Spring 2011

An incentive based approach to detect selfish nodes in Mobile P2P network

Hemanth Meka

AN INCENTIVE BASED APPROACH TO DETECT SELFISH NODES IN MOBILE P2P NETWORK

by

HEMANTH MEKA

A THESIS

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN COMPUTER SCIENCE

2011

Approved by

Dr. Sanjay Kumar Madria, Advisor

Dr. Sriram Chellappan

Dr. Maciej Zawodniok

PUBLICATION THESIS OPTION

This thesis consists of the following article that has been submitted for publication as follows:

Pages 13-45 have been submitted to the 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE MASS 2011).

ABSTRACT

The growth of mobile devices led to the wide use of Mobile P2P networks. These networks are used in a wide variety of areas and hence there is lot of research in the field of mobile networks. Detecting selfish nodes is one of the research topics triggered due to the popularity of mobile P2P networks. It is necessary to detect selfish nodes in such networks to improve the efficiency of the network. In this thesis, an incentive based approach to detect selfish nodes is designed and evaluated. This approach differs from the existing work as it (i) can be used with any underlying routing protocol assuming there are no attacks due to routing protocol (ii) is able to detect selective behavior of nodes where nodes drop some packets and forward some (iii) prevents a wide variety of malicious activities or attacks by nodes in the network (iv) prevents false positives due to connectivity issues in the network. We assume the presence of some trusted nodes called Broker nodes and propose a way using which nodes in the network communicate. Each intermediate node sends a receipt to the Broker node which it uses to identify selfish nodes in the network. Each node has a currency assigned which it uses to pay others for the forwarding service. Currency of a node is changed based on the receipts sent by that node. When the currency level of a node below some threshold, it is designated as selfish node in the network. This approach is experimentally evaluated and is found to outperform some of the recent work in this area in terms of time to detect selfish nodes and overhead involved.

ACKNOWLEDGMENTS

I owe a debt of gratitude to all those who have helped me with this thesis. First of all, I would like to thank my advisor Dr. Sanjay Kumar Madria, who gave me an opportunity to work on this research. His suggestions and encouragement carried me through difficult times. His valuable feedback contributed greatly to this thesis. Secondly, I would also like to express my gratitude to Dr. Sriram Chellappan and Dr. Maciej Zawodniok for serving on my thesis committee and for taking time to review this work.

I'm grateful to all my lab mates and family members for their professional help and inputs. This project is partially funded by the Air Force Research Laboratory (AFRL) in Rome, New York and I express my gratitude to them as well.

TABLE OF CONTENTS

Page

PUBLICATION THESIS OPTION iii

ABSTRACT iv

ACKNOWLEDGMENTS v

LIST OF ILLUSTRATIONS viii

LIST OF TABLES ix

SECTION

1. INTRODUCTION 1
   1.1. ROUTING ALGORITHMS 2
   1.2. EFFICIENCY OF MOBILE P2P NETWORK 3
   1.3. MOTIVATION 4
      1.3.1. Need to Identify Selfish Nodes 4
      1.3.2. Preventing Selfish Behavior 5
      1.3.3. Trusted and Untrusted Nodes in a Network 5
2. RELATED WORK 7
   2.1. USING PROMISCUOUS MODE OF NODES 7
   2.2. USING ACKNOWLEDGEMENT 8
   2.3. USING REPUTATION OF NODES 9
   2.4. USING CREDIT/CURRENCY OF NODES 11

PAPER

I. An Incentive Based Approach to Find Selfish Nodes in Mobile P2P Network 13

ABSTRACT 13

1. INTRODUCTION 13

2. RELATED WORK 16

3. SYSTEM CONFIGURATION 19

ASSUMPTIONS 20

4. COMPLAINT BASED COMMUNICATION PROTOCOL 21
   4.1. SENDING AND RECEIVING DATA 21
   4.2. STORING AND SENDING TOKENS 24

4.3. CREATING VIRTUAL PATH AND ANALYSIS 26
4.4. IDENTIFY NODES DROPPING PACKETS 29
5. MAINTAINANCE OF NETWORK 32
5.1. CHOOSING BROKER NODE 32
5.2. CREDIBILITY OF NODES 32
5.3. PREVENTING MALICIOUS ACTIVITY 33
5.4. MAINTAINING VIRTUAL CURRENCY 34
5.5. DETECTING SELFISH NODES 35
6. SIMULATION AND EXPERIMENTAL RESULTS 35
PERFORMANCE EVALUATION 36
6.1. Time Taken to Detect Selfish Nodes vs. Selective Behavior 36
6.2. Time vs. % of Selfish Nodes Detected 38
6.3. Time Taken to Detect all Selfish Nodes vs. No. of Selfish Nodes 39
6.4. Time Taken to Detect Selfish Nodes vs. Speed of Nodes 40
6.5. Time Taken to Detect Selfish Nodes vs. Total no. of Nodes in the Network 41
6.6. No. of Control Packets Generated vs. No. of Data Packets Sent 42
7. CONCLUSION AND FUTURE WORK 43
REFERENCES 44
VITA 46

LIST OF ILLUSTRATIONS

Figure Page

1.1 Overview of Mobile P2P network 2

2.1 Path taken by packet from source to destination 7

PAPER

3.1 Overview of the network 20

4.1 Token structure generated at the source 22

4.2 Packet structure transmitted by the source 23

4.3 Algorithm to send a message 23

4.4 Algorithm for storing receipts 24

4.5 Receipt structure stored at each intermediate node and destination 25

4.6 Different types of virtual paths possible 26

4.7 Currency distribution and identifying nodes dropping packets 30

6.1 Time taken vs. selective behavior 37

6.2 Time vs. % of selfish nodes detected 38

6.3 Time taken to detect all selfish nodes vs. No. of selfish nodes in the network 39

6.4 Time taken to detect selfish nodes vs. Speed of nodes 40

6.5 Time taken to detect selfish nodes vs. Total no. of nodes in the network 41

6.6 Number of control packets generated vs. Number of data packets sent 42

LIST OF TABLES

PAPER Page

6.1 Simulation Parameters 36

1. INTRODUCTION

The growth of laptops and PDA's has led to the increased use of mobile networks. A Mobile P2P network is a network of mobile devices connected by wireless links. Each device in the network is called a node. Each node in the network can move independently in any direction, hence its links to other nodes in the network can be constantly changing and thus, it changes the structure of the network too. A mobile P2P network could also be connected to the Internet or outside world using a gateway. An overview of the Mobile P2P network is shown in Figure 1.1. Mobile P2P networks are used in battle fields for communication or data transfer as there is no infrastructure possible in battle field. They can also be used during rescue operations or disaster recovery where there is no existing infrastructure. They can also be used in offices or other buildings when infrastructure is not available. More advanced applications of mobile networks are vehicular networks where vehicles communicate among themselves about traffic or for various other purposes. These networks can also be used for data transfer or file sharing between PDA's or laptops without the presence of any network. The main advantage of mobile P2P networks is that they are very easy to create and self-configured and can also move from one place to another. The disadvantage of these networks is that all the nodes use battery power for sending and receiving messages and nodes can move out at any time. So the battery power of each node has to be preserved to increase the life time of the network and connectivity information has to be maintained.

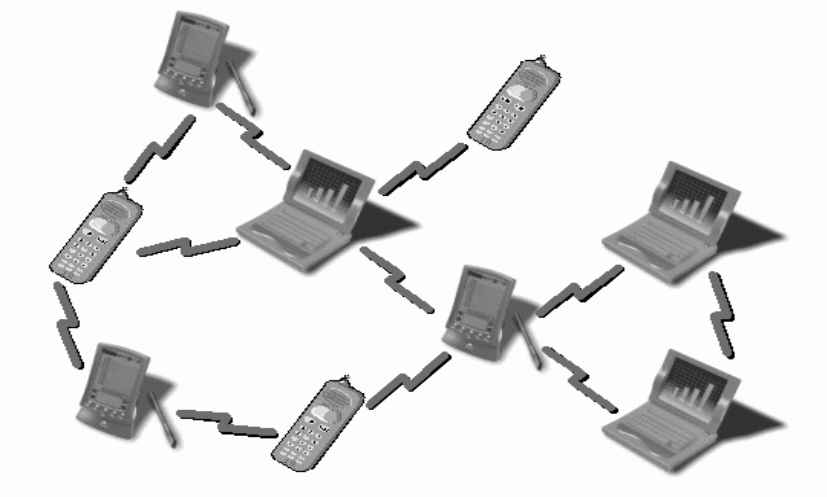

Figure 1.1 Overview of Mobile P2P network

1.1. ROUTING ALGORITHMS

In a mobile P2P network, nodes too far apart have to communicate with each other. As there is no direct connection between all nodes in the network, it is important that all the nodes in the network help each other. Each node in the network has to forward packets it receives towards the destination. There are various routing algorithms which help each node take decisions about forwarding packets. The routing algorithms are broadly classified into two types:

a. Source based routing protocol
b. Hop by hop routing protocol

In source based routing, all the decisions about the route are taken at the source before the packet is sent. The whole route to be followed by the packet is fixed at the source and attached to the message/data. Each intermediate node in the route then forwards it to the next hop based on the route attached. This reduces the load on the network as the routing decisions are taken only when a packet has to be sent, but could also increase the energy used in some cases. If an intermediate node cannot forward the packet to the next hop, a new route has to be found to forward the packet. This increases the routing time and hence delays sending packets. An example of a source based routing

is DSR [18] where RREQ is used to find the route to the destination and the route found is used to send packet to the destination.

In hop by hop based routing, the next hop towards the destination is decided at each intermediate node. The source node just sends the packet without attaching a route and each intermediate node finds the next hop to forward the packet. A routing table is maintained at each node to forward packets without delay. This increases the load on the system as next hop is maintained for each destination and is updated regularly. The delay in sending the packet is low as it is directly sent without any route discovery. An example of hop by hop routing is DSDV [19] where each node maintains a routing table and is used to forward packet depending on the destination.

Any routing protocol can be used for a mobile network based on the needs of the network. But nodes depend on battery power for longitivity, so the routing algorithms also have to make sure that energy is conserved at nodes in the network.

1.2. EFFICIENCY OF MOBILE P2P NETWORK

All the nodes in the network have to forward packets towards destinations for the network to work efficiently. The efficiency of the network can also be measured based on the packet delivery ratio as described in [6]. If many packets are dropped by intermediate nodes, the packet delivery ratio decreases and the efficiency of the network decreases as the energy spent in sending packets is wasted. The efficiency of the network and packet delivery ratio should be higher to properly utilize the network created.

$$Packet\ delivery\ ratio = \frac{No.of\ packets\ reaching\ the\ destination}{No.of\ packets\ generated\ in\ the\ network} \quad (1)$$

The packet delivery ratio may change due to environmental reasons or due to the various intermediate nodes which can selectively forward. Various nodes in the network may not forward packets towards destinations, called selfish nodes. They may either drop packets when they are not able to reach the next hop or just drop them to save battery power. The number of packets dropped by selfish nodes is much higher compared to the number of packets dropped due to environmental reasons or non-reachability. Nodes could also be involved in other malicious activity to disrupt the network. As packet drop

due to environment is very less it can be ignored, but we should stop packet drop to save battery power and also prevent other malicious activity to disrupt the network.

The packet delivery ratio also depends on the length of the path taken by packets. If the path from source to destination is very long, the probability of packet delivery is also low. If the path from source to destination is short, the probability of packet delivery is high.

For example, assume the probability of an intermediate node dropping a packet or packet getting lost is 0.3. So the probability of an intermediate node forwarding the packet is (1 – 0.3) = 0.7. The length of the path from source to destination is say 'n' hops. Thus, the probability of packet reaching destination = n * 0.7 assuming that all the intermediate nodes has to forward packet towards the destination. As the value of 'n' increases, the probability of packet reaching the destination decreases. So path with less hops reduce the energy needed to reach the destination and also increases the probability of packet reaching the destination.

1.3.MOTIVATION

1.3.1 Need to Identify Selfish Nodes Any node in the network can drop packet/misbehave at any point of time. Although the reasons to drop packet are many, a node should not drop packet to disrupt the network or to save its battery power. If nodes in the network drop packets, it will affect the whole network because each node is dependent on other nodes in the network for communication. These networks are used in emergency situations like battle field or disaster recovery where packet delivery ratio should be high. Some examples of selfish behavior are:

a. An intermediate node dropping a packet instead of forwarding it.
b. An intermediate node can change or drop the data part to conserve its energy used.
c. An intermediate node can drop the packet and inform the source that there is no route to the destination.

The routing attacks can be handled by the routing algorithms but they cannot differentiate if the packets are dropped by an intermediate node or due to the routing issue. We have to make sure that such behavior of the nodes is not encouraged.

1.3.2 Preventing Selfish Behavior Selfish behavior should be prevented to increase the efficiency of the network. Such behavior can be prevented using the following ways:

a. Finding new routing algorithms.

b. Identifying selfish nodes in the network.

In the first method, new algorithms are found to route packets to the destination. The routing algorithms are created in such a way that packet can be routed to the destination even in the presence of selfish nodes in the network. The main aim of these algorithms is to find a route with maximum number of trusted nodes so that probability of packet reaching the destination increases. Game theory approach has been used widely in this approach. One such algorithm is presented in [15], where author uses a safe and secure method to forward packets but does not identify the selfish nodes in the network. Even with the most secure routing algorithm, there could be ways in which an intermediate node can drop packets. Also some properties like scalability are lost to ensure that the problem of selfish nodes is addressed. To make the network and routing more secure, it is advisable to identify nodes dropping packets in the network and take action against them. Some of the algorithms address the selfish nodes problem using this approach.

Some of the methods identify selfish nodes by sending Acknowledgement back to the source; an improvement of which is described in 2-ACK [2, 6], using reputation [1, 4, 10, and 11] and Credit/Currency of nodes [3, 5, 9, 12, and 13]. The latest research in this field involves reputation or use of currency to find selfish nodes in the network. Once identified, action can be taken on these selfish nodes. These selfish nodes could be removed from the network or denied use of resources in the network for a certain period of time.

1.3.3 Trusted and Untrusted Nodes in a Network To increase the security of the network, a network should have certain trusted nodes. These nodes could either be dropped by the network administrator or by the owner of the network. Consider a metropolitan area with very few mobile towers. These towers can act as the trusted nodes in the network and normal users are the users/untrusted nodes in the network. These nodes can be called Broker nodes or Credit Clearance Service as described in [4]. These

nodes help creating a hierarchical view of the network. These nodes can also be used to store currency or reputation of various other nodes of the network. This helps improve the security of the system as the currency cannot be changed by other nodes in the network. Creating a tree structure of the network also improves connectivity in the network. This tree structure makes sure that nodes on the top of a tree can be trusted and nodes near the leaf cannot be trusted. These trusted nodes can help find selfish nodes in the network based on the currency of the nodes and also take action on the selfish nodes in the network.

2. RELATED WORK

There was a lot of research done to identify selfish nodes in a mobile network. In this section, we review some significant works done in the field on detecting selfish nodes in mobile P2P networks. The most popular work done in detecting selfish nodes can be categorized into the following types:

1. Using Promiscuous mode of nodes
2. Using Acknowledgement
3. Using Reputation of nodes
4. Using Credit/Currency of nodes

2.1 USING PROMISCUOUS MODE OF NODES

Promiscuous mode is a mode in which node forwards all the traffic it can listen to the application layer rather than just to the frames addressed to it. In this mode, node listens to all the communication it can listen to in the network. This mode can be used to detect nodes which are dropping packets in the network. Assume that the route from source to destination is as shown in Figure 2.1 below.

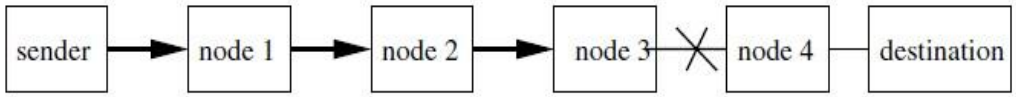

Figure 2.1 Path taken by packet from source to destination

Source of the packet is the sender and the path to the destination is node 1, node 2, node 3, node 4, and destination. Assume the selfish node in the network is node 3. Node 1 forwards the packet to node 2. Node 2 then forwards the packet to node 3 during which node 1 also listens to the packet if it is in promiscuous mode. So node 1 is sure that node

2 has forwarded the packet if it is able to hear the packet being forwarded by node 2. When node 2 is not able to listen to node 3 forwarding the packet, it assumes that the packet is dropped by node 3. If node 3 is found to drop many packets, it is found to be selfish and action is taken against it.

Some methods involving promiscuous mode are described in [7] and [8]. In [7], author has given a method to find probability of a node dropping packets. This probability helps find the possible selfish nodes in the network and prevent them from dropping packets by monitoring them. In [8], author has proposed a method to find critical nodes in the network. Using this method, few important nodes in the network are identified. We then make sure that those critical nodes forward the packets properly. This can be done by changing the mode of nearby nodes to promiscuous mode. This helps make sure that the critical nodes do not drop packets and the network works properly.

Although selfish nodes can be detected using this method, there are various disadvantages of this method. This method works only in the presence of Omni-directional antennas. Node 3 can forward the packet but it may not reach node 4 when they are far apart; this is not detected by node 2. Promiscuous mode uses lot of energy at each node as it has to process unnecessary data and is not advised for mobile networks which run on battery power.

2.2 USING ACKNOWLEDGEMENT

An idea from TCP protocol is to send an acknowledgement (ACK) back to the sender. It is normally sent by the destination back to the source and it proves that destination has received the packet. The acknowledgement only helps find if the packet reached the destination or the packet is dropped by an intermediate node. However, if an ACK is not received, either the packet or the ACK might have been dropped. Also with the wireless network, it is better to save bandwidth rather than sending an ACK back all the way to the source. This method is not secure and does not prevent other intermediate nodes from generating ACK. Also, sending an ACK back to the source increases the delay as source has to wait for an ACK. It would also impact the efficiency of the network as the available bandwidth decreases and the number of data packets sent in the network decreases considerably.

To make the acknowledgement scheme more secure, changes are made to the algorithm and are described in [2] and [6]. In [6], the author used the idea of 2-ACK to make sure that every node forwards the packet. Assuming the network to be as shown in Figure 2.1, node 2 sends an ACK back to source proving that node 1 has forwarded the packet and node 3 sends an ACK back to node 1 proving that node 2 has forwarded the packet. When node 3 drops the packet, node 2 does not receive any ACK from node 4. This helps prove that node 3 has dropped the packet. To prevent energy wastage due to the large amount of ACK messages, node 2 sends an ACK to source for every 5th data packet received. This helps reduce the number of ACK's generated by the intermediate nodes. Node 1 can create an ACK and send it to source as if node 2 has sent the ACK. To prevent such a behavior, each ACK packet is appended with the digital signature and the source of the ACK can be verified.

An enhancement of this scheme is described in [2] where author reduces the number of ACK's by dividing the path from source to destination into sets and groups. This method also uses digital signature to verify the source of the packet.

These improvements increase the security of the network but still have a few disadvantages. A selfish node (node 3) could drop the packet without sending an ACK back to node 1. This results in node 2 being detected as selfish. Also with only one ACK sent for every 5th packet received, it is hard to detect selective behavior, i.e. a node dropping 2-3 packets out of every 5 packets is very safe.

2.3 USING REPUTATION OF NODES

Most of the recent research to increase co-operation among nodes used reputation/trust values. In this method each node in the network is assigned a reputation/trust value. The reputation of a node is increased if it forwards a packet to the next hop and decreased if the node drops a packet. There is a threshold value for the network and a node is selfish if the reputation of that node is below the threshold value. The reputation of all the nodes in the network could be distributed or maintained at a single location.

Each node in the network can have a reputation for all the other nodes in the network. This reputation depends on the behavior of one node towards another. These values are

used to find the actual reputation of the node. A node requests reputation from other nodes when it needs to calculate reputation. This increases the number of messages to find the reputation of a node. To save the energy and messages sent to find reputation, reputation of all the nodes in the network can be stored at a single place. A node which wants to find the reputation sends a request to that node and it replies with the reputation value. This reduces the messages required to find the reputation value but increases the number of messages to maintain reputation at a single place.

To decrease the reputation of a node dropping packets, the nodes dropping packets has to be identified in the route. There are various methods to find such intermediate nodes. Some methods to find selfish nodes using reputation are described in [1, 4, 10, and 11]. In [1], an ACK is sent from destination to source. This ACK proves that all the intermediate nodes have forwarded the packet and reputation is increased. If there is a retransmission from the source instead of an ACK from the destination, then the reputation is decreased. Methods [4], [10] and [11] focus on new innovative formulas to find trust value in the presence of distributed reputation and selfish nodes.

Reputation is highly useful for file sharing networks where nodes with high reputation can be trusted. Reputation is a good method to find selfish nodes but it has a few disadvantages. When a node has to request a service from other node in the network, the node with maximum reputation is chosen. All the nodes in the network choose a node with highest reputation to request service or forward the packet and this increase the load on that node. Once the node receives more requests/packets than it can forward, it will drop the packets and the reputation of that node decreases again. Also a node with low reputation is not given a chance to forward packets. So the node with low reputation always has low reputation. Also there is no way in which traffic is distributed equally, i.e. one node could be sending 100 packets while other node could be sending 10 packets and both the nodes use the whole energy as they are connected to the network. Bad mouthing attack cannot be addressed using reputation scheme where a node decreases the reputation of other node to disrupt the network.

2.4 USING CREDIT/CURRENCY OF NODES

To address the disadvantages of the reputation scheme, recent research focused on assigning Currency/Credit to all the nodes in the network. Each node in the network is assigned an initial currency. This currency is used to send packet to other nodes in the network. Currency is increased for all the intermediate nodes as they help the source forward packets to the destination. So the currency of a node increases if the node forwards a packet towards the destination and this currency is used to send packets to other nodes in the network. There are many advantages of using this method compared to reputation. In this method, all the nodes are given equal importance for forwarding packets and every node is given equal opportunity to use the system. Also the currency will always stay distributed in the system as it is a peer-to-peer network. A node will run out of currency if it only sends the packet without forwarding other packets. Also currency of a node is decreased if it is found to be dropping packets. A node which runs out of currency could be either a node dropping packets or a node which is using too many resources of the system. Various methods involving currency/credits are [3, 5, 9, 12, and 13].

An example of currency based method is Sprite [3]. In this method, a credit clearance service is used to store currency at all the nodes in the network. This helps reduce attacks possible in distributed reputation like bad mouthing attack. Each intermediate node which forwards a packet sends a certificate along with token to the credit clearance service. All these nodes are then given currency by the credit clearance service. It provides currency to all the nodes forwarding packets and decrements the total amount of currency from the source. Once the currency of a node is below threshold, that node is denied resources of the network for a certain period of time as it might be dropping packets. Another important method using virtual currency was proposed in [5] that helps improve co-operation among nodes in the network. Methods [9, 12, and 13] also focus on similar approaches to find selfish nodes. Some of these methods suggest charging the destination while some charge the source for forwarding the packets.

There are various algorithms and various approaches as described above to detect selfish nodes in the network. But the main advantages of any selfish node detection algorithm should be the following.

a. The algorithm should work with any existing routing protocol.
b. The algorithm should be scalable and efficient with less overhead.
c. The algorithm should not use up more battery from the nodes as nodes in P2P network require battery power to function.

PAPER

I. An Incentive Based Approach to Find Selfish Nodes in Mobile P2P Network

Hemanth Meka and Sanjay K. Madria

Department of Computer Science
Missouri University of Science and Technology, MO 65401
Email :{ hmff8@mail.mst.edu, madrias@mst.edu}

ABSTRACT In a mobile P2P network, it is assumed that nodes not only send their own packets, but they also route packets sent by other nodes. However the presence of -selfish nodes which drop packets can affect the efficiency of the whole network. We propose a mechanism using virtual currency to find selfish nodes in the network and thus improve co-operation among nodes. We do this by issuing a receipt message for the data forwarded. Later each intermediate node uses the receipt to prove that it has forwarded the packet, and based on the receipts received selfish nodes are identified and the virtual currency for providing service is distributed accordingly. When compared to the previous approaches, our scheme does not require the presence of any tamper proof hardware at each node, and is efficient in terms of time to detect and the number of packets exchanged and can be integrated with any routing algorithm. Once identified, selfish nodes are punished to make the network more efficient. We also propose measures to prevent some other kinds of malicious activity like spoofing, eavesdropping and replay attacks to make the network more secure.

1.INTRODUCTION

A Mobile P2P (M-P2P) network is a set of nodes which move independently within an area and co-operate among themselves towards accomplishing a mission. For example, M-P2P networks can be used in many military and rescue operations where there is no existing infrastructure/communication media or where there is a necessity for a network to exist but without infrastructure. It can also be used with restricted

infrastructure such as in vehicular networks, where travelers have to communicate with other moving cars and other mobile towers present.

The nodes in Mobile P2P network communicate only using wireless links whose strength depend mainly on the distance between nodes. Thus, these links can be created or broken at any time as the nodes keep moving independently. As they move, their neighbors keep changing as well. So it is important that all the nodes help each other by providing services such as forwarding messages and moving data packets to the destinations. However, mobile nodes are constrained on battery power and limited bandwidth. Some of the intermediate nodes could be selfish and could save power by dropping packets instead of forwarding them and use bandwidth for forwarding their own data. They could also be part of a malicious activity where nodes can try to spoof packets, or eavesdrop between two communicating nodes and also try to modify data. Such malicious behaviors have also to be prevented to improve the network efficiency.

Other constraints associated with mobile networks like mobility and unreliable connectivity also make it difficult to detect such activities. So methods are needed to identify selfish nodes and make the mobile P2P network much more efficient in terms of energy savings and reducing latency. There are various methods proposed for wireless networks [1, 3, 9, 11, 13, 14] to help improve co-operation among nodes for better network efficiency. Some of them also provide incentives for helping other users/nodes which is similar in concept to our approach while some uses reputation to find selfish nodes. However, [3, 9, 13] assume that the path to the destination is fixed which cannot be guaranteed in M-P2P network. Also most of them do not consider prevention of malicious activity like spoofing, replay attacks etc. They also do not find selfish nodes in the network but do provide currency to all the nodes which send receipt as a move to entice cooperative behavior. Method described in [1] uses reputation to find selfish nodes in the network and can work with any routing protocol. It also prevents a variety of malicious behavior by other intermediate nodes like replay attacks etc. Each node stores the reputation of its neighboring nodes and uses this value to find selfish nodes in the network. This method is a low cost and efficient method to find selfish nodes considering the attacks possible and the restrictions imposed by the author. The method described in [6] sends a 2-ACK to the second previous hop. This acts as a proof that the node between

the two nodes has forwarded the message. A 2-ACK is sent for every R_{ack} packets sent to decrease the overhead involved. This helps find selfish nodes in the network by measuring the number of ACK's missing and that value has to be more than the R_{ack} as R_{ack} number of ACK's are not sent to prevent overhead. We compare our algorithm with [1] and [6] because [1] has the most similar assumptions and [6] is the most recent promising algorithm in this field.

We propose a scheme to identify selfish nodes for a Mobile P2P network in the presence of minimum infrastructure. The mobile network structure assumed is as follows. We divide the mobile p2p network into three kinds of nodes (Broker nodes, Access Point nodes and Non-broker nodes). Broker nodes and access Point nodes are trusted as they belong to the infrastructure while non-broker nodes cannot be trusted. They could be selfish or act maliciously at any point of time. Each node is associated with a pair of private key and public key pair while all the Broker nodes and access point nodes have the same (public, private) key.

We propose a protocol to detect selfish nodes and also prevent malicious activity in a network with the presence of minimum infrastructure. For this purpose, we propose a method to create receipts at the sender node which is appended to the data to be sent to the destination. Some of the previous methods involving receipts [4, 11, 12] send a request to the Broker node for a certificate which acts as a receipt for the packet. In our method, we reduce the latency in sending data by creating a receipt at the source and piggybacking it with the data packet. These packets are sent to the destination using any given routing algorithm. All the intermediate nodes store the receipts and then send the receipts to trusted Broker nodes/access point nodes. The Broker nodes then create a virtual path for each packet and distribute currency among intermediate nodes. The Broker node also creates a complaint if a packet is dropped. The selfish nodes are then punished based on the amount of currency and complaints, thus improving the efficiency of the network and also making it more secure. This scheme also prevents other malicious activities but does not identify the nodes responsible. The main advantage of our scheme is that a source can create its own receipt and then send data which decreases the latency to disseminate data. It is also able to find the selfish nodes in the network which considerably improves the co-operation among nodes in the network.

We reduce the latency in sending data packets which improve the performance of the network. In addition, finding selfish nodes will help use routing through trustworthy nodes so the overall performance of the whole network is much better than [1].

The rest of the paper is organized as follows. We provide a brief overview of the previous research in Section 2, describe the system configuration in Section 3, propose our method in Section 4 and then provide the working details of the algorithm. We propose the simulation and performance evaluation in Section 6. The conclusion and future work is given in Section 7.

2.RELATED WORK

Several models [3, 6, 7, 9, 11, 13] have been proposed to route packets in a network where all the nodes cannot be trusted. Some of them use acknowledgements similar to TCP protocol, while others make sure that all nodes can work properly and force them to participate in the network in various ways.

An idea from TCP protocol is to send an acknowledgement (ACK) back to the sender. It is to make sure the destination has received the packet. However, the network being a wireless network it is better to save bandwidth rather than sending an ACK back all the way. It would also impact the efficiency of the network as the number of data packets sent decreases considerably. In [6], the author used the idea of 2-ACK to make sure that every node forwards the packet. If node ‘A’ forwards the packet to node ‘B’, and ‘B’ forwards it to node ‘C’, then node ‘C’ sends an ACK to the node ‘A’. The details and drawbacks of the algorithm are explained later.

The idea in [7, 8] was to find nodes in the network which can drop packets and call them critical nodes. The protocol will then make sure that these nodes don’t drop packets. In a mobile network, it is difficult to ensure that critical nodes remain the same; other nodes which are non-critical can also start dropping packets at any point of time. In [7, 8], such ideas of finding selfish nodes and critical nodes have been discussed but the method fails in most cases. One could also make sure that each node is willing to participate in the routing which is discussed in [9] called Participation Willingness (PW). This will make sure that all nodes in the path are willing to participate and this helps in

building a safe path. However, we cannot force nodes; they can start dropping packets later which will make the computed values and categorization useless.

One of the latest ideas [10, 11, 12] involved finding/maintaining trust values for nodes in the network. If the objective is file sharing, a trust value proposed in [10] is very useful. This can be used by finding the trust value of the source of the file and comparing it with a threshold. These trust values keep increasing if there is a successful transfer of a file with that node and decreased if a transfer fails. Some other ideas like tit-for-tat proposed in Prisoners game [12] where faulty nodes co-operate with each other, trusty nodes co-operate with each other while a faulty and a trusted node do not co-operate. These reputation values can be stored at each node or at a central node which can be trusted. To maintain the reputation of all the nodes, we need to have proper hardware at each node so that selfish/ malicious nodes cannot manipulate the reputation of any node. This is needed because each node has the reputation of other nodes. We could store the reputation of all the nodes at a trusted node to prevent manipulation but this result in many requests and replies to find trust values. Also, reputation is not effective against Bad mouthing attack. A node can keep sending information to other nodes to improve its reputation and then start dropping packets. Also, reputation doesn't care about the percentage of the network being used i.e. a node has a high reputation receives many requests and a node with less reputation does not receive any requests and is idle.

Most of the present research in the area of Virtual Currency is to promote co-operation among nodes. Virtual currency has also been proposed to remove in-efficiencies in the reputation system. Some of the methods involving virtual currency are described in [3, 13, 14, 15, 16]. In [13, 14] the path to the destination is found before any data is sent. It is similar to a TCP connection and is not a good option for mobile networks. It should be avoided. In [3], author has proposed a very good review of the amount to be paid to each node during various conditions in the network. It is similar to our approach but does not find the nodes which drop packets. It pays currency to all the nodes which send a receipt to the Broker node but does not propose any method to find selfish nodes. The main difference comparing with our algorithm is that we make sure that a node has actually forwarded the packet before paying the currency.

A latest method to find selfish nodes [2] is an improvement to an already existing algorithm [6] and involves advanced acknowledgement. In this method author has divided the path from source to destination into groups and proposed a method which gives us the ACK's to be sent. Although this is the latest paper in the area, it has many restrictions. It assumes that there are no colluding nodes, selfish nodes do not drop ACK's and nodes stay in promiscuous mode.

A model to find selfish nodes which has similar assumptions compared to our algorithm is given in [1]. It works on any routing protocol and the network is also not restricted in terms of the type of attacks possible. It is cost efficient as the reputation is not shared among various nodes. It also prevents attacks involving colluding nodes and nodes do not enter promiscuous mode. In this method, each node stores the reputation of all its neighboring nodes. Source sends data to the destination and destination sends an ACK back to the source. All the intermediate nodes increase the reputation of the next hop if an ACK is received and decrease the reputation of the next hop if there is a retransmission by the source. Based on the amount of reputation of the neighboring node, a node decides whether to forward the packet to that node. Based on the various assumptions, restrictions and malicious activities prevented, this is one of the best way to find selfish nodes.

Another important model is using 2-ACK which is described in [6]. In this paper, each hop sends an ACK to the second previous hop. If the route is (A, B, C, D, and E), then C sends a 2-ACK to A, D sends a 2-ACK to B and E sends a 2-ACK to C. To prevent the overhead only one ACK is sent for every R_{ack} packets sent. Based on the number of 2-ACK packets missing, the selfish nodes are identified. However the network assumes all the nodes can be trusted. Each node broadcasts information about the link dropping packets. This method does not prevent malicious node from proving trusted nodes as malicious. This also does not guarantee that every node is actually forwarding the packets as the middle node can be selfish/ malicious and try to send ACK for itself. To prevent such behavior, digital signature has been proposed. Also the data is encrypted and decrypted at each node which increases the latency of the system. We also find false positives if the nodes are moving in the network.

We propose a method in which a node can send a message to any node using its virtual currency. We assume the network to have very less restrictions. Nodes do not enter promiscuous mode in our algorithm. If a node keeps dropping packets, it is detected by a trusted node called Broker node which punishes the nodes dropping the packet. Only Broker nodes store the currency of nodes to prevent malicious behavior by non-broker nodes. We use receipts to detect nodes co-operation and Broker nodes detect the selfish nodes in the network based on these receipts. Also currency is given to nodes which forwarded data. We also reward the nodes which help detect selfish nodes by providing additional currency. The selfish nodes are then punished as needed and they could also be banned from the network based on their behavior.

3.SYSTEM CONFIGURATION

The network is composed of three kinds of nodes. There are Broker nodes (BN), Access Point nodes (APN) and many Non-broker nodes (NBN). The Broker nodes act as the central authority for the network. They maintain the virtual currency of all the non-broker nodes in the network. All access point nodes are at a 1-hop distance from Broker nodes and are used as a bridge between non-broker nodes and the Broker nodes. All non-broker nodes connect to the Broker node using access point nodes. If no access point node is reachable, they connect to non-broker nodes which in turn connect to a Broker node. The type of each node is decided when the network is created. The type of node does not change later. A sample network is shown in Figure 3.1 below.

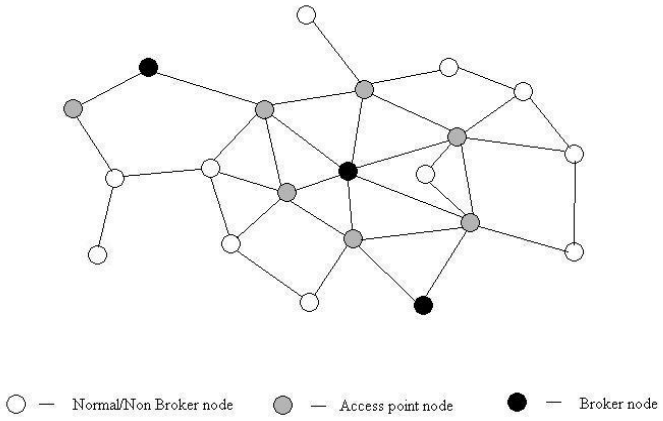

Figure 3.1 Overview of the network

In the Figure 3.1, dark nodes represent the Broker nodes, the grey nodes represent access point nodes and white nodes represent normal nodes. Broker nodes can be connected to each other in one of the three ways 1) Direct link, where Broker nodes are directly connected with each other 2) 1-hop link, where an APN nodes connects two Broker nodes and 3) 2-hop link, where there are two APN's between Broker nodes.

ASSUMPTIONS

The Broker node and Access point nodes are considered to be trustworthy and assumed to never drop packets based on selfish behavior. Each non-broker node has a public and private key associated with it. All the Broker nodes and access point nodes share the same public and private key pairs. We assume that public key of a node is known to all other nodes in the network and private key of a node is known only to that node. We assume that the MAC address and private key of a node cannot be guessed by another node. A node has to use its own MAC address. We do not assume attacks due to routing algorithm. We also assume all the non-broker nodes know the ID/address of Broker nodes in the network.

4.COMPLAINT BASED COMMUNICATION PROTOCOL

This section discussed the protocol used for data communication and elaborates on how currency is used by the system. We use a currency based approach; nodes earn currency for their services in the network. Each non-broker node has a *virtual currency (currency)* associated with it. The currency is stored at Broker nodes to prevent forgery by other nodes. When a source node wants to send data to another node (destination), all the nodes in the route collaborate by forwarding the data to the destination. Source node thanks the intermediate nodes by paying currency to them as they help the source node forwarding the packet. Therefore, these intermediate nodes gain currency by helping the source forward packets to the destination. We encourage every node to forward more packets and earn virtual currency as a node that has more currency can send more packets. Having more currency than a threshold also proves that the node is not selfish. The currency of a node is increased or decreased based on the receipts sent by each node. Each packet has an *Authentication Token (token)* associated with it, which acts like the digital signature of the source node. It is also used to prove that a node having the token has helped forward the packet to the destination. Based on the receipts received for a packet, we create complaints on nodes when packet drops have been detected. Complaints are then analyzed to find the actual nodes which are dropping packets in the network. Next, the currency of the node dropping packets is deducted and the currency of the nodes which have helped finding that node is increased. As complaints are converted into currency, the selfish nodes in the network can be detected based on currency earned. A node which has currency less than the threshold value is declared as selfish.

4.1 SENDING AND RECEIVING DATA

To send data to the destination, we append some more information to guarantee authenticity and integrity of the data payload. The information that is appended to the data is an MD5 of the data and token. MD5 is used for message integrity while a token is used for message authenticity and for tracking of messages. Both MD5 of data and token prove that the message is sent by the source and not altered by any intermediate node. The details of the packet and the creation of the packet are explained as follows.

When a source *S* wants to send data to destination D, it creates the packet consists of the actual data payload, MD5 of the data for data integrity and an encrypted token. MD5of data is a piece of information used to authenticate the data at any point. Private Key of the source is used to generate MD5 of the data. Public key of the source is used by the destination to prove that the message is not altered by any intermediate node. The data and MD5 of the data are encrypted with the public key of the destination. The encryption is necessary to prevent intermediate nodes from changing the data; MD5 also helps prevent any changes to the data and improves security. A token contains the source node, destination node, Broker node to which this token has been sent, sequence number of the source, and MAC address of the sender. The structure of the token is shown in Figure 4.1.

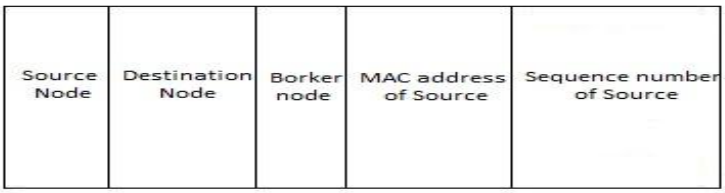

Figure 4.1 Token structure generated at the source

Source node and sequence number of source node are used by the Broker node. MAC address of the sender is used to prove that the packet is sent by the actual source and not created by any other node. We assume that the MAC address of one node cannot be guessed by another node. Destination node is added so that the destination is not changed by any intermediate node. Since this token is used only by the Broker nodes, it is encrypted with the public key of the Broker nodes and added to the message. The structure of the packet sent by the source is shown in Figure 4.2. This message is sent to the destination with the help of other nodes in the network. The message is routed using any hop by hop routing or source routing algorithm (e.g. DSR, DSDV etc.). Our

algorithm is independent of the routing algorithm used. The algorithm to send a message at source is given in Figure 4.3.

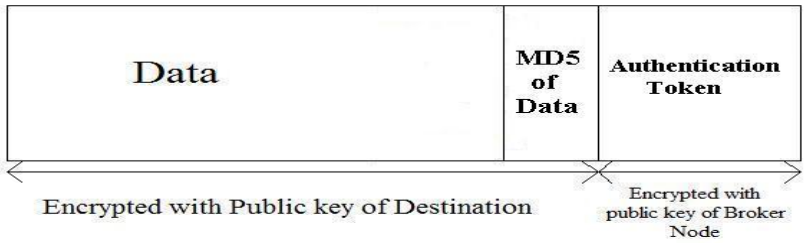

Figure 4.2 Packet structure transmitted by the source.

/* Generate a message at source and add it into the message table. The messages are then transmitted from the message table. We only describe creating a message and adding it to the message table. */

Generate Message {

INPUT: Destination D, Data to be sent, Private key of Source K_S, Public key of destination K_D and Broker node P_B.

OUTPUT: Packet can be transmitted to the destination.

INITIAL PARAMETERS: Message= NULL, Token= NULL, Data = Data to be sent.

Hash = MD5(Data, K_S); *//H- Hash function*

Message = Data + Hash;

Encrypt(Message, K_D);

Token = Source + Destination + Broker node + Sequence no. + MAC address of Source;

Encrypt(Token, P_B);

Packet = Message + Token;

}

Figure 4.3 Algorithm to send a message

When the destination node receives the packet, it decrypts the message using its private key and checks the authenticity of the message using the MD5 hash data attached. If the data passes the test, the data is correct and the destination can use the data.

Each intermediate node and the destination store the token along with additional information when the packet is forwarded or when the packet reached the destination. This is explained in Section 4.2.

4.2 STORING AND SENDING TOKENS

The source provides currency to all co-operating intermediate nodes. But the entire currencies are stored at the Broker nodes so a Broker node should know which nodes have forwarded the packet. In this regard, each node sends tokens to the Broker node confirming that it has helped forwarding the packet. Broker node can verify that node 'A' has forwarded the data after it receives receipts from 'A'. To prove about its service/behavior to the Broker node, every intermediate node stores the token sent with the message along with additional information before forwarding the data to the next hop. This token is now called receipt and is sent to the Broker node later to prove that it has helped in forwarding the message to the destination.

```
FORWARD PACKET(packet) {
/*Store required information from the packet and then forward the packet to the next intermediate node based on the routing algorithm used.*/
INPUT: Packet to be sent
OUTPUT: Receipt stored and Packet forwarded to next hop.
Receipt = Token from packet + Node from which packet is received + Destination of packet;
Find next node based on the routing algorithm;
Receipt = Receipt + Next hop node + Present Timestamp;
Enqueue (Receipt, Receipt_repository) //Store the receipt along with other receipts to be sent.
}
```

Figure 4.4 Algorithm for storing receipts

Each intermediate node and the destination store the token for the packet that has been forwarded or received. They also store some additional information which is used by the Broker nodes. The additional information saved in the receipts along with token is "*Node from which packet has been received, Node to which packet is forwarded, Destination of the packet and Timestamp at which packet is processed*". The token along with the additional information is now called receipt. The algorithm to store the receipts at the intermediate node is as given in Figure 4.4.

All the intermediate nodes save the receipts so that they can be sent directly to trusted nodes. Whenever a Non-broker node (NBN) is within the range of an Access Point node, it will send all the stored receipts to the Access Point node. The receipts are sent as data packets through other Non-broker nodes if a node cannot reach Access Point nodes directly for a long time. This is necessary so that Broker node does not wait for a long time to give out virtual currency credits. Also the node pays currency to all the intermediate nodes for forwarding the packet.

The receipt format is shown in Figure 4.5. These receipts cannot be created or forged by any node as they contain the MAC address and sequence number of the source and are encrypted with the public key of the Broker node.

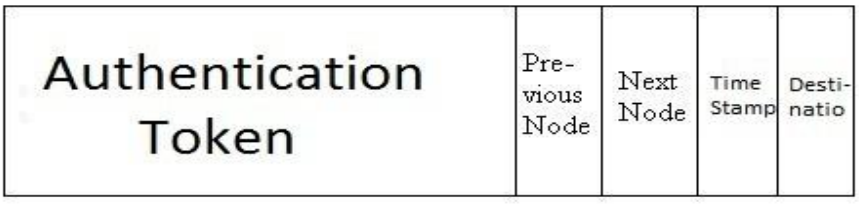

Figure 4.5 Receipt structure stored at each intermediate node and destination

4.3 CREATING VIRTUAL PATH AND ANALYSIS

Virtual path is a path created at the Broker node to track the path taken by the packet from Source to Destination. This path is created from the receipts sent by various intermediate nodes.

Upon receiving the receipts, an access point node first decrypts the receipt and sends it to the Broker node based on the Broker node present in the receipts. This is done so that all the receipts of a given packet reach the same Broker node. The Broker nodes then separate the receipts received based on the source node and the sequence number. The receipts with the same source node and the sequence number are arranged in order from the source to the destination based on the previous node and next node received along with the receipt. The total route followed by each packet is thus known to the Broker node based on the receipts. The path thus created at the Broker node is called a *virtual path*. This virtual path is then confirmed based on the timestamps in the receipts. The various types of virtual paths possible are shown in Figure 4.6.

The virtual path is analyzed when the path is completed or when the timer associated with the path expires. The timer depends on the various network characteristics like connectivity. Based on the type of virtual path, it can be found if the packet has been forwarded correctly by the intermediate nodes or if the packet is dropped by any intermediate node. Based on the virtual path, an action is taken and currency is credited.

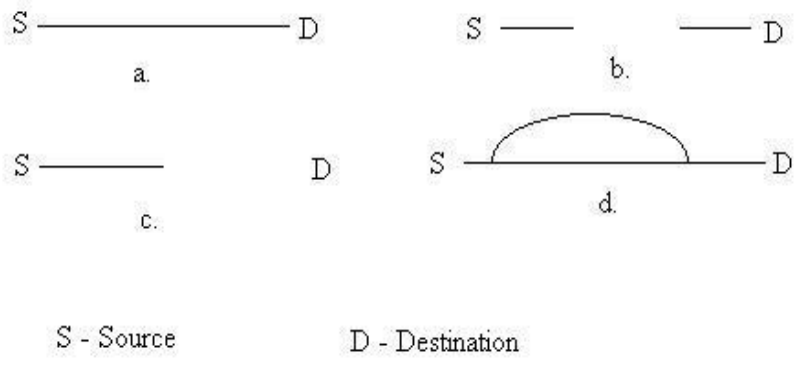

Figure 4.6 Different types of virtual paths possible

All deserving nodes are given currency based on the power used by the nodes and the total currency is deducted from the source. The amount of currency is derived based on the formula below.

P_t – Power to transmit data

P_r – Power to receive data

Since an intermediate node both receives and sends data, the power used is:

$$\alpha = P_t + P_r \tag{1}$$

According to Frii's equation, the ratio of P_r to P_t is given by:

$$P_r = \frac{P_t}{L_p} \tag{2}$$

where L_p is the Path loss. The equation of path loss is:

$$L_p = \left(\frac{4\pi R}{\lambda}\right)^2 \tag{3}$$

where R represents the maximum distance a transmission can be sent, $\lambda = V_w/f \approx V_w/B$ assuming bandwidth represents the range of frequency and V_w represents the velocity of wave propagation.

So the total power used at the intermediate node is:

$$\alpha = P_t\left(1 + \frac{{V_w}^2}{(4\pi RB)^2}\right) \tag{4}$$

So the currency that can be given to a node for co-operation is given by

$$\beta = factor * P_t\left(1 + \frac{{V_w}^2}{(4\pi RB)^2}\right) \tag{5}$$

Since the energy used is found in terms of Nano Joules, we need a factor so that currency is not too low to be managed. So we use a "factor "which is decided based on the network parameters so that the currency credits in the system can be used and can be related to power used by a node.

A maximum amount of currency β is given to each intermediate node and a maximum of β/2 is given to the destination. This is because destination only receives the packet but does not transmit the data. So the energy used by the destination is almost half of the energy used by intermediate nodes. *The total amount is then deducted from the source node as the cost for sending data to the destination.*

The virtual path is as shown in Figure 4.6(a) if every node worked perfectly. So the currency is paid as discussed above to all the intermediate nodes and the destination. The total currency is then deducted from the source. This currency credits changes are done at the Broker node at which all receipts are received.

The virtual path is as shown in Figure 4.6(b) when the destination has received the message but the intermediate nodes were not able to send receipts to the Broker node. This could happen if some intermediate nodes are not able to reach an access point node. However as the destination has received the message, currency is paid to the intermediate nodes who have sent their receipts and is deducted from the source.

The receipts are as shown in Figure 4.6(c) if the destination has not received the message. It could also be possible that destination has received the message but a part of the network is not able to reach an access point node. The Broker node starts a timer and when the timer expires, it creates a complaint on the node which might have dropped the packet. It is hard to find the exact intermediate node which has dropped the packet. If the last receipt received is from node 'A', it could be either dropped by A or by the next hop of 'A' which hasn't sent a receipt. So a complaint is created as a pair on (A, next hop of A). The next hop of A is found from the receipt sent by A. Since the destination has not received the message, no virtual currency is deducted from the source, but currency is paid to the remaining intermediate nodes except the nodes on which complaint is created.

The receipts are as shown in Figure 4.6(d) if the receipt has been sent to other colluding nodes and they are using the same receipt to gain currency. This is similar to

replay attack where nodes try to use the same receipt to gain currency. In such a situation it is hard to find the actual route taken since nodes co-operate among themselves to destroy the network. In this case, we give currency to only nodes which are actually trustworthy. So currency is given to all the intermediate nodes where there is no duplicate route. The total currency is also deducted from the source since the packet has reached the destination.

The currency allocation algorithm and the selfish node detection algorithm are given in Figure 4.7.

4.4 IDENTIFY NODES DROPPING PACKETS

Based on the various complaints created due to virtual path of the type 7.c, nodes dropping packets are detected.

All the complaints are stored at a single Broker node so that they can be processed easily. The Broker node at which complaints are stored is decided by all the Broker nodes. Whenever a Broker node has to create a complaint, the parameters of the complaint are sent to the decided Broker node. A complaint is created upon receiving the parameters by that Broker node.

A node having many complaints is likely to be the one dropping packets. Currency is debited for the node dropping packets and is credited for the other nodes that made complaints. This is done so that selfish nodes can be detected based on the amount of currency present.

We compute a threshold value for the complaints. This value is calculated every time the complaints are evaluated. The average number of complaints on a node is found from the number of complaints and number of nodes involved in the complaints. We also compute the standard deviation of the complaints.

$$Threshold = Average + n * Standard\ Deviation + 1.$$

The value of ‘n’ is either ‘1’ or ‘2’ and it depends on the network characteristics like load on the network, bandwidth etc. If the load on the network is too high or if the connectivity is low, ‘n’ is set to ‘2’ and it is set to ‘1’ if connectivity is good and load is normal. A node having complaints more than the threshold value is considered to be dropping packets.

The nodes dropping packets are then punished. To punish node 'A', currency is deducted for node 'A' and other nodes in the complaints (-, A) or (A, -) are awarded as they have helped identifying the selfish nodes in the network. The amount of currency deducted from node 'A' depends on the number of complaints 'm'. That currency is divided among other nodes in the complaints as a reward.

