TRUST WORTHY CONTENT BASED ROUTING

IN MOBILE AD-HOC NETWORKS

By

SWATHI NARAYANAM

Bachelor of Computer Science

Bangalore University

Bangalore, Karnataka

1997

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

TRUST WORTHY CONTENT BASED ROUTING

IN MOBILE AD-HOC NETWORKS

Thesis Approved:

Dr. Johnson Thomas

Thesis Adviser

Dr. George Hedrick

Dr. Istvan Jonyer

Dr. A. Gordon Emslie

Dean of the Graduate College

ACKNOWLEDGEMENT

I sincerely thank my advisor Dr. Johnson P. Thomas, for the encouragement, guidance and support he provided throughout the entire duration of my thesis. I would also like to thank my committee members Dr. George Hedrick and Dr Istvan Jonyer for serving on my committee.

The Computer Science Department, staff and faculty alike deserve special mention for the continual assistance in the technical areas that I required throughout the entire period of my education here.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Modern distributed computing demands not only scalability, as witnessed by the Internet, but also an unprecedented degree of adaptability to dynamic conditions. Mobile computing is evidence of this trend. The mobility of network nodes undermines many of the traditional assumptions of distributed systems. Topology becomes fluid as hosts move and yet retain the ability to communicate wirelessly. Communication occurs over a shared media that is not only unreliable, but also largely unpredictable as it strongly depends on the characteristics of the local environment hosts. Therefore, applications frequently experience disconnection, which is no longer just a network accident, but rather, often induced deliberately for long periods of time to save power. Other modern distributed scenarios raise similar issues in terms of dynamicity: peer-to-peer networks and sensor networks come to mind.

Coping with these demands is a challenging task. In recent years, the publish-subscribe paradigm has emerged as a promising and effective way to tackle many of these issues. The implicit and asynchronous communication paradigm that characterizes publish-subscribe supports a high degree of de-coupling among the components of a distributed application. In principle, it is possible to add or remove one component without affecting the others—only the dispatcher, the element in charge of collecting

subscriptions and routing messages, needs to be aware of the change. Clearly, this form of decoupling is desirable in a scenario where the set of available components undergoes continuous change as in mobile ad hoc networks.

A Mobile Ad Hoc Network (MANET) is a dynamic collection of wireless mobile devices that is able to communicate and move at the same time through dynamic wireless links. Neither preexisting infrastructures nor centralized administration functions; are required thus self-organization and adaptiveness are important properties. MANETs represent a concrete example of support for pervasive computing.

One of the main issues in MANETs is how to provide the application layer with suitable communication abstractions for the very dynamic nature of the underlying communication network. Content-based publish-subscribe (cb-ps) is a very appealing candidate for such dynamic contexts since it offers a flexible many-to-many communication pattern that decouples components of a distributed application in time, space, and flow.

Content-based publish-subscribe routing is a routing service where by the flow of messages from senders to receivers is driven by content of the messages, rather than by explicit addresses assigned by senders and attached to the messages. Using a content-based routing service, receivers declare their interest by means of selection predicates,

while senders simply publish messages. The dispatcher service consists of delivering to any and all receivers each message that matches the selection predicates declared by those receivers. In the content-based routing model message content is structured as a set of attribute/value pairs, and a selection predicate is a logical disjunction of conjunction of elementary constraints over the values of individual attributes. For example a message might have the following content

[class="E-mail alert". severity=9. device-type="server" alert-type='software failure"]

which would match a selection predicate such as this:

[aIert-type="intrusion" ^ severity>4 v cIass="E-mail alert" ^device-type=” server”]

Content based ad Hoc networks can only exists if the nodes demonstrate a cooperative behavior. However, this is always not true, and there may always exist malicious nodes that aim to eavesdrop on, corrupt, or disrupt the network traffic. As routing protocols play a major role in the communication set-up, it is vital that the protocols have a consistent and accurate performance. A number of such protocols were thereby developed to secure the routing process. (A comparison of these protocols was carried out by Pirzada and McDonald [1] and it revealed that all the secure routing protocols were dependent on a central trust authority for implementing traditional cryptographic algorithms. All the protocols just gave the assurance of either the presence of 100% security or its absence. None of these had an intermediate level of security protection. As authentication is one of the initial requirements of a secure channel, the

nodes were required to be in possession of preshared keys or digital certificates. This requirement of a central trust authority and preconfiguration is neither practical nor feasible in an ad-hoc network. To distinguish this environment the term “managed ad-hoc networks” was introduced in which the nodes could be initially configured before the network was established. This is in contrast to the actual aim of ad-hoc networks, which targets to establish an improvised network.) We call such a network a “Pure ad-hoc network”, which has no assumed infrastructure and is created on the fly. We also introduce the notion of trust in ad-hoc networks rather than inclusion of regular cryptographic schemes. By computing trust levels from the inherent knowledge present in the network, the trustworthiness of routes can be computed. The routes computed through this mechanism may not be secure but certainly have an accurate measure of their reliability.

This thesis is focused on introducing a trust model suitable for Content Based Ad-hoc networks. Trust Model is developed by listening to incoming and outgoing messages. Knowledge of how these messages are modified is used to develop trust tables. For message forwarding these trust tables are used to decide which messages need to be ignored or forwarded depending on threshold trust levels in the tables, thus developing a trust mechanism.

In chapter 2, a literature review describes the background of the thesis area and a detailed

description of previous models is given. In chapter 3, a more detailed view of the proposed architecture, the details about its operation and the security features implemented are provided. In chapter 4 we validate our approach using simulation and present the results of simulation. The thesis concludes in chapter 5.

CHAPTER II

REVIEW OF LITERATURE

2.1 Publish-Subscribe Routing Strategies An Overview

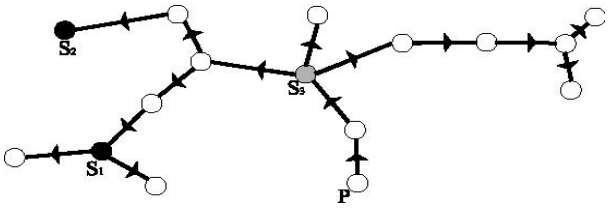
Distributed applications exploiting publish-subscribe middleware are organized as a collection of autonomous components, the clients, which interact by publishing messages and by subscribing to the classes of messages they are interested in. The core component of the middleware, the dispatcher, is responsible for collecting subscriptions and forwarding messages from publishers to sub-scribers. This scheme results in a high degree of decoupling among the communicating parties. These ideas have been recently popularized by a wealth of systems, each interpreting the publish-subscribe paradigm in a different way.

The first point of differentiation is expressiveness of the subscription language drawing a line between subject-based and content-based systems. In the first case, subscriptions contain only the name of a class of messages—usually called subject, channel, or topic—chosen among a set of predefined classes. Instead, in content-based systems the selection of a message is determined entirely by the client, which uses expressions (often called Predicates) that allow sophisticated matching on the message content

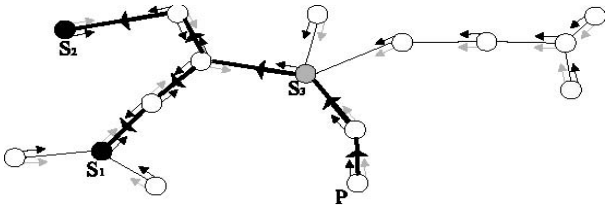
The second point of differentiation is the architecture of the dispatcher, which

can be either centralized or distributed. In this middleware, a set of brokers (see Figure 1) are interconnected in an overlay network and cooperatively route subscriptions and messages sent by clients connected to them, therefore increasing the scalability of the system.

There are several tree based routing strategies available in the literature.



(a) Message forwarding



(b) Subscription forwarding

Figure 1:Public-subscribe routing strategies

The simplest approach is message forwarding in which a published message is forwarded by a broker to all the others along the dispatching tree fig1a. However, subscriptions are never propagated beyond the broker receiving them. This broker stores these subscriptions in a subscription table that is used to determine which clients, if any, should receive incoming messages. Message forwarding generates high overhead

since messages are sent to all brokers regardless of the interests of the clients attached to them.

An alternative strategy, called subscription forwarding fig1b, limits this overhead by spreading knowledge about subscriptions throughout the system. When a broker receives a subscription from one of its clients, not only does it store the associated filter in its subscription table as in message forwarding, but it also forwards it to all the neighboring brokers. During this propagation, each dispatcher behaves as a subscriber with respect to its neighbors. Consequently, each of them records the predicate (filter) associated with the subscription in its own subscription table and reforwards it to all its neighboring dispatchers except the one that sent it to. This process effectively sets up routes for messages through the reverse path followed by subscriptions.

Hierarchical forwarding strikes a balance between the two fore mentioned strategies by assuming a rooted tree topology. Subscriptions are forwarded towards the root to establish the routes that published messages follow downstream towards subscribers. Messages, in fact, are always propagated upstream up to the root, and flow downstream along the tree only if a matching subscription has been received from the corresponding sub-tree.

In Fig.1 the above strategies are compared by showing the same setting,

characterized by a distributed dispatcher composed of 16 brokers. Two of them, namely $S_1$and S_2, have components connected to them (not shown to avoid cluttering the figure) that subscribed to the same predicate, represented as a black color, while broker S_3 received a “gray” subscription. Finally, P broker received a message matching the black predicate but not the gray one. The path followed by this message is shown through thick, directed lines, while black and gray arrows represent the content of subscription tables. More specifically, each broker has a colored arrow oriented towards another broker if it received the corresponding subscription from that broker. Figure 1(a) shows how message forwarding incurs in the highest overhead at publishing time, while it does not require subscriptions to be propagated. Subscription forwarding (Fig. 1(b)) fills the subscription tables of each broker but offers the best performance at publishing time.

