International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 1 Issue 5, July - 2012

Disadvantage of AMRIS Protocol and its solution

P.K. SAHU
AITAM, TEKKALI, A.P, INDIA

Abstract

Multicasting reduces the communication costs for applications that send the same data to multiple recipients within a network. Instead of sending via multiple unicasts, multicasting minimizes the link bandwidth consumption, sender and router processing, and delivery delay. In addition, multicasting provides a simple yet robust communication mechanism whereby a receiver's individual address is known or changeable without any knowledge by the source. This paper concentrates on the one current multicast routing protocol. The reason for this is to handle the problem encountered when traditional wire-line multicast protocols have been modified for the wireless environment. It is found, in research, that these protocols are not suitable in terms of scalability and performance with increased mobility. These protocols do not need to keep an entry for each destination router in the routing table and maintain the information by periodic updates of the routing table. The overhead of storage and channel utilization limits the scalability of mobile ad hoc networks where each mobile node is a router. By maintaining only the active entries on an on-demand basis, one can reduce overhead, thus improving performance and scalability of network. The AMRIS protocol is employed in ad hoc networks. The protocol has its merits and demerits. In this paper, a careful and critical study of this protocol has been done. On the basis of the study, some problems with this protocol were identified and their solutions have been proposed. The AMRIS, a tree-based protocol suffers from some drawbacks such as collision of data packets with beacons. The solution proposed for the problems are based on a complex structure, which is used to store the information about the parent node and sibling node in the network.

Introduction

A mobile ad hoc network (MANET) consists of a collection of dynamic nodes with sometimes rapidly changing multihop topologies. In MANETs, there is no assumption of an underlying fixed infrastructure. Nodes are free to move around arbitrarily. Each mobile node functions as a router to establish connections between any two nodes. Since each node has a limited transaction range, not all messages may reach all the intended hosts. To provide communication through the whole network, a source to destination path could be relayed through several intermediate neighboring nodes. Unlike typical wireless routing protocols, ad hoc routing protocols must address a diverse range of issues. For instance, the network topology can change randomly and rapidly at unpredictable times. As well, since wireless links generally have lower capacity, congestion is typically the norm rather than the exception. The majority of nodes will rely on some exhaustible means for energy, thus routing protocols must limit the amount of control information that is passed between nodes. In summary, an ad hoc network routing protocol must be simple, robust, and minimizes control message exchanges. The goal of MANET is to extend mobility into the realm of autonomous, mobile, wireless domains, where a set of nodes from the network routing infrastructure in ad hoc fashion. The majority of applications for the MANET technology are in areas where rapid deployment and dynamic reconfiguration is necessary and the wire line network is not available. These include military battlefields, emergency search and rescue sites, classrooms, and conventions where participants share information dynamically using their mobile devices. Alongside the growth in wireless applications, there has been a tremendous

growth in the demand for group-oriented computing. There are more and more applications where one-to-many dissemination is necessary. The multicast service is critical in applications characterized by the close collaboration of teams with audio and video conferencing requirements and sharing of text and images. In general, wireless mobile multicasting poses several key challenges. Multicast sources move, making source oriented multicast protocols inefficient. Multicast group members move, thus precluding the use of a fixed multicast topology. Transient loops may form during tree reconfiguration. As well, tree reconfiguration schemes should be simple to keep channel overhead low. In multicasting, the key problem is to enable efficient routing of packets from a sender to multiple receivers. Now, coupled with the MANET characteristics, one can appreciate that providing a suitable multicast service within an ad hoc network becomes extremely challenging.

Ad Hoc Multicast Routing Protocol Utilizing Increasing Id Numbers

AMRIS [1, 33] is a source-initiated multicast routing protocol in which a shared-tree is constructed to support multiple sources and receivers. The main idea in this protocol is that each tree node has a session specific multicast session member identifier (MSM-ID) which indicates its logical height in the shared tree. The purpose of MSM-ID is to avoid any loop formation and repair the broken links locally.

