

Routing and DDoS attacks in MANET

DilipSimha.N.M
CSE, RVCE

nmdilipsimha@gmail.com

Vivek.J
CSE,RVCE

vivekshastry@gmail.com

6-4-2005

Project Guide: Prof Yogananda
Internal guide: H.K.Krishnappa

Abstract

Ad-hoc wireless networks provide direct peer-to-peer communication between nodes without the need for any infrastructure (hubs, access points etc.). Where source and destination are out of range of one another, packets may be forwarded by one or more intermediate nodes. In the simplest system, every node broadcasts its neighbour list from time to time, and a source sends the packet to a forwarder that has the required destination on its neighbour list. This can be extended to more than two hops by using the shortest route based on hop-count. This does not always lead to the selection of the best route when there is a choice. A better approach may be to try to minimise the number of medium access attempts along the entire route (e.g. by choosing hops based on their previous probability of success). However, probability of success depends on two different factors: firstly, the signal strength depends on the length of the hop, and secondly, interference depends on the traffic congestion around the receiving node. The best route may be different in the two directions. A further factor is the length of queues in the source nodes. A hop that appears to have a high probability of success may be chosen as preferred forwarder by many other nodes, and therefore end up with a long queue in its buffers, leading to longer delays than sending the packet over a less reliable route. So a direct measure of delay may be better than probability of success. Algorithms that take account of more sophisticated information sets will generally require more signalling overhead to provide this information, so there is a balance between theoretical performance and what is practically achievable.

The model we have proposed here uses The concept of clusters which contain a leader and some nodes. The leader is responsible for handling all routing information on behalf of the cluster. We use the concept of Packet sequence numbers to avoid unwanted broadcasts at all times hence avoiding problems (such as 'counting to infinity' associated with standard distance vector protocols).

Contents

1	Introduction	3
1.1	How they work	4
2	CADV Terminology	6
3	Message Formats	8
3.1	Data Transfer	8
3.2	Acknowledgement	9
3.3	Route Request	10
3.4	Route Error	11
3.5	RequestIpAddress	12
3.6	IP_Assign_Sorry	12
4	Format of the Tables used	13
5	CADV Routing Operation	14
5.1	Cluster Mechanism	14
5.1.1	Leader	14
5.1.2	Nodes	15
5.1.3	Information stored	15
5.1.4	Merging and Splitting	16
5.1.5	LookUpTable operation	16
5.1.6	Communication b/w nodes and leader	16
5.2	Route establishment	20
5.2.1	Packet Sequence Numbers	20
5.2.2	RouteRequest messages	21
5.2.3	Acknowledgement	21
5.3	Data Transfer	22
5.3.1	Sending and recieving messages	22
5.3.2	Acknowledgement	22
5.3.3	Route Error	22

5.4	Updating tables	23
5.4.1	Frequency of updates	23
5.4.2	Mapping moved nodes	24
5.5	Actions after reboot	25
6	Assigning Ip Addresses	26
6.1	Basic mechanism	26
6.2	Actions on merging clusters	26
6.3	Actions on splitting clusters	27

Chapter 1

Introduction

A mobile ad hoc network is an autonomous system of mobile routers connected by wireless links the union of which forms an arbitrary graph. The routers are free to move randomly and organize themselves arbitrarily; thus, the network's wireless topology may change rapidly and unpredictably. Such a network may operate in a standalone fashion, or may be connected to the larger Internet. Ad hoc networks are planned to work in situations like home automation, disasters (earthquake, war, etc.), mobile or nomad computing (Conferences, interactive shows, etc.) Or any other one where the quick settling up and easy communication is especially important. The vision of mobile ad hoc networking is to support robust and efficient operation in mobile wireless networks by incorporating routing functionality into mobile nodes. Such networks are envisioned to have dynamic, sometimes rapidly-changing, random, multihop topologies which are likely composed of relatively bandwidth-constrained wireless links. We adopt distance vector Routing using cluster based techniques. So this has a huge advantage of using minimal table space and fast routing.

MANETs have several salient characteristics:[1]

- Dynamic topologies: Nodes are free to move arbitrarily; thus, the network topology—which is typically multihop—may change randomly and rapidly at unpredictable times, and may consist of both bidirectional and unidirectional links.
- Bandwidth-constrained, variable capacity links: Wireless links will continue to have significantly lower capacity than their hardwired counterparts. In addition, the realized throughput of wireless communications—after accounting for the effects of multiple access, fading, noise, and interference conditions, etc.—is often much less than a radio's maximum transmission rate.

