CHAPTER 4

P2P RESOURCE DISCOVERY PROTOCOL

Mobile peer-to-peer systems are systems where mobile devices can collaborate together and share the content with the help of peer to peer applications, without the intervention of a central server. These systems can either connect together spontaneously in an ad-hoc fashion, or use the telecom operator’s mobile networks to connect to peer-to-peer systems on which they may collaborate with fixed peers. In the present research work wireless ad hoc networks are preferred more than the infrastructure based networks like cellular networks since, peer to peer networks and MANET do share some similarities, such as decentralization, self-configuration and self-healing; combining both networks could provide a solution for various purposes such as data storage, data sharing, information retrieval and data dissemination. However, this combination poses great challenges as these networks operate on different layers (peer to peer network on application layer and MANET at the network layer). Hence, more attention is needed for routing in the overlay network and maintaining the overlay network with adaptation to the underlying network. Therefore, the simple deployment of common P2P protocols on top of MANET is inefficient and does not scale well. The main reason for these limitations is due to the fact that the virtual overlay network does not match the frequently changing physical network (MANET). In the present research, a mechanism to overcome this mismatch issue has been proposed has been.
Castro et al (2010) explain the numerous peer-to-peer overlay networks proposed with very different architectures and protocols to handle the issues raised by MANET. Despite the similarities between P2P networks and MANET, it is unclear that simply adopting existing P2P overlay techniques in MANET is desirable, since there are also differences. P2P networks tend to be very large–scale with millions of simultaneous users, and are designed as overlays for deployment on the “edge” of the Internet, where the nodes generally go on and off. On the other hand, MANET tend to have far fewer nodes, the devices are severely resource–constrained in comparison, and the links between the nodes usually have higher delay.

The majority of peer–to-peer computing research has focused on the issues related to efficient lookup and routing (Charlie-Hu et al 2005). This might lead to flooding the query throughout the network. The location of the content must be discovered quickly and efficiently when a search has been initiated. This requires a specific topology to be enforced to channelize the search to reduce the delay. Given that gathering of relevant content exhibit clustering in most real-world situations, tuning the overlay topology to reflect the clustering can improve the system performance (Lehtinen 2006). The resources (or services) are totally distributed to peers and there is usually no relation between the location of resources and the network topology. Searching the whole network by flooding would result in unnecessary zigzag routes and congestion in the network.

It is important to consider how to run effectively a P2P overlay in an infrastructure less network such as MANET. As nodes mobility might lead to topology changes in the MANET routing layer, there might be potential for misrouted messages if the overlay routing and the MANET routing have inconsistent topology information. Searching the whole network by flooding would result in unnecessary zigzag routes and congestion in the network. In
mobile peer to peer networks, it is better if the overlay topology reflects the physical layout of the underlying nodes, otherwise a single overlay hop may equate to the diameter of the network. In turn it would increase traffic overhead. The present research avoids flooding and ensures that the change taking place in the physical MANET is reflected as early as possible in the overlay network. Figure 4.1 depicts the manner the hierarchal overlay has been formed with the underlying network.

![Figure 4.1 Overlays with underlay network](image)

The remainder of this chapter is organized as follows. Sections 4.1 and 4.2 present the overlay formation and neighbor peer discovery process in details respectively. Section 4.3 details the management of overlay with neighbor connectivity updating. The cross layer interaction with two layers has been explained in detail in section 4.4. Section 4.5 describes the multicast group formation and their relevant activities. Section 4.6 summarizes the chapter.
4.1 P2P OVERLAY MANAGEMENT

The major challenges dealt in the present research are latency and overhead. The challenges are faced by incorporating features like proximity based overlay formation, content based group routing and multicast routing mechanism in the overlay network. Moreover, these mechanisms are implemented in the system in such a way that all these mechanisms are able to co-ordinate among themselves without the need for a centralized coordinator.

