Chapter 7

Cluster based Multicast Routing using ACO

In the previous chapters we have discussed various building blocks of the hybrid architecture proposed for routing of multimedia traffic over dense MANET environment. In this Chapter we will discuss the routing methodology to combine the benefits of both proactive and reactive components. The cluster based multicast ad hoc On-demand vector protocol optimizes the performance of the current ad hoc on-demand vector protocol with multicast(MAODV). The protocol forms clusters to achieve the goal of higher throughput with lower control overhead. The protocol has been enhanced to use the clusters elected using AEWCA as explained in Chapter 5 and have been referred as Cluster based Multicast Ad hoc On-demand Vector protocol(CMAODV) in the rest of the thesis work.

The advantages of ACO have been explored in the previous chapters. ACO based weighted clustering mechanism has shown a significant advantage in the election of CH based on QoS parameters like bandwidth availability, power, reachability of a node etc. The proposed routing approach considered in this thesis work is a hybrid multicast routing algorithm, consisting of both reactive and proactive components. The routing mechanism does not maintain routes to all possible destinations at all times (like the original ACO algorithms for wired networks), but only sets up paths when they are needed at the start of a data session. This is done in a reactive route setup phase, where the source node, sends set of forward ants(FA), in order to find multiple paths to the destination, and backward ants(BA) return to the source to set
up the paths. According to the common practice in ACO algorithms, the paths are set up in the form of pheromone tables indicating their respective quality. After the route setup, data packets are routed stochastically over the different paths following these pheromone tables. While the data transmission is going on, the paths are monitored, maintained and improved proactively using different agents, called proactive forward ants.

7.1 Algorithm Description

The proposed protocol extends the MAODV as its routing protocol. The protocol is a reactive protocol, that is, it forms the multicast trees only when necessary. The protocol distributes the control of the multicast tree among its members so that there is no single point of failure. As mentioned in the future section of the paper [Royer and Perkins 1999], the MAODV protocol might benefit from core based topology to reduce the control overhead. Thus the solution first forms the clusters throughout the network which divides the network in two-tier hierarchy. The nodes in the clusters are at lower hierarchy. The upper level consists of communication between the cluster heads of different clusters. The control packets can be restricted to the cluster itself if the node required is in the cluster. Hence, a considerable number of unnecessary forwarding of the control packet in the network is reduced. Next, the nodes are not allowed to broadcast the packets, instead the nodes send the request packet with TTL as only one hop as the clusters formed are one hop clusters. If the required node is not available in the cluster, then the permission to forward the control packet is given only to the cluster heads and the gateway nodes. So, when a node is not found in the cluster, the cluster head will forward the packet with one hop and then only the gateway will forward it again. This will reach the cluster head of next cluster and same steps are repeated till the requested node is not reached. The basic algorithmic flow of the ACO based routing adopted for the proposed architecture is presented in Figure 7.1. The procedures mentioned in the flowchart can be summarized in the sections below.
7.2 Structural Components of Proposed Algorithm

Assume that the network used in the algorithm is modeled as $G(N,E)$, where $N$ is the set of nodes and $E$ is the set of edges. The other key terms/parameters used in the algorithm are as under:

- $R^+ : \text{real positive number}$
- $(i,j) \in E : \text{link from node } i \text{ to node } j; (i,j) \in V$
- $s \in V : \text{source node of multicast group}$
- $c_{ij} \in R^+ : \text{capacity of link}$
- $d_{ij} \in R^+ : \text{delay of link } (i,j)$
- $N_p \in V - s : \text{set of destinations of multicast group}$
- $B \in R^+ : \text{traffic demand in bps}$
- $T(s,N_p) : \text{multicast tree with source 's' and set of destination } \{N_p\}$
- $P_T(s,n) \subseteq T(s,N_p) : \text{path connecting 's' and a set of destination } n \in N_p$
- $d(P_T(s,n)) : \text{delay of path } P_T(s,n) = \sum_{i,j \in P_T(s,n)} d_{ij}$
- $t_{ij} : \text{current traffic of link } (i,j); t_{ij} \in R^+$
- $z_{ij} : \text{cost per bps of link } (i,j); z_{ij} \in R^+$

The objective functions considered are:

$f_1(T) : \text{Cost of tree}$
$f_2(T):$ Maximum end to end delay

$f_3(T):$ Average delay

$f_4(T):$ Maximum link utilization

The function definitions can be presented as,

- $f_1(T) = B \sum_{(i,j) \in T} c_{ij}$
- $f_2(T) = \text{Max}\{d(P_T(s,n))\}; \; n \in N_p$
- $f_3(T) = \frac{1}{|N_p|} \sum_{n \in N_p} d(P_T(s,n))$
- $f_4(T) = \text{Max} \left\{ \frac{B+t_{ij}}{c_{ij}} \right\}; \; (i,j) \in T$

while satisfying the following constraint:

$(B + t_{ij}) \leq c_{ij} \; \forall \; (i,j) \in T(s,N_p)$

So,

$X = T(s,N_p)$ and $Y = [f_1(T) \; f_2(T) \; f_3(T) \; f_4(T)]$

Various constructs associated with the routing process covered in the following sections, uses a set of data structures for routing path creation and maintenance. They have been summarized as under:

- **Pheromone Table:** A pheromone table $T_i$ is a two-dimensional matrix. An entry $T_{ij}^d$ of this matrix contains information about the route from node $i$ to destination $d$ over neighbor $j$. This includes a regular pheromone value $\tau_{ij}^d$, a virtual pheromone value $\omega_{ij}^d$, and an average number of hops $h_{ij}^d$. The regular pheromone value $\tau_{ij}^d$ is an estimate of the goodness of the route from $i$ to $d$ over $j$. Goodness is expressed as the inverse of a cost. Regular pheromone is updated by backward ants. These can be reactive, proactive or repair backward ants. The virtual pheromone value $\omega_{ij}^d$ forms an alternative estimate of the goodness of the route from $i$ to $d$ over $j$. Differently from $\tau_{ij}^d$ it is obtained through information bootstrapping using goodness values reported by neighbor nodes during the proactive route maintenance process. The average number of hops $h_{ij}^d$ is, like the regular pheromone, updated by backward ants.