Overview of Amris

The protocol is explained in the following sequence:

1. Terminology Used
2. Tree Initialization
3. Tree Maintenance Phase
4. Where does Sid come from?
5. Joining the Multicast Session
6. Forwarding Policy
7. Reaction to Network partition
8. Group Membership
9. Terminating the Multicast Session

1 Terminology Used

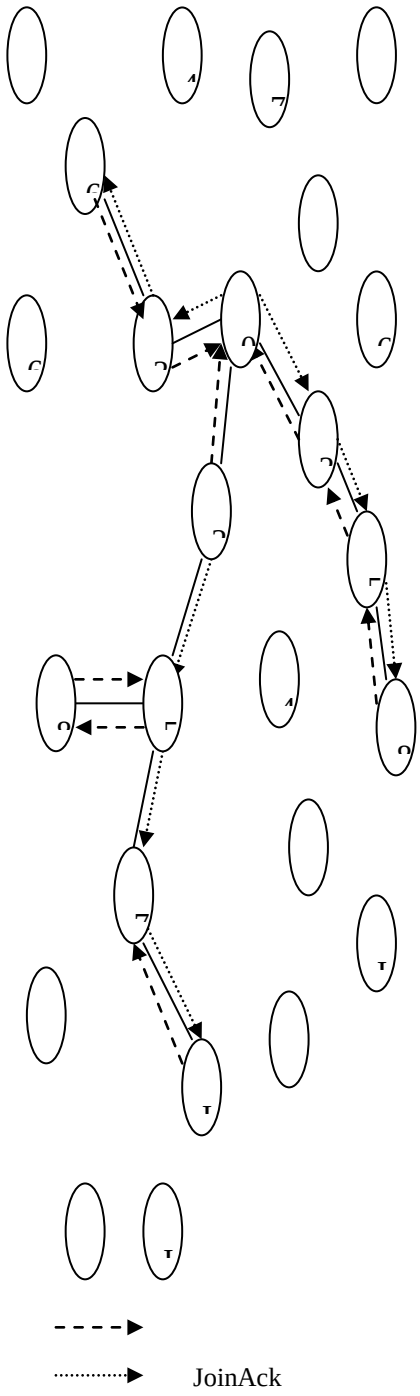
Smallest-ID node (Sid): A node is called Sid that starts/initiates a multicast session because it has the smallest msm-id compared with all other nodes for that multicast session.
Node: A node is a device in the ad-hoc network that willing to participate in the routing protocol.
Potential Parent Node (PPx): If a node X receive any New Session message from PPx and PPx has a smaller msm-id than node X, then PPx is considered as a Potential Parent Node of X.
Parent Node: A node X is the parent node of a node Y if node Y successfully joined the multicast session through X. Node X must have a smaller msm-id than node Y.
Child Node: A node Y is the child node of a node X if node Y successfully joined the multicast session through X. Node Y must have a larger msm-id than node X.
Bigger-id node: If a node X has a bigger multicast id then Y, then X is called bigger-id node.
Smaller-id node: If a node X has a smaller multicast id then Y, then X is called smaller-id node.
Multicast group: Nodes that are participating in multicast communication which includes the senders/sources, receivers and intermediate nodes are called multicast group.
Multicast Session ID (ms-id): MS-ID is a unique value identifying a particular multicast session.
Multicast Session Member ID (msm-id): MSM-ID is an ID number that all nodes must have within a multicast session. The ID differs among nodes and generally increases in numerical value from Sid.
Neighbour-Status Table: Neighbor-status table is a conceptual data structure that is employed to keep information regarding the neighbouring nodes and their status.
Multicast Beacon: This is a periodic one-hop broadcast sent by all nodes to update neighbouring nodes about its multicast state information. Multicast state information would include the node's unique ID, msm-id as well as it's parent and child msm-ids.
Interested node (I-node): A node that is interested in a specific multicast session and wishes to join is called interested node.
Uninterested Node (U-node): A node that is not interested in a specific multicast session but is "forced" and became part of the session anyway because it is required to be a relay/intermediate node on the multicast delivery path is called uninterested node.

Leaf node: Leaf node is a node at the edge of the multicast tree. A leaf node may be a sender or a receiver or both.
Intermediate node: Internal nodes are the internal branches of the multicast tree. It may be either an I-Node or an U-Node.

2. Tree Initialization Phase

To initialize the multicast tree, one of the sources in the multicast group (which is called Sid) broadcasts the new session message. The new session message contains Sid's MSM-ID, multicast session ID, and routing metrics. When all the nodes receive the new session message, they store the information derived from the new session message in their neighbor-status tables up to some fixed amount of time and generate their own MSM-ID. The new MSM-ID is larger than the received MSM-ID and it should not be consecutive in order to make use of the successive sequence numbers for quick local repair. In AMRIS [1, 33], each node maintains a neighbor-status table which stores the list of existing neighbors and their MSM-IDs. The nodes in the ad hoc network periodically transmit beacons to indicate their presence. In Fig. 1, the number inside the circle

Represents the MSM-ID of the corresponding node. Some nodes may not have a valid MSM-ID due to their absence during new session propagation, or because of the MSM-ID stored in the neighbor-status table might have been erased due to timeout. When a node wants to join the multicast group, it gets the list of potential parent nodes from its neighbor-status table and sends the join request control packet to one of its neighbor nodes. The potential parents are those neighbor nodes which have the MSM-ID less than the MSM-ID of the node which has to send the join request control packets.



In Fig. 1, node S1 sends the join request to its potential parent node I2. Since nodes I2 is not in the multicast tree, it repeats the same process by sending the join request to node I1. After a successful search of the path, node I2 sends join acknowledgement to its downstream nodes to establish the route. Other members also join the multicast group in the same way.

3. Tree Maintenance Phase

Once the tree is formed, then the process of tree maintenance comes into picture. Tree maintenance is required to repair the broken links. In AMRIS [1, 33], when a node is isolated from the tree due to link failure, it rejoins the tree by executing branch reconstruction (BR), which has two subroutines: BR1 and BR2. The subroutine BR1 is executed if the node has neighboring potential parent nodes. If the node does not have any neighboring potential parent node, then the BR2 subroutine is executed. In Fig..2, when node R1 moves from A to B, then the link I3-R1 fails. Since node R1 has neighboring potential node I4, it sends the join request packet to node I4. The node I4 is not in the tree and hence it repeats the same process to join the tree. If node I4 succeeds, then it sends join acknowledgement to node R1; otherwise, it sends JoinNack (Join not acknowledged). If node R1 receives JoinNack or times out on the reply, then it selects another neighboring potential parent node and repeats the same process. If none is available, node R1 executes the BR2 subroutine. The other case is that node R1 moves from A to C as shown in Fig..3. Since node R1 does not have any neighboring potential parent node, it executes BR2 subroutine in which it floods the join request packet with some TTL value. Hence it receives a number of join acknowledgement packets. It chooses a one of the routes by sending a Jouncing packet.

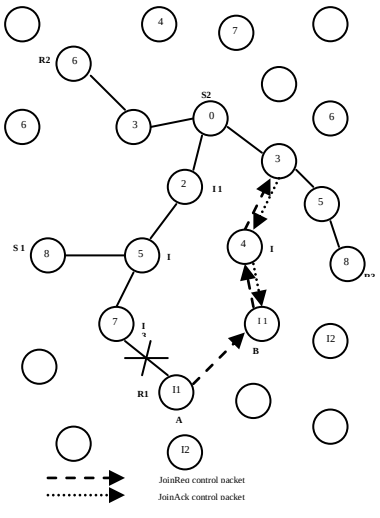


Fig. 2: Tree maintenance by BR1 subroutine

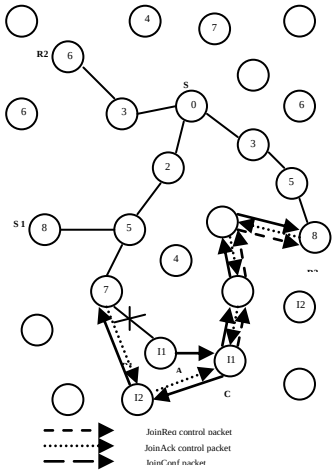


Fig. 3: Tree maintenance by BR2 subroutine.

4. About the Sid In a single-sender, multiple-receiver environment, the single-sender would most likely assume the role of Sid. In a multiple-sender, multiple-receiver environment, it is most probable that one of the multiple-senders will assume the role of Sid in initiating the multicast session. Any core-election type of algorithm can be used if there are multiple nodes contending for the role of Sid. Obviously, the choice of the core will have an initial effect on the shape of the delivery tree formed.

5. Joining the Multicast Session

When a new session message is received, a node X decides if it wishes to join the session by examining the contents of the message. The contents of the message will obviously contain information about the multicast session. The joining approach attempts to fulfil the join in an adjacent approach rather than resorting to localized broadcast right away. If the immediate neighbouring nodes are nodes already on the multicast tree, then this 1-hop approach is very fast and efficient. The presence of msm-id helps in identifying the likelihood of a successful join. A node X joins a multicast session in one of several ways depending on the situation:

1. Node X has a valid msm-id and has a neighbour that also has a valid msm-id. (This is normally the case after tree initialization.)
2. Node X does not have a valid msm-id and has a neighbour that also has a valid msm-id. (This is normally the case when node X missed the new session broadcast.)
3. Node X has a valid msm-id and does not have any neighbours with valid msm-id. (This is normally the

case when the neighbours have timed out that msm-ids. In this case node X will find other nodes that have valid msm-ids.)
4. Node X does not have a valid msm-id and does not have any neighbours with valid msm-id. (This is normally the case when node X missed the new session broadcast.)

6. Forwarding Policy

The forwarding policy rules are as follows for multicast data traffic:
1. When a multicast datagram is received, it first checks if the originator is itself, if so, it will drop the datagram.
2. If a multicast datagram is received from its registered parent node, it will forward to all registered child nodes.
3. If a multicast datagram is received from a registered child node,it will forward to all other registered child nodes and to all its registered parent node.
4. If a multicast datagram is received from any other node, it is not forwarded to any other nodes.

7. Reaction to Network partition

A network partition will cause the multicast tree to be divided into different segments. If BR does not succeed after n-tries, it can be assume that a network partition has take place. Within each partition, multicast traffic continues to be forwarded in accordance to the forwarding rules i.e.intra-partition traffic can continue as per normal but inter-partition traffic cannot take place.When this happens, the child node at the point of link breakage will become the new temporary Sid-temp, i.e. the node with the smallest msm-id within that partition. At a later time, the network may be such that a path now exists either between the Sid-temp or one of its child node back to the original partition. When a partition takes place, the Sid-temp should send a Partition-Id change message to its decendent nodes. The message can be sent standalone or piggybacked along multicast traffic. This will update all child nodes eventually. The multicast beacon will carry the new partition-id. Subsequently, any node that hears a beacon for the same multicastsession but with a different partition id will send a Found-Partition message to its own partition's Sid-temp. Only the node with the smallest msm-id will send the message to its own partition's core. The node in the other partition will not. If there is a tie, the one with largest unique id will inform its core. The core will send a join request back to its child node if after it compares the msm-ids,unique ids, of the reporting nodes and the other partitions reporting nodes. The temp core may receive several of such messages. It will choose the one with the best routing metric and send a join request message back to the node. When a join acknowlegement returns in response to the join request, the temp core may need to send out modify-msm-id messages to a subset of its child ndoes that lie on the path to the other partition. So in this situation, there are two partitions to join together. The node with the smallest msm-id will eventually become the new Sid. However there is a problem if it has two partitions whose core have the same msm-id, then neither partition will join with each other but they are still in different partitions because the original core may be destroyed.