In order to reduce the overhead the overlay network should be efficient and avoid sending too many discovery or network maintenance messages in the network. To reduce the overhead and to improve the efficiency of the search method, a selective search mechanism has been proposed for easy accessing. Content aware cluster is formed by grouping peers in the application layer when constructing the overlay network. After the construction of the overlay network multicast routing algorithm is applied for distributing the resources in a timely manner. The proposed mobile peer-to-peer protocol employs efficient resource based search and cross layer communication to decrease the overhead as far as possible and to match the virtual P2P topology with the physical topology of the MANET. For the formation of the multicast groups and the retrieval of resources a protocol by name Mobile Peer-to-Peer Resource Sharing Protocol (MPRSP) has been proposed. Since a cross layer approach is used, Link State Dynamic Source Routing (LSDSR) and MPRSP interact with each other to make the formation of overlay network, retrieval of resources and multicast routing more effectively.

For instance, a peer randomly choosing logical neighbours without any knowledge about the underlying physical topology causes topology mismatch between the P2P logical overlay network and physical underlying network. Figure 4.2 gives an example.
Figure 4.2 Example of topology mismatch problem

For a query message sent along the overlay path A to C and A to B, node B is visited twice. Even though B is a peer, B is also visited as a mobile node. Owing to the mismatch problem, the same message may traverse links, BE, EF and FC, many times, causing unnecessary traffic, and increasing the P2P users’ query search latency as well. The primary problem with the P2P networks is inefficient mismatching between overlay topology and the underlying network results in unnecessary blind flooding (Qiu et al 2007) which has certain undesirable consequences which leads to problems like performance degrading, and deadlock.

The new node has to make random attempts to connect to the existing node. To implement this in a P2P network, the new node has to have information about the global topology, which might be very hard to maintain in reality. The topology mismatch between the P2P logical overlay network and physical underlying network is common in almost all the P2P networks. The message may traverse the same physical link multiple times causing traffic eventually resulting in poor performance. In this case, a query is flooded to multiple paths, but merges to the same peer. Thereby, two neighboring peers may forward the same query message to each other and the
same query message may traverse the same logical link twice. This results in overall P2P traffic and congestion. Hsiao et al (2010) shows unstructured peer-to-peer (P2P) network like Gnutella the participating peers choose their neighbours randomly and hence the resultant P2P network mismatches its underlying physical network. This increases the communication time between the peers and thereby increases the traffic. Hence, the present research focus is on improving search performance with better topology organization.

In unstructured P2P systems, the occurrence of topology mismatch between the P2P logical overlay network and the physical layer causes traffic issues and the main reason for the same is the random joining and leaving of a peer in the P2P network. There have been studies and experiments on the issue of nullifying the topology mismatch in peer-to-peer (P2P) systems and there by significantly improving the response time and reduce traffic (Qiu et al 2007 and Srivatsa et al 2006). Hence, a novel topology matching algorithm based on the Vivaldi method has been proposed. The proposal considers real time analytical readings and has an optimal design. Specifically, the proposal constructs an unstructured P2P network in which a message, originated by any node r, reaches any other node t by taking approximately only physical end-to-end delay between r and t. Solutions have been proposed (Hung-Chang et al 2010) to enhance the routing efficiency of the existing decentralized overlay networks. However these works have not solved all the problems associated with the formation of overlay and the mismatch between the overlay network and the underlying physical network. The present research deals with solving the problem with a topology construction algorithm based on cross layer communication which is explained in section 4.4. Another work that has been done is the construction of the neighborhood using computed Round Trip Time (RTT).
The overlay network proposals of the present research employ two basic mechanisms:

(1) An algorithm for forming neighbours using delay among members.
(2) An algorithm for managing the topology of the overlay network with periodic updates.

Although several approaches have been proposed, performance in terms of delay and number of identical copies of packets for large groups remains a significant concern. These problems are proposed to tackle using a unique delay-aware approach. The approach used in the present research exploits the underlying network topology information for constructing efficient overlay networks, assuming that the underlying network is able to provide the required information. The phrase “underlying network topology,” refers the wireless mobile network.