- **Neighbor Table:** The neighbor table $N_i$ kept by each node $i$, maintains the list of all the address of all one hop neighbors of the node and the information
whether a node is cluster head or cluster member. The entry $N_{ij}$ corresponding to i’s neighbor j contains a time value $th_{ij}$ indicating when did node i last hear from node j. Node i uses this time value to derive whether there is a wireless link with node j, and to detect link failures.

- **Unicast Route Table:** This table records the next hop for routes to other destinations for unicast traffic. Usually, the destination is one of the other nodes in the network. A special case is when the destination is a multicast address, which happens when the node is not a multicast tree member but has multicast data packets to send to that multicast group.

- **Multicast Route Table:** It lists the next hops for the tree structure of each multicast group. Each entry represents one group tree structure. Every node that belongs to that group tree should maintain such entries, with its own identity as group leader, group member, or router (non-multicast member that is in the tree to provide connectivity). Every next hop is associated with direction either downstream or upstream. If the next hop is one-hop nearer to the group leader, the direction is upstream; otherwise, the direction is downstream. The group leader has no upstream, while other nodes in the tree should have one and only one upstream.

- **Group Leader Table:** It records the currently-known multicast group address with its group leader address and the next hop towards that group leader when a node receives a periodic Group Hello packets (GRPH).

- **Node Status Table:** This is the table where, each node store their own status as either cluster head or cluster member or undecided. It also maintains the number of cluster heads it is member of. If the node is a cluster head than this value should be zero. The node also maintains the status of whether it is a gateway node or not based on the number of cluster head it is member of. The Information related to the QoS availability at a node. If the node is a member of more than one cluster head only then its status is set to gateway node, else it is set to not a gateway node.
7.2.1 QoS Extensions

The routing decisions in our work is based on QoS parameters like; end-to-end delay, bandwidth availability at a node, and packet loss. If \( P \) is the set of nodes visited by the forward ants \( (P \in N) \), and \( e \) is the set of edges \( (e \in E) \) corresponding to these nodes. The end-to-end delay in that case can be given as:

\[
delay (p) = \sum_{e \in p} delay (e)
\]

The end-to-end delay consists of not only the transmission delay over the wireless links but also the queuing delay in the buffer. Hence, as per the M/M/1 Queuing Model, the total time taken to reach the destination \( d \) from a source \( s \) is given by:

\[
T = d_{ij} + \frac{(q_{ij} + S_a)}{B_{ij}}
\]

where,

- \( B_{ij} \) is the bandwidth of link between two nodes \( i \) and \( j \),
- \( d_{ij} \) is the link propagation delay between two nodes \( i \) and \( j \),
- \( S_a \) is the size of ant packet,
- \( q_{ij} \) is the queue length between two nodes \( i \) and \( j \) and it can be expressed as

\[
q_{ij} = \frac{\rho_{ij}}{1 - \rho_{ij}}
\]

- \( \rho_{ij} \) which is the utilization of the link between two nodes \( i \) and \( j \) is depicted as

\[
\rho_{ij} = \frac{\lambda_{ij} \cdot \tau_{ij}}{\mu \cdot \tau_{ij}}
\]

- \( \lambda_{j} \) and \( \mu_{j} \) are the arrival rate and service rate at node \( j \),
- \( N_p = \) nodes traversed between \( s \) and \( d \).

The total end-to-end delay between the source and destination is an additive sum of all the respective delays between two subsequent intermediate nodes. Hence, it can be represented as:

\[
TotalDelay = \sum T_{ij}, \text{ where } ij \in N_p.
\]

The expressions for other metrics can be given by the following expressions.

\[
loss (p) = 1 - \prod (1 - Loss (n))
\[
bandwidth (p) = \min \{Bandwidth (e)\}
\]

\[
hopcount(p) = cost (p) = \sum_{e \in p} Cost (e)
\]

7.2.2 Reactive Path Setup

The initial invocation of the protocol is reactive in nature. The Reactive Forward Ants (RFA) is sent for this purpose, with the TTL value set to one (this indicates that
only one-hop neighbor is to be considered. The request sent by the source node can be forwarded by only the CH and CG. Hence, if the requested node is in another cluster, the path will be formed only through CH and CG. This reduces the number of control packets sent on the network.

**Reactive Forward Ants**

Ants are very small size control packets, containing the fields mentioned in Table 7.1. When a source node $s$ starts a communication session with a destination node $d$, and it does not have routing information for $d$ available, it broadcasts a RFA $F^s_d$. Due to this initial broadcasting, each neighbor of $s$ receives a replica of $F^s_d$. Each ant of the same generation finds a path connecting $s$ and $d$. The routing information of a node $i$ is represented in its pheromone table $T_i$. At each node, an ant is either unicasted or broadcasted, according to whether or not the node has routing information for $d$. If the routing information is available, the node chooses the next hop for the ant probabilistically, based on the different pheromone values associated with next hops for $d$. Or else, if the routing information is not available, the next node is selected

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source address</td>
<td>Address of the Source Node</td>
</tr>
<tr>
<td>Destination address</td>
<td>Address of the Destination Node</td>
</tr>
<tr>
<td>Packet size</td>
<td>Size of the packet</td>
</tr>
<tr>
<td>Generation number</td>
<td>This is used to identify the ants with the same source and destination address. It also limits the number of acceptable same generation ants.</td>
</tr>
<tr>
<td>Trip time</td>
<td>It is used to compare the time of same generation ants.</td>
</tr>
<tr>
<td>List of hops</td>
<td>It is used to guarantee that the paths are loop free.</td>
</tr>
<tr>
<td>Hop count</td>
<td>It is used to decide if the intermediate node is suitable to generate backward ant or it is close to destination.</td>
</tr>
<tr>
<td>Status</td>
<td>It is used to instigate the generation of backward ants.</td>
</tr>
<tr>
<td>QoS Vector Expected</td>
<td>A vector of allowable QoS values expected.</td>
</tr>
<tr>
<td>QoS Vector Available</td>
<td>A vector of allowable QoS values that is available at each node.</td>
</tr>
</tbody>
</table>
based on Roulette Wheel Selection.