8. Group Membership

Group membership is dynamic in nature, any node is free to join or leave the node at any time except for the Sid. If Sid wishes to leave, another node will have to be designated as the new Sid. On-tree nodes can be detected by listening to its neighbours beacons and detecting if any of the neighbours are already an on-tree node. If the node is unable to locate any neighboring on-tree nodes, it will begin a N-hop-lifetime join request broadcast to its neighbours. N begins from 2 to Max-TTL. After sending the broadcast, it will wait for a reply. If no reply is received within time T1, it will increase N by one and send again.Any node that receives the join request and is not an on-tree node will update its Neighbor-Status table and rebroadcast. The behaviour here is similar to tree initialization except that the intermediate nodes may not yet have a msm-id. Similarly the I-node also does not yet have a msm-id. Eventually, the join request will be received by an on-tree node which will reply with a join acknowledgement. The nodes along the path from the I-node to the on-tree node will then dynamically program their msm-id using their respective parent node's msm-id. A leaf node can leave the multicast session by sending a session leave message to its parent node. If the parent node is a U-node in the multicast session, it will also send a session leave message to its own parent node provided it.

9. Terminating the Multicast Session

Only Sid can close a multicast session. Nodes are of course free to leave. Sid closes the multicast session by sending a Session-End message down to its child nodes. This is rebroadcasted by intermediate nodes. All nodes that receive the Session-End message purge any entries/state associated with the multicast session. The nodes store the session-id for up to time T seconds before finally expiring that entry. If the network is partitioned, the original Sid may have closed the session but the other partitions have not since they did not receive the message. For this reason, nodes which receive the End-Session message will remember the session-id for time T seconds. If the node hears a beacon with state about the session, it will forward the End-Session message to it. Each multicast entry in the neighbour-status table has an associated timeout value. Session termination can also be done via a soft state approach, i.e. when the entries expire in the neighbour-status table.

Problems Identified in Amris

After a critical analysis of the AMRIS protocol, the following problems were observed.
(1) Wastage of bandwidth due to usage of beacons. A beacon is a very small message which is periodically sent by each node to show the connectivity with the network. Due to these beacons some amount of bandwidth is wasted.
(2) Loss of data packets due to collisions with beacons. Some times data packets use full channel bandwidth. At this stage the beacon collides with the data packets and results in loss of some data packets.
(3) The protocol follows hard state route maintenance scheme. When a parent node leaves network or it fails, its child node needs another parent node so that it could remain connected in the network. To find suitable parent node, the child node has to search a potential parent amongst the existing nodes in the network. This search process, wastes a lot of time, delays packet delivery and loss of data packets.
(4) Selection of potential parent nodes based on MSM-ID. In the existing protocol, the selection of potential parent node by a new node is based upon the MSM-ID value. This results in increase in the average hop-length between the receivers and the source, leading to increased delay and increased probability of packet losses.

ENHANCED AMRIS PROTOCOL

This section presents the solutions to the problems, which were identified The solution to these problems has been proposed through resolving the following issues.
1. What is required information and what is its storage structure.
2. How to initialize the structure.
3. How a new node will join the network.
The next subsections discuss the above issues.

