INTERNET-DRAFT
draft-ietf-manet-cbrp-spec-01.txt

Mingliang Jiang
Jinyang Li
Y.C. Tay
National University of Singapore

14 August 1999

Cluster Based Routing Protocol(CBRP)

Status of this Memo

Abstract

Cluster Based Routing Protocol (CBRP) is a routing protocol designed for use in mobile ad hoc networks. The protocol divides the nodes of the ad hoc network into a number of overlapping or disjoint 2-hop-diameter clusters in a distributed manner. A cluster head is elected for each cluster to maintain cluster membership information. Inter-cluster routes are discovered dynamically using the cluster membership information kept at each cluster head. By clustering nodes into groups, the protocol efficiently minimizes the flooding traffic during route discovery and speeds up this process as well. Furthermore, the protocol takes into consideration the existence of uni-directional links and uses these links for both intra-cluster and

inter-cluster routing.

This updated draft has a more detailed description of the cluster formation and routing process. In addition, it describes 2 major new features that have been added to the protocol: route shortening and local repair. Both features make use of the 2-hop-topology information maintained by each node through the broadcasting of HELLO messages. The route shortening mechanism dynamically shortens the source route of the data packet being forwarded and informs the source about the better route. Local route repair patches a broken source route automatically and avoids route rediscovery by the source.

1. Introduction

There are several major difficulties for designing a routing protocol for MANET. Firstly and most importantly, MANET has a dynamically changing topology due to the movement of mobile nodes which favors routing protocols that dynamically discover routes (e.g. Dynamic Source Routing[4], TORA[3], ABR[5] etc.) over conventional distance vector routing protocols [6]. Secondly, the fact that MANET lacks any structure makes IP subnetting inefficient. However, routing protocols that are flat, i.e. have no hierarchy, might suffer from excessive overhead when scaled up. Thirdly, links in mobile networks could be asymmetric at times. If a routing protocol relies only on bi-directional links, the size and connectivity of the network may be severely limited; in other words, a protocol that makes use of uni-directional links can significantly reduce network partitions and improve routing performance.

CBRP has the following features:

* fully distributed operation.

* less flooding traffic during the dynamic route discovery process.

* explicit exploitation of uni-directional links that would otherwise be unused.

* broken routes could be repaired locally without rediscovery.

* sub-optimal routes could be shortened as they are used.

The idea of using clusters for routing in Ad hoc networks has appeared in [7],[8]. In these protocols clusters are introduced to

minimize updating overhead during topology change. However, the overhead for maintaining up-to-date information about the whole network's cluster membership and inter-cluster routing information at each and every node in order to route a packet is considerable. As network topology changes from time to time due to node movement, the effort to maintain such up-to- date information is expensive and rarely justified as such global cluster membership information is obsolete long before they are used. In comparison, simpler and smaller clusters are found in [9] and [10]; however, the use of these clusters is mainly for the task of channel assignment --- how they can help in the routing process is not discussed.

CBRP adopts the cluster formation algorithm as proposed in [9], but unlike [9], CBRP mainly concentrates on the use of clusters in the routing process.

2. CBRP terminology

This section defines terms used in CBRP that do not appear in [2]:

* Node ID

Node ID is a string that uniquely identifies a particular mobile node. Node IDs must be totally ordered. In CBRP, we use a node's IP address as its ID for purposes of routing and interoperability with fixed networks.

* Cluster

A cluster consists of a group of nodes with one of them elected as a cluster head. The cluster formation procedure is described in section 6.1.

A cluster is identified by its Cluster Head ID. Clusters are either overlapping or disjoint. Each node in the network knows its

corresponding Cluster Head(s) and therefore knows which cluster(s) it belongs to.

* Host Cluster

A node regards itself as in cluster X if it has a bi-directional link to the head of cluster X. In such a case, cluster X is a host cluster for this node. A node could have several host clusters.

* Cluster Head

A cluster head is elected in the cluster formation process for each cluster. Each cluster should have one and only one cluster head.

The cluster head has a bi-directional link to every node in the cluster.

A cluster head will have complete knowledge about group membership and link state information in the cluster within a bounded time once the topology within a cluster stabilizes.

Conceptually, two cluster heads are not allowed to have a direct bi-directional link to each other. If such a link exists, one of the cluster head will relinquish its role as cluster head to the other. However in CBRP, we do allow two cluster heads to be able to hear each other directly for CONTENTION_PERIOD seconds before one cluster head has to give up its cluster head status; this delays postpones any cluster re-organization, in case the two clusters are next to each other only in passing.

* Cluster Member

All nodes within a cluster EXCEPT the cluster head are called members of this cluster.

* Gateway Node

Any node a cluster head may use to communicate with an adjacent cluster is called a gateway node.

* HELLO message

All nodes broadcast HELLO messages periodically every HELLO_INTERVAL seconds; a node's HELLO message contains its Neighbor Table and Cluster Adjacency Table. A node may sometimes broadcast a triggered HELLO message in response to some event that needs quick action.

3. Conceptual Data Structures

* Neighbor Table

The neighbor table is a conceptual data structure that we employ for link status sensing and cluster formation. Each entry contains
- the ID of the neighbor that it has connectivity with and
- the role of the neighbor (a cluster head or a member).
- the status of that link (bi-directional or uni-directional)

* Cluster Adjacency Table

The Cluster Adjacency Table keeps information about adjacent clusters

and is maintained by CBRP's Adjacent Cluster Discovery procedure. Each entry contains:

- the ID of the neighboring cluster head
- the gateway node (a member) to reach the neighboring cluster head
- the status of the link from the gateway to the neighboring cluster head (bi-directional or uni-directional)

* Two-hop Topology Database

In CBRP, each node broadcasts its neighbor table information periodically in HELLO packets. Therefore, by examining the neighbor table from its neighbors, a node is able to gather `complete' information about the network topology that is at most two-hops away from itself. This two-hop topology information is kept in a data structure in each node.

4. Physical and Link Layer Assumptions

This section lists the assumptions we made about the underlying physical and link layers when designing CBRP.

Each MANET node that runs CBRP is equipped with one wireless transceiver. CBRP is capable of handling multiple transceivers per host and multiple hosts per router if the concept of a router ID is introduced. For example, a host with multiple transceivers may select the smallest IP interface address as its router ID.

CBRP assumes omnidirectional antennas. Each packet that a node sends is broadcast into the region of its radio coverage. CBRP is designed to operate on top of a single-channel broadcast medium, however, it also accommodates the presence of multiple channels by forming different sets of clusters for each channel for the same group of mobile nodes in a manner similar to that described in [11].