One effect of the relatively low to moderate link capacities is that congestion is typically the norm rather than the exception, i.e. aggregate application demand will likely approach or exceed network capacity frequently. As the mobile network is often simply an extension of the fixed network infrastructure, mobile ad hoc users will demand similar services. These demands will continue to increase as multimedia computing and collaborative networking applications rise.

- **Energy-constrained operation:** Some or all of the nodes in a MANET may rely on batteries or other exhaustible means for their energy. For these nodes, the most important system design criteria for optimization may be energy conservation.
- **Limited physical security:** Mobile wireless networks are generally more prone to physical security threats than are fixed-cable nets. The increased possibility of eavesdropping, spoofing, and denial-of-service attacks should be carefully considered. Existing link security techniques are often applied within wireless networks to reduce security threats. As a benefit, the decentralized nature of network control in MANETs provides additional robustness against the single points of failure of more centralized approaches. In addition, some envisioned networks (e.g. mobile military networks or highway networks) may be relatively large (e.g. tens or hundreds of nodes per routing area). The need for scalability is not unique to MANETs. However, in light of the preceding characteristics, the mechanisms required to achieve scalability likely are.

These characteristics create a set of underlying assumptions and performance concerns for protocol design which extend beyond those guiding the design of routing within the higher-speed, semi-static topology of the fixed Internet.

1.1 How they work

To give a very simple example, let us assume that there is already a small ad-hoc network in place. When a new node in this example it can be the PDA of Tom joins the ad-hoc network, there are a number of things to do: The device needs to set up contact to other nodes in range, telling them: I am here. By this, the new node learns who the neighbour nodes are, and vice versa. Another point is that the new node, in this example the PDA, needs a unique identifier to make it addressable an IP address in IP networks. For

all this, the new node is on its own, as there is neither a central controlling entity nor a pre-existing fixed infrastructure in ad-hoc networks. When Tom wants to send a message from his PDA to that of Maria, other nodes serve as a relay station in a process called multi-hop routing, if the PDA of Maria is not in direct reach, using one of the routing protocols designed for ad-hoc networks.

This small example shows a few imminent advantages of ad-hoc networks: They can extend the range of the wireless technology in use, e.g. WLAN or Bluetooth, they can reduce the nodes power consumption due to a lower transmission power required, and they increase the nodes mobility. To make this work, though, ad-hoc networks require a critical mass of well-behaving nodes, willing to forward others traffic.

Chapter 2

CADV Terminology

- Active route

All entries in the cluster table are cached. So as long as they are present in this table, the routes they indicate are said to be active. These entries are periodically refreshed to avoid wrong information on routes.

- Broadcast

Broadcasting means transmitting to the IP Limited Broadcast address, 255.255.255.255. A broadcast packet may not be blindly forwarded, but broadcasting is useful to enable dissemination of CADV messages throughout the ad hoc network.

- Intermediate/forwarding node

A node that agrees to forward packets destined for another node, by retransmitting them to a next hop that is closer to the unicast destination along a path that has been set up using routing control messages.

- Forward route

A route set up to send data packets from a node originating a Route Discovery operation towards its desired destination.

- Originating node

A node that initiates an CADV message to be processed and possibly retransmitted by other nodes in the ad hoc network. For instance, the node initiating a Route Discovery process and broadcasting the RouteRequest message is called the originating node of the RouteRequest message.

- Reverse route:

A route set up to forward a reply (acknowledgement) packet back to the originator from the destination or from an intermediate node having a route to the destination.

- Previous hop:

The ForwardedPackets table has a cluster-id corresponding to a sequence number. This cluster-id denotes the originating node. So when forwarding acknowledgements or RouteError packets this information is made used. So previous hop denoted the hop just before it in any route.

- Packet Sequence Number:

This consists of the ip address of source concatenated with random number. Packet once sent from the source will have the same sequence number until it reaches the destination. This is used to avoid unnecessary broadcasts.

- Leader:

This is a special node which is elected as a leader of the cluster using standard election algorithm

- Node:

This is representation of the end user in the ad-hoc network. eg: mobilephones, laptops.

Chapter 3

Message Formats

A format is used to identify different packets which are used for specific purposes.

3.1 Data Transfer

$\langle 0a; b; \dots; dest_addr; message \rangle$

- 0 stands for transfer of data.
- a,b are cluster id's separated by some delimiter other than ','
- dest_addr is the ip-address of the destination.
- message is the actual data to be transferred.

3.2 Acknowledgement

$\langle 1; \textit{pointer}; a; b; \dots; \rangle$

- this indicates an acknowledgement packet
- Pointer is used so that the intermediate nodes can suitably update their cluster tables.
- a,b are cluster id's of nodes along the path seperated by some delimiter other than ','

3.3 Route Request

$\langle 2a; b; \dots; packet_number \rangle$

- this indicates a RouteRequest packet
- a,b are cluster id's that r appended along the path
ex: cluster 1 appends id as 2(of next hop) and sends the message to ip address corresponding to this cluster id 2

3.4 Route Error

packet number is the sequence number which is also maintained in forwarding packets table. This is to avoid retransmitting the same packets again and again.

⟨3Packet_Sequence_Number⟩

- This indicates RouteError
- This packet is sent to the node corresponding to the packet_number in the forwardingPacket Table
- This type of packet is sent only when the route is established and data transfer/acknowledgement is taking place

3.5 RequestIpAddress

$\langle 4R; no_of_nodesinthecluster; packetsequencenumber \rangle$

- This indicates the Request for ip_address. this happens on cluster merging
- This is a broadcast message to all neighbours
- The 'R' flag stands for recursive repeat. This is 0 for first time broadcast.

If none of the nodes have enough of pool of ip_addresses then none of the neighbours reply for the first time request. So the source then sends the message with R flag set to 1. On seeing this each neighbour broadcasts this message to their neighbours asking for help. The process goes on. If the neighbour even fails when the R flag is set then it sends a Ip_Assign_Sorry message.

3.6 IP_Assign_Sorry

$\langle 5 \rangle$

This packet type(5) denotes a error message for not finding a ip_address for donation.

Chapter 4

Format of the Tables used

- NodeTable:- ip-address,average time taken to send a message to each of the node from the leader.
- ClusterTable:- ip-address of the destination,route to the destination in terms of cluster ids,frequency counter(for replacement)
- ForwardedPackets table:- Packet sequence number,Cluster id of the previous hop(The leader which sent the packet),time alive(this field is incremented periodically after UPDATE_TIME SECONDS)
- NeighbourTable:- cluster id ,ipaddress of the cluster leader(cluster id information is relative and can be different for the same cluster in different tables)
(eg: cluster A can have id 3 in cluster B and id 5 in cluster C)
- Mapping_old_to_new_ip_address:- old ip address,new ip-address (so that any reference to the moved node will now be forwarded to the new ip address.),time alive(this field is incremented periodically after UPDATE_TIME SECONDS)
- IP_address_pool:- Starting address,number of addresses(it is an array).

Chapter 5

CADV Routing Operation

5.1 Cluster Mechanism

Cluster[3] is a group of nodes which are close to each other. Each cluster has a limit to the number of nodes it can have, both minimum and maximum. Clusters can move and can change its topology dynamically.

5.1.1 Leader

- Each cluster has a leader which has some special functionalities.
- Each leader is elected based on an election algorithm.
- Leader is responsible for receiving messages on behalf of the nodes in its cluster.
- Leader then forwards the messages to the corresponding nodes.

Election algorithm to choose Leader

When the leader dies or moves from one cluster to another, election is held.

To elect new leader:

Every node periodically sends "i am here" message to the leader.

```
if it get's a acknowledgement from it's leader then  
    no problem  
else  
    choose a new leader.  
end if
```

so each node refer to it's nodetable and find's the leader(ex: take average of all time fields and chose the node which is nearest to it.). That leader will send a message to the group "i am the leader" to the group.

To impelement this create a new object of Header class and transfer the settings from node(new leader) to this new object. Thus all nodes update their leader address to this new one and the newly elected leader update's it's old_leader to the leader that died. Now inform all neighbours abt the arrival and build all tables required for the header.

So whenever a message comes in with that(old_leader) ip address, assume it is u and take actions suitably.

If there is only one node then that node is the leader.This takes the entire range of ip addresses. The node which is at the centre is choosen as the leader.Later even if the leader is not at the centre, new leader is not chosen. When clusters merge or split leaders are created or destroyed respectively.This is explained in ip address assignment.

5.1.2 Nodes

Nodes basically represent the end users(eg mobiles). Each Node is responsible to hold enough information about the cluster to take over as a leader(if need arises.)

5.1.3 Information stored

LEADER:

ip_address

NodeTable

ClusterTable

ForwardedPackets Table

NeighbourTable

Mapping_old_to_new_ip_address Table

previous_header

Ip_address_pool Table

NODE:

Ip_address

NodeTable

ClusterTable

Ip_address_poolTable

my_leader
Mapping_old_to_new_ip_address Table

5.1.4 Merging and Splitting

When the number of nodes in the cluster goes below the MIN_NUMBER, it merges with a nearby cluster. The cluster with lesser number of nodes will merge with larger one. The leader of the merged group will be the leader of the larger group.

When the number of nodes in the cluster crosses MAX_NUMBER ,the cluster must split into two clusters. This is explained in the ip address assignment.

5.1.5 LookUpTable operation

Table look up operation is done in the following order :

- Node table
- Neighbourhood table
- Cluster table
- Mapping table

The above tables are searched in order for a given ip address.

Mapping table Handling

If the address is found in the mapping table, then call the RouteReq(destination address) where destination address is the new ip adress in the mapping table

5.1.6 Communication b/w nodes and leader

Nodes and leader communicate frequently either to send or to receive messages. These can be the hello messages or the data transfer messages.

Sending and receiving messages.

LEADER: The leader will handle messages in the following manner:

```
if message is from application layer then
  do LOOKUP TABLE operation
  if found then
    use that address and send
  else
    call ROUTEREQ(dest_ip_address) function //this replies the route
    to the destination and this route information consists of cluster ids.
    First cluster id along the route is looked up in the neighbour table
    and message is sent to the corresponding ip address.
  end if
else if message is from lower layer //It is not the source then
  Obtain the cluster id(next hop) from the route using cluster id pointer.
  If this is the last along the route, then node will be in this cluster so
  lookup node table
  if the id is found in the node table then
    the node is in this cluster so message is sent to that node
  else
    if it is in the mapping table then
      do mapping table handling and then send the data
    else
      it is intermediate node along the route
      if the packet is RouteERR then
        send RouteERR()//the message is sent to the previous node along
        the route.
      else
        look up neighbour table for the cluster id
        if found then
          send data
        else
          send RouteERR()//the message is sent to the previous node
          along the route.
        end if
      end if
    end if
  end if
else
  Impossible case!!!
end if
```

Procedure to send Route error: just see the sequence number of the packet and lookup the forwarding packets table and get the appropriate clusterid and get the the ip-address for this cluster-id from neighbour table.

NODE:

The node will handle messages in the following manner:

```
if message is from application layer then
  do LOOKUP TABLE opeartion
  if found then
    use that address and send the message along the route.
  else
    call ROUTEREQ(dest_ip_address) function //this replies the route
    to the destinationand route information consists of cluster ids.First
    cluster id along the route is looked up in the neighbour table and
    message is sent to the corresponding ip address.
  end if
else if message is from lower layer then
  Extract the message from the packet
  if it is a route error then
    redo the entire process.(remove this cached entry in cluster table).
  else if it is a route request then
    this is the destination node. SO prepare the acknowledgement and
    send back along the route it arrived.
  else if it is a acknowledgement then
    extract the message
    if it is a acknowledgement for data packet then
      remove all temorarily stored data concerned with that packet.
    else
      this will be for RouteRequestmessage.So start sending the data
      along the route discovered now.
    end if
  else
    this is data packet. So accpet it and send a acknowledgement.
  end if
else
  impossible case!!!
end if
```

Hello Messages.

Periodically every node sends a message "Hello Dude" (broadcasts). The leader who sees it will reply back with the message "i am ur leader" and sends the tables...node table,ip_address_pool table,clusterTable,forwardingTable.

The client gets that message and verifies whether the message came from new leader (ie... it has moved to new cluster) or from the already existing leader.

If it's the old leader then it sends back a dummy message. The old leader will get this message and will measure the round trip delay and updates in the node table.

If it's a new leader then it is now responsible for assigning ip-address to this node. So it sends a "i am ur new leader..welcome" message to the node. There may be many cluster leaders replying to this message. So the one which replies earlier is considered.

Now the node selects the new cluster and sends to it the old-ipaddress and the previous cluster leader's ip-address. The cluster leader assigns a new ip-address.

Then it checks if the new node was the leader of some other cluster previously.

if so then that cluster will now be in a process of creating leader. So wait for MAX_LEADER_ELECTION_TIME.

Then it sends a broadcast message to all neighbours (along with the new ip address) asking them to recognise this leader (node which was previously a leader). The newly elected leader in that cluster (from where the node came) will reply to this and makes a mark in its mapping table reflecting this change.

If this node was not a leader previously then the leader sends a message to the cluster leader of this new node in previous cluster announcing the change. That leader will now enter this info in its mapping table and removes the entry from nodetable.

NOTE: The cluster leader removes an entry from its table only when it gets a message from the other cluster announcing the change.

This will always work because: If the leader gets the hello message and sends the reply, but fails to reach client due to some reason, then the client

accepts the offer from any other cluster. So even though the cluster which lost the reply may be closer than the cluster to which the node joins, the decision is correct because the route along that (loss) path has more probability for packet loss than this (chosen) one.

If the node doesn't get any reply for hello message, then it keeps sending the Hello messages until MAX_TIMEOUT_HELLO period. After that interval the node thinks that it is alone and stays idle until it gets message from any other node.

5.2 Route establishment

To establish the route to a given destination, Special control packet called Route Request packets are sent by the source. These packets move through the network until the destination is found. The basic idea is that the source broadcasts the route request packets to its neighbours, which in turn check their respective tables to see if they have the route to the destination in which case the Route request packet continues in that path, otherwise these nodes in turn broadcast the request packets to their neighbours and so on. Hop count field and sequence numbers are used to ensure that the packets do not roam infinitely. These avoid unnecessary broadcasts and information about route once found are stored in the nodes so that these can be made use of in future. The destination once found is supposed to return an acknowledgement back to the source which contains the route in terms of cluster ids.

5.2.1 Packet Sequence Numbers

Packet sequence numbers are globally unique numbers that are assigned to each packet by the originating node. The main use is to avoid unnecessary broadcasts. The sequence number consists of IP address of the originating node followed by the random number which will be globally unique. Now each cluster leader in the network will maintain a Forwarding Packet Table which keeps track of the sequence numbers of the packets it has sent over a certain period of time. So every time a Route request packet arrives, the node checks the table to see whether the packet has already been sent, if it has been sent, then the packet is not broadcasted again, if it is not found then the normal broadcasting of this packet proceeds. This avoids the unnecessary broadcasts thereby reducing the network traffic considerably.

5.2.2 RouteRequest messages

RouteRequest messages are used by the source to find the route to the destination. The source broadcasts the route request packets to its neighbours, which in turn check their respective tables to see if they have the route to the destination. If route is found the Route request packet continues in that path, otherwise these nodes in turn broadcasts the request packets to their neighbours. The nodes receiving these packets look up tables for the destination. If the route is found then the Route Request packet is forwarded only along that route until it reaches destination. This is done because the entry in the node may be outdated and the destination node or any other node along the path may have moved to a different cluster. Every intermediate node through which the Route request passes appends the cluster id of the next node along the path through which it is going to send the packet. eg:If the source node is sending the packet to node A then in that packet it will append relative id 4(for the node A in source Node) and further node A appends the id say 5 of node B along which it sends the route request packet. This continues until the destination is found.Thus Route request packets establish the route to the destination.

Now if no acknowledgement is got then send the route request packet again.(the sequence no changes)

5.2.3 Acknowledgement

The acknowledgement packets are sent from the destination back to the source node to indicate that the route has been successfully found.The acknowledgement packet contains the route from source to the destination in terms of relative cluster ids. The source will use this information to send the data packets to the destination.The destination sends the acknowledgement packet back to the node which sent it the Route request packet .This information is found out using the forwarding packet table which has the packet sequence number and cluster id which sent the packet. Thus the path back to the source is found and the acknowledgement is sent.Now as the acknowledgement is being sent each node receiving it will update the path to the destination in their respective cluster tables.There is a cluster id pointer in the acknowledgement packet which keeps track of the present node.Now in each intermediate node the Path from Clusterid pointer to the end(identified by delimiter) is entered in to the cluster table to keep track of destination address. The cluster id pointer is initially at the destination cluster id and at each node it is decremented.so that each node can have the required path (ie from the pointer to the the delimiter)to the destination.Incase while the

acknowledgement packet is being sent the node positions change, then the route is not correct and Route Error messages should be sent appropriately which is discussed in the Route Error section.[5.3.3]

5.3 Data Transfer

All messages are sent directly from the source node to the destination node without creating excess traffic at the leader. But for receiving messages all messages has to come to the leader and then it is forwarded by the leader to the corresponding node.