Topology management involves constructing the overlay topology in a self-organizing way. It affects the overlay topology of the network by defining principles for nodes choosing their neighbours and thus making the network more efficient for the given purpose. Topology management includes two processes that together determine the topology. First is the process of finding new nodes for establishing connection in the network. The second process includes methods to define when and how to make changes, and add or remove connections. The purpose of the topology management is to maintain the neighbourhood of the node so that neighbours are the best nodes available to a specific node according to some defined criteria.
4.2 NEIGHBORHOOD DISCOVERY PROCESS

In this section, the method of finding the neighbor peers to form the overlay topology of the proposed system has been described. Since, there is no global knowledge of the network; nodes make decisions regarding their neighbors based on local knowledge only. Nodes to which a node is connected are called neighbors. A new node i joining the system start to build overlay connection by selecting its neighbors among the existing peers. Bootstrapping is a mechanism through which a newly joined node learns about the existing peers in the P2P network. Karbhari et al (2004) applied a centralized bootstrapping mechanism used by various Gnutella clients. When a node is initialized, it contacts rendezvous peer to get the bootstrap nodes for populating its local host-cache. In P2P, a rendezvous peer provides peers with a network location to use to discover other peers and their resources. Peers issue discovery queries to a rendezvous peer, and the rendezvous provides information on the peers it is aware of on the network. Nodes also frequently update their host-cache by exchanging ping and pong messages with their neighbors as described in (Chawathe et al 2003). Entries in cache are refreshed by nodes making the bootstrap requests. The cache size has been limited to 1000 entries in the simulations and random cache eviction has been performed when it is full.

In a typical overlay network, a node must select a fixed number of say k immediate overlay neighbours for routing traffic. Peers decide on taking nodes as their neighbours based on the knowledge about some property of the node. Peers either collect this information locally through the transaction with the nodes or from periodic messages from their neighbors. One popular criterion is that the node’s neighbors should be the nodes which are the closest nodes in the physical network. The purpose is to match logical
topology to the underlying physical topology and thus prevent a situation where the message goes through the same path several times in the physical level, although it is handled at most once by the node in the application level. Thus, the purpose is to decrease the amount of traffic in the physical network and to decrease the delay. However, in the logical topology, this solution may increase the amount of hops needed to find a searched resource. If the used time-to-live values of the flooded messages need to be increased to find a certain amount of resources, it increases the traffic both on the logical and physical levels.

In the present research the delay-aware peer selection algorithm has been applied for the overlay formation that selects one peer from the neighbourhood. There are many network properties and characteristics that can be exploited to improve the system behaviour, however, most of them are very difficult to measure (e.g., available bandwidth between two peers). The RTT instead is very simple to measure accurately and is normally a very good indicator of how far nodes are and how much congested the network is between them. Thus, trying to communicate with near-by (in terms of RTT) peers should improve performance and have a lighter impact on the network. The main difference between the random and the delay oriented peer selection is that the second one picks peers with lower RTT with higher probability for neighbour selection to enhance data exchanges through the closest neighbours. The process is depicted in Figure 4.3.

In order to maintain the efficiency of the network, RTT information, delivered usually by the node, is used as the criteria for selecting the neighbours. The purpose is to reduce the latency in the network. This information helps nodes to avoid long distance neighbours and decreases the processing delay in the network (Lv, et al 2002), (Chawathe et al 2003). The
other criterion is the similarity of the content. When the sender peer is defined as a node providing a reply for the resource requests, the purpose is to place nodes with a similar content peers close to each other in the logical network. This clustering may be established based on the information gathered periodically from peers through the cross layer architecture which may involve semantic metadata describing resources. For the first criteria, the idea is that nodes that have provided resources to the node can be reached with minimum delay. In the second criteria selection, there have to be some global and predefined rules for describing, classifying, and matching resources and measuring similarities. This is difficult in the system with a distributed nature, but when a node has neighbors with similar interests, it receives required resources closer and thus the value of time-to-live in the queries may be decreased. This decreases traffic in the network.