1 Required information and its storage structure

In order to avoid the problem of collision of beacons with data packets, one possible solution is to provide a separate path for beacons. The nodes which are not involved in transmission of data packets can only be the part of this separate path on which beacons will be sent. The node, which want to send beacons and also involved in transmission will choose a neighbor node to establish a separate path from receiver to source. The neighboring node will go on storing the beacon information from receiving node and after a specified interval of time the neighbor node will send this beacon information to source node to inform the presence of receiving node. This way collision can be avoided and also the bandwidth of the path on which data packets are transferred can be saved. This separate path can be found out if a node knows sibling of its ancestor nodes. Also a node requires its sibling information in order to transfer it to its new children. This required information at each node of the network has been stored in hierarchical structures as shown in Fig 4. The structure L1 is used to store the information about the ancestor nodes' siblings of a node and L2 is used to store the information of sibling/children of a node (i.e. node under consideration). If a node is source node (i.e. initial node) its L1 linked-list will be empty as it does not have any parent node but it may contain the information of its immediate neighbors/siblings in the L2 depending upon current status of network. Each Node in L1 linked-list will contains a field for node ID of some parent node, a pointer to the linked-list containing children information, if they exist and a pointer to next Node containing information about the node having same status as that of the node which

is specified by the previous Node of linked list, if the node exists as shown in Fig. 4. The each Node in linked-list L2 contains two fields: a field to hold the node ID of child node and a pointer to the next Node corresponding to another child of the node.

Fegure: 4

Format of a node in linked-list L1

Node ID	Child List	Sibling Pointer

Format of a node in linked-list L2

Child Node ID	Next Child Pointer

it.

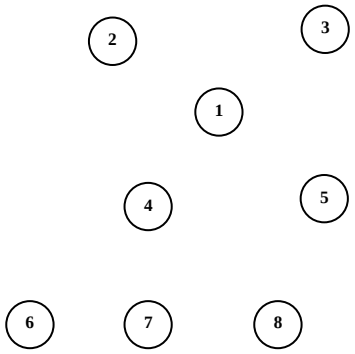


Fig. 6..: Nodes before tree creation in AMRIS

2 Initialization of the structure

Let us suppose initially nodes are as shown in Fig. 5 and once they are connected according to AMRIS protocol, the structure of network will become as shown in Fig. 6. Initially, only single node will form the network. This node is known as source node. As stated above, at each node in the network, two separate linked-list structures L1 and L2 are maintained and used. Both the linked-list structure of source node will be empty initially. Their content will change when a new nodewill

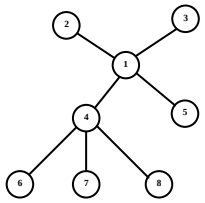


Fig. 5.: Nodes after tree creation in AMRIS

3 Joining a New Node

Whenever a new node will try to connect, it may join the network in the following manner: the new node will send its ID to the node from which it wants to connect (the potential parent), the potential parent will perform following tasks.

1. It will update its child list L2,
2. Send updated child list L2 to existing children so that they can update their sibling list.
3. Send L1 and L2 linked-list to newly joined node.

Now the newly joined node will perform the following task:

1. Based on potential parent's list L1 and L2, it will create its own L1 list.
2. Create a blank L2 list.

Fig. 7.: Arrangement of L1 Linked List

Let there is a new node that wants to join the network. The new node will send a join request to the nearest potential parent. Potential parent can be an

intermediate node or one of the last node of the network. Intermediate node is one who has a non blank child linked-list L2 and the last node is that one who has a blank child linked-list L2.
The actions at the following levels will be performed.
A. At parent node level

1. Parent node will update its child list L2.
2. Parent node will send updated list.
3. Parent will send list L1 and updated list L2 to the newly joined node.

B. At the level of already connected children of parent node.

1. Existing child will update their sibling's information.

C. At newly joined node's level

1. New node will make its list L1 and L2.

The following example shows the action taken by the various nodes in order to connect a new node. Let new node with node ID 9 wants to join the network. The following actions have to be taken.