The RFAs that reaches the destination within a threshold time $T_{\text{thresh}}$, is converted into a RBA, while subsequent copies received after that are destroyed. The RBA retraces the exact path that was followed by the forward ant back to the source. On its way, it collects quality information about each of the links of the path.

Reactive Backward Ants

At the destination, the RFA is converted into a RBA, which follows the list of nodes $P$ visited by the forward ant back, while traversing back to source $s$. While the RBA is sent back to the source node along the reverse route, every intermediate node and the source node updates the route to that tree member with the destination address set to the multicast group address, thus the forwarding route is established in their Unicast Route Tables. The fields of the RBA are mentioned in Table 7.2.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Final destination address</td>
<td>The original FA are sent with this address in the destination address field. It is necessary to check if the ant has arrived or not, as the IP destination address will change at every visited node.</td>
</tr>
<tr>
<td>Birth time</td>
<td>It is the time when the ant has been generated.</td>
</tr>
<tr>
<td>Arrival time</td>
<td>This is the time when the ant will reach the final destination. This value is used to calculate the trip time.</td>
</tr>
<tr>
<td>Memory stack</td>
<td>It contains the addresses of the visited nodes and the departure time towards the next-hop.</td>
</tr>
</tbody>
</table>

Pheromone Updation: At each intermediate node $i$ that an ant visits, it updates the value of the pheromone deposition by an amount of $\Delta \tau$, with respect to the previously deposited pheromone. The expressions for the same are presented as under:

$$\tau_{ij} = (1 - \rho) * \tau_{ij} + \rho * \Delta \tau$$

Here;

$Y_{\text{known}} = \text{partial solution}$

$$\Delta \tau = \frac{1}{\sum_{T \in Y_{\text{known}}} (f_1(T) + f_2(T) + f_3(T) + f_4(T))}$$

$\rho \in (0, 1]$
A range for the values of $\tau$ is defined $[\tau_{\text{min}}, \tau_{\text{max}}]$

$\tau_{\text{min}} = \frac{\Delta r}{2w(1-\rho)} \quad ; \quad w = \text{number of ants in generation}$

$\tau_{\text{max}} = \frac{\Delta r}{1-\rho}$

Following condition are checked for each new value:

If $(\tau_{ij} < \tau_{\text{min}})$ $\tau_{ij} = \tau_{\text{min}} \quad \forall i, j \in E$

If $(\tau_{ij} > \tau_{\text{max}})$ $\tau_{ij} = \tau_{\text{max}} \quad \forall i, j \in E$

As covered in the previous Chapter 5 an ACO based Weighted Cluster Head Election algorithm was used for the effective election of cluster heads based on various QoS parameters. We extended the same scheme for routing by assuming the weighted sum of the relative importance of the delay, bandwidth and loss parameters as mentioned below.

$W_D + W_L + W_B = 1$, where $W_D, W_L, W_B, W_C$ are positive weight functions of the relative importance of Delay, Loss and Bandwidth values. The weights can be varied depending upon different types of multimedia applications. For example,

- Delay sensitive traffic (voice): $W_D > W_L$ and $W_B$
- Loss sensitive traffic (ftp): $W_L > W_D$ and $W_B$
- Bandwidth sensitive traffic (voice): $W_B > W_L$ and $W_D$
- Video Conferencing: $W_D > W_L$ and $W_B$

Here, assuming that $K$ is the weighted sum, the expression for the same can be given by, $K = W_D f_D + W_L f_L + W_B f_B$, where the functions $f_D$, $f_L$ and $f_B$ are based on the penalty functions. The expressions for the penalty functions can be given as,

$f_D = \varphi_D[\text{Delay}(p(s,d)) - D_q]$

$f_L = \varphi_L[\text{Loss}(p(s,d)) - L_q]$

$f_B = \varphi_B[\text{Bandwidth}(p(s,d)) - B_q]$

Further,

If $\text{Delay}[p(s,d)] \leq D$ then $\varphi D(z) = 1$ else $\varphi D(z) = R_D$

If $\text{Loss}[p(s,d)] \leq L$ then $\varphi L(z) = 1$ else $\varphi L(z) = R_L$

If $\text{Bandwidth}[p(s,s)] \leq B$ then $\varphi B(z) = 1$ else $\varphi B(z) = R_B$

Here, the values of $[R_D, R_L, R_B, R_C] \in [0, 1]$. Also, as it can be seen from above, there is a correlation between the values of $K$ and the path quality. The Reactive FA with
higher value of $K$ will lay pheromone with the higher value.

Updating of the regular pheromone $\tau_{ij}^d$ is done based on the cost of the route from source $s$ to destination $d$.

The algorithm uses the estimates of cost calculated locally by the nodes as well as the global values. This is in order to improve reliability of the measured values. Depending on the cost metric used, the high variability of the wireless medium can cause large differences between values measured by subsequent samples. Multimedia files contain different types of frames, like I, P and B frames. The size of I-frames are bigger as compared to P and B frames. Also, they are highly important referential frames. The First Backward Ant reaching the source $s$, signals the transfer of referential frames like I and P-frames. And, then, the second best is used for the transfer of B-frames.

