performance of a path drops below the acceptable level, a binary search is initiated to locate the link responsible for dropping the paths performance level below the acceptable level. A digital signature authenticates these detection packets are authenticated, which are then onion encrypted, such that each node forwarding the packet decrypts the outer layer, processes any probe requests, and forwards the packet.

2.7 Conclusion

In this chapter, we give a survey on the security vulnerabilities of the current ad hoc routing protocols and the existing secure routing protocols. We will present the security issues related to MANET MAC layer in Chapter 3.

Chapter 3

MAC Vulnerabilities in Mobile Ad Hoc Networks

3.1 Introduction

Although MANETs have exhibited unique advantages compared with one-hop wireless networks, e.g. cellular networks and wireless local area networks (WLAN), they also introduce new challenging issues on the MAC (medium access control) protocol design and implementation. First, MANET is a self-organized network without any centralized management. As a result, reliable communication totally depends on coordination between nodes themselves. In the presence of nodes (selfish or malicious) that are unwilling to cooperate, normal network operation may be disrupted. Second, due to the shared medium and current hardware constraints, it is hard to make efficient usage of wireless channel capacity. For example, a hidden terminal[1] can cause unexpected collisions which lead to retransmissions of data frames [8, 47, 48]. An exposed terminal[2], on the other hand, can reduce the bandwidth utilization. Third, node mobility may lead to frequent route breakage and hence additional route discoveries which could affect the interaction between MAC layer and network layer or other higher layers. Fourth, MAC protocol should also be designed to minimize the consumption of limited hardware resources, such as energy. Finally, fairness in individual stations and flows are also open problems that need to be solved.

In Section 3.2, we first overview the IEEE 802.11 standard that is widely used as MAC protocol in simulation and testbeds of ad hoc networks. In Section 3.3, we present the current security vulnerabilities in IEEE802.11. MAC misbehaviors are described in details in Section 3.4. A case study of MAC hybrid misbehavior is presented in Section 3.5. Finally we summarize the existing detection and reaction schemes designed to counter against current MAC misbehavior in Section 3.6.

3.2 Overview of IEEE 802.11

MAC protocols designed for wireless network can be classified as two main categories: *Random Access* and *Controlled Access*. Random access requires nodes to compete with each other to gain full access to the channel, while controlled access has infrastructure-based management node to assign the shared medium to each node. The lack of an infrastructure and the peer-to-peer nature of MANET, make random access by nature the

[1]A hidden terminal is a station that can neither sense the transmission of a transmitter nor correctly receive the reservation packet from its corresponding receiver.

[2]An exposed terminal is a station that senses the transmission of a transmitter and can not interfere with the reception at the receiver

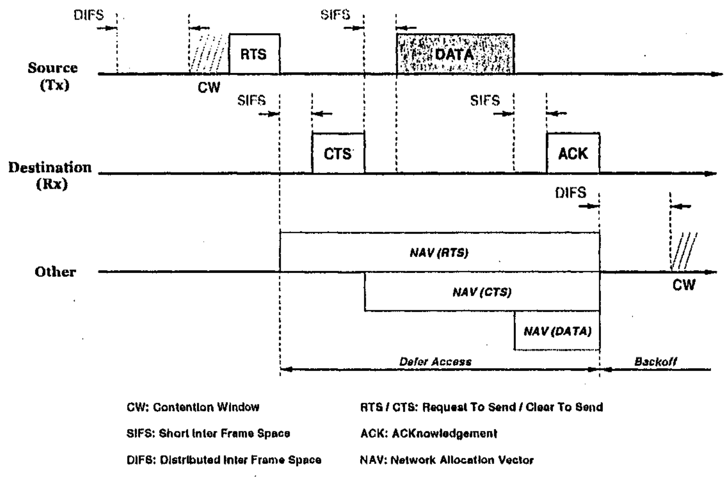


Figure 3.1: RTS/CTS/DATA/ACK Handshaking in DCF mode

choice of medium access control protocol for MANET. Examples include MACA (Multiple Access with Collision Avoidance) [31], MACAW (MACA with Acknowledgment) [7], MACA-BI (MACA with Invitation) [45], DBTMA (Dual Busy Tone Multiple Access) [16]. Among all these, a variant of MACA — CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) is selected by the IEEE as the basis for IEEE 802.11 standard for its capability to solve the hidden terminal and exposed terminal problems [8,47,48].

The IEEE 802.11 [1] defines two basic access methods: (1) a fully distributed mechanism called Distributed Coordination Function (DCF), which allows contention access for the wireless media; (2) a centralized mechanism called Point Coordinator Function (PCF), which requires centralized access points. DCF is the MAC layer basic access method for ad hoc networks. It is also known as Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). CSMA/CA is designed to reduce collisions when multiple nodes access the shared medium. Carrier Sense is performed by both physical sense and virtual sense mechanisms. There are two communication options in DCF: (1) four-way handshaking, i.e., RTS-CTS-DATA-ACK, which is suitable for long frame data transmission (see Fig. 3.1); (2) two-way handshaking, i.e., DATA-ACK, which is suitable for short frame data transmission. We only describe the operation of four-way handshaking in this section.

A station with a new data packet to transmit will sense the channel activity first. If the channel is idle for a period of time equal to DIFS[3] (Distributed Inter Frame Space), the station transmits a short control frame RTS (Request to Send). The time immediately following an idle DIFS is slotted, and a station is allowed to transmit only at the beginning of each time slot. The time slot size is set equal to the time needed at any station to detect the transmission of a packet from any other station. This size depends on the physical layer, and it accounts for the propagation delay, for the time needed to switch from the receiving to the transmitting state (RX_TX_TurnaroundTime), and for the time to signal to the MAC layer the state of the channel (busy detect time).

If the channel is sensed busy, the station has to wait until the channel is sensed idle for a DIFS time. At this point, the station generates a random backoff time interval before transmission, in order to minimize the probability of collisions with packets being transmitted by other stations. DCF adopts an exponential backoff scheme. At each packet transmission, the backoff time is uniformly chosen in the range $[0, CW]$. The value CW is called contention window, and depends on the number of transmissions failed for the packet. At the first transmission attempt, CW is set to CW_{min}. After each retransmission, CW will be exponentially increased until the maximum CW value, i.e., $min(2^{i-1} \times (CW_{min}+1) - 1, CW_{max})$. Upon successful transmission, CW will be reset to CW_{min}. In addition, to avoid the capture

[3]A station needs to wait for a EIFS instead of DIFS if the last received signal is not correctly decoded.

channel effect[4], the station has to backoff between two consecutive new packet transmissions, even if the medium is sensed idle during the DIFS time[5].

Upon correct reception of the RTS, the receiver will reply with a short control frame CTS (Clear to Send). Once the transmitter receives the CTS, it will start to transfer DATA. After the successful reception of DATA, the receiver sends an ACK to the transmitter. The exchange of RTS/CTS prior to the actual data transmission reduces the high collision probability by distributing the medium reservation information and partially solves the hidden terminal problem. The RTS/CTS contains a duration field indicating the time after the end of present frame transmission that the channel will be reserved to complete the data or management frame transmission. Any node within the transmission range of either the sending node or the receiving node hears the RTS/CTS will learn the medium reservation and adjust its Network Allocation Vector (NAV), which indicates the amount of time that the node should defer. The collision will mostly happen when the current node completes its transmission and multiple nodes are waiting for contending for the channel. Thus each node with data to transmit will generate a random backoff number from range $[0, CW]$ for an additional deferring time. While the channel is idle, the backoff number will decrease by one after one time slot, which is defined in the IEEE 802.11 standard and stop decrementing when the channel is busy. Once the backoff counter reaches zero, the sending node will reserve the channel by exchanging RTS/CTS as described above.

[4]Capture channel effect is that a node with heavy traffic is easy to capture the channel than a node with light traffic which will cause short term unfairness.

[5]As an exception to this rule, the protocol provides a fragmentation mechanism, which allows the MAC to split an MSDU (MAC Service Data Unit, the packets delivered from higher layers to the MAC layer) into more MPDUs (MAC Protocol Data Unit, packets delivered from the MAC layer to the physical layer), if the MSDU size exceeds the maximum MP DU payload size. The different fragments are then transmitted in sequence, with only one SIFS (Short Inter Frame Space) between them, so only the first fragment need to contend for the channel access.

If a node sends RTS but does not receive CTS within certain time, the node will defer by doubling its *CW* size and choosing a random value from the new range and retransmit RTS with limited times. If the RTS retry time is more than the station short retry count (SSRC) the sending node will drop the DATA packet and inform the network layer of a link breakage. Alternatively, if the ACK is not received within certain time, the sending node will retransmit the DATA packet for limited time (i.e. SSRC for a short frame DATA, or station long retry limit SLRC for a long frame DATA[6]).

3.3 MAC Security Issues

As mentioned earlier, there are two modes of operation for IEEE 802.11: (1) DCF (infrastructure-less ad hoc); (2) PCF (infrastructure via access points, APs). In DCF, each node communicates with each other directly; while in PCF, each node communicates with each other through a single AP. A network may consist of more than one AP and the nodes need to roam among them to effectively switch the associated AP. Authentication and association should be done before a node starts communication with a AP.

IEEE 802.11 defines two types of authentication service: (1) *open system*; (2) *shared key*. This authentication subtype information is performed as unicast frames between pairs of stations, i.e., multicast authentication is not allowed. De-authentication information are advisory and can be sent as group-addressed frames.

- **Open System.** Basically it is a *null* authentication algorithm. Any node requesting

[6]Short/long frame is decided by the *RTSThreshold* maintained by each station [1]

authentication with this algorithm will be authenticated. It is the default authentication algorithm. Two-step authentication transaction sequence is involved: (1) identity assertion and request for authentication; (2) authentication result.

- **Shared Key.** This authentication algorithm does require the use of the WEP[7] (Wired Equivalent Privacy) privacy mechanism while it does not need to distribute the secret in clear. Therefore, this authentication is only available if the WEP option is implemented. And the required secret, shared key, is presumed to be delivered to active nodes through a secure channel.

We briefly describe how to perform some common attacks against MAC, especially with regard to the MAC authentication and association mechanisms.

3.3.1 Passive Attacks

Consider an environment in which users change often, one efficient way to handle with authentication, as adopted by many administrators [4,9], is to leave the network open to the users who request. This is simply because of the difficulty in WEP based key distribution. Many wireless cards may be put in *rfmon* mode or *promiscuous* monitor mode, which allows the card firmware to pass all the received data to the operating system [9]. This makes the eavesdropping[8] trivial. The only constraint is that only one channel at a time may be monitored, though this is only a minor inconvenience because some cards support

[7]*Wired Equivalent Privacy* is designed to protect privacy of authorized users from casual eavesdropping attack. It only encrypts the data section in a data frame. While data confidentiality should depend on the external key management scheme. Therefore, it is possible that a system can run with WEP but without authentication.

[8]Eavesdropping is the intercepting and reading of messages by unintended recipients.

hardware channel hopping, which makes it feasible to monitor more than one channel. As eavesdropping attack is passive and non-malicious by nature, it is difficult to protect wireless network from such passive attack when the wireless communication takes place on unlicensed public frequencies. Armed with a wireless network adapter that supports promiscuous mode, the eavesdropper can capture network traffic for analysis using easily available tools, such as Network Monitor in Microsoft products, or TCPdump in Linux-based products, or AirSnort.

Passive attacks are by their very nature difficult to detect. If an administrator is using dynamic host configuration protocol (DHCP) on the wireless network (this is not recommended), he or she might notice that an authorized MAC address has acquired an IP address in the DHCP server logs. Perhaps the administrator notices a suspicious-looking car sporting an antenna out of one of its windows. If the car is parked on private property, the driver could be asked to move or possibly charged with trespassing. But, the legal response is severely limited. Only if it could be determined the wardriver was actively attempting to crack any encryption used on the network or otherwise interfering or analyzing wireless traffic with malicious intent would he or she be susceptible to being charged with a data-related crime, but this would depend on the country or state in which the activity took place.

Passive attacks on wireless networks are extremely common, almost to the point of being ubiquitous. Detecting and reporting on wireless networks has become a popular hobby for many wireless wardriving enthusiasts. It becomes so popular to advertise both the availability of APs and the services they offer.

One of the most popular tool used for passive listening called Netstumbler, which is available from *www.netstumbler.com.* The Netstumbler program works primarily with wireless network adapters that use the Hermes chipset because of its ability to detect multiple APs that are within range and WEP, among other features (a list of supported adapters is available at the Netstumber web site). The most common card that uses the Hermes chipset for use with Netstumbler is the ORiNOCO gold card. Another advantage of the ORiNOCO card is that it supports the addition of an external antenna, which can greatly extend the range of a wireless network to many orders of magnitude, depending on the antenna. A disadvantage of the Hermes chipset is that it does not support promiscuous mode, so it cannot be used to sniff network traffic. For that purpose, people need a wireless network adapter that supports the PRISM2 chipset. The majority of wireless network adapters targeted for the consumer market use this chipset, for example, the Linksys WPC network adapters. Sophisticated attackers will arm themselves with both types of cards, one for discovering wireless networks and another for capturing the traffic.

In spite of the fact that Netstumbler is free, it is a sophisticated and feature-rich product that is excellent for performing wireless site surveys, whether for legitimate purposes or not. Not only can it provide detailed information on the wireless networks it detects, it can be used in combination with a GPS to provide exact details on the latitude and longitude of the detected wireless networks.

Netstumbler displays information on the Service Set Identifier (SSID), the channel, and the manufacturer of the wireless AP. There are a few things that are particularly noteworthy about this session. The first is that a couple of APs are still configured with the default SSID supplied by the manufacturer, which should always be changed to a non-default value upon set up and configuration. Another is that at least one network uses a SSID that may provide a clue about the entity that has implemented it; again this is not a good practice when configuring SSIDs. Finally, we can see what networks have implemented WEP.

If the network administrator has been kind enough to provide a clue about the company in the SSID or he is not encrypting traffic with WEP, then the job of a potential eavesdropper is made a lot easier. Using a tool such as Netstumbler is only a preliminary step for the attacker. After discovering the SSID and other information, the attacker can connect to the wireless network to sniff and capture network traffic. This network traffic can reveal a lot of information about the network and the company that uses it. For example, looking at the network traffic, the attacker can determine what DNS servers are being used, the default home pages configured on browsers, network names, log-on traffic, and so on. The attacker can use this information to determine if the network is of sufficient interest to proceed further with other attacks. Furthermore, if the network is using WEP, the attacker can, given enough time, capture a sufficient amount of traffic to crack the encryption.

Netstumbler works on networks that are configured as open systems. This means that the wireless network indicates that it exists and will respond with the value of its SSID to other wireless devices when they send out a radio beacon with an empty set SSID. This does not mean, however, that wireless network can be easily compromised, if other security measures have been implemented. To defend against the use of Netstumbler and other programs to detect a wireless network easily, administrators should configure the wireless network as a closed system. This means that the AP will not respond to empty set SSID beacons and will consequently be invisible to programs such as Netstumbler which rely on this technique to discover wireless networks. However, it is still possible to capture the raw 802.11b frames and decode them through the use of programs such as Ethereal and

WildPackets AiroPeek to determine this information. As well, RF spectrum analyzers can be used to discover the presence of wireless networks.

We should note that on the wireless side, APs are half-duplex devices and work just like other half-duplex devices, such as hubs and repeaters. This means that all the devices on the network can potentially see all the traffic from other devices. The only defense against sniffing on a wireless network is to encrypt MAC layer messages and higher traffic whenever possible through the use of WEP, VPNs, SSL, Secure Shell (SSH), Secure Copy (SCP), and so on. Some of these defensive strategies will be more effective than others, depending on the circumstances.

3.3.2 Active Attacks

Once an attacker has gained sufficient information from the passive attack, the hacker can then launch an active attack against the network. There are a potentially large number of active attacks that a hacker can launch against a wireless network. For the most part, these attacks are identical to the kinds of active attacks that are encountered on wired networks. These include, but are not limited to, unauthorized access, spoofing, and Denial of Service (DoS) and Flooding attacks, as well as the introduction of Malware and the theft of devices. With the rise in popularity of wireless networks, new variations of traditional attacks specific to wireless networks have emerged along with specific terms to describe them, such as drive-by spamming in which a spammer sends out tens or hundreds of thousands of spam messages using a compromised wireless network.

Attacks to WEP

The fact that WEP is insecure is quite well known [18] — contrary to considerations in other forms of cryptography, a 128 bit key will require weekly changes if even a moderate degree of security is desired. There are two main branches of WEP attacks: the weak IV attacks (dealing with key recovery) and injection attacks (sending traffic without knowing the key). The most practical way to break WEP is to use a hybrid of both methods. This requires certain skills which is why, according to many, these attacks remain somewhat theoretical. However, this view is somewhat short-sighted: the current nonexistence of a fully automated tool to recover WEP keys with minimal effort does not mean that the vulnerabilities in WEP are nonexistent or negligible.

The IEEE 802.11 standard specifies WEP with a 64 bit key, though most current hardware also supports 128 bit keys. WEP is an implementation of the RC4 [43] stream cipher. It may be split into two main steps:

- Key Setup Algorithm (KSA): establishes a 256 byte state array, which is key-dependent.
- Pseudo-Random Generation Algorithm (PRGA): creates a pseudo-random stream based on the state array, without using the key explicitly.

The overall process consists of setting up RC4 with a 64 bit seed (key) and generating a pseudo-random stream of the same length as the clear-text. This stream is then XOR-ed with the clear-text to produce cipher-text. XOR is a symmetric operation – XOR-ing the cipher-text with the same stream allows retrieval of the clear-text.

The term "64bit seed" is used instead of "key" to differentiate between the WEP secret key and the actual seed that is passed to the RC4 algorithm. The secret WEP key is 40 bits

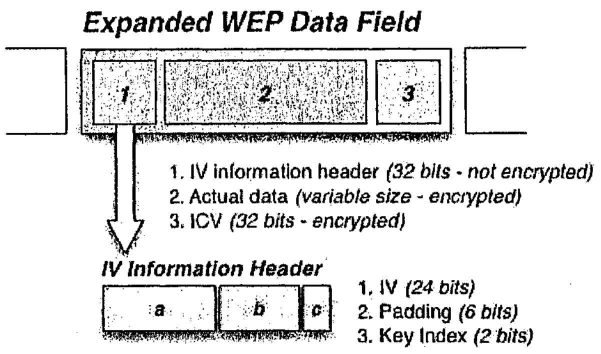

Expanded WEP Data Field
1
2
3
1. IV information header (32 bits - not encrypted)
2. Actual data (variable size - encrypted)
3. ICV (32 bits - encrypted)
IV Information Header
a
b
c
1. IV (24 bits)
2. Padding (6 bits)
3. Key Index (2 bits)

Figure 3.2: Frame Structure using WEP

long, to which is prepended the 24 bit Initialization Vector (IV), making up the full 64 bit seed. The IV is a 24 bit number generated by the sender, that should be unique and re-used as little as possible. However, many implementations currently use a simple incremental counter to generate IVs.

In order to decrypt the data successfully and be able to detect errors, the data portion in WEP packets contains some additional fields. The expansion of the data field in WEP packets may be seen in Fig. 3.2. The IV information is expanded further. The first (left to right) 24 bits are the actual IV used in the encryption. The next six bits are for padding, leaving the last two for the Key index. WEP may have up to four (22) different keys in use within a single network. The key index indicates which key is being used. The ICV is the CRC32 algorithm run over the clear-text data. Thus, when a packet is decrypted, the CRC32 checksum of the clear-text is calculated and matched with the ICV in order to detect possible errors.

Brute-Force and Pass-Phrases: The secret key is only 40 bits long. On an average modern PC, it takes around a month to search the entire key-space testing possibilities (brute-force). Distributing the work makes this even easier. Also, if Murphy's law does not apply, on average the key will be recovered in half the time. Brute-forcing 104 bit keys is infeasible. In practice, keys are not entered as hex digits all the time, as other mechanisms may be provided instead. Remembering a 40 bit hex key may be impractical. Many vendors have implemented ways of transforming a pass-phrase into the equivalent hex key. Such algorithms are not standard. For example, by default Microsoft Windows XP maps the ASCII value of the pass-phrase to its equivalent hex value ('A' becomes 0x41, an so on). Furthermore, the pass-phrase has to be exactly either five (40 bit WEP) or thirteen characters long (104 bit WEP). Brute-forcing alphanumeric, five character pass-phrases, takes very little time, and some networks seem to be using them. Other vendors may take some hash of the pass-phrase, although this is still vulnerable to dictionary attacks and possibly even the hash function itself, or its implementation. For example, an attacker may iteratively hash dictionary words and use the result as a possible key, until the correct one is found.

