12-15-2006

Detecting Routing Misbehavior In Mobile Ad Hoc Network

Kejun Liu
University of New Orleans

Detecting Routing Misbehavior In Mobile Ad Hoc Network

A Thesis

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Master of Science
in
Computer Science

by

Kejun Liu

B.S. Tongji University, 1999

December, 2006

To my parents, my sister and brother-in-law,
and my husband, for their support as always.

Acknowledgments

First of all, I would like to give my sincere thanks to my advisor, Dr. Jing Deng. for giving me such a wonderful opportunity to work with him. His patient guidance and precious advices are tremendous treasure not only for this thesis, but also for my career in future. I deeply appreciate for what he did.

I would also like to thank my committee members, Dr. Shengru Tu and Dr. Vassil Roussev. Thanks for kindly serving as my committee members, and providing valuable comments on my thesis work.

In addition, I would like to express my gratitude to Dr. Bin Fu. I had participated in his cryptography class and got a chance to have some discussions with him, which helps me a lot on this thesis.

Last but not the least, I would like to thank Dr. Pramod K. Varshney in Syracuse University, Associate Editor, Dr. J.-P. Hubaux in IEEE Transactions on Mobile Computing and the three anonymous reviewers for their valuable comments to improve this thesis. I want to thank Y. Xue and K. Nahrstedt for providing the simulation code for the BFTR scheme [35], too.

This work was supported in part by the SUPRIA program of the CASE Center at Syracuse University and Louisiana EPSCoR Pfund LBOR0049PR00C.

Contents

Abstract

Routing misbehavior in MANETs (Mobile Ad Hoc Networks) is studied in this thesis. In general, routing protocols for MANETs are designed based on the assumption that all participating nodes are fully cooperative. However, due to the open structure and scarcely available battery-based energy, node misbehaviors may exist. One such routing misbehavior is that some selfish nodes will participate in the route discovery and maintenance processes but refuse to forward data packets. Therefore, we propose the 2ACK scheme that serves as an add-on technique for routing schemes to detect routing misbehavior and to mitigate their adverse effect. The main idea of the 2ACK scheme is to send two-hop acknowledgment packets in the opposite direction of the routing path. In order to reduce additional routing overhead, only a fraction of the received data packets are acknowledged in the 2ACK scheme. Analytical and simulation results are presented to evaluate the performance of the proposed scheme.

List of Figures

List of Tables

Chapter 1

Introduction

Since the world's first wireless local area network (WLAN), ALOHANET, emerged in 1971 at the University of Hawaii, the growth of the wireless network is significantly. Contrasted to the wired network, the wireless network is more flexible and convenient, especially for those who like to use some mobile devices, such as laptop, Personal Digital Assistant (PDA) etc. There is no need to look for an Ethernet port when network connection is needed. The news and Email can be read even in a coffee shop or airport. People get access to the network almost whenever and wherever they want.

However, such wireless connections actually are not really available anywhere. The connection is constrained by the pre-existed base stations (or access points). In order to get connected, people have to at least stay within the communication radius of one base station. Ad Hoc network emerged under the demand of some special tasks, such as search/rescue after an earthquake, or communication in a battle field, where the network infrastructures are either destroyed or never existed.

A Mobile Ad Hoc Network (MANET) is a collection of mobile nodes (hosts) which communicate with each other via wireless links either directly or relying on other nodes as routers. The operation of MANETs does not depend on pre-existing infrastructure or base stations. Network nodes in MANETs are free to move randomly. Therefore, the network topology of a MANET may change rapidly and unpredictably. All network activities, such as discovering the topology and delivering data packets, have to be executed by the nodes themselves, either individually or collectively. Depending on its application, the structure of a MANET may vary from a small, static network that is highly power-constrained to a large-scale, mobile, highly dynamic network.

There are two types of MANETs: closed and open [27]. In a closed MANET, all mobile nodes cooperate with each other toward a common goal, such as emergency search/rescue or military and law enforcement operations. In an open MANET, different mobile nodes with different goals share their resources in order to ensure global connectivity. However, some resources are consumed quickly as the nodes participate in the network functions. For instance, battery power is considered to be most important in a mobile environment. An individual mobile node may attempt to benefit from other nodes, but refuse to share its own resources. Such nodes are called *selfish* or *misbehaving* nodes, and their behavior is termed *selfishness* or *misbehavior* [7]. One of the major sources of energy consumption in mobile nodes of MANETs is wireless transmission [12]. A selfish node may refuse to forward data packets for other nodes in order to conserve its own energy.

There are three misbehaving node models related to a routing protocol such as Dynamic Source Routing (DSR [18])[1] are defined in [26]:

- **Selfish node of type 1 (SN1)**: An SN1 node does not perform any packet forwarding function for the data packets unrelated to itself. However, it operates normally in the Route Discovery and the Route Maintenance phases of the DSR protocol;

- **Selfish node of type 2 (SN2)**: An SN2 node neither participates in the Route Discovery/Maintenance phase nor in data packet forwarding. It only spends its battery

[1]Due to DSR's popularity, we use it as the basic routing protocol to illustrate our proposed add-on scheme. The details of DSR can be found in [18]. The implementation of our scheme on other routing schemes will be discussed in Chapter 7.

energy to send or receive its own data packets;

- **Selfish node of type 3 (SN3)**: An SN3 node's behavior depends on the level of its battery energy. When the energy level is higher than a threshold E_1, the node behaves normally. When the energy level is lower than E_1 but still higher than a lower threshold E_2 ($E_2 < E_1$), it behaves as an SN1 node. When the energy level is lower than E_2, the node behaves as an SN2 node.

It is clear that SN3 represents a dynamic behavior involving the behaviors of a well-behaving node, SN1, and SN2 depending on the energy level of a node. The difference between SN1 and SN2 is whether or not the node participates in regular routing operations such as Route Discovery and Route Maintenance. It is relatively easy to distinguish an SN2 node from a well-behaved one. The neighbors can simply make their own judgment by checking their records of routing topology. Once an SN2 node is found, the well-behaved nodes may refuse to serve the traffic generated by the SN2 node. Therefore, an SN2 node can be simply isolated and removed from the MANET.

On the contrary, distinguishing an SN1 node from a group of well-behaved nodes is much more difficult. By participating in the routing process, an SN1 node will be included in the routing topology and its data traffic will be forwarded by other well-behaved nodes. The damage caused by SN1 nodes on MANETs is significant. Such misbehaving nodes support the Route Discovery phase but interrupt data forwarding. When the source node of the data traffic notices the problem with the chosen route, it can either choose an alternate route from its route cache or initiate a new Route Discovery phase. The alternate route may again contain misbehaving nodes and therefore data transmission may fail again. The new Route Discovery phase will return a similar set of routes that may contain misbehaving nodes. After a few failed tries, the source node may conclude that routes are unavailable to deliver the data packets and the MANET fails to provide a reliable communication structure for mobile nodes. Therefore, it is an important task to identify an SN1 node in a MANET (SN3 nodes may be identified using a similar dynamic method of identifying SN1 nodes.)[2]

The adverse effects of routing misbehavior caused by SN1 nodes can be explained in the following.

SN1 nodes participate in the Route Discovery phase. Therefore, they may be included in the routes chosen to forward the data packets from the source. However, the SN1 nodes do not forward the data packets when such packets arrive. This leads to the source being confused. The alternative route chosen from the route cache of the source or the results of a new round of Route Discovery phase may contain SN1 nodes as well.

In order to mitigate the adverse effects of SN1 nodes, the nodes need to be detected so that these nodes can be avoided by all well-behaved nodes. In this thesis, we focus on the following problem:

*(***Misbehavior Detection and Mitigation***) In MANETs, routing misbehavior can severely degrade the performance at the routing layer. Specifically, nodes may participate in the route discovery and maintenance processes but refuse to forward data packets. How do we detect such misbehavior? How to make such detection process more efficient (i.e., with less control overhead) and accurate (i.e., with low false alarm rate and missed detection rate)?*

We propose the 2ACK scheme to mitigate the adverse effects of SN1 nodes. The basic idea of the 2ACK scheme is that, when a node forwards a data packet successfully over the next hop, the destination node of the next-hop link will send back a special two-hop acknowledgment called 2ACK to indicate that the data packet has been received successfully. Such a 2ACK transmission takes place only for a fraction of data packets, but not all. Such a "selective" acknowledgment[3] is intended to reduce the additional routing overhead caused

[2]Since SN3 nodes can be detected with techniques similar to those that detect SN1 nodes, we focus on SN1 nodes henceforth.

[3]It will become clear later that the acknowledgment in the 2ACK scheme is different from SACK in TCP.

by the 2ACK scheme. Judgment on node behavior is made after observing its behavior for a certain period of time.

In this thesis, we present the details of the 2ACK scheme and our evaluation of the 2ACK scheme as an add-on to the Dynamic Source Routing (DSR [18]) protocol. The rest of the thesis is organized as follows: In Chapter 2, we summarize the various approaches for route misbehavior detection and mitigation that have been proposed and studied in the literature. In Chapter 3, we briefly describe the DSR protocol. In Chapter 4, we present the problem and discuss the performance degradation caused by the misbehaving nodes in MANETs. The details of the 2ACK scheme and related discussion are given in Chapter 5. In Chapter 6, we present our simulation results that compare the DSR scheme, the DSR+2ACK scheme, and other related schemes. We conclude the work in Chapter 7.

Chapter 2

Related Work

The security problem and the misbehavior problem of wireless networks including MANETs have been studied by many researchers, e.g., [21, 34, 37] and [2]. Various techniques have been proposed to prevent selfishness in MANETs. These schemes can be broadly classified into two categories: *credit-based* schemes and *reputation-based* schemes.

2.1 Credit-Based Schemes

The basic idea of *credit-based* schemes is to provide incentives for nodes to faithfully perform networking functions. In order to achieve this goal, virtual (electronic) currency or similar payment system may be set up. Nodes get paid for providing services to other nodes. When they request other nodes to help them for packet forwarding, they use the same payment system to pay for such services [8, 9, 15, 17].

In [8], Buttyan and Hubaux used the concept of *nuggets* (also called *beans*) as payments for packet forwarding. They proposed two models: the Packet Purse Model and the Packet Trade Model. In the Packet Purse Model, nuggets are loaded into the packet before it is sent. The sender puts a certain number of nuggets on the data packet to be sent. Each intermediate node earns nuggets in return for forwarding the packet. If the packet exhausts its nuggets before reaching its destination, then it is dropped. In the Packet Trade Model, each intermediate node "buys" the packet from the previous node for some nuggets, and "sells" it to the next node for more nuggets. Thus, each intermediate node earns some nuggets for providing the forwarding service, and the overall cost of sending the packet is borne by the destination.

In [9], each node maintains a counter termed *nuglet counter*. The counter is decreased when the node sends packets of its own, but increased when it forwards packets for the other nodes. The counter should be positive before a node is allowed to send its packet. Therefore, the nodes are encouraged to continue to help other nodes. Tamper resistant hardware modules are used to keep nodes from increasing the nuglet counter illegally.

Another credit-based scheme, termed Sprite, was proposed by Zhong et al. in [36]. In Sprite, nodes keep receipts of the received/forwarded messages. When they have a fast connection to a Credit Clearance Service (CCS), they report all these receipts. The CCS then decides the charge and credit for the reporting nodes. In the network architecture of Sprite, the CCS is assumed to be reachable through the use of Internet, limiting the utility of Sprite.

The main problem with credit-based schemes is that they usually require some kind of tamper-resistant hardware and/or extra protection for the virtual currency or the payment system. We focus on reputation-based techniques in this thesis instead.

2.2 Reputation-Based Schemes

The second category of techniques to combat node misbehavior in MANETs is *reputation-based* [6,24]. In such schemes, network nodes collectively detect and declare the misbehavior of a suspicious node. Such a declaration is then propagated throughout the network, so that the misbehaving node will be cut off from the rest of the network.

In [24], Marti et al. proposed a scheme that contains two major modules, termed *watchdog* and *pathrater*, to detect and mitigate, respectively, routing misbehavior in MANETs. Nodes operate in a promiscuous mode wherein, the watchdog module overhears the medium to check whether the next-hop node faithfully forwards the packet. At the same time, it maintains a buffer of recently sent packets. A data packet is cleared from the buffer when the watchdog overhears the same packet being forwarded by the next-hop node over the medium. If a data packet remains in the buffer for too long, the watchdog module accuses the next-hop neighbor to be misbehaving. Thus, the watchdog enables misbehavior detection at the forwarding level as well as the link level. Based on watchdog's accusations, the pathrater module rates every path in its cache and subsequently chooses the path that best avoids misbehaving nodes. Due to its reliance on overhearing, however, the watchdog technique may fail to detect misbehavior or raise false alarms in the presence of ambiguous collisions, receiver collisions, and limited transmission power, as explained in [24].

The CONFIDANT protocol proposed by Buchegger and Le Boudec in [6] is another example of reputation-based schemes. The protocol is based on selective altruism and utilitarianism, thus making misbehavior unattractive. CONFIDANT consists of four important components - the Monitor, the Reputation System, the Path Manager, and the Trust Manager. They perform the vital functions of neighborhood watching, node rating, path rating, and sending and receiving alarm messages, respectively. Each node continuously monitors the behavior of its first-hop neighbors. If a suspicious event is detected, details of the event are passed to the Reputation System. Depending on how significant and how frequent the event is, the Reputation System modifies the rating of the suspected node. Once the rating of a node becomes intolerable, control is passed to the Path Manager, which accordingly controls the route cache. Warning messages are propagated to other nodes in the form of an *Alarm* message sent out by the Trust Manager.

The Monitor component in CONFIDANT scheme observes the next hop neighbor's behavior using the overhearing technique. This causes the scheme to suffer from the same problems as the watchdog scheme.

In [27], Miranda and Rodrigues adopted a similar approach. Each node i maintains a data structure $Status_i[j]$ about every other node j, as an indication of what impression node i has about node j. Along with a credit counter, node i also maintains lists of nodes to which node j will and will not provide service. Every node periodically broadcasts relevant information in the form of a self-state message. Other nodes update their own lists based on the information contained in these self-state messages.

2.3 End-to-end Acknowledgment Schemes

There are several schemes that use end-to-end acknowledgments (ACKs) to detect routing misbehavior or malicious nodes in wireless networks.

In the TCP protocol, end-to-end acknowledgment is employed. Such acknowledgments are sent by the end-receiver to notify the sender about the reception of data packets up to some locations of the continuous data stream. The Selective Acknowledgment (SACK) technique is used to acknowledge out-of-order data blocks.

The 2ACK technique differs from the ACK and the SACK schemes in the TCP protocol in the following manner: the 2ACK scheme tries to detect those misbehaving nodes which have agreed to forward data packets for the source node but refuse to do so when data packets arrive. TCP, on the other hand, uses ACK and SACK to measure the usefulness of the current route and to take appropriate action. For example, congestion control is based on the reception of the ACK and the SACK packets.

In order to identify malicious routers that draw traffic towards themselves but fail to correctly forward the traffic, Padmanabhan and Simon proposed the *secure traceroute* protocol [29]. The normal traceroute protocol allows the sender to simply send packets with increasing Time-To-Live (TTL) values, and wait for a warning message from the router at which time the packet's TTL value expires. The secure traceroute protocol authenticates the traceroute packets and disguises them as regular data packets.

In [3], Awerbuch et al. proposed an On-Demand Secure Routing Protocol to adaptively probe faulty links on the route being used. Similar to the secure traceroute scheme, binary search is initiated on faulty routes. Asymptotically, $\log(n)$ probes are needed to identify a faulty link on a faulty n-hop route. This technique only works with static misbehaviors and needs to disguise the probing messages as regular routing control packets. Once a link is identified as faulty, the link weight is increased so that the future link selections will avoid this link.

The Best-effort Fault-Tolerant Routing (BFTR) scheme due to Xue and Nahrstedt [35] also employs the end-to-end ACKs. The BFTR scheme continuously monitors the quality (i.e., packet delivery ratio) of the path in use. This is compared with the predefined expected behavior of good routes. If the behavior of the route in use deviates from the behavior of good routes, it is marked as "infeasible" and a new route is used. Since BFTR throws out the entire route before detecting the misbehaving nodes, the newly chosen route may still include the same misbehaving nodes. Even though the new route will be detected as infeasible by the source after a period of observation time, data packet loss will occur in traffic flows when using protocols such as UDP. Such a repeated detection process is inefficient. In contrast with BFTR, we try to identify such misbehaving links in this thesis. Therefore, more accurate information on routing misbehavior can be obtained in the 2ACK scheme.

Compared with the schemes in [3, 29, 35], the 2ACK scheme does not rely on end-to-end acknowledgment. Such an acknowledgment scheme may not exist in some traffic flows (such as UDP). Instead, the 2ACK scheme tries to detect misbehaving links as the links are being used. Such a proactive detection approach results in quicker detection and identification of misbehaving links. Note that it may be beneficial to include end-to-end acknowledgments in the 2ACK scheme. In such a combined scheme, the 2ACK transmission and the monitoring processes are turned on only when routing performance degrades. It will further reduce the routing overhead of the 2ACK scheme.

In [10], Conti et al. proposed a scheme to choose routes based on the reliability index of each outgoing neighbor. Each node maintains a table of reliability indices of its neighbors. Such a reliability index reflects the past success/failure experience of packet transmissions through this neighbor. For example, a successful end-to-end transmission will result in an increase of the reliability index of the neighbor associated with the route. When choosing routes for data transmissions, nodes prefer those rooted at the neighbors with higher reliability indices. Different policies for route selection were investigated in [10]. Since a source node judges all potential routes through its immediate neighbors, the overall reliability of the chosen route depends on how the neighbors choose the rest of the route. Here, we propose a scheme to detect misbehaving links and to avoid them as much as possible.

2.4 Other Prior State-of-the-art Schemes

The misbehavior problem that we focus on in this thesis was referred to as the Black Hole attack in [2, 14]. In [2], Aad et al. investigated the JellyFish attack for closed-loop flows such as TCP. It was shown that a JellyFish attacker may stealthily re-arrange, delay, or periodically drop packets while still remaining protocol-compliant. Such attacks may cause end-to-end throughput of closed-loop flows to drop. Similarly, the Black Hole attack was also shown to have adverse effect on open-loop flows such as UDP. Unlike [2], we propose a 2ACK technique to detect such misbehaviors.

Several other interesting techniques have been proposed to address the issue of potential node misbehavior in MANETs. For example, Srinivasan et al. addressed the issue of user cooperation in MANETs [33]. Behavior of nodes was assumed to be rational, i.e., their actions were strictly determined by self interest. A Generous TIT-FOR-TAT (GTFT) scheme was used to make sure that a Nash equilibrium would be achieved. Such an equilibrium will lead to optimized throughput performance for all nodes in the network. The problem of a few misbehaving nodes cannot be solved by this approach.

Mahajan et al. proposed a CATCH scheme to allow cooperative nodes to detect free-riders in the neighborhood [23]. A free-rider is defined as a node that does not provide service to other nodes but requests service from others. The CATCH scheme also allows the cooperative neighbors of a free-rider to isolate it from the rest of the network. The CATCH scheme is essentially built on top of the watchdog scheme in [24]. We will discuss the difference between our proposed scheme and the watchdog scheme in Section 5.2.

2.5 The TWOACK and S-TWOACK Schemes

The TWOACK scheme was proposed in In [4]. The 2ACK and the TWOACK schemes have the following major differences: 1) the receiving node in the 2ACK scheme only sends 2ACK packets for a fraction of received data packets, while in the TWOACK scheme TWOACK packets are sent for every data packet received. Acknowledging a fraction of received data packets gives the 2ACK scheme better performance with respect to routing overhead; 2) the 2ACK scheme has an authentication mechanism to make sure that the 2ACK packets are genuine.

The Selective TWOACK (S-TWOACK) scheme proposed in [4] is different from 2ACK as well. Mainly, each TWOACK packet in the S-TWOACK scheme acknowledges the receipt of a number of data packets, but a 2ACK packet in the 2ACK scheme only acknowledges one data packet. With such a subtle change, the 2ACK scheme has an easier control over the tradeoff between the performance of the network and the cost as compared to the S-TWOACK scheme.

Chapter 3

Overview of Dynamic Source Routing

Based on routing information update mechanism, routing protocols for MANET can be divided into three major categories: proactive (or table-driven) routing protocols, reactive (or on-demand) routing protocols, and hybrid routing protocols.

In proactive routing protocols, every node maintains a routing table, which includes the routes to all the destinations, even it is not necessary. In order to maintain the updated routing information, the nodes exchange the routing information each other periodically by flooding in the whole network. Therefore, the routing overhead may be extremely high for a large network or high mobility network, but the route finding delay will be relatively low. The Destination Sequenced Distance Vector (DSDV) [31] routing protocol, Wireless Routing Protocol (WRP) [28], and the Optimized Link State Routing (OLSR) [16] protocol are some examples of proactive routing protocols.

On the contrary, the reactive routing protocols, such as the Dynamic Source Routing protocol [18], the Ad hoc On-Demand Distance Vector (AODV) routing protocol [30], etc., do not maintain such routing information all the time. They request a route finding process only when a node needs to communicate with the others and does not know a route to the destination node. Such on-demand routing protocol lowers the entire control overhead, but may cause a longer route finding delay.

The hybrid routing protocols combines the advantages of both proactive routing protocol and reactive routing protocol.

DSR [18] is one of the most popular routing protocols in MANETs. It is specifically designed for multi-hop communication in wireless network with mobile nodes. DSR adapts to the fully autonomous network, without the need of pre-existed infrastructure. The protocol is composed of two main processes which are *Route Discovery* and *Route Maintenance* respectively. *Route Discovery* is a the process by which a node S tries to find a source route reaching a destination node D to achieve its transmission. Route Discovery is used only when S attempts to communication with D but does not know a route to D yet. *Route Maintenance* is the process by which node S is able to get the knowledge of the recent topology changes in the network, such as an unavailable path to destination D due to a broken link. When Route Maintenance informs S of a source route is broken, S will either try to use another route if it happens to have it cached, or will initiate a Route Discovery again to find a new route for subsequent packets to D. These two mechanisms execute totally "on demand". DSR does not advertise any routing information periodically in the network, which lowers the extra control overhead transmitted in the channel. The above two mechanisms work together to allow mobile nodes to discover and maintain routes to arbitrary destinations in MANET as needed.

We will only presents the aspects of DSR which is relevant to our research in this chapter. More detail about DSR scheme can be found in [18].

3.1 Route Discovery

When a node S attempts to send the data packets to a destination node D, but does not know a source route towards the destination D, it will initiate a route discovery process by sending out a ROUTE REQUEST packet (RREQ). S is known as the *initiator* of RREQ packet, and D is the *Target*. When a route to the target is found, a ROUTE REPLY (RREP) packet will be sent back to the initiator, and the initiator will record this route into its *Route Cache*.

The RREQ packet is propagated in the network by flooding. Each node receiving the RREQ packet, will rebroadcast to its neighbors, unless one of the following situation happened: 1) It is the target of the RREQ packet. 2) It knows a route towards the target. 3) It has already forward the packet. 4) The Time-To-Live (TTL) of the RREQ packet is expired.

When a source node creates an RREQ packet, it chooses a unique sequence number for the RREQ packet, Upon receiving an RREQ packet, an intermediate node will execute the following steps in sequence.

- If the target address field in the packet matches its own address, then it should return an RREP packet to the initiator along the reverse direction of the path through which the RREQ packet arrived.

- Else, the node must check the sequence number of the RREQ packet, to see whether this is an "old" request which has been processed before. If so, the RREQ packet will be discarded to avoid duplicate transmission of a same request and potential route loop.

- Else, the node should check its route cache, to see whether it has known a route to the target. If so, the node generates an RREP packet and sends back to the initiator, and stops propagating the RREQ packet further.

- Else, the node will append its own address to the RREQ packet, and rebroadcast to its neighbors if the Time-To-Live (TTL) value is not expired.

When initiating a route discovery, the initiator node records a copy of the original data packet (that triggered the Discovery) in a local buffer called the *Send Buffer*. The send buffer contains a copy of each data packet that cannot be sent out by this node due to the unavailable source route to the destination. Each packet in the send buffer is stamped an entry time. If it stays in the send buffer exceeding a certain period of time, the packet is discarded. So if the send buffer of a node is not empty, the node should occasionally originate a new route discovery for the packet's destination address. However, the node MUST limit such re-originate of RREQ at a certain rate to avoid the potential RERR flood, because it is possible that the destination node is not currently reachable due to the probable network partition issue.

3.2 Route Maintenance

When a node originates or forwards a packet using a source route, it is responsible to make sure the availability of the link between itself and the next hop, and in some way to guarantee the successful reception of its next hop. This acknowledgement is usually provided at no cost in wireless network, either as a part of existing standard (such as the link-layer acknowledgement frame defined by IEEE 802.11), or by a "passive acknowledgement" [20] (for example, the node confirms receipt at its next hop node by overhearing its next hop node forwards the same packet).

If there is no such built-in confirmation mechanism in use, the node should request an explicitly DSR-specific acknowledgement by setting an acknowledgement request option in the DSR options header in the packet.

When a node is unable to verify the availability of the link to its next hop node after exceeding a maximum number of retransmission attempts, it will send a ROUTE ERROR (RERR) message to the source node of the data packet. All the node, upon receiving or overhearing this RERR packet, will remove the route from its route cache. Furthermore, the source node will resume the transmission of the subsequent packets or retransmit the failed packets (if required by the upper layer. For example, TCP service.) either using an alternate route (if existed) or initiating a fresh route discovery process.

Chapter 4

Problem of Routing Misbehavior

In this chapter, we describe the problems caused by routing misbehavior. But first, we summarize our notations and assumptions used throughout this thesis.

4.1 Notations and Assumptions

This chapter outlines our assumptions regarding the properties of the physical and network layers. Throughout this thesis, we assume bi-directional communication. Such a symmetry of links is needed for the transmission of the designed 2ACK packets. Our scheme works with source routing, such as DSR [18]. We further assume that there is no collusion among misbehaving nodes. We argue that misbehavior caused by selfishness are usually limited to individual nodes in MANETs.

We use the following notations throughout the thesis:

- $X * Y$: the size of network area;
- N: the total number of nodes in the network;
- R: the transmission range of each node. We assume that the transmission of all nodes is omni-directional and the transmission range is homogeneous. We assume $R = 250$

m in our simulations;

- V_m: the maximum speed of a mobile node;
- h: the average number of hops from the source node to the destination node;
- ℓ: the expected progress of one-hop transmission;
- d: the expected distance between the source node and the destination node;
- p_m: the fraction of nodes that are misbehaving. This is also the probability of a node being a misbehaving node. The misbehaving nodes are selected among all network nodes randomly. In our simulations, p_m ranges from 0 to 0.4;
- p_r: the probability of a misbehaving route, i.e., the probability of a route with at least one misbehaving router;
- R_{mis}: the threshold to determine the allowable ratio of the total number of 2ACK packets missed to the total number of data packets sent;
- R_{ack}: the acknowledgment ratio, the fraction of data packets that are acknowledged with 2ACK packets (maintained at the 2ACK sender);
- τ: the value of *timeout*, beyond which time a data packet will be considered as unacknowledged;
- T_{obs}: the observation period prior to declaring node misbehavior;
- C_{mis}: the counter of missing 2ACK packets (maintained at the observing node);
- C_{pkts}: the counter of forwarded data packets (maintained at the observing node).

4.2 Routing Misbehavior Model

We present the routing misbehavior model considered in this thesis in the context of the DSR protocol [18]. Due to DSR's popularity, we use it as the basic routing protocol to illustrate our proposed add-on scheme. The details of DSR can be found in [18]. The implementation of our scheme as an add-on to other routing schemes will be discussed in Chapter 6.

We focus on the following routing misbehavior: a selfish node does not perform the packet forwarding function for the data packets unrelated to itself.[1] However, it operates normally in the Route Discovery and the Route Maintenance phases of the DSR protocol. Since such misbehaving nodes participate in the Route Discovery phase, they may be included in the routes chosen to forward the data packets from the source. The misbehaving nodes, however, refuse to forward the data packets from the source. This leads to the source being confused.

In guaranteed services such as TCP, the source node may either choose an alternate route from its route cache or initiate a new Route Discovery process. The alternate route may again contain misbehaving nodes and therefore the data transmission may fail again. The new Route Discovery phase will return a similar set of routes including the misbehaving nodes. Eventually, the source node may conclude that routes are unavailable to deliver the data packets. As a result, the network fails to provide reliable communication for the source node even though such routes are available. In best-effort services such as UDP, the source simply sends out data packets to the next-hop node which forwards them on. The existence of a misbehaving node on the route will cut off the data traffic flow. The source has no knowledge of this at all.

In this thesis, we propose the 2ACK technique to detect such misbehaving nodes. Routes

[1] In some networks, a router may be considered well-behaved as long as it sends out the packet toward the next-hop node. This, however, does not guarantee the successful reception of the packet at the next-hop node. Such a behavior by the router, if consistently repeated, will be considered as misbehavior in this work. After all, it is the router's responsibility to make sure of the successful reception of the packet at the next-hop node when it responded to the route-discovery process.

containing such nodes will be eliminated from consideration. The source node will be able to choose an appropriate route to send its data. in this thesis, we use both UDP and TCP to demonstrate the adverse effect of routing misbehavior and the performance of our proposed scheme.

The attackers (misbehaving nodes) are assumed to be capable of performing the following tasks:

- dropping any data packet;
- masquerading as the node that is the receiver of its next-hop link;
- sending out fabricated 2ACK packets;
- sending out fabricated h_n, the key generated by the 2ACK packet senders;
- claiming falsely that its neighbor or next-hop links are misbehaving.

4.3 Probability of Misbehaving Routes

In order to demonstrate the adverse effect of routing misbehavior, we estimate the probability of misbehaving routes in this section. A route is defined as misbehaving when there is at least one router along the route that can be classified as misbehaving.

Our analysis is based on the following assumptions:

- The network nodes are randomly distributed over the entire network area. Each node's location is independent of all other nodes' locations. There are N nodes in the network area of size $X * Y$;
- The source and the destination of each transaction are chosen randomly among all nodes;

- Nodes (other than the source and the destination) are chosen as misbehaving nodes, independently, with probability p_m.

We examine a route with an average number of hops, h. There are $h-1$ routers between the source and the destination. Each of these routers may misbehave with probability p_m. The probability of the route with at least one misbehaving node is:

$$p_r = 1 - (1 - p_m)^{h-1} \ . \tag{4.1}$$

In order to estimate p_r, we need to know h, the average number of hops of a route. We use the following approach: we first estimate the average progress of each hop, ℓ, in the network; we then estimate the average distance, d, between the source and the destination; the value of h can be estimated as d/ℓ.

The average one-hop progress, ℓ, can be approximated as the average of the maximum distance between a sender and each of the neighbors within its transmission range.[2] We calculate the average number of nodes in the transmission circle, ξ:

$$\xi = \frac{N}{X * Y} \cdot \pi R^2 \ , \tag{4.2}$$

where $X * Y$ is the size of the network area and $\frac{N}{X*Y}$ is the node density.

For simplicity of discussion, we assume that ξ is an integer. The probability of all ξ nodes

[2]Note that this is only an approximation, which assumes that the farthest neighbor from the sender is always in the direction toward the destination. Our simulation results presented later in this subsection show that our approximation works quite well.

residing within distance r from the center of the transmission circle can be expressed as

$$
\begin{aligned}
F(r) &= \text{Prob(All } \xi \text{ nodes reside within a circle of radius } r) \\
&= [\text{Prob(a node resides within } r)]^{\xi} \\
&= \left[\frac{\pi r^2}{\pi R^2}\right]^{\xi} \\
&= \frac{r^{2\xi}}{R^{2\xi}} ,
\end{aligned}
$$

where we have used the assumptions of node location independence and randomness.

The Probability Density Function (pdf) of progress r from the source is

$$f(r) = \frac{\partial}{\partial r} F(r) = \frac{2\xi \cdot r^{2\xi - 1}}{R^{2\xi}} .$$

The average progress is then the expected value of r with respect to pdf $f(r)$,

$$\ell = \int_0^R r f(r) dr = \frac{2\xi \cdot R}{2\xi + 1} . \tag{4.3}$$

Based on (4.3): when $\xi = 0$, no progress can be made ($\ell = 0$); when $\xi = 1$, the progress is the expected value of the distance at which the sole node is located from the center, $\ell = \frac{2}{3}R$; when ξ is large, the progress approaches R, $\ell \to R$.

In a network area of size $X * Y$, the average distance between the source and the destination can be approximated by

$$d \approx (0 + \sqrt{X^2 + Y^2})/2 . \tag{4.4}$$

Therefore, the expected number of hops can be estimated as

$$h \approx \frac{d}{\ell} \approx \frac{\sqrt{X^2 + Y^2}}{2\ell} \approx \frac{(2\xi + 1) \cdot \sqrt{X^2 + Y^2}}{4\xi R} , \tag{4.5}$$

where we have implicitly assumed that the average progress made on a hop is independent of the average progress made on the previous hops.

Combining (4.1) and (4.5), we have

$$p_r = 1 - (1 - p_m)^{\frac{(2\xi+1)\cdot\sqrt{X^2+Y^2}}{4\xi R} - 1} , \tag{4.6}$$

where ξ is given by (4.2).

We have compared the numerical results based on (4.6) and simulation results. Our simulation results were obtained through 20 runs with different seeds in NS2 [1]. In Table 4.1, we show the results for different network areas and number of nodes. The transmission range is $R = 250$ m for every node.