If the peer is the first in the overlay, it is bootstrapped as the sole overlay peer. Peer which joins after, go through the neighbour discovery process. When a peer \( p_i \) joins an overlay network, it must connect to at least one overlay peer member, and later with numerous ones. The delay calculation is applied to all known peers in the overlay, and the list is sorted. Peer \( p_i \) contacts neighbours in the ascending order of their delay. Let \( p_j \) be the peer whose delay is minimum. However if \( p_j \) state is in full state then \( p_j \) rejects \( p_i \)'s request. Peers are associated with their state position. The first peer \( p_j \) that \( p_i \) contacts will agree to the connection request, provided \( p_j \) state is connected. This is necessary because \( p_i \) will not be in any peer’s list as it is joining newly. Peer \( p_i \) makes the connection request one by one because this provides an accurate picture of connectivity to the potential neighbours. Once it has the state full, \( p_i \) has successfully bootstrapped itself into the overlay if it
has at least one neighbour. At this point the second part of the heuristic algorithm, topology maintenance, begins for \( p_i \).
\( T = \) Time

\( RTT = \) Round Trip Time

\( NT = \) Neighbour Table

**Input:** \( P, i \)'s list of all possible neighbour nodes

**Output:** \( I, i \)'s list of eligible neighbour nodes

while \( t < T \) do

  if (conn_status(\( p_i \)) not equal “Full”) then

    select random peer, \( p_j \)

    for each \( p_j \) note equal \( p_i \) do

      if \( p_j \in P_i \) then

        find RTT

        if \( RTT < RTT_{th} \) then

          \( P_i \leftarrow P_i \cup p_j \)

          Update NT

        End if;

      End if

    End for

  End if

Increment \( t \)

End while

**Figure 4.3 Algorithm for delay-aware peer discover**
4.3 OVERLAY UPDATING PROCESS

The next phase of overlay topology maintenance is the updating process. The topology management approaches are more or less characterized by these choices, namely, what information to collect, what criteria to apply to form the overlay network, and what action to take. When the information used for topology management is exchanged between the neighbours, the advantage is that all nodes do not have the same information about the same neighbor. That is the nodes are not making decisions based on the experience or observation about candidates but they have to rely on the information delivered by the neighbors and trust that information. The staying duration of a nodes’ neighbourhood is used as criteria when updating the neighbours’ connection. The integrity depends on the lifetime of the nodes connectivity. The underlying assumption is that the longer the node has been the neighbor, the longer it will be in the network and provide resources. The need for searching and establishing new connections is in turn reduced and decreases the traffic generated by topology management.

Actual change of topology consists of adding and removing connections. Addition and removal of connections are naturally needed when peers join or leave the network. When a node joins the network, it needs to add a connection to a node in the network. When a node leaves the network, it may inform the neighbours of it and disconnect all the connections to the neighbours. The topology adaptation can also appear in the other situations when a node wants to change its neighbourhood to a better one. The node may also replace the connection when it adds a new connection to a node and removes an existing connection. Topology management has to define rules of when and how the topology is managed to optimize the node’s neighborhood. The methods rank the nodes based on the defined criteria and use ranking information when selecting new neighbors or removing the existing
neighbors. A topology management method includes the actions to reconstruct the overlay, initiator of the actions, and the extent of the actions. The methods may use one or several of the criteria mentioned to rank the neighbor nodes. The method used in the present research has a fixed optimization target, such as to decrease the delay. Thus a predefined criterion has been used, but there are also more general solutions without set criteria.

Since the node’s mobility is a major concern for resource transaction, it is necessary to track the movement of each peer. However, tracking the movement of each and every peer in the overlay is a mammoth task. Keeping the task’s enormity in mind the refreshing act for neighbours connectivity is made based on the action of monitoring the status of each of their neighbour peer. For that we derive a metric called neighbour coefficient (Neigh_Coff) is derived to evaluate the connection position and call the topology management for recalculating neighbours stability. This would lead to addition and deletion of new neighbour peers according to the proposed metric. Neigh_Coff is computed as in Equation (4.1).

\[
\text{Neigh}_\text{Coff} = \left( \frac{\text{Common neighbors between } P_i \text{ and } P_j}{\text{Total neighbors count of } P_j} \right)
\]  

(4.1)

where

\( P_i \) is the neighbours list of peer \( p_i \)

\( P_j \) is the neighbours list of peer \( p_j \)

\( j = \{1,2,...,m\} \)
The value of the Neigh_Coff helps to decide the changes of link connectivity in a dynamic environment. For the maