DESIGN OF HIERARCHICAL INTRUSION DETECTION UNIT FOR AD-HOC NETWORKS BASED ON BAYESIAN NETWORKS

By

PRANAV KULKARNI

Bachelor of Science in Information Technology

Shivaji University,

Kolhapur, India

2003

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 2008

DESIGN OF HIERARCHICAL INTRUSION

DETECTION UNIT FOR AD-HOC NETWORKS BASED

ON BAYESIAN NETWORKS

Thesis Approved:

Dr. J. P. Thomas

Thesis Adviser

Dr. G. P. Chandler

Dr. G. E. Hedrick

Dr. A. Gordon Emslie

Dean of the Graduate College

ACKNOWLEDGMENT

I am greatly indebted to my parents for their moral and financial support to complete my Masters degree in Computer Science. Many people have ambitions in life but only few are fortunate to fulfill them. In this regard I am greatly thankful to the almighty for guiding me in the right direction and rewarding my efforts by fulfilling my ambition.

At this moment I must mention some important people who have always been a pillar to the successful completion of my master's program. Firstly, I would like to thank my thesis advisor Dr. Johnson P. Thomas for his valuable time and patience in guiding and supporting me with his ideas and suggestions. Secondly, my heartfelt thanks to my committee members, Dr. G. E. Hedrick and Dr. J. P. Chandler, for their comments and suggestions about my thesis.

I would like to thank my friends for having encouraged me during the entire Masters Degree program. And last but not least; I would like to thank my wife Asmita for being with me all the time and supporting me throughout my Masters' degree.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

1.1 Mobile Ad-hoc Networks

A Mobile Ad-hoc NETwork (MANET) is a self-configuring network of mobile nodes that include routers and hosts connected by wireless links. The union of these nodes forms the arbitrary configuration of the network topology. There is no backbone for such a topology. Wireless nodes in the network act as routers and deliver data and control messages between nodes. If node A cannot directly send a message to node B, nodes within range of node A will act as a router for that node and forward the packet to the destination. The routers are free to move randomly and organize themselves arbitrarily; the network topology changes rapidly and unpredictably. MANET may operate in a standalone fashion as for the small organization or it may be connected to a larger wired network like the Internet. This arbitrary network topology gives immense power to the working of MANETs. These networks can be deployed wherever there is no proper backbone for the network. Nodes of MANET can work together and carry out the task without the other's help.

MANETS are immensely useful in scenarios where it is not easy to establish access points or base receivers and where we need to update the information quickly and where maximum mobility is required. For example, in video conferencing, users participating in the conference may be geographically distributed. Some of them may not be having dedicated wired connections. MANETS come handy in such situations. Other

examples include military battlefields, disaster relief operations in remote areas, home networking as well as for enforcement of law. MANETS can be diagrammatically represented as follows.

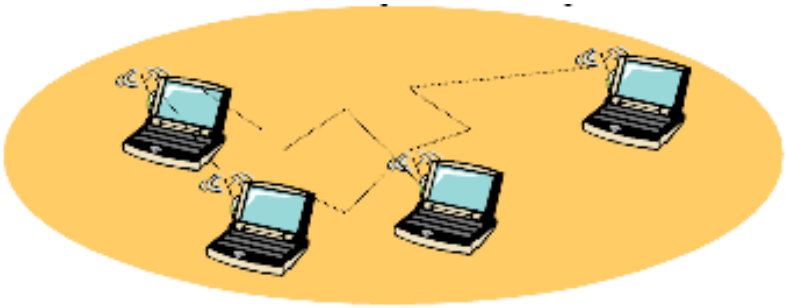

Figure 1.1: MANETS

Even though, these networks are very flexible and useful in day to day life, there are many factors that hinder the development of these networks. One of the main factors of concern is security implementation. Users of any networking service demand "anywhere, anytime" services. Even if there is a small discontinuation in the service, users would lose valuable information. To provide these continued services, the network must be free of physical channel errors and it should also prevent break-ins by adversaries. Wireless networks like MANETs are susceptible to various types of attacks that include snooping to passive eavesdropping to active interference and DOS attacks.

Unlike their wired counterparts, Mobile Ad Hoc Networks don't have firewall or central servers where all the intrusion can be blocked. Hence, every node should be equipped to deal with all kinds of attacks. In order to ensure security, each node in the network is authenticated using cryptographic techniques supported by a certificate authority (CA). However as explained earlier, MANETS have a dynamic configuration

and a centralized Certificate Authority cannot be deployed in the network. MANETS operate as a decentralized model and it is essential to establish trust among the nodes of the network. A good trust model should be scalable as well as robust.

Even if trust is established between nodes of the network, a node in the network may turn malicious. This node can misroute packets or flood the network with bogus packets. This node can also change the information contents of content sensitive packets. In order to detect this kind of malicious nodes, there should be a mechanism for intrusion detection along with trust establishment and secured routing.

In this thesis we propose a hierarchical intrusion detection system based on Bayesian networks. However, a node that detects an intrusion may not be trustworthy and may therefore report false intrusions or report intrusions that have not occurred. Hence, in our approach the intrusion detection system is integrated with a trust mechanism. Both the intrusion detection system and the trust mechanism are Bayesian based. In [1], it is shown that Bayesian Networks can be used efficiently to establish trust among nodes of MANETS. By combining the trust model with the Intrusion Detection Unit, we study the effects of intruders on MANETs and how nodes in MANETs efficiently handle such kinds of attacks.

In chapter 2, we review previous work in the area of secure communication in MANETS. In chapter 3, the problem is defined formally. In chapter 4, a solution is proposed for the problem defined in Chapter 3. Chapter 5 discusses the simulation setup and presents results and graphs for the experiments conducted and chapter 6 concludes the thesis.

CHAPTER 2

REVIEW OF LITERATURE

2.1 Routing Protocols for MANET:

The current MAC protocols for Wireless Ad Hoc Networks are all vulnerable. There are many protocols for routing packets between nodes in MANETs. There are no dedicated routers in mobile ad hoc networks. Each node acts as a router and transmits packets between source and destination. The node within the range of the source node and is not the destination node, accepts the packet sent by the source and forwards it along the path to the destination node. In a contention based method, each node must compete for control of the transmission channel each time it sends the packet. In order to avoid collisions, a node must adhere to strict rules and conditions. In a contention free method, each node must gain access to the transmission medium with the consent of other nodes in the network. Regardless of contention based or contention free protocols, if any node is malicious, it can easily break the network by consuming packets or misrouting them thus breaking the whole protocol. In wired networks, there is reduced possibility of such kinds of attacks as networks are protected by gateways and firewalls. However, in wireless networks, due to the lack of a centralized authority, firewalls and gateways cannot be deployed. Thus in wireless networks, every node is vulnerable to attacks in the open wireless medium.

To conquer this problem, many wireless routing protocols are proposed that solve the problems mentioned above. The important routing protocols for ad hoc Wireless Networks include, DSDV, DSR, AODV, TORA etc.[5,6]. All these protocols assume that there is a trust already established among participating nodes. Many of the security threats lie in this assumption. A brief description of these routing protocols is given below.

Routing protocols are classified into 3 broad categories. Those are Proactive protocols, Reactive protocols and Hybrid protocols. Proactive protocols maintain routes to all nodes, including nodes to which no packets are sent. They adjust well to topology changes. Reactive protocols consider the demand for data transmission. Routes between hosts are determined only when they are needed. This can significantly reduce routing overhead. Hybrid protocols combine both proactive and reactive protocol properties to come up with a better routing protocol for efficient packet routing. [16]. Examples of proactive routing protocols include DSDV (Destination Sequenced Distance Vector). Popular reactive protocols are DSR (Dynamic Source Routing) and AODV (Ad-hoc On-demand Distance Vector). TORA (Temporally Ordered Routing Algorithm) has the advantages of both types of routing protocols. It is hybrid routing protocol.[6]

2.2 Types of attacks on MANETS:

The lack of conventional authentication and identification mechanisms in Ad hoc Wireless Networks make these networks very vulnerable. These vulnerabilities can be easily exploited by malicious nodes, thus compromising the security and integrity of the network. The routing protocols used for MANETs including those discussed above,

provide little or no authentication or security. There are many ways in which malicious node can attack other nodes in a MANET or the MANET as a whole. There are various types of attacks that a malicious node can employ in order to compromise security and integrity of MANETs.

As discussed earlier, routing protocols for MANETs are highly susceptible to security breaches as they assume the working of wireless ad hoc networks in non-hostile environments. Encryption and establishment of trust among nodes of network try to address this problem [SAODV] [7]. However, this can not completely eliminate the possibility of intrusion.

A malicious node can employ various types of attacks on a wireless ad-hoc network that can range from active interference to passive eavesdropping. The malicious node may also act maliciously intermittently making it more difficult to detect.

The important goals of secured ad hoc network are Authentication, non-repudiation, availability, integrity and privacy. The following figure summarizes the attacks that can take place in wireless ad-hoc networks.[4]

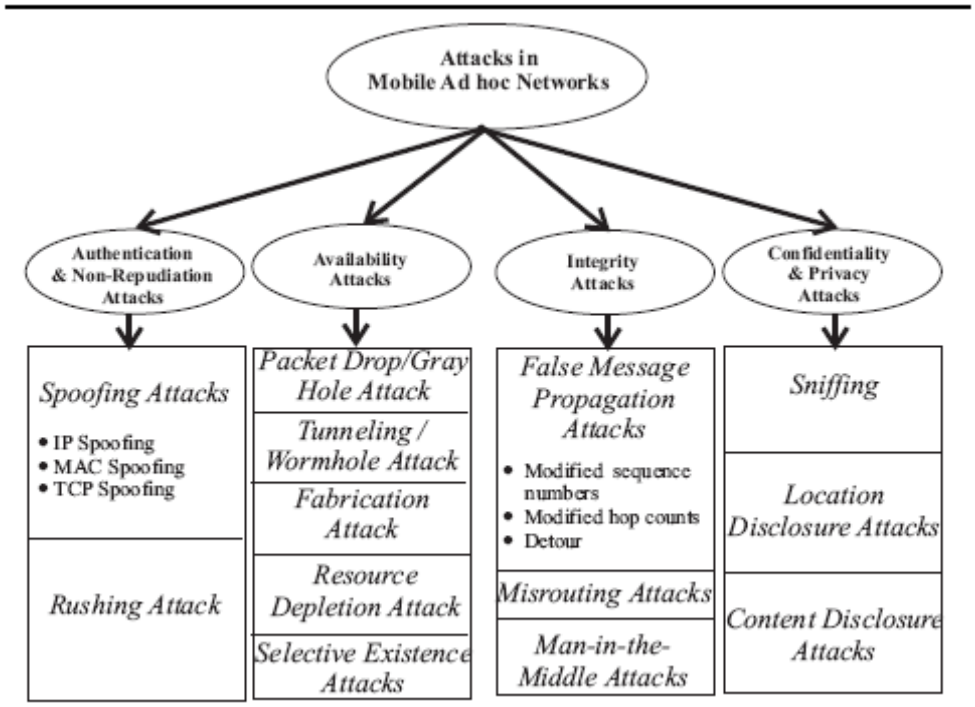

Figure 2.1: Classification of Attacks [3]

As seen from the figure, attacks on MANETs can be classified as follows.

1. Authentication and non- repudiation attacks.
2. Availability attacks.
3. Integrity attacks.
4. Confidentiality and privacy attacks.

Authentication and no-repudiation attacks can be further classified as spoofing attacks that include IP spoofing, MAC spoofing and/or TCP spoofing and Rushing attack. Availability attacks mainly consists of Gray hole attacks, Wormhole attacks, Fabrication attacks and resource exhaustion attacks. False message propagation attacks, misrouting attacks and Man-in-the-Middle attack come under integrity attacks. In confidentiality and privacy attacks; sniffing, location disclosure and content disclosure attacks are prevalent.

Authentication and non-repudiation attacks: Authentication can be defined as verifying the identity of one entity against given credentials. For example, a node in MANET can be authenticated with the help of its shared secret key or its MAC address. Authentication allows a node to identify itself in a network with the other node and communicate with the other node.

Non repudiation is the ability to prove that the sender did actually send the message. Almost all routing protocols for mobile ad hoc networks use either MAC or IP address to authenticate nodes in the network. Therefore, the very first step for an intruder to attack ad hoc wireless network is to spoof either the IP address or the MAC address of any node in the network and gain access to network resources. Spoofing attacks are the simplest method to compromise the authentication of the network.

For protocols that use the suppression function for duplicate packets (for example TCP/IP), rushing attacks can be a non trivial possibility. In this type of attack, malicious node sends spurious packet to a destination before the original packet reaches the destination. This makes the original packet look like a duplicate packet and the receiving node eventually drops the original packet. The damage caused by this attack is protocol dependent. The UDP protocol is mode prone to such kinds of attacks while TCP is less prone to a rushing attack.

Availability attack: Availability states that network resources are available to authorized entities in the network whenever they want to use those resources. Availability guarantees that network services are accessible to nodes in the network in a timely manner. These types of attacks are also known as “Denial of Service (DOS)” attacks. They are used to deny or reduce the availability of network services. This is also known

as a resource consumption attack. An attacker might try to consume resources by selecting dropping of packets that result in increased number of route requests from neighbor nodes that have limited routing capabilities. Packets dropped can either be data packets or control packets.

In fabrication attacks, a malicious node can masquerade as a node along the path between a source node and a destination node and it continually discards packets coming from the source node or sends route error messages to the destination node. When a node receives a route error message, it will delete the route table entry for that destination node. If the alternate path doesn't exist between the source node and the destination node, then destination node will be thrown out of the network.

In the tunneling attack, two malicious nodes pretend to be directly adjacent to each other with the help of tunneling. One node builds the message and sends it to another malicious node through intermediate nodes and falsely claims that there is a direct link with the destination node. In this way, malicious nodes can force the traffic through them. In the wormhole attack, a malicious node listens to a message in one part of the network and replays it in another part of the network with the help of another colluding malicious node. This type of attack can be classified under aggregated attacks which involve more than one malicious node.

If two malicious nodes transfer a heavy volume of data between them thereby blocking the network transmission bandwidth, other nodes will be deprived of precious network bandwidth. This type of attack is termed as resource depletion attack. Other nodes in the network are depleted of the available network bandwidth thus affecting the

network throughput. A resource depletion attack can also be carried out using control messages.

In a selective existence attack, a malicious node behaves selfishly for reasons such as saving his battery or using the network only for his own profit. A malicious node may not participate in any route discovery process or it may not send any control message to other nodes but it will only update its routing information so that it can correctly route packets. This will make that node an invisible node that consumes precious network resources. Although this type of attack can be harmless, it still can block network resources and if the node is snooping or modifying content of packets, it will be difficult to detect.

Integrity Attacks: Integrity guarantees that a message is not altered on its path to the destination. These types of attacks pose the more serious problems for MANETs. Various types of integrity attacks are identified until now. Some of them are discussed here.

False message propagation attack is a kind of integrity attack in which a malicious node advertises a route to the node with a destination sequence number greater than the authentic value. By doing this, it diverts the traffic towards the attacker because the nodes will select the route reply message with the highest destination sequence number. Only AODV is susceptible to this kind of attack since it uses the sequence number to find the route. A single node can carry out this attack. It is also known as a forging sequence number attack. A similar attack is possible by forging the hop count.

In misrouting attack, the intruding node tries to send a data packet to a wrong node thus totally misrouting the packet. This type of attack can be dealt with snoopy intrusion detection units.

Man-in-the-Middle attack can be launched by any node that is in the routing path. It can be considered as "invisible node attack". Here, the attacker issues fake route request and route reply messages to spoil other node's routing table. The damage caused by this attack is only limited to the routing path on which the malicious node is present.

Confidentiality and privacy attacks: Privacy is termed as retaining the personal information stored at the node and not revealing it any other node in the network. On the same note, confidentiality ensures that certain information be revealed only to authorized nodes in the network. There are two main types of attacks that can compromise confidentiality and security. Those are Location disclosure attacks and Content disclosure attacks.

The location disclosure attack, as the name suggests, reveals physical location of a particular node in the network. This attack is based on the principle that for all multihop routing protocols any two consecutive nodes in a route must be geographically close.

The content disclosure attack enables the malicious user to get the contents of the messages those are transmitted in the network. If the message contains important information, this attack can pose a dangerous threat to the privacy of the network as a whole. Sometimes messages are secured using encryptions such as RSA. If an attacker can break the encryption, then he can easily access the payload field of the IP packet and disclose the information contained in the packet.

2.3 Intrusion Detection in MANETs:

Any node in the wireless network should operate in a node that trusts no peer. Current MAC protocols for Ad Hoc Networks are all vulnerable for security attacks such as DOS. Intrusion prevention measures can be employed in Ad Hoc networks to reduce the threat of intrusion. However, this alone can not completely eliminate them. Most of the intrusion detection systems proposed so far are based on the AODV and DSR protocols. Some IDSs like WatchDog[2,3] are very effective for intrusion detection but they still suffer from some drawbacks like dependency on AODV, a node traveling out of range and false positives.

Intrusion detection systems for MANETs:

According to [17], there are three basic architectures of intrusion detection systems for MANETs.

1. Stand-alone intrusion detection systems
2. Distributed and cooperated intrusion detection systems.
3. Hierarchical intrusion detection systems.

Stand-alone Intrusion Detection Systems: As the name suggests, each node in the network is responsible for detecting intrusion in the network. Nodes do not communicate with each other to detect an intrusion. Each node decides by itself about the possible intrusion. In a network where there is no issue of battery life for nodes, this model would work well. In networks where battery life of node is limited, this model would be redundant.

Distributed and Cooperative Intrusion Detection Systems: In this type of intrusion detection systems, nodes in the network cooperate with each other to detect an intrusion.

Nodes exchange different control messages or information related to intrusion detection among them. In this model also, each node runs the intrusion detection algorithm and sends the information collected to neighboring nodes in the network. This model is useful in detecting intrusion which involves coordinated intrusion on the network. Coordinated intrusion means intrusion which involves coordination of more than one node.

Hierarchical Intrusion Detection Systems: Hierarchical intrusion detection is basically a cooperated intrusion detection scheme. The only difference is the network is divided into clusters and there is a definitive hierarchy in the network. Generally, clusterheads of each cluster participate in intrusion detection. Only clusterheads need to run the intrusion detection algorithm instead of each node in the network. This helps nodes in the network conserve the battery.

Examples of intrusion detection systems in MANETs:

Distributed and cooperative IDS: Zhang and Lee proposed the model for a distributed and cooperative IDS[18]. The model for an IDS agent is structured into six modules. The *local data collection* module collects real-time audit data, which includes system and user activities within its radio range. This collected data will be analyzed by the *local detection engine* module for evidence of anomalies. If an anomaly is detected with strong evidence, the IDS agent can determine independently that the system is under attack and initiate a response through the *local response* module (i.e., alerting the local user) or the *global response* module (i.e., deciding on an action), depending on the type of intrusion, the type of network protocols and applications, and the certainty of the evidence. If an anomaly is detected with weak or inconclusive evidence, the IDS agent can request the

cooperation of neighboring IDS agents through a *cooperative detection engine* module, which communicates to other agents through a *secure communication* module.

Local Intrusion Detection System (LIDS): Albers *et al.* [19] proposed a distributed and collaborative architecture of IDS by using mobile agents. A Local Intrusion Detection System (LIDS) is implemented on every node for local concern, which can be extended for global concern by cooperating with other LIDS. Two types of data are ex-changed among LIDS: security data (to obtain complementary information from collaborating nodes) and intrusion alerts (to inform others of locally detected intrusion). In order to analyze the possible intrusion, data must be obtained from what the LIDS detects, along with additional information from other nodes.

CHAPTER 3

PROBLEM DEFINITION

There are many models and schemes proposed for Mobile Ad Hoc Networks that deal with security implementation and support. Some of them facilitate security support using shared secret key mechanism and certificate revocation schemes [13]. Routing protocols designed in these schemes take care of the certificate issuance to nodes participating in the network and using public and secret keys they implement secure communication. All of these schemes rely on some intrusion detection mechanism. Based on the knowledge about the intrusion detected at a node, these schemes may revoke the certificate. There are a few solutions proposed for intrusion detection in MANETS. Most of them are based on the AODV routing protocol for MANETS [2]. To date, the most efficient security scheme for MANETs uses sharing of secret key for certificate revocation and issuance. However, such an approach is expensive for MANETS due to the limited resources in MANETs.

As far as we are aware, the role of trust in intrusion detection has not been investigated. Trust can be defined as reliance on the integrity, strength, ability, surety etc. of a thing or entity. In the context of communicating wireless nodes, trust can be defined as the reliance on the integrity and authenticity of the node. In normal intrusion detection systems, which do not incorporate trust into their intrusion detection, it is very difficult to deduce the correct information based on the information sent by peer nodes. This is because a node that sends out the intrusion information may itself be

compromised or not trustworthy. As opposed to this, if a trustworthy node sends out the intrusion information, it is more likely that the information is correct. So, whenever intrusion detection is coupled with trust, it will give better results for detecting an intrusion.

A malicious node may report an intrusion and accuse another node of the intrusion. This may result in the accused node being ejected from the network. In this thesis we propose an intrusion detection mechanism that takes trust into account. This will result in fewer false accusations and intrusions reported will be genuine.

In [1], the authors propose a trust model for certificate revocation in Ad Hoc Networks based on Bayesian networks. Our proposed Intrusion Detection Unit (IDU) is coupled with the underlying Bayesian network for establishing trust. In this model, trust is built periodically over time.

In [1], the authors describe the IDU to be located in each node. Trust is calculated based on the recommendation of the IDU along with other parameters. The trust model they propose is as follows [1].

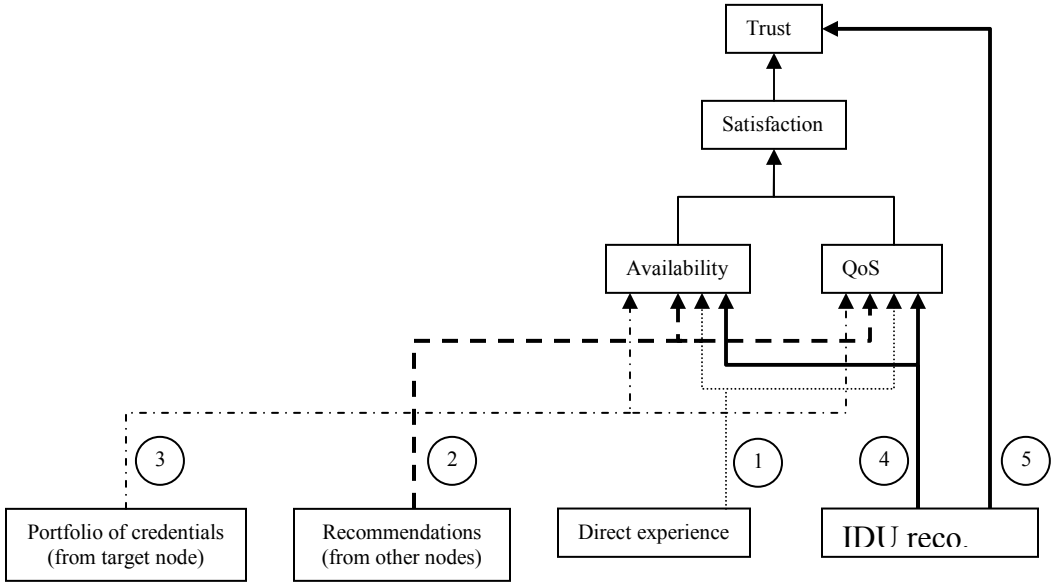

Figure 2.2: Trust Model for MANETS

According to this diagram, if the IDU signals, the trust is immediately set to zero for the particular node. Other than intrusion detection, trust is calculated based upon other factors such as direct experience of previous communication with the node, recommendations for other trustworthy nodes etc.

The IDU works as follows. When a node is in snooping state, it searches for the nearby node within its range. If that snooping node can hear the transmission between two nodes, then that node goes to the monitor state [2,5]. This means that it monitors the traffic between those two nodes. The main aim of this node is to ensure that the node does not alter the packet or the path.

We assume that all nodes can monitor other nodes and be monitored by other nodes as well, as long as they are within communication range.

CHAPTER 4

PROPOSED SOLUTION

In this work we study the following types of attacks:

- DoS attack
- Packet misrouting
- spoofing
- Information Destruction attack

Of the attacks mentioned above, Dos attack can be thought of as an availability attack. Packet misrouting and spoofing is considered as an Integrity attack. These four attacks represent the broad spectrum of security attacks on Ad Hoc networks.

In the proposed solution a node is snooped by one or more nodes within communication range of the current node. Based on the information that the snooping node gets, it builds a table and communicates with every other node except the node that is being snooped. In our approach not all nodes can snoop on the given node. We divide the network into clusters and only cluster-heads can snoop. If the network is large, it would be very difficult to maintain the network and mark out intrusive nodes efficiently. If the network is divided into clusters, we would get the localized information of the particular area of the network. Subsequently, we can easily mark out intrusive nodes from that localized area. Moreover, by allowing cluster-heads to snoop at a given time, the flow of control packets needed for detecting intrusion remains low. Furthermore, by

having only few nodes in the network as snoopers, only those nodes will need the extra computing power as opposed to all the nodes.

In our approach, a cluster-head can only snoop at the nodes within its cluster. Due to mobility, clusters and cluster-heads change and the group of snooping nodes is therefore dynamic. The group of snooping nodes called "Active snoopers" is decided using the Dominating Sets algorithm proposed by Fie Dai and Jai Wu. [15]

The Dominating set algorithm is very useful and efficient in partitioning a network into clusters. The dominating sets approach is explained in detail below. The dominating set algorithm divides the network into clusters and assigns cluster-heads to each cluster. These cluster-heads act as active snoopers in the network and help in detecting an intrusion.

The network life cycle consists of 3 phases:

- Network initialization phase
- Network uptime phase:
- Network shut down phase:

In the network initialization phase, network parameters are set, The Certificate Authority id is announced and all the links and bandwidth made available. When a node wants to join the network, initially it needs to get a certificate from the Certificate Authority and authenticates itself against its credentials. This authentication is carried out before the node joins the network. The certificate authority issues a valid certificate to the node by a checking node's authenticity. This is beyond the scope of this research and it is assumed that the node is authenticated before joining the network by the certificate authority. After the node has been initialized, the Intrusion Detection Unit (IDU) will

boot up along with all other services. It will remain active throughout the lifetime of the node in the network whenever the node is an active snooping node. If the node is not snooping at the given instance of time, it can turn the IDU off to save battery.

In the network shutdown phase, all nodes save the information they collected to the permanent storage, turn their IDUs off, relinquish their authentication credentials and leave the network.

4.1 Network Initialization phase: Formation of Dominating Sets:

A finite set of nodes in the network form a group together that act as "Active Snoopers". To form such a group, the Dominating Set algorithm [14] can be used efficiently. This is a localized algorithm which can quickly converge to a group. In [14] the authors propose a simple algorithm for the formation of Dominating Sets where the dominating set is formed in two phases. In the first phase, called a "marking process", hosts interact with others in the neighborhood. Each host is marked true if it has two "unconnected neighbors". In this context, "unconnected neighbor" means those nodes which are not within communication range of each other. In the second phase, called "pruning process", the set of marked nodes is pruned according to two rules. According to dominant pruning rule 1, a marked node can unmark itself if its neighbor set is covered by another marked node. According to rule 2, a marked host can unmark itself if its neighborhood is covered by two other directly connected marked nodes. By applying rules 1 and rule 2, we get a dominating set that is small and that covers almost all the area on which the network is spread.

The nodes in the dominating set form the clusterheads that is the group of active snooping nodes. Each cluster-head is connected to another cluster-head through the

hierarchical relationship. For example, in the following figure, a solid line node is the cluster-head of a cluster.

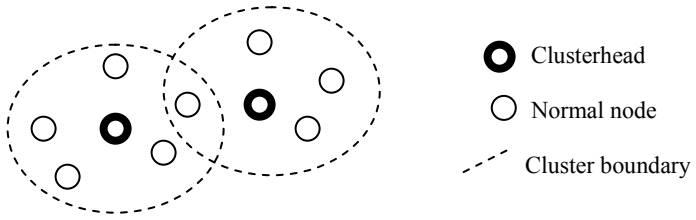

Fig. 4.1: Formation of Clusters and cluster-heads

The clusterhead of the sending node can observe the transmission. However, it cannot observe the data when that data travels out of its cluster boundary. Hence, the clusterhead contacts the neighboring clusterhead and informs it about the incoming data. All clusterheads are interconnected with each other through the dominating set.

Network update phase: Maintaining Dominating sets

The idea of dominating sets and cluster formation fits well for detecting intrusions in Ad Hoc Networks. Nodes in the network can move freely within the network. Each time a node moves out of the cluster, its cluster-head membership needs to be re-assigned again. Whenever a node has moved out of range of the cluster-head, the cluster-head notifies its parent in the tree or the neighboring cluster-head. If the network is divided into multiple levels of cluster-heads, then the child cluster-head will notify its parent cluster-head. Otherwise, the cluster-head will notify its neighboring cluster-head.

The formation of clusters and cluster-heads is a recursive algorithm. Initially, the network is divided into clusters with respective cluster-heads. If in a given cluster, there are more nodes than the cluster-head can manage, then that cluster will again be sub-clustered within the cluster. The new sub-cluster will have a new cluster-head which will

now be the child node of the original cluster-head. This process is iterated until the number of nodes in all the clusters is within a certain threshold.

In the case where the cluster is further divided into subclusters, the cluster-head of each subcluster will be member of a cluster with it own cluster-head as parent. This is shown below.

There will not be a single root to the tree at any given time if the threshold for the number of nodes in a cluster is arranged properly. In other words, there will not be a single point of failure in the network at any given point in time. There is also a possibility that two nodes would be present in two clusters at the same level if they are boundary nodes. In that case, both cluster-heads will snoop on those nodes.

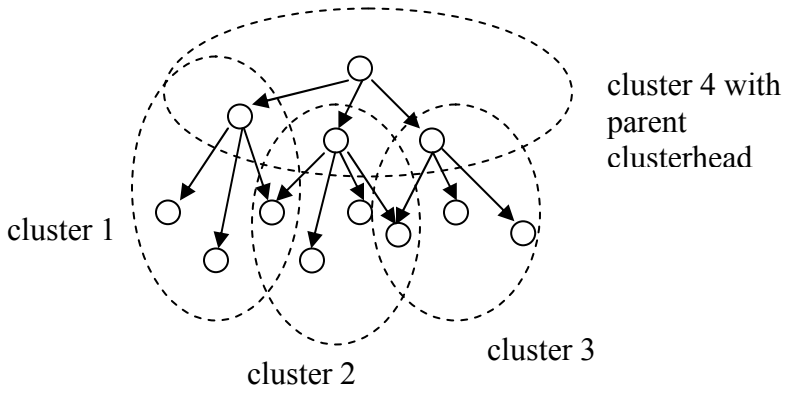

Fig. 4.2: Formation of sub-clusters and cluster-heads

If a child cluster-head notifies the parent about possible node movement, the parent in turn, notifies neighboring cluster-heads about the possible arrival of the node. If a node arrives in another cluster, the head of that cluster associates a relationship with that node and notifies the parent.

The parent in turn notifies the previous head about the ownership change. Also, there might be a node in the parent cluster which is not a cluster-head. That node is treated as a regular node in any other cluster for this scheme.

Although this solution given above takes care of the mobility of nodes, it poses another problem. If all nodes move to one part of the network, then that part will be overcrowded and difficult to maintain while another part will be empty resulting in a network partition. To deal with this problem, a threshold value is set for each cluster. This threshold measures the number of nodes in the cluster that cluster-head can handle. If the number of nodes in the cluster becomes greater than the threshold, then the cluster is recomputed again. In this manner all clusters are maintained at a manageable level.

The tree is formed based on the density of nodes in the network. First of all, the network is divided into clusters as mentioned above. If there is more than a certain number of nodes in a given cluster, then that cluster is again divided into subclusters with a new sub-cluster head. This process is recursive till the size of cluster is manageable. In the simulation, which is carried out as a part of this research, only one iteration of the algorithm was necessary. Our approach has good scalability over big networks as it recursively forms sub-clusters.

The above scenario can be divided into 2 cases and its sub-cases as follows.

Case 1: Child in a cluster moves

1A – When a child moves out

For every specified time interval, the cluster head sends HELLO message to all its child nodes. If after a certain RTT (Round Trip Time) value, it doesn't receive an ACK, the cluster-head marks that node as possibly a moved out node and sends a second

HELLO message. After the second HELLO message, if the cluster-head still does not receive an ACK, it assumes that node has traveled outside the cluster boundaries. If the parent is at the top of the tree, it will notify the neighboring cluster-head of all child cluster-heads about the possible arrival of the node. Neighboring cluster-heads will run the search algorithm to find the new node in their cluster if there is any. If a cluster-head finds the new node, it establishes the parent-child relationship between itself and the new node and adds that node to the list of its child nodes. If a parent is not the top of the tree, it will notify its parent in turn about the node moving out of the cluster. The parent in turn notifies its child nodes which are neighbors of the first cluster-head and then the neighboring cluster-heads run the search algorithm described above. This is done recursively up the tree hierarchy if needed. As explained above, if the number of child nodes within a cluster reaches a certain lower threshold limit, the cluster is re-computed again by running the dominating sets algorithm.

1B – When cluster-head is notified about possible node arrival

When a cluster-head is notified about the possible arrival of the node from a neighboring cluster-head or its parent in the tree, it runs the search algorithm to find the new node if there is any. This is explained below.

If clusterhead A detects a new node it will inform its child cluster-head A1 or its neighbor B. A node in the network can move freely with no knowledge of clusters. It is the duty of the cluster-head to make correct associations with nodes within its cluster. Cluster-head B broadcasts an IDENTIFY message in its cluster. The IDENTIFY message tells every node in the cluster that cluster-head needs to identify its children. Every node in its cluster will reply with the Id of the node in encrypted form. It then matches the Ids it has

received with the Ids in its table of cluster members. If the Ids are exactly the same as that in the table, it assumes that there is no new node in its cluster and ignores the notification of the possible arrival of a new node.

If there is a reply from the node with the new Id, it adds that Id to its cluster members; notifies the peer about the new arrival and updates the table. The peer node in turn updates its tables as well. A peer node may be the neighboring cluster or the parent up the tree hierarchy. When a neighboring cluster-head notifies the cluster-head about the arrival of a new node, the peer node is the neighboring cluster-head. When a parent notifies one of its children about the arrival of a new node, the peer node is parent cluster-head. If the peer is the parent, it notifies the child node about the node movement. The child node in turn updates its information.

Case 2: Parent moves out of the cluster.

In this case, there are two scenarios.

Case 2A: Parent is not the top node of the tree.

If the parent of the cluster is not the root of the tree, then it is the child node of some other node in the tree hierarchy. This cluster-head is treated as a node in the cluster with the cluster-head as parent of the node. This is similar to case 1. Therefore, the cluster-head is moved to another cluster and its relationship is established with another cluster-head as described in case 1. The only problem here is child nodes of a moved out cluster-head become orphans as they have no cluster-head. To deal with this problem, the grand-parent of those nodes acts as a cluster-head for those nodes. If there are more nodes in the cluster than the threshold, the dominating sets algorithm is executed again to re-compute cluster boundaries.

Case 2B: Parent is the top node in the tree.

In this case, if the cluster-head moves, it goes to another cluster. As a result, child nodes in a particular cluster do not receive HELLO message from the cluster-head. Therefore, child nodes in that cluster chose some node from themselves as a temporary cluster-head depending on some voting scheme. For this simulation, it has been assumed that the node with highest Id will be new cluster-head. This cluster-head then broadcasts the NEW-CLUSTER-HEAD message all over the network. When a neighboring cluster-head receives this message, it updates the table with the new cluster-head's id and send out PEER-CLUSTER-HEAD messages to the new cluster-head. In this way, every cluster-head is updated.

After a new cluster-head is assigned for a given cluster, it then notifies neighboring cluster-heads about the possible arrival of a new node which was an old cluster-head of its cluster. The neighboring clusters will then run the search algorithm described above and establish the relationship with the new node in the cluster if there is any.

4.2 Hierarchical Distributed intrusion detection

In the trust architecture proposed in [1] there is an IDU at each node to detect intrusions. However, if the node is not trustworthy, any intrusions detected may not be reported by the node or a node may report false intrusions. We propose a hierarchical intrusion detection system to

1. Detect intrusions

2. Ensure that detected intrusions that are reported are trustworthy. In other words, even if an untrustworthy node reports (or fails to report) an intrusion, the hierarchical intrusion detection system will determine whether a trustworthy intrusion has taken place.

4.2.1 Intrusion detection

For detecting an intrusion in the network, we have used a divide and conquer approach. If the intrusion can be detected within the cluster, then the cluster-head of the cluster will detect the intrusion by itself. If it can not be detected using one cluster-head, neighboring cluster-heads will detect the intrusion together in a cooperative manner. This is explained in detail below. If the source node and destination node are in the same cluster, then intra-cluster intrusion detection is invoked. On the other hand, if source node and destination node are in different clusters, then inter-cluster intrusion detection is invoked.

Phase 1: Intra-cluster Intrusion Suspicion formation

The hierarchical structure of our approach makes it scalable. A suspicion is raised when the activity is not as expected, but this may be due to unusual but innocent factors. It is therefore not advisable to classify such nodes as malicious. Or the other hand, there are some activities which can be immediately detected as being malicious and in such cases, an intrusion is detected and the node is branded untrustworthy. We do not discuss each attack in detail, instead we use the DoS attack to illustrate our approach. The logic for detecting other attacks is exactly the same as DoS. We simply need to modify the parameter values for the other types of attack. For the DoS attack, the monitoring node sets a timer and checks if the node is sending packets before the RTT (Round Trip Time) expires. If the

number of packets exceeds a certain threshold value, that node is labeled as suspect. The suspicion is defined as

Suspicion = (no of packets over threshold – threshold level)/threshold level

The suspicion is normalized. This information is sent to the neighboring clusterhead in the path of the packet.

As stated earlier, the underlying model used here is the Bayesian network model [1]. All nodes are divided into clusters and each cluster has a cluster head. If a node in a cluster finds out that a particular node in the network is misbehaving, it calculates the conditional probability that the node is malicious. This probability is based upon many parameters. Some of the important parameters are node history, active time in the network, ratio of transmitting invalid data or control information against valid data or control information etc. Based on these factors, probability of maliciousness is calculated for the node in question using the Bayesian principle.

The naïve Bayes' Theorem is mathematically expressed as

$$P(h \mid e) = \frac{P(e \mid h) * P(h)}{P(e)} \tag{1}$$

$P(h)$ is called the prior probability of hypothesis and $P(e)$ is the prior probability of evidence e. $P(h|e)$ is the conditional probability of occurrence of hypothesis h given evidence e is true is known as the posterior probability. Similarly $P(e|h)$ is the probability of e given h. Hence when we get all the values, we can calculate the posterior probability using Bayes' theorem.

Using the Naïve Bayes' theorem, we can calculate the probability of an attack by a node on the network. Consider the example of a packet misrouting attack. Assume that, node 1 in cluster A is sending packets to node 3 in cluster B through node 2 in cluster A.

The clusterhead of node A will snoop on nodes 1 and 2 to check the intrusion. In the initialization phase, node 2 has been assigned some prior probability value for attack by the Certification Authority depending upon the trust of the node. Suppose that probability is denoted as $P(h)$. If the clusterhead of A finds that node 2 is misbehaving by misrouting the packet, it calculates the posterior probability of attack $P_{att.}$

If we assume that the probability of packet misrouting by node 2 is $P(e)$, the probability of attack P_{att} is obtained from (1) as.

$$P_{att} = \frac{P(e \mid h) * P(h)}{P(e)} \qquad (2).$$

Over a given range on samples, e.g., if out of 10 times, node 2 misroutes the packet 6 times, $P(e)$ can be calculated as

$$P(e) = \frac{\#\_of\_packets\_misrouted}{Total\_\#\_of\_packets\_received}$$

Hence, $P(e)$ in this case would be 6/10 = 0.6.

Similarly, probability of evidence $P(e)$ is calculated for different types of attacks as follows.

Information Destruction attack:

$$P(e) = \frac{\#\_of\_packets\_destroyed}{Total\_\#\_of\_packets\_received}$$

Spoofing:

$$P(e) = \frac{\#\_of\_packets\_with\_nodeID\_changed}{Total\_\#\_of\_packets\_received}$$

$P(e|h)$ is the conditional probability of evidence given hypothesis h is true. Thus if out of 100 attacks identified, if 20 attacks are DoS, then $P(e|h)$ for DoS is 0.2. Similarly, $P(e|h)$ can be calculated for other types of attacks.

Now we have all the values on the right hand side of equation 2. Putting those values in the equation, we can calculate the posterior probability of attack by a node that causes packet misrouting.

P_{att} calculated using the above procedure is then compared with the pre-defined threshold value for the network. This threshold value is set up at the time of network initiation phase and is one of the parameters for tuning the performance of the scheme. If the threshold value is set up too high, then there would be many false alarms. If it is too low, then many nodes would not be detected by the scheme. For this simulation, it is set up at 0.6. The threshold value of 0.6 is optimal in the sense that, it is greater the average trust i.e. 0.5 and it is also not too high. If the probability of attack is greater than the threshold, the node 2 will be marked as intrusive and the clusterhead will convey the information to neighboring clusterheads.

Phase 2: Inter-cluster level Intrusion suspicion/detection

It is possible that several nodes may be taken over by a malicious node for a DoS or other attack. Each node along the path may initiate a small DoS attack resulting in a full-fledged attack on the network. Suspicion may be raised with each cluster, but that does not indicate that an attack is actually taking place. In such cases it becomes necessary to obtain information from multiple cluster-heads to determine if an intrusion is occurring or not.

In our approach, a clusterhead cannot detect intrusions within its own cluster. This is because the cluster-head itself may not be trustworthy and any intrusion it claims to detect (or vice-versa) may not be true. Therefore a cluster-head only detects intrusions based on data it receives from other cluster-heads. If the cluster-head itself is being

malicious, its maliciousness is detected by neighboring cluster-heads or the parent of that cluster-head (if there is one) using the underlying trust model.

Working of Intrusion Detection Unit:

In our approach, each node will construct a simple Bayesian network to determine if an attack is actually taking place. In particular, we consider the following scenarios:

Single attack by a single node within a cluster

If the attack is a DoS attack by a node in cluster *n*, the cluster-head *n* sends the suspicion to the next cluster-head *p* in the path of the packet. Cluster-head *p* derives the Bayesian network as shown below (fig 4.4) Here $Tr(n)$ is defined to be the trust on the cluster-head *n* that sent the information. This trust is determined using the bayes trust model described in [1]. As mentioned in [1], each node keeps a trust table of every other node in the network. $Att(n)$ is defined to be the probability of attack, which is the normalized suspicion where *n* is the id of cluster-head *n*. $Att_i(n)$ is the conditional probability that the attack has taken place given the trustworthiness of node *n* where *i* is a specific event like Spoofing. The procedure for calculating $Att(n)$ is explained in the previous section.

$$Att(n) = \frac{P(e \mid h) * P(h)}{P(e)}$$

Here $P(h)$ = Prior probability of hypothesis = Prior Probability that the attack is DoS.

$P(e)$ = Prior probability of evidence = Probability that the attack occured is found out to be DoS attack.

The probability of detecting an attack given that the node *n* is reporting an intrusion suspicion (fig 4.3): In this equation, *Tr*(*n*) is trust of the node *n* which reports the intrusion.

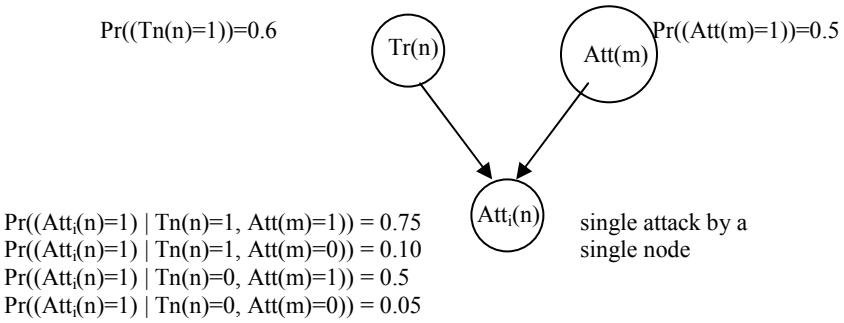

Figure 4.3: Bayesian Network model for single node attack

$$\Pr(Tr(n) \wedge Att(n) \wedge Att_i(n) = \Pr(Tr(n) \times \Pr(Att(m) \times \Pr(Att_i(n) \mid Tr(n) \wedge Att(m))) \qquad (3)$$

In this equation, if we substitute the values of trust of node and normalized probability values, we get the probability of attack by the node. This can be clarified with an example. Suppose, node A is a DoS intruding node (this is not a cluster-head). Node B which is the clusterhead of cluster with node A detects the intrusion and it notifies the neighboring cluster-head C of the possible intrusion. Node C has a certain level of trust on node B along with the value for probability of DoS attack. Suppose the trust on node B by node C is 0.75. This trust value is obtained from the underlying trust model proposed in [1] and the probability of attack with evidence is 0.6, and the posterior probability of DoS attack by node A is 0.5.

In other words, $Tr(B) = 0.75$,

$$Att_{DoS}(B) = 0.6$$

$Att_{DoS}(A) = 0.5$

Here using equation 1, the probability of DoS attack by node A can be calculated as

Probability of DoS attack. by A = $Tr(B)$ * (($Att_{DoS}(B)$ * $Att_{DoS}(A)$) / $Tr(B)$ AND $Att_{DoS}(B)$

= 0.75 * ((0.6 * 0.5) / 0.75 AND 0.6)

= 0.225

If an attack is determined, the trust in the attacking node is adjusted by the formula explained below. Probability of attack by a given node is calculated using the above formula. As explained above, cluster-head *p* has the information of the Trust probability of node *n*. Considering the trustworthiness of the node which raises intrusion suspicion to node *p*, node *p* calculates the probability of attack by constructing the Bayesian Network as shown in figure 4.4. Based on the trust of the node and conditional probability of the attack, normalized probability of the attack is calculated. For example, if probability trust of node is 1 and probability of attack is also 1, then there is high probability that the attack has actually taken place. This is denoted in figure 4.4 as $Pr(att(n)) = 0.75$. Similarly, if the trust of node is 0 and probability of attack is also 0, then there is slight possibility that actual attack have taken place. Hence in the last case, $Pr(att)$ is 0.05.

There is one problem remaining – what if the intrusion detection node is itself not trustworthy. Since the intrusion suspicion is broadcast, other clusterheads will also be computing the intrusion detection and a simple majority vote taken.

Each node maintains the table which contains the following information. Node Id, Cluster Id, Trust of a node on scale of 0 to 1 where 0 is no trust and 1 is full trust, $P(e)$ for DoS,

$P(e)$ for Misrouting, $P(e)$ for Information Destroyal, $P(h)$ for DoS, $P(h)$ for Misrouting, $P(h)$ for Information Destroyal, Total # of attacks, # DoS attacks, # Misrouting attacks, # Information Destroyal attacks. At each time, the algorithm is run, probability values and trust is updated for the given node.

Single attack by a multiple nodes in multiple clusters

Here multiple clusterheads may send suspicions to peer cluster-heads Here $Att_i(m)$ is the probability of attack for a specific event i reported by node m, that is, the suspicion.

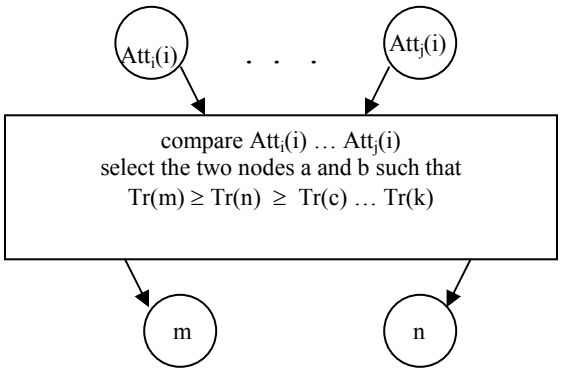

Fig 4.4: selection of two most trustworthy nodes

When multiple cluster-heads send a suspicion, two of them are chosen in the following manner: The trust level of each of the cluster-heads is compared with each other. The cluster-heads which have the highest levels of trust among them are chosen for constructing the Baysian Network. This is because, every cluster-head will have the same information about the intrusion suspicion. So there is no need to take into account the information sent by every node. Instead, two cluster-heads with highest trust level are chosen and their information is used in detecting the intrusion.

As shown in figure 4.5, multiple cluster-heads raise the suspicion on the particular node. After comparing trusts of all nodes, only nodes *m* and *n* that have the highest trust levels are chosen. A clusterhead may receive reports on a single attack *i* from many nodes. This may result in a large Bayesian network. Resources in ad hoc networks are limited and to keep the Bayesian network simple, only the reports from the two most trustworthy nodes m and n are considered. This process is shown in fig 4.5 above. Next the network obtains the probability that cluster-head *m* and *n* are suspicious of a particular attack *i* and calculates the total probability (fig 4.5).

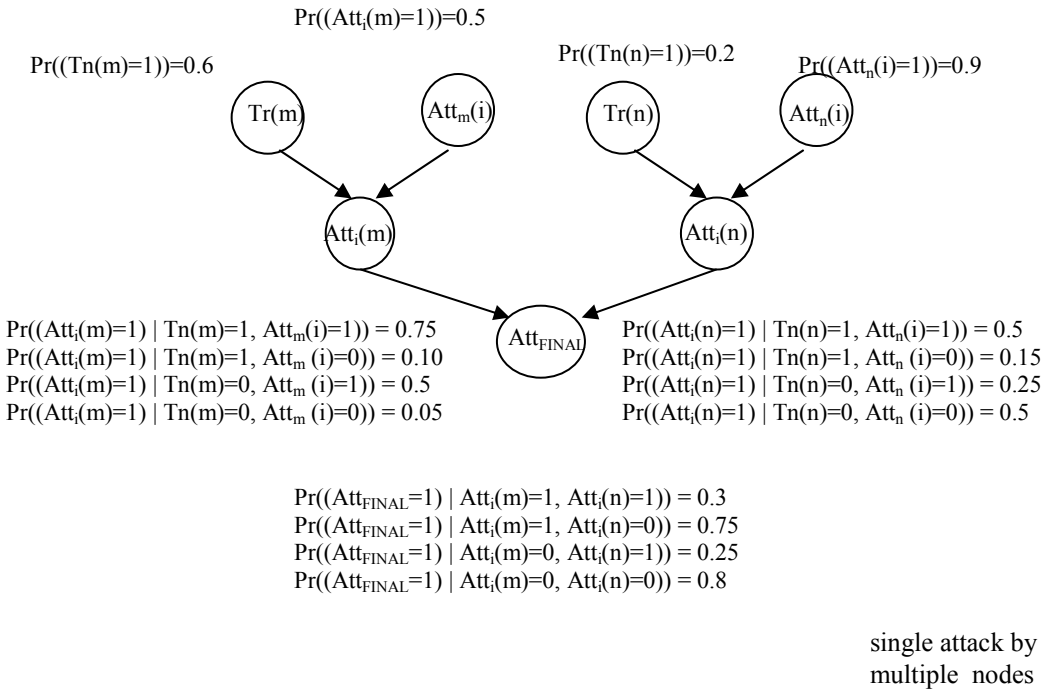

Fig 4.5: Bayesian network on two nodes

$$\Pr(Tr(n) \wedge Att(i) \wedge Att_i(n) \wedge \Pr(Tr(m) \wedge Att(i) \wedge Att_i(m) = \Pr(Tr(n) \times \Pr(Att(i) \times \Pr(Att_i(n) \mid Tr(n) \wedge Att(i)))$$
$$\times \Pr(Tr(m) \times \Pr(Att(i) \times \Pr(Att_i(m) \mid Tr(m) \wedge Att(i)) \times \Pr(Att_{FINAL} \mid Att_i(m)) \wedge Att_i(n))$$

There will be more entries in the tables in this case, but it will still be minimal and not take up much storage. Furthermore table look up is fast.

If an attack is determined, the trust in the attacking node is adjusted by underlying trust model explained in [1]. Therefore in both cases the probability of an attack being classified as an attack is at its highest when the node reporting a suspicion is trustworthy and the attack suspicion is high.

The above two would be the most common attack scenarios in an ad hoc network. Other possible attack scenarios would include a cluster-head reporting multiple attacks at the same time or multiple cluster-heads reporting multiple attacks at the same time.

For other types of attacks viz. Packet misrouting, spoofing and Information destruction attack, the cluster-head will form the Bayesian network similar to the one explained in the above example. However, for each type of attack, there would be a different probability of attack value associated with the given node.

If there are multiple attacks by the same node, the cluster-head of that node will notify each attack separately to other cluster-heads. Each cluster-head then will treat this attack as a separate attack and construct the Bayesian network for each attack.

4.3 Trust Based Intrusion Detection Algorithm

The algorithm for the Intrusion Detection Unit is outlined below:

for each node *i,* set: threshold values th_{DoS} for DoS attacks; inactivity timeout t_{DoS}; updating time *update* for the intrusion table
Define clusters based on dominating sets algorithm. All Clusterheads snoop by monitoring the incoming packet and outgoing packet for each node

```
While network is up
begin
        If TimeToUpdate == true
        begin
                Update network tables.
                Check for new nodes
```

Remove nodes which are moved out of network
If ReconstructDominatingSets == true
b*egin*
Re-organise Dominating sets
end
end

If clusterhead *j* detects mismatch between incoming packet and outgoing packet at node *i then*
begin
clusterhead j runs probability calculation on node i to determine probability of attack;
clusterhead *j* sends probability to neighboring clustehead *k*
clusterhead *k* uses Bayesian network to determine maliciousness nm_i of node *i* depending upon the trust of clusterhead *j*
If node maliciousness value $nm_i > {}^{th}_{DoS}$ *then*
network is flooded with accusation message for node *i*.
If cluster-head *j* is malicious, *then* clusterhead *k* will determine it with the help of other neighboring cluster-heads
end
end

CHAPTER 5

SIMULATION AND RESULTS

The following performance metrics were studied during the simulation.

- Accuracy of detection
- Accuracy of trust model
- Overheads (cluster formation, Control messages, trust calculation etc.)
- Packet delivery ratio

This simulation is carried out using the following scenario.

- Number of nodes in the network: 100.
- % of attacking nodes:
    - Case 1: 10%
    - Case 2: 25% and
    - Case 3: 50 %
- Attacks simulated: DoS, Packet misrouting, Spoofing.
- Nodes for each attack: 10, 25 and 50.
- Communications taking place: Point to Point wireless IP communication with no access points.
- Information about the nodes is gathered every 10 seconds.
- Packet size: Each packet carries a variable payload. It is determined randomly. Values can range from 0 to 1024 bytes.
- Speed: Speed of movement of nodes is random. Speed can range from 1 to 10 m/s

- Transmission rate: A node can transmit packets randomly. It depends upon when the node is selected to transmit the packet. The transmission interval is also chosen randomly. Transmission interval can range from 0 seconds to 1 minute. Number of packets transmitted can vary between 0 to 100 packets in a minute. Regular payload on the packet is at max 1024 bytes. (1 KB)

Detailed simulation pseudo code is explained in appendix. In brief, following steps are performed for the simulation

Network Initiation Phase:

// Initialize node parameters i.e. node-id, trust of node and probability of attacks.

- For each node in the network, assign a node id, randomly assign trust value and various probability values.
- Choose some nodes are intrusive nodes.
- Divide network into clusters using dominating sets algorithm proposed in [14].

Network Uptime Phase:

This phase marks the actual working of the wireless network, nodes move throughout and out of the network, they communicate with one another, they die due to lack of energy etc. It is in this phase that, IDUs of cluster-heads detect if there is intrusion in the network. This works as follows.

- Randomly move nodes from point to point
- Simulate communication between nodes
- Cluster-heads snoop on the network
- If intrusion is detected network tables are updated
- If needed, clusters are recalculated using dominating sets algorithm

The algorithm proposed above takes finite time to converge to a result. It marks intrusive nodes with desirable accuracy. Here desirable accuracy means, the accuracy measure that was set up prior to the simulation run. For 3 runs of the simulation, there was only one false positive over 50% intruding nodes which is very good accuracy. The most complex part of the algorithm is to divide the network into clusters. For performance, we can run the cluster formation in a separate thread so that it would not affect the working of the main thread. Detecting an intrusion is a straightforward process. Therefore is fast and it does not consume more resources on the node.

After running the simulation for different sets of nodes and different number of intrusive nodes, it is proved that the simulation marks intrusive nodes and eliminates them.

The above cases cover all the scenarios in real time. This means that, there is no waiting time to get the result of the intrusion detection. It is calculated while the algorithm is running and it is applied instantly to mark and remove intruding nodes. In the first case, it is assumed that 10% of nodes are intrusive. To detect all the intrusive nodes in the network under this condition, it is seen that it needs few number of iterations and control messages to mark all intrusive nodes. In the second case, it is assumed that, 25 % of nodes are intrusive. To detect all intrusive nodes, the number of iterations and control messages needed is more than that of case 1. Similarly, for the case 3, it is assumed that 50 % of nodes are intrusive. It takes considerably longer time to find all intrusive nodes. Moreover, in this particular case, the number of control messages exceeds the number of actual data messages.

By looking at the trend in the above test cases, it can be stated that as the number of intrusive nodes increase in the network, the rate of convergence decreases and the number of control messages needed to detect intrusion increases. Rate of convergence means the rate at which bad nodes are removed from the network.

Below is the graphical representation of the scenarios discussed above. In the following graph, one iteration constitutes for one complete scan of the system – One scan includes running the algorithm for one minute. Every 1 minute, the whole process is executed again to scan the network. One iteration consists of timeframe of 1 minute.

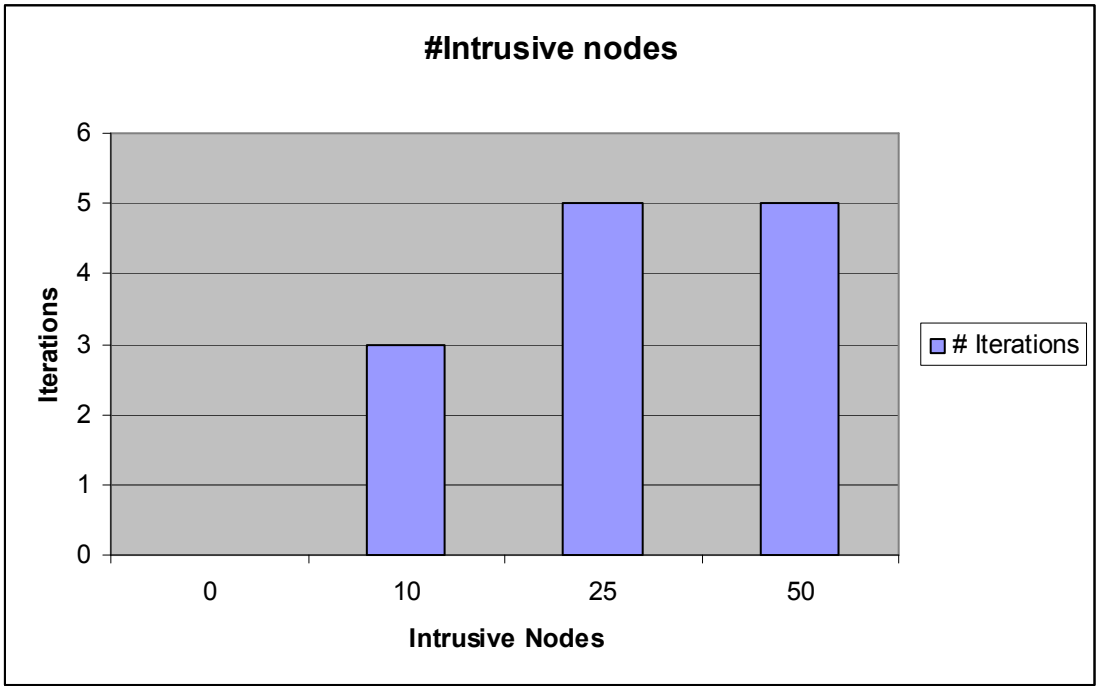

Figure 5.1: Number of intrusive nodes

The simulation is executed for 3 times for each different percentage of intrusive nodes and the average taken. In first case, all the 10 % intrusive nodes were detected correctly by the 3rd iteration. In second and third case, 25 % and 50 % intrusive nodes were detected correctly by the 5th iteration. This is because, as number of intrusive nodes increase in the network, the number of attacks also increase. This would cause cluster-

heads to report more attacks in a comparatively small timeframe. Thus 25% and 50 % nodes were detected in 5 iterations. This shows that algorithm converges quickly for increasing number of intrusive nodes.

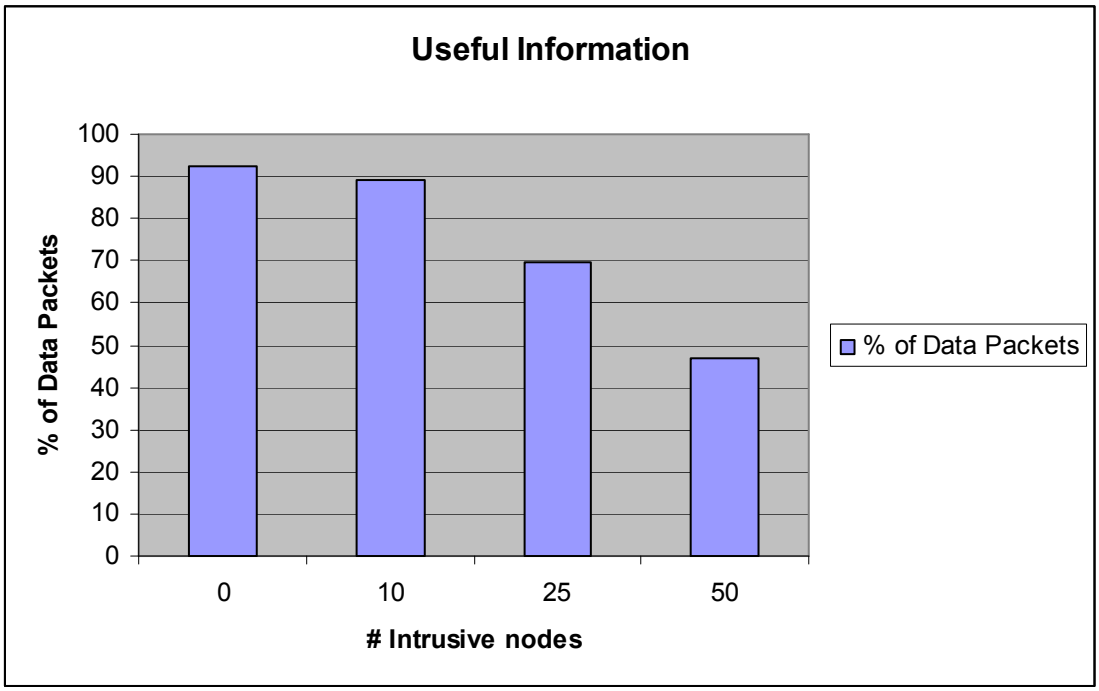

Figure 5.2: Useful information during communication

In the above graph, useful information means the percentage of data packets sent while the algorithm is running. To simulate a real time situation, the network consists of data packets as well as control packets and messages to maintain and form dominating sets. There is also considerable flow of messages between cluster-heads. Considering all these scenarios, it is seen that the percentage of data packets decrease as the number of intrusive nodes increase. The simulation is run till all the intrusive nodes are removed from the network. This is due to the fact that as number of intrusive nodes increase, it takes more effort and time for cluster-heads and nodes to mark them as intrusive and broadcast that information all over the network. Hence, the number of control packets increases.

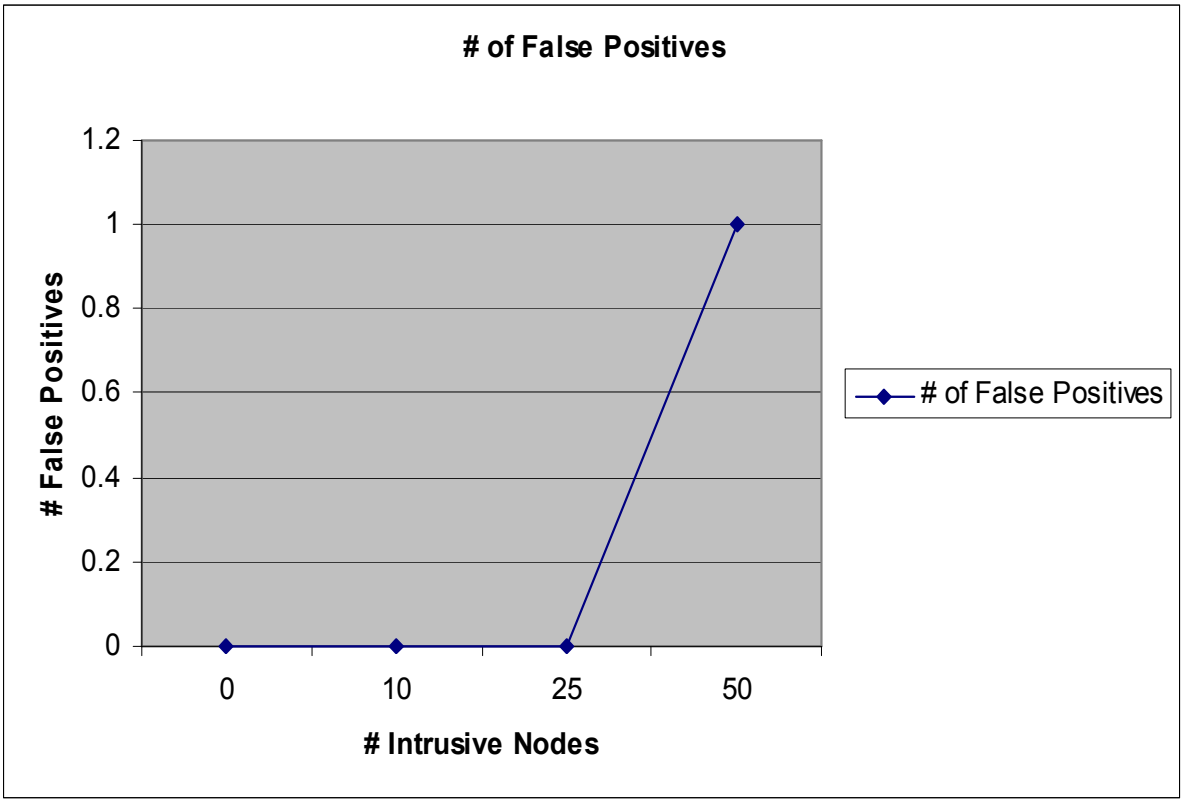

Figure 5.3: Number of false positives

This is the graph of false positives vs. intrusive nodes. From the graph, it is seen that out of 100 nodes there was not a single false positive until the number of intrusive nodes reached 50%. This confirms the fact that the algorithm proposed here is very accurate. When the number of intrusive nodes was set to 50 % of total nodes in the network, there was one false positive; meaning, one node was detected as intrusive node even though it was not one.

To compare the effectiveness of this trust scheme with a non-trust scheme, one more analysis is performed. Initially, the algorithm is executed with no trust consideration for all nodes. It is noted that, as time progresses, cluster-heads which are intrusive tend to raise false intrusion notification. This causes an increase in the number of false positives over time. This is shown in the following graph.

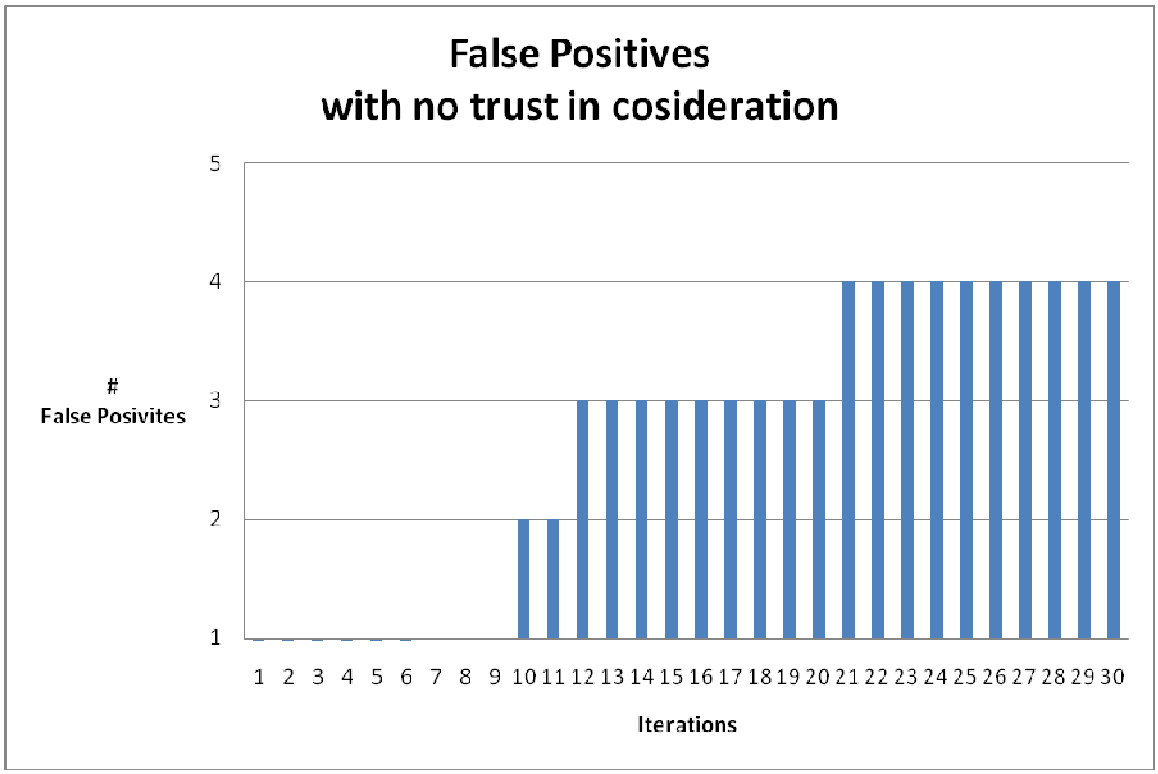

Figure 5.4: False positives with no trust in consideration

In the next run, trust is restored into the nodes and is increased gradually according to equation (1). It is interesting to note that until the trust in the cluster-heads becomes stable, the number of false positives is almost similar to that of the above run which does not include trust. But after the network stabilized, the number of false positives is reduced. This is because, when a cluster-head with less trust notifies about an intrusion, that node is not marked as intrusive immediately., instead, it is marked as a "potentially intrusive" node. In the subsequent intrusion detection process by other cluster-heads, it is found that the cluster-head which initially reported an intrusion is lying. Thus trust in that cluster-head is reduced and the node which is marked as potentially intrusive is unmarked. This is depicted in following graph.

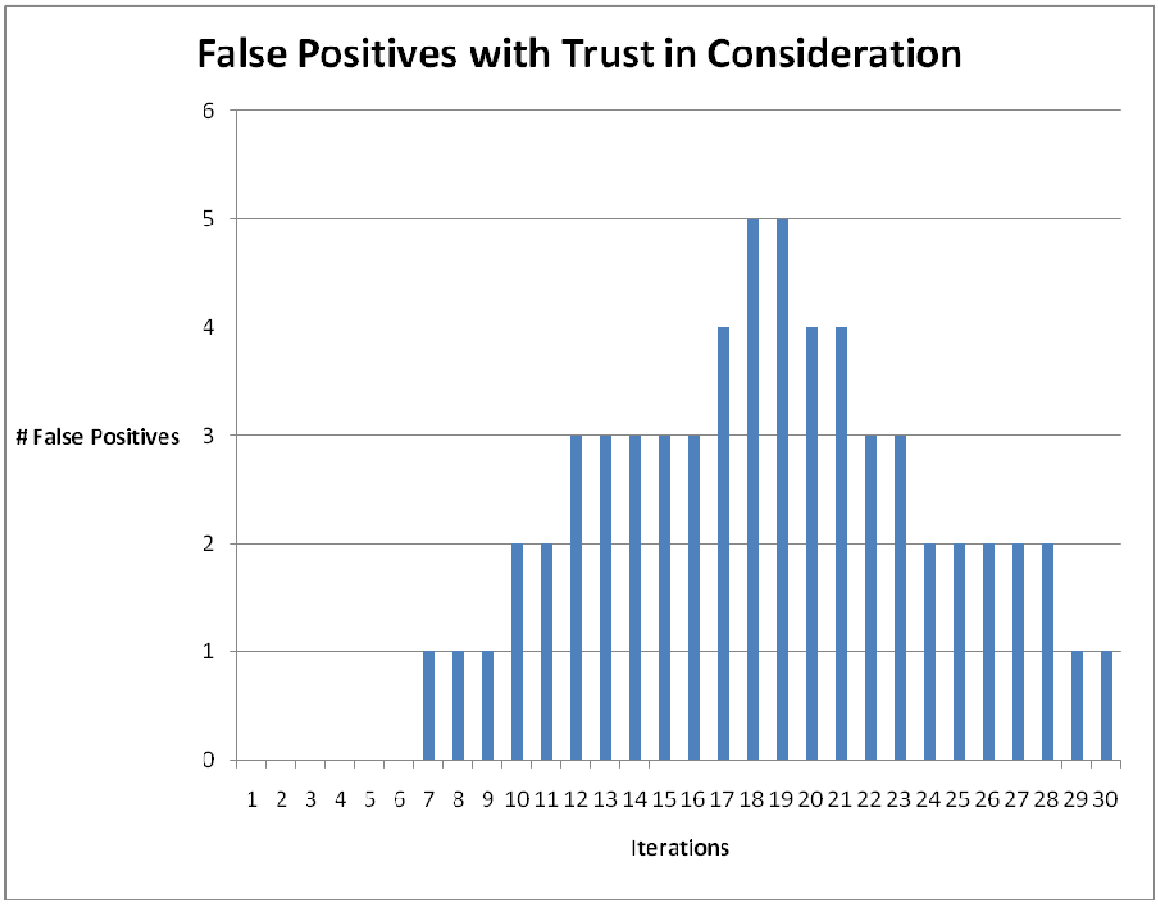

Figure 5.5: False positives with trust in consideration

As can be seen, initially, the trust in not built. Over a period of time, the network matures and correct trust is built between the nodes. Thus until the trust is built we get many false positives meaning nodes which are innocent are marked as malicious by an intrusive cluster-head. However once the correct trust of the intrusive cluster-head is determined, the number of false positives reduce because intrusive cluster-heads are identified by neighboring cluster-heads.

CHAPTER 6

CONCLUSIONS

In this thesis, we have proposed a unique hierarchical approach to detect intrusions in Ad-Hoc Wireless Networks. The basic trust model introduced by J Thomas [1] ensures that trust is established between peers in the network. Depending upon the trust and the maliciousness of the node, the algorithm efficiently detects intrusive nodes and excludes them from utilizing the network resources. Furthermore due to the conditional probability methods used in conjunction with Bayes Theorem, it is ensured that even if some innocent node misbehaves sometimes, it will not be treated as an intrusive node immediately. From the graphs, it is evident that the model is very accurate in detecting intrusions. Furthermore, it converges very well for a large percentage of intrusive nodes in the network. In terms of battery power, it uses the minimal power required to detect intrusions as not all nodes need to be involved in intrusion detection. This method has little overhead in detecting the intrusion as it divides the network into dynamic clusters. Moreover, after a specified amount of time, clusters are recalculated to ensure the integrity of network. Although there are some overheads, considering the advantages and simplicity of use, these are overheads that are worth the cost.

There is room for improvement on this work in several areas. Firstly, the algorithm used for calculating dominating takes exponential time as the network grows. This model is simulated for 4 types of security attacks. It can be extended to all other

attacks with little modification. The Bayesian model can also be extended to make it more generic. For example, multiple attacks by a single node, or multiple nodes launching attacks at the same time can be considered.

References:

[1] Sudha Chinni, Johnson Thomas, Georghitta Ghinea and ZhengMing Shen, "Trust Model for Certification Revocation in Ad hoc Networks", Journal of Ad Hoc networks (accepted).

[2] Marti S, Giuli T J, Lai K, Baker M, "Mitigating routing misbehavior in mobile ad hoc networks", Proceedings of the Annual International Conference on Mobile Computing and Networking (MOBICOM), pp. 255-265, 2000.

[3] Parker J, Undercoffer J, Pinkston J, Joshi A, "On Intrusion Detection and Response for Ad hoc Networks", Proceedings 23rd IEEE international performance computing and communications conference – Workshop on information assurance, Apr 2004

[4] Vigna G, Gwalani S, Srinivasan K, Belding-Royer E M, Kemmerer R A, "An Intrusion detection tool for AODV-based Ad hoc Wireless Networks.", Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04), pp. 16-27, 2004.

[5] Hubaux J, Buttyan L, Capcun S, "The Quest for Security in Mobile Ad hoc Networks", ACM Special Interest Group on Mobility of Systems, Users, Data and Computing. pp. 146-155, 2001

[6] Chin-Yang Tseng, Balasubramanyam P, Ko K, Limprasittiporn R, Rowe J, Levitt K, "A specification based intrusion detection System for AODV", Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks., pp. 125-134, 2003

[7] Patwardhan A, Parker J, Joshi A, "Secure Routing and Intrusion Detection in Ad Hoc Networks" Proceedings of the 3rd International Conference on Pervasive Computing and Communications, pp. 191-199, March 2005

[8] Zhang Y, Lee W, "Intrusion Detection in Wireless Ad Hoc Networks", Proceedings of the 6th annual international conference on Mobile Computing and Networking, pp. 275-283, 2004

[9] Kong J, Zerfos P, Luo H, Lu S, Zhang L, "Providing Robust and Ubiquitous Security Support for Mobile Ad hoc Networks", Proceedings of the Ninth International Conference on Network Protocols (ICNP'01), pp. 251-260, 2001

[10] Luo H, Zerfos P, Kong J, Lu S, Zhang L, "Self-Securing Ad Hoc Wireless Networks", Proceedings of the Seventh IEEE International Symposium on Computers and Communications (ISCC), pp. 567-574, July 2002.

[11] Nekkanti R K, Wei-Lee C, "Trust based adaptive On-Demand Ad Hoc Routing Protocol", Proceedings of the 42nd ACM Southeast Regional Conference, pp 88-93, 2004

[12] Zheng Y, Zhang, Peng, "Trust Evaluation Based Security Solution in Ad Hoc Networks", pp. 1-14, Proceedings of Seventh Nordic Workshop on Secure IT Systems, 2003.

[13] Claude C., Carlton R D, "A Certificate Revocation Scheme for Wireless Ad Hoc Networks", Proceedings 1st ACM Workshop on security of Ad-Hoc and Sensor Networks, pp. 54-61, 2003

[14] Dai F, Wu J, "A Extended Localized algorithm for Connected Dominating Set Formation in Ad Hoc Wireless Networks.", IEEE transactions on parallel and distributed systems, Vol. 15, No. 10, pp. 1-13, 2004

[15] Neapolitan R E, "Learning Bayesian Networks", Prentice Hall Series in Artificial Intelligence, "ISBN #: 0 – 13 – 012534 - 2"

[16] Kui Wu., University of Victoria, British Columbia, Canada, Department of Computer Science, "http://www.csc.uvic.ca/~wkui/Courses/topic/Lecture4.pdf", Date last accessed 06/21/2006.

[17] Tiranuch Anantvalee, Jie Wu, "A Survey on Intrusion Detection in Mobile Ad Hoc Networks",http://www.cse.fau.edu/~jie/research/publications/Publication_files/intrusion06.pdf, pp. 170-196, Date last accessed, 04/07/2008.

[18] Y. Zhang, W. Lee, and Y. Huang, "Intrusion Detection Techniques for Mobile Wireless Networks" ACM/Kluwer Wireless Networks Journal (ACM WINET), Vol. 9, No. 5, pp. 546-556, September 2003.

[19] P. Albers, O. Camp, J. Percher, B. Jouga, L. M, and R. Puttini, "Security in Ad Hoc Networks: a General Intrusion Detection Architecture Enhancing Trust Based Approaches" Proceedings of the 1st International Workshop on Wireless Information Systems (WIS-2002), pp. 1-12, April 2002.

APPENDIX

Detailed pseudo code of intrusion detection algorithm:

Network Initiation Phase:

1. Initialize node parameters i.e. node-id, trust of node and probability of attacks.

```
For i = 1 To n……..\\n = no. of nodes in network.
    Node(i).Id ← i;
    Node(i).Tr ← rand();
    Node(i).Pr-DOS ← rand();
    Node(i).Pr-Spoof ← rand();
    Node(i).Pr-PMR ← rand();
    Node(i).Pr-ID ← rand();
EndFor
```

```
// Set some nodes as Intrusive.
    Badids ← GenerateRandomIds();
    For I = 0 to BadIds.Length()
        Node(id).IsIntrusiveNode ← True;
    EndFor
//  Divide the network into clusters.
 ClusterNetwork();
```

For clustering network, dominating Sets algorithm proposed by Fie Dai and Jai Wu [14] is used. The above algorithm explains how the nodes in the network are initialized and clustered. For testing purposes, some nodes are assigned as intrusive nodes.

Network Uptime Phase:

As stated above, this phase marks the actual working of the wireless network, nodes move throughout and out of the network, they communicate with one another, they die due to lack of energy etc. It is in this phase that, IDUs of clusterheads will find out the intrusion in the network. This works as follows.

To simulate the exact working of the network, variable timer value is used. In every swipe of the code, timer is assigned a random value generated by a function rand(). This will enable moving of nodes randomly at random time To move nodes, ids of the nodes are selected randomly and those nodes are moved in random directions in the network. So, in this simulation, any node can move any time in any direction. Also, more than one node can move in any direction at any time. This is the similar condition for any kind of real AdHoc Network. For mobility, random waypoint model is used.

```
// Note: Each function is explained in detail below.
TimerFunc()
{
        MoveNodes();
        Communicate();
        Snoop();
        UpdateNetworkValues();
```

```
        TimerInterval = rand();}
// Function Definition for MoveNodes();
MoveNodes()
{
        NodesTobeMoved ← GenerateRandomIds();
        For I = 0 To NodesTobeMoved.Length()
                Direction ← GenerateRandomDirection();
                Speed ← GenerateRandomSpeed();
                Node(id).Move(Direction, Speed);
        EndFor
}
// Function Definition for Snoop();
Snoop()
{       clusterHeads ← GetClusterHeads();
        foreach cluster_head in clusterheads
                Foreach node in clusterHeadChildren
                        if (Node.IsCommunicating)
                                CheckForIntrusion(Node.Id, Node.Packet);
                        EndIf
                EndFor
        EndFor
}
```

The code above is very self explanatory. During snooping activity, each cluster-head scans its own cluster. If any node in the cluster is in communicating state, it extracts the source, destination and next-hop address from the packet. These values are then used to check the intrusion. Here it is assumed that a cluster-head can snoop on one transmission at a given time.

```
// Function definition for UpdateNetworkValues()
{
        RecalculateClustersIfNeeded();
        UpdateTables();
}
```

This function checks to see if any network parameters such as clusters, intrusion tables need to be updated. For example, it may be needed to re-organize clusters because these clusters are not optimized for their operations. Also, intrusion tables may need to be updated with new information collected by cluster-heads. All this is done during this phase and the result is notified to other cluster heads.

```
//Function definition for RecalculateClustersIfNeeded()
{
        If TimeSinceLastClusterFormation > 5 Mins
        Foreach Cluster-Head in cluster-heads
                If (nodes in cluster-head < 3)
                        ClusterNetwork();
```

```
                Break;
            EndIf
        EndFor
        EndIf
}

//Function definition for CheckForIntrusion(NodeId, DataPacket);
CheckForIntrusion()
{
        If(DataPacket.NextHop is Not in CurrentCluster)
            ProcessInterClusterIntrusionDetection(NodeId, DataPacket);
        EndIf
        Else
            If (DosAttackDetected)
                Temp ← (Cluster-Head.Pr-DOS * Node(NodeId).Pr-DOS /
                Cluster-Head.Tr AND Cluster-Head.Pr-DOS);
                Prob. That the node is DOS intrusive ← Cluster-Head.Tr * Temp;
            EndIf
            If (Spoofing Detected)
                Temp ← (Cluster-Head.Pr-SP* Node(NodeId).Pr-SP/ Cluster-
                Head.Tr AND Cluster-Head.Pr-SP);
                Prob. That the node is Spoofing ← Cluster-Head.Tr * Temp;
            EndIf
```

```
        If (Packet Misrouting Detected)
                Temp ← (Cluster-Head.Pr-PMR * Node(NodeId).Pr-PMR /
                Cluster-Head.Tr AND Cluster-Head.Pr-PMR);
                Prob That the node is misrouting packet ← Cluster-Head.Tr *
                Temp;
        EndIf
    }
    SendDataToNeighbouringCluster();
    If Probability is greater than Threshold, mark that node as intrusive. And Update
neighboring ClusterHeads.
}

ProcessInterClusterIntrusionDetection(NodeId, DataPacket);
{
    Cluster-Head ← GetClusterHead(NodeId);
    NeighboringClusterHeads ← Cluster-Head.Neighbors;
    PeerClusterHead ← InformPossibleAttack(NeighboringClusterHeads, Cluster-
Head);

    If (DosAttackDetected)
            Temp1 ← (Cluster-Head.Tr  * Cluster-Head.Pr-DOS *
            Node(NodeId).Pr-DOS / Cluster-Head.Tr AND Cluster-Head.Pr-
            DOS);
```

Temp 2 ← PeerClusterHead.Tr * PeerClusterHead.Pr-DOS * Node(NodeId).Pr-DOS / PeerClusterHead.Tr AND PeerClusterHead.Pr-DOS);

Prob. That the node is DOS intrusive ← Temp1 * Temp 2;

EndIf

If (Spoofing Detected)

Temp1 ← (Cluster-Head.Tr * Cluster-Head.Pr-SP * Node(NodeId).Pr- SP / Cluster-Head.Tr AND Cluster-Head.Pr-SP);

Temp 2 ← PeerClusterHead.Tr * PeerClusterHead.Pr- SP * Node(NodeId).Pr- SP / PeerClusterHead.Tr AND PeerClusterHead.Pr- SP);

Prob. That the node is SP intrusive ← Temp1 * Temp 2;

EndIf

If (Packet Misrouting Detected)

Temp1 ← (Cluster-Head.Tr * Cluster-Head.Pr-PMR * Node(NodeId).Pr- PMR / Cluster-Head.Tr AND Cluster-Head.Pr-PMR);

Temp 2 ← PeerClusterHead.Tr * PeerClusterHead.Pr- PMR * Node(NodeId).Pr- PMR / PeerClusterHead.Tr AND PeerClusterHead.Pr- PMR);

Prob. That the node is PMR intrusive ← Temp1 * Temp 2;

EndIf

VITA

Pranav Kulkarni

Candidate for the Degree of

Master of Science

Thesis: DESIGN OF HIERARCHICAL INTRUSION DETECTION UNIT FOR AD-HOC NETWORKS BASED ON BAYESIAN NETWORKS

Major Field: Computer Science

Biographical:

Education:
Completed the requirements for the Master of Science in computer Science at Oklahoma State University, Stillwater, Oklahoma in May, 2008.

Completed Bachelor of Science in Information Technology from Shivaji University, Kolhpur, India in July 2003.

Experience:

Software Engineer, Smartmax Software Inc.	Jul 05 to May 06
Software Developer, Screenplay Inc.	May 06 to Nov 06
Software Development engineer in Test, Volt	Feb 07 to Oct 07
Software Development engineer in Test, Microsoft	Oct 07 till date

Name: Pranav Kulkarni Date of Degree: May, 2008

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: DESIGN OF HIERARCHICAL INTRUSION DETECTION UNIT FOR AD-HOC NETWORKS BASED ON BAYESIAN NETWORKS

Pages in Study: 58

Candidate for the Degree of Master of Science

Major Field: Computer Science

Scope and Method of Study: Mobile ad-hoc networks (MANETs) are susceptible to security attacks due to their limited resources and the wireless medium they operate in. These attacks range from snooping to passive eavesdropping to active interference and DOS attacks. Every node should be equipped to deal with all kinds of attacks since there is no central security mechanism. Hence there should be a mechanism for intrusion detection along with trust establishment and secured routing. In this thesis we propose a hierarchical intrusion detection system based on Bayesian networks. Our work builds on the trust model proposed in [1] to establish trust between peers in the network. To detect an intrusion, a subset of nodes is selected for detecting an intrusion. The detecting node snoops on the network and constructs a Bayesian network in order to detect an intrusion. Based on the information collected by the detecting node, the intruding node is identified.

Findings and Conclusions: Simulations results show that depending on the trust and the maliciousness of the node, the algorithm efficiently detects intrusive nodes and excludes them from utilizing the network resources. Furthermore due to the conditional probability methods used in conjunction with Bayes Theorem, even if some innocent node misbehaves sometimes, it will not be treated as an intrusive node immediately. The proposed approach is therefore a more robust and accurate method for intrusion detection

ADVISER'S APPROVAL: Johnson P. Thomas