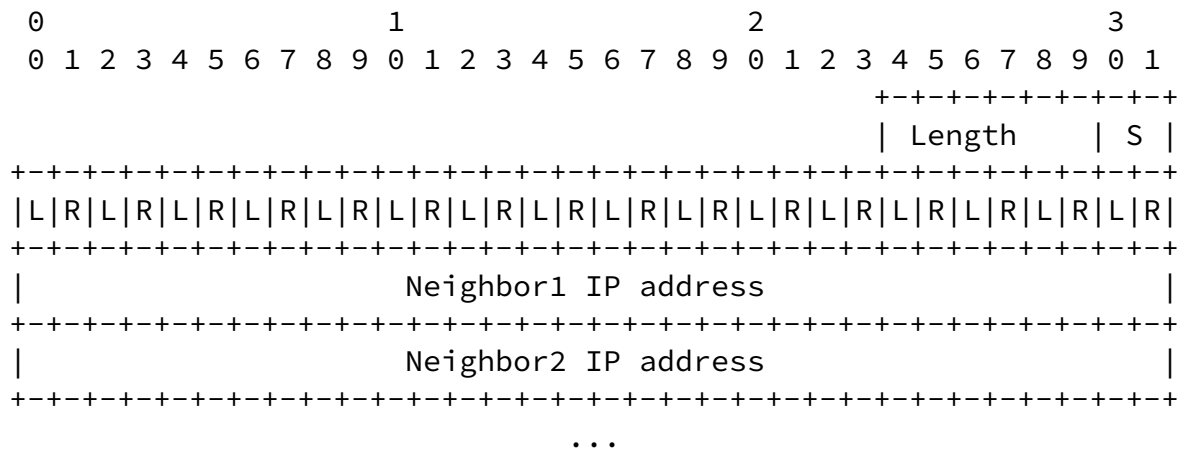
5. Link/Connection Status Sensing Mechanism

In CBRP, each node knows its bi-directional links to its neighbors as well as uni-directional links from its neighbors to itself. For this purpose, each node maintains a Neighbor Table as follows:

NEIGHBOR_ID	LINK_STATUS	ROLE
neighbor 1	bi/unidirectional link to me?	is 1 a cluster head?
neighbor 2	bi/unidirectional link to me?	is 2 a cluster head?
...		
neighbor N	bi/unidirectional link to me?	is N a cluster head?

Each node periodically broadcasts its Neighbor Table in a HELLO mes-

sage, as shown below, every HELLO_INTERVAL.



Length The number of neighbours listed.

S(Status) The current status of the sender.
0 --- Undecided (C_UNDECIDED)
1 --- Cluster Head (C_HEAD)
2 --- Cluster Member (C_MEMBER)

L Link Status of the corresponding neighbor of the sender.
0 --- LINK_BIDIRECTIONAL
1 --- LINK_FROM

R Role of the corresponding neighbor of the sender.
0 --- Non-Cluster-Head(C_MEMBER or C_UNDECIDED)
1 --- Cluster Head (C_HEAD)

The IP addresses of the neighbors and the L and R bits are taken from the neighbor table. The ID of the sender is specified in the source field of the IP header. The status field tells whether the sender is a cluster head, member or undecided. (The state of Undecided is defined in Section 6.1) An 4 byte long L/R bit block appears after every 16 neighbor addresses. In the case that there are no more than 16 neighbors listed, there will be only one such L/R bit block.

Upon receiving a HELLO message from its neighbor B, node A modifies its own Neighbor Table as follows:

1. It checks if B is already in the Neighbor Table; if not, it adds one entry for B if it has heard from B in the previous HELLO_INTERVAL before (i.e. A enters B in its Neighbor Table only when A has heard from B's HELLO messages twice in HELLO_INTERVAL interval).

   If B's Neighbor Table contains A,
   A marks the link to B as bi-directional in the relevant entry

   else A marks the link to B as uni-directional (uni-directional from B to A).

2. If B is already in A's Neighbor Table,

   2.1. If the link_status field of B's entry says bi-directional but A is not listed in B's hello message, then change it to uni-directional;

   2.2. If the link_status field of B's entry says uni-directional but A is listed B's hello message, then change it to bi-directional.

3. Update the role of B in the Role field of B's entry.

Each entry in the Neighbor Table is associated with a timer. A table entry will be removed if a HELLO message from the entry's node is not received for a period of (HELLO_LOSS+1)*HELLO_INTERVAL, allowing HELLO_LOSS consecutive HELLO messages to be lost from that node.

When a node's neighborhood topology stabilizes, the Neighbor Table of a node will have complete information of all the nodes that have a

bi-directional or uni-directional link to it within a bounded time. However, a node would not know to whom it has a uni-directional link. For example in Figure 1, the Neighbor Table of node 7 will show that 4 has a uni-directional link to it, however node 4 would not know of the existence of such a link.

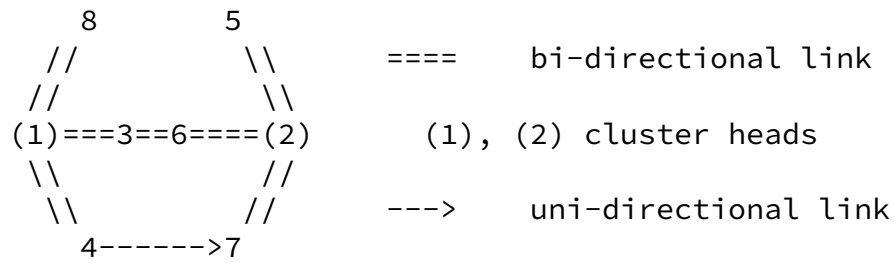


Fig. 1

Note that the 2nd and 3rd column of the neighbor table and the status field of the hello message need not be used for the purpose of link sensing; they are there mainly for cluster formation and adjacent cluster discovery, which will be discussed in Section 6.

In addition to maintaining the Neighbor Table at each node, the periodic broadcasting of HELLO messages also enables each node to gain complete information about all bi-directional links within two-hops. Such information is kept in a data structure we call the two-hop-topology database. By checking its 2-hop-topology database, a node can tell who are its 2-hop bi-directionally linked neighbors and through which intermediate nodes can they be reached. The format of this 2-hop-topology database is implementation dependent.

6. Protocol Operation

The operations of CBRP are entirely distributed. The major components are: Cluster Formation, Adjacent Cluster Discovery and Routing.

6.1 Cluster Formation

The goal of Cluster Formation is to impose some kind of structure or hierarchy in the otherwise completely disorganized ad hoc network. The algorithm is a variation of the simple "lowest ID" clustering algorithm in which the node with a lowest ID among its neighbors is elected as the Cluster Head [11].

Apart from the states of C_MEMBER and C_HEAD, we define a transitional state called C_UNDECIDED for smoother operation of cluster formation. "Undecided" means that a node is still in search of its

host cluster. All nodes wake up in the Undecided state. We will refer to a node in the undecided state as an undecided node hereafter.

A node uses the information obtained from the HELLO messages for Cluster Formation. An Undecided node schedules a u_timer to go off in UNDECIDED_PD seconds and broadcast a HELLO message whenever it enters the C_UNDECIDED state. When a cluster head receives a HELLO message from an Undecided Node, it will send out a triggered HELLO message immediately. If an undecided node receives a HELLO message from a Cluster Head indicating a bi-directional link in between, it aborts its u_timer and sets its own status to C_MEMBER. When the u_timer times out, if the node's Neighbor Table contains no bi-directional neighbors, then it re-enters the Undecided state; otherwise it elects itself as a Cluster Head. The new Cluster Head will change the first field in its subsequently broadcast HELLO messages from C_UNDECIDED to C_HEAD thereafter.

A cluster head regards all the neighbors that it has bi-directional links to as its member nodes. A node regards itself as a member node for a particular cluster if it has a bi-directional link to the corresponding cluster head. Note that a member node may hear from several cluster heads and therefore have several host clusters; its host cluster heads are implicitly listed in the HELLO messages it broadcasts.

As clusters are identified by their respective cluster heads, we would like to have the cluster heads change as infrequently as possible. We use the following rules for changing cluster head, as described in [10].

1. A non-cluster head never challenges the status of an existing cluster head, i.e. if X is a non-cluster head node with a bi-directional link to cluster head Y, X does not become a cluster head even if it has an ID lower than Y's.
2. When two cluster heads move next to each other (i.e. there is a bi-directional link between them) over an extended period of time (for CONTENTION_PERIOD seconds), then only will one of them lose its role of cluster head.

As a result, whenever a cluster head hears HELLO messages from another cluster head indicating a bi-directional link, it sets c_timer to expire in CONTENTION_PERIOD seconds. When c_timer expires, it will check if it is still in contention with the other

cluster head, by checking if the other cluster head is still in its neighbor table. If so, it compares its own ID with that of the other cluster head's. The one with a smaller ID will continue to act as cluster head. The one with a bigger ID gives up its role as cluster head and changes from C_HEAD to C_MEMBER in its subsequent HELLO

messages. This might trigger reorganization of other clusters.

Whenever a member node's last cluster head entry times out, this node checks among its bi-directionally linked neighbors if it has the lowest ID, if so it changes its state to C_HEAD and sends out a triggered HELLO, otherwise it goes to C_UNDECIDED state.

A simplified state transition diagram without considering unidirectional links for Cluster Formation is shown in Appendix A.

6.2 Adjacent Cluster Discovery

Cluster X and cluster Y are said to be bi-directionally linked, if any node in cluster X is bi-directionally linked to another node in cluster Y, or if there is a pair of opposite uni-directional links between any 2 nodes in cluster X and cluster Y respectively. For example in Figure 2, cluster 1 and cluster 2 are bi-directionally linked by the pair of links 3->4 and 5->6.

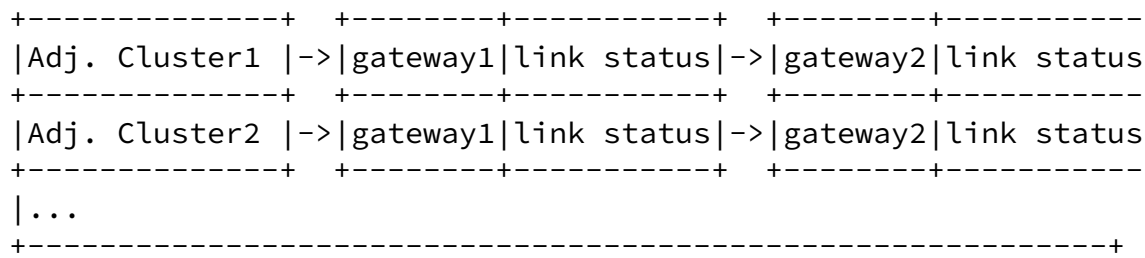
Cluster X is said to be uni-directionally linked to cluster Y if they are not bi-directionally linked and if there exists some node in cluster X that is uni-directionally linked to some node in cluster Y. X is called Y's upstream uni-directionally linked adjacent cluster, and vice versa. For example, in Figure 1, cluster 1 is cluster 2's upstream uni-directionally linked adjacent cluster.

The goal of Adjacent Cluster Discovery is for a cluster to discover all its bi-directionally linked adjacent clusters. For this purpose, each node keeps a Cluster Adjacency Table (CAT) that records information about all its neighboring cluster heads.

Note that for member nodes, neighboring cluster heads are always two hops away and can be discovered by checking received HELLO messages. For a cluster head, its neighboring cluster heads could be 2 or 3 hops away (see Figure 2). Using the HELLO messages alone, a cluster

head is able to discover the cluster heads 2 hops away. However, it has to rely on its member nodes' Cluster Adjacency Table to discover those neighboring cluster heads that are 3 hops away.

The format of CAT at each node is as follows:



Gateway field contains the ID of the gateway node through which the neighboring cluster head could be reached. Gateway node is the member node of the adjacent cluster and hence always has a bi-directional link to the neighboring cluster head. The link status refers to the status of the link between a node's gateway node and itself. The possible values are LINK_BIDIRECTIONAL, LINK_FROM, LINK_TO where LINK_FROM means there is a uni-directional link from gateway node to itself and LINK_TO means there is a uni-directional link from itself to the gateway. Note that there may be several gateway nodes leading towards a particular adjacent cluster.

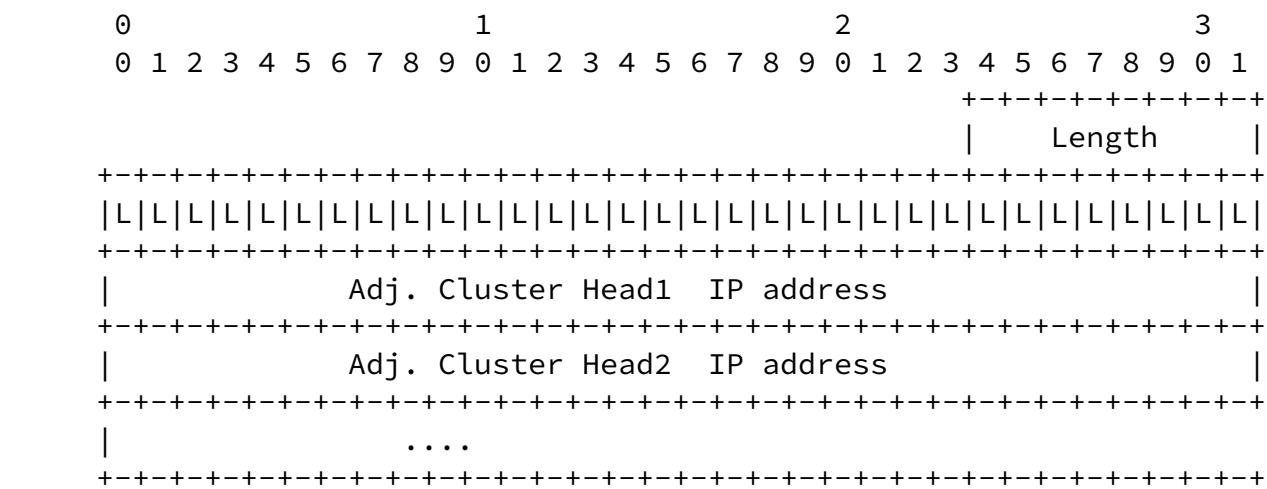
This table is updated by the periodic HELLO messages a node hears. A member node finds out cluster heads that are exactly 2 hops away and records them in its CAT. In particular, whenever a node A receives a HELLO message from B, it scans through the message's list of entries. If there is a cluster head C and the link status of the entry is LINK_BIDIRECTIONAL (i.e. B is a member node of cluster C) and, furthermore, C is not already A's host cluster, A adds an entry in CAT for adjacent cluster C with gateway node B and link status specifying status of link between A and B.

In order for cluster heads to gain information on their adjacent clusters that are 3 hops away, each member node broadcasts its summarized CAT as a Cluster Adjacency Extension to the HELLO message. (Only member node will send out this information, this is not the case with cluster heads.) The rules for summarizing is as follows:

- If there is at least one gateway with whom the node has a bi-directional link, it advertise the neighboring cluster as bi-directionally reachable.

- If there are only uni-directional links (LINK_FROM), the neighboring cluster will be advertised as LINK_FROM. (Note that, by HELLO messages alone, a node is not able to detect any LINK_TO link to the gateway.)

The format of the Cluster Adjacency Extension to HELLO message is shown below:



Length The number of clusterheads listed in the Extension.

L Link Status of the corresponding Adjacent Cluster head. If there is at least one gateway node having bi-directional link to the Adjacent Cluster head, the Link status is specified as LINK_BIDIRECTIONAL.

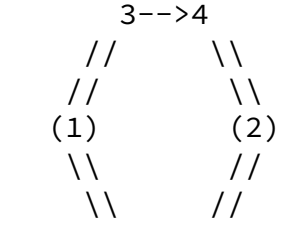
Otherwise, it is LINK_FROM.

0 --- LINK_BIDIRECTIONAL
1 --- LINK_FROM

In addition to using HELLO messages to construct its CAT consisting of 2-hop away neighboring cluster heads, a cluster head could check its members' Cluster Adjacency Extension to find out its 3-hop away neighboring cluster heads in its CAT. In particular, suppose cluster head A receives B's HELLO message. After updating its CAT with the Neighbor Table entries in HELLO, A proceeds to check B's Cluster Adjacency Extension in HELLO, if it finds adjacent cluster head C that is not already reachable within 2 hops, it creates a new entry in CAT for it with the gateway node as B and link status as specified in the extension.

If a cluster head finds that some adjacent clusters are reachable only through a LINK_FROM uni-directional link, it will flood its neighborhood with a message of TTL 3 in search for a "to" link that corresponds to this "from" link. This message will contain the corresponding entry for the adjacent cluster. When a cluster head receives such a message searching for it, it will check if it contains a corresponding LINK_FROM entry to the source cluster head. If so, it will send out a reply with the corresponding CAT entry and update its CAT with the new LINK_TO entry as specified in the

message. As a result, cluster heads will have complete knowledge of all its bi-directionally linked adjacent clusters even if there is no actual bi-directional links in between. For example, in Figure 2, Cluster 1 knows that 2 can reach cluster 1 through 5, but 1 does not know that cluster 2 can be reached through node 3. In CBRP, cluster head 1 and 2 will discover this scenario and disseminate the information to node 3 and 5 respectively.



1 and 2 are heads of their respective clusters, 3,4,5,6 are members.

6<--5

Fig. 2

6.3 Routing Considerations

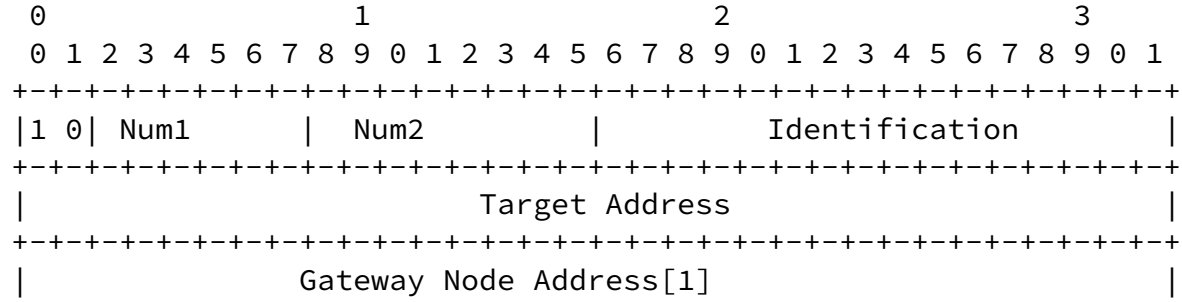
Routing in CBRP is based on source routing. It can be viewed as consisting of 2 phases: route discovery and the actual packets routing.

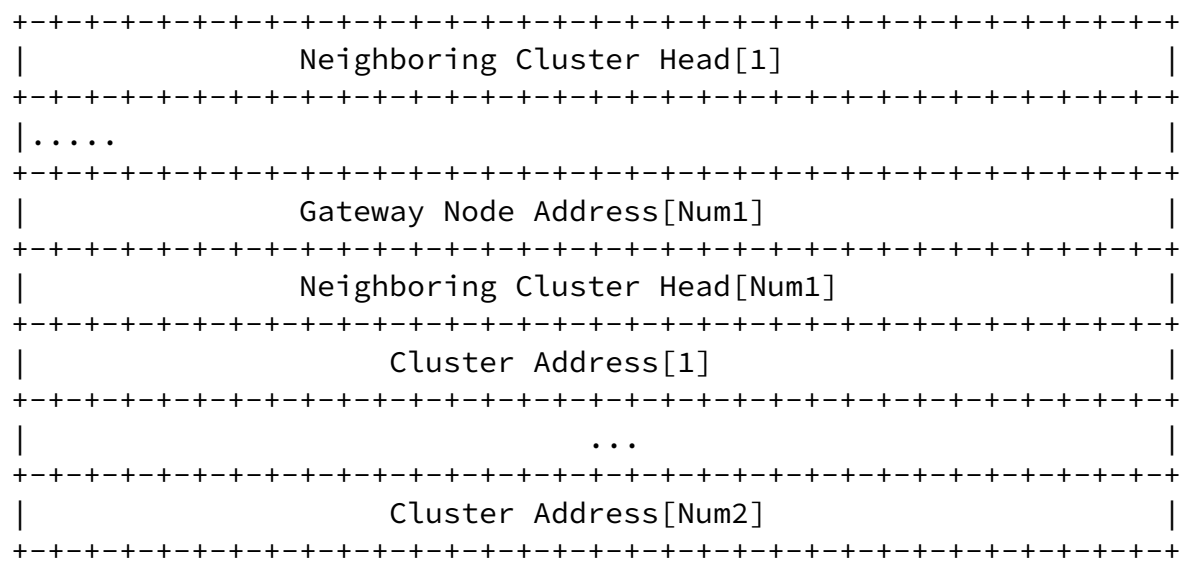
Cluster structure is exploited to minimize the flooding traffic during route discovery phase. Furthermore, certain uni-directional links are discovered and used, thus increasing the network connectivity.

6.3.1 Route Discovery

Route Discovery is the mechanism whereby a node S wishing to send a packet to a destination D obtains a source route to D. Similar to other MANET protocols [4] [1], the way S finds a route(or multiple routes) to D is also done by flooding, however, because of the clustering approach the number of times nodes are disturbed are much less in general.

Essentially, in Route Discovery, only cluster heads are flooded with Route Request Packets (RREQ) in search for a source route. The format of such a packet is shown below:





10	type of CBRP packet (Route Request)
Num1	the number of Gateway Node Address & Adjacent Cluster Address pairs
Num2	the number of Cluster Addresses
Identification	a number the RREQ originator generates to uniquely identify this RREQ sent by it.
Gateway Node Address	the gateway node who will forward this RREQ
N'ghboring ClusterHead	the corresponding cluster head the gateway node will forward this RREQ to.