### 7.2.3 Multicast Tree Creation

For the multicast tree construction, same RREQ and RREP packets are used with different flags. MACT control packet is also used the packet to initialize the multicast tree. The following steps are followed, for the formation of the multicast tree.

- When a non tree member wants to join a tree first it creates an entry in a multicast route table with itself as a group member, group leader address as unknown and any upstream and downstream next hop. Then it sends a RREQ packet with join flag.

- In normal protocol, the RREQ-J packet is broadcasted in entire network, but in this protocol it is just broadcasted with one hop ttl. If the node has the information about the group leader, then it unicasts the RREQ-J packet.

- The nodes with same or larger multicast sequence number reply to the RREQ-J with RREP-J.

- When the RREP-J travels back along the reverse route, the nodes on the path insert the information about the upstream next hop towards the tree in multicast route table. If in future any RREP-J is received with better route, that is, with larger sequence number or smaller number of hop count with same sequence number, then the multicast entry is updated in the table.
• Once the source node receives the RREP-J and waits for specific time, it sends MACT with a join flag MACT-J towards the upstream node and adds next hop in multicast route table.

• All the nodes which receive the MACT-J updates the multicast table with next hop and indicates it as downstream. If the node is a tree member then a branch is formed in a tree.

• If a node is in a tree but not a group member wants to become a group member, it simply changes its identity from router to group member.

The algorithm for the multicast tree construction based on ACO is displayed as under:

```
Initialize α, β, a, b, c, B, (s, Np), tij
Set T = φ / *Tree*/
Set Ds = φ / *set of destination's reached*/
Set R = s / *list of starting nodes*/
while R ≠ φ OR Ds ≠ Np do
    Select a node i from R and generate set Ni
    if Ni = φ then
        R = R \ i; /* erase non-feasible neighbour node*/
    else
        Set probability Pij to each node of Ni
        Select node j of Ni using algorithm 1
        T = T ∪ (i, j)
        R = R ∪ j
    end
    if j ∈ Np then
        Ds = Ds ∪ j / *node j is a destination*/
    end
    Update τij
end
Print Tree T / *remove unused edges*/
return T
```

### 7.2.4 Proactive Path Maintenance

The path maintenance is done in a proactive manner, if the route between CH is set. The process updates and extends the available routing information. In particular, it allows to build a mesh of multiple routes around the initial route created during the reactive route setup process. Path maintenance mainly involves the maintenance of the multicast tree. This is done by a combination of ant path sampling and slow-rate
pheromone diffusion (meaning, the routing information obtained through the ant path sampling is spread between the nodes of the MANET and used to update the routing tables according to a bootstrapping scheme that in turn provides main guidance for the ant path exploration). Each node first checks if it is itself in the multicast tree. If the node itself is a tree member, it will follow its Multicast Route Table to forward the packets. If it is not a tree member, it will check its Unicast Route Table to find the next hop for the multicast address. If it has the information, the data packets are forwarded towards the next hop.

**Multicast Tree Maintenance**

The maintenance of the multicast tree includes tasks such as group leader selection, periodic group hello propagation, neighbor connectivity maintenance, tree merge and membership revocation.

- **Group Leader Selection** When there is a partition in the tree or when a group leader revokes its group membership, a new leader must be selected. The duty of the group leader is to periodically send the group hello packets GRPH and maintain the multicast tree. Following are the steps for selection of new group leader:

  - When there is a partition in a tree and the current node is a group member, it will become new group leader, else it will force one of the tree neighbors to be the leader.

  - If the current node only has one downstream node, it cancels its entry in multicast route table, indicating that it no longer belongs to the tree and sends a MACT with a prune flag MACT-P to a downstream node indicating it will leave the tree and tree will need new group leader.

  - If the current node has more than one downstream node, it selects one downstream, change its direction from downstream to upstream, and sends a MACT with a group-leader flag (MACT-GL) towards that node, indicating that it has other branch(es) in the tree and the tree needs a leader. If the node is a group member it becomes the group leader else it sends the packet ahead till the group member is not reached.
Once the group leader is selected, it will send GRPH-U to every node downstream unicastly so that the nodes can update the group information about new group leader.

- Periodic Group Hello Propagation Because the nodes in MANET are mobile, to maintain the multicast tree and to keep the status of the tree updated, group leader of the tree periodically sends the group hello message GRPH throughout the network. In this protocol, this broadcast is a controlled broadcast, as the GRPH is forwarded only by the cluster heads and gateway nodes. Thus stopping the flooding if GRPH messages in the network and yet all the nodes in the network receive the information. Following are the steps that are taken when a GRPH message is received.

- The node updates its group leader table with information about group leader and the route towards it. Then the node if it is a cluster head or gateway retransmits the first-time received GRPH.

- A node that is a tree member and receives GRPH from its own upstream uses it to update its current group sequence number, current group leader and the current distance from the group leader.

- If the tree member receives GRPH from other upstream and the group leader is same, then it discards the packet and waits for GRPH from its own upstream.

- If the GRPH indicates there is another group leader with same multicast group address, then this two trees can be connected and tree merge is initialized by the tree member with a group leader with smaller leader address. If its leader address is larger than in the GRPH then it discards it.

- Neighbor Connectivity Maintenance For the maintenance of the neighbor connectivity, periodic one-hop Neighbor-Hello messages are used. To reduce this one-hop Neighbor-Hello overhead, if a node already sent a broadcast message including data packets, it can delay the transmission of the Neighbor-Hello. The node realizes that the link is broken by not receiving any broadcast messages
from that neighbor in a specific time which is usually three times of Neighbor-Hello interval. Following are the steps which takes place when the node detects that there is breakage in the link.

- Once the breakage is detected by the downstream node, it deletes the information about the upstream node from the multicast route table and the sends a RREQ-J packet with an extension to find new branch. The RREQ-J packet includes the information about the hop count to the group leader.

- Only those nodes can reply to RREQ-J with RREP-J which has a larger sequence number and equal or smaller hop count to the group leader.