Weak IV Attack: This attack was formalized by Fluhrer, Mantin, and Shamir (FMS) [18] although, chronologically, Wagner [46] first noticed this vulnerability in 1995 (before 802.11 was published). It consists in eavesdropping for packets that use weak IVs. A weak IV is a particular IV that will set up the key in a specific way that might give information about a key byte. A single weak IV packet will give about 5% probability of guessing the correct value for a particular key byte. This makes it a statistical attack, as we need to build a table of the most probable candidates for each key byte. The authors in [18] recommend collecting about 60 weak IVs per key byte. The attack is also incremental — we need to know the current key byte before we can recover the next (the first key byte may be recovered on its own).

Injection Attacks: In contrast to the weak IV attack, injection attacks mainly exploit the PRGA part of the RC4 algorithm. In fact, they do not aim to recover the WEP key, but instead a PRGA stream, which will be used to inject arbitrary traffic without knowing the actual WEP key. In WEP, the PRGA stream acts as a one-time pad. The stream is XOR-ed with the clear-text to produce the cipher-text. Knowing that stream will allow us to recover the clear-text (XOR cipher-text with stream). Each PRGA stream is constant and unique to the seed (IV and WEP key pair) that did the key setup (KSA). One reason why WEP introduced IVs is to avoid having a single PRGA stream for the network (the seed is constant as there is no variable IV). Instead, each IV will generate a different PRGA stream. If a PRGA stream is recovered for a particular IV, arbitrary traffic may be encrypted and sent to the network using that IV.

Attacks to WPA

The WPA standard is a subset of the 802.11i wireless security standard intended to address the cryptographic shortcomings of Wired Equivalent Protocol (WEP). WPA comes in two forms: per-user based security designed for enterprises, and a pre-shared key mode designed for consumers. While the former utilizes a RADIUS server to ensure per user keying, the latter greatly simplifies deployment for home and SOHO users by having a master key (based on a pass phrase) for the wireless LAN. WPA and 802.11i are necessary because WEP has known weaknesses, poor key manageability, and lacks simplicity needed among home users for deployment.

Two modes are defined in WPA:

- *Consumer mode:* Consumer Mode - Pre-shared key (PSK) based protocol with the combination of Pre-Shared Key, TKIP key management and Michael integrity checking aimed for home use. Simplicity of deployment is of primary concern.

- *Enterprise mode:* Per-user authentication based protocol with the combination of 802.1x security framework, authentication server, TKIP key management and Michael integrity checking aimed for enterprise use.

However, the PSK version of WPA suffers from an offline dictionary attack because of the broadcasting of information required to create and verify a session key which is quite self-contradicting. The nature of a "pass phrase" is likely to constrain the entropy of the pass chosen, leaving a great opportunity for tools to automate the process of effective off-line password cracking.

Spoofing

Because of the nature of wireless networks and the weaknesses of WEP, unauthorized access and spoofing are the most common threats to wireless networks. Spoofing occurs when an attacker is able to use an unauthorized station to impersonate an authorized station on a wireless network. A common way to protect a wireless network against unauthorized access is to use MAC filtering to allow only clients that possess valid MAC addresses access to the wireless network. The list of allowable MAC addresses can be configured on the AP, or it may be configured on a RADIUS server that the AP communicates with. However, regardless of the technique used to implement MAC filtering, it is a relatively easy matter to change the MAC address of a wireless device through software to impersonate a valid station. In Windows, this is accomplished with a simple edit of the registry, in UNIX through a root shell command. MAC addresses are sent in the clear on wireless networks, so it is also a relatively easy matter to discover authorized addresses.

WEP can be implemented to provide more protection against authentication spoofing through the use of Shared Key authentication. However, as we discussed earlier, Shared Key authentication creates an additional vulnerability. Because Shared Key authentication makes visible both a plaintext challenge and the resulting ciphertext version of it, it is possible to use this information to spoof authentication to a closed network.

Once the attacker has authenticated and associated with the wireless network, he or she can then run port scans, use special tools to dump user lists and passwords, impersonate users, connect to shares, and, in general, create havoc on the network through DoS and Flooding attacks. These DoS attacks can be traditional in nature, such as a ping flood, SYN, fragment, or Distributed DoS (DDoS) attacks, or they can be specific to wireless networks through the placement and use of Rogue Access Points to prevent wireless traffic from being forwarded properly.

Jamming Attacks

Jamming is a special kind of DoS attack specific to wireless networks. Jamming occurs when spurious RF frequencies interfere with the operation of the wireless network. In some cases, the jamming is not malicious and is caused by the presence of other devices, such as cordless phones, that operate in the same frequency as the wireless network. In a case like this, the administrator must devise and implement policies regarding the use of these devices, such a banning the use of Bluetooth devices, or choose wireless hardware that uses different frequencies. Intentional and malicious jamming occurs when an attacker analyzes the spectrum being used by wireless networks and then transmits a powerful signal to interfere with communication on the discovered frequencies. Fortunately, this kind of attack is not very common because of the expense of acquiring hardware capable of launching jamming attacks. Plus, jamming a network represents a kind of pyrrhic victory for the attacker — a lot of time and effort expending merely to disable communications for a while.

Man in the Middle Attack

MiM (Man in the Middle) is an attack that a malicious or compromised node is situated (physically or logically) between the communication path of two nodes. The major objective of this attack is to (1) eavesdrop the communication; or (2) intercept and change messages; or (3) cause failure of current data communication. Note that the victim of MiM does not need to run any vulnerable programme or communicate directly with a malicious/compromised node. There are several ways to realize MiM attack, such as DNS spoofing or ARP spoofing, etc. In Fig. 3.3, we show an simple example of MiM attack via ARP spoofing. The goal of this attack is to eavesdrop the wired LAN.

In Fig. 3.3, *Alice* is a wired LAN user and wants to communicate with an internet user *Bob* through the wired *Router*. Also there is a wireless user *Eve* connected to *Alice's* wired LAN. Now *Eve* will try to apply the MiM attack via ARP spoofing to poison *Alice's* and Router's ARP cache. *Eve* will try to map its MAC address [ee:ee:ee:ee:ee:ee] to *Router's* IP address 10.0.0.1 and map its MAC address [ee:ee:ee:ee:ee:ee] to *Alice's* IP address 10.0.0.10 also. Next *Eve* will distribute these forged ARP information through ARP requests and replies. In this way, if *Eve* is successful, the route $Alice \rightarrow Router \rightarrow Bob$ will

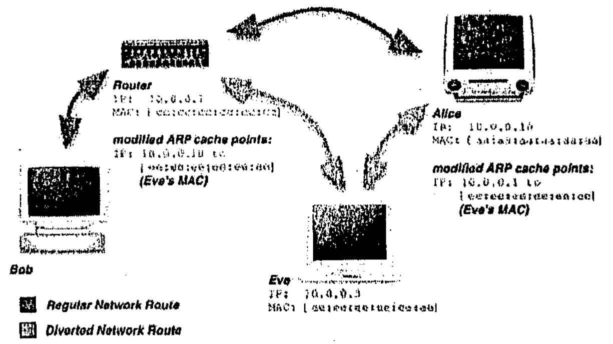


Figure 3.3: MiM attack via ARP spoofing

change to $Alice \rightarrow Eve \rightarrow Bob$.

A wireless-specific variation of MiM is to place a rogue access point within range of wireless stations. If the attacker knows the SSID in use by the network (which as we have seen is easily discoverable) and the rogue AP has enough strength, wireless users will have no way of knowing that they are connecting to an unauthorized AP. Using a rogue AP, an attacker can gain valuable information about the wireless network, such as authentication requests, the secret key that may be in use, and so on. Often, the attacker will set up a laptop with two wireless adapters, in which one card is used by the rogue AP and the other is used to forward requests through a wireless bridge to the legitimate AP. With a sufficiently strong antenna, the rogue AP does not have to be located in close proximity to the legitimate AP. So, for example, the attacker can run the rogue AP from a car or van parked some distance away from the building. However, it is also common to set up hidden rogue APs (under desks, in closets, etc.) close to and within the same physical area as the legitimate AP. Because of their undetectable nature, the only defense against rogue APs is vigilance through frequent site surveys using tools such as Netstumbler and AiroPeek, and physical security.

DNS Tunneling

DNS tunneling is a way in which un-authenticated users may route to the Internet by building up a fullfeatured and even bidirectional IP tunnel through Nameservers. Some setups, mainly found in hot-spots and large companies, authenticate users at a later stage with a different mechanism, rather than relying on 802.11. Consider a fully open wireless network with DHCP, where, as soon as you try to visit any web page, you are redirected to some login page where authentication details must be provided. Prior to authentication, all Internet traffic will be denied by the local gateway. The idea of this attack is to exploit the ability to send DNS queries through the network's local DNS server in order to communicate with a party (i.e., server) that is under our control, and potentially, through this server, freely access the Internet. Messages destined to or coming from this server are encoded in DNS queries and replies.

DHCP normally gives the IP address of a DNS server (or one may be found also by scanning the network). These DNS servers are sometimes fully functional (i.e., any Internet hostname may be resolved into its correct IP address). This means that data is sent and received from the authoritative name-server of a particular domain-name. Therefore a tunnel may be formed by encapsulating the IP packets into nameserver requests and answers contains the traffic of the other direction. For example, the request would look something like a host name look up to "frqv.dd_2trT-XEQ.Dd1q.domain.com". The traffic is encoded as a legal domain name. The answer will contains the payload

As the DNS protocol only allows 512 bytes/ pkt, fragmentation is needed. In addition, as it uses UDP and not TCP - some mechanisms are required to ensure that the fragments are reassembled correctly. Additionally, the client can "contact" the fake nameserver everytime it wants to send traffic out - but the server is only able to answer, never to send on its own. So some polling is needed in order to have real bidirectional communication. A protocol called 'NSTX' (NameServer Transfer Protocol) has claimed to achieve all this.

3.4 MAC Misbehavior in MANET

Host misbehaviors in MANET can be classified into three categories; namely, selfish misbehavior [32, 42], malicious misbehavior [3, 22] and hybrid misbehavior (selfish and malicious) [19]. Selfish hosts typically misbehave to improve their own performance; this includes hosts that refuse to forward packets on behalf of other hosts in order to conserve energy. Greedy hosts may exploit the vulnerabilities of IEEE 802.11 [1] to increase their share of bandwidth at the expense of other users. For example, IEEE 802.11 requires hosts competing for the channel to wait for backoff interval [32] before any transmissions. A selfish host may also choose to wait for a smaller backoff interval, thereby increasing its chance of accessing the channel and hence reducing the throughput share received by well-behaved users. The authors of [32] showed that such selfish misbehavior can seriously degrade the performance of the network and accordingly they proposed some modifications for the protocol (e.g., by allowing the receiver to assign backoff values rather than the sender) to detect and penalize misbehaving nodes. Similarly, the authors of [42] addressed the same problem and proposed a system, DOMINO, to detect greedy misbehavior and backoff manipulations of IEEE 802.11.

Alternatively, malicious misbehavior aims primarily at disrupting the normal operation of the network. This includes colluding adversaries that continuously send data to each other in order to deplete the channel capacity in their vicinity (i.e., causing a denial of service attack, DoS) and hence prevent other legitimate users from communicating [50]. Another example of malicious misbehaviors is the JellyFish [3]; JellyFish (JF) is a protocol compliant DoS attack, which targets closed-loop flows (such as TCP) that are responsive to network conditions (e.g., delays and loss). Although JF conforms to all routing and forwarding operations, it is capable of reducing the goodput of all traversing flows to near zero while dropping zero or very small fraction of packets.

3.4.1 Selfish Behavior

A selfish node can deliberately misuse the MAC protocol to gain more network resources than well-behaved nodes. The node can benefit from this behavior by: (1) obtaining a large portion of channel capacity (hence improved throughput); (2) reduced power consumption; (3) improved quality of service, e.g. low network latency; (4) the attack is hidden from upper layer detection and reaction systems; (5) the attack is more efficient than upper layer attacks.

Here we present a simple taxonomy of MAC layer selfish misbehaviors, introducing several techniques that do not rely on security weaknesses of the standard and are simpler and more efficient than known methods. We can divide the MAC misbehavior space into two major dimensions: (1) scramble MAC frames; (2) manipulate pre-set MAC protocol parameters.

- Selectively scramble frames (see Fig. 3.4) sent by other stations in order to increase

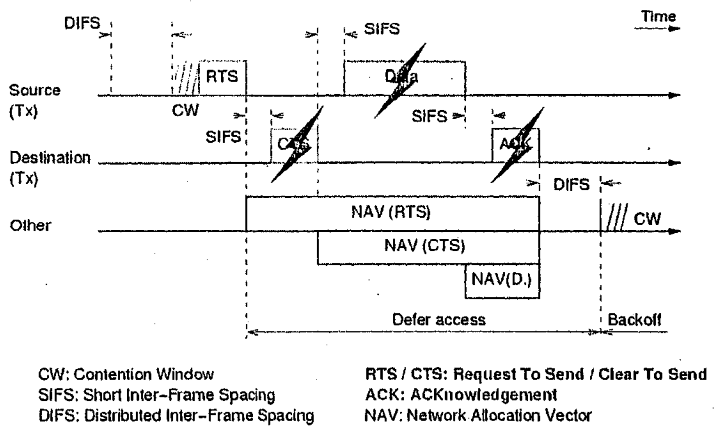


Figure 3.4: Selfish Misbehavior: scramble frames

their contention windows. The frames to be targeted can be the following:

- CTS frames. In this case the cheater hears an RTS frame destined to another station and intentionally causes collision and loss of the corresponding CTS frame in order to prevent the subsequent long frame exchange sequence (RTS/CTS handshake is used for large frames). As a result, the channel becomes idle after the corrupted CTS and the cheater gets a chance to send its data.

- ACK and DATA frames. Although this does not result in saving the data frame transmission time, it causes the contention window of the ACK destination (i.e., the DATA source) station to be doubled and consequently makes the latter select larger backoffs. As before, the cheater increases its chances to get access to the channel.

• Manipulate protocol parameters to increase bandwidth share:

  - Choose smaller interval frame space than normal value, e.g. SIFS/DIFS/EIFS [19] (see Fig. 3.5).

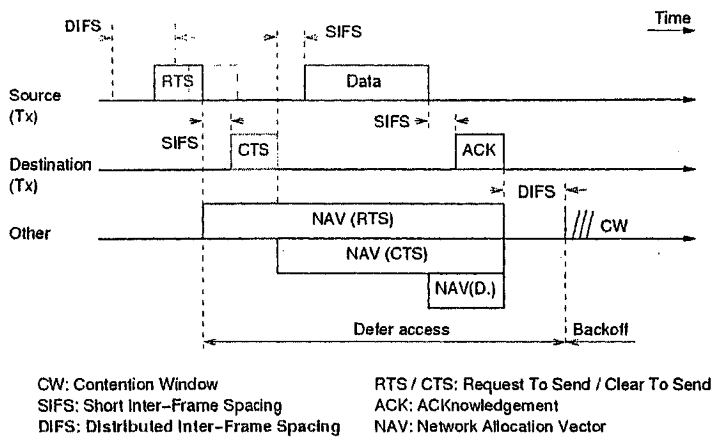


Figure 3.5: Selfish Misbehavior: manipulation of IFS

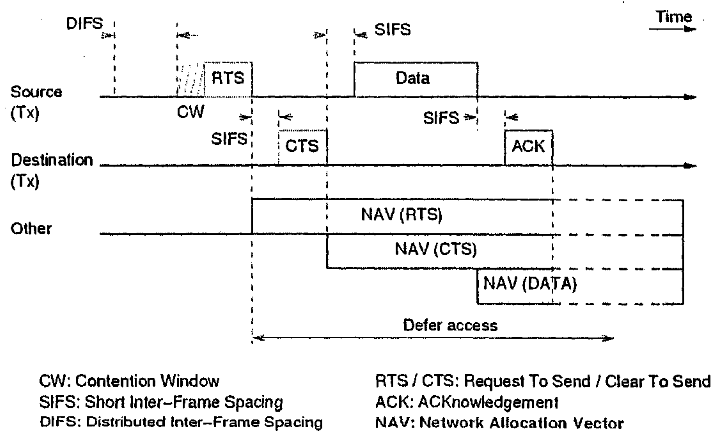
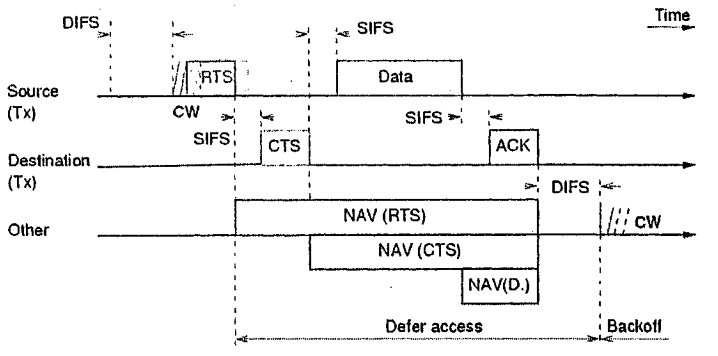


Figure 3.6: Selfish Misbehavior: manipulation of *NAV*



CW: Contention Window
SIFS: Short Inter-Frame Spacing
DIFS: Distributed Inter-Frame Spacing
RTS / CTS: Request To Send / Clear To Send
ACK: ACKnowledgement
NAV: Network Allocation Vector

Figure 3.7: Selfish Misbehavior: manipulation of *CW*

- When the channel is idle, send packets immediately or after SIFS but before DIFS/EIFS [6] (see Fig. 3.5).
- When exchanging RTS or CTS or DATA frames, increasing the NAV value in order to force other stations in range to backoff [42] (see Fig. 3.6).
- Reduce the backoff interval, e.g., always choosing a smaller *CW* instead of uniformly picking up a value from the defined range [32, 42] (see Fig. 3.7). For example, a misbehaving node can select backoff values from a different distribution with a smaller average backoff value, than the distribution specified by the DCF, e.g. by selecting *cw* from $[0, CW_{min} \times \alpha]$ $(0 \leq \alpha < 1)$ instead of $[0, CW_{min}]$ or not doubling its *CW* when collision happens.

3.4.2 Malicious Behavior

Unlike selfish misbehavior where the nodes target at gaining extra network resources, malicious nodes aim at disrupting overall network performance.

Packets Injection

In [23], the authors analyzed attacks that deny channel access by causing pockets of congestion in mobile ad hoc networks, e.g., injecting excessive raw data packets. Such attacks would essentialy prevent one or more nodes from accessing or providing specific services. In particular, the authors considered various traffic patterns that an intelligent attacker might generate in order to cause denial of service and concluded that providing MAC layer fairness could alleviate the effects of such attacks.

Drop RTS/DATA

Compromised nodes could cause devastating effects in the network by selectively dropping a number of RTS/DATA packets, which will cause the transmitter node to retransmit the same control packet several times. As we mentioned above, the transmitter will use binary exponential backoff when it does not receive the CTS/ACK in the specified time. Dropping either RTS or DATA within retry times (e.g. RTS can be retried for a maximum SSRC while DATA can be retried for a maximum SLRC) may increase the network latency. Dropping RTS/DATA beyond the limited retry times will trigger the detection of link breakage and may cause flows to change their routes, higher channel contention and congestion. Since this attack is protocol compliant, it may be hard for the transmitter to decide whether the absence of CTS/DATA is caused by a collision, interference in the channel during RTS or CTS frame, or because the receiver has an actual virtual carrier sense condition.

Colluding Attack

In [50], the authors discussed two types of malicious attacks that can be implemented by simple users with limited expertise.

- *SAA* (Single Adversary Attack) leverages unauthorized data transmission in IEEE 802.11. An intruder simply sends enormous flows to legitimate nodes, and hence drain the energy of legitimate nodes as well as substantially reduce the available channel capacity for legitimate communications.
- *CAA* (Colluding Adversaries Attack) leverages unfairness present in IEEE 802.11. Two or more colluding adversaries can send data flows directly to each other in order to deplete the channel capacity in their vicinity.

Both of these two attackers pose great threat to MANET survivability. SAA can be prevented by simple authentication scheme; however, to protect the network from CAA attack, a fair MAC protocol is needed.

3.4.3 New MAC Misbehavior: Hybrid misbehavior

we have shown that hosts could misbehave simply to achieve better share of the wireless channel by continuously selecting small backoff values upon contentions. Similarly, a compromised node that selectively drops RTS/DATA packets from other nodes could force a sender to continuously backoff and retransmit beyond the allowed retry times (e.g. RTS can be retried for a maximum SSRC while DATA can be retried for a maximum SLRC); ultimately the sender will declare a link breakage to the network layer and the routing protocol will trigger its route maintenance to establish new routes and reroute the affected flows [22]. This malicious attack, if successful, could have severe performance degradation by disrupting the route discovery process of routing schemes such as AODV and DSR. Schemes developed for mitigating similar effects (e.g., nodes that agree to forward packets but fail to do so) such as watchdog and path rater [33] could be used to detect these malicious nodes and isolate them. In the following we present one similar attack that is, however, not straight forward to detect with schemes such as watchdog.

Recall that in order to prioritize access to the wireless medium, DCF defines three time windows (SIFS, DIFS, and EIFS); only the first two are important for the purpose of our discussions. Prior to the transmission of any frame, a node must observe a quiet medium for one of the defined window periods. The short interframe space (SIFS) is used for frames sent as part of a preexisting frame exchange (e.g., CTS or ACK frames sent in response to previously transmitted RTS or DATA frames). DCF Interframe Space (DIFS) is used for nodes wishing to initiate a new frame exchange; after the channel is sensed idle for a DIFS time, a node waits for an additional backoff time after which the frame is transmitted. To completely manipulate the channel, a node could transmit a signal after a short SIFS [19] and to achieve a notable increase in the bandwidth a node could transmit after SIFS but before DIFS when the channel is idle [42] .

What happens however when a node transmit after a larger SIFS value rather than shorter one? To answer this question, one needs to take a closer look at the functionality of DCF, namely the CTS procedure and the ACK procedure [1]. When a particular host transmits an RTS, it also computes a duration field (RTS field, rf) that is transmitted in the RTS frame:

$$rf = sifs + T_{CTS} + sifs + T_{DATA} + sifs + T_{ACK} \tag{1}$$

Where, T_{CTS}, T_{DATA}, T_{ACK} are the transmission time of CTS, DATA, and ACK frames correspondingly.This duration field is used by other nodes in the vicinity of the sender to adjust their NAV value. The sender also computes a CTS timeout (TO_{CTS}), a time during which the sender expects a CTS response from the receiver host:

$$TO_{CTS} = T_{RTS} + 2\delta + sifs + T_{CTS} \tag{2}$$

Where, δ is the maximum propagation delay. If the timer expires before the arrival of a CTS packet, then the transmitter[9] infers that either interference caused the RTS to be lost, or the receiver[10] has its NAV value set (i.e., the medium around the receiver is busy). In either case, the transmission of the RTS is deemed failed and the sender will invoke its backoff procedure and schedules a retransmission for a new RTS frame. On the other hand, when a host receives an RTS, it shall wait for a SIFS period and transmit a CTS frame only if the NAV at the receiving station indicates that the medium is idle. A duration field (CTS field, cf) is also computed and transmitted along with the CTS frame:

$$cf = rf - sifs - T_{CTS} \tag{3}$$

This duration field is used by all stations in the vicinity of the receiver to adjust their

[9]In a single handshaking process, a node will play two different roles, i.e. transmitter or receiver. To avoid confusion in the following sections, we use transmitter (Tx) to refer to the source of a MAC DATA frame. A Tx is also the source of a RTS frame in case of the four-way handshaking.

[10]In the rest of the thesis, receiver (Rx) only refers to a node which is the destination of a MAC DATA frame. A Rx is the source of a CTS MAC control frame in case of four-way handshaking.

NAV accordingly. The recognition of a valid CTS frame sent by the recipient of the RTS shall be interpreted as successful response permitting the frame sequence to continue.

In this case, the sender will send a DATA frame, after waiting for a SIFS period, along with a duration field (DATA field, df):

$$df = T_{ACK} + sifs \tag{4}$$

The sender will also compute an ACK timeout interval (TO_{ACK}), after which if no ACK is received from the receiver, then the sender concludes that the DATA frame transmission failed and subsequently invokes backoff procedure and schedules retransmission of DATA.

$$TO_{ACK} = T_{DATA} + 2\delta + sifs + T_{ACK} \tag{5}$$

Note that, as mentioned earlier, IEEE 802.11 allows only for a limited retry for the transmission of both the RTS and DATA frames. In order to misbehave, a node needs only to alter the value of SIFS. Rather than selecting a small value (which will result in a selfish attack as mentioned earlier [22, 32]), a node could select a larger value for SIFS (*sifs**, larger than the nominal value, *sifs*, plus 10% of one slot time [1]) and hence force a transmitter to timeout every time it transmits either an RTS frame or a DATA frame. After successive unsuccessful retransmissions, the transmitter will drop the data packet and report a link breakage to the network layer. Here, detection systems like watchdog [11] will fail to detect this malicious misbehavior since the malicious node (receiver) is sending CTS or ACK frames, however they arrive after their corresponding timeout timers at the sender

[11] We assume that a watchdog system is capable of monitoring the link layer communication.

expire. Malicious nodes of this category aim primarily at disrupting the route discovery process from discovering routes through them; therefore forcing packets of other hosts to go through non optimal routes. As a result, such a node will conserve its battery power by refusing to forward packets of no direct interest to the node. Moreover, since flows are forced away, such a malicious node can access the medium with less contention and hence achieves a larger throughput share of the wireless channel without modifying its backoff interval. This hybrid attack is referred to TimeOut attack (TO).

3.5 Case Study

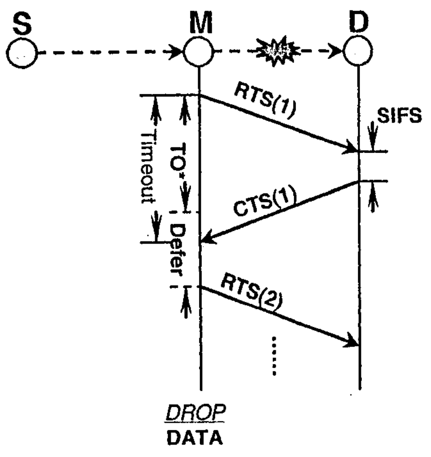
We use a simple topology with three fixed nodes to show the impact of *TO* misbehavior. There exists a data flow from Node S to Node D, where S and D stand for the source and the destination respectively. Node M is the only intermediate node of the flow S $\rightarrow$ D. All the nodes are completely identical, i.e. the same transmission range, carrier sense range, transmission power, etc. The routing protocol used here is AODV [40]. A flow first is established when node M is well-behaved; later, however, it commences its misbehavior to disrupt the existing flow. We consider the following 4 cases:

3.5.1 $sifs^* < sifs$: a misbehaved node chooses a smaller $sifs^*$

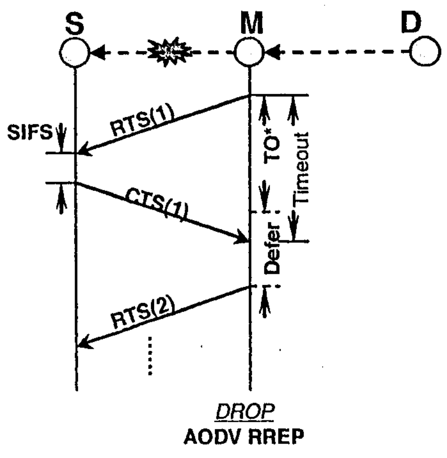
Route Breakage Process (Fig. 3.8(a))

Assume a flow already exists between nodes S and D routed through node M.

M is the transmitter and D is the receiver - When node M successfully receives a data packet from node S, it should forward the packet to node D. M will first send a RTS frame



(a) $SIFS^* < SIFS$, Route Breakage



(b) $SIFS^* < SIFS$, Discovery Failure

Figure 3.8: $SIFS^* < SIFS$

and sets its transmitter status to sendRTS [12] [1]; it will also calculate a timeout interval for this RTS as shown in the previous section (see Equation (2)). Since $sifs^*$ is smaller than the actual $sifs$, the computed timeout is $TO^*_{CTS} < TO_{CTS}$. When node D receives this RTS, it will send back a CTS frame after a $sifs$ period only if the medium is idle. Now, M will not receive the CTS frame during the timeout interval (since $TO^*_{CTS} < TO_{CTS}$) and hence it shall conclude that the RTS transmission has failed and reschedule its transmission after a corresponding backoff (to successfully follow the operation of the protocol). This same procedure proceeds and after a limited number of retries, node M will drop the DATA packet; the network layer infers that the link is broken and a route maintenance procedure of AODV is triggered to reroute the affected flow. Similarly, if the attack occurs during the DATA/ACK handshaking process, the DATA will be retransmitted over the limited retry times until ultimately the link will be claimed broken.

[12]This indicates the medium around the transmitter is busy.

Note, however, a necessary condition for a route breakage is that $(sifs - sifs^*)$ is not very small (e.g., at least $2us$). That is because TO^*_{CTS} is computed based on the maximum propagation delay δ; whereas, the actual propagation delay between M and D is $\delta' \leq \delta$. Hence, although $TO^*_{CTS} \leq TO_{CTS}$, the sender (M) may still receive the CTS before it times out if $(sifs - sifs^*)$ is very small and accordingly there will be no route breakage. In other words, a small variation of $sifs^*$ can be absorbed by the TO^*_{CTS}, which is computed according to δ.

S is the sender and M is the receiver - When S sends a RTS to M, M will wait for a $sifs^*$ and send back a CTS. Since $sifs^*$ is smaller than the default $sifs$, the CTS message will arrive during the TO_{CTS} period at the sender. Hence, the handshake will succeed and there will be no route breakage.

Note that some implementation of 802.11 [17] allows the receiver upon sending a CTS to compute a timeout for receiving the DATA from the sender, so that it does not keep on waiting for receiving DATA, TO_{DATA}:

$$TO_{DATA} = T_{CTS} + 2\delta + sifs + T_{DATA} \tag{6}$$

Since T_{DATA} is unknown at the receiver, the receiver will use the rf value (as advertised by the transmitter, Equation (1)) to compute:

$$T_{DATA} = rf - (sifs + T_{CTS} + sifs + sifs + T_{ACK}) \tag{7}$$

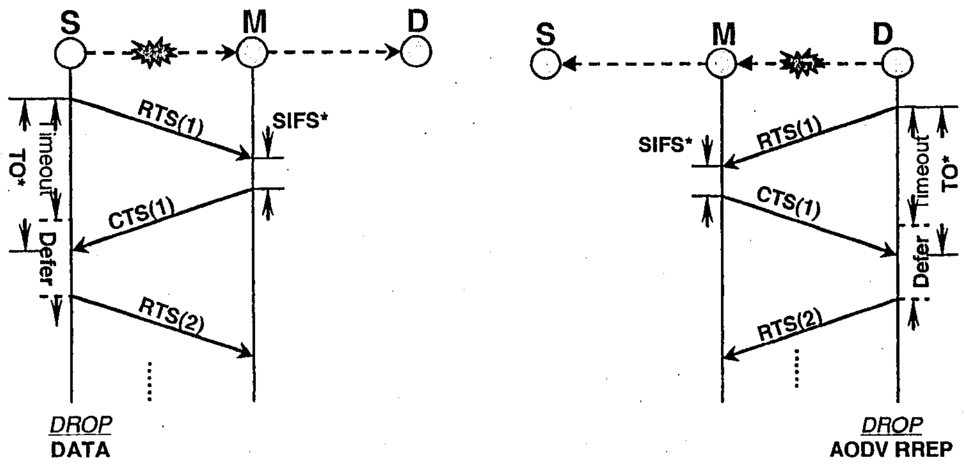
Note that since the receiver is computing equations 6 and 7, the same value of $sifs$ will be used; hence,

$$TO_{DATA} = T_{CTS} + 2\delta - 2 \times sifs + rf + T_{ACK} \tag{8}$$

Accordingly, if the receiver M selects a $sifs^* \leq sifs$, then $TO^*_{DATA} \geq TO_{DATA}$. That means, the receiver has an extended timeout and therefore DATA sent by S will be received by M.

Route Discovery failure process (Fig. 3.8(b)).

This attack aims at disrupting the route discovery process. The source node initiates a route discovery by broadcasting a route request (RREQ) control packet. Node M will receive the packet (note, the sender broadcasts the packet, hence no handshaking with M is taking place and as a result node M successfully receives the RREQ packet) and broadcast the RREQ further to its neighbors (D in this case). Node D unicasts a route reply (RREP) packet over the reverse path (D-M-S). Since node D initiates the handshake with node M, and node D has a *sifs* > *sifs**, the RREP will be accepted by node M. Node M further unicasts the RREP to node S by initiating a 4-way handshake. Here, the transmission of the RREP packet will fail because the source node (i.e., M) will timeout and reach its retry limit until eventually the RREP packet is dropped at node M. Ultimately, the RREP packet will never reach the source node S and hence no data transmission is allowed over this single route. Note that if multiple routes exist between nodes S and D, then the flow will be routed over a longer path (in case S-M-D is the shortest), which could lead to performance degradation (e.g., additional end to end delays, increased routing overhead).



(a) $SIFS^* < SIFS$, Route Breakage (b) $SIFS^* > SIFS$, Discovery Failure

Figure 3.9: $SIFS^* > SIFS$

Summary 1: when $sifs^* < sifs$, the attack is effective only when M is a transmitter and $sifs - sifs^* \geq 2us$.

3.5.2 $sifs^* > sifs$: a misbehaved node chooses a larger $sifs^*$

Route Breakage Process (Fig. 3.9(a)).

We consider, as before, the same flow exists between nodes S and D and routed through node M. We first consider the case of a *malicious receiver* M. When source S has a packet to send to D, it will start by sending an RTS frame (after waiting for a DIFS and appropriate backoff period) to M and schedules accordingly a CTS timeout. When node M receives the RTS frame, if the channel is free it will send back a CTS after waiting for a $sifs^*$ period. Since $sifs^*$ is larger than $sifs$, the CTS will arrive at the source S upon the expiration of the CTS timeout timer (we assume that $sifs^* - sifs$ is sufficiently large). When S does not receive CTS within the appropriate CTS timeout period, it will set its transmitter status to idle and schedule a new retransmission of RTS after increasing its backoff interval. Thus, when the CTS sent by M arrives at S, it will be simply dropped because the transmitter status is idle (i.e., indicating that no RTS was transmitted). After several retransmissions of RTS, the data packet from S to M will be dropped and the network layer will be notified by the MAC layer of a link breakage along the route. However, when node M is a *malicious transmitter*, the handshake between M and the receiver will succeed since the $TO^{*}_{CTS} \geq TO_{CTS}$ (i.e., M has enough time to receive the CTS from D) or alternatively the receiver will have larger TO_{DATA} (i.e., enough time for D to receive DATA from M).

Route Discovery failure process (Fig. 3.9(b)).

Similar to case (2), when source S initiates a route discovery, it broadcasts a RREQ to its neighbors (M in this case); M in turn broadcasts the RREQ to its neighbors (D in this case). When node D receives the broadcasted RREQ, it unicasts a RREP back to the source S over the collected route (D-M-S). For this reason, D sends an RTS frame to node M. This transmission will fail since node M will wait for a period $sifs^{*} > sifs$ to transmit its CTS frame (if the medium is idle) and accordingly the CTS arrives at node D after the timeout. After a limited number of unsuccessful retries, node D will eventually drop the RREP packet and the route discovery process is disrupted (unless there is an alternate route from S to D that does not traverse node M).

Here, note that node M successfully cooperates in the forward discovery process; however, it then succeeds in hiding itself in order to force the data flow to be routed away, which otherwise would have been routed through the shortest path passing by M (in case

Table 3.1: Parameters used in *TO* attack

Notation	Usage
CW_{min}	31
CW_{max}	1023
$SlotTime$	$20us$
$SIFS$	$10us$
$MaxPropDelay$	$2us$
$PreambleLength$	$144bits$
$PLCPHeaderLength$	$48bits$
$PLCPDataRate$	$1Mbps$
$ShortRetryLimit$	7
$LongRetryLimit$	4

the shortest path is S-M-D). Additionally, node D will do no further actions after dropping the packet. Since node M is cooperative in forwarding downstream routing packets, it completely renders detection systems like watchdog ineffective.

Summary 2: when $sifs^* > sifs$, the attack is effective only when M is a receiver and $sifs^* - sifs \geq 2us$.

Summary 1 and Summary 2 are obtained based on the parameters we used in our simulation (see Table 3.1), the value difference between $sifs^*$ and $sifs$ might be slightly changed under different δ ($MaxPropDelay$) as described above.

3.6 Detection and Reaction to MAC Misbehavior

3.6.1 Detection and Reaction to Selfish Behavior

The IEEE 802.11 [1] defines a Distributed Coordination Function (DCF) as a basic access method; DCF is designed under the assumption that all participating nodes are well behaved. With the implementation of MAC protocol in software rather than hardware or firmware in network access cards, it is more likely to modify the protocol by a selfish or malicious node [42]. Simple changes of several protocol parameters in one or a set of nodes can have devastating effects on the overall network performance which could lead to Denial of Service (DoS). While a well-behaved node strictly obeys the pre-defined protocol operation, the misbehaving nodes may deviate from the standard to either cause unfairness problems or disrupt the network services. This misbehavior may be hard to distinguish from some normal cases. A misbehaving node may keep on sending packets in order to reduce the chance of a node with light load to transmit. A node may send large amount of packets to a specific victim (or other nodes with the victim being a forwarding node) thus draining out the energy of the victim. Two nodes may also collude with each other to establish a flow with continuous data transmission, which can deplete the channel capacity in their vicinity [50]. A selfish node may adjust its backoff in different ways to access the channel with higher probability. One way is to choose a small backoff value rather than a valid generated random number by the backoff algorithm, e.g. using range [0, $CW/2$] rather than [0, CW] or always generating small random value regardless of the range. In the presence of a collision or busy medium or retry, the selfish node will have more chance to win the channel than other nodes. A selfish node may also set longer time duration than the actual transmission time in its RTS/CTS. Those nodes that overhear the exchange will have to adjust their Network Allocation Vector (NAV) accordingly and consequently defer longer time before transmission. Or it can even adjust the DIFS or SIFS time (by selecting smaller values) to further exacerbate the unfairness.

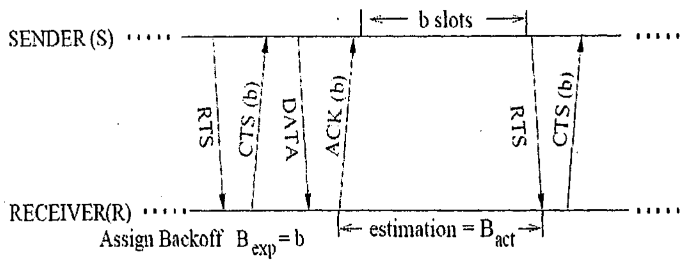


Figure 3.10: Operation Procedure: normal scenario

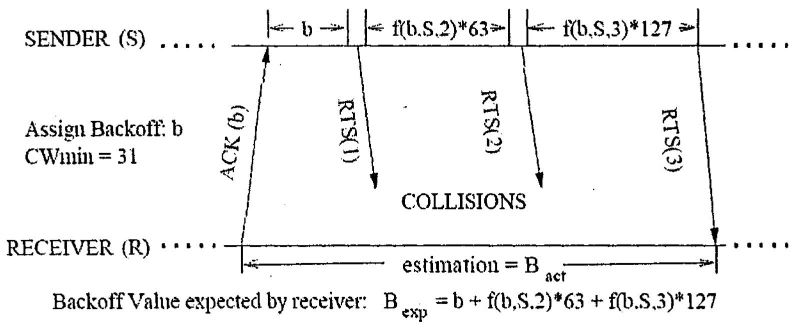


Figure 3.11: Operation Procedure: collision scenario

Kyasanur et al [32]

The authors of [32] proposed modifications to 802.11 for facilitating the detection of selfish nodes. Here, the receiver assigns backoff values to the sender and monitors the sender for any potential misbehavior. If the sender's actual backoff interval B_{act} deviates from the assigned backoff B_{exp}, i.e., $B_{act} \leq \alpha \times B_{exp}, 0 < \alpha < 1$, the sender will be designated as misbehaved node for a certain number of transmissions. In case of misbehavior, the receiver penalizes the sender by increasing its backoff values for next transmissions. If the sender deviates repeatedly, then it is considered as misbehaving and appropriate measures are taken in order to isolate the host.

Fig. 3.10 depicts the receiver-sender interaction in the modified protocol. When the receiver R receives a RTS from the sender S, R assigns a backoff value $B_{exp} = b1$ to S in the CTS packet as well as the subsequent ACK packet as shown in Fig. 3.10 (the assigned backoff may be included in either of CTS or ACK when RTS/CTS exchange precedes data transfer). S is required to use this backoff value $b1$ for sending the next packet to R. If S does not backoff for the assigned time, then based on $B_{act} < \alpha \times B_{exp}, 0 < \alpha \leq 1$, node S will be designated as misbehaved node if B_{act} is larger than $\alpha \times B_{exp}$.

Fig. 3.11 demonstrates the working of this scheme after a collision. In the figure, the number in parenthesis next to the RTS is the value of the attempt number. When a RTS from the sender is unsuccessful, it selects a new backoff using a deterministic function f, of the backoff previously assigned by the receiver (backoff), the unique node identifier (*nodeId*), the attempt number (*attempt*) and contention window (CW) as follows (in Fig. 3.11, $backoff = b$, $nodeId = S$, and the attempt numbers are 1, 2 and 3.):

$NewBackoff = f(backoff, nodeId, attempt) \times CW$

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

The main drawback for this approach in ad hoc networks (as opposed to infrastructure-based wireless networks) is that hosts may not be trusted and hence a receiver itself may misbehave by assigning different backoff values to different senders (colluding attack) or by overhearing neighboring transmissions and selecting appropriate backoff values to cause collisions.

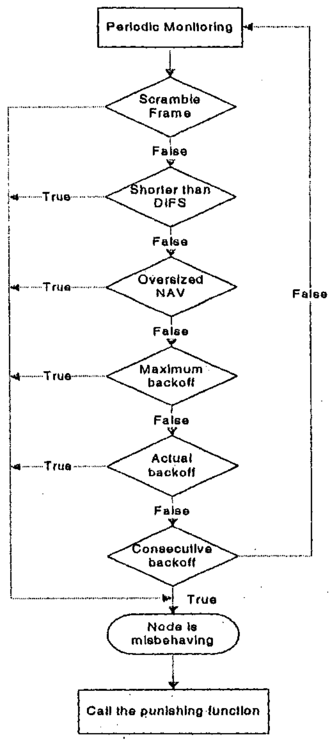


Figure 3.12: Operation Procedure of DOMINO

DOMINO [42]

In [42], the authors presented a detection system called DOMINO (system for Detection Of greedy behavior in the MAC layer of IEEE 802.11 public NetwOrks) that does not require any modification to the MAC protocol and they presented several procedures for detecting misbehaviors that aim at altering protocol parameters (e.g., shorter than DIFS, oversized NAV, and backoff manipulation). The system is implemented at the AP (access point) and the AP is assumed to be trusted; traffic traces of sending hosts are collected periodically during short intervals of time called monitoring periods. This gathered data is then passed to several tests within the DOMINO algorithm. Each of these tests corresponds to a designated misbehavior (e.g., Backoff manipulation, oversized NAV, etc.). The result of each test is then fed into a *check* procedure, which in turn will infer whether a particular

Table 3.2: Components of DOMINO

Test 1 : Scrambled Frame
$condition_1 := num_{rtx}(S_i) < \phi \times E_{j \neq i}[num_{rtx}(S_j)]$
call **Check**$_1(S_i, condition_1)$
Test 2 : Shorter than DIFS
$condition_2 := idle\_time\_after\_ACK(S_i) < DIFS$
call **Check**$_2(S_i, condition_2)$
Test 3 : Oversized *NAV*
$condition_3 := NAV(S_i) > A \times tx_{duration}(S_i)$
call **Check**$_3(S_i, condition_3)$
Test 4 : Maximum Backoff
$condition_4 := Max_{backoff}(S_i) < Threshold_{Max\_backoff}$
call **Check**$_4(S_i, condition_4)$
Test 5 : Actual Backoff
$condition_5 := B_{act}(S_i) < \alpha \times B_{act\_norm}$
call **Check**$_5(S_i, condition_5)$
Test 6 : Consecutive Backoff
$condition_6 := B_{co}(S_i) < \alpha \times B_{co\_norm}$
call **Check**$_6(S_i, condition_6)$

station is misbehaving or not. A node misbehaves when its corresponding *cheat* counter exceeds a certain threshold (i.e., to reduce the false positives). Misbehaving nodes are then punished using a punishing function. Fig. 3.12 shows the general operation procedure of DOMINO.

The detailed algorithms to detect various selfish misbehavior techniques have been described in Table 3.2. Based on these algorithms, the detection system implemented near or on the APs can periodically monitoring the behavior of senders. If their behavior deviates from the standard, DOMINO will increment the corresponding local counter unless the counter exceeds certain threshold.

ERA [13]

The authors of [13] highlighted the MAC selfish misbehavior and proposed extensions for the detection system of [32] under the assumption that at least one of the parties involved is honest. Their approach follows that of [32] wherein the receiver assigns a backoff value for the sender; however, both sender and receiver will exchange some additional commitment information to ensure complete randomness and to verify that none of the hosts is misbehaving prior to the assignment. Any detected misbehavior (whether from the receiver or the sender) is reported to a reputation management system.

3.7 Conclusion

In this chapter, we reviewed the current security problems in MANET MAC layer and the existing detection and reaction systems against MAC misbehaviors. We presented a new type of MAC misbehavior, i.e., hybrid misbehavior, and describe the operation of one specific example (TO attack) of this kind of misbehavior in details.

In Chapter 4, we will evaluate the impact of these misbehaviors on the network performance. In Chapter 5, we will present our new detection system [20] *DREAM* (a system for Detection and REAction to timeout mac layer Misbehavior) that identifies the malicious nodes through a set of monitoring and reaction procedures. Once a misbehaving node is detected, the system reacts, by adapting simple protocol parameters, to mitigate the negative effects.

Chapter 4

MAC Misbehavior Impact on Routing Performance

4.1 Introduction

As mentioned in previous chapters, significant research efforts have been made towards increasing the survivability of MANET either by developing secure routing algorithms or by improving the robustness of MAC layer protocol in the presence of selfish or compromised nodes. In addition, some recent studies have focused on quantifying the resiliency of MANET against MAC layer misbehaviors; however little work has been done on quantifying the impact of these misbehaviors on the performance of ad hoc routing protocols. In this chapter, we will study the impact of MAC layer misbehavior on the ad hoc routing protocols, more specifically AODV and DSR. Via computer simulation, we show that simple MAC attacks on the link layer can propagate to the upper network layer and disrupt the routing mechanism, therefore cause devastating negative effects on the performance of

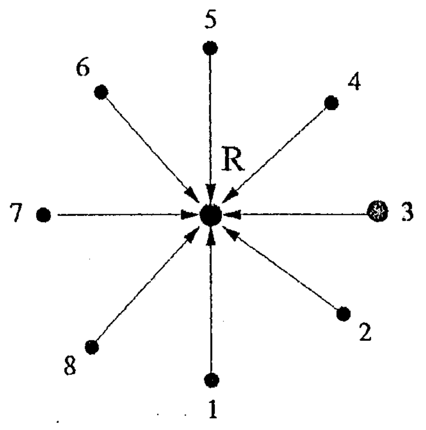


Figure 4.1: A simple 8-node star topology

well-behaved nodes as well as the whole network. Although several secure routing protocols have been proposed, none of them are resilient to MAC misbehaviors.

We compare the performance of DSR and AODV in terms of: (1) the average end-to-end throughout, (2) the average packet delays, (3) the normalized routing load which is defined as the number of routing packets transmitted per data packet delivery. In Section 4.2, we evaluate the impact of selfish misbehavior on routing performance. In Section 4.3, we evaluate the impact of malicious misbehavior on routing performance. Last, we evaluate the impact of hybrid misbehavior impact on routing performance in Section 4.4.

4.2 Impact of Selfish Misbehavior on Routing Performance

4.2.1 Example of Selfish Behavior Impact on Routing

Fig. 4.1 shows a simple network topology to illustrate the effects of MAC misbehavior on network layer performance. Nodes (1, 2, ... 8) all have equal distance to node R. These eight nodes are uniformly distributed on the ring. R is the center of the ring, which is the destination for all the eight source nodes. Node 3 is the only selfish node in this setup. We change the selfish node minimum contention window (CW_{min}) to $\alpha \times CW_{min} (0 \leq \alpha \leq 1)$, where α is a misbehavior coefficient; $\alpha = 1$ means there is no misbehavior.

First, we show a scenario where a source S (node 3) starts its transmission first and node 2 starts after. Nodes 4 to 8 are removed from the topology in this case. Nodes 1, 2, and 3 are within transmission range of each other. In DSR, S will first broadcast a non-propagating RREQ with hop limit count set to 0 (if no RREP receives, it will broadcast a second RREQ with hop limit count increased). Nodes R, 1 and 2 will receive this non-propagating RREQ. Since R is the destination as well as the neighbor of 3, it will send back a RREP. Nodes 1 and 2 will also learn that S (i.e., node 3) is their neighbor. After receiving this RREP, 3 will start transmitting its packets to R. Node 2 by overhearing the ongoing transmission will discover its neighbor R. After some time, node 2 tries to communicate with R. It will first look at its cache and find the route $2 \rightarrow$ R and will use it for transmission. When there is no attack (i.e., node 3 is not misbehaving), nodes 2 and 3 will fairly share the bandwidth. However since node 3 is a selfish node, it will have higher chance of capturing the shared medium. Node 2 will inevitably face the situation where no CTS/ACK is received due to collision. Node 2 will double its contention window size and schedule to retransmit again. After a number of retransmission failure (6 times for short frame retransmission, 3 times for long frame retransmission [1]), it will be notified of a link breakage from the MAC layer. Node 2 will have to choose another path to R. By looking at its cache, it will find and use the route 2-3-R. Now, if node 3 has a higher sending rate, this alternate route can also fail and node 2 will use new route discovery by sending RREQ. Nodes R, 1, and 3 will receive this RREQ; 1 and 3 will generate RREP first because they are closer to 2 with node

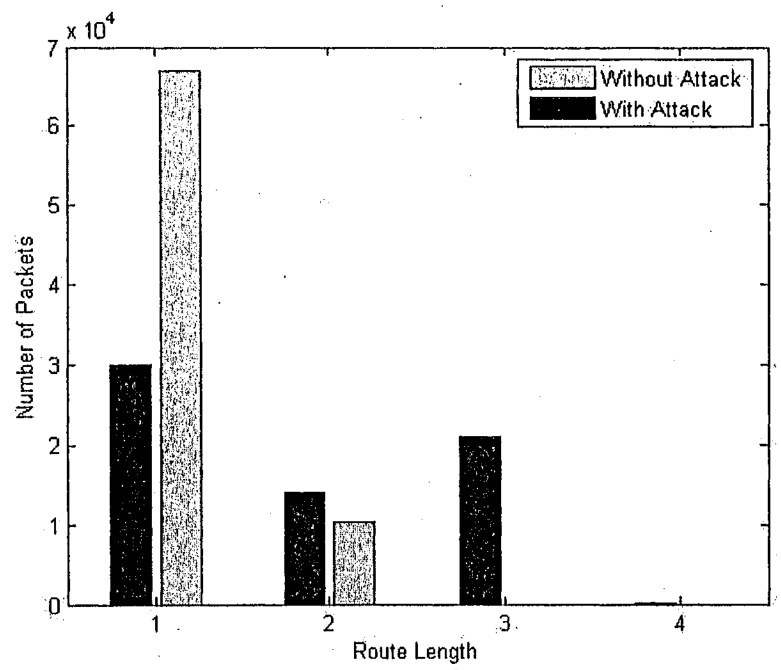


Figure 4.2: Example: Packet Distribution vs. Route Length

3 having higher chance to send the RREP before node 1. Once node 2 receives the RREP, it will start its transmission. Similar behavior occurs had AODV been used instead.

From this example, we can see two interesting things. First, traffic of well-behaved nodes will experience an increase in the length of its route to the destination which will definitely cause performance degradation (e.g., increased delays). Second, the selfish node will attract more traffic from disrupted neighboring flows and hence will forward more packets for other nodes ultimately draining its own energy. Furthermore, things can get more severe if eight flows are all turned on to contend for the same channel. More collisions will happen which could lead to a number of retransmissions and ultimately to many links failures. Traffic of a flow will change its original one-hop route to a longer path. Fig. 4.2 shows that the hop count for the eight flows have changed to traverse longer routes, while in normal case one-hop route is preferred for each of these eight flows.

4.2.2 Simulation Results and Analysis

Simulation Setup

We use the ns-2 network simulator [17] with CMU Monarch mobile extension for our simulation. The "random waypoint" model is used to generate mobility scenarios.

One 64-packet sending buffer is used for both routing protocols and each node maintains an interface queue with size of 50. The maximum source route length for DSR is fixed at 16 hops. Expanding ring search is used.

Here we only show the static scenario where the pause time is fixed to the total simulation time. A small size network with 50 nodes in a $1500m \times 300m$ rectangular space is used. We use CBR traffic and the packet size is fixed to 512 bytes. The offered load of the system can be changed by varying the number of sources and sending rate: (1) 10 sources with 4 packets / second; (2) 30 sources with 4 packets / second.

Simulation Results

We consider two cases: (1) persistent misbehavior, wherein a selfish node keeps misbehaving with the incentive of increasing its throughput. (2) Periodic misbehavior where a node's motive is rather to disrupt or disturb the neighboring flows (and hence impact the network services) while reducing the chances of being detected by misbehaving for only short period of times. We use AODV in our current evaluation. The selfish node selects a backoff from the range $[0, \alpha CW_{min}]$, where α $(0 \leq \alpha \leq 1)$ is a misbehavior coefficient and $CW_{min} = 31$; $\alpha = 1$ means there is no misbehavior. We consider a network with three flows, one selfish and two normal. The data rates are 25 packets per second for the normal flows

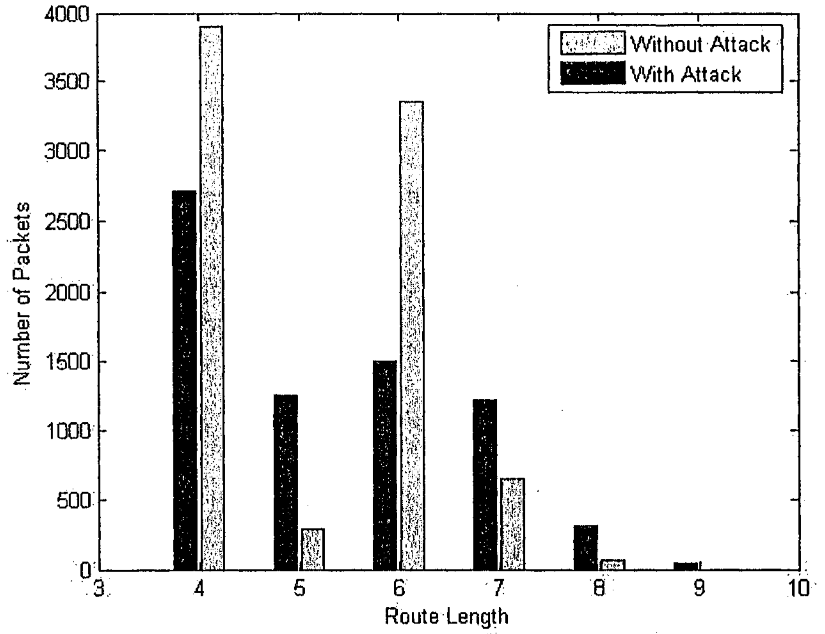


Figure 4.3: Backoff Attack: Packet Distribution vs. Route Length (1)

and the same rate is used for the selfish flows.

As we have shown in Section 4.2.1, a selfish host can disrupt the communication of a normal flow whose path is in the vicinity of the selfish node, causing the source node of the normal flow to find an alternate path around the failed link to resume its communication. Fig. 4.3 shows the packet distribution vs. the hop count for the two normal flows (flows 1 and 2) with $\alpha = 0.1$ in a large network with 50 nodes. The figure shows that under normal operation (i.e., no selfish hosts), most of the packets are carried by their shortest paths (4-hop path for flow 1 and 6-hop path for flow 2). However, when the source node of the third flow starts misbehaving, the original routes of flows 1 and 2 will fail and many more packets will be forced to travel through new and longer routes. Note that our intention from this experiment is to show that the selfish node has effectively caused a link failure (and hence a route failure) by preventing a neighboring node (a node along the original route of

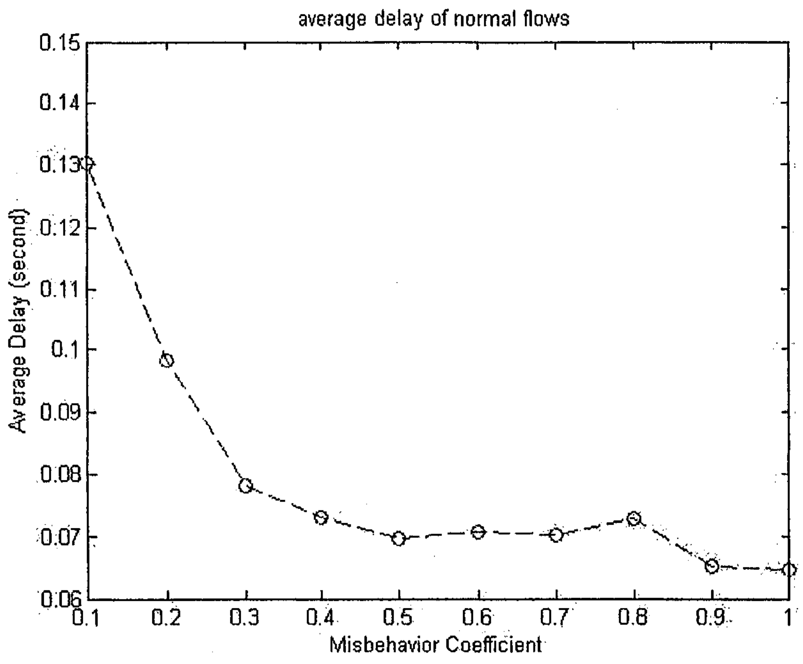


Figure 4.4: Backoff Attack: Average Packet Delay

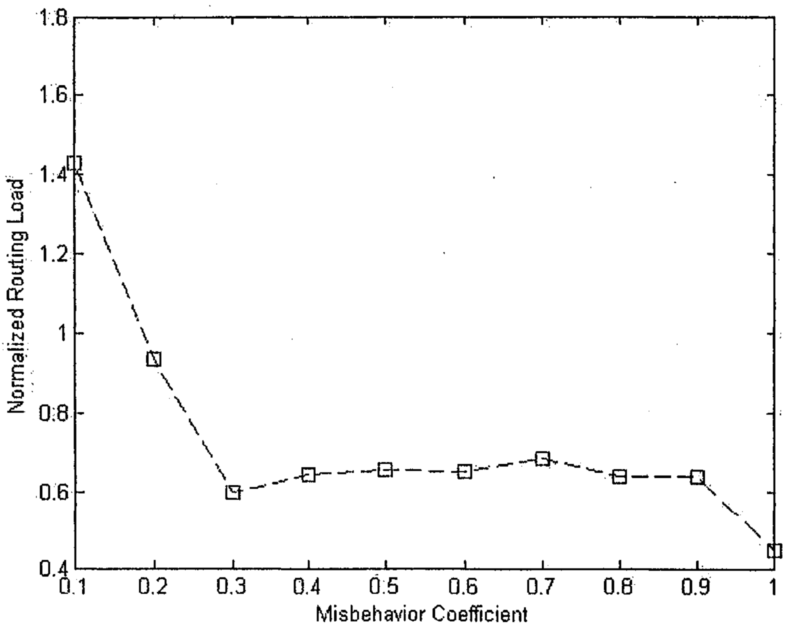


Figure 4.5: Backoff Attack: Normalized Routing Load

flow 1 or flow 2) from accessing the channel. Moreover, we only considered one selfish flow in this case study; more flows (with more selfish ones) will cause more damage in the network. One obvious and immediate symptom of this misbehavior is an increase in the average delays for the normal flows, as shown in Fig. 4.4.

Now, link breakage due to the malicious node will trigger the route maintenance of AODV to generate RERR packets to inform the source of the flow of a link failure. The source will then initiate a route discovery to find a new path to the destination. Hence, this causes an increase in the normalized routing load as shown in Fig. 4.5. The figure shows an increase in the routing load from 0.1 (no malicious behavior, $\alpha = 1$) to 0.4 when $\alpha = 0.1$. This clearly shows that a selfish node can disrupt the network services, forcing traffic flows in its vicinity to time out after unsuccessful number of retries to capture the channel and ultimately trigger new route discovery.

Finally, we present a simulation study on the topology presented in Fig. 4.1 to show the number of additional packets the selfish node (node 3, see Fig. 4.1) is forwarding in addition to its own. Here, there are two flows ($2 \rightarrow R$ and $3 \rightarrow R$) in the network. Initially both flows follow the routes $2 \rightarrow R$ and $3 \rightarrow R$ correspondingly. When node 3 misbehaves, the flow $2 \rightarrow R$ will be rerouted through node 3 (as explained in Section 4.2.1). If node 3 periodically misbehaves, then the normal flow will be sending data over its new route (2-3-R). This is clearly shown in Fig. 4.6 where under no attack, node 3 will not be forwarding any packets for the normal flow (flow 1), i.e., number of forwarded packets is 0. Whereas, when node 3 periodically misbehaves, then a large number of packets will be diverted from the shortest route (2-R) to be routed through node 3. Alternatively, if node 3 continuously misbehaves then the first flow will be rerouted through node 3 due to link (2-R) failure; however, node

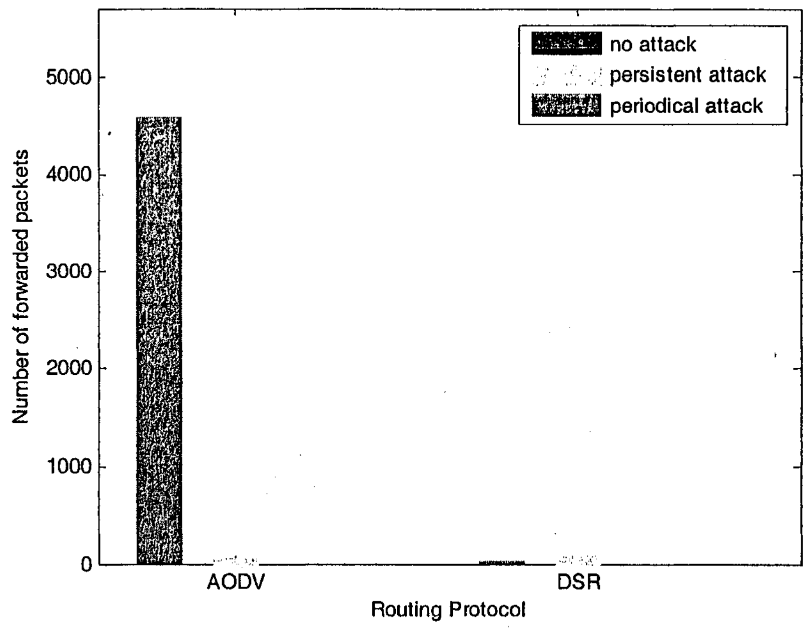


Figure 4.6: Backoff Attack: Packet Distribution vs. Route Length (2)

2 will have very small chance of capturing the channel since node 3 is actively sending its own traffic to node R. This is explained by the simulation results presented in Fig. 4.6. Only few packets from the normal flow will be forwarded by node 3 (when it persistently misbehaves) while link 2-R is effectively broken. This will cause the interface queue at node 2 to overflow and eventually packets will be dropped.

On the other hand when DSR is used instead, if node 3 is periodically misbehaving, the flow (2-R) will be attracted to its new route (2-3-R) as shown with AODV. However, when node 3 behaves normally, the flow (2-3-R now) will later switch to its shorter route (2-R). This is due to the fact that DSR discovers new routes from current transmission (i.e., node 2 learns route 2–R by observing the transmission between nodes 3 and R, as explained in section 5). Alternatively, if node 3 is persistent in its misbehavior then, similar to AODV, packets of flow 2-R will be dropped from the interface queue of node 2. One notable observation is that although a node may increase its throughput share by misbehaving, the selfish node may end up attracting other flows to be routed through itself. The node will then be forced to forward a large amount of packets in addition to its own (especially in a highly congested environment) ultimately facing the risk of draining its own energy.

4.3 Impact of Malicious Misbehavior on Routing Performance

4.3.1 Simulation Setup

We use ns-2 simulator and the traffic type is CBR with the data packet size is 512 bytes. The routing protocols to be compared are AODV and DSR, each has a 64-packet sending buffer which buffers packets waiting for a valid route. Each node also maintains a 50 bytes interface queue, which buffers all packets to be sent. The interface queue is a priority queue, which means routing packets have higher priority than data packets.

4.3.2 Simulation Results and Analysis

In this section we evaluate the ad hoc routing performance under one simple but common malicious MAC misbehavior: periodically dropping RTS/DATA packets. A node can retransmit RTS frames a maximum number of 6 times and DATA packets a maximum of 6/3 times for short/long frames. We vary the number of source-destination pairs which will also change the offered load in the system: (1) 10 sources with 4 packets per second; (2) 30 sources with 4 packets per second. The flows are setup incrementally with the last starting

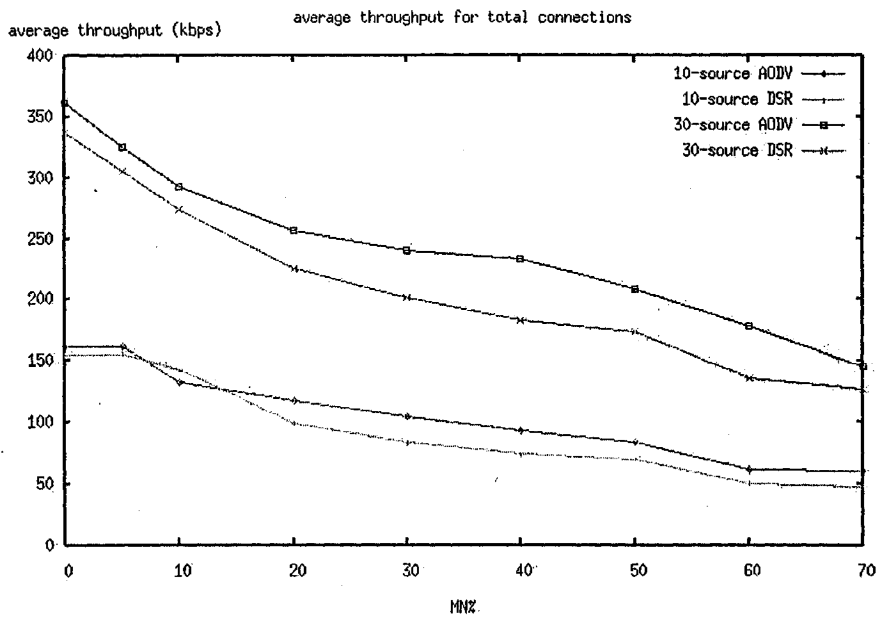


Figure 4.7: Drop RTS/DATA: Average Throughput

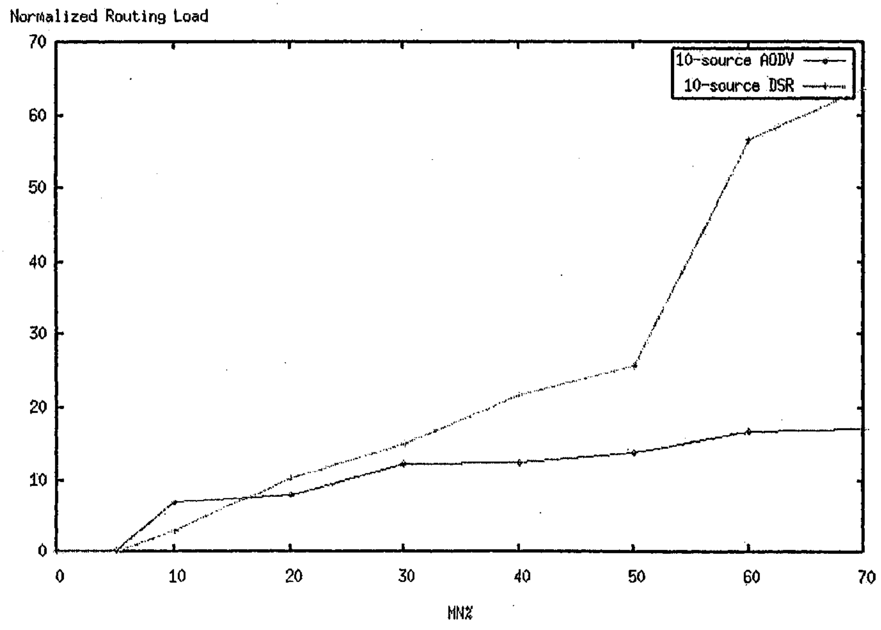


Figure 4.8: Drop RTS/DATA: Normalized Routing Load

at 180 seconds. The attack starts at 300 seconds in our simulation. A fraction of malicious nodes is randomly selected. We only discuss the scenario that a malicious node will drop 7 times RTS and 4 times DATA.

We use a pause time of 900s throughout the simulation. Fig. 4.7 shows a comparison of the routing protocols for different traffic loads (namely, 10 and 30 source-destination pairs) and under different percentage of MNs. Clearly, and as expected, AODV shows better performance than DSR at higher loads with DSR slightly performing better when the network is free of MNs. As the fraction of misbehaving nodes increases, the network performance degrades; this is due to the fact that links will break more often, therefore triggering route maintenance more frequently. Consequently, the nodes queues (Interface queues and network layer buffers) will overflow and start dropping large fraction of packets.

Fig. 4.8 shows the normalized routing load for both protocols. The normalized routing load is a very good parameter to measure the performance of ad hoc routing under MAC misbehavior. Here, the lower the normalized routing load, the better is the performance. Fig. 4.8 shows that DSR outperforms AODV at lower MNs percentage (15%). The reason for this is that DSR makes use of its route caching to find alternate paths whereas AODV resorts to discovering new paths every time a route breaks. This in turn will increase the routing overhead of AODV. Note that at lower percentage of MNs, some of the routes maintained in the caches may still be operational and therefore a node retrieves a route from its cache with high probability of not being stale [22]. Alternatively, when the percentage of MNs increase in the network, all routes in the caches will become stale and therefore DSR will be forced to initiate a new route discovery for broken flows. The figure clearly shows the large increase in the routing overhead, which result in higher routing load for

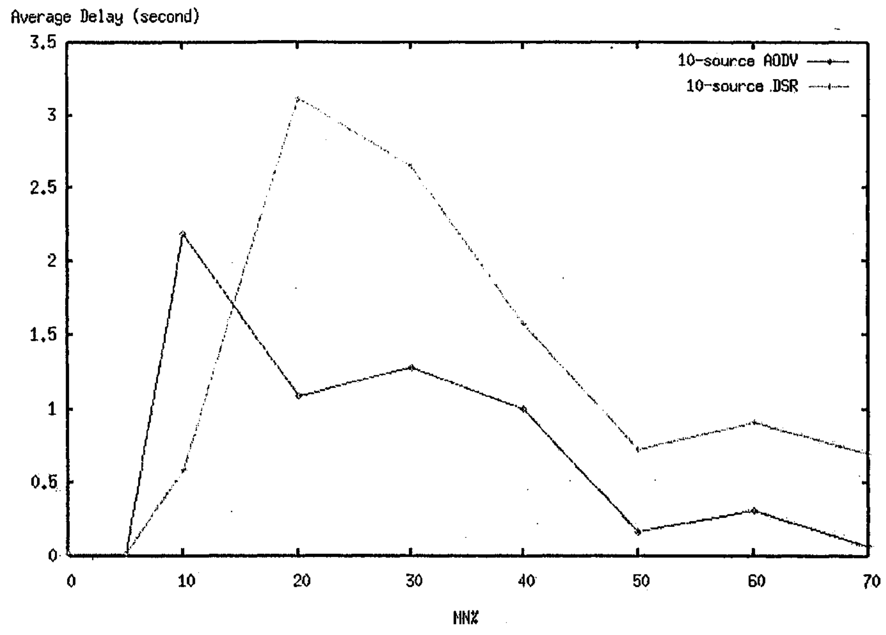


Figure 4.9: Drop RTS/DATA: Average Packet Delay

DSR.

Fig. 4.9 shows the performance of the two routing protocols in terms of average delay. The two protocols exhibit similar behavior; that is the average packet delays increase as the perçentage of MN increases; then after reaching a maximum, the average packet delay starts gradually decreasing. This is explained by the fact that when the attack starts, traffic flows that have been routed through their shortest paths will be forced to discover new longer paths since their shortest paths have failed. Hence, the delay starts increasing to reach a maximum. When the percentage of MNs become sufficiently large, the nodes almost become dispersed and the network is virtually divided into islands with no links between them. Hence, the packets will be prevented from even taking longer routes and therefore packet delays (as well as network throughout, as we showed before) will decrease.

This behavior is also explained in Fig. 4.10. The figure shows the number of packets

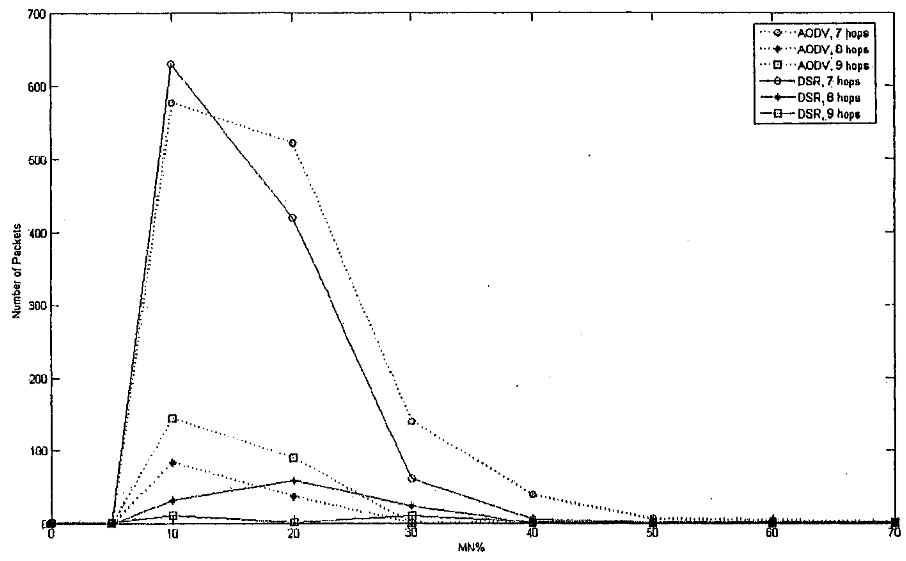


Figure 4.10: Drop RTS/DATA: Impact of Attacks on Route Lengths

delivered to their destination (under different fractions of MNs) over different path lengths for the 10 different flows. For example, when there is no attack, all packets travel through paths shorter than 7 hops. However, as some nodes start misbehaving, packets are forced to be delivered through longer paths. Finally, as more and more nodes misbehave, all longer paths will be blocked from delivering packets to their destinations (i.e., alternate routing has no benefit under such circumstances).

4.4 Impact of Hybrid Misbehavior on Routing Performance

4.4.1 Simulation Setup

We use ns-2 with CMU Monarch project multi-hop wireless extension. We use CBR traffic and the data packet size is 512 bytes. The data rate for each CBR flow is 8 packets/second (this is a relatively low data rate for which backoff manipulation attack is not very effective). The routing protocol under consideration is AODV. Every node has a 64 packets sending buffer which buffers packets waiting for a valid route to be established and a 50 packets interface queue, which buffers all packets to be sent at the link layer. The interface queue is a priority queue, which means routing packets have higher priority than data packets. The channel bit rate is 2Mbps and the total simulation time is 200 seconds. Initially, all nodes are well behaved; soon after establishing the flows (i.e., after 50 seconds), some percentage of the nodes will misbehave in order to disrupt the network services. For simulating the TO attack, we use a randomly generated network topology with 50 nodes; a node misbehaves by choosing a $sifs^*$ value of 13 us (nominal $sifs$ value is 10 us). For simplicity, the position for each node is fixed; every node has a transmission range of 250m and a sensing range of 550m and there is a total of 10 flows in the network.

4.4.2 Simulation Results and Analysis

Fig. 4.11 shows the number of packets forwarded by a set of 25 nodes (out of the 50 nodes in the network) during the total simulation time. When the percentage of misbehaving nodes (MN) is 50%, these 25 nodes are all malicious; when the percentage is 10%, 5 nodes out of these selected 25 are malicious. First, from the simulation we notice that after the attack starts, all flows routed through malicious nodes will be disrupted (for the reasons we explained in Chapter 3.5) and hence those flows will be forced to find alternate longer routes away from the misbehaving nodes. Therefore, the malicious nodes will not forward any more packets after the attack starts. This figure shows that when the percentage of malicious

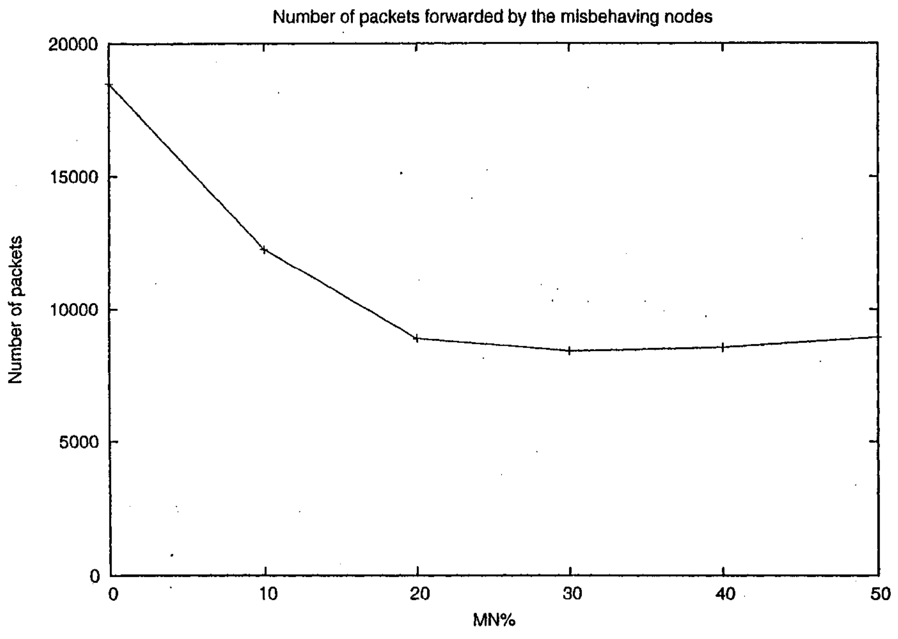


Figure 4.11: Data packets forwarded by the hybrid misbehaving nodes

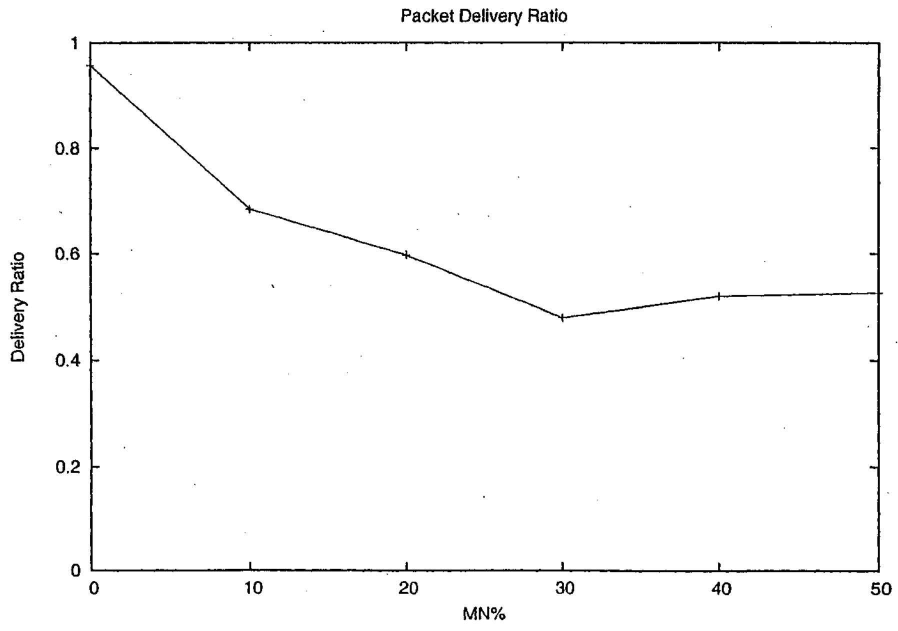


Figure 4.12: Hybrid Behavior: Packet Delivery Ratio

nodes is 20% of the total number of nodes, the number of packets forwarded by the nodes (25 nodes) is almost 9000 packets less. This results in devastating effects on the network performance as shown in Fig. 4.12. The figure shows the delivery ratio of the network during the simulation time; the malicious nodes cause a drop of almost 50% in the number of packets delivered as the percentage of MN increases beyond 20%. This experiment shows that such an attack could cause serious damage to the network throughput. Here, as the percentage of MN starts to increase (e.g., 10%), some of flows will be disrupted and hence rerouted through longer routes around MN nodes. However, further increasing the number of MN in the network would even block flows from being routed through longer paths. In other words, after the flows are disrupted the routing protocol attempts to find alternate routes for these flows, the route discovery will fail due to the existence of other malicious nodes along the routes. This is explained by the results obtained in Fig. 4.13. The figure shows that when 10% of the nodes are malicious, some flows are broken and rerouted through longer routes (please see the larger average packet delays in the figure). As this percentage increases, longer paths will be blocked and hence the average packet delay decreases. Only few flows with short routes are admitted/allowed in the network.

To see the impact of different values of $sifs$, we conduct another experiment on the same network where only 20% of the nodes are malicious and we measure the number of packets forwarded by these malicious nodes during the simulation time. Fig. 4.14 shows the result of our experiment; as expected from our explanation in Chapter 3.5 , a node that selects a $sifs^*$ outside the range of *sifs* ±10% of *slotTime* (*slotTime* is 20us) will interrupt the routing protocol from its normal operation and cause damage to crossing flows. The number of packets forwarded by malicious nodes is around 1500 packets (i.e., 3500 packets

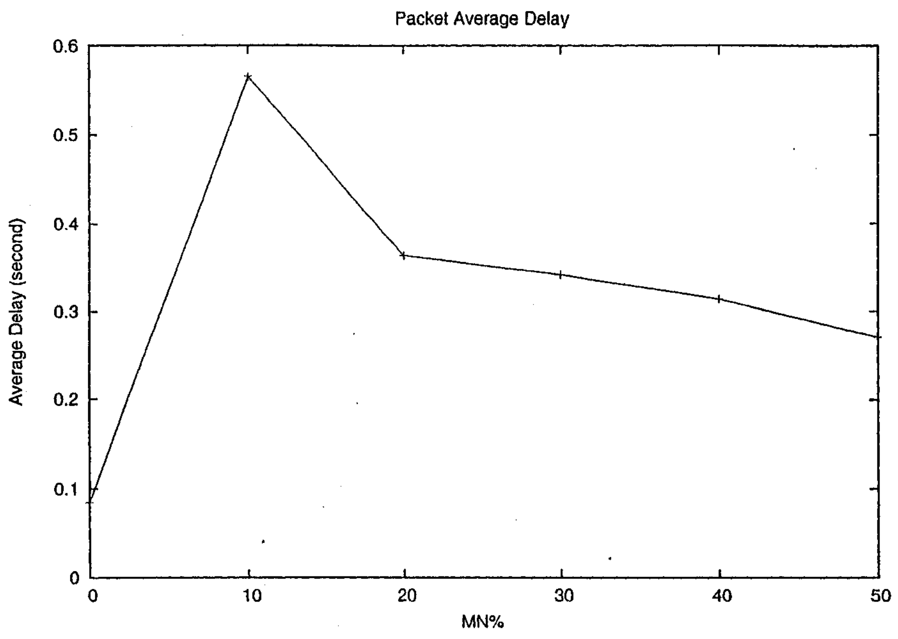


Figure 4.13: Hybrid Behavior: Average Packet Delay

less than the normal case); again these packets are forwarded before the attack takes place.

Next we consider the case where the misbehaving node has traffic of its own to send to the network. For this reason, the misbehaving node selects a larger $sifs^*$ for crossing flows and uses the nominal $sifs$ value for its local traffic. In this way, they can force the crossing flows to change their routes to other nodes and send their own traffic. Moreover, we also consider the case where the misbehaving node manipulates its backoff interval selection by choosing a smaller backoff and compare the performance of the network under both attacks. A hybrid attack could also be implemented where a misbehaving node selects a small backoff (i.e., selfish attack) for its own traffic and chooses a larger $sifs^*$ for crosssing traffic and nominal $sifs$ for local traffic (malicious attack). Fig. 4.16 shows the simulation results for a grid network (4x4 grid and the grid unit is 200m, see Fig. 4.15) with 8 CBR flows each of 0.4 Mbps data rate. One misbehaving node is chosen at the center of the grid

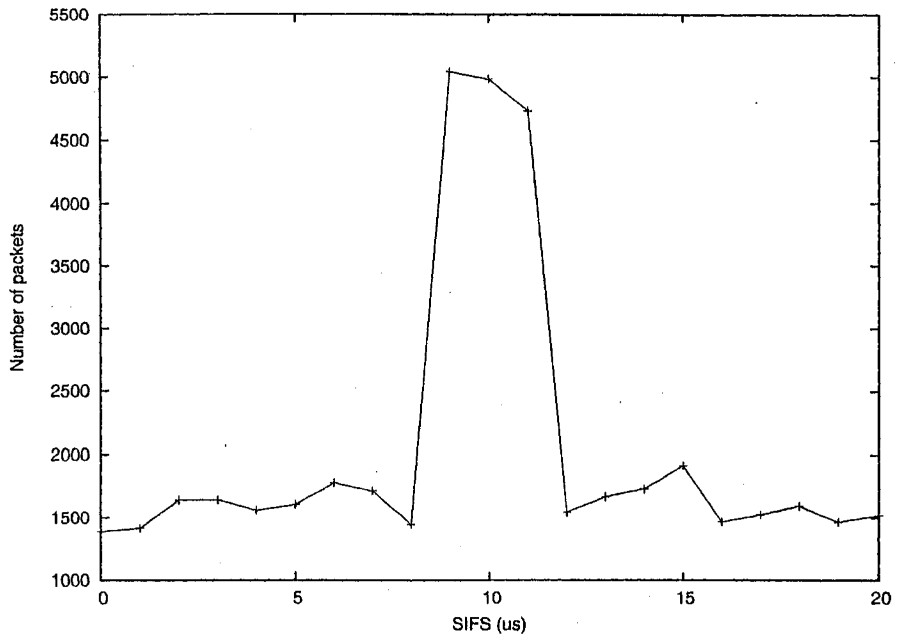


Figure 4.14: Hybrid Behavior: Packets forwarded vs. different sifs values

and the *sifs** value is set to 13 us. For the backoff attack, the misbehaving node always chooses a $CW_{min} = 3$ and $CW_{max} = 127$ for its own traffic. Fig. 4.16 shows the throughput of the flow (f). Clearly, under normal operation (i.e., $sifs* = sifs =$ 10us and $CW_{min} = 31$), a throughput of 125 Kpbs is achieved by flow f. However when the node manipulates its backoff ($CW_{min} = 3$), a throughput of 350 Kbps is achieved. Alternatively, when the node acts maliciously (i.e., only change its *sifs*) then a throughput of almost 200 Kbps can be achieved and finally for the hybrid attack a throughput of 350 Kbps is obtained.

Here, when a node acts selfishly it will continuously transmit data by refusing to backoff and hence the sole objective of the node is to unfairly increase its access to the channel at the expense of well behaved nodes. Other nodes in the vicinity of the selfish node will contend for transmission but continuously backoff and ultimately fail to transmit. On the other hand, in the malicious attack the objective is to force flows away from the misbehaving node by

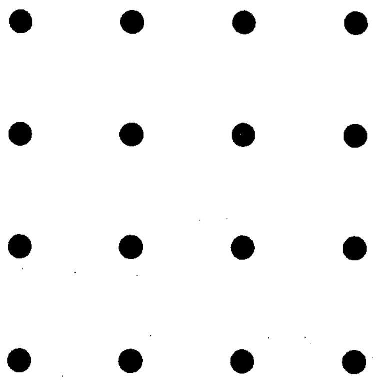


Figure 4.15: 4 × 4 Grid Topology

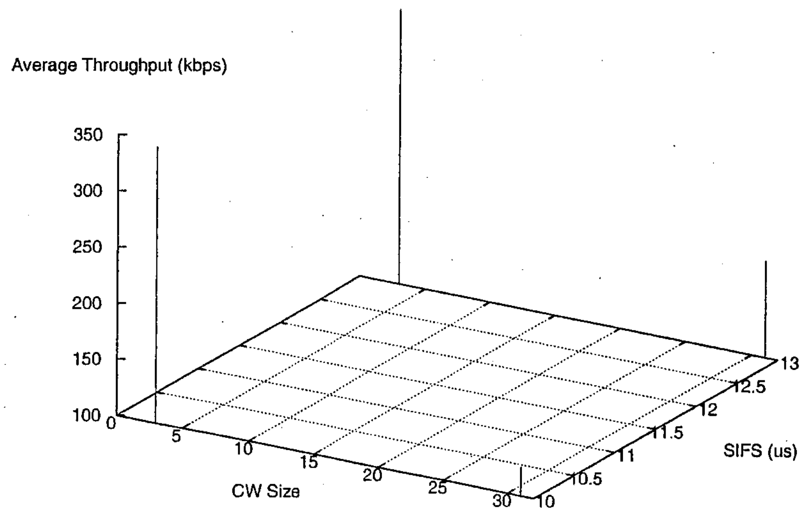


Figure 4.16: Average Throughput under different attacks

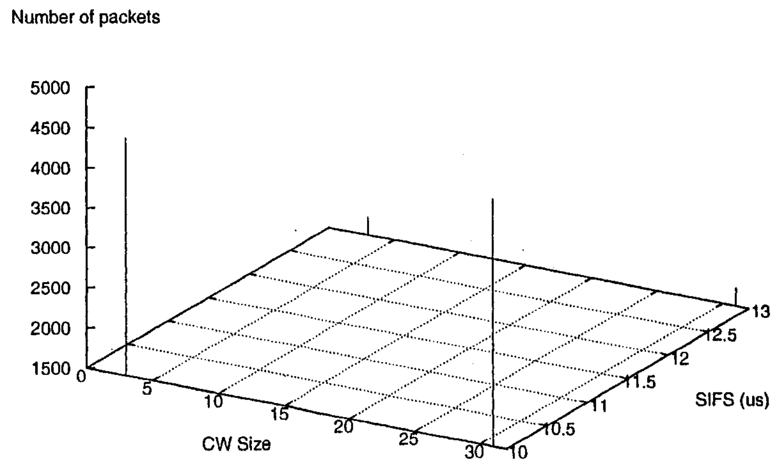


Figure 4.17: Number of packets forwarded by the misbehaving node under different attacks

either disrupting the route discovery or by prohibiting the cross traffic. The motive for the node could be to conserve its own energy (i.e., greedy behavior) by refusing to forward data packets of no direct interest to itself. As a result, flows will be routed or rerouted around the MN. This means less congestion in the medium close to the malicious node and hence less contentions for the channel. Accordingly a notable increase in the throughout for flow f is achieved as our simulation results show. Similar discussion is valid for the hybrid attack as well.

Note that, as mentioned earlier, recent research studies [32, 42, 50] have proposed systems for detecting backoff manipulation attacks and those models can also be applied for detecting backoff cheating. Therefore it is not in the best interest for a node to misbehave by changing its backoff since it could be detected and isolated from the network. However, those same detection systems fail to detect the malicious attack highlighted here. As a result, the node will obtain a larger throughout (as shown in Fig. 4.16) and conserve more of its energy. For this reason, new detection methods are required to limit or mitigate the effects of these misbehaviors. Finally, the number of packets that are forwarded (i.e., cross traffic) by the malicious node is shown in Fig. 4.17. The figure shows that under normal case, the node forwards around 4500 packets (a similar result is also obtained with selfish attack). However, when the node starts misbehaving maliciously (after 50 seconds from the begining of the simulation) all the crossing flows are disrupted and a total of only 1600 packets is forwarded by the MN. Note, these 1600 packets are forwarded before the malicious attack starts, i.e., in the first 50 seconds of the simulation time. The same results are obtained for the hybrid attack. This shows that a malicious node can effectively disrupt the traffic of any flow crossing through the node and hence force the flow to be routed elsewhere over a longer route.

4.5 Conclusion

In this chapter, we evaluated the impact of MAC layer selfish, malicious and hybrid misbehaviors on the network performance. It is evident that these misbehaviors can severely degrade the overall network performance such as reduced packet delivery ratio and increased network latency. However, the current secure routing protocols and MAC detection systems implemented can not effectively defend MANET against these attacks. Therefore, it is required to design new detection and reaction systems. In Chapter 5, we will present our new *DREAM* system to detect and react to the TO misbehavior presented in Chapter 3.

Chapter 5

Detection and Handling MAC Misbehavior

5.1 Introduction

A new class of vulnerabilities, *TO* attack (TimeOut), is highlighted in Chapter 3. Chapter 4 evaluated the devastating impact of this new vulnerability can have on the overall network performance and we have explained that *TO* attack can take effect when either the transmitter or the receiver misbehaves. Therefore, detection schemes should be implemented for a receiver (Case (I), i.e., misbehaved transmitter) or a transmitter (Case (II), i.e., misbehaved receiver) respectively. We only consider four-way handshaking procedure in the following discussions. The same detection schemes can be applied to the two-way handshaking procedure as well. In this chapter, we present a new detection and reaction system *DREAM* (a system for Detection and REAction to timeout mac layer Misbehavior) to mitigate the effects of *TO* attack. In Section 5.2, we describe the detection functions and first reaction for a well-behaved node. We present the procedures of the second reaction in Section 5.3. Simulation results and analysis are shown in Section 5.4.

5.2 Detection and Handling Hybrid MAC Misbehavior

5.2.1 Detection Function for a Well-behaved Receiver

We start first by describing the method for detecting the misbehavior of a misbehaved transmitter. In the rest, Tx and Rx refer respectively to the transmitter and receiver of a packet (which may or may not be the same as source and destination nodes).

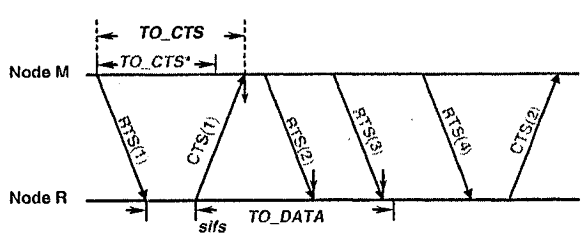
Suspect identification

When a node R (Rx) receives a RTS frame from node M (malicious node), it increments the value of $RTS\_seq(src)$ ($RTS\_seq(src)$ is used by R to record the number of consecutively received RTS frames from M for a single DATA frame). If R receives the first RTS from M, i.e., $RTS\_seq(src) = 1$, it will send back a CTS after waiting for a $sifs$ time interval. Meanwhile, R computes a TO_{DATA}, a timeout interval during which it expects to receive the DATA packet from M (see Fig. 5.1(a)). Note that since M is a misbehaved node, its CTS timeout TO^{*}_{CTS} is computed to a smaller value using $sifs^{*}$ ($sifs^{*}$ is less than the nominal value $sifs$ used by a well-behaved node, e.g., node R). Therefore, node M receives the CTS from R upon the expiration of its timeout timer (see Fig. 5.1(a)); M will accordingly drop the received CTS and send the second RTS after appropriate deferral interval [1]. When node R receives the second RTS for the same DATA frame, R does not know whether its previous CTS was lost because (a) M is a malicious node, or (b) M is the victim of an attack from a third node that intentionally transmits frames at the same time R sends its CTS, or further (c) whether M is under another attack from colluding nodes transmitting data flows in the vicinity of R. All these possibilities indicate that M is not trustworthy for a reliable communication (albeit M may itself be a victim). We distinguish between two cases; (1) node R may receive subsequent RTS during the data timeout, TO_{DATA}, or (2) after the timeout. Node R also maintains a parameter (*badCredit*) to evaluate the trustworthiness of every node that it communicates with (or every node that communicates with R). When the second RTS arrives at node R during TO_{DATA}, it is more likely that node M is misbehaving (since its TO_{CTS} has expired earlier than it should) and hence R will punish node M by increasing its *badCredit* parameter heavily[1] (e.g., by increasing or adding a constant, or even doubling its value). Alternatively, if the second RTS arrives after the TO_{DATA}, then it is more likely that node M is a victim of an attack of type (b) or (c); In this case, node R will only increase the *badCredit* value of M slightly since node M is still not reliable for communication. Once the *badCredit* reaches a certain *creditThreshold*, M becomes a *suspect* and node R will call the adjustment scheme to react for the first time. The value of *creditThreshold* does not need to be high enough; a small value (e.g. 5) can be used to quickly indicate whether a node is deviating from the normal operation (see Fig. 5.2 for the operation of the system).

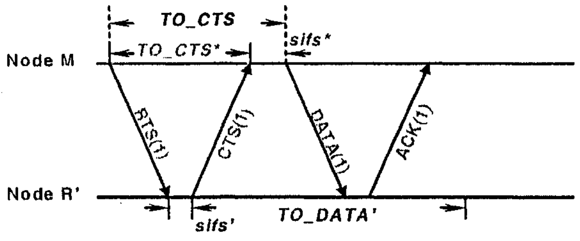
Adjusting timeout

The adjustment scheme is used by the well behaved receiver to ensure correct misbehavior diagnosis of node M. Once node M is designated as a suspect, then node R will react

[1]See Fig. 5.2, (++) designates a heavy increase in the bad Credit, e.g., increase by 2, and (+) designates a light increase of bad credit, e.g. increase by 1.



(a) Attack Case: Misbehaved Tx



(b) Detect and First React: Misbehaved Tx

Figure 5.1: Misbehaved Tx

by expediting the transmission of a CTS frame. That is, node R will adjust its $sifs$ to a smaller value $sifs'$ (e.g., use a $sifs$ of 2us if 10us is used as the nominal value), as shown in Fig. 5.1(b). Note that when node R selects a smaller $sifs'$, TO'_{DATA} increases (see Equation (1)) and hence the chances that the DATA packet arrives at node R within the timeout would be higher. After sending the CTS, R will watch the reaction of M.

$$TO_{DATA} = T_{CTS} + 2\delta - 2 \times sifs + rf + T_{ACK} \tag{1}$$

Note that choosing a $sifs'$ smaller than $sifs^*$ will cause no *TO* attack as we explained in Chapter 3. That is because R will compute a larger TO'_{DATA}, i.e., $(TO'_{DATA} - TO^*_{DATA}) = 2(sifs^* - sifs')$ which means that the data timeout interval is extended and therefore chosing a $sifs'$ will yield no *TO* attack as shown in Fig. 5.1(b).

Handling misbehavior

Upon transmitting a CTS frame with an adjusted smaller $sifs$ value, if node R receives the DATA packet from node M, then this indicates that (1) either node M is no longer under attack, or (2) node M is not aware of the detection system implemented by R, or (3) M is

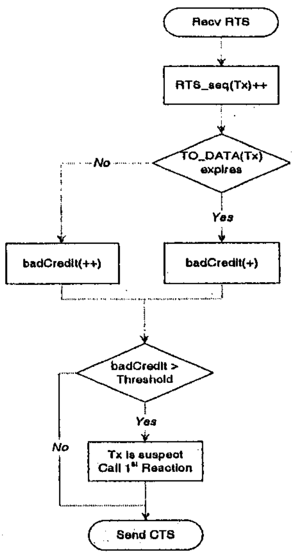
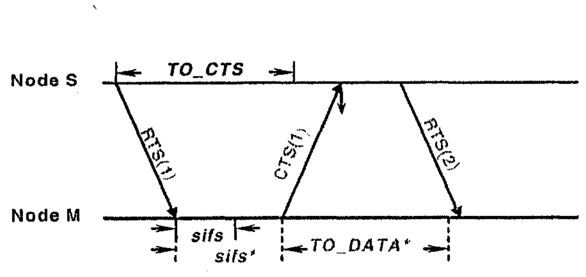
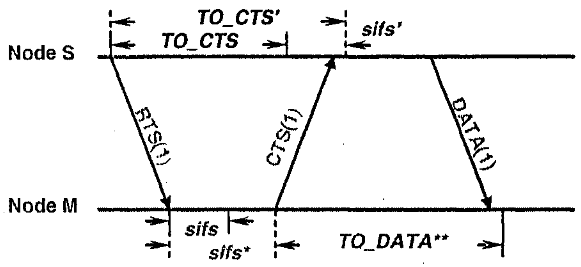


Figure 5.2: Detection Function for a Well-behaved Receiver

aware but is avoiding degrading further its trust level. In the second case, when M receives a CTS from R during the timeout period, according to the specifications, it must transmit the DATA packet. In the third case, the node cooperates for the successful operation of the protocol in order to avoid being isolated from the network if its trust level falls below a threshold. In all cases, the communication will successfully continue through node M. Alternatively, if node R does not receive the DATA packet after reacting to the suspect node, this indicates that node M may have detected the reaction of R and have adjusted further its $sifs^*$ value or may be intentionally dropping the received CTS or may still be under attack. In all cases, node M is not reliable for any future communication. At this point, the trust level of node M is reduced by node R. This monitoring and reacting process continues for a predetermined *monitoring period* until the trust level of node M falls below a trust level threshold and node R invokes its second reaction scheme as explained later.