The amount of currency deducted at the selfish node is:

$$\beta_1 = m * factor * P_t\left(1 + \frac{{V_w}^2}{(4\pi RB)^2}\right) \quad (6)$$

Each of the nodes which have proved that node 'A' is malicious is given an amount found using the formula:

$$\beta_2 = factor * P_t\left(1 + \frac{{V_w}^2}{(4\pi RB)^2}\right) \quad (7)$$

The details of the formula are explained in the explanation above in Section 4.3.

```
/* Based on all the receipts received, the path followed by a packet is checked and
currency credits provided*/
CHECK_PATH_FOLLOWED(receipts) {
INPUT: Receipts from various intermediate nodes including destination
OUTPUT: Provide Virtual currency credits to intermediate nodes and detect selfish
nodes.
Arrange the receipts based on the next and previous node present in receipts.
Verify the receipts based on the timestamps
If (path == 3a)
 { for all intermediate nodes and destination
    { currency credits = currency credits + β;
      total = total + β;
    }
    currency credits[source] = currency credits – total;
```

Figure 4.7 Currency distribution and identifying nodes dropping packets

```
  }
If (path == 3b)
  {  for all intermediate  nodes and destination  known
     {   currency  credits = currency  credits + β;
         total = total + β;
     }
     currency  credits[source]  = currency  credits  – total;
  }
If (path == 3c)
   {  for all intermediate  nodes known
     {   currency  credits = currency  credits + β;
     }
     Create Complaint  based on route
  }
If (path == 3d)
   {  for all intermediate  nodes without  duplicate  route
     {   currency  credits = currency  credits + β;
         total = total + β;
     }
     currency  credits[source]  = currency  credits  – total;
  }
}
```

```
μ = (No. of complaints  *2)/No. of distinct  nodes in the complaints;
σ = St.Dev(Complaints);//Find  Standard deviation
Threshold  = μ + σ + 1;
If (complaints  on node 'A' >Threshold)
     currency  credits[A]  = currency  credits  – β1;
for all nodes 'X' where complaints  ϵ (X,A) or (A,X)
{   currency  credits[X]  = currency  credits + β2;
}
```

Figure 4.7 Currency distribution and identifying nodes dropping packets (continued)

5. MAINTAINANCE OF NETWORK

In this section, we describe various maintenance or administrative issues of the network and also describe the malicious activities that we prevent.

5.1 CHOOSING BROKER NODE

When a source node has to send a packet, it chooses a broker node and adds it in the token. This is done so that all the receipts of that packet are sent to the same broker node and virtual path can be created.

Each access point node has a Broker node associated with it. This means that the Access point node is used to increase the connectivity of that Broker node. A non-broker node chooses the nearest Access point node and chooses the Broker node associated with it. The node then uses the Broker node for all the packets that it has to send. If the non-broker node has moved a lot, it finds the Broker node again.

5.2 CREDIBILITY OF NODES

There is a possibility of a node dropping packets due to the poor communication or interferences. We try to distinguish such problems at the network layer from the selfish behavior of nodes using credibility.

Credibility of a node is a 4-bit integer and is stored at various Broker nodes. Whenever a virtual path is completed, it is analyzed. If the path is as shown in Figure 4.6(a), 4.6(b) or 4.6(d), packet has reached the destination. The credibility of nodes is not changed in this situation. If the virtual path is as shown in Figure 4.6(c), the packet has not reached the destination. The basic idea is to decrease the credibility of the intermediate nodes which might have dropped the packet. Also the timestamp at which this is done is noted. Whenever the packet does not reach the destination, the difference between the time-stamp associated with credibility and the present timestamp is found. If the difference is low, it means the packet is dropped frequently, so the 4-bit integer is right shifted, a ‘0’ is inserted to the left and present time-stamp stored along with credibility. If the difference between time-stamps is high, all the bits are reset to ‘1’ and then a ‘0’ is inserted to the left. It means the credibility is set to 0111. This is done so that the credibility is reset when the packet is dropped after a long time.

Whenever the credibility of a node reaches '0', a message is sent to (all the near-by nodes/all nodes in the network) so that the node with least credibility is not used to forward packets. Each such message is associated with a timer, so that the node can be used to forward packets after a certain period of time. This is done so that the node can be used to forward packets after it has moved from its position.

Credibility does not affect finding nodes which drop packets in the network. Credibility of a node is zero when 4 packets between (X, Y) are dropped. But a node is dropping packets only when a node involved in each complaint is different.

5.3 PREVENTING MALICIOUS ACTIVITY

The various attacks possible and the actions taken against the attack are as explained below.

Refusal to Pay: The source node cannot refuse to pay because the currency is handled by the Broker node and the message has a token which contains the MAC address of the sender. So there is no chance of the packet being created by any malicious node.

Dishonest Nodes: Nodes which do not actually forward packets but send receipts are not paid any currency because the currency is only paid when the whole route is known to the Broker node and nodes are not paid when there are duplicate routes as explained for Figure 4.6(d).

Replay attack: Each receipt has information about the source node and the sequence number. An intermediate node sending the same receipt is ignored as it is considered a duplicate. So the same receipt cannot be sent again to the Broker node.

Invasive adversary: A node cannot decrypt any data because data is already encrypted with the public key of the destination and can be decrypted only by the destination. The receipt also cannot be decrypted because it is encrypted with the public key of the Broker node and can only be decrypted by the Broker node. Also the token cannot be forged without decrypting it. So there is no possibility of changing the message contents.

Man in Middle attack: A malicious intermediate node can put in junk data by finding MD5 and encrypting it with public key of the destination. But the MD5 of data is generated using private key of source. Destination generates the MD5 of data using

public key of the source and finds that the MD5 does not match. So it is detected that the data is altered.

Selfish node sends receipt: A selfish node can drop a packet and send a receipt to the Broker node to get its currency credits. It acts as if it hasn't dropped the packet but the next hop has dropped the packet. Since the Broker node registers a complaint with both the nodes, selfish node cannot escape.

Selfish node does not send receipt: A selfish node can drop a packet and acts like it hasn't received the packet. The last node in the virtual path is the previous hop of selfish node. The Broker node then registers a complaint on the previous node and the selfish node. A complaint is still registered against the selfish node and selfish node is responsible for packet drop.

Free riders: An intermediate node can send data to any intermediate node by adding its information to the data but since the network is not fixed, route is also not fixed. The data also cannot be tunneled. Also changing a part of data is useless since it is easily found at the destination using MD5 of data.

5.4 MAINTAINING VIRTUAL CURRENCY

With the presence of many Broker nodes, we have many locations to store virtual currency of a node. Also a node 'A' can gain currency at Broker node 'Y' while it has to use it at Broker node 'Z' to which it is connected. It is necessary to manage currency of non-broker nodes among the Broker nodes. The currency of a node is used only when the node has to send a packet. When the currency of any node 'A' is low at Broker node 'Z', the Broker node 'Z' adds currency to that node 'A' and sends a message to other Broker nodes to deduct currency. Other Broker nodes which have the currency of node 'A' can deduct currency at their position. If they are out of currency too, they send the request to other broker nodes. This will spread the message over the whole network till currency can be deducted. When the request's TTL or number of hops allowed expires, an ERROR message is sent back to the broker node 'Z' which has started the message. Based on the number of ERROR messages received, it is known if a node 'A' is running out of currency credits. Based on the ERROR messages received, the currency of a node in the network can be known and various decisions taken.

5.5 DETECTING SELFISH NODES

We can now find selfish nodes based on the amount of currency of a node. Currency of a node is low when many ERROR messages are received for a currency credits request by Broker node. Currency is a node is now low if a node is just sending packets through the network and not co-operating in forwarding other packets. It could also be low if it is dropping packets. Whenever the amount of currency falls below a certain level, that node is selfish and a broadcast is sent to the whole network so that node is ignored later in the subsequent routing.

6. SIMULATION AND EXPERIMENTAL RESULTS

We built a simulation environment in Java to study the performance of the complaint based approach. We conduct experiments on the Complaint based approach described in Section 4 and compared it to the distributed Double Decrement Single Increment Reputation (DDSIR) model. The simulation area is approximately 1000 X 1000 m^2 and it can afford a range of 10 – 150 nodes in the network. The maximum connection distance between two nodes is 200m. The bandwidth between any two nodes ranges between 128 kbps and 512 kbps. Messages are randomly sent between nodes in the network at an average rate of 50 messages per minute. The movement of nodes is implemented using random way point model. Each node moves in a zigzag line from one point to other. The speed of a node is 10 m/s and the entire network moves at the same speed. The routing algorithm used is DSR. If the route found by DSR is broken while sending data packet, a new route is found by the intermediate node rather than sending a RERR message back to the source. This prevents unnecessary packet drop by intermediate nodes. The various simulation parameters are given in Table 1 and the other parameters are explained in respective sections.

The broker nodes and access point nodes are introduced so that they do not drop packets. The broker nodes are decided by us and the nodes adjacent to broker nodes are then set as access point nodes. All the nodes work as normal node for other algorithms. The whole network now acts as a mobile peer to peer network with nodes randomly sending messages to one another. The effectiveness of the protocol can be evaluated by performing experiments on the network.

Table 6.1 Simulation Parameters

Parameter	Range
Simulation Area	1000 X 1000 m^2
Number of nodes	0 – 150 (50 in general)
Maximum connection distance	200 m
Bandwidth between nodes	128 – 512 kbps
Messages sent per minute in the network	100
Routing Algorithm	DSR(modified to remove RERR)

PERFORMANCE EVALUATION

The results are studied to analyze the metrics such as selective behavior, number of selfish nodes in the network, speed of the nodes and total number of nodes in the network. We study the effect of these parameters on the time taken to detect selfish nodes in the network.

6.1 Time Taken to Detect Selfish Nodes vs. Selective Behavior

We define the time taken to detect selfish nodes as the time taken to find 75% of the selfish nodes in the network. The other 25% may take time depending on their position in the network and their presence in path to the destination. We define selective behavior as the probability that a selfish node drops packets instead of forwarding them. A node with selective behavior of 100% will drop all the packets and a node with selective behavior 25% will drop around 25% of the packets and forward 75% of the packets. So a node drops packets with probability equal to the selective behavior. We perform the experiment by fixing the number of selfish nodes in the network to 8 and varying their selective behavior from 100% to 25%. The plot of time taken vs. selective behavior is shown in Figure 6.1. From the figure, we observe that as the selective behavior by nodes increases, i.e. as selective behavior changes from 100% to 25%, the time taken to detect selfish nodes increases linearly in our protocol and 2-ACK protocol while it increases

exponentially in DDSIR. The 2-ACK algorithm has values till 90% selective behavior as it can only find a selfish node if the selfish behavior is above 90% and below 90% the method fails.

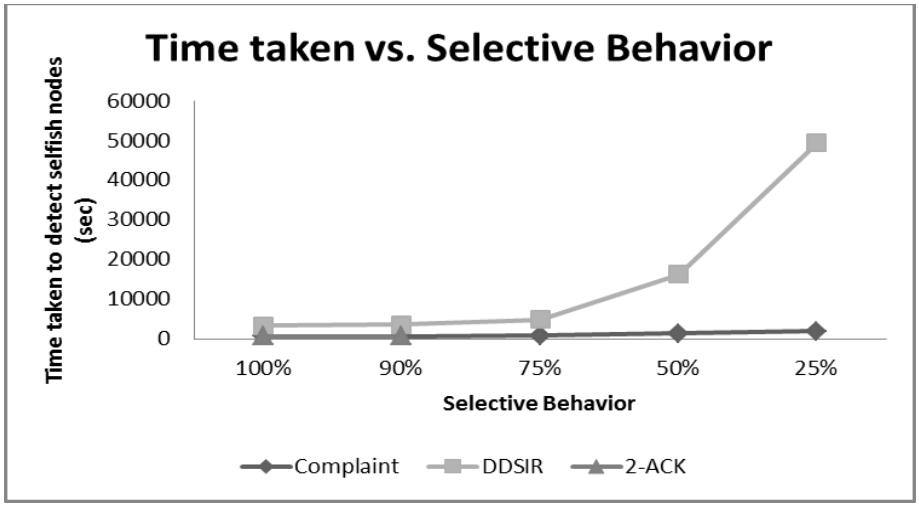

Figure 6.1 Time taken vs. selective behavior

Initially when the selective behavior is 100%, we see that DDSIR takes little more time while 2-ACK takes almost the same time compared to our protocol. But as the selective behavior changes from 100% to 25%, the time taken by DDSIR is very huge compared to our protocol. 2-ACK protocol takes almost same time compared to our algorithm. 2-ACK can be used to detect selective behavior till 90% as overhead due to ACK's is reduced. The difference in time between DDSIR and our protocol is because our protocol and 2-ACK protocol find selfish nodes from the number of packets dropped by the node while DDSIR finds selfish nodes from the percentage of packets dropped. This experiment shows that our method find selfish nodes much faster compared to DDSIR. The variation in the values is found to be ±10%. Our protocol cannot be used to find selfish nodes if the selective behavior reaches 5%.

6.2 Time vs. % of Selfish Nodes Detected

In all the remaining experiments, we fix the selective behavior at 100%.We also fix the total number of nodes in the network to 50 and number of selfish nodes to 10.We find the time taken to detect each selfish node in the network and plot a graph to show the percentage of selfish nodes detected and the time taken. The plot between time and the % of selfish nodes detected is shown in Figure 6.2.

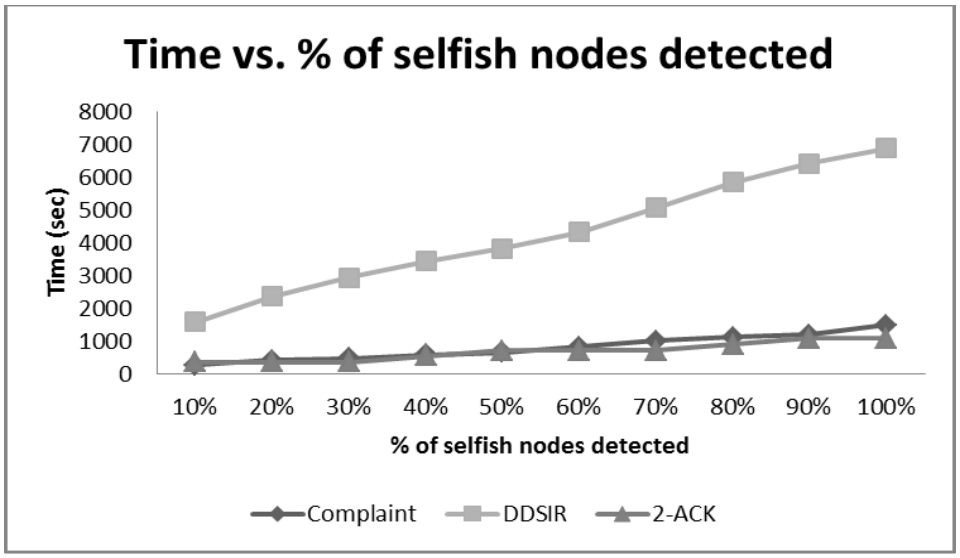

Figure 6.2 Time vs. % of selfish nodes detected

From Figure 10, we observe that our protocol is able to find selfish nodes at a similar rate compared to 2-ACK and faster than DDSIR. We also perform better in terms of finding the number of false positives in the network. This is also due to the mobility of the nodes and the distinct complaints while 2-ACK protocol generates a few false positives. DDSIR and 2-ACK finds selfish nodes from neighboring nodes but with mobility and with neighboring nodes changing DDSIR and 2-ACK takes longer time to detect selfish nodes. Since our protocol considers the whole network, we are able to find selfish nodes much faster.

The variation in the values found during the experiment is ±15% but our protocol shows better results even after considering the variation.

6.3 Time Taken to Detect all Selfish Nodes vs. No. of Selfish Nodes

We define the time taken to detect all selfish nodes as the difference between time at which all the selfish nodes are detected and the time at which nodes start sending messages randomly. We perform the experiment by fixing the total number of nodes to 50 and varying the number of selfish nodes in the network from 4 to 16. We then find the time taken to detect all the selfish nodes in the network. The plot of time taken to detect all selfish nodes vs. number of slefish nodes is shown in Figure 6.3. We observe that as the number of selfish nodes in the network increases, the time taken to detect all selfish nodes also increases linearly with our protocol peforming similar to 2-ACK algorithm but outperforms the DDSIR algorithm.

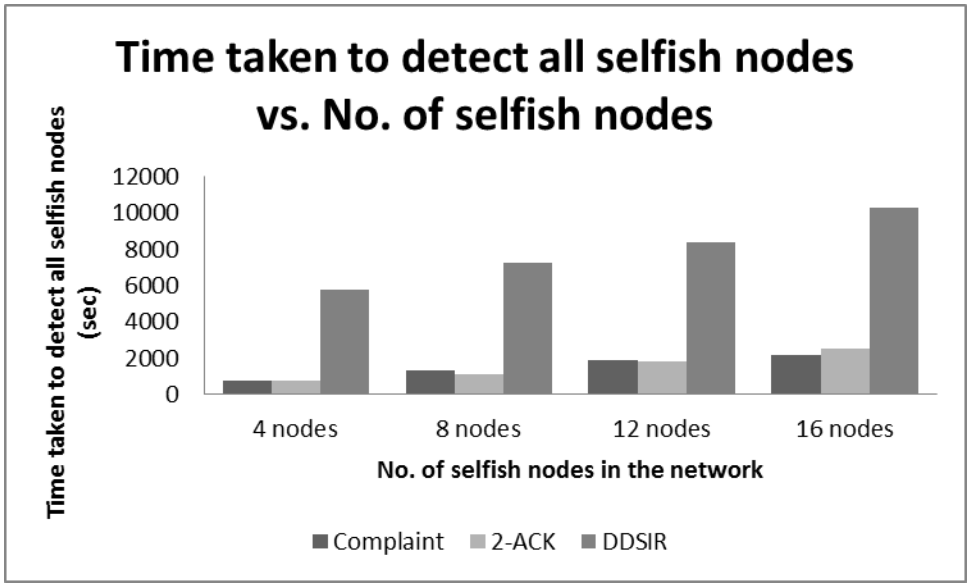

Figure 6.3 Time taken to detect all selfish nodes vs. No. of selfish nodes in the network

When the number of selfish nodes in the network is low, our protocol performs similar to 2-ACK but outperforms DDSIR. As the number of slefish nodes in the network increases, in all the methods time taken increase linearly and our protocol still performs similar to 2-ACK and better compared to DDSIR. The variation in the values is found to be ± 6.5%. The variation is very low and hence the network finds selfish nodes faster using our protocol.

6.4 Time Taken to Detect Selfish Nodes vs. Speed of Nodes

We define the time taken to detect selfish nodes as the time taken to find 75% of the selfish nodes in the network. The other 25% may take time depending on their position in the network and their presence in path to the destination. We perform the experiment by fixing the number of selfish nodes to 8 and increasing the speed of nodes in the network from 5 m/s to 20 m/s. The plot between time taken to detect selfish nodes and speed of nodes is shown in Figure 6.4. We observe from the figure that as the speed of the nodes increases, the time taken to detect selfish nodes decreases in our protocol while the time taken increases in DDSIR and 2-ACK algorithms.

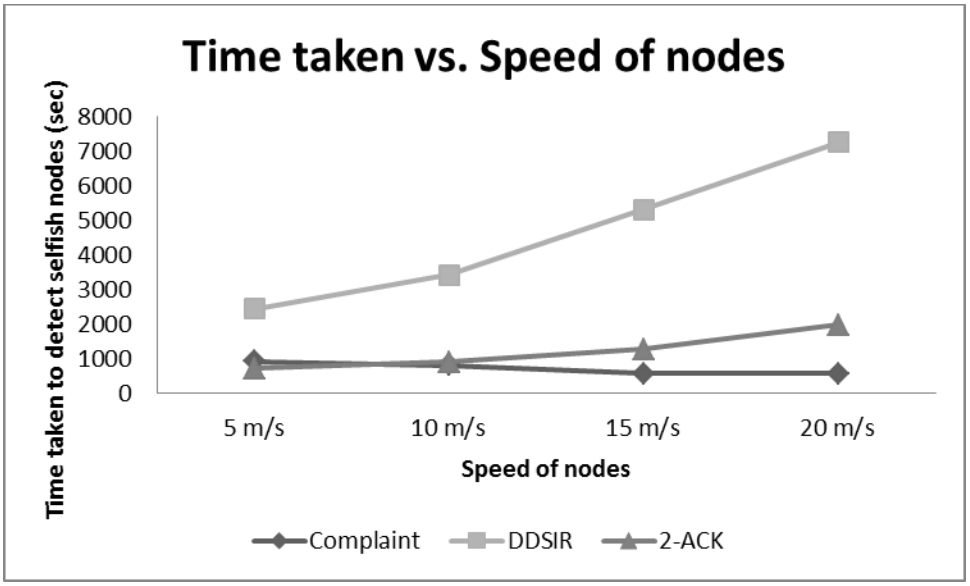

Figure 6.4 Time taken to detect selfish nodes vs. Speed of nodes

When the speed of nodes is low, our protocol performs better compared to DDSIR and similar to 2-ACK. As the speed of the nodes increases, the time taken to detect selfish nodes decreases in our protocol. This is due to the fact that we receive distinct complaints as the speed increases. Whereas in DDSIR and 2-ACK, the neighboring nodes keep changing as the speed increases and hence the time taken to find selfish nodes increases. After 20 m/s, as the speed of nodes increases, the time taken in our protocol

also starts increasing slowly. This is because the route found from source to destination fails before the packet is sent. So the number of packets sent also decreases. The variation in values during this experiment is ± 7.5%.

6.5 Time Taken to Detect Selfish Nodes vs. Total no. of Nodes in the Network

We define the time taken to detect selfish nodes as the time taken to find most of the selfish nodes in the network. We want to find the effect of the number of nodes in the network on the time taken to detect selfish nodes. We vary the total number of nodes in the network from 50 to 125 and selfish nodes from 8 to 20 respectively. We find the time taken to detect selfish nodes in the network. The plot between time taken and total number of nodes in the network is shown in Figure 6.5. From the figure, we observe that our protocol performs better than DDSIR and similar to 2-ACK.

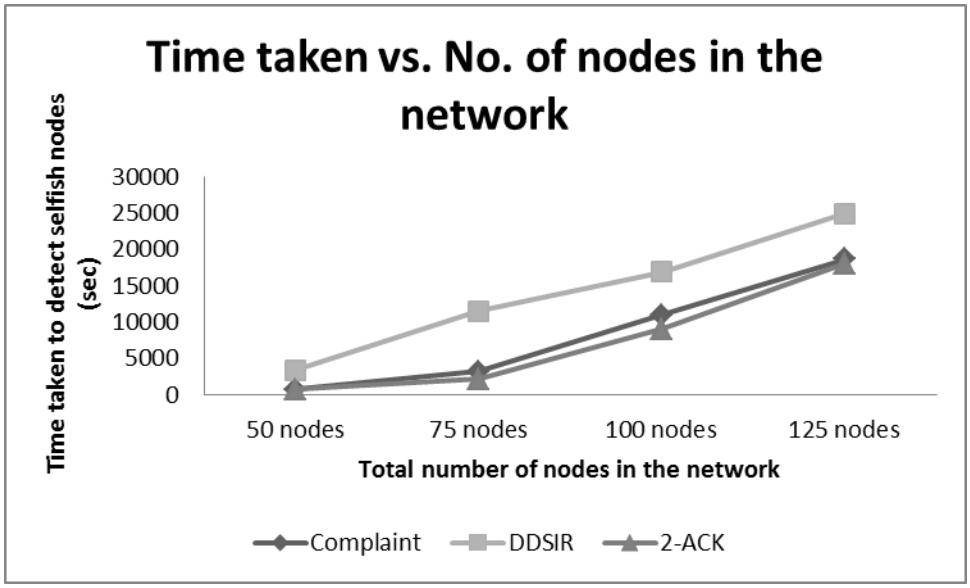

Figure 6.5 Time taken to detect selfish nodes vs. Total no. of nodes in the network

Initially, when the total number of nodes in the network is 50, the time taken to detect selfish nodes is almost similiar for our protocol and 2-ACK but more using DDSIR algorithm. As the total number of nodes in the network increases, the time taken to detect selfish nodes also increases in all the algorithms but increases rapidly in DDSIR. This is

because, as the total no. of nodes increases, the reputation of a node in the network increases in DDSIR but it remains same in our protocol and 2-ACK. The variation in the values found during this experiment is found to be ± 8%.

6.6 No. of Control Packets Generated vs. No. of Data Packets Sent

We define the number of control packets generated as the number of receipt/ACK packets generated by the all the nodes in the network. We define the number of data packets sent as the number of messages sent by one node to another.We increase the number of data messages sent from 200 to 2000 and compare the number of control packets generated by all the nodes in the network. The plot between number of control packets generated and the number of messages sent is shown in Figure 6.6. We observe that the number of control packets generated is almost the same in our protocol and DDSIR but is little higher in 2-ACK. Though the size of control packet in our algorithm is large compared to DDSIR or 2-ACK, the number of control packets generated is low.

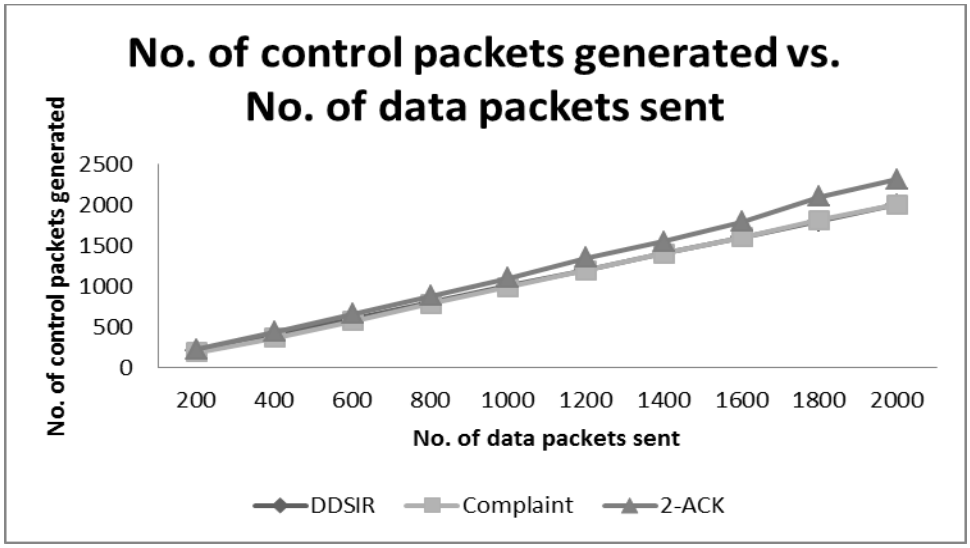

Figure 6.6 Number of control packets generated vs. Number of data packets sent

The number of control packets generated in our protocol depends on the number of receipts sent by all the intermediate nodes and the destinataion. The number of control packets sent in DDSIR is the number of ACK's sent, which is equal to the number of messages sent. The number of 2-ACK's send in 2-ACK algorithm is reduced by sending one ACK for every 2-3 packets received. By fixing the time after which we send receipts to the Broker node, we can send the same number of control packets as generated by DDSIR and lower than 2-ACK. The variation of the values in this experiment is found to be ±3%.

7.CONCLUSION AND FUTURE WORK

Using a combination of Virtual currency and complaints, we are able to find more efficiently the selfish nodes in the network. We can find selfish nodes till the selective behavior of a node reaches 5%. We consider the credibility of nodes so that nodes dropping packets due to buffer constraints or routing constraints are not proven selfish. Although the amount of data sent in a receipt is large, the number of overhead packets is close to the number of ACK messages thus reducing the overhead incurred by the algorithm. Our simulation studies show that our protocol is efficient and faster in finding selfish nodes in the network and performs better even when various parameters like speed of node, no. of selfish nodes are changed. Also we are able to detect/prevent various malicious activities in the network.

Although we have proposed a good scheme to find malicious nodes in the network, it can still be improved by reducing the infrastructure cost incurred from managing Broker nodes and Access Point nodes. We could also include a customized routing protocol and explore new methods to find credibility.

REFERENCES

[1] Refaei M.T, Vivek Srivastava, DaSilva L, Eltoweissy M, "A reputation-based mechanism for isolating selfish nodes in ad hoc networks", MobiQuitous, pp. 3- 11, July 2005.

[2] Aishwarya Sagar Anand Ukey, Menu Chawla, "Detection of Packet Dropping Attack Using Improved Acknowledgement Based Scheme in MANET", IJCSI, vol. 7 issue 4, 2010.

[3] Sheng Zhong, Jiang Chen, Yang Richard Yang, "Sprite: A Simple, Cheat-proof, Credit-based system for Mobile Ad-Hoc networks", INFOCOM, vol.3, pp.1987-1997, 2003.

[4] Xu Wu, Jingsha He, and FeiXu, "An enhanced trust model based on Reputation for P2P networks", SUTC, pp.67-73, 2008.

[5] Anil Jade, Sanjay Kumar Madria, and Mark Linderman, "An Incentive based Routing Protocol incorporating QoS for Mobile Peer to Peer Networks", MDM, pp.285-292, 2009.

[6] Kajun Liu, jing Deng, Pramod K. Varshney, and KashyapBalakrishnan, "An Acknowledgement-Based Approach for the Detection of Routing Misbehavior in MANET's", IEEE Transactions on Mobile Computing, vol. 6, no. 5, pp.536-550, May 2007.

[7] JyotirmoyKarjee, and Sudipta Banerjee, "Tracing the Abnormal Behavior of Malicious Nodes in MANET", WiCOM, pp.1-7, 2008.

[8] A. Karygiannis, E. Antonakakis, and A. Apostolopoulos, "Detecting Critical Nodes for MANET Intrusion Detection Systems", SecPerU, pp.7-15, 2006.

[9] S. Zhou, G. Rogers, M. Hogan, S. Ardon, T. Hu, and A. Seneviratne, "An Incentive based routing Algorithm for improving message forwarding in structured Peer-to-Peer Networks", AusWireless, pp.58, 2007.

[10] Ralph Levien, and Alex Aiken, "Attack-Resistant trust metrics for Public key Certification", *Proc. USENIX Security Symp.*, pp.229-242, 1998.

[11] Ali AydinSelcuk, ErsinUzun, and Mark ResatPariente, "A Reputation-based trust management system for P2P networks", CCGrid, pp.251-258, 2004.

[12] Alberto Blanc, Yi-Kai Liu, and Amin Vahdat, "Designing Incentives for Peer-to-Peer Routing", INFOCOM, vol.1, pp.374-385, 2005.

[13] Naouell Ben Salem, LeventeButtyan, Jean-Pierre Hubaux, and Markus Jakobsson, "A charging and rewarding scheme for packet forwarding in multi-hop Cellular networks",*Proc. Fourth ACM Int'l Symposium, Mobile Ad Hoc Networking and Computing,* pp.13-24, 2003.

[14] Daniel Figueiredo, Jonathan Shapiro, and Don Towsley, "Incentives to promote availability in Peer-to-Peer Systems", ICNP, pp.110-121, 2005.

[15] Rong Cheng, Hai Jin, Ke Shi, and Bin Cheng, "An Anycast-based P2P Routing protocol for Mobile Ad Hoc Networks", IEEE & IFIP, 5 pages, 2005.

[16] Huafeng Wu, Chaojijan Shi, Xi Zhou, Haiguang Chen, and ChuanshanGao, "eSeagull: Design of Mobile P2P file sharing system at Sea", AINAW, pp.954-959, 2008.

[17] Krishna P. N. Puttaswamy, and Ben Y. Zhao, "A Case for Unstructured Distributed Hash Tables", IEEE Global Internet Symposium, pp.7-12, 2007.

[18] David B. Johnson, David A. Maltz, and Josh Broch, "DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks", Ad Hoc Networking, C.E. Perkins, ed., chapter 5, Addison-Wesley 2000.

[19] Charles E. Perkins, and Pravin Bhagwat, "Highly Dynamic Destination-Sequenced Distance –Vector Routing (DSDV) for Mobile Computers", SIGCOMM 1994.

VITA

Hemanth Meka was born on February 21, 1987 in Hyderabad, India. He received distinction in Bachelor of Engineering degree in Computer Science and Engineering from Osmania University, India in 2008. He has been a graduate student in the Computer Science Department at Missouri University of Science and Technology since August 2008 and worked as a Graduate Research Assistant under Dr. Sanjay Kumar Madria from June 2009 to December 2010. He received his Master's in Computer Science at Missouri University of Science and Technology in May 2011.