- If in the process the requesting node changes the information such as group leader or group sequence or hop count, it sends GRPH-U to its downstream nodes so that they can update the information.

- If the upstream node that realizes link breakage, the upstream node deletes that next hop in its multicast route table. If this upstream node is not a group member and a leaf node without any downstream, it stays for a while in the tree and after that if it still is a leaf node, it begins self-prune.

- **Member Revocation** A group member including the group leader can revoke its group membership at any time. Following are the steps when a node revokes its membership.

  - If a group leader revokes its membership, it changes its identity to router and a new group leader is selected.

  - If the node is not a group leader, then it first changes its identity to router and then if it has downstream nodes, it stays in the tree to connect a group member. The node can also completely remove itself from the tree through pruning.

  - When a node prunes itself, it removes the entry of that multicast address from its multicast route table. Then it sends MACT-P to its upstream node.
If receiving a MACT-P makes the upstream node a leaf and it is also not a group member, it can similarly prune itself from the tree with the same action. The procedure terminates when a group member or non-leaf tree member is met.

- **Tree Merge** Tree merge is detected when a tree member with a smaller group leader address receives a GRPH generated by another group leader with a larger address for the same group. Once the tree merge is detected following steps take place to merge two trees.

  - The node which has detected the tree merge sends RREQ with repair flag (RREQ-R) upstream to the group leader, for the permission to rebuild the tree.
  
  - If the group leader has not given permission to any other node, it gives the permission to the node by sending RREP-R through the same route.
  
  - The request node sends the RREQ join and repair (RREQ-JR) to the group leader with larger address. The group leader sends the RREP-JR to the request node. During the travel of RREP-JR, the group information, such as group leader address, group sequence number, and hop count to group leader, is updated.
  
  - When the tree member with smaller group leader address is reached, the node adds a next hop from which it received the RREP-JR as upstream. This node then sends the RREP-JR upstream towards the group leader with smaller address.
  
  - At each node, the nodes change the status from where they received the RREP-JR as upstream and the nodes which send the RREP-JR upstream as the status of those nodes as downstream.
  
  - When the group leader is reached, it changes the downstream nodes to upstream nodes and changes its identity from group leader to group member. And then it sends GRPH-U to its downstream to indicate change of information.

  - If the group leader itself detects the tree merge, then the RREQ-R and
RREP-R steps are omitted and it itself starts building the new tree.

Proactive Cluster Maintenance

The reactive route setup process leads to the availability of a single route from the source to its destination, indicated by regular pheromone values in the pheromone tables of the nodes. The route established is between the CH. The set of available CH changes depending upon the mobility and the density of the network. This is termed as cluster maintenance. An ACO based scheme has been adopted for this purpose. It has been covered in detail in Chapter 5. In this process GRPH (like hello) messages are broadcasted every $t_{grph}$ seconds asynchronously by all the CH of the network throughout their whole lifetime.

7.2.5 Data Packet forwarding

Data packets are forwarded from their source to their destination in hop-by-hop fashion, taking a new routing decision at each intermediate node. Routing decisions for data packets are based only on regular pheromone. This means that they only follow the reliable routes that are the result of ant based sampling, and leave the virtual pheromone information that is the result of information bootstrapping out of consideration. The combination of the reactive route setup and the proactive route maintenance processes leads to the availability of a full mesh of such reliable routes between the source and destination of each session. The probability $P_{in}d$ for an intermediate node $i$ to pick next hop $j$ when forwarding a packet with destination $d$ is given in the formula of equation as shown below.

$$P_{ijd} = \frac{[\tau_{ij}]^\alpha [D_{ijd}]^\beta [\eta_{ijd}]^\gamma [B_{ijd}]^\delta [C_{ij}]^\phi}{\sum_{i \in N_i} [\tau_{il}]^\alpha [D_{ild}]^\beta [\eta_{ild}]^\gamma [B_{ild}]^\delta [C_{ij}]^\phi}$$ (7.1)

The formulations for the calculation of the value of $\eta$ is mentioned as under.

$$\eta_{i,j} = \frac{(D - dij)^A + (L - lij)^B + bij^C}{cij}$$ (7.2)

The protocol provides a loop free mechanism for repairing the broken links due to its proactive path maintenance phase. The reactive property of the protocol reduces the number of packets that are transmitted and hence, optimizes the usage of bandwidth.
Tree based structure provided by MAODV provides advantage over a mesh architecture. However, a major drawback of the protocol is that it has large control overhead because it broadcasts the control packets throughout the network [Royer and Perkins 1999]. Due to this nature of the protocol, unnecessary flooding of control packets is done which reduces the efficiency of the protocol. In our protocol, we form clusters before starting the routing operation. Hence, the number of hops of a control packet is reduced to one and number of nodes that can forward the packets is also reduced.

### 7.2.6 Stagnation problems

Stagnation occurs when a network reaches its convergence (or equilibrium state); an optimal path is chosen by all ants and this recursively increases an ants preference for the same path. During the entire process, the ACO based algorithm may suffer from stagnation problems, if the evaporation rate is not controlled. This is undesirable for a dynamic network since, the paths are already prone to failure. It is also possible that other non-optimal paths may become optimal due to changes in network topology, and new or better paths may be discovered, if the algorithm is able to adapt to deal well with the changing dynamics of a fast changing MANET.