Based on Table 4.1, we can conclude that, as expected, the probability of misbehaving route, p_r, increases with p_m. This probability also increases with network area because the routes are longer. The values of p_r obtained analytically are larger than those obtained using simulation. This is due to our estimation of d in (4.4) that is higher than the actual values. In addition, the estimation of ℓ in (4.3) is smaller than the actual value. The adverse effects of misbehaving nodes in MANETs can be seen clearly in Table 4.1. For example, in a network of $5R * 5R$ and $p_m = 0.2$, around 50% of the routes contain at least one misbehaving node. With such a high probability of misbehaving route, p_r, the throughput performance of the MANET will be severely degraded. This motivates our development of an efficient approach for detection and mitigation of routing misbehavior.

Table 4.1: Probability of misbehaving routes for different misbehavior ratio, p_m

Results for p_m = 0.1			
Network Area, X*Y	4R*4R	5R*5R	10R*10R
Number of Nodes, N	70	100	400
Analytical Results	0.18	0.25	0.49
Simulation Results	0.17	0.22	0.43
Results for p_m = 0.2			
Network Area, X*Y	4R*4R	5R*5R	10R*10R
Number of Nodes, N	70	100	400
Analytical Results	0.35	0.45	0.76
Simulation Results	0.31	0.39	0.65
Results for p_m = 0.3			
Network Area, X*Y	4R*4R	5R*5R	10R*10R
Number of Nodes, N	70	100	400
Analytical Results	0.50	0.62	0.90
Simulation Results	0.42	0.52	0.76

Chapter 5

The 2ACK Scheme

The watchdog detection mechanism in [24] has a very low overhead. Unfortunately, the watchdog technique suffers from several problems such as ambiguous collisions, receiver collisions, and limited transmission power. The main issue is that the event of successful packet reception can only be accurately determined at the receiver of the next-hop link, but the watchdog technique only monitors the transmission from the sender of the next-hop link.

Noting that a misbehaving node can either be the sender or the receiver of the next-hop link, we focus on the problem of detecting *misbehaving links* instead of misbehaving nodes. In the next-hop link, a misbehaving sender or a misbehaving receiver has a similar adverse effect on the data packet: it will not be forwarded further. The result is that this link will be tagged [3]. Our approach discussed here simplifies the detection mechanism significantly.

5.1 Details of the 2ACK Scheme

The 2ACK scheme is a network-layer technique to detect misbehaving links and to mitigate their effects. It can be implemented as an add-on to existing routing protocols for MANETs, such as DSR. The 2ACK scheme detects misbehavior through the use of a new type of acknowledgment packet, termed 2ACK. A 2ACK packet is assigned a fixed route of two

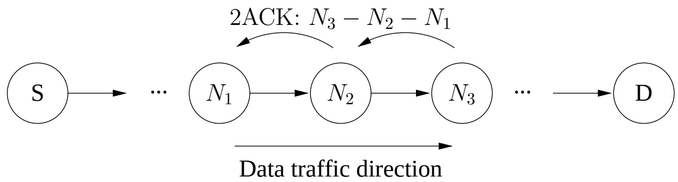

Figure 5.1: The 2ACK Scheme

hops (three nodes), in the opposite direction of the data traffic route.

Figure 5.1 illustrates the operation of the 2ACK scheme. Suppose that N_1, N_2, and N_3 are three consecutive nodes (*triplet*) along a route. The route from a source node, S, to a destination node, D, is generated in the Route Discovery phase of the DSR protocol. When N_1 sends a data packet to N_2 and N_2 forwards it to N_3, it is unclear to N_1 whether N_3 receives the data packet successfully or not. Such an ambiguity exists even when there are no misbehaving nodes. The problem becomes much more severe in open MANETs with potential misbehaving nodes.

The 2ACK scheme requires an explicit acknowledgment to be sent by N_3 to notify N_1 of its successful reception of a data packet: when node N_3 receives the data packet successfully, it sends out a 2ACK packet over two hops to N_1 (i.e., the opposite direction of the routing path as shown), with the ID of the corresponding data packet. The triplet $[N_1 \rightarrow N_2 \rightarrow N_3]$ is derived from the route of the original data traffic. Such a triplet is used by N_1 to monitor the link $N_2 \rightarrow N_3$. For convenience of presentation, we term N_1 in the triplet $[N_1 \rightarrow N_2 \rightarrow N_3]$ as the 2ACK packet receiver or the *observing node* and N_3 as the *2ACK packet sender*.

Such a 2ACK transmission takes place for every set of triplets along the route. Therefore, only the first router from the source will not serve as a 2ACK packet sender. The last router just before the destination and the destination will not serve as 2ACK receivers.[1]

[1]The 2ACK packet is different from the selective acknowledgement (SACK) [25] in TCP. The SACK packets are used by the TCP data receiver to acknowledge non-contiguous blocks of data that are not covered by the Cumulative Acknowledgement field. A 2ACK packet, on the other hand, acknowledges the received data packet. In addition, the SACK packets are sent by the data traffic receiver, but the 2ACK packets are sent by the third node in every set of triplets along the traffic route.

N_2 Next Hop Receiver	N_3 Second Hop Receiver	C_{pkts} Packets Transmitted	C_{mis} 2ACK packets Missed	LIST List of data packet IDs

Figure 5.2: Data structure maintained by the observing node

To detect misbehavior, the 2ACK packet sender maintains a list of IDs of data packets that have been sent out but have not been acknowledged. For example, after N_1 sends a data packet on a particular path, say, $[N_1 \rightarrow N_2 \rightarrow N_3]$ in Fig. 5.1, it adds the data ID to LIST (refer to Fig. 5.2, which illustrates the data structure maintained by the observing node), i.e., on its list corresponding to $N_2 \rightarrow N_3$. A counter of forwarded data packets, C_{pkts}, is incremented simultaneously.

At N_1, each ID will stay on the list for τ seconds, the timeout for 2ACK reception. If a 2ACK packet corresponding to this ID arrives before the timer expires, the ID will be removed from the list. Otherwise, the ID will be removed at the end of its timeout interval and a counter called C_{mis} will be incremented.

When N_3 receives a data packet, it determines whether it needs to send a 2ACK packet to N_1. In order to reduce additional routing overhead caused by the 2ACK scheme, only a fraction of the data packets will be acknowledged via 2ACK packets. Such a fraction is termed the acknowledgment ratio, R_{ack}. By varying R_{ack}, we can dynamically tune the overhead of 2ACK packet transmissions.

Node N_1 observes the behavior of link $N_2 \rightarrow N_3$ for a period of time termed T_{obs}. At the end of the observation period, N_1 calculates the ratio of missing 2ACK packets as C_{mis}/C_{pkts} and compares it with a threshold R_{mis}. If the ratio is greater than R_{mis}, link $N_2 \rightarrow N_3$ is declared misbehaving and N_1 sends out an RERR (or the misbehavior report) packet. The data structure of RERR is shown in Fig. 5.3. Since only a fraction of the received data packets are acknowledged, R_{mis} should satisfy $R_{mis} > 1 - R_{ack}$ in order to eliminate false alarms caused by such a partial acknowledgment technique (see Section 5.6).

Option Type	Opt data len	Error Type 2ACK Report Misbehavior	Reserved	Salvage	error source address N_1 (Misbehaving report sender)	Destination S Report receiver	Type−specific information $N_2 - > N_3$ Misbeheving Link

Figure 5.3: Data structure of the RERR packet (the misbehavior report)

Each node receiving or overhearing such an RERR marks the link $N_2 \rightarrow N_3$ as misbehaving and adds to the blacklist of such misbehaving links that it maintains. When a node starts its own data traffic later, it will avoid using such misbehaving links as a part of its route.

The 2ACK scheme can be summarized in the pseudo-code provided in Appendix for the 2ACK packet sender side (N_3) and the observing node side (N_1).

5.2 Comparison with Overhearing Techniques

Compared with the overhearing techniques such as watchdog in [24], the 2ACK scheme solves the problems of ambiguous collisions, receiver collisions, and limited transmission power:

- **Ambiguous Collisions**: Ambiguous collisions may occur at node N_1. When a well-behaved node N_2 forwards the data packet toward N_3, it is possible that N_1 cannot overhear the transmission due to another concurrent transmission in N_1's neighborhood. The 2ACK technique solves this problem by requiring N_3 to send a 2ACK packet explicitly.

- **Receiver Collisions**: Receiver collisions take place in the overhearing techniques when N_1 overhears the data packet being forwarded by N_2, but N_3 fails to receive the packet due to collisions in its neighborhood. A misbehaving N_2 will not retransmit the data packet, which costs extra energy. Again, the 2ACK technique overcomes this problem due to the explicit 2ACK packets.

- **Limited Transmission Power**: A misbehaving N_2 may maneuver its transmission power such that N_1 can overhear its transmission but N_3 cannot. This problem is similar to the Receiver Collisions problem. It becomes a threat only when the distance between N_1 and N_2 is less than that between N_2 and N_3. The 2ACK scheme is immune to limited transmission power problem.

- **Limited Overhearing Range**: A well-behaved N_2 may use low transmission power to send data toward N_3. Due to N_1's limited overhearing range, it will not overhear the transmission successfully and will thus infer that N_2 is misbehaving, causing a false alarm. Both this problem and the limited transmission power problem are caused by the potential asymmetry of communication links. The 2ACK scheme is immune to the limited overhearing range issue.

With the explicit requirement of 2ACK transmissions, the 2ACK scheme solves the above problems. Compared with overhearing techniques, the 2ACK scheme has a disadvantage of higher routing overhead. This additional routing overhead is caused by the transmission of 2ACK packets. However, we will show later that, by reducing the acknowledgment ratio, R_{ack}, the number of 2ACK transmissions can be significantly lowered (Section 5.6).

5.3 Authenticating the 2ACK Packets

We look into the problem of 2ACK packet fabrication in this subsection. Since the 2ACK packets are forwarded by an intermediate node (e.g., node N_2 in Fig. 5.1). Without proper protection, a misbehaving node N_2 can simply fabricate 2ACK packets and claim that they were sent by node N_3. Therefore, an authentication technique is needed in order to protect 2ACK packets from being forged.

A straightforward way to stop N_2 from forging the 2ACK packets is to use the digital signature algorithm. A digital signature is a small number of extra bits of information attached by node N_3. The signature is unique and usually computationally impossible to forge unless the security key of node N_3 is disclosed. Furthermore, the signature may be used to assure the integrity of the transmitted data, i.e., any changes on the signed information will be detected. Typically, the digital signature is implemented relying on asymmetric cryptography, using techniques such as RSA [19]. However, such asymmetric operations are too expensive for the mobile nodes in MANETs which are usually resource constrained.

In [13], an efficient algorithm termed *one-way hash chain* [22] was used to guard against security attacks such as DoS and resource consumption attacks in the destination-sequenced distance vector (DSDV) routing protocol [31]. A one-way hash chain can be constructed based on a one-way hash function, H. The hash function is a transformation that takes a variable-length input and returns a fixed-length bit string, that is, $H : \{0,1\}^* \rightarrow \{0,1\}^\rho$, where ρ is the length, in bits, of the output of the hash function. An ideal hash function H should have the following properties:

- The input can be of any length;
- The output has a fixed length;
- $H(x)$ is relatively easy to compute for any given input x;
- It is computationally infeasible to calculate x from $H(x)$;
- $H(x)$ is collision-free.

The collision-free property assures that the hash results are unique. Examples of such hash functions include MD5 [32] and SHA1 [11].

To create a one-way hash chain, a node picks up a random initial value $x \in \{0,1\}^\rho$ and computes its hash value. The first number in the hash chain h_0 is initialized to x. By using the general formula $h_i = H(h_{i-1})$, for $0 < i \leq n$, for some n, a chain of h_i is formed:

$$h_0, h_1, h_2, h_3, \cdots, h_n \ . \tag{5.1}$$

It can be proven that, given an existing authenticated element of a one-way hash chain, it is feasible to verify the other elements preceding it. For example, given an authenticated value of h_n, a node can authenticate h_{n-3} by computing $H(H(H(h_{n-3})))$ and comparing the result with h_n [13].

Our scheme uses the above one-way hash chain to protect the 2ACK packets against fabrication. In order to use the one-way hash chain in (5.1) to authenticate 2ACK packets, node N_3 must distribute the h_n element to N_1. A traditional approach for such information distribution is through a trusted *certificate authority*. However, in a MANET, nodes roam from one place to another and there is usually no central server or base station to act as a trusted certificate entity. We propose two techniques to distribute the initial authentication element h_n from node N_3 to node N_1.

The first technique is the "transmission extension" mechanism. Using this technique, N_3 increases the transmission power to send the h_n element directly to N_1. This technique bypasses N_2, the potential threat to the distribution of h_n. While such a technique consumes more energy from node N_3, it takes place rather infrequently. It will be seen later that every 2ACK packet uses one element in the one-way hash chain in (5.1). The distribution of a new h_n element is only needed when the entire chain has been used.

An alternative technique to deliver the h_n element is the "multi-path transmission" mechanism. In this method, N_3 sends its h_n through a number of different paths. For instance, a packet carrying the h_n element may be flooded to the local neighborhood. The packet has a Time-To-Live (TTL) value of 2 or 3 hops. This is similar to the broadcast of the RREQ packets in DSR. N_1 employs a majority vote technique to obtain h_n after it receives several copies of h_n. Note that only the misbehaving N_2 is interested in forging a new h_n. Since a majority of the nodes are well-behaved, the true value of h_n can be obtained.

Once the h_n element is distributed from N_3 to N_1, N_3 can use h_i $(0 \le i < n)$ sequentially to sign the 2ACK packets to be sent to N_1. The h_i elements will be disclosed by N_3 one at

N_2 Next Hop Receiver	N_1 Destination	ID sequence number	MAC Signature	h_i hash release

$$MAC = [N_2, N_1, ID]_{h_{i-1}}$$

Figure 5.4: The Packet Format of 2ACK

a time.

Assume that h_{i+1} has been disclosed (initially $i = n - 1$). When node N_3 needs to send a 2ACK packet, it calculates a Message Authentication Code (MAC) based on h_{i-1}, $[N_2, N_1, ID]_{h_{i-1}}$, and attaches the MAC and the h_i value to the 2ACK packet. Figure 5.4 illustrates the packet format of a 2ACK packet. The fields in Fig. 5.4 are explained below:

- N_2: the receiver of next hop, in the opposite direction of the route;
- N_1: the destination of the 2ACK packet, the observing node, that is two-hop away from the 2ACK packet sender;
- ID: the sequence number of the corresponding data packet;
- $[N_2, N_1, ID]_{h_{i-1}}$: Message Authentication Code (MAC), signed with h_{i-1};
- h_i: the newly disclosed element in the one-way hash chain, $0 < i < n$.

Since h_{i+1} is known to N_1, it compares $H(h_i)$ with h_{i+1}. If the results match, the h_i element is accepted and recorded. The 2ACK message must have been sent from node N_3. However, the integrity of the 2ACK packet can only be proven when the next 2ACK packet arrives (with h_{i-1}). When h_{i-1} is disclosed to N_1, it can be used to verify the integrity of the 2ACK packet received last time by calculating the MAC and comparing it with the received one. This is the so-called "delayed disclosure" technique due to Hu et al. [13].

in this thesis, we do not study the overhead caused by the authentication of the 2ACK packets. Compared to traditional security measures, the computation cost of the one-way hash function is relatively low [13]. The communication overhead depends on the length of each element and the value of n, i.e., the size of the one-way hash chain. When n and the size of each element are chosen reasonably, we expect low overhead due to the transmission of h_n.

5.4 *Timeout* for 2ACK Reception, τ

The parameter *timeout*, τ, will be used to set up a timer for 2ACK reception. If the timer expires before the expected 2ACK packet is received, the missing 2ACK packet counter, C_{mis}, will be incremented. Thus, an appropriate value of τ is important for the successful operation of the 2ACK scheme.

It is clear that false alarms may be triggered if τ is too small. On the other hand, if τ is too large, the observing node will have to maintain a longer list, requiring a large memory size. Therefore, τ should be set at a value that is large enough to allow the occurrence of *temporary link failures* (for example, the unsuccessful transmission due to node mobility or local traffic congestion).

It is essential that τ should satisfy

$$\tau > 4 * [\text{single-hop transmission delay}] \ ,$$

where a single-hop transmission delay includes packet transmission delay, random back-off delay at the Medium Access Control (MAC) layer, data processing delay, and potential retransmission delay.

5.5 Observation Period, T_{obs}, and Dynamic Behavior

The 2ACK scheme distinguishes link misbehaviors and temporary link failures by observing the reception of 2ACK packets over a certain period of time, termed observation period, T_{obs}. Since the temporary link failures do not usually last long, such a technique is able to distinguish temporary link failures from link misbehavior.

The value of T_{obs} should be large enough so that several 2ACK packets are transmitted from the 2ACK packet sender to the observing node. This is especially important when the acknowledgment ratio R_{ack} is small. For example, when $R_{ack} = 0.1$, one 2ACK packet will be transmitted for every 10 data packets received. However, the observation period should not be too long. A long observation period means that the observing node takes more time to observe the behavior of the next-hop link before a misbehavior is declared. Data packets may be dropped over this extended period of time and the effectiveness of the misbehavior detection algorithm is reduced.

The observation process should be initiated by the observing node randomly and repeatedly. Therefore, the 2ACK packet sender or forwarder has to transmit 2ACK packets for the entire data duration (based on the acknowledgment ratio, R_{ack}). Such repeated observations will help in the detection of misbehaving nodes which have dynamic behavior depending on their energy levels. When such nodes are well-behaved, the links associated with them will be treated as normal links and used. Once such nodes misbehave, the links associated with them will be detected as misbehaving and other nodes will stop using them.

5.6 Acknowledgment Ratio, R_{ack}

The additional routing overhead caused by the transmission of the 2ACK packets can be controlled by the parameter acknowledgment ratio, R_{ack}, at the 2ACK packet sender. With the use of the parameter R_{ack} in the 2ACK scheme, only a fraction of the received data packets will be acknowledged. Therefore, the parameter R_{ack} provides a mechanism to tune the overhead.[2]

The reduction of overhead comes with a cost: the shrinking of the range over which R_{mis} can take values. When only R_{ack} of the data packets received are acknowledged via the 2ACK packet, $1 - R_{ack}$ of them are not acknowledged. Since $1 - R_{ack}$ of all data packets are not acknowledged at all, R_{mis} should be greater than $1 - R_{ack}$. That is

$$R_{mis} > 1 - R_{ack} \ . \tag{5.2}$$

In a sense, the difference between R_{mis} and $1 - R_{ack}$ serves as the buffer to avoid false alarms. Therefore, increasing R_{ack} lowers the potential buffer to avoid false alarms. We investigate the effect of R_{ack} on routing overhead in Section 6.

5.7 False Misbehavior Reports and Intentional Dropping of 2ACK

A misbehaving node N_1 as shown in Fig. 5.1 may send false misbehavior reports regarding the next-hop link, $N_2 \rightarrow N_3$. However, the 2ACK scheme makes sure that such a behavior will not benefit node N_1: 1) N_1 may still be included in alternative routes; 2) N_1 needs to forward data packets to N_2 as necessary. Otherwise, it will be detected as part of a misbehaving link (by the node preceding it on the route).

A misbehaving node N_3 may refuse to send any 2ACK packet for the data packets that have been received. As a result, N_1 declares the link $N_2 \rightarrow N_3$ as misbehaving and sends a

[2]In practice, the value of an appropriate R_{ack} may depend on the actual extra-cost of sending a 2ACK packet and projected traffic load of the network. It is also possible to change R_{ack} dynamically. We leave this as a future work.

misbehavior report to the source. Since N_3, as a misbehaving node, refuses to forward data packets, N_2 will also declare the link of $N_3 \rightarrow N_4$ (the node following N_3) as misbehaving. Thus, links around node N_3 are declared misbehaving and will be avoided by future route selections.

Note that this might seem to have achieved the goal of slandering node N_2 by N_3. On the contrary, our mechanism of misbehaving link detection instead of misbehaving node detection protects node N_2. The link $N_2 \rightarrow N_3$ will be marked as misbehaving, but there is no accusation of N_2 (or N_3). Other links associated with node N_2 might still be used. Detection of the misbehaving node N_3 and its punishment are trickier. Essentially, consensus needs to be developed among the majority of neighbors of node N_3 to punish it.

Similarly, when there are consecutive misbehaving nodes on the route, the first misbehaving node and its forwarding link will be detected and reported to the source. Such a route will be avoided in the next round of route-discovery.

Topology changes may also lead to false misbehavior reports. When two well-behaved neighboring nodes move out of each other's range, the link between them will fail in terms of data delivery. In 2ACK, this is taken care of by the routing scheme in use (DSR). When the sender of the link notices that the receiver is out of range, it will submit a Route Error (RERR) message to report the link failure.

5.8 Partial Data Forwarding

A misbehaving node may forward data packets partially by forwarding a fraction of the packets and try to cheat the monitoring system. Such a behavior will be detected by the 2ACK scheme. We use the triplet $N_1 \rightarrow N_2 \rightarrow N_3$ in Fig. 5.1 as an example for explanation:

Assume a misbehaving node N_2 receives N_D data packets from N_1 successfully and only forwards a fraction of data packets, say, R_{part} ($0 \leq R_{part} \leq 1$), of N_D toward N_3. We further assume that all data packets forwarded by N_2 are received successfully by N_3. Thus, N_3 receives $R_{part} \cdot N_D$ data packets, and only $R_{ack} \cdot R_{part} \cdot N_D$ of them will be acknowledged by 2ACK packets sent from N_3.

Therefore, in order to cheat the system, a misbehaving node N_2 has to make sure that

$$1 - R_{ack} \cdot R_{part} < R_{mis} \ . \tag{5.3}$$

As the gap between $1 - R_{ack}$ and R_{mis} shrinks, the feasible value of R_{part} approaches 1. Therefore, the 2ACK scheme effectively guards against partial forwarding. Rearranging (5.3), we have

$$R_{part} > \frac{1 - R_{mis}}{R_{ack}} \ . \tag{5.4}$$

Thus, by increasing $\frac{1-R_{mis}}{R_{ack}}$, we force N_2 to forward more data packets. The disadvantage of such an approach is the loss of protection from false alarms.[3]

[3] While we provide some suggested values for the parameters such as R_{mis} and R_{ack}, the system managers of such operating networks will have the flexibility to vary them.

Chapter 6

Performance Evaluation

In this section, we present our implementation and simulation results for performance evaluation. Since the 2ACK scheme works as an add-on technique for the DSR protocol, the performance of the 2ACK scheme is actually the performance of the DSR+2ACK scheme.

6.1 Simulation Methodology and Performance Metrics

In the simulations, we used a version of Network Simulator (*ns-2*) [1] that includes wireless extensions developed by the CMU Monarch project group. We modified the DSR module in ns-2 to simulate misbehaving nodes. The observation period of the 2ACK scheme was set to $T_{obs} = 0.8$ second. Unless specified otherwise, the 2ACK scheme used $R_{ack} = 0.20$, $R_{mis} = 0.85$, and a timeout value of $\tau = 0.15$ second.

The IEEE 802.11 MAC was used with a channel data rate of 11 Mbps. The data packet size was 512 bytes. The wireless transmission range of each node was $R = 250$ m. In the simulations, $N = 50$ mobile nodes were randomly distributed in a 700 m by 700 m flat area. The source and the destination nodes were randomly chosen among all nodes in the network. The total simulation time was 800 seconds. For each data point, 20 simulations (with different seeds) were run to obtain the average value. The 95% confidence intervals of all results are shown as vertical line segments.

Both UDP and TCP traffics have been simulated to evaluate the performance of 2ACK. A random way-point mobility model was assumed with a maximum speed of $V_m = 0, 10, 20$ m/sec and a pause time of 0 second. The mobility scenarios were generated by the "random trip" generic mobility model due to La Boudec and Vojnović [5]. Constant Bit Rate (CBR) traffic was used. Each simulation included 10 CBR sessions, each of which generated 4 packets per second. In simulations for TCP traffic, the maximum node speed was $V_m = 20$ m/sec with a pause time of 0 second. Each simulation ran 10 Telnet sessions.

We used the following metrics to measure the performance of the 2ACK scheme with respect to UDP traffic:

- **Packet Delivery Ratio, PDR:** the ratio of the number of packets received at the destination and the number of packets sent by the source;

- **Routing Overhead, RO:** the ratio of the amount of routing related transmissions (RREQ, RREP, RERR, and 2ACK) to the amount of data transmissions. The amounts are in bytes. Both forwarded and transmitted packets are counted;

- **Number of False Alarm, N_{FA}:** the number of false misbehavior reports.

For TCP traffic flows, the packet delivery ratio as defined in the UDP traffic scenario would be similar for different schemes. This is because the TCP senders automatically detect end-to-end transmission failures. When misbehaving links appear on a route and the acknowledgments from the destination are missing, the source node of a TCP session may slow down or even stop sending packets. Therefore, a more reasonable performance metric is the total number of packets that are received at the destination. We compared a relative throughput, normalized number of packets that are received, of different schemes in the TCP traffic scenario.

6.2 Simulation Implementation

The simulation is written in Tcl script. The mobile node is configured as follows:

$ns_ node-config **-adhocRouting $opt(rp)** \ ;# Routing Protocol

-llType $opt(ll) \ ;# Link Layer Type

-macType $opt(mac) \ ;# Mac Type

-ifqType $opt(ifq) \ ;# Interface Queue Type

-ifqLen $opt(ifqlen) \ ;# Maximum number of packets in IFQ

-antType $opt(ant) \ ;# Antenna Model

-propType $opt(prop) \ ;# Radio Propagation Model

-phyType $opt(netif) \ ;# Network Interface Type

-topoInstance $topo \ ;# Topography

-movementTrace OFF \ ;# Movement Trace Set

-agentTrace ON \ ;# Agent Trace Set

-routerTrace ON \ ;# Routing Trace Set

-macTrace OFF ;# Mac Layer Trace Set

The above configures a mobilenode with all the given values of adhoc-routing protocol, network stack, channel, topography, propagation model, and tracing turned on or off at different levels (router, mac, agent). In case hierarchical addressing is being used, the hier address of the node needs to be passed as well.

Then the mobile node is created.

```
for {set k 0} {$k < $opt(nn)} {incr k} {
      dsr-create-mobile-node $k $misRV_($k) $simMisFrac    ;# Create DSR Node
}

 ### RANDOM MOTION ###
for {set j 0} {$j < $opt(nn)} {incr j} {
       set node_($j) [$ns_ node]
       $node_($j) random-motion 1
       $ns_ initial_node_pos $node_($j) 20
}
```

simMisFrac is the percentage of the misbehavior in the network. **misRV** is the probability of a node being misbehavior. **dsr-create-mobile-node** creates the misbehaving node or well-behaved node by comparing the value of **misRV** and **simMisFrac**. If **misRV** < **simMisFrac**, the node is a misbehaving node. The node is created with random movement by **$node_($j) random-motion 1**.

Then, to start the nodes' random motion by applying the following.

```
### START Nodes ###
for {set j 0} {$j < $opt(nn)} {incr j} {
$node_($j) start
}
```

It starts the mobile node with a random position and updates to change the direction and speed of the node periodically. The destination and speed values are generated in a random fashion.

The source nodes and destination nodes are chosen randomly and independently within all $N = 50$ mobile nodes. A node can either be a traffic source or destination. Each communication duration is also arranged randomly and independently.

```
### DURATION SRC DST ###
for {set num 0} {$num < $opt(numCBR)} {incr num} {
    set duration($num) [$duration_ value]   ;# Set the Traffic Duration
    set src($num) [expr round([$srcdst_ value])]   ;# Pick the Source Node
    set dst($num) [expr round([$srcdst_ value])]   ;# Pick the Destination Node
    if {$src($num) == $dst($num)} {
        puts "SRC = DST ($num)"
        set num [expr ($num - 1)]
    }
}
```

Next, we setup the traffic for each pair of source node and destination. We use the UDP traffic to illustrate the setup configuration.

```
### SETUP TRAFFIC ###
for {set i 0} {$i < $opt(numCBR)} {incr i} {
    set udp($i) [new Agent/UDP]
    $ns_ attach-agent $node_($src($i)) $udp($i)   ;# Setup the UDP Traffic
    set cbr($i) [new Application/Traffic/CBR]   ;# Using CBR
    $cbr($i) set packetSize_ $simCBRPktSize   ;# Setup Packet Size
    $cbr($i) set interval_ $simCBRInterval   ;# Setup Packet Interval
    $cbr($i) attach-agent $udp($i)   ;# Using UDP to Transmit CBR Traffic
    set null($i) [new Agent/Null]
    $ns_ attach-agent $node_($dst($i)) $null($i)   ;# Setup the Destination
    $ns_ connect $udp($i) $null($i)   ;# UDP Traffic is Done
}
```

The UDP source was initiated by **set udp($i) [new Agent/UDP]**, and destination was **set null($i) [new Agent/Null]**. CBR traffic is used for the UDP connections. To attach the UDP source and destination with the source node and destination node respectively, then create the connection by **$ns_ connect $udp($i) $null($i)**.

After the setup, the traffic flow can start.

```
### START ###
for {set i 0} {$i < $opt(numCBR)} {incr i} {
    set cbrstart($i) [expr ($opt(cbrstart) + [$arrival_ value])]
    $ns_ at $cbrstart($i) "$cbr($i) start"
    $ns_ at [expr $cbrstart($i) + $duration($i)] "$cbr($i) stop"
}
```

Stop the traffic after it reaches the arranged traffic duration, then stop the simulation.

```
### END of Simulation ###
for {set i 0} {$i < $opt(nn) } {incr i} {
    $ns_ at $simTime.0 "$node_($i) reset"
}
$ns_ at $simTime.0 "stop"
$ns_ at $simTime.01 "puts \"NS EXITING...\"   $ns_ halt"
```

6.3 Simulation Results for UDP Traffic

Figure 6.1 compares the packet delivery ratio of the 2ACK scheme, the BFTR scheme [35], the S-TWOACK scheme and the original DSR protocol as a function of misbehavior ratio, p_m. We varied p_m from 0 (all the nodes are well-behaved) to 0.4 (40% of nodes misbehave). The maximum speed is $V_m = 20$ m/sec. From the figure, we can observe that most packets were delivered by all four schemes when $p_m = 0$ (no misbehaving nodes). The packet delivery ratio decreases as p_m increases. Compared with the original DSR scheme, the 2ACK scheme

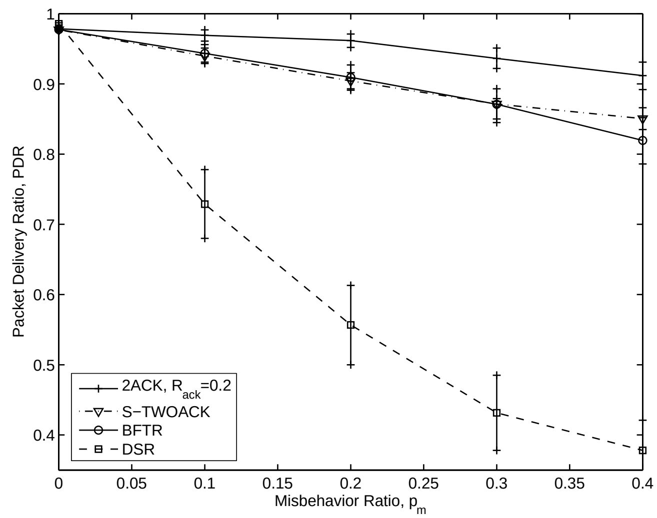

Figure 6.1: Packet Delivery Ratio of 2ACK, BFTR, S-TWOACK and DSR

maintains a much higher PDR. For example, the 2ACK scheme delivered over 90% of data packets even when $p_m = 0.4$. The rest of the packets were dropped because no well-behaved routes could be found from the source to the destination. On the other hand, DSR delivered about 40% of the packets in the same scenario.

Based on Fig. 6.1, the BFTR scheme and the S-TWOACK scheme with *maximum_IDs_Carried* set to 5, i.e., one TWOACK packet is sent for every 5 consecutively received data packets [4], have similar PDR performance. Both are outperformed by the 2ACK scheme. For example, the BFTR scheme delivered roughly 82% and the S-TWOACK scheme delivered about 85% data packets when $p_m = 0.4$. Compared with the 2ACK scheme, the BFTR scheme does not detect misbehaving node/link, it may choose an alternate route which still contains the misbehaving node. The S-TWOACK scheme takes more time to detect misbehaving links, causing more packets being dropped before an alternate route is used.

In Fig. 6.2, we compare the routing overhead of the 2ACK scheme (with $R_{ack} = 0.2$), the BFTR scheme, the S-TWOACK scheme (with *maximum_IDs_Carried* $= 5$), and the DSR scheme. The higher routing overhead in the 2ACK and the S-TWOACK schemes is due to

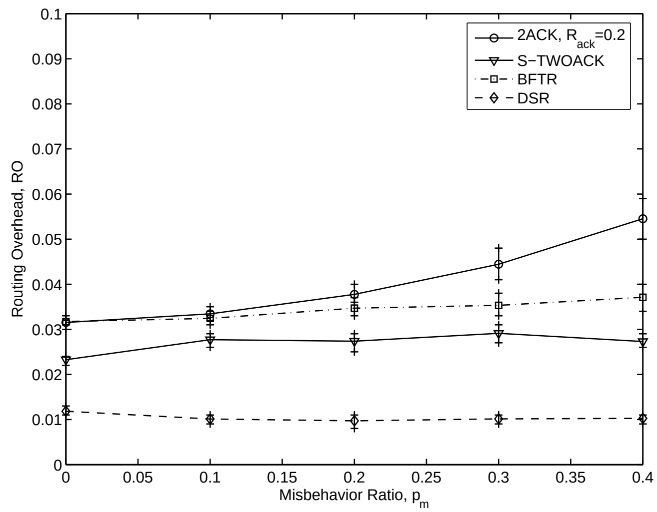

Figure 6.2: Routing Overhead of 2ACK, BFTR, S-TWOACK and DSR

the transmission of extra acknowledgment packets. The extra routing overhead of the BFTR scheme is caused by the extra route discovery processes. The overhead of 2ACK increases with the increase of misbehavior percentage. This is because more RERR (the misbehavior report) and RREQ packets are sent to report misbehaviors and to find alternate routes in a more hostile network environment.

In Fig. 6.3, we show the PDR of the 2ACK scheme with different acknowledgment ratios, R_{ack}. The acknowledgement ratio R_{ack} was set to 0.05, 0.2, 0.50, and 1.0, respectively. The corresponding R_{mis} was 0.98, 0.85, 0.6, and 0.33, respectively. Note that R_{mis} and R_{ack} need to satisfy (5.2). Based on Fig. 6.3, we can see that the PDR performance of the 2ACK scheme is not appreciably affected by R_{ack}.

We compare the routing overhead of the 2ACK scheme with different R_{ack} values in Fig. 6.4. As expected, the routing overhead of the 2ACK scheme is the highest when $R_{ack} = 1$. This is due to the large number of 2ACK packets transmitted in the network. As the value of R_{ack} decreases, the routing overhead reduces dramatically. Therefore, R_{ack} in the 2ACK scheme provides an effective "knob" to tune routing overhead.

Comparing Figs. 6.4 and 6.2, we have the following observations on routing overhead: as

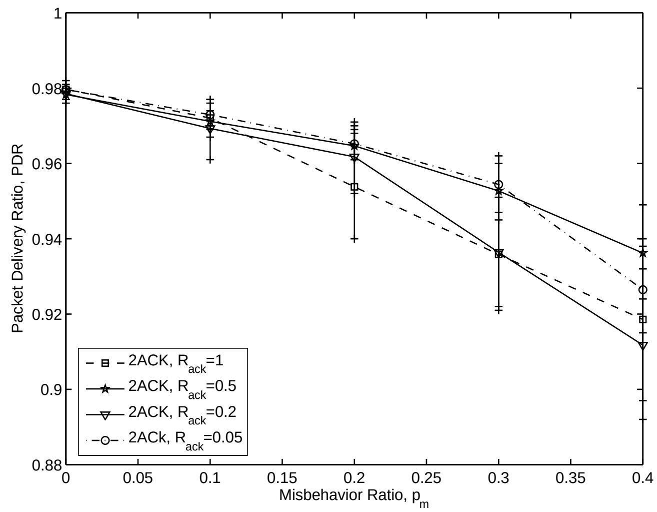

Figure 6.3: Packet Delivery Ratio of 2ACK for different R_{ack}

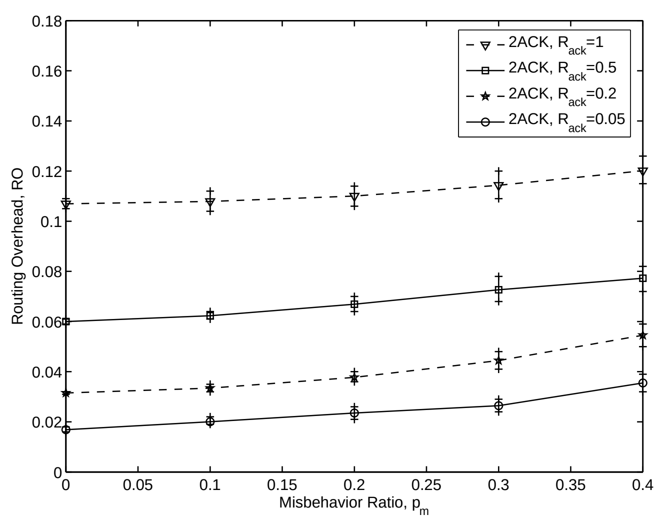

Figure 6.4: Routing Overhead of 2ACK with different R_{ack}

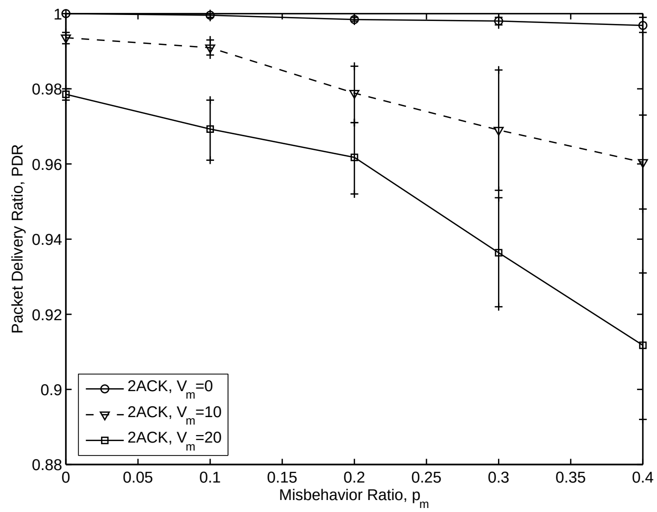

Figure 6.5: The Packet Delivery Ratio of 2ACK for different V_m

R_{ack} decreases, the routing overhead of the 2ACK scheme reduces to a level that is slightly higher than that of the DSR scheme but cannot be lowered further. This can be explained by the additional route discovery processes initiated by the sources receiving the misbehavior reports. The DSR scheme does not initialize such new route discovery processes (note that these simulations were based on UDP traffic).

In Fig. 6.5, we present the packet delivery ratio of the 2ACK scheme as a function of misbehavior ratios p_m with different maximum speeds V_m. We can observe that the packet delivery ratio reduces when mobility increases, regardless of p_m. There are two possible reasons causing PDR to decrease: packets being dropped due to node mobility and false alarms in the 2ACK scheme. We investigate the false alarm issue in Fig. 6.6.

In Fig. 6.6, we show the number of false alarms as a function of *timeout* value, τ, for different maximum speeds V_m. It can be observed that the number of false alarms reduces as *timeout* increases. The number of false alarms increases when the nodes move more rapidly. This is due to the fact that routes are broken more frequently in a high mobility network, and, in some rare cases, the 2ACK scheme may treat such broken routes as misbehaving. The results reveal the appropriate values for *timeout*, τ. Based on the results, $\tau = 0.1 - 0.15$

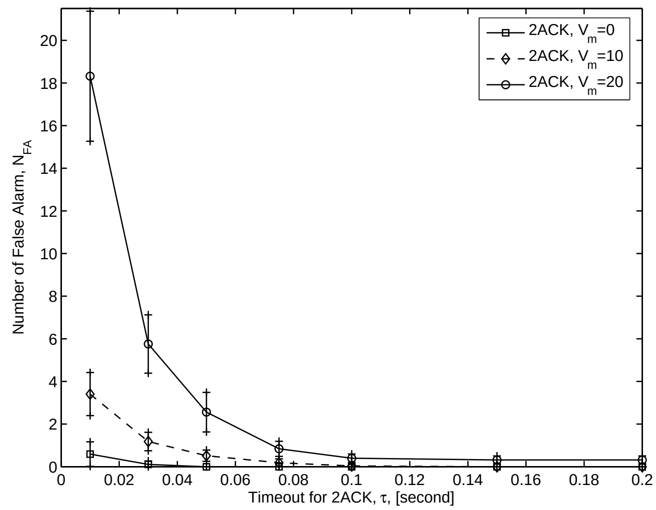

Figure 6.6: Number of False Alarms in 2ACK ($p_m = 0$)

Table 6.1: The relative throughput supported by 2ACK and DSR for TCP traffic

p_m	0	0.1	0.2	0.3	0.4
2ACK	0.963	0.884	0.777	0.697	0.614
DSR	0.974	0.783	0.606	0.513	0.472

seconds works well in networks with $V_m \leq 20$ m/sec.

6.4 Simulation Results for TCP Traffic

In Fig. 6.7, we compare the PDR value of the 2ACK scheme and the regular DSR scheme for TCP sessions. Relatively close PDR values for both schemes can be observed. This is expected, as the senders of the TCP sessions slow down or even stop their transmissions when the acknowledgments from the destination are missing. Comparing the results in Figs. 6.7 and 6.1 or 6.5, we can see that the 2ACK scheme supports slightly higher PDR for the TCP traffic than for the UDP traffic. This is due to the additional acknowledgment and route selection performed in the TCP protocol.

In Table 6.1, we present the relative throughput, normalized number of packets received, when the 2ACK scheme and the DSR scheme are used. Based on Table 6.1, the relative

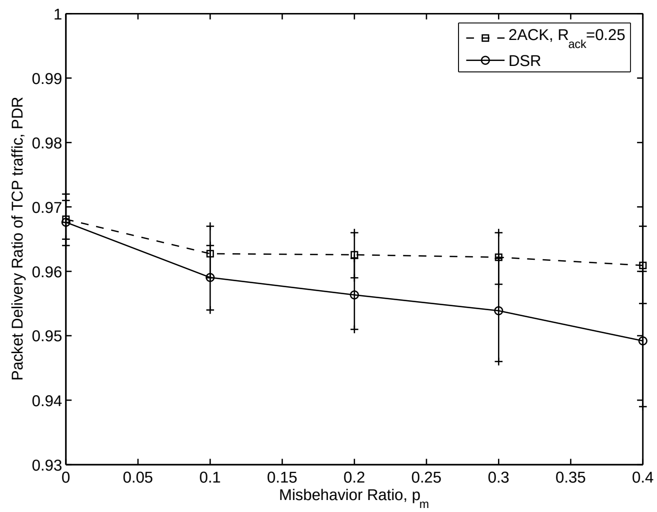

Figure 6.7: Packet Delivery Ratio of 2ACK & DSR for TCP traffic (V_m=20 m/sec)

throughput reduces when p_m increases, due to higher chances of using routes with misbehaving links and longer time being spent to switch to good routes. From the table, we can observe that the 2ACK scheme outperforms the DSR scheme in terms of relative throughput, especially in the networks with larger p_m. For instance, when $p_m = 0.4$, the 2ACK scheme is able to support a relative throughput of 0.614 but the DSR scheme can only support 0.472. The relative throughput of the 2ACK scheme is slightly lower than that of the DSR scheme at $p_m = 0$. This is due to the false alarm reports in the 2ACK scheme in a high mobility network, as shown in Fig. 6.6.

Note that comparisons cannot be made directly between the values in Fig. 6.1 and the numbers in Table 6.1. The former represents packet delivery ratio (PDR); the latter represents the total number of packets that are received (normalized over a fixed number, the average number of packets transmitted).

Chapter 7

Conclusions

Mobile Ad Hoc Networks (MANETs) have been an area for active research over the past few years, due to their potentially widespread application in military and civilian communications. Such a network is highly dependent on the cooperation of all its members to perform networking functions. This makes it highly vulnerable to selfish nodes. One such misbehavior is related to routing. When such misbehaving nodes participate in the Route Discovery phase but refuse to forward the data packets, routing performance may be degraded severely.

In this thesis, we have investigated the performance degradation caused by such selfish (misbehaving) nodes in MANETs. We have proposed and evaluated a technique, termed 2ACK, to detect and mitigate the effect of such routing misbehavior. The 2ACK technique is based on a simple 2-hop acknowledgment packet that is sent back by the receiver of the next-hop link. Compared with other approaches to combat the problem, such as the overhearing technique, the 2ACK scheme overcomes several problems including ambiguous collisions, receiver collisions, and limited transmission powers. The 2ACK scheme can be used as an add-on technique to routing protocols such as DSR in MANETs.

We have presented the 2ACK scheme in detail and discussed different aspects of the 2ACK scheme. Extensive simulations of the 2ACK scheme have been performed to evaluate its performance. Our simulation results show that the 2ACK scheme maintains up to 91% packet delivery ratio even when there are 40% misbehaving nodes in the MANETs that we have studied. The regular DSR scheme can only offer a packet delivery ratio of 40%. The false alarm rate and routing overhead of the 2ACK scheme are investigated as well. One advantage of the 2ACK scheme is its flexibility to control overhead with the use of the R_{ack} parameter.

in this thesis, we have focused only on link misbehavior. It is more difficult to decide the behavior of a single node. This is mainly due to the fact that communication takes place between two nodes, and is not the sole effort of a single node. Therefore, care must be taken before punishing any node associated with the misbehaving links. When a link misbehaves, either of the two nodes associated with the link may be misbehaving. In order to decide the behavior of a node and punish it, we may need to check the behavior of links around that node. This is a potential direction for our future work.

The 2ACK scheme has been implemented on top of DSR. It is also possible to implement the 2ACK scheme over other routing schemes. The main challenge is how to derive the triplet information so that the 2ACK sender and the observing node are informed of such information. Knowledge of topology of the 2-hop neighborhood may be used. In addition, the 2ACK scheme can only work in managed MANETs (as compared to open MANETs). The main reason is that the parameters such as R_{ack} and R_{mis} need to be set. In our future work, we will investigate how to add the 2ACK scheme to other types of routing schemes and open networks. Theoretical analysis of the performance gain of the 2ACK scheme is of interest as well.

Appendix: Pseudo Code of the 2ACK Scheme

We use the triplet of $N_1 \rightarrow N_2 \rightarrow N_3$ in Fig. 5.1 as an example to illustrate 2ACK's pseudo code. Note that such codes are run on each of the sender/receiver of the 2ACK packets.

A. 2ACK packet sender side (node N_3)

```
publish h_n    // Send authenticated element to node N_1
C_pkts ← 0, C_ack ← 0, i ← n    // Initialization at node N_3
while true do
    if (data packet received) then
        C_pkts ++    // Increase the counter of received packets
        if (C_ack/C_pkts < R_ack) then    // The data packet needs to be acknowledged
            prepare MAC with h_{i-1}
            prepare 2ACK with ID, MAC, h_i    // Add authentication to 2ACK packet
            send 2ACK
            C_ack ++, i --    // Increase the counter of acknowledged packets
        end
    end
end
```

B. Receiver (Observer) side (node N_1)

Parallel process 1 (receiving h_n)

while true **do**
  if receive h_n from the 2ACK packet sender **then**
    record h_n, $i \leftarrow n$
  end
end

Parallel process 2 (Forwarding the data packet to N_2)

while true **do**
  randomly select $T_{start} >$ current time // Start the observation
  while current time $< T_{start}$ **do**
    // null
  end
  LIST$\leftarrow \phi$, $C_{pkts} \leftarrow 0$, $C_{mis} \leftarrow 0$ // Initialization at node N_1
  while current time $< T_{start} + T_{obs}$ **do** // Observation period is not expired
    if (data packet forwarded) **then**
      LIST $\leftarrow$ LIST $\cup$ data ID // Add a data ID to LIST
      C_{pkts} ++ // Increase the counter of forwarded packets
      setup timer (τ) for data ID // Record the time
    end

Parallel process 3 (receiving 2ACK packets from N_3)

if (2ACK packet received) **then**
search data ID carried by 2ACK from LIST
if (found) **then** // A 2ACK packet for a data ID received
check validity of h_i
LIST $\leftarrow$ LIST - data ID // Remove data ID from LIST
clear timer for ID
end
end
if (timeout event happens) **then** // 2ACK packet for a data ID is not received
LIST $\leftarrow$ LIST - data ID // Remove data ID from LIST
C_{mis} ++ // Increase misbehavior counter
end
end
if ($C_{mis}/C_{pkts} > R_{mis}$) **then** // The observation period expires
send link misbehavior report
end
end

Bibliography

[1] The network simulator(ns-2). http://www.isi.edu/nsnam/ns/.

[2] I. Aad, J-P. Hubaux, and E-W. Knightly. Denial of service resilience in ad hoc networks. In *IEEE/ACM Mobicom*, 2004.

[3] B. Awerbuch, D. Holmer, C-N. Rotaru, and H. Rubens. An on-demand secure routing protocol resilient to byzantine failures. In *ACM Workshop on Wireless Security (WiSe)*, September 2002.

[4] K. Balakrishnan, J. Deng, and P. K. Varshney. TWOACK: preventing selfishness in mobile ad hoc networks. In *Proc. of IEEE Wireless Communications and Networking Conference(WCNC)*, New Orleans, LA, March 2005. IEEE.

[5] J. Y. Le Boudec and M. Vojnović. Perfect simulation and stationarity of a class of mobility models. In *Proc. of Infocom'05*, Miami, FL, USA, March 13-17 2005.

[6] S. Buchegger and J-Y. Le Boudec. Performance analysis of the CONFIDANT protocol: Cooperation of nodes, fairness in dynamic ad-hoc networks. In *Proc. of MobiHoc'02*, June 2002.

[7] L. Buttyan and J-P. Hubaux. Security and cooperation in wireless networks. available at http://secowinet.epfl.ch/.

[8] L. Buttyan and J-P. Hubaux. Enforcing service availability in mobile ad-hoc WANs. In *Proc. of First IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Boston, MA, USA, August 2000.

[9] L. Buttyan and J.-P. Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *ACM/Kluwer Mobile Networks and Applications*, 8(5), 2003.

[10] M. Conti, E. Gregori, and G. Maselli. Towards reliable forwarding for ad hoc networks. In *Proc. of Personal Wireless Communications (PWC '03)*, September 2003.

[11] D. Eastlake and P. Jones. RFC 3174 - US secure hash algorithm 1 (SHA1). Technical report, Motorola and Cisco Systems, September 2001.

[12] L. M. Feeney and M. Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *IEEE INFOCOM*, 2001.

[13] Y. 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.

[14] Y. Hu, A. Perrig, and D. B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *Proc. of the Eighth ACM Annual International Conference on Mobile Computing and Networking (MobiCom'02)*, September 2002.

[15] J-P. Hubaux, T. Gross, J-Y. Le Boudec, and M. Vetterli. Toward self-organized mobile ad hoc networks: The terminodes project. In *IEEE Communications Magazine*, January 2001.

[16] P. Jacquet, P. Mhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol. In *IEEE INMIC'01, 28-30 December 2001, Lahore, Pakistan*, pages 62–68, December 2001.

[17] M. Jakobsson, J-P. Hubaux, and L. Buttyan. A micropayment scheme encouraging collaboration in multi-hop cellular networks. In *Proc. of Financial Crypto 2003*, January 2003.

[18] D. Johnson, D. Maltz, Y. C. Hu, and J. Jetcheva. The dynamic source routing protocol for mobile ad hoc networks (DSR). Internet-draft, February 2002.

[19] D. B. Johnson. Future resiliency and high security systems. ECC, March 1999. available from www.certicom.com.

[20] J. Jubin and J-D. Tornow. The darpa packet radio network protocols. In *Proceedings of the IEEE*, pages 21–32, January 1987. 75(1).

[21] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang. Providing robust and ubiquitous security support for mobile ad-hoc networks. In *Proc. of IEEE International Conference on Network Protocols (ICNP'01)*, Riverside, CA, USA, 2001.

[22] L. Lamport. Password authentication with insecure communication. *Proc. of Communications of ACM*, November 1981. 24(11), 770-772.

[23] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Sustaining cooperation in multi-hop wireless networks. In *Proc. of the 2nd Symposium on Networked Systems Design and Implementation*, April 2005.

[24] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proc. of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom)*, August 2000.

[25] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. RFC 2018 - TCP selective acknowledgement options. Technical report, PSC, LBNL, Sun Microsystems, October 1996.

[26] P. Michiardi and R. Molva. Simulation-based analysis of security exposures in mobile ad hoc networks. In *Proc. of European Wireless Conference*, February 2002.

[27] H. Miranda and L. Rodrigues. Preventing selfishness in open mobile ad hoc networks. In *Proc. of the Seventh CaberNet Radicals Workshop*, October 2002.

[28] S. Murthy and J-J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *ACM/Baltzer Journal on Mobile Networks and Applications, Special Issue on Routing in Mobile Communication Networks*, 1(2):183–197, October 1996.

[29] V-N. Padmanabhan and D-R. Simon. Secure traceroute to detect faulty or malicious routing. *SIGCOMM Computor Communication Review*, 33(1), January 2003.

[30] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (AODV) routing. Internet-draft, July 2003.

[31] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *Proc. of ACM Special Interest Group on Data Communications(SIG-COMM)'94*, pages 234–244, August 1994.

[32] R. L. Rivest. RFC 1321 - the MD5 message-digest algorithm. Technical report, MIT Laboratory for Computer Science and RSA Data Security, Inc., April 1992.

[33] V. Srinivasan, P. Nuggehalli, C. F. Chiasserini, and R. R. Rao. Cooperation in wireless ad hoc networks. In *Proc. of Infocom'03*, San Francisco, CA, USA, March 30 - April 3 2003.

[34] F. Stajano and R. Anderson. The resurrecting duckling: Security issues in ad-hoc wireless networks. In Bruce Christianson, Bruno Crispo, and Mike Roe, editors, *Security Protocols, 7th International Workshop Proceedings*. Springer-Verlag, 1999.

[35] Y. Xue and K. Nahrstedt. Providing fault-tolerant ad-hoc routing service in adversarial environments. *Wireless Personal Communications, Special Issue on Security for Next Generation Communications, Kluwer Academic Publishers*, 29(3-4):367–388, 2004.

[36] S. Zhong, J. Chen, and Y. R. Yang. Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks. In *Proc. of Infocom'03*, San Francisco, CA, USA, March 30 - April 3 2003.

[37] L. Zhou and Z. J. Haas. Securing ad hoc networks. *IEEE Network Magazine*, 13(6), November/December 1999.

Vita

Kejun Liu was born in Shanxi province, P.R.China. She recieved her B.S. degree in Computer Communication from Tongji University in Shanghai, in 1999. Then she moved to Beijing to work in Beijing Hangyu System Institute (CASC-BHSI) from 1999 to 2002, as a Network/System Administrator (Jul.1999 - Oct.2001) and Network/System Manager (Nov.2001 - Dec.2002) respectively. She has started her study for Master's Degree in Computer Science department of University of New Orleans since 2004. Her research focuses on Mobile Ad Hoc Networks and wirless network security. Her Email is kliu@uno.edu.