Apart from the three strategies above, which are the most common especially in the presence of content-based subscription languages, other, more complex strategies are possible, in which brokers are connected in a more complete graph (a complete graph is a graph in which each pair of graph vertices is connected by an edge) to improve routing efficiency and increase system availability and fault tolerance.

2.2. Proximity-Driven Routing: An Overview

Message routing based on a distributed set of brokers interconnected in an overlay dispatching network is hard to implement efficiently in a MANET due to the cost

required to cope with the frequent changes in the topology of the physical network. To succeed in a MANET, and particularly in those including fast moving nodes, a cb-ps protocol should not require any pre-defined network-wide structure. Starting from this observation [2]developed a diffusion protocol, dubbed proximity-Driven Routing, whose general concepts are described in this section. Details are given in the next section

2.2.1 Assumptions

In our description we assume that the MANET is composed of *N* mobile nodes, each running a broker. When necessary, to stress the difference between nodes and brokers, we will use the notation n_i to indicate the i-th mobile node of the network, and b_i to refer to the broker running on that node.

Each broker b_i acts as an entry point to the cb-ps dispatching service for every application running on node n_i. When a component running on a node n_i wants to receive some message, it subscribes to b_i, which then stores the predicate associated with the subscription into its subscription table. Similarly, to publish a message, a component running on a node n_i sends it to the broker b_i. The protocol does not rely on any network layer protocol; rather it only assumes the availability of a local broadcast communication primitive, which allows a node to unreliably send a message to all its one-hop neighbors via a single transmission. Finally, it assumes that the interests of all the application components connected with a broker b_i can be condensed into a single predicate, which reflects the content of b_i's subscription table.

2.2.2 The Protocol

Let now consider how the basic message forwarding scheme works. Each broker b_i periodically broadcasts a beacon message containing the predicate that summarizes its own subscription table. A broker b_j, which is adjacent with b_i, receives this message and stores the predicate together with the time it received the beacon into its proximity table. This mechanism allows each broker to determine the time elapsed since it lost contact with any other broker. This value, which is infinite if the two brokers never came in contact and zero if they are still adjacent, is the basis to calculate the proximity value(or simply "proximity") p_{ji} of b_j with respect to b_i.

Each message *m* carries a destination list: the (estimated) list of brokers interested in receiving the message, each coupled with the lowest proximity computed by the brokers that forwarded the message so far. As an example, the destination list of a message *m* includes a couple <*i*, *p*> if broker b_i is known to be interested in receiving the message (i.e. *m* matches a subscription issued by some subscriber attached to b_i) and p is the lowest proximity from b_i calculated by all the brokers that forwarded *m*. The message has also a unique network-wide identifier provided by the source broker, it will be referred to, with the notation *m.id*.

Suppose now that at time *t* the broker b_i receives a message *m* for the first time. It will resend the message if (i) it is aware of some new broker not mentioned in the destination list carried by *m* or (ii) its proximity table holds for some broker b_k a

proximity lower than that associated to the same broker b_k into *m*'s destination list.

Such a condition is in general not sufficient to trigger the actual transmission of the message. The broker b_i, in fact, schedules the transmission of the message after a delay proportional to P_{ik} (the lowest value is considered if such a condition holds for more than one broker, see later). If during such a time interval it doesn't hear the same message (i.e. a message with the same identifier) again, then the transmission will take place. Otherwise b_i silently drops the message. The rational behind this decision is to avoid that two adjacent brokers will forward the same message and also to let brokers closest to some destination to "suppress" transmission of adjacent brokers less close.

To clarify this basic mechanism, let us consider Fig. 2, which shows a set of nodes (the black circles) together with their transmission ranges (the gray circles surrounding the node). Imagine broker b_o publishes a message matching broker b_4's subscriptions. The message is sent via broadcast and received both by b_1 and b_2. Assume that b_o and b_4 have never came in contact so that the destination table carried by m is initially empty. Assume that b_2 missed p_{24}=5 beacons from b_4. The broker b_2 schedules the transmission with some delay proportional to 5. However, b_1 is adjacent to b_4(i.e., P_{14}=0) and immediately sends the message. Broker b_2, on receiving the message from b_1 aborts the scheduled transmission and silently drops *m*. Moreover, since the proximity carried by the message sent by b_1 is zero, the broker b_3 ignores the message (by definition zero is the lowest possible proximity).

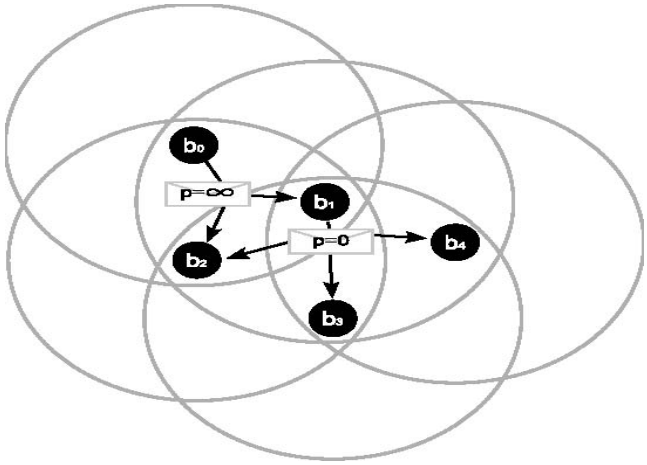


Fig. 2. The basic coordination mechanism.

2.2.3 Protocol Details

Each broker maintains the following data structures:

- A subscription table that holds information about the subscriptions issued by application components running on the same node. This table is organized as an array st of pairs <*pred*, *id*>, where *pred* is the predicate carried by a subscription and *id* is the identifier of the component that issued the subscription.
- A proximity table organized as an array *pt* of triples <*id, pred, time*>, where *id* is the identifier of a broker, *pred* is the predicate received from that broker, which summarizes its subscription table, and *time* is the time when the predicate was received.

Every Δt seconds each broker bi beacons a summary of the predicates stored into its subscription table using a broadcast packet. A broker b_j that is within the transmission

range of b_i receives such a beacon and updates its proximity table. If the same predicate was already received from the same broker, then the entry is refreshed, i.e. the time associated to the entry is set to the current time. Otherwise a new element is appended to the table. After a timeout, experimentally set to $10\,\Delta T$ seconds, entries are deleted from the table (procedure cleanup in Fig. 3). In other words, information about brokers for which more than 10 beacons have been missed, are dropped. This reflects the general intuition, also confirmed by the model provided in the next section, that too large proximity values are not correlated with the effective distance between brokers.

The information stored in the proximity table, together with the fact that the beacon interval ΔT is known globally, allows each broker b_i to calculate the proximity value p_{ij} at time *t* with respect to any other broker b_j as follows: p_{ij} is infinite if b_j is not present in b_i's proximity table; otherwise it is a value in the range [0..1] calculated as the number of b_j's beacons missed by b_i divided by 10.

Remembering from the previous section that each message carries a unique identifier and a destination list composed of couples *<id, proximity>*, we can describe how message forwarding proceeds. On receiving a message m a broker checks if the same message, i.e., a message with the same identifier has been received before. If this is the case, the message is removed from the list of messages scheduled for trasmission (if present) and it is dropped without any further processing. If m was never received before

then the broker checks if it matches some predicate in its subscription table. If it does, the broker delivers *m* to the corresponding subscriber and set the proximity for itself into *m*'s destination list to 0. This step will avoid triggering further transmissions aiming at hitting the broker, as clarified next. Further, the broker determines if it has to re-forward the message. This happens when *m* matches at least a predicate advised by a broker b_i such that: (1) b_i doesn't belong to the destination list of the message or (2) the proximity value for b_i computed by the receiving broker according to its proximity table is strictly

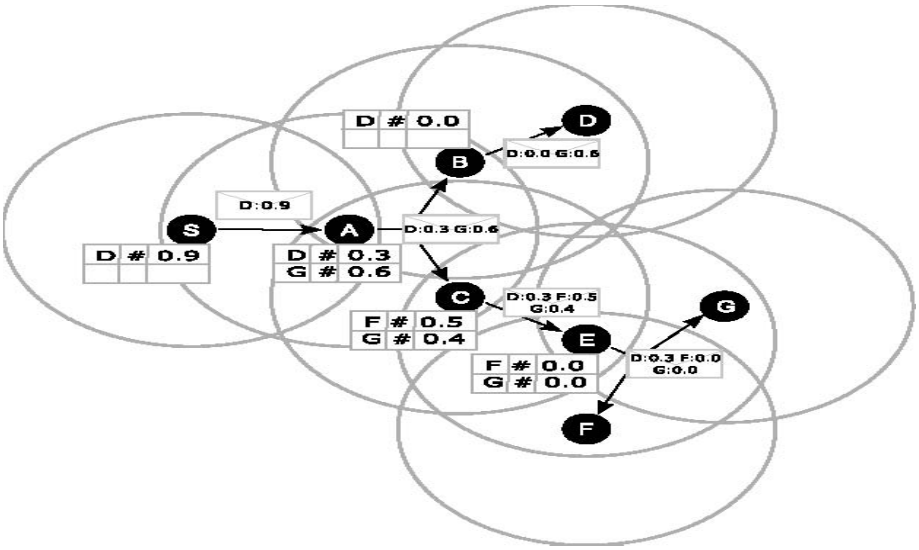


Fig. 3. An example of message routing

lower than the one carried in the message.