Various techniques to check-out the stagnation related problems of ACO based algorithms, and allow the exploration of new paths, based upon the current scenario of available paths has been proposed in the past [Li, Ma, and Cao 2005]. They are listed as under:

- Limiting the number of ants updating the pheromone values in the backward direction.
- Limiting on the number of nodes selected for pheromone deposition.
- Reducing the pheromone values of the earlier paths by a constant number, throughout

Furthermore, it has been pointed out in [Bonabeau, Dorigo, and Theraulaz 1999], that the success of ants in collectively locating the shortest path is only statistical. If by chance, many of the ants initially choose a non-optimal path, then other ants are more likely to select leading to further reinforcement of the pheromone concentration along that path. This is undesirable for static networks since, it is inefficient if ants
always choose a stagnant path that is non-optimal. The approaches to mitigate stagnation as referred from [Li, Ma, and Cao 2005] are Pheromone Control, Pheromone Heuristic Control and Pheromone Privileged Laying. The Table 7.3 below presents a comparative analysis of the stagnation control measures of ACO based algorithms. In our protocol, rather than switching to different path after congestion, we divide the total load into multiple paths as per their priority of the frame types. There is a high possibility that the two paths we choose having different initial neighboring nodes have many nodes in common. In this case, the modification would not help to solve the congestion problem in the network because ultimately both the paths are having similar nodes causing congestion. To avoid this scenario, we need to choose those paths which have less number of common nodes so that algorithm can work up to its capability. To identify which paths have less number of common nodes, we need to change the routing table to incorporate the node it traverses. And, by comparing all the paths with each other, we can group similar paths and easily pick paths those are distinct as much as possible. In case of link failure, the packets traveling through the next best path will still reach and we need not to send them again. Such division can be helpful in multimedia file transfer. For example, in case of H.264, the data is divided into 3 different data partition. Each partition has different type of data and so different priority level. The data partition having higher priority is sent through a better path.

7.3 Implementations strategy

The cluster heads election was carried out as mentioned in Chapter 5. Multicast routing was implemented over AODV and then clustering was integrated over the multicast routing protocol and changes were made in the functions of the protocol. New contents were added in the header of the packet and new table was added. Some of the files that were added for this process are; aodv_mcast.cc, aodv_mtable_aux.cc, aodv_mtable_aux.h, aodv_mtable.cc and aodv_mtable.h. Likewise, some files like were modified during the implementation. This includes: aodv.h, aodv.cc, aodv_packet.h, aodv_rqueue.cc, aodv_rqueue.h, aodv_rtable.cc, aodv_rtable.h and e.cc.
### Table 7.3: Comparison of Stagnation Control Protocols

<table>
<thead>
<tr>
<th>ACO Based Approach for stagnation control</th>
<th>Description</th>
<th>Parameters on which it depends</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pheromone Control</td>
<td>Evaporation</td>
<td>Evaporation Factor ( p ).</td>
</tr>
<tr>
<td></td>
<td>Aging and delay ants</td>
<td>Delaying ants depends on the spare capacity (the percentage of the capacity that is still available on the node).</td>
</tr>
<tr>
<td></td>
<td>Limiting and Smoothing Pheromone (ensures that only a small amount of pheromone is allowed on paths where the current pheromone concentration is closer to ( \tau_{max} )).</td>
<td>Upper bound ( \tau_{max} ).</td>
</tr>
<tr>
<td>Pheromone Heuristic Control</td>
<td>To configure ants, by the probability function so that they do not exclusively rely on sensing pheromone for their routing preferences.</td>
<td>Pheromone concentration and heuristic function (It is function of the cost of edge which may have parameters like queue length, delay and distance)</td>
</tr>
<tr>
<td>Privilege Pheromone Laying</td>
<td>First, ants evaluate the quality of their solution and then deposit the amount of pheromone based on the quality.</td>
<td>The quality of a solution is evaluated on the basis of the trip time of a forward ant and few other statistical parameters.FDC is used to compare the fitness of the solutions.</td>
</tr>
<tr>
<td>Multiple Ant Colony Optimization (MACO)</td>
<td>More than one colony of ants is used, and each deposits a different type of pheromone represented by a different color. All mobile agents consult one another to avoid routing to those links that are highly preferred by the other group.</td>
<td>Depends on number of ant colonies used.</td>
</tr>
<tr>
<td>Multi-path Routing</td>
<td>Multiple shortest optimal paths are selected with negligible overhead so it is possible to distribute the traffic over those selected paths.</td>
<td>Threshold value is selected.</td>
</tr>
</tbody>
</table>
The Collaboration graph for the implementation model of the project is displayed in the Figure 7.2

### 7.3.1 ACO based extensions

The following files are used for implementing the ACO based multicast routing using clustered architecture:

**Antnet** Class to implement Antnet Agent

**Ant_timer** Class to implement timer for interval between generation of forward ants.

**antnet_rtable** Class to implement routing table

**hdr_ant_pkt** Ant packet header

**memory** Represents memory of ant

**pheromone** Represents an entry in routing table

#### Description of Files

a. **ant_pkt.h** – defines ANT packet struct hdr_ant_pkt is created. The Packet attributes are

- **pkt_type** - Packet type [forward / backward]
- **pkt_src** - address of source node (which originated the packet)
- **pkt_dst** - address of destination node
- **pkt_len** - packet length
- **pkt_seq_num** - packet sequence number
- **pkt_start_time** - packet start time
- **pkt_memory** - [MAX_NUM_NODES] packet’s memory

b. **antnet.cc** – Implementation file for Agent Antnet

- **AntHeaderClass** tcl binding for new packet: Ant
- **AntnetClass** tcl binding for new Agent: Antnet

c. **antnet.h** – Definition file for Agent Antnet
Figure 7.2: Collaboration Graph
int command(int, const char *const *) interface for tcl commands

void recv(Packet *, Handler *) method to handle packet receive events at the Agent

void send_ant_pkt() generate forward ant

void recv_ant_pkt(Packet *) receive an ant packet

void create_backward_ant_pkt(Packet *) generate backward ant

void forward_ant_pkt(Packet *) send a forward ant to next hop as per AntNet algorithm

void backward_ant_pkt(Packet *) send a backward ant to next hop as per AntNet algorithm

void memorize(Packet *) add visited node to memory of forward ant

void update_table(Packet *) update routing table

void initialize_rtable() initialize routing table

d. antnet_common.h – Implementation file for globally available methods

e. antnet_rtable.cc – Implementation file for routing table in AntNet

struct pheromone Represents an entry in routing table

class antnet_rtable Class to implement routing table
7.4 Simulation and Results

The parameters used for simulation and the scenarios selected for the above protocol is mentioned in the Table 7.4 and Table 7.5.