To perform Route Discovery to D, the source node S sends out an RREQ, with the target node address field set to D. It fills the Neighboring Cluster Head entries with its host cluster head(s) and adjacent cluster head entries(from CAT), the corresponding Gateway Node Address is either the host cluster head itself or the adjacent cluster's gateway node. This initial RREQ is broadcast.

1. Whenever a member node M receives an RREQ,

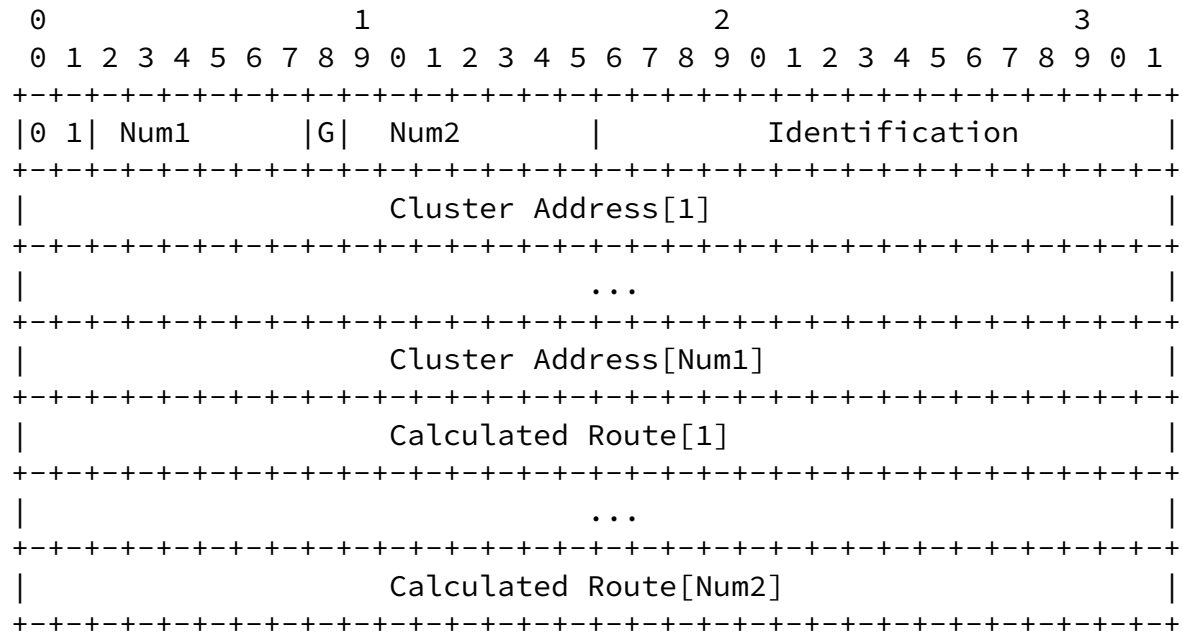
```
    if D is its neighbor, RREQ is just uni-cast to D.
    else
       if M is specified as the Gateway Node[x],
           uni-cast(relay) the RREQ to Cluster Head[x].
       else
           discard the RREQ.


  2. Whenever a cluster head C receives an RREQ,
     2.1 if it has seen this RREQ before (by looking at the identification
         field) discard it; otherwise, continue.
     2.2 records its own address in Cluster Address list.
     2.3
        if D is its neighbor or is 2-hop away
           uni-cast RREQ to D.
        else
           for each bi-directionally linked cluster head CH in C's CAT
              if CH is already in previous RREQ's
              Neighboring Cluster Head list,
                 skip
              else if CH is in Cluster Address list
                 skip
              else
                 record CH entry in Neighboring Clusterhead/Gateway Node pai
           broadcast RREQ
```

Each cluster head node forwards an RREQ packet only once and it never forwards it to a node that has already appeared in the recorded route. In CBRP, the RREQ will always follow a route with the following pattern to reach destination D:

```
        S,CH1,G1,CH2,G2,G3,CH3 ..... D
```

The recorded route in Cluster Address list will be CH1, CH2, CH3 ...

When the target of the Request, node D, receives the RREQ, D sends out an RREP packet to S as a reply, with the following format:



01	type of CBRP packet (Route Reply)
G	the flag indicating if this RREP is a gratuitous reply packet(refer to 5.3.3)
Identification	the identification number copied from the corresponding RREQ.
Num1	the number of Cluster Addresses
Num2	the number of addresses in Calculated Route
Cluster Address	copied from the corresponding RREQ. This is the sequence of cluster heads the RREP will traverse in order to reach RREQ originator. (This sequence will be shortened by forwarding cluster heads along the way and the last address will always be the next stop cluster head to forward this RREP).
Calculated Route	A sequence of addresses of the hop by hop source route calculated by the clusterhead.

D copies the list of Cluster Addresses into the RREP packet. (D may choose to memorize the reversed list of Cluster Addresses to S so that if D wants to find out a route to S, it may directly probe this route of cluster heads). D also copies the identification field in RREQ to RREP and puts its own address in Calculated Route[1].

The recorded Cluster Addresses gives the complete information about the SEQUENCE OF CLUSTERS RREP should traverse in order to reach S. Since each cluster head has knowledge of how to reach its neighboring

CHs, the RREP packet will be routed to S eventually using IP loose source routing.

While forwarding the Route Reply, intermediate cluster heads will calculate an optimized hop-by-hop route according to the information contained in the list Cluster Addresses and put it in the Calculated Route field.

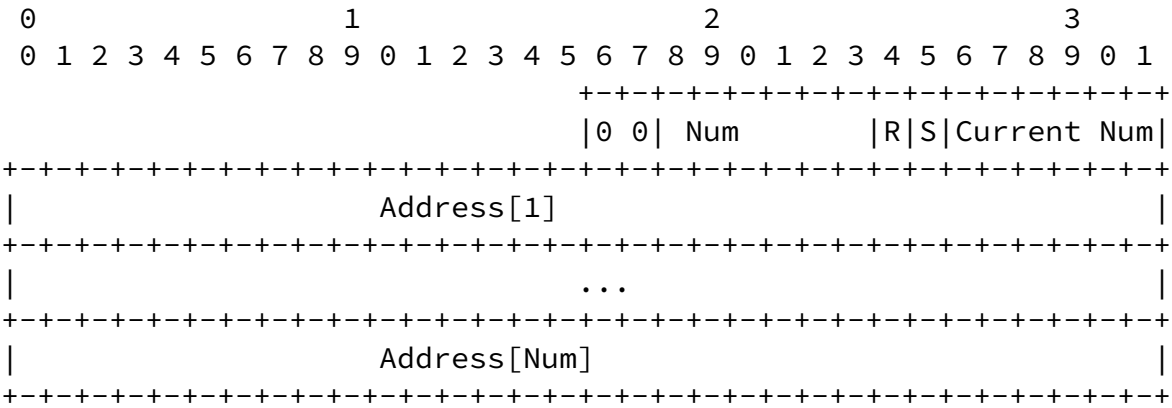
For example, 1. suppose cluster head C receives a RREP
1.1 It decrements Num1 by 1.
Cluster Address[Num1] is the neighboring cluster head
that C should forward RREP to in order to reach S.
1.2 C tries to find out a gateway node X to Cluster Address[Num1]
such that the Calculated Route[Num2] could reach X directly.
If it succeeds,
send RREP to X
else,
C increment Num2 by 1 and C records itself in Calculated
Route[Num2]. 2. suppose member node M receives a RREP
2.1 It increments Num2 by 1 and records itself in
Calculated Route[Num2].
2.2 if Cluster Address[Num1] is its Neighbor,
send RREP to it directly
else if Cluster Address[Num1] could be reached by X,
send RREP to X.

If a source does not receive any RREP after sending out RREQ for certain period of time, it goes into exponential backoff before re-sending RREQ.

As usual, all source routes learned by a node are kept in a Route Cache. When a node wishes to send a packet, it always examines its own Route Cache first before performing Route Discovery.

6.3.2 Routing

The actual routing is done using Source Routing and the format of the packet is shown below:



00	CBRP packet type (Normal data packet containing a source routing header.)
Num	the number of addresses in the source route Address[1..Num]
Current Num	specifies the currently visited address.
R flag	indicates if this route has been salvaged using local repair
S flag	indicates if this route has been shortened

* Route Shortening

Due to node movement or other reasons, a source route may become less optimal over time and should be shortened whenever possible. The route shortening mechanism shortens sub-optimal routes locally using the 2-hop-topology database information.

It works as follows:

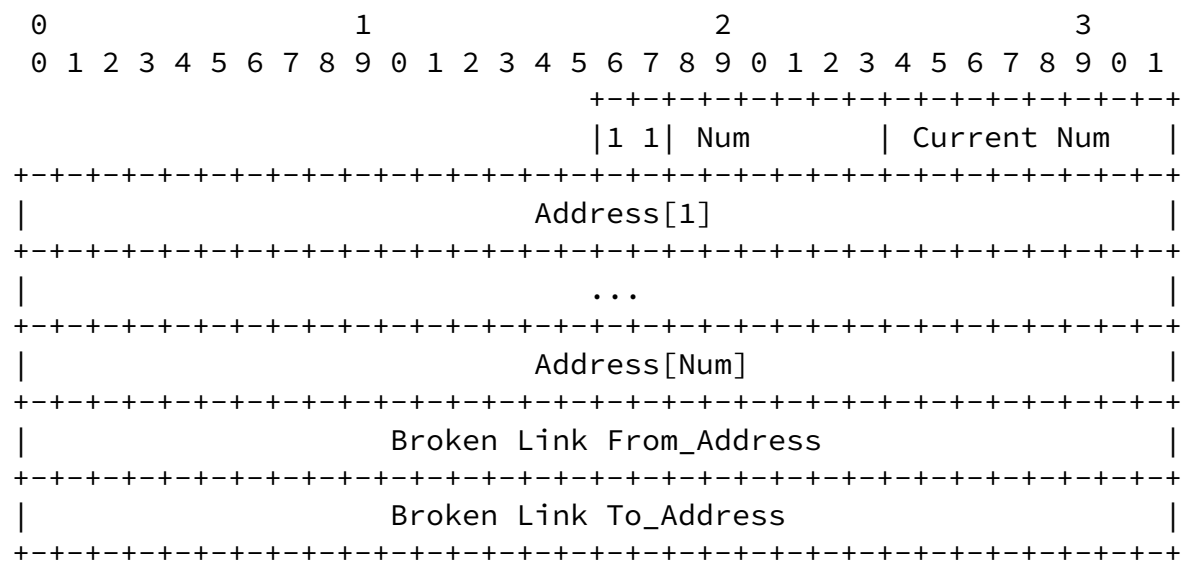
Whenever a node receives a source-routed data packet, it tries to find out the furthest node in the unvisited route that is actually its neighbor. If it succeeds, it shortens the source route accordingly and sets the S flag before forwarding the packet.

When a destination node receives a data packet with S flag set, it sends back a gratuitous RREP (setting the G flag in RREP) containing the shortened route to the packet source to inform it of the better route.

* Route Error

When a forwarding node finds out that the next hop along the source route for an unsalvaged packet is no longer reachable, it will create a Route Error (ERR) packet and send it back to the packet source to

notify it of the link failure. The format is shown below:



11	CBRP Packet type (Route Error packet).
Num	the number of addresses in the source route.
Address	the source route this ERR packet has to traverse in order to reach its destination. It is taken from the source route of the data packet.

From_Address	the ERR originator's own address.
To_Address	unreachable next hop as specified in the original source route of the data packet.

* Local Repair

After the forwarding node detects a broken route and sends out an ERR packet, it will try to salvage the data packet the best way it can using its own local information:

1. It checks if the hop after next in the source route is reachable through an intermediate node other than the one specified as the next hop by searching through its 2-hop-topology database.

2. It checks if the unreachable next hop could be reached through an intermediate node by checking its 2-hop-topology database.

3. If the packet could be saved, it modifies the source route, sets the R flag and sends out the packet to the new next hop.

Because of spatial locality, even though the next hop node moves out

of reach from the current node, it is possible that it can still be reached within 2 hops. Our simulation shows that this Local Repair mechanism works very well, saving a majority of data packets.

When a destination node receives a data packet with R flag set, it sends back a gratuitous RREP (setting the G flag in RREP) containing the repaired route to the packet source to inform it of the repaired route, avoiding unnecessary route re-discovery.

7. Discussions and Implementation Considerations

7.1 ARP Optimization

ARP is necessary as a node needs to know the MAC address of the next hop in order to send a packet. In CBRP, however, the next hop a node sends the packet to is always its neighbor node. If a node records the source MAC to IP mapping in ARP cache whenever it receives a

HELLO message, it could completely avoid ARP message exchange during routing.

7.2 Problems with Uni-directional links

This section discusses some of the problems that we have encountered during the design of CBRP. In particular, they are related to the use of uni-directional links in routing.

7.2.1 ARP problem

In a MANET context, links between 2 nodes can be bi-directional and uni-directional. However, when the link layer does MAC-layer address-based packet filtering (which most current technologies do), special care has to be taken with ARP for uni-directional links. For example, when there is a uni-directional link from node A to node B as shown in Figure 3, node A should not rely on the conventional ARP protocol to resolve node B's MAC layer address, because node B's ARP reply will never reach node A directly.

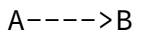


Fig. 3

CBRP is able to use uni-directional links. For an intra-cluster uni-directional link, the upstream node can be informed by its cluster head of the MAC-layer address of the downstream node. For an inter-cluster uni-directional link, as shown in Figure 2, node 3(or 5) will

know node 4(or 6)'s MAC-layer address by the process of adjacent cluster discovery.

7.2.2 802.11 Link Layer Technology

It seems hard to make good use of uni-directional links without violating the standard 7-layer OSI model. The current 802.11 MAC layer does not consider the existence of uni-directional links, nor does it support them. The sequence of RTS/CTS/Data/ACK exchange will automatically exclude the use of any uni-directional links even if one knows their existence.

One possible extension to this technology is to allow ACKs to be forwarded by neighboring cluster heads back to the sender if a uni-directional link between two clusters are being used. Such a return path is bounded by maximum 5 hops. The forwarding of ACK could be viewed as collapsing layer 2 and layer 3 routing functionality, a violation of the layering model which states that lower layer should hide details away from upper layers. Although this seems rather inefficient at first sight, if we consider the fact the bi-directional links are preferred over uni-directional links in CBRP, we would realize that such a uni-directional link will not be used unless one cannot reach a specific node using bi-directional links only.

7.3 Rate of Stale Route Discovery and Uni-directional Links

In general (not pertaining to CBRP), when uni-directional links are used, discovery of stale routes can be slow. As shown in Figure 4, when the link between 1 and 2 breaks, node 1 will not be aware until a message comes from 2 by way of 3,4,5,6. This observation justifies CBRP's use of inter-cluster uni-directional links only between 2 clusters.

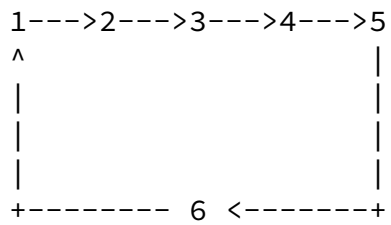


Fig. 4

8. Constants and Suggested Values

These are the CBRP constant values that we have experiemented with in the simulation.

HELLO_INTEVAL	1.5s or 2s

HELLO_LOSS	1
CONTENTION_PERIOD	1.5s

9. Applicability Statement

This section summarizes the characteristics of CBRP, as specified in the Applicability Statement draft.

9.1 Network Context

The protocol is designed for medium to large networks with relatively large average nodal degree (> 5). Nodal mobility should not be too high, i.e. node cannot move quicker than the speed of cluster formation which is defined as 1/HELLO_INTERVAL.

This protocol is suited for networks where most of the traffic is among a small set of sender-receiver pairs compared to the possibility of N*(N-1)/2 number of pairs. Also, the applications supported should be able to tolerate the delay of route discovery time.

9.2 Protocol Characteristics and Mechanisms

* Does the protocol provide support for unidirectional links? (if so, how?)

  Yes. It selectively makes use of those uni-directional links that could give two-way-route to nodes that are otherwise inaccessible using only bi-directional links.

* Does the protocol require the use of tunneling? (if so, how?)

  No.

* Does the protocol require using some form of source routing? (if so, how?)

  Yes. routes are discovered in route discovery stage and will be carried in the packet headers in actual routing. (Refer to Section 6.2) However, source routing is actually not

essential to the correct working of CBRP. As source routing poses a non-negligible overhead when the network sizes grow, we are currently designing alternatives to replace it with more efficient mechanisms.

* Does the protocol require the use of periodic messaging? (if so, how?)

  Yes. Periodically, each node in the network sends a HELLO message containing its current neighbor table. The size of the message is proportional to the degree of the node (i.e. the number of neighbors), which is around 6 to 15 for networks of average density.

* Does the protocol require the use of reliable or sequenced packet delivery? (if so, how?)

  No.

* Does the protocol provide support for routing through a multi-technology routing fabric? (if so, how?)

  Yes. Each network interface is assigned a unique IP address used for routing purpose.

* Does the protocol provide support for multiple hosts per router? (if so, how?)

  Yes. A number of hosts, each having a unique IP address, could associate itself with a router that forwards packets and participates in the routing protocol on the hosts' behalf.

* Does the protocol require link or neighbor status sensing (if so, how?)

  Yes. Neighbor status sensing is required.

* Does the protocol have dependence on a central entity? (if so, how?)

  No. All the functions are achieved in a distributed manner.

The cluster formation process differentiates the roles of

nodes as cluster heads and member nodes. Cluster heads are flooded during the route discovery phase to find a route to the destination. However the actual routing will try to bypass cluster heads as intermediate nodes.

* Does the protocol function reactively? (if so, how?)

  Yes. It defers getting the route information until such a route is explicitly asked for by the application. Routing is done on demand with 3 phases: route discovery, packet routing, route removal.

* Does the protocol function proactively? (if so, how?)

  No. But it proactively acquires its 2-hop topology information through the exchange of HELLO messages.

* Does the protocol provide loop-free routing? (if so, how?)

  Yes. Source routing records all nodes along the route that could be easily checked for loop-freedom.

* Does the protocol provide for sleep period operation? (if so, how?)

  No.

* Does the protocol provide some form of security? (if so, how?)

  Not by itself. It has to rely on other protocols (e.g. IMEP, IPsec) to give security support.

* Does the protocol provide support for utilizing multi-channel, link-layer technologies? (if so, how?)

Yes. Cluster-formation algorithms could be run on different channels to yield different sets of clusters for each channel. Routing could be done independently on each of the set of clusters.

References

[1] Perkins, C.E., "Ad-Hoc On-Demand Distance Vector Routing", MILCOM'97 panel on Ad-Hoc Networks, Monterey, CA, November 3, 1997.

[2] Perkins, C.E., "Mobile Ad Hoc Networking Terminology", draft-ietf-manet-term-00.txt, work in progress.

[3] Park V., Corson M.S, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks," Proc. IEEE INFOCOM '97, Kobe, Japan, Apr 1997.

[4] Johnson, D.B., and Maltz, D.A., "Dynamic Source Routing in Ad-Hoc Wireless Networks", Mobile Computing, T.Imielinski and H.Korth, editors, Kluwer, 1996.

[5] Toh,C.K., "A Novel Distributed Routing Protocol To Support Ad-Hoc Mobile Computing", International Phoenix Conference on Computers and Communications (IPCCC'96), pg 480-486, March 1996.

[6] Hedrick, C., "Routing Information Protocol", RFC 1058, June 1998.

[7] Das, B., Raghupathy, S, Vaduvur, B, "Routing in Ad Hoc Networks Using a Spine", Proceedings of 6th International Conference on Computer Communications and Networks, Las Vegas, USA, September, 1997.

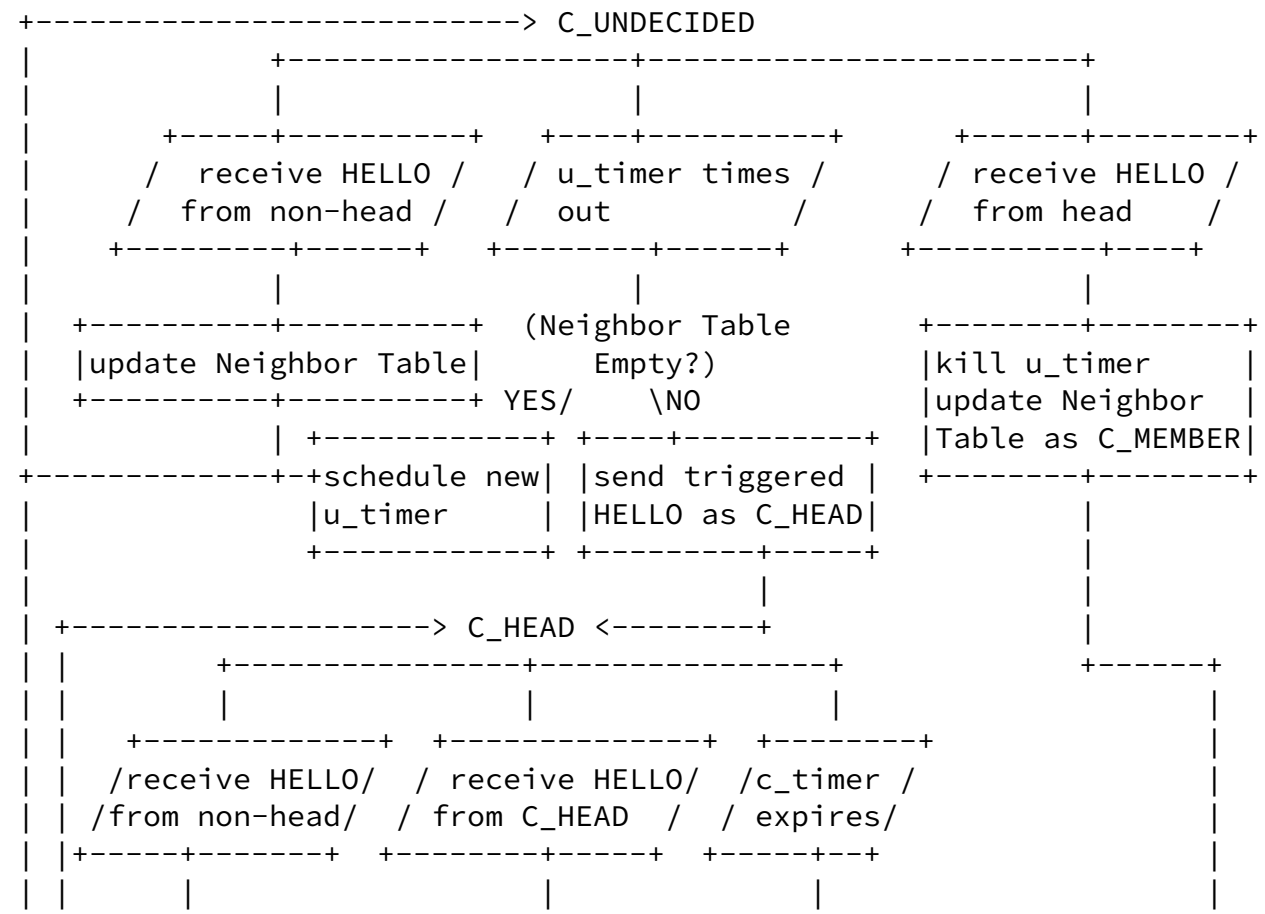
[8] Krishna, P., Vaidya, N.H., Chatterjee, M., Pradhan, D.K., " A Cluster-based Approach for Routing in Dynamic Networks", Proceedings of the Second USENIX Symposium on Mobile and Location-Independent Computing, 1995.

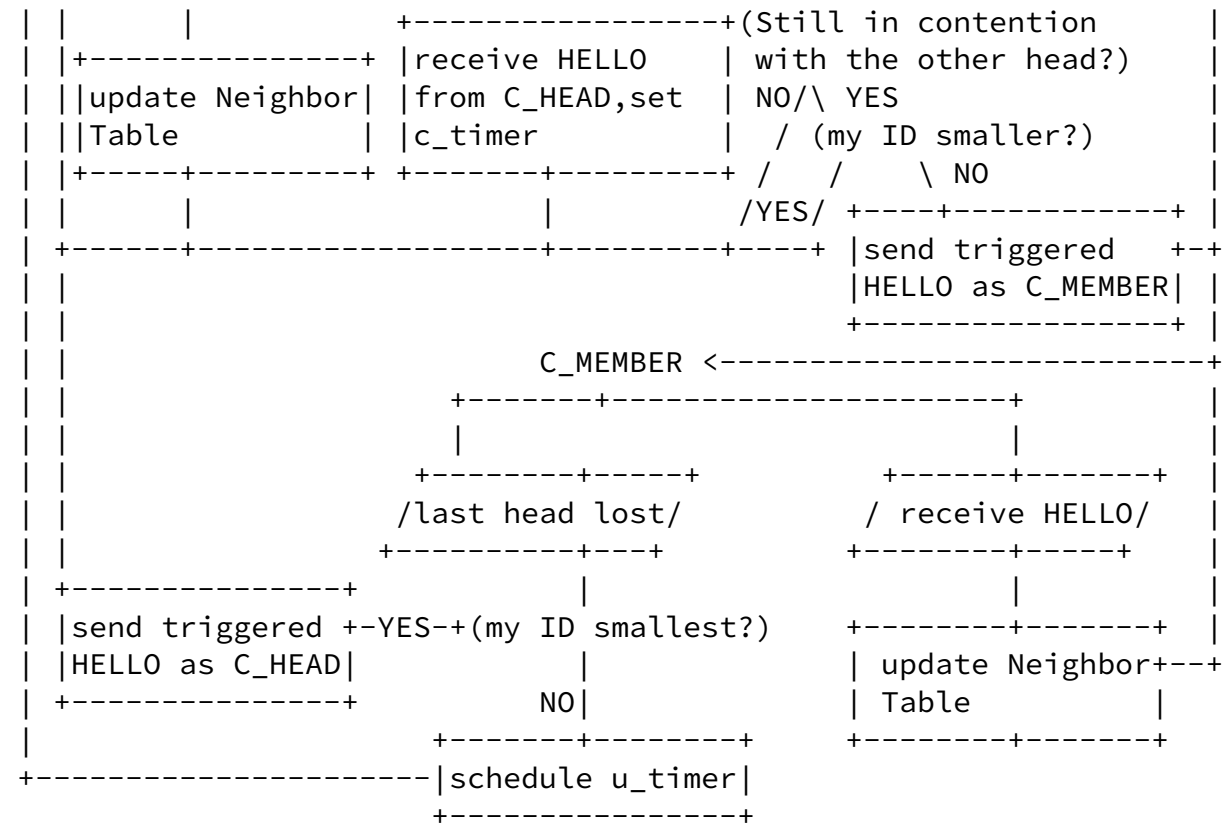
[9] Gerla, M., Tsai, T.C., "Multiculster, mobile, multimedia radio network", ACM/Balzer Journal of Wireless Networks, 1995.

[10] Chiang, C.C., Wu, H.K., Liu, W., Gerla, M., "Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel", The Next Millennium, the IEEE SICON, 1997.

[11] Ephremides A., Wieselthier J.E., Baker D.J., "A Design Concept for Reliable Mobile Radio Networks with Frequency Hopping Signaling", Proceedings of the IEEE, Vol.75, No.1, Jan 1987

Appendix A





This work was supported in part by National University of Singapore ARF grant RP960683.

Authors' Information

CBRP's URL:
Contains information on CBRP's latest development and simulation code
http://cram.comp.nus.edu.sg/cbrp/

Jiang Mingliang
Mobile Computing Lab
School of Computing
National University of Singapore
Singapore 119260
Email: jiang@eudoramail.com

Li Jinyang
Mobile Computing Lab
School of Computing
National University of Singapore
Singapore 119260
Email: lijy@acm.org

Y.C. Tay
Department of Mathematics
National University of Singapore
Singapore 119260
Email: tay@acm.org