In both cases the retransmission of the message *m* is scheduled after a delay proportional to the proximity for b_i owned by the receiving broker. When more than one broker exits that satisfies the conditions above, the delay is determined by the lowest

proximity.

If none of the above cases hold, the message should be dropped, but to increase delivery at the price of some more traffic, a new chance is given to the message for being forwarded. To this end, a message also carries an integer value, called the *credit* of the message, which represents the number of times a broker can force the retransmission of the message despite the fact that the conditions stated above about proximity do not hold. If such a case occurs, the message is scheduled for transmission with the delay associated to the maximum possible proximity value, i.e. *one*. This way, forwarding due to credit, tends to be cancelled by forwarding due to proximity.

Figure 4 portraits an example of message forwarding. The proximity table of a node is reported close to the node, while messages show the destination list they carry. For the sake of simplicity instead of storing the absolute time when the node received a beacon message, the last column of the proximity table stores the proximity value computed as explained above.

Suppose broker *S* generates a message matching subscriptions on brokers *D*, *F*, and *G*. *S* is only aware of the subscriptions at *D*, for which it holds a proximity of 0.9. It then sends the message with destination list D : 0.9.On receiving the message, broker *A* decides to forward it. Indeed, it knows another broker, broker *G*, which is interested in the message. Moreover, the proximity for *D* calculated by *A* is lower than 0.9. Brokers *B*

and *C* both receive the message sent by *A*. Broker *B* re-forwards the message since it calculates a proximity 0.0 for *D*. Similarly, broker *C* re-forwards the message because it is aware of new broker *F* and also has a proximity for *G* (0.4) lower than that included into the destination list of the message (0.6). Finally, broker *E* re-forwards the message since it calculates two proximity values for *F* and *G* lower than those included in the message against 0.5 and 0.4, respectively.

The execution and survival of content based ad-hoc networks is solely dependent upon the cooperation and trusting nature of its nodes. However, this naive dependency on intermediate nodes makes Content based ad-hoc network vulnerable to passive and active attacks by malicious nodes. We next look at security in secure ad-hoc networks .

2.3 Trust and Security

Trust and security are two tightly interdependent concepts that cannot be desegregated. Trust establishment in ad-hoc wireless networks is still an open and challenging field. Ad-hoc networks are based on naive “trust-your-neighbour” relationships. These relationships originate, develop and expire on the fly and have usually short life spans. As the overall environment in such a network is cooperative by default, these trust relationships are extremely susceptible to attacks. For a number of reasons, including better service, selfishness, monetary benefits or malicious intent, some nodes can easily mould these relationships to extract desired goals. To overcome these problems, trust has been established in ad-hoc networks using a number of assumptions

including pre-configuration of nodes with secret keys, or presence of an omnipresent central trust authority. In our opinion, these assumptions are against the very nature of ad-hoc networks, which are supposed to be improvised and spontaneous. We categorize these networks into “managed ad-hoc networks” and “pure ad-hoc networks”.

According to Mayer, Davis and Schoorman [10] trust is defined as “the willingness of a party to be vulnerable to the actions of another party based on the expectation that the other party will perform a particular action important to the trustor, irrespective of the ability to monitor or control the party”. Jøsang [11] defines trust in a passionate entity (human) as the belief that it will behave without malicious intent and trust in a rational entity (system) as the belief that it will resist malicious manipulation. Trust in entities is based on the fact that the trusted entity will not act maliciously in a particular situation. As no one can ever be absolutely sure of this fact, trust is solely dependent on the belief of the trustor. The derivation of trust may be due to direct trust based on previous similar experiences with the same party, or in-direct trust based on recommendations from other trusted parties. Trust is also time dependent, it grows and decays over a period of time. A pure ad-hoc network closely resembles this human behavior model, where a number of people/nodes that have never met each other, are able to communicate with each other based on mutual trust levels developed over a period of time. Trust cannot be treated as a property of trusted systems but rather it is an assessment based on experience that is shared through networks of people (Denning[12]).

As in real life, trust levels are determined by the particular actions that the trusted party can perform for the trustee. Similarly trust levels can be computed based on the effort that one node is willing to expend for another node. This effort can be in terms of battery consumption, packets forwarded or dropped or any other such parameter that helps to establish a mutual trust level. A trust model that is based on experience alone may not be secluded from attacks in an ad-hoc network but it can identify routes with a certain measure of confidence

2.3.1 Specific Ad hoc Network attacks

2.3.1.1 Attacks on Wireless Networks

Two kinds of attacks can be launched against ad-hoc networks [13], *passive* and *active*. In passive attacks the attacker does not disturb the routing protocol. It only eavesdrops on the routing traffic and endeavors to extract valuable information like node hierarchy and network topology . In active attacks, the aggressor node has to expend some of its energy in order to carry out the attack. Nodes that perform active attacks with the aim of disrupting other nodes are considered to be malicious, In active attacks, malicious nodes can disrupt the correct functioning of a routing protocol by modifying routing information, by fabricating false routing information or by impersonating nodes.

2.3.1.2 Attacks Using Modification

Attacks using modification are generally targeted against the integrity of routing

computations. By modifying routing information an attacker can cause network traffic to be dropped, redirected to a different destination, or take a longer route to the destination increasing communication delays. An example is for an attacker to send fake routing packets to generate a routing loop, causing packets to pass through nodes in a cycle without getting to their actual destinations, consuming energy and bandwidth. Similarly, by sending forged routing packets to other nodes, all traffic can be diverted to the attacker or to some other node. The idea is to create a *black hole* by routing all packets to the attacker and then discarding it. As an extension to the black hole, an attacker could build a *grey hole*, in which it intentionally drops some packets but not others, for example, forwarding routing packets but not data packets

2.3.1.3 Attacks Using Fabrication

Fabrication attacks are performed by generating false routing messages. These attacks are difficult to identify as they are received as legitimate routing packets. The *rushing attack* is a typical example of malicious attacks using fabrication. This attack is carried out against on-demand routing protocols that hold back duplicate packets at every node. An attacker rapidly spreads routing messages all through the network, suppressing legitimate routing messages when nodes discard them as duplicate copies.

2.3.1.4 Attacks Using Impersonation

A malicious node can initiate many attacks in a network by masquerading as

another node (spoofing). Spoofing occurs when a malicious node misrepresents its identity by altering its MAC or IP address in order to alter the vision of the network topology that a benign node can gather.

2.3.2 Trust Model

Our trust model is an adaptation of the trust model by Marsh [9] configured for use in pure ad-hoc networks. Marsh's model computes situational trust in agents based upon the general trust in the trustor and on the importance and utility of the situation in which an agent finds itself. General trust is basically the trust that one entity assigns another entity based upon all previous transactions in all situations. Utility is considered similar to knowledge so that an agent can weigh up the costs and benefits that a particular situation holds. Importance caters for the significance of a particular situation to the trustor based upon time. In order to reduce the number of variables in our model, they merge the utility and importance of a situation into a single variable called weight, which in turn increases or decreases with time. In their model they make use of trust agents that reside on network nodes. Each agent operates independently and maintains its individual perspective of the trust hierarchy. An agent gathers data from events in all states, filters it, assigns weights to each event and computes different trust levels based upon them. Each trust agent basically performs the following three functions: Trust Derivation, Quantification, and Computation.

2.3.3.1 Trust Derivation

Trust is computed based upon the information that one node can gather about the other nodes in passive mode i.e. without requiring any special interrogation packets. Vital information regarding other nodes can be gathered by analyzing the received, forwarded and overheard packets. Possible events that can be recorded in passive mode are the measure and accuracy of:

- Frames received
- Data packets forwarded
- Control packets forwarded
- Data packets received
- Control packets received
- Streams established
- Data forwarded
- Data received

the information from these events is classified into one or more trust categories. Trust categories signify the specific aspect of trust that is relevant to a particular relationship and are used to compute trust in other nodes in specific situations. For example, they might trust a particular node for the category “data forwarding” but not for the category of “accurate routes”.

2.3.3.2 Trust Quantification

Discrete representation of trust is not sufficient to clearly represent trust that normally has a continuous trend. Secure routing protocols represent trust levels by either the presence of security or its absence. Trust in ad-hoc networks is always in a fluid state and is continuously changing due to the mobility of the nodes. As the period of interaction with any node may be brief, it is imperative that the trust be represented as a continual range to differentiate between nodes with comparable trust levels. In our trust model we represent trust from –1 to +1 signifying a continuous range from complete distrust to complete trust.

2.3.3.3 Trust Computation

Trust computation involves an assignment of weights (utility/importance factor) to the events that were monitored and quantified. The assignment is totally dependent on the type of application demanding the trust level and varies with state and time. All nodes dynamically assign these weights based upon their own criteria and circumstances. These weights have a continuous range from 0 to +1 representing the significance of a particular event from unimportant to most important. The trust values for all the events from a node can then be combined using individual weights to determine the aggregate trust level for another node. Marsh [9] defines this trust T, in node *y*, by node *x*, as $T_x(y)$ and is given by the following equation:

$$T_x(y) = \sum_{i=1}^{n} [W_x(i) \times T_x(i)]$$

Where $W_x(i)$ is the weight of the i^{th} trust category to x and $T_x(i)$ is the situational trust of x in the i^{th} trust category. The total number of trust categories n is dependent on the protocol and scenario to which the trust model is being applied.