7.4.1 Simulation Parameters

<table>
<thead>
<tr>
<th>Table 7.4: Parameters for nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>Link Layer Type</td>
</tr>
<tr>
<td>MAC Layer</td>
</tr>
<tr>
<td>Type of Queue</td>
</tr>
<tr>
<td>Antenna Type</td>
</tr>
<tr>
<td>Wireless Propagation Model</td>
</tr>
<tr>
<td>Type of physical interface</td>
</tr>
<tr>
<td>Type of channel</td>
</tr>
<tr>
<td>Transmission and Receiving range of antenna</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 7.5: Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>Area</td>
</tr>
<tr>
<td>Wireless Propagation Model</td>
</tr>
<tr>
<td>Type of channel</td>
</tr>
<tr>
<td>Speed of node</td>
</tr>
<tr>
<td>Type of motion</td>
</tr>
<tr>
<td>Pause Time</td>
</tr>
<tr>
<td>Number of nodes</td>
</tr>
</tbody>
</table>
7.4.2 Results

The simulation results have been plotted by varying the number of nodes, pause time and mobility. The delay suffered by AODV, MAODV and CMAODV is plotted by varying

![Figure 7.3: Delay for 25 nodes](image1)

![Figure 7.4: Delay for 50 nodes](image2)
CHAPTER 7. CLUSTER BASED MULTICAST ROUTING USING ACO

Figure 7.5: Delay for 75 nodes

Figure 7.6: Delay for 100 nodes
Figure 7.7: Throughput for 25 nodes

Figure 7.8: Throughput for 50 nodes
CHAPTER 7. CLUSTER BASED MULTICAST ROUTING USING ACO

Figure 7.9: Throughput for 75 nodes

Figure 7.10: Throughput for 100 nodes
Figure 7.11: Delay for speed 5 m/s

Figure 7.12: Delay for speed 10 m/s
Figure 7.13: Delay for speed 15 m/s

Figure 7.14: Delay for 20 nodes
Figure 7.15: Throughput for speed 5 m/s

Figure 7.16: Throughput for speed 10 m/s
Figure 7.17: Throughput for speed 15 m/s

Figure 7.18: Throughput for speed 20 m/s
Figure 7.19: Delay for 0s pause

Figure 7.20: Delay for 30s pause
CHAPTER 7. CLUSTER BASED MULTICAST ROUTING USING ACO

Figure 7.21: Delay for 60s pause

Figure 7.22: Delay for 120s pause
Figure 7.23: Delay for 300s pause

Figure 7.24: Delay for 600s pause
Figure 7.25: Throughput for 0s pause

Figure 7.26: Throughput for 30s pause
Figure 7.27: Throughput for 60s pause

Figure 7.28: Throughput for 120s pause
Figure 7.29: Throughput for 300s pause

Figure 7.30: Throughput for 600s pause
CHAPTER 7. CLUSTER BASED MULTICAST ROUTING USING ACO

Figure 7.31: Total Control Packets for 25 nodes

Figure 7.32: Total Control Packets for 50 nodes
Figure 7.33: Total Control Packets for 75 nodes

Figure 7.34: Total Control Packets for 100 nodes
7.4.3 Result Analysis

- From Figure 7.7, 7.8, 7.9 and 7.10 it can be concluded that as the number of nodes increase, i.e. as the density in the network increase, CMAODV gives better throughput as compared to AODV and MAODV.

- From Figure 7.15, 7.16, 7.17 and 7.18 it can be observed that as the speed of nodes increase, the performance of CMAODV degrades with respect to throughput in comparison to MAODV and AODV.

- From Figure 7.25, 7.26, 7.27, 7.28, 7.29 and 7.30 it is observed that as the pause time of the nodes is increased, that is, the mobility of the nodes is reduced; CMAODV gives better throughput than MAODV and AODV.

- From Figure 7.3, 7.4, 7.5 and 7.6 it is observed that at lower number of nodes, CMAODV has lowest delay but also throughput is less, while as the number of nodes increase, the delay of CMAODV is more than that of MAODV but less than AODV.

- From Figure 7.19, 7.20, 7.21, 7.22, 7.23 and 7.24 it can be concluded that with increasing pause time, the delay of CMAODV increases and is more than that of MAODV but is less than AODV in majority of cases. Only in the cases where there was minimum mobility (refer Figure 7.24), delay of CMAODV was more than AODV.

- From Figure 7.11, 7.12, 7.13 and 7.14 it has been observed that with increase in the speed, the delay of CMAODV becomes more than that of MAODV but is still less than AODV.

- From Figure 7.31 it can be observed that the control packets that are generated by CMAODV is higher than that of both MAODV and AODV.

- While Figure 7.32 shows that for 50 nodes, the control overhead for CMAODV is less than that of MAODV but more than AODV.

- Figure 7.33 suggests that for 75 nodes, at lower mobility, the control overhead of CMAODV is higher than that of both MAODV and AODV but at higher mobility, the control overhead is lower than both MAODV and AODV.
• For 100 nodes, Figure 7.34 CMAODV has lowest control overhead for all the scenarios.

• Thus, from Figures 7.31, 7.32, 7.33 and 7.34 it can be concluded that for high density of nodes, control overhead of CMAODV is much less than that of MAODV and AODV.