A). Action at Parent Node's level

1. Receive request, update child list L2.

Case 1: When new node connects to node 4 (i.e. intermediate node)

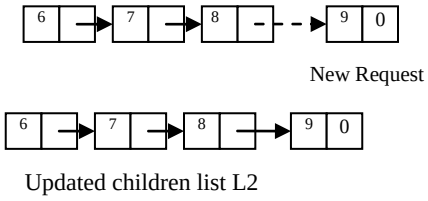


Fig. 8: Arrangement of Linked List

Case 2: When new node connects to node 6 (i.e. leaf node)

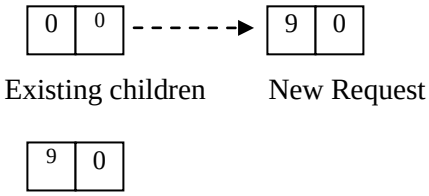


Fig. 9: Arrangement of Linked List

2. Parent node will send updated list

Case 1: When new node connects to node 4 (i.e. intermediate node)
Send updated List L2 to node 6, 7, and 8

Case 2: When new node connects to node 6 (i.e. leaf node)
This node has no existing children so it will not follow step 2.

3. Parent will send list L1 and updated list L2 to the newly joined node.

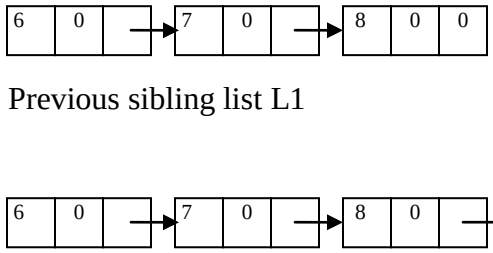
Case 1: When new node connects to node 4 (i.e. intermediate node)
Send L1 and updated L2 to node 9.

Case 2: When new node connects to node 6 (i.e. leaf node)
Send L1 and updated L2 to node 9

B) Existing Child's level

Case 1: When new node connects to node 4 (i.e. intermediate node)

Existing children 6, 7, and 8 will change its sibling



(Same for children 6, 7, and 8)

Fig. 10: Arrangement of Linked List

Case 2: When new node connects to node 6 (i.e. leaf node)
No change, because instruction A-2 is not followed.

C. Newly joined node's level

Case 1: When new node connects to node 4 (i.e. intermediate node)

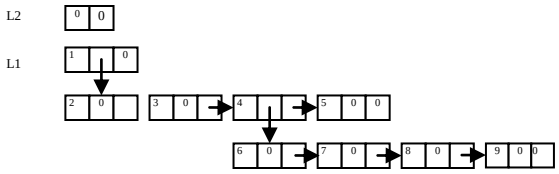


Fig. 11: Arrangement of Linked List

Case 2: When new node connects to node 6 (i.e. leaf node)