(a) Attack Case: Misbehaved Rx



(b) Detect and First React: Misbehaved Rx

Figure 5.3: Misbehaved Rx

5.2.2 Detection Function for a Well-behaved Transmitter

Suspect identification

When node S has a data packet to transmit, it will send a RTS frame to node M (Fig. 5.3(a)). Accordingly, S increases the $RTS\_seq(dst)$ counter that it maintains to record the number of consecutively transmitted RTS frame to the destination for the same DATA frame. S then computes TO_{CTS}, the maximum timeout interval during which S expects to receive back the CTS frame from M. If M is well-behaved, it will send back a CTS if the medium is free upon the reception of a RTS frame. If M is misbehaving, it will delay the transmission of CTS by increasing its own SIFS value to $sifs^*$ ($sifs^*$ is larger than the nominal value $sifs$ used by a well-behaved node, e.g., node S). Therefore, on the transmitter side, node S will wait for the CTS frame until the expiration of TO_{CTS} timer. After the timer expires, S will defer and schedule a retransmission for RTS.

There are two different scenarios that can be distinguished: 1) the CTS frame from M arrives during the deferring period; 2) the CTS frame does not arrive even when the deferring period has finished. For the former scenario, it means that M has delayed the

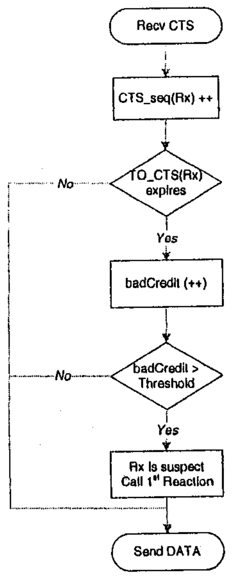


Figure 5.4: Detection Function for a Well-behaved Transmitter

transmission of the CTS frame by slightly increasing its $sifs^*$, and hence causing the CTS to be dropped by the sender S because TO_{CTS} has expired before the arrival of the CTS frame. In this case, S will increase the counter $CTS\_seq(src)$ to indicate the number of delayed CTS frames arriving from M. Moreover, node S will punish M by increasing its *badCredit* (e.g. by 2) since this is an obvious abnormal transmission. Alternatively, if the CTS frame does not arrive during the deferring period, then either node M has selected a larger $sifs^*$ or node M has its NAV indicating a busy medium. In both cases, S will slightly increase the *badCredit* for the receiver M. Once the *badCredit* is above the chosen *creditThreshold*, M becomes a suspect host and the first reaction scheme will be called by S. Again notice that the *creditThreshold* does not necessarily have to be high enough to ensure correct diagnosis. It is mainly used to invoke the adjustment scheme (see Fig. 5.4 for the operation of the system).

Adjusting timeout

When node M is identified as a suspect, node S will trigger its first reaction by increasing its TO'_{CTS}, e.g., by incrementing the value of $sifs$ to $sifs'$ ($sifs'$ is larger than the nominal value of $sifs$), as shown in Fig. 5.3(b). This is to circumvent the misbehavior of node M by increasing the CTS time out.

Handling misbehavior

Upon the transmission of an adjusted MAC frame (e.g., RTS frame), if S receives the CTS from M within the newly adjusted TO'_{CTS}, then this indicates that (1) either node M is no longer misbehaving (i.e., CTS arrives within TO'_{CTS}), or (2) it is not aware of the reaction scheme of S, or (3) it silently follows the adjustment to avoid degradation of its trust level. As a result, S will send to node M the DATA frame. On the other hand, if the CTS is still delayed, it indicates that the value chosen for $sifs'$ by node S is not larger than $sifs^*$ and hence further adjustment for $sifs'$ may follow. Note that, node S cannot keep on increasing its $sifs'$ value and an upper limit is defined, $sifs_{max}$. Every time S reacts by increasing $sifs'$, it updates the *badCredit* parameter for node M. In addition, instead of changing $sifs$, node S can simply adjust its TO'_{CTS} until TO'_{CTS} hits the defined upper limit.

Furthermore, note that choosing a larger $sifs'$ value by node S ($sifs' > sifs^*$) will not be considered as an attack. That is, a larger $sifs'$ yields at node M a larger $TO'_{CTS} > TO_{CTS}$ and $rf' > rf$. At the receiver side, node M will compute TO^{**}_{DATA} where $TO^{**}_{DATA} - TO^{*}_{DATA} = 3(sifs' - sifs) > 0$. This means the data timeout interval is extended at the receiver M as shown in Fig. 5.3(b) and therefore, no *TO* attack.

After M receives the DATA packet, it will either send back an ACK or continue to drop the DATA packet. For the former case, the communication will work at the cost of increased packet delay and S will give M a slightly decreased trust level. For the latter case, M will end up with a heavily decreased trust level and get punished. As mentioned before, we use the trust level as a long term monitor parameter (i.e., during a monitoring period interval) to watch the behavior of a node and invoke the second reaction schemes.

5.3 Procedures of the Second Reaction

In the designed system, there are two separate reaction stages as shown in Fig. 5.5. As mentioned in Section 5.2, a node will be marked as a suspect when its *badCredit* is above a predetermined *creditThreshold* during a short term monitoring period (i.e., per-frame monitoring). Once the node becomes a suspect, the first reaction scheme will be invoked to mitigate the negative impact caused by the potential misbehaved node. Therefore, the first reaction quickly responds to the misbehavior and triggers the procedure for the second reaction.

Here, note that once a node is identified as a suspect node (SN), the system needs to further determine whether the SN is an untrusted node (UN) or a trusted node (TN). For this reason, a long term monitoring is triggered during which the trust level of every SN will be evaluated based on its cooperation for successful data transmission. Every time the SN misbehaves, its *trustLevel* is reduced and a *trustThreshold* is defined below which the SN is considered as a UN. Based on the different value of *trustLevel*, a well-behaved node (WN) can invoke corresponding punishment methods.

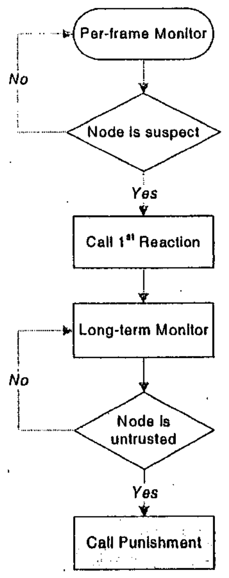


Figure 5.5: Procedure of Two-stage Reaction

5.3.1 Monitoring Scheme

The authors of [32] and [42] have proposed similar methods for monitoring the misbehavior of nodes manipulating the backoff selection. In [32], a receiver maintains a moving window containing information of the last W packets received from each sender. When a new packet is received, the difference $B_{exp} - B_{act}$ is stored in the moving window, where B_{exp} is the expected backoff value calculated by the receiver and B_{act} is the actual backoff of the sender. A positive (negative) difference indicates that the sender waits for less (more) than the backoff duration expected by the receiver. If the sum of these differences in the previous W packets from the sender is greater than a threshold $Thresh$, then the sender is designated as "misbehaving".

On the other hand, in [42], the DOMINO system also computes the difference of $B_{exp} - B_{act}$ for each received data packet. However, unlike [32], the backoff difference for each packet are collected during configurable intervals of time T. At the end of each interval (e.g., every 10 seconds), the detection mechanism is run. An alarm will be generated if the sum of these difference exceeds a $Thresh$ and accordingly a node will be marked as "misbehaving".

The systems presented in [32] and [42] monitor node deviation for each data packet. In order to avoid overloading the monitoring node with per-data-packet computation, both schemes make the decision whether a node misbehaves or not either after W successful data packet transmissions or at the end of every time interval T. In this way, the monitoring node can save extra computation power and device energy compared with alarm per-data-packet. In addition, period-monitoring (either based on number of packets or time interval) can improve the correct diagnosis ratio compared with per-data-packet because more statistical data can be collected. However, the period-monitoring schemes implemented by [32] and [42] could result in failed detection and increased delay of detection. For example, assume a node implements a fixed monitoring interval, e.g., $interval_0, ...interval_i$ (either W data packets or T seconds time interval) and $interval_0 = interval_1 = ... = interval_i$; the alarm will be invoked only at the end of each interval. Further, assume a node will be detected as a "misbehaving" node if it misbehaves for 70 data packets transmission during a monitoring interval. First, the $interval_i$ has to be chosen appropriately to collect data from at least 70 data transmission; a smaller interval will not allow the node to monitor and detect at least 70 data frame transmissions. A larger interval will result in delaying the detection of a misbehaved node. Second, a smart attacker can guess the approximate length of $interval_i$ (not necessary the exact value) and try to misbehave for a number less than the value that triggers the alarm (e.g., misbehaving for 40 data packets). Or the attacker can adaptively choose a series of backoff values, e.g., $0, CW_{max}, 0, CW_{max}, 0...$, which can make the sum of total difference of $B_{exp} - B_{act}$ below the threshold and guarantee the attacker to share a large portion of the bandwidth especially in the presence of multiple contenders. Moreover, there is another disadvantage for the monitoring scheme of [42]. The system uses time interval based monitoring, $t_0, t_1, ...t_i$ and $t_0 = t_1 = ... = t_i$. The alarm will only be issued at the end of each time interval $t_0, t_1, ...t_i$. However, during each time interval there could be different number of transmissions (e.g., 100 data transmissions in t_0 and 10 data transmissions in t_1[2]). As the alarm decision is made based on the average backoff deviation from the expected backoff value during the same interval [42], this could lead to the misdetection of a WN. For example, a WN happens to choose 10 very "small" backoff, i.e.,$CW_1, CW_2, ..., CW_{10}$, will be identifed as "misbehaving" during t_0 if $\frac{\sum_{i=0}^{10}(B^i_{act} - B^i_{exp})}{10} > Thresh$. However, it will not be identified as a MN if it chooses the same 10 small backoff during t_1 because the difference of B_{act} and B_{exp} is averaged over 100 data transmissions, i.e. $\frac{\sum_{i=0}^{100}(B^i_{act} - B^i_{exp})}{100} < Thresh$.

One obvious advantage of per-frame based monitoring is to give quick response to the misbehavior. However, as mentioned earlier, this monitoring scheme might cause high overload of system especially in a congested environment. Our first reaction scheme is based on per-MAC-frame monitoring to try to mitigate the potential misbehavior in a prompt manner. For the second reaction scheme, instead of monitoring node behavior for a fixed time interval, we monitor the node for a certain number of frame transmissions (monitoring window size). In order to reduce the chance of an intelligent node evading from the detection system (as mentioned above), the trust level of a node is accumulated

[2]This can be caused by the node mobility, newly joint or left nodes that lead to the establishment of new data flows or temporarily route breakage of existing flows.

through multiple monitoring sessions and reset after the successive monitoring sessions have elapsed. We describe how this scheme works in details in the following section.

5.3.2 Trust List

Each node maintains a trust level list for all the nodes it communicates (either acts as a Tx or a Rx); the default *trustLevel* for each node is set to T_0. A node will monitor (either as a Tx or Rx) each MAC frame transmission between itself and a SN and update the *trustLevel* of a SN only at the end of each *MW* (monitor window size). The *trustLevel* of a SN will be decreased by m for each failed frame transmission and increased by αm $(0 < \alpha \leq 1)$[3] for each successful frame transmission. Consequently, the trust value $T[SN_i]$ for node i at the end of a monitoring window is updated by $T[SN_i]^n = T[SN_i]^{n-1} - Num_{fail} \times m + Num_{succ} \times \alpha m$, where $T[SN_i]^0 = T_0$. $T[SN]^0$ is the initial trust value for every SN. $T[SN_i]^n$ is the *trustLevel* of SN i at the end of the n^{th} MW. Num_{fail} is the number of failed frame transmission and Num_{succ} is the number of successful frame transmission counted during each MW.

If $T[SN_i]$ is less than the pre-defined *trustThreshold* during a *MW* (e.g., $T[SN_i] < trustThreshold$), the SN will be maked as an UN. Otherwise $T[SN_i]$ will be accumulated to the next monitor window. Finally $T[SN_i]$ will be reset to the initial value T_0 after multiple *MW* sessions (e.g., N times MW[4]) if node i is trusted, i.e. $T[SN_i] \geq trustThreshold$.

A UN could be: (1) a MN that continuously implements the *TO* attack regardless of

[3]In order to prevent a misbehaved node attempts to fool the reaction system, e.g., when $\alpha = 1$ MN can misbehave and well-behave for the same number of transmissions thus make no change to the *trustLevel*, we always choose α is much less than 1, e.g., 0.2.

[4]N can be randomly generated to reduce the possibility to be guessed by a MN.

the first reaction scheme; or (2) a WN that is continuously under attack. Under both cases, this node is not reliable for future data communication. Thus the system will call the punishment scheme to handle this UN. For case (2), as long as the WN is no longer under attack, it will gradually increase its trust level until it becomes trusted. Finally note that a MN that obeys the first reaction is considered as a TN as the impact of its misbehavior can be mitigated by the first reaction scheme.

5.3.3 Punishment

Upon the identification of a UN, the system will call the punishment scheme. The *trustLevel* information can be propagated to the upper layer, e.g. network layer. In this way, a node can rate its neighbor nodes with different trust level. The routing protocol can use this information as one of the criterion to choose an optimal route, e.g., the node with higher trust level has more priority to be selected for the next hop. A node will also refuse to forward any traffic from a UN until it becomes a TN again. To this point, although a MN seems to realize its goal to conserve energy by not being chosen as a forwarding node, it also has the difficulty to send out any traffic and use the service of the network.

Moreover, if we have such a system installed on all the neighbor nodes of a UN, as long as more than half of the neighboring nodes are well-behaved, the UN gets isolated.

However, this scheme will face difficulty to handle with smarter *TO* attack. The MN can selectively change the timeout scheme based on different source nodes. As a result, some sources are capable of sending traffic while some not. Hence, this MN can reduce its consumption of resources while limiting the chance to be eliminated.

The most difficult issue is how to react if we detect a misbehaved node. In ad hoc networks, even if a node can judge a node is misbehaving with 100% confidence, it is still difficult to distribute this information throughout the network and convince other nodes this information is not flawed. In addition, it is a difficult task to perfectly handle with the misbehavior in networks which lack of centralized management.

Further we believe a layered security cooperation mechanism may be a better solution than a pure detection and reaction implemented at a single layer.

5.4 Simulation and Analysis

In order to evaluate the performance of our proposed approach, we use ns2 [17] to simulate our detection and reaction system. As the detectionfunctionn for a well-behaved Tx is similar to the detectionfunctionn for a well-behaved Rx, in this section we only focus on examining in details of the detectionfunctionn for a well-behaved Rx.

5.4.1 Simulation Setup

Simulation Topology: The topology of this experiment is a grid network of 7×7 nodes. The grid unit is 100m. There are 49 nodes that are positioned on the grid. All the nodes are fixed. The transmission range for each node is 250m and the carrier sense range is 550m. There are 8 flows across the grid topology. The traffic type from the source to the destination is CBR. The packet size is 512 bytes/packet and the data rate is 4 packets/second. Note that the traffic load is relatively low[5]. The channel bit rate is 2Mbps. Ad-hoc On-demand

[5]In [22], we have shown that the backoff attack only has negative effects in a congested environment and trivial impact on the whole network performance. Here we use low traffic load scenario to evaluate the

Distance Vector protocol (AODV) is used as the routing protocol. The total simulation time is 100 seconds for each single run. Each data point is averaged over 10 runs. Each run is seeded with a different seed and the set of seeds used for different data points is the same.

Simulation Metrics: We use the following metrics to study the performance of our proposed approach:

- *Correct Detection*: ratio of the number of misbehaved nodes that are correctly marked by the detection system as suspects to the total number of active misbehaved nodes in the network;

- *Misdetection*: ratio of the number of well-behaved nodes that are incorrectly diagnosed as suspects to the total number of well-behaved nodes in the network;

- *Packet Delivery Ratio*: ratio of the data packets successfully delivered to the destination to those generated by the source;

- *Average Packet Delay*: average end-to-end delay for each successfully delivered data packet, which includes all the possible delays caused by route buffering, MAC interface queue, retransmission delays.

5.4.2 Results

In this section, we evaluate the performance of our detection and reaction system DREAM. To model a misbehaved Tx, we define two attacks:

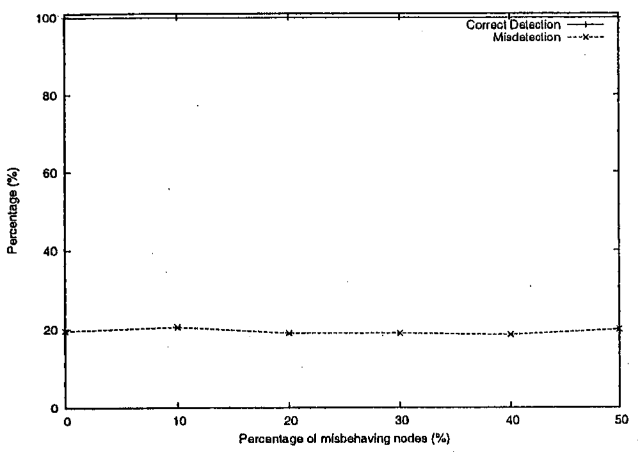
- *Attack 1:* the misbehaved Tx will follow the first reaction system after noticing its exposure to the system;

impacts of *TO* misbehavior.

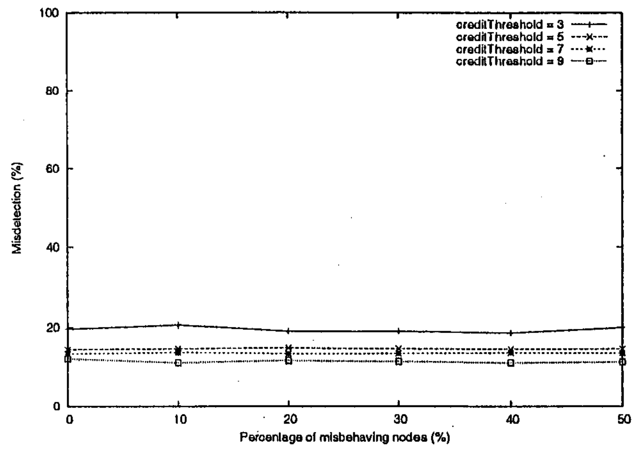
- *Attack 2:* the misbehaved Tx will not follow the first reaction system and keep on misbehaving by adaptively changing its SIFS.

The detection parameter *creditThreshold* is used to identify a suspect. In order to allow fast identification of a misbehaved node, we choose a relatively smaller value 3. In the presence of a suspect node, a well-behaved node will invoke the first reaction by reducing its timeout interval, (e.g. adjusting its $sifs$ to $sifs'$). This adjustment can take several stages until finally the $sifs'$ of this node reaches to a lower bound. For simplicity, upon identification of a suspect, a well-behaved node will immediately adjust its $sifs'$ to 2us for the data exchange with the suspect node. The trust level $T[SN_i]$ for each suspect is initialized to 0 and the *trustThreshold* is also defined as 0. The monitor window (*MW*) is set to 5 (i.e., every 5 consecutive transmissions the trust list is updated) and after every 5 monitor windows the trust level is reset to 0 for a well-behaved node.