7.5 Analysis based on various Video types

As mentioned in Chapter 6 related to DiffServ based scheduling of multimedia traffic, we have considered four different benchmark videos (Claire-news reader, Silent-Deaf news reader, Coastguard and Foreman) with different distribution of frames and number of objects. These videos are used as input traffic in our implementation model and tested using the Evalvid architectures mentioned previously. All the subsequent comparisons assume two types of models; namely, Best-effort network and our implementation model termed as DiffServ network in all the subsequent results/graphs/plots. The comparisons have been done with respect to the PSNR value representations, as mentioned in Chapter 6.

In the Best-effort network, packets are dropped from queue tail(drop-tail) without considering their importance/priority, when the queue size reaches the queue limit. The packets are dropped, thereafter.

The Claire-news Reader and the Silent-Deaf news reader video have low motion and fewer number of objects in the frame. Hence, the PSNR value is relatively the same with Best-effort network and our model. This is shown in Figure 7.35 and Figure 7.36 respectively.

The Coastguard and Foreman videos have relatively higher changes in the bitrate because of high motion and more number of objects in the frame. With DiffServ architecture, we can reduce the loss of high priority packets; whereas in Best-effort delivery, the packets start dropping without considering their priorities. The PSNR value comparison for Coastguard and Foreman videos are shown in Figure 7.37 and 7.38 respectively.
The Figure 7.39 shows comparison of PSNR values for all the four benchmark videos. PSNR values are good for Claire news-reader and Silent-Deaf news-reader video respectively. However, the PSNR values is below 20dB from frame number 50 to frame number 200 for Coastguard video when the two objects (ship and boat) cross each other. Also, the PSNR values are below 15dB from frame number 270 to frame number 360 for Foreman video because of more number of objects in the frame.

These videos when tested against our network model showed considerable improvements. Visual representation of the Coastguard and the Foreman video with Best Effort network model is displayed in Figure 7.40 and Figure 7.42. The improvement in the quality of these videos when transmitted through our DiffServ based implementation model is displayed in Figure 7.41 and Figure 7.43.
Figure 7.36: Silent-Deaf news reader video-PSNR values for Best Effort and DiffServ

Figure 7.37: Coastguard video-PSNR values for Best Effort and DiffServ
Figure 7.38: Foreman video-PSNR values for Best Effort and DiffServ

Figure 7.39: PSNR value for the four video types
Figure 7.40: Coastguard video with Best Effort

Figure 7.41: Coastguard video with DiffServ

Figure 7.42: Foreman video with Best Effort

Figure 7.43: Foreman video with DiffServ
When the video is delivered over DiffServ network; I, P and B-frame packets are classified based on the DSCP. I-frame packets are pre-marked with DSCP 10(AF10), P-frame packets are pre-marked with DSCP 11(AF10+1) and B-frame are pre-marked with DSCP 12(AF12). When queue exceeds a given threshold value, the WRED starts to drop the packets by a specified drop probability. Each frame type is specified with a WRED parameter specified as minimum threshold, a maximum threshold, and a maximum drop probability, i.e. \( \text{min}_{th} \), \( \text{max}_{th} \), and \( \text{max}_p \). In this thesis, the WRED parameter are specified as \{4, 8, 0.025\} for I-frame packets, \{2, 4, 0.05\} for P-frame packets and \{1, 2, 0.1\} for B-frame packets respectively. The Table 7.6 represents a reduction in the average delay and jitter of Foremen video for Best effort and DiffServ networks. Likewise, the Table 7.7 also shows that we can reduce average delay and average jitter with our DiffServ network model.

### Table 7.6: QoS parameters with DiffServ and Best-Effort for Foreman sequence

<table>
<thead>
<tr>
<th>Foreman video</th>
<th>Delay (seconds)</th>
<th>Jitter (seconds)</th>
<th>PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best-Effort</td>
<td>0.195504</td>
<td>0.212496</td>
<td>25.818733</td>
</tr>
<tr>
<td>DiffServ network</td>
<td>0.136452</td>
<td>0.158187</td>
<td>28.7868</td>
</tr>
</tbody>
</table>

### Table 7.7: QoS parameter with DiffServ and Best-Effort for Coastguard sequence

<table>
<thead>
<tr>
<th>Coastguard video</th>
<th>Delay (seconds)</th>
<th>Jitter (seconds)</th>
<th>PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best-Effort</td>
<td>0.202501</td>
<td>0.22223</td>
<td>22.67663</td>
</tr>
<tr>
<td>DiffServ network</td>
<td>0.135268</td>
<td>0.175043</td>
<td>24.3966</td>
</tr>
</tbody>
</table>

### 7.6 Summary

The proposed protocol, Core Based Multipath Multicast Routing Protocol (CBMMRP) presents a solution for multicast routing over MANET ensuring the QoS. It uses the cores (cluster-heads) elected using the AEWCA approach covered in Chapter 5. The protocol chooses IntServ or DiffServ according to the service, which means resources will be reserved typically when larger losses in the subsequent frames will be
recorded. The multimedia frames are scheduled based on the frame type, which gets marked based on application type. The protocol reduces the control overhead and is more scalable as it uses core based topology. The topology is better managed by introducing the nodes with lesser mobility and better resources as the cluster heads. From the current level of implementation it is observed that the clustering in multicast increases the throughput, even when the density of nodes increase in the network and also provides delay that is lesser than the unicast routing protocol. However, the delay is slightly higher than that of multicast routing protocol without clustering for sparse MANET.

It has also been observed that the impact of losses in multimedia frames in fast moving video is higher than that of video with less motion. However, our scheme of DiffServ based scheduling with WRED queue management technique, of the frames ensures that, even if the multimedia data is bursty, the perceptual quality of the video does not deteriorate much. QoS parameter like average delay, average jitter and drop of high priority packets are reduced with DiffServ architecture compare to best-effort delivery. Further, based on analysis, we observe that user perceive bad quality of video when PSNR values are continuously low because of consecutive packets drop or corrupted. Hence, with consecutive drop of frames, we retransmitted the packets to improve the quality of multimedia.