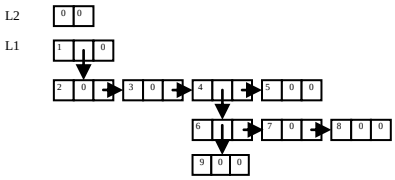


Fig. 12: Arrangement of Linked List

Working of proposed solution

In AMRIS, the main drawbacks are wastage of bandwidth due to usage of beacons and loss of many data packets due to collisions with beacons. To remove these problems, a solution has been proposed. To send a beacon in order to show the connectivity of the receiving node, the receiving node will not use the established path. For this purpose, it will use the nearest neighboring node which is not participating in the current transmission. After a certain time, the receiving node will send its beacon information to the neighboring node (called n1). The selection of the neighboring node will be based on its availability provided that its path value from receiving node is minimum. Also, this node should not be a member of established path used for current transmission form receiving node to the source node. The n1 will collect the beacon information periodically sent by receiving node. The node n1 will establish a route in order to send this beacon information to the source node. The frequency of sending beacon information is set greater than the frequency of receiving beacon from receiving node. This way the collision between beacons and data packets can be avoided. For example, in fig 7, if node 6 is receiving node and node 1 is the source node. Now if there is transmission form node 1 to node 6, a path via node 4 will be established. In order to send beacon, node 6 will select node 5 as neighboring node based upon the criteria mentioned above. The node 5 will store the last received beacon and after some pre-calculated time this beacon information will be send to the source node. By providing the separate route for beacon and data packets, we can avoid collision between beacon and data packets at the intermediate node 4. To overcome last two problems defined in section.1, we propose a solution. According to this solution, a linked-list structure will be hierarchically maintained for each node in the network, as shown in Fig 7. Whenever a node will connect to a node, it will receive linked-list structure from its parent node. In this linked-list structure, the information about all the already connected nodes to parent node and parent of parent node is stored. On the basis of this linked-list structure information, a node can select potential parent node in case the current parent node is failed or disconnected form the network. For example, in Fig 7, the node 1 is source node and contains the information of all the nodes, which are directly connected to it. The sibling nodes (2, 3, 4 and 5) which are directly connected to node 1 contain their respective linked-list structures. A linked-list structure stores the information about the parent node and its sibling nodes. For example, linked-list structure of node 2 contains 3, 4, and 5 indicating that nodes 2, 3, and 5 are sibling nodes of parent. The entry 1 indicates that node 1 is the parent of the node, as shown in Fig 7. For example if node 4 is failed, then nodes 6, 7, and 8 are disconnected from the network. If the nodes 6, 7, and 8 wants to re-connect with the network, then a suitable and nearest potential parent is required. This potential parent can be finding out very easily with the help of linked-list structure table that is contained by each node. These failure nodes can be re-connected to the network using owned linked-list table. With the help of linked-list structure, a list of potential parent nodes can be found out very easily. For example, node 6 wants to re-connect with the network. For node 6, the possible parent nodes are nodes 2, 3, and 5. Here node 6 will be connected with the network through the nearest neighboring potential parent node. Suppose the nearest neighboring potential parent node is 5 among the available potential parent nodes 2, 3, and 5. The nearest node is chosen based on the number of hop count. After deciding the nearest neighboring node, node 6 is connected to the node 5. By using the linked-list structure maintained at each node, all possible routes and a shortest route based on minimum number of nodes or number of hop count for a particular node can be find out very quickly.

Conclusion: The AMRIS Protocol is employed in Ad hoc Networks this Protocol have merits and Demerits. In this paper a careful and critical study of the

protocol has been gone on the basis of the study some problem were identified and solution have been proposed the enhanced protocol can be implemented or simulated in future, in order to evaluate them and to compare actual behavior with the existing protocol.

References:

1. C.W. Wu, Y.C. Tay and C-K. Toh, \Ad-hoc Multicast Routing protocol utilizing Increasing id-numbers (AMRIS): Functional Specication," Internet Draft, http://www.ietf.org/-internet-drafts/draft-ietf-manet-amris-spec-00.txt, November 1998.

2. A. Acharya and B.R. Badrinath, "A framework for delivering multicast messages in networks with mobile hosts", ACM / Baltzer Journal of Mobile Networks and Applications, I(II), 1996,199-219.

3. E. Bommaiah, M. Liu, A. McAuley, and R. Talpade. AMRoute: Adhoc Multicast Routing Protocol.Internet Draft, draft-talpade-manet-amroute-00.txt, Aug. 1998. Work in progress.

4. M. Gerla, et al., \On-Demand Multicast Routing Protocol," Internet Draft, http://www.ietf.org/internet-drafts/draft-ietf-manet-odmrp-00.txt, November 1998. Work in progress.

5. E.D. Kaplan (Editor). Understanding the GPS: Principles and Applications, Artech House, Boston, MA, Feb. 1996.

6. C.-C. Chiang, M. Gerla, and L. Zhang. Forwarding Group Multicast Protocol (FGMP) for Multihop, Mobile Wireless Networks. ACM/Baltzer Cluster Computing, vol. 1, no. 2, 1998.