5.3.1 Sending and receiving messages

Once the route is established the actual data is transmitted by sending the packets to the next hop along the route. The next Hop along the route(from RouteRequest message) contains the cluster_id and the current_pointer. The source node begins by initializing the pointer to 0.The hop pointed to by this pointer along the route and the corresponding cluster-id is searched in the neighbour table and the ip-address is extracted. Now the node increments the pointer and sends the message to the ip_address extracted. The exact process is explained in the section [5.1.6].

5.3.2 Acknowledgement

To guarantee the delivery of any message, the destination node sends a acknowledgement back to the source. This can be to a data packet or route request packet. The acknowledgement is traced back to the source using the packet_sequence number in the packet. The forwarding packets table is referred for the packet sequence number and a cluster-id is got. This cluster-id is then searched in the neighbour table and the ip-address of the leader is extracted and acknowledgement is sent.

5.3.3 Route Error

When the route is established b/w source and destination, data transfer process begins. Now if a cluster is merged, then the reference to the cluster-id is lost. This is a problem because the cluster-id's that are maintained along the route are relative to each cluster. Now a route error message should be issued.

The packet_sequence_no from the packet is looked up in the forwarding packet table.

```
if found then
  cluster_id is extracted and looked up in the neighbour table
  if found then
    send the route error message to the ip_address corresponding to the
    cluster_id
  else
    ignore the packet.(the source will timeout and will come to know of
    this gradually)
  end if
else
  then this packet is for the node in my cluster or has to be ignored
  if the cluster_id found is -1 then
    this packet is for the node in my cluster.
    . So send the routeerror packet to the node directly using the ip_address
    in the sequence number found in error message.
  else
    This packet is not for me.(common in broadcast!!) So ignore it.
  end if
end if
  Then the node does a route establishment and sends the message.
```

5.4 Updating tables

5.4.1 Frequency of updates

The entries in the tables are periodically updated.

- Neighbour table: The neighbour hood table is updated at regular intervals. Each cluster leader sends Hello messages to the neighbouring clusters, which when receive the above message update their neighbouring table, if any neighbour does not receive a message of a given cluster leader after a certain time (MAX_HELLO_TIME) it removes that entry from the table. Thus this table is updated regularly based on the hello messages.
- Cluster table: Cluster table is updated when ever a route to the new destination is found. If sufficient space is available then it boils to just addind one more entry. In case the table is full, the replacement policy is followed. The policy used is to maintain the frequency of the

usage of the route . whenever a route request packet arrives for a destination already in the table the count is incremented, so while replacing the entry with the least count is replaced and this ensures that the most frequently used routes are kept in the table. Thus the cluster table is updated whenever the acknowledgement from the destination is received and the route is different from the already existing one.

- Forwading packet table: Every time a packet with a new sequence number arrives this table is updated. Again when there is enough space , the updation is trivial.In case the table is full ,then oldest entry is removed from the table. To keep track of the duration of the entry , a time field(TIME ALIVE) is kept which is incremented periodically (ie... every UPDATE_TIME seconds).
- Node table: Even updating node tables is based on the Hello messages. The table is updated periodically (ie after hello messages are sent based on the replies from the nodes).If no reply is received , then the entry is deleted.else updated suitably based on the reply.
- Mapping table: This table is updated whenever a node moves from one cluster to another ie(when the address of the node changes. Again trivial case is when enough space is available.When the table is full agaain the replacement is done based on the time field(TIME ALIVE), which is incremented periodically(UPDATE_TIME).
- IP_Address_pool: This table is updated whenever the two clusters merge in which case a new entry containing starting address and the range is added ,or when a cluster splits when each cluster will have its own table which is initialised suitably to denote the address space that cluster has with itself.