Diagnosis Accuracy: Fig.5.6(a) shows the correct detection and misdetection percentage under different percentage of misbehaving nodes (MN%). As the figure shows, the correct detection ratio is 100% which means our approach is successful in recognizing all the misbehaved nodes as suspect nodes. On the other hand, we observe a relatively high misdetection ratio, i.e., 20%. Recall that a well-behaved node can be misdiagnosed due to successive frame retransmissions caused by collisions. Moreover, the misdetection is directly related to the value of *creditThreshold*. In this test, we use a smaller threshold, e.g., 3, which allows reasonably fast misbehavior detection, i.e., quick responsive time, but however at the cost of a higher misdetection. The increment of the threshold value will reduce the misdetection ratio. In Fig. 5.6(b), we plot the misdetection ratio for different



(a) Accuracy of Detection

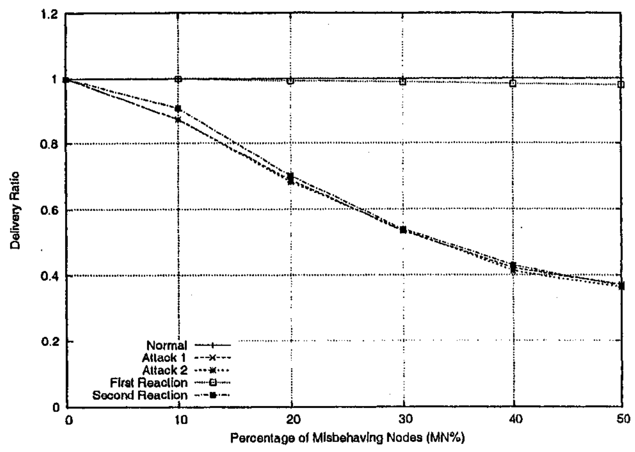


(b) creditThreshold Effect on the misdiagnosis

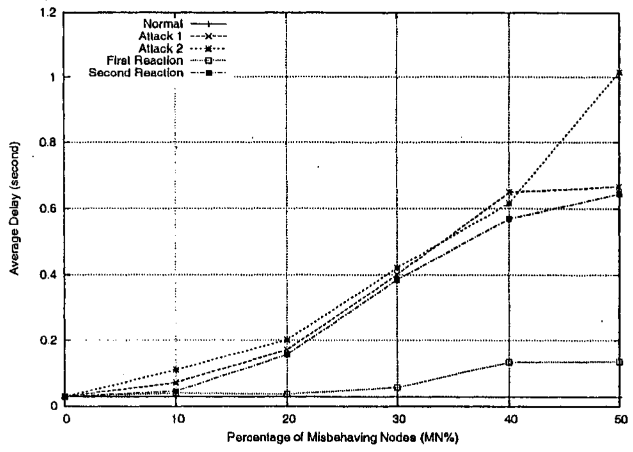
Figure 5.6: Efficiency of DREAM: detection

creditThreshold. Here we can find that the misdetection is decreased with the increase of the threshold value but at the cost of slow reaction towards misbehaved nodes. For example, the misdetection ratio is around 10% when *creditThreshold* is 9. Here, notice that a well-behaved node which is misdiagnosed as a suspect node will not be affected by the reaction mechanisms: for the first reaction scheme, a well-behaved node will continue its normal communication; additionaly, for the second reaction scheme, a well-behaved node will never get a low trust level as will be shown later.

Packet Delivery Ratio: Fig. 5.7(a) compares the delivery ratio versus MN% obtained under five cases: 1) no active misbehaved nodes (designated as *Normal*); 2) misbehaved nodes implement *Attack 1* while there are no DREAM systems installed on the well-behaved nodes (designated as *Attack 1*); 3) misbehaved nodes implement *Attack 2* and only the first reaction scheme is being used (designated as *Attack 2*); 4) misbehaved nodes use *Attack 1* while well-behaved nodes are using the first reaction scheme against Attack 1 (designated as *First Reaction*); 5) misbehaved nodes use *Attack 2* and the second reaction scheme is being used to evaluate the trust level of a suspect and punish the UNs (designated as *Second Reaction*). The misbehaved nodes are randomly selected. As seen from the figure, in the presence of misbehaved nodes the delivery ratio is decreased sharply as MN% increases (e.g., 75% loss when MN% is 50 compared with the normal case). This is due to the fact that routes will be broken under the attack and some flows may even not be able to establish any new available routes [19]. Alternatively, the delivery ratio of the network when all nodes follow the first reaction scheme is almost not affected, except that there is a slight decrement compared with the normal case when MN% is close to 50%. Hence, the proposed first reaction scheme is fairly successful in ensuring normal data communication in a malicious environment. For the Attack 2 case, the second reaction succeeds in identifying the MNs and invoke the punishment method. Here, we only consider isolation of these nodes (widely used in current detection and reaction system) as a punishment method. Note that, cross-layer interaction is necessary to handle the misbehavior impact on the routing layer. Clearly, the figure shows that when the MN% is small (e.g., less than 20%) the network delivery ratio increases after the second reaction because some of the packets will be able to find alternate routes to their destination. However, as the MN% increases further, the packet delivery ratio of the network after the second reaction is very close to that of a network without any reaction. That is due to the fact that some of the UNs will be eliminated from the network and the routing algorithm cannot establish any valid routes to send the packets to their destinations. This will lead to a severe performance degradation of the whole network. This same result is obtained by measuring the instantaneous throughput as shown in Fig. 5.8. All these results have shown that isolation of misbehaved node may not



(a) Packet Delivery Ratio



(b) Average Packet Delay

Figure 5.7: Efficiency of DREAM: reaction

be an optimal solution especially when the number of misbehaved nodes in the network is not small.

Average Packet Delay: Fig. 5.7(b) shows the average packet delay for the system with and without the detection and reaction scheme. It is clear that in both cases, the average delay will increase. For the case where nodes are implementing the DREAM system, there is an increased delay which is due to the fact that a well-behaved node needs to monitor the node behavior for a very short period to make its judgment. As long as the suspect is identified and handled with the first reaction scheme, its negative impact will be mitigated. As a result, the average delay is less than the attack case and comparable with normal case especially when MN% is low.

Evolution of Trust Level: Fig. 5.9 shows the evolution of the trust level for TNs and UNs throughout the simulation. We can see that, the trust level of a TN is above the threshold and is reset after $N \times MW$, i.e., 25. Here we give different weight for the successful

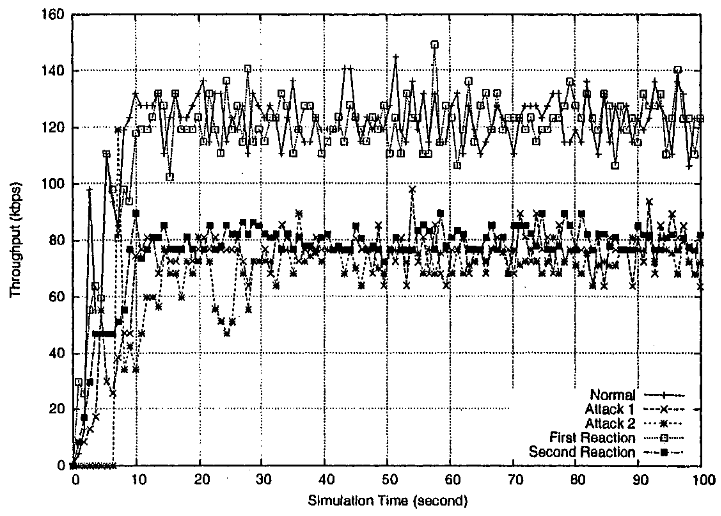
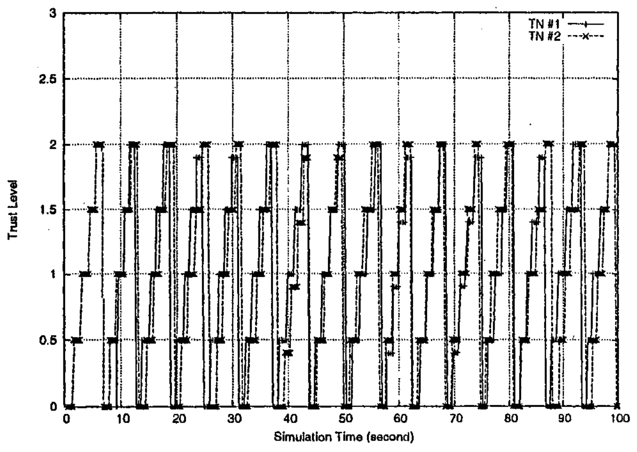
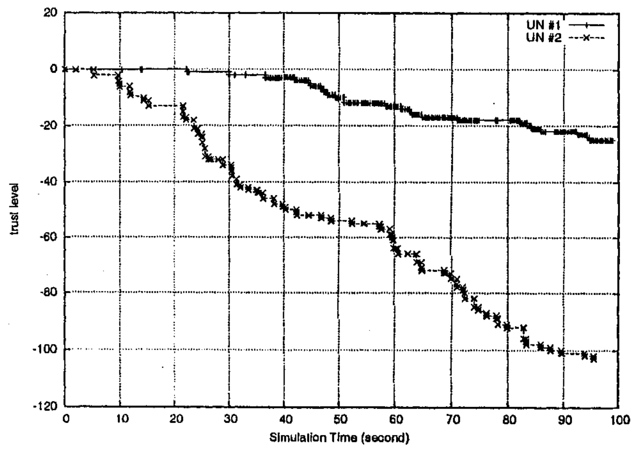


Figure 5.8: Instant Throughput



(a) Trusted Node



(b) Untrusted Node

Figure 5.9: Evolution of trust level

transmission and the failed transmission, i.e., $m = 1$ and $\alpha = 0.1$. A well-behaved node or a misbehaved node that obeys the first reaction will never get a low trust level, see Fig. 5.9(a). Furthermore, a MN will have to successfully cooperate in forwarding data for some time in order to restore its trust level. That is, once it becomes a suspect it is very difficult to gain the trust of other nodes unless it well-behaves for an extended period of time. Clearly, a UN that always misbehaves will get its trust level decreased as shown in Fig. 5.9(b).

5.5 Conclusion

In this chapter, we have presented a new detection and reaction system towards MAC misbehavior TO attack. Rather than just correctly identifying the misbehaved nodes, we have developed a two-stage reaction (first reaction stage is to mitigate and second reaction stage is to punish) mechanism that can improve the network performance in the presence of misbehaved nodes. Through simulations, we have shown that our system achieves high accuracy in identifying misbehaved nodes. Moreover, we have also shown that the first reaction system is very effective in mitigating the misbehavior effect and improve the network performance (e.g., throughput and delays).

Chapter 6

Summary and Future Work

In this thesis, we quantified the impact of MAC misbehavior on network layer performance in mobile ad hoc networks and we studied the vulnerabilities of two on-demand routing protocols using three types of MAC misbehavior: (1) selfish misbehavior; (2) malicious misbehavior; (3) hybrid misbehavior. We showed that these attacks could propagate to the upper network layer and disrupt the routing mechanism, therefore causing devastating effects on the overall network performance. Overall, our results have showed that AODV presents higher resiliency to the second DoS attack when the intensity of malicious nodes in the network is high; whereas DSR slightly outperforms AODV when the fraction of malicious nodes is small. We also showed that selfish nodes implementing the backoff attack can cause severe service disruptions for multi-hop flows in their vicinity, forcing routes for normal flows to fail due to unsuccessful attempts to capture the channel. We showed that such nodes will then be forced to forward a large amount of packets (in addition to its own) for the failed flows upon rerouting and ultimately facing the risk of draining their own energy.

Moreover, current work has mainly concentrated on handling MAC selfish misbehaviors and detection systems have been proposed. Detection and reaction against malicious MAC misbehaviors, however, is still relatively unexplored. In this thesis, we also have presented a new type of malicious behavior (*TO* attack) and provided the corresponding detection and reaction schemes DR^2TO. Rather than just correctly identifying the misbehaved nodes, we have developed a two-stage reaction (first reaction stage is to mitigate and second reaction stage is to punish) mechanism that can improve the network performance in the presence of misbehaved nodes. Through simulations, we have shown that our system achieves high accuracy in identifying misbehaved nodes. Moreover, we have also shown that the first reaction system is very effective in mitigating the misbehavior effect and improve the network performance (e.g., throughput and delays).

In the future, we intend to evaluate the power cost the misbehaving nodes may incur as a result of misbehaving. That is, although clearly we have shown that a misbehaved node can forward more packets of no direct interest to the node, we have not evaluated the power penalty the node must endure.

Moreover, we will also thoroughly evaluate the efficiency of the detection and reaction system we proposed in Chapter 5. We have shown that the first reaction can mitigate the network performance degradation caused by misbehaved nodes and in the future we intend to evaluate the efficiency of the second reaction mechanism and measure the associated false positives and false negatives.

Bibliography

[1] IEEE802.11 wireless lan media access control (MAC) and physical layer (PHY) specifications. 1999.

[2] Rice University. Monarch Project: Mobile networking architectures. Technical report, http://www.monarch.cs.rice.edu/, 1999.

[3] I. Aad, J. P. Hubaux, and E. W. Knightly. Denial of service resilience in ad hoc networks. In *Proc. of ACM MobiCom*, September 2004.

[4] G. Athanasiou, L. Tassiulas, and G. S. Yovanof. Overcoming misbehavior in mobile ad hoc networks: an overview. *ACM Crossroads*, July 2005.

[5] C. N.-R. Baruch Awerbuch, David Holmer and H. Rubens. An on-demand secure routing protocol resilient to byzantine failures. In *ACM Workshop on Wireless Security (WiSe)*, Atlanta, Georgia, September 2002.

[6] J. Bellardo and S. Savage. 802.11 denial-of-service attacks: Real vulnerabilities and practical solutions. In *USENIX*, 2003.

[7] V. Bhargavan, A. Demers, S. Shenker, and L. Zhang. Macaw: A media access protocol for wireless lans. pages 212–2225, 1994.

[8] G. Bianchi, L. Fratta, and M. Olivieri. Performance evaluation and enhancement of the CSMA/CA MAC protocol for 802.11 wireless LANs. In *Proc. PIMRC*, volume 2, pages 392–396, Oct. 1996.

[9] A. Bittau. Wifi exposed. *ACM Crossroads*, August 2005.

[10] J. Broch, D. B. Johnson, D. A. Maltz, Y.-C. Hu, and G. Jetcheva. The dynamic source routing protocol for mobile ad hoc networks. *Internet-Draft, draft-ietf-manet-dsr-05.txt*, 2001.

[11] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Mobile Computing and Networking*, pages 85–97, 1998.

[12] S. Buchegger and J.-Y. L. Boudec. Performance Analysis of the CONFIDANT Protocol (Cooperation Of Nodes: Fairness In Dynamic Ad-hoc NeTworks). In *The ACM Symposium on Mobile Adhoc Networking and Computing (MOBIHOC 2002)*, Lausanne, Switzerland, June 9–11 2002.

[13] A. Cardenas, S. Radosavac, and J. S. Baras. Detection and prevention of MAC layer misbehavior for ad hoc networks. In *Proc. of SASN*, October 2004.

[14] C.-C. Chiang, H.-K. Wu, W. Liu, and M. Gerla. Routing in clustered multihop, mobile wireless networks with fading channel. In *IEEE SICON'97, Singapore*, pages 197–211, April 1997.

[15] T. Clausen, P. J. (editors), C. Adjih, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L.Viennot. Optimized link state routing protocol (olsr). RFC 3626, October 2003. Network Working Group.

[16] J. Deng and Z. Haas. Dual busy tone multiple access (dbtma): A new medium access control for packet radio networks. In *Proc. IEEE ICUPC*, 1998.

[17] K. Fall and K. Varadhan. NS notes and documentation. technical report, uc berkley, lbl, usc/isi. In *Xerox PARC*, 2002.

[18] S. Fluhrer, I. Mantin, and A. Shamir. Weaknesses in the key scheduling algorithm of rc4. In *Eighth Annual Workshop on Selected Areas in Cryptography*, Aug. 2001.

[19] L. Guang and C. Assi. On the resiliency of ad hoc networks to MAC layer misbehavior. In *Workshop on PE-WASUN, ACM MsWiM*, October 2005.

[20] L. Guang and C. Assi. A self-adaptive detectin system for MAC misbehavior in ad hoc networks. *Submitted to IEEE ICC06*, September 2005.

[21] L. Guang and C. Assi. Vulnerabilities assessment of ad hoc networks to MAC layer misbehavior. *Submitted to Wiley Wireless Communication and Mobile Computing*, August 2005.

[22] L. Guang and C. Assi. Vulnerabilities of ad hoc network routing protocols to MAC misbehavior. In *IEEE/ACM WiMob*, August 2005.

[23] V. Gupta, S. Krishnamurthy, and M. Faloutsous. Denial of service attacks at the mac layer in wireless ad hoc networks. In *Proc. of MILCOM*, 2002.

[24] Z. Haas and M. Pearlman. The zone routing protocol (ZRP) for ad hoc networks. IETF internet draft, 1997.

[25] Y.-C. Hu, D. B. Johnson, and A. Perrig. Sead: Secure efficient distance vector routing for mobile wireless ad hoc networks. *Ad Hoc Networks*, 1(1):175–192, 2003.

[26] Y.-C. Hu and A. Perrig. A survey of secure wireless ad hoc routing. *IEEE Security & Privacy, special issue on Making Wireless Work*, May/June 2004.

[27] Y.-C. Hu, A. Perrig, and D. Johnson. Packet leashes: A defense against wormhole attacks in wireless ad hocnetworks. In *Proc. of INFOCOM*, 2003.

[28] Y.-C. Hu, A. Perrig, and D. Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. In *Proc. ACM WiSe*, 2003.

[29] Y.-C. Hu, A. Perrig, and D. B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *Proc. Mobicom 2002*, Sept. 2002.

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

[31] P. Karn. Maca - a new channel access method for packet radio. In *Proc. ARRL/CRRL Amateür Radio 9th Computer Networking Conference*, pages 134–140, 1990.

[32] P. Kyasanur and N. Vaidya. Selfish mac layer misbehavior in wireless networks. *IEEE Transactions on Mobile Computing.*, April 2004.

[33] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Mobile Computing and Networking*, pages 255–265, 2000.

[34] S. Murthy and J.J.Garcia-Luna-Aceves. A routing protocol for wireless networks. *Mobile Networks and Nomadic Applications (NOMAD)*, October 1996. Special Issue on Routing in Mobile Communication Network.

[35] R. G. Ogier, F. L. Templin, B. Bellur, and M. G. Lewis. Topology broadcast based on reverse-path forwarding, 2002. IETF Internet Draft (work in progress).

[36] P. Papadimitratos and Z. Haas. Secure link state routing for mobile ad hoc networks. In *Proc. IEEE Workshop on Security and Assurance in Ad Hoc Networks*, 2003.

[37] V. Park and M. S. Corson. Temporally-Ordered Routing Algorithm (TORA). Internet Draft (draft-ietf-tora-spec-04.txt), July 2001. Work in Progress.

[38] G. Pei, M. Gerla, and T.-W. Chen. Fisheye state routing in mobile ad hoc networks. In *Proc. of ICDCS Workshop on Wireless Networks and Mobile Computing*, April 2000.

[39] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.

[40] C. Perkins and E. Royer. Ad hoc on demand distance vector routing. In *2nd International Workshop on Mobile Computing Systems and Applications*, February 1999.

[41] C. E. Perkins, E. M. Belding-Royer, and I. D. Chakeres. Ad hoc On-Demand Distance Vector (AODV) Routing. *IETF Internet Draft, draft-perkins-manet-aodvbis-01.txt*, January 2004. (Work in Progress).

[42] M. Raya, J. P. Hubaux, and I. Aad. DOMINO: A system to detect greedy behavior in ieee 802.11 hotspots. In *Proc. of ACM MobiSys*, June 2004.

[43] R. L. Rivest. *The RC4 Encryption Algorithm*. RSA Data Security, Inc., Mar 1992. (Proprietary).

[44] K. Sanzgiri, B. Dahill, B. Levine, and E. Belding-Royer. A secure routing protocol for ad hoc networks, Nov 2002.

[45] F. Talucci, M. Gerla, and L. Fratta. Maca-bi (maca by invitation): A receiver oriented access protocol for wireless multihop networks. In *Proc. PIMRC*, Oct. 1997.

[46] D. Wagner. Re: Weak keys in rc4. *Posts*, Sep. 1995.

[47] S. Xu and T. Saadawi. Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks? *IEEE Communications Magazine*, pages 130–137, June 2001.

[48] S. Xu and T. Saadawi. Revealing the problems with 802.11 medium access control protocol in multi-hop wireless ad hoc networks. *Computer Networks*, 38(4), March 2002.

[49] M. Zapata. Secure Ad Hoc on-demand distance vector (SAODV) routing. Technical report, http://www.ietf.org/internet-drafts/draft-guerrero-manet-saodv-00.txt, Internet Draft, 2001.

[50] Y. Zhou, D. Wu, and S. Nettles. Analyzing and preventing MAC-layer denial of service attacks for stock 802.11 systems. In *Workshop on BWSA, BROADNETS*, October 2004.