CHAPTER III

PROPOSED APPROACH

Security problems in content–based routing

The security of secure Content-based routing has not been investigated by anyone as far as we are aware. Content based routing is susceptible to a number of malicious attacks. In this thesis we identify a number of these attacks.

- Predicate modification on message from subscriber: Any node may subscribe to the message defined by a particular predicate. The predicate is forwarded to neighboring brokers which in turn forward the predicate to other brokers in the network. Finally it reaches the subscriber which takes it and changes the predicate, so that other subscribing nodes down the path never get the original message to which it has subscribed.
- Predicate modification on message from Intermediate broker: Any node may subscribe to the message defined by a particular predicate. The predicate is forwarded to neighboring brokers which in turn forward the predicate to other brokers in the network. We can assume a simple setup where there is a broker at each node. An intermediate broker on node may alter the predicate so that the subscribing node never gets the original subscribing message.
- Proximity modification on the message from subscriber: A subscriber who gets a message changes the proximity values of destination nodes. This results in

messages being diverted to a different path i.e. taking a longer route.

- Proximity modification on the message from intermediate node: An intermediate node may change the proximity value of destination nodes in the message. This results in messages being diverted to a different path.

- Proximity modification on the message from publisher: A node may add a proximity value to itself in the destination list, so that it creates a loop and overhead in the system.

3.1 Trust derivation

We assume a node is able to overhear the communications of the nodes within its signal range. From this information it can derive if the nodes it overhears are behaving as expected. In figure 4 below, node N_1 forwards the message which is received by N_3 and all its neighboring nodes. When N_3 re forwards the message to all the neighboring nodes, N_1 if still in neighborhood picks it up. It can therefore compare the outgoing messages and incoming messages. For example, if the predicate on the outgoing message is different from that on the incoming message from N_3, then N_3 is not behaving as expected.

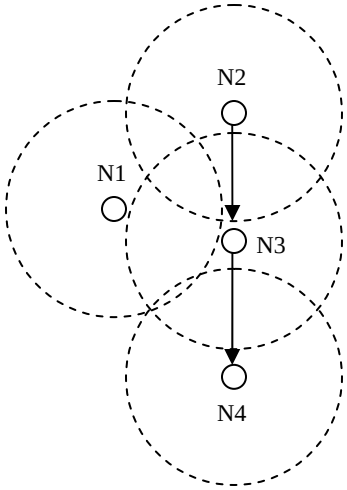


Fig. 4. The basic trust mechanism

Node N_1 maintains tables for each node it has interacted with

For example N_1 maintains the following table about N_3, N_2, N_4

Predicate modification M_S from subscribers

Nodes	No of times modified	No of times not modified
N_2	$M_{S\text{-}T}$	$M_{S\text{-}NT}$
N_3	$M_{S\text{-}T}$	$M_{S\text{-}NT}$
N_4	$M_{S\text{-}T}$	$M_{S\text{-}NT}$

Predicate modification M_I from Intermediate broker

Nodes	No of times modified	No of times not modified
N_2	$M_{I\text{-}T}$	$M_{I\text{-}NT}$
N_3	$M_{I\text{-}T}$	$M_{I\text{-}NT}$
N_4	$M_{I\text{-}T}$	$M_{I\text{-}NT}$

Proximity modification M_{PP} from publisher

Nodes	No of times modified	No of times not modified
N_2	$M_{PP\text{-}T}$	$M_{PP\text{-}NT}$
N_3	$M_{PP\text{-}T}$	$M_{PP\text{-}NT}$
N_4	$M_{PP\text{-}T}$	$M_{PP\text{-}NT}$

Proximity modification M_{PS} from subscriber

Nodes	No of times modified	No of times not modified
N_2	$M_{PS\text{-}T}$	$M_{PS\text{-}NT}$
N_3	$M_{PS\text{-}T}$	$M_{PS\text{-}NT}$
N_4	$M_{PS\text{-}T}$	$M_{PS\text{-}NT}$

Proximity modification M_{PI} from Intermediate node

Nodes	No of times modified	No of times not modified
N_2	$M_{PI\text{-}T}$	$M_{PI\text{-}NT}$
N_3	$M_{PI\text{-}T}$	$M_{PI\text{-}NT}$
N_4	$M_{PI\text{-}T}$	$M_{PI\text{-}NT}$

3.2 Trust Quantification

Trust Category Ms

$$M_S = \frac{M_{S-NT} - M_{S-T}}{M_{S-NT} + M_{S-T}} \quad \text{for } M_{S\text{-}T} + M_{S\text{-}NT} \neq 0 \text{ else } M = 0$$

$$T_3(M_S) = W(M_S) * M_S$$

Trust Category M_I

$$M_I = \frac{M_{I-NT} - M_{I-T}}{M_{I-NT} + M_{I-T}} \quad \text{for } M_{I\text{-}T} + M_{I\text{-}NT} \text{ != } 0 \text{ else } M = 0$$

$$T_3(M_I) = W(M_I) * M_I$$

Trust Category M_{PP}

$$M_{PP} = \frac{M_{PP-NT} - M_{PP-T}}{M_{PP-NT} + M_{PP-T}} \quad \text{for } M_{PP\text{-}T} + M_{PP\text{-}NT} \; != 0 \text{ else } M = 0$$

$$T_3(M_{PP}) = W(M_{PP}) * M_{PP}$$

Trust Category M_{PS}

$$M_{PS} = \frac{M_{PS-NT} - M_{PS-T}}{M_{PS-NT} + M_{PS-T}} \quad \text{for } M_{PS\text{-}T} + M_{PS\text{-}NT} \; != 0 \text{ else } M = 0$$

$$T_3(M_{PS}) = W(M_{PS}) * M_{PS}$$

Trust Category M_{PI}

$$M_{PI} = \frac{M_{PI-NT} - M_{PI-T}}{M_{PI-NT} + M_{PI-T}} \quad \text{for } M_{PI\text{-}T} + M_{PI\text{-}NT} \; != 0 \text{ else } M = 0$$

$$T_3(M_{PI}) = W(M_{PI}) * M_{PI}$$

3.3 Trust Computation

The situational trust values from all trust categories (M_S, M_P, M_I, M_{PS}, M_{PP}, M_{PI}) are then combined according to assigned weights, to determine an aggregate trust level for a particular node. Trust T in node N_3 by node N_1 is represented as $T_{N1}(N_3)$ and given by the following equation:

$$T_{N1}(N_3) = T_3(M_S) + T_3(M_I) + T_3(M_{PS}) + T_3(M_{PP}) + T_3(M_{PI})$$

The aggregate trust table is shown in Table 6.4.

Node	Pred(S)	Pred(I)	Prox(PP)	Prox(PS)	Prox(PI)
N_3	T_3 (M_S)	T_3 (M_I)	T_3 (M_{PP})	T_3 (M_{PS})	T_3 (M_{PI})

Table 6.4 : Aggregate Trust Table

3.4 Accusation messages

Node N_1 is able to form a trust value on N_3 only when it is within communication range of N_3. Assume N_1 finds N_3 to be untrustworthy. Other nodes besides N_1 may have also had experience of N_3 and found it to be untrustworthy as well. Node N_1 is not aware of the experiences of the other nodes regarding N_3 and similarly these other nodes are not aware of the experiences of node N_1 regarding N_3. Therefore, there is a need for nodes to exchange information about their experiences about other nodes. However, if there is a continuous exchange of such information regularly, the overheads in the network will be substantial. We therefore create 2 levels of trust.

- Node is behaving in a trustworthy manner. There is no exchange of trust information with other nodes
- Node is determined to be untrustworthy. An accusation message is flooded in the network and no further communications take place with the accused node.

3.5 Routing

Once a node has been accused, it should no longer be in the path. In other words,

the proximity measure should be only between nodes that are trustworthy. Each node keeps track of the nodes that have been accused. It therefore determines to route only between nodes that are trustworthy.

CHAPTER IV

IMPLEMENTATION

4.1 Secure Content Based Routing Protocol

We now develop the proposed trust mechanism with a basic message forwarding schema in content based routing. Suppose at time t broker b_i receives a message. It will resend the message

i. if the forwarding node has trust values below a threshold level in the trust table and node is not in malicious node list.

ii. if deemed trustable, the broker b_i then checks if the forwarding node is the publisher and if it has modified proximity.

iii. if deemed trustable, broker b_i checks if forwarding node is either subscriber or intermediate node and has modified predicate or has increased proximities.

If condition (i) is false it would not resend any messages from that forwarding node. If conditions (ii) or (iii) are true the broker b_i does trust calculations and updates its trust table and malicious node list identifying it as malicious and does not forward the message. If conditions (ii) and (iii) are false, broker b_i updates its trust table, goes into regular content based routing and before re forwarding messages it puts the message into the trust queue along with the proximity table at that instance.

When ever node N_1 is checks for predicate and proximity modifications by subscriber and intermediate nodes, node N_1 sees if it is the same message that it had previously sent that it is now receiving back. This is done by checking the trust queue for matching message ids and for matching forwarding node id in the proximity table of trust queue. If a node N_1 is malicious every message it receives is modified randomly and re forwarded.

4.1.1 Computation Secure Content Based Routing Protocol Algorithm

```
Message Received
{
    // Malicious node randomly modified predicate or proximity
    if ( Node type = malicious )
    {
        if ( nodetype = predicate modifier )
        {
            if ( random modification )
            {
                predicate = changed predicate;
                reforward;
            }
            else
            {
                reforward
            }
        }
       if ( nodetype = proximity modifier )
       {
         if ( random modification )
         {
           // This 0.1 is some number  that we picked to increase the proximity value
            message.dest list.proximity = message.destlist[2].proximity + 0.1;
            reforward;
         }
```

```
else
{
   reforward;
}


if ( forwarding_node_id ∈ trust table ) and (forwarding_node_id .trust values <
     threshold trust value) and ( forwarding node id ∈ malicious list)
          return
else
 {  //Proximity modification on message from publisher
          if  forwarding nodeid . type = publisher
          {
                If  ( message . destList . id = forwarding nodeid )  and
                      (message . destlist . id . proximity > 0)
                   {
                            Trustable.Update ( modified )
                            Calculate Final Trustable()
                             Malicious list .Update
                             break
                    }
                   else
                   {
                             Trustable . Update ( not modified );
                    }
  }

  if  (message . mid = trust queue . mid) and ( forwarding nodeid  ∈ trust queue .
       proximity table)
          {
             //Predicate modification by subscriber and intermediate node
              if (message . pred  == trust queue message.pred)
              {
                   if ( forwarding nodeid = subscriber )
                   {
                            Trust table . Update (not modified)
                   }
                   if ( forwarding nodeid = intermediate )
                   {
                             Trust table . Update (not modified)
                   }
```

```
else
{
        if ( forwarding nodeid = subscriber )
        {
                Trust table . Update ( modified )
                Calculate Final Trustable ()
                Malicious List . Update
                return
        }
         if ( forwarding nodeid = intermediate )
        {
                 Trust table . Update ( modified )
                 Calculate Final Trustable ()
                 Malicious List . Update
                 return
         }
 }
```

// Proximity modification by subscriber and intermediate node

```
 if( message.dest list .proximities <= trust queue destlist.proximities)
 {
   if (forwarding node id = subscriber)
   {
     Trust table . Update ( not modified )

   }
   if ( forwarding nodeid = intermediate )
   {
        Trust table .Update (not modified)
    }
 else
 {
    if ( forwarding nodeid = subscriber )
   {
     Trust table . Update (modified)
     Calculate Final Trustable ()
     Malicious List .Update
     break
    }
    if ( forwarding nodeid = intermediate )
   {
      Trust table . Update (modified)
```

```
                Calculate Final Trustable ()
                Malicious List .Update
                break
            }
    }
    }
else
{
   Content based message forwarding
   If( message queue . add (message))
   {
       trust queue . add (message)
       trust queue . add (proximity table)
   }
}
```

4.2 Simulations results

The simulation has the simulation parameters

No of Nodes - 50
Field Area - 900 * 200
Maximum speed - 1
Minimum Speed - 20
No of publishers - 2
Publishing rate - 10ms
No of subscribers - 5
ΔT - 1000ms
Message Credits - 0
Threshold - 1.0

Mobility model

All nodes are assigned random x and y coordinates and random value with increment or decrement is chosen at the start of the simulation. Once a node hits an end coordinate, which are 0, 200, or 900, the node will move in opposite direction in which it is moving . Depending on which coordinate hits end coordinate x or y is either incremented or decremented at different speeds thus making the nodes move.

Mobility model Algorithm

```
nodeinfo.x += nodeinfo.xMove; //move the x node by random value(value could be + or-)
nodeinfo.y += nodeinfo.yMove; //move the y node by random value(value could be + or-)

if(nodeinfo.x < 0.0D || nodeinfo.x > worldXSize) check to see if the x boundaries are
touched
{
    nodeinfo.xMove = nodeinfo.xMove * -1D;  //node will move in the opposite direction
    in which it is moving
    nodeinfo.x += nodeinfo.xMove;  //increase the node value by some move factor
    nodeinfo.x += nodeinfo.xMove;  // increase it again to so that we get some angle of
```

```
    movement
 }
 if (nodeinfo.y < 0.0D || nodeinfo.y > worldYSize) //similarly do this for the y cordintes
as well
{
      nodeinfo.yMove = nodeinfo.yMove * -1D; //node will move in the opposite
      direction in which it was moving
      nodeinfo.y += nodeinfo.yMove;  //increase the node value by some move factor
      nodeinfo.y += nodeinfo.yMove; //increase it again to so that we get some angle of
      movement
   }
```

All graphs are an average of 6 simulations

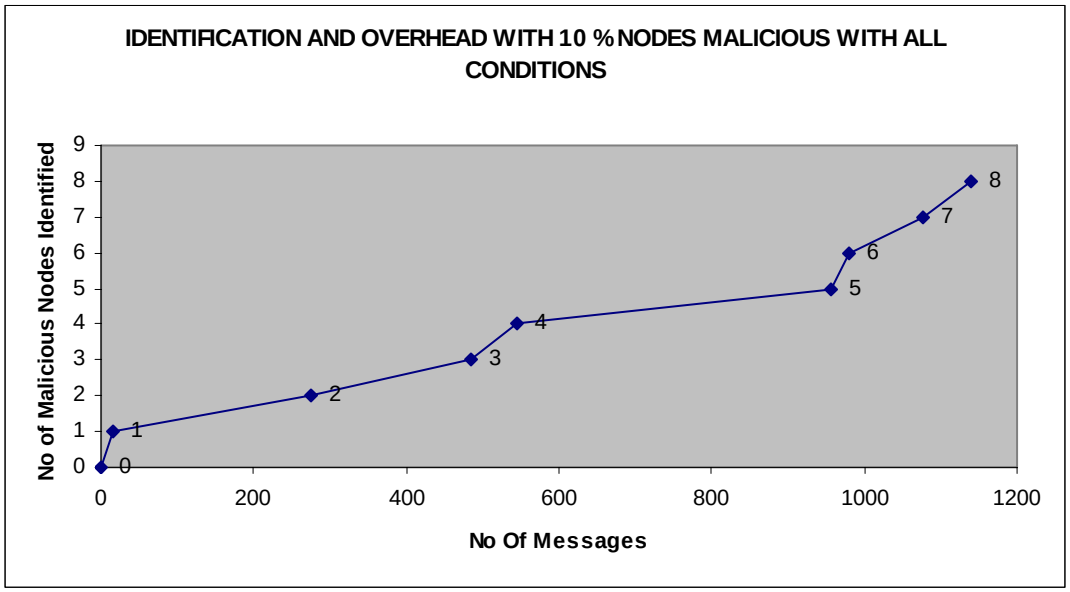


Figure 5: 10% of nodes malicious all attacks

We included the following types of attacks in our simulations:

- Predicate modification – in our simulation this was achieved by modifying the predicate of publisher, subscriber and intermediate node

- Proximity modification – in our simulation this was achieved by modifying proximity value of subscriber and intermediate node.

In the graphs below the no of messages is the total number of communication messages excluding beacon signals. These communication messages are signals passed from node to node to forward the message.

From figure 5 we can see that as the number of messages increase over time, more malicious nodes are detected. All the malicious nodes are detected within 1200 messages. Our approach therefore is very effective as it detects all the malicious nodes over time, but the proposed approach also has a high number of false positives. This is an area that needs future research. However once all malicious nodes are detected there are no more false accusations. Nodes 1,2,4,5,8 are true malicious nodes. The others are assumed to be good nodes, although detected to be bad.

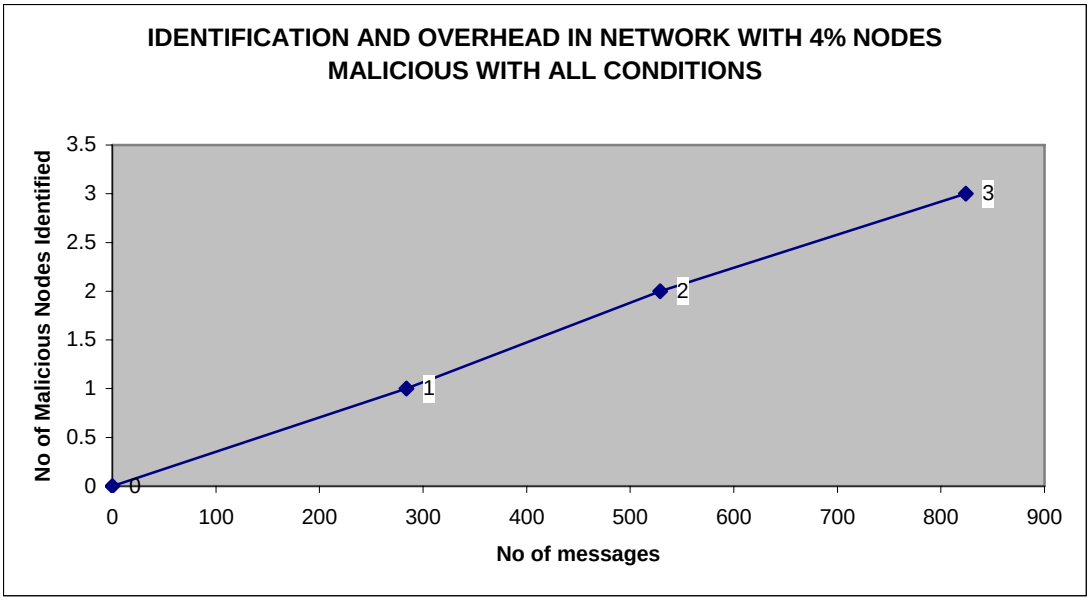


Figure 6: 4% of nodes malicious all attacks

We included the following types of attacks in our simulations:

- Predicate modification – in our simulation this was achieved by modifying the predicate of publisher, subscriber and intermediate node
- Proximity modification – in our simulation this was achieved by modifying proximity value of subscriber and intermediate node.

From figure 6 we can see that as the number of messages increase over time, more malicious nodes are detected. All the malicious nodes are detected within 900 messages. It is interesting to note that all malicious nodes were detected in equal intervals. This may be because malicious nodes were farther apart. However we do have false positives. This is are that needs future research. Our approach therefore is very effective as it detects all the malicious nodes over time. However once all malicious nodes are detected there are

no more false accusations. Nodes 1,3 are true malicious nodes. The others are assumed to be good nodes, although detected to be bad.

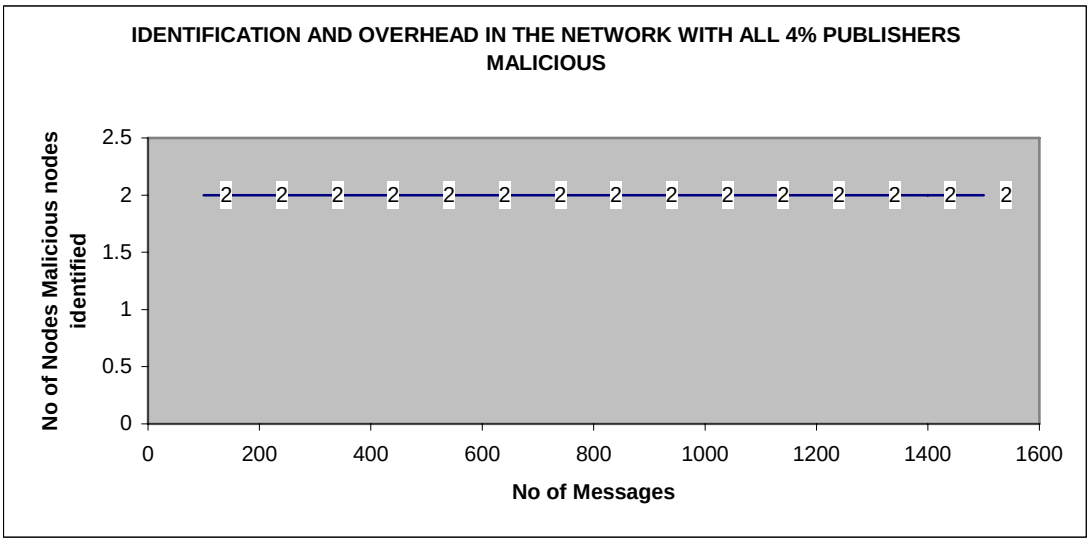


Figure 7: 4% of malicious nodes detected with publisher modification

Publisher modification attack are implemented by introducing malicious publishers into the simulation.

From figure 7 we can see that as the number of messages increase over time, more malicious nodes are detected. All malicious publishers are detected with in 200 messages. We do not have false positives in this case because there is no need for messages to go out and be returned to identify; the message being transmitted is enough to identify a malicious publisher. This is therefore quick and efficient. Here the graph is constant, because once a malicious publisher comes into contact with any node it is immediately identified.

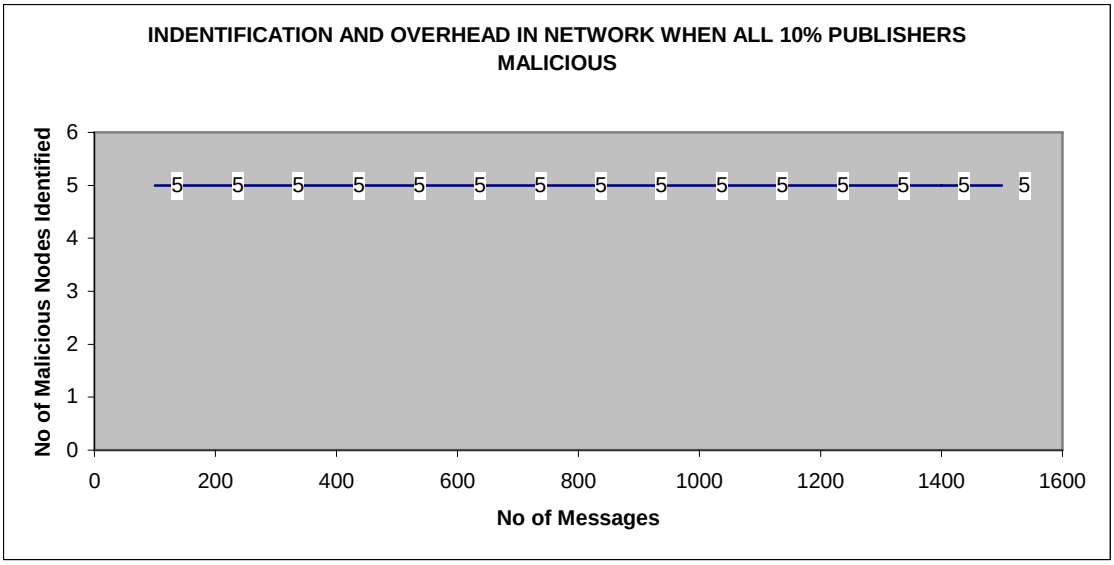


Figure 8: 10% of malicious nodes detected with publisher modification

Publisher modification attack are implemented by introducing malicious publishers into the Simulation.

From figure 8 we can see that as the number of messages increase over time, more malicious nodes are detected. All malicious publishers are detected with in 200 messages. We do not have false positives in this case because there is no need for messages to go out and be returned from to identify, just going out is enough to identify malicious publisher. This is therefore quick and efficient. Here graph is constant because once malicious publisher comes in contact with any node it is immediately identified.

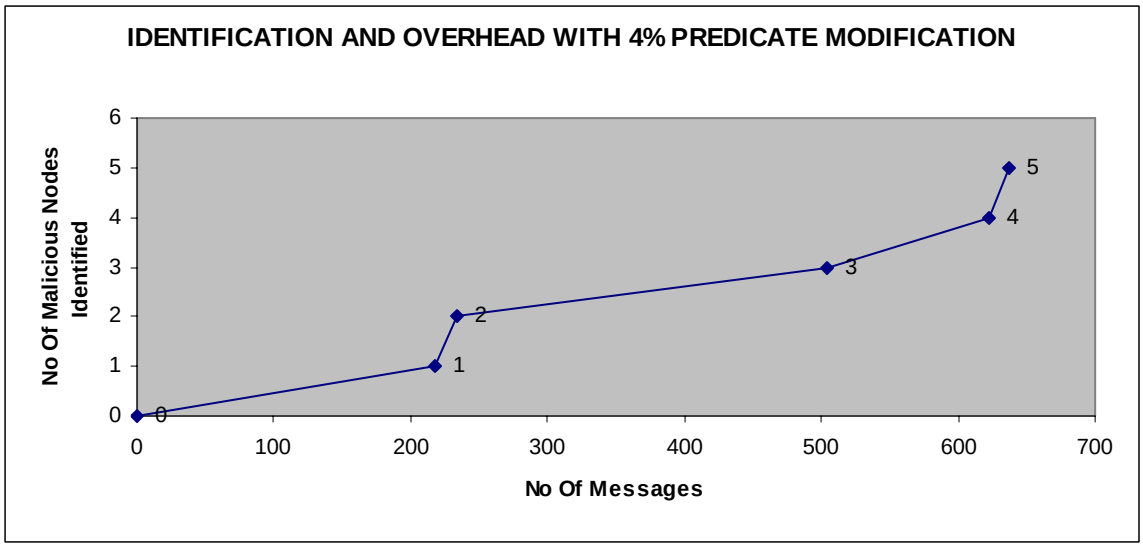


Figure 9: 4% of malicious nodes detected with all predicate modification

Predicate modification attack are implemented by introducing malicious subscribers and intermediate nodes, which modify predicate into the Simulation.

From figure 9 we can see that as the number of messages increase over time, more malicious nodes are detected. All the malicious nodes are detected with-in 700 messages. It is interesting to note that there is a sudden increase in the rate of malicious node detection at 200 to 300 and 600 to 700 messages range. This may be because a publisher is in the vicinity of many malicious and non malicious nodes, and there may be therefore be more false accusations during that time. However the number of false positives is surprisingly large. This issue is addressed later in this chapter. Although our approach is very effective as it detects all the malicious nodes over time, it also has a high number of false positives for predicate modifications. This is an area that needs future research.

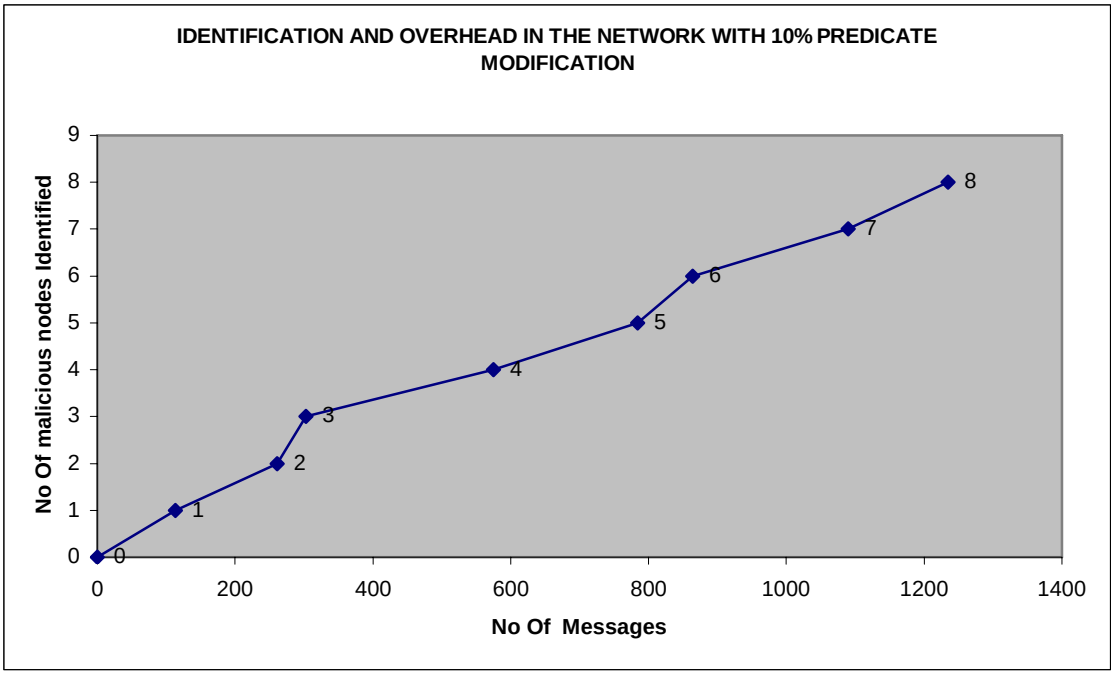


Figure 10: 10% of malicious nodes detected with all predicate modification

Predicate modification attack are implemented by introducing malicious subscribers and intermediate nodes, which modify predicate into the Simulation.

From figure 10. 1f we can see that as the number of messages increase over time, more malicious nodes are detected. All the malicious nodes are detected with in 1400 messages. It is interesting to note that all nodes are detected in about equal intervals. This may be because malicious and non-malicious nodes are evenly distributed in the field , However the number of false positives is surprisingly large. This issue is addressed later in this chapter. Although our approach is very effective as it detects all the malicious nodes over time, it also has a high number of false positives for specifically predicate modifications. This is an area that needs future research

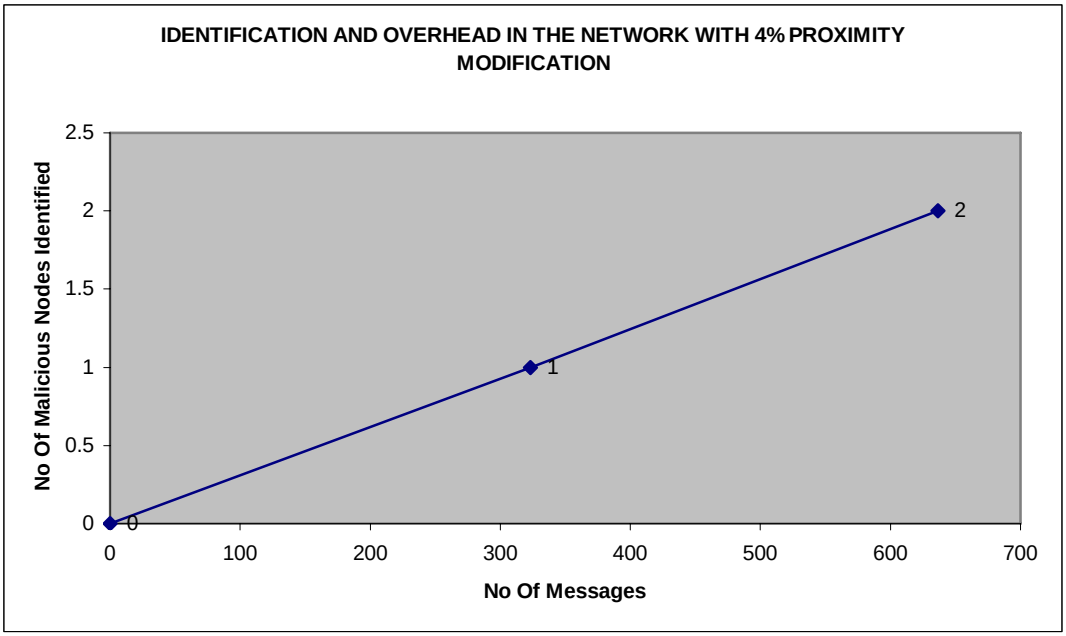


Figure 11: 4% of malicious nodes detected with proximity modification

Proximity modification attack are implemented by introducing malicious subscribers and intermediate nodes, which modify proximity into the Simulation.

From figure 11 we can see that as the number of messages increase over time, more malicious nodes are detected. All the malicious nodes are detected within 700 messages. It is interesting to note that unlike other graphs nodes are identified in equal intervals. This may be because a publisher was in the vicinity of the first malicious node around 300 messages and took some time for it to travel and get the to vicinity of another malicious node. However, we rarely have false positives in proximity modifications. This issue is addressed later in this chapter. Our approach therefore is very effective as it detects all the malicious nodes over time. Moreover, but the proposed approach is

efficient because once all malicious nodes are detected, there are no more false accusations.

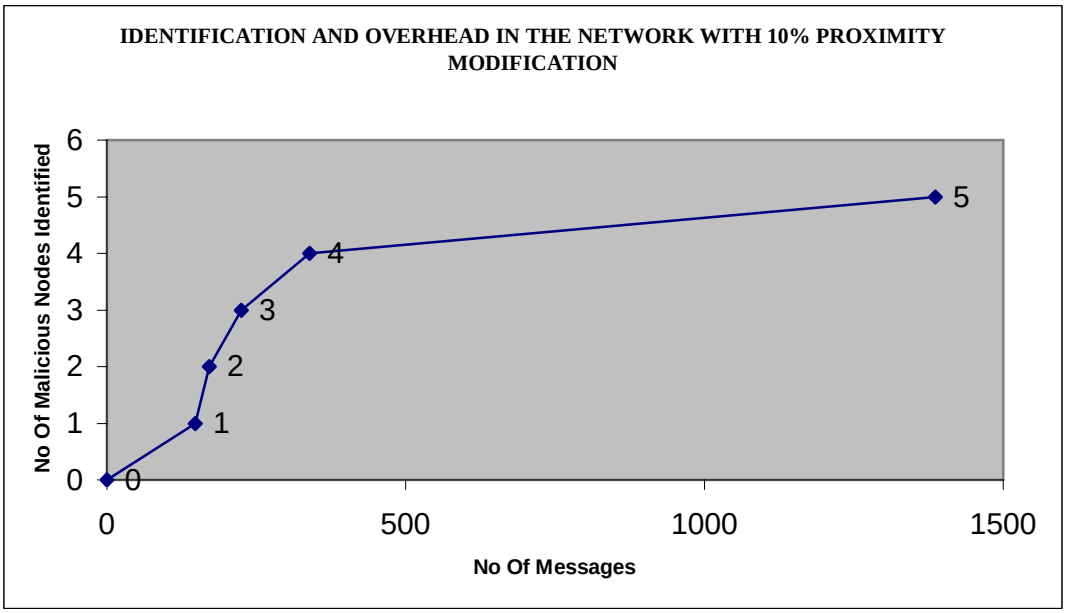


Figure 12: 10% of malicious nodes detected with proximity modification

Proximity modification attack are implemented by introducing malicious subscribers and intermediate nodes, which modify proximity into the Simulation.

From figure 12 we can see that as the number of messages increase over time, more malicious nodes are detected. Most of the malicious nodes are detected within 400 messages. It is interesting to note that all malicious nodes were detected with in a span of very few messages. This may be because a publisher was in the vicinity of malicious nodes publishing a lot of messages and lot of message exchanges took place. However

we rarely have false positives in proximity modifications. This issue is addressed later in this chapter. Our approach therefore is very effective as it detects all the malicious nodes over time, Moreover, the proposed approach is efficient because once all malicious nodes are detected and there are no more false accusations .

4.2 Trust Calculations

In this section final trust table is built by calculations based on simulation results.