5.4.2 Mapping moved nodes

This table maintains the old ipaddress and the new ip addresses of the nodes that have moved from this cluster so that the nodes can be located. Now whenever the node moves from one cluster to another,the new leader of the cluster to which node now belongs will identify node.And from the node gets information about the old leader and then sends the new ip address that has been assigned to the node,to the old leader so that the node can be tracked. The idea is that when ever a route request comes for this moved node , the request messages can be sent to the appropriate cluster without too much

additional costs. The replacement policy is used in case required. The oldest entry is removed.

5.5 Actions after reboot

When a node reboots all the information it had stored is lost. So when it comes to life again, it starts off by generating a "hello" message. The leader in the vicinity observes this node and gives it a ip_address and thus the node is established in that cluster. The detailed process is explained in section [6] on page number [26].

Chapter 6

Assigning Ip Addresses

Ip-address keeps changing when nodes move from one cluster to another. So we have developed a algorithm to dynamically handle this.

6.1 Basic mechanism

[2]

Initially there is only one cluster. Whenever a new node comes up, it joins a cluster, the cluster leader is responsible to assign a ip_address to this new node. The cluster leader refers to it's pool of ip_addresses to allocate a ip_address. After this operation leader updates the ip_address_pool table suitably. A pool of ip_addresses is maintained so that any leader can donate some part of it to a new leader(in case of a split).

6.2 Actions on merging clusters

If the no of nodes in a cluster crosses below the min limit(MIN_NUMBER) then it has to join with one of it's neighbouring clusters. So the leader sends messages to it's neighbours asking for merging. The earliest of the replies is considered and then the two clusters merge. The cluster which merged, that leader will donate all its ip-address to the leader of the big cluster. also this leader will inform it's nodes abt the new leader. Later all nodes will be updated about the topology of the cluster.

6.3 Actions on splitting clusters

If the cluster limit crosses `MAX_NUMBER` then the cluster head should split the cluster into two parts. Now the leader recognises the nearest half of the group and takes them into it's cluster. Then among the other half, the leader is chosen by taking the average of the timefields and finding the nearest matching node. So two clusters are formed. The leader will donate some part of ip-addresses to the new leader. This is done by:

- If there are more than one entry in the `ip_address_pool` table give-away the block with least no of ip-addresses(ie the last element of the array)
- If there is only one entry then split that block into 2 and give-away one half.
- If there is only one entry and it has not enough number of ip_addresses to donate to the new leader, then it asks for help from neighbouring clusters and donates them to it.

After this operation update the `ip_address_pool` table suitably.

Bibliography

- [1] Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. S. Corson, J. Macker, January 1999
- [2] IP ADDRESS ASSIGNMENT IN A MOBILE AD HOC NETWORK, Mansoor Mohsin and Ravi Prakash, MILCOM 2002
- [3] University Model for Designing Ad Hoc Networks, Jnos Horvth Cz; Sndor Imre

Index

- acknowledgement, 9, 21
- active route, 6
- address pool, 13, 24, 26
- application layer, 17, 18
- autonomous, 3

- bandwidth, 3
- broadcast, 6, 19, 20

- cluster, 14
- cluster table, 6, 13, 23
- clusterid pointer, 21
- communication, 16
- congestion, 4

- donate, 26
- dynamic topology, 3

- election algorithm, 14
- energy, 4

- forwarded packets, 13
- forwarding packet table, 24

- hello, 25
- hello message, 19

- idle, 20
- intermediate node, 21
- ip_assign_sorry, 12

- leader, 7, 14, 26
- lookup table, 16, 18

- MANET, 3
- mapping, 13, 17, 24

- MAX_HELLO_TIME, 23
- MAX_NUMBER, 27
- MAX_TIMEOUT_HELLO, 20
- merge, 16, 24, 26
- message format, 8
- MIN_NUMBER, 26

- neighbour table, 13, 23
- neighbours, 12
- node, 7, 15, 18
- node table, 13, 24

- PDA, 4
- previous hop, 7

- range, 24
- recieve, 17
- replacement, 25
- replacement policy, 23
- request ip-address, 12
- route error, 11, 17
- route request, 10, 18
- router, 3

- security, 4
- semistatic, 4
- send, 17
- sequence number, 7, 20
- split, 16, 27

- TIME ALIVE, 24
- topology, 3, 26
- traced, 22

- UPDATE_TIME, 24
- update_time, 13