Predicate modification M_S from subscribers

Nodes	No of times modified	No of times not modified
N_2	1	9
N_3	0	20
N_4	1	0

$M_I = M_{S\text{-}NT} - M_{S\text{-}T} / M_{S\text{-}NT} + M_{S\text{-}T}$ *for* $M_{S\text{-}T} + M_{S\text{-}NT} = 0$

else M = 0

$T_2 (M_S) = 1 * 9\text{-}1/9+1 = 0.8$

$T_3 (M_S) = 1 * 20\text{-}0 / 20+0 = 1.0$

$T_4 (M_S) = 1 * 0\text{-}1 / 0+1 = -1.0$

Proximity modification M_{PS} from subscribers

Nodes	No of times modified	No of times not modified

N_2	0	20
N_3	1	9
N_4	0	1

$M_{PS}= M_{PS-NT} – M_{PS-T} / M_{PS-NT} + M_{PS-T}$ for $M_{PS-T} + M_{PS-NT} = 0$

else M = 0

T_2 (M_{PS}) = 1 *20-0 / 20+0 = 1.0

T_3 (M_{PS}) = 1 *9-1 / 9+1 = 0.8

T_4 (M_{PS}) = 1 *1-0 / 1+0 = 1.0

Predicate modification M_I from Intermediate Node

Nodes	No of times modified	No of times not modified
N_5	1	8
N_6	0	20
N_7	1	0

$M_I = M_{I-NT} – M_{I-T} / M_{I-NT} + M_{I-T}$ *for* $M_{I-T} + M_{I-NT} = 0$ *else* $M = 0$

T_5 (M_I) = 1 * 8-1/8+1 = 0.77

T_6 (M_I) = 1 * 20-0/20+0 =1.0

T_7 (M_I) = 1 * 0-1/0+1 = -1.0

Proximity Modification M_{PI} from Intermediate Node

Nodes	No of times modified	No of times not modified
N_5	0	20
N_6	1	9
N_7	0	1

$M_{PI} = M_{PI\text{-}NT} - M_{PI\text{-}T} / M_{PI\text{-}NT} + M_{PI\text{-}T}$ *for* $M_{PI\text{-}T} + M_{PI\text{-}NT}$ *!= 0 else M =0* $T_5 (M_{PI}) = 1$

* 20-0/20+0 = 1

$T_6 (M_{PI})$ = 1 * 9-1/9+1 = 0.8

$T_6 (M_{PI})$ = 1 * 1-0/1+0 = 1

Proximity Modification M_{PP} from Publisher

Nodes	No of times modified	No of times not modified
N_8	1	5
N_9	0	9
N_{10}	4	0

$M_{PP} = M_{PP\text{-}NT} - M_{PP\text{-}T} / M_{PP\text{-}NT} + M_{PP\text{-}T}$ *for* $M_{PP\text{-}T} + M_{PP\text{-}NT}$ *!= 0 else M=0*

$T_8 (M_{PP})$ = 1 * 5-1 / 5+1 = 0.66

$T_9 (M_{PP})$ = 1 * 9-0 / 9+0 = 1.0

$T_{10} (M_{PP})$ = 1 * 0-4 / 0+ 4 = -1.0

TRUST CALCULATIONS

N_2 = (0.8 +1) / 2 = 0.9

N_3 = (1 +0.8) / 2 = 0.9

N_4 = (-1 +1) = 0

N_5 = (0.77 +1) / 2 = 0.88

N_6 = (1 +0.8) / 2 = 0.9

N_7 = (-1 +1) = 0.0

N_8 = 0.6

N_9 = 1

N_{10} = -1

FINAL TRUST TABLE

Nodes	**No of times modified**
N_2	0.9
N_3	0.9
N_4	0.0
N_5	0.88
N_6	0.9
N_7	0.0
N_8	0.6
N_9	1.0
N_{10}	-1.0

4.4 Reason For False Accusations:

Predicate and Proximity modification

- Node A was in the proximity of Node B. Node B is a fast moving node and therefore, within 10 beacon signals it picks up a modified predicate message from some malicious node and gives it to node A. Node A has sent the same message to all its neighbors and thought that B also got it, modified it and sent it back to it, hence it thinks Node B is malicious even though it is not.
- Node A was in proximity of Node B. Node B is fast moving node so with in 10 becon signals it goes and picks up a modified proximity message from some malicious node and gives it to node A. which apparently sent the same message to all its neighbors with at least one node having lower proximity and thought that B also got it, modified it and sent it back to it, so thinks Node B is malicious even though it is not.

CHAPTER V

CONCLUSION

We presented in this thesis an approach for establishing and managing trust in ad-hoc networks. This is not another type of hard-security cryptographic or certification mechanism. Instead it aims at building confidence measures regarding trustworthiness of nodes that are dynamically computed and modified. In ad-hoc networks where doubt and uncertainty are inherent, our trust model creates and maintains trust levels based on incoming and outgoing message mechanism. Cryptographic techniques may be applied on top of our approach to improve the security of the system. However, encrypting each message is computationally expensive and can drain the energy of a node On the other hand, the proposed approach uses simple computations and the overhead is therefore minimal. The proposed approach establishes relative levels of trustworthiness within nodes. We believe that our model will be most suited to pure ad-hoc networks where there is no trust infrastructure and the trust relationships are less formal, temporary or short-term. The main findings of our work are all nodes, which are malicious, are identified over time; however there are a high number of false positives for predicate modifications.

The proposed approach needs to be extended to reduce the number of false positives. The approach can also be extended to detect other potential modifications that can be done by a malicious node

REFERENCES

[1] Asad Amir Prazada and Chris MacDonald, "Establishing Trust in Pure Ad Hoc Networks", *Proceedings of the 27th Australian Computer Science Conference,*. pp.47-54, 2004

[2] Roberto Baldoni, Roberto Beraldi, Leonardo Querzoni, Gianpaolo Cugola, Matteo Migliavacca, "Content-Based Routing in Highly Dynamic Mobile Ad Hoc Networks", *Journal of Pervasive Computing and Communications*, Vol 1, No 4, pp.277-288, December 2005

[3] Elizabeth M Royer, Charles E.Perkins, "Multicast operation of the Ad Hoc On Demand Vector Routing Protocol", *Proceedings of the 5th annual ACM/IEEE international conference on Mobile Computing and Networking, MOBICOM,* pp.207-218, 1999,

[4] Antonio Carzaniya, Matthew J. Rutherford and Alexander L Wolf, "A Routing Scheme for Content-Based Networking", *Proceedings of IEEE INFOCOM 2004*, Vol 2, pp. 918–928, 2004,

[5] Eiko Yoneki and Jean Bacon, "An Adaptive Approach to Content-Based Subscriptionin Mobile AdHoc Networks", *Proceeding of the 2nd IEEE Annual conference on Pervasive Computing and Communications Workshops,* pp. 92-97, 2004.

[6] Gianpaolo Cugola Amy L.Murphy,Gian Pietro Picca, "Content-Based Publish-Subscribe in a Mobil Environment" Invited contribution to the book.*The handbook of Mobile Middleware*, A. Corradi and P. Bellavista eds., CRC Press, 2006.

[7] Hu Zhou, Suresh Singh, "Content based multicast (CBM) in ad hoc networks", *Proceedings of the 1st ACM international symposium on Mobile ad hoc Networking & Computing*, pp. 51–60, 2000

[8] A A Rahman, and S Hailes: A Distributed Trust Model, *Proc. of the ACM New Security Paradigms Workshop*, pp. 48-60, 1997.

[9] Marsh, S. P: Formalizing Trust as a Computational Concept. Ph.D. Thesis. Department of Mathematics and Computer Science, University of Stirling,1994.

[10] Mayer, R. C., J. H. Davis and F. D. Schoorman :An Integrative Model of Organizational Trust, *Academy of Management Executive*, 20 (3): pp.709-73,1995.

[11] Jøsang, A.: The right type of trust for distributed systems. *Proc. of the ACM New Security Paradigms Workshop*, pp.119-131,1996.

[12] Denning, D.: A new paradigm for trusted systems. *Proc. ACM New Security Paradigms Workshop*, pp. 36-41,1993.

[13] Hu, Y-C, Perrig, A. and Johnson, D. B: Ariadne : A secure On-Demand Routing Protocol for Ad Hoc Networks, *Proc of the eighth Annual International Conference on Mobile Computing and Networking*, pp. 12-23,2002

VITA

Swathi Narayanam

Candidate for the Degree of

Master of Science

Thesis: TRUSTWORTHY CONTENT-BASED ROUTING IN MOBIL AD HOC NETWORKS

Major Field: Computer Science

Biographical:

Personal Data:
Born in Bangalore, India to Mr. N.G Krishnamacharyulu and Mrs. Anjani Devi

Education:
Completed Bachelor of Engineering in Computer Science from PESIT Bangalore, India. Completed the requirements for the Master of Science in Computer Science at Oklahoma State University, Tulsa, Oklahoma in May 2008.

Experience:
Programmer in UBAH Jan 2002 to Jan 2003 , Systems Analyst Colamco 2004-2005 and Systems Analyst EPASS 2005 to date.

Name: Swathi Narayanam Date of Degree: May, 2008

Institution: Oklahoma State University Location: Tulsa, Oklahoma

Title of Study: TRUSTWORTHY CONTENT-BASED ROUTING IN MOBILE AD-HOC NETWORKS

Pages in Study: 52 Candidate for the Degree of Master of Science

Major Field: Computer Science

Abstract:

This thesis proposes a trust based security model suitable for content-based routing in Ad-hoc networks. The trust model is developed by listening to incoming and outgoing messages in passive mode. Knowledge of how these messages are modified is used to update trust tables. This work looks at a number of attacks in such network, for example, predicate modification and proximity modification. For message forwarding these trust tables are used to decide which messages need to be ignored or forwarded depending on threshold trust levels in the tables, thus developing a trust mechanism. The main finding for the research shows that all nodes which are malicious are identified over time; however, there is a high number of false positives for predicate modifications. This is an area for future research.

ADVISER'S APPROVAL: Dr Johnson Thomas

ADVISER’S APPROVAL: Dr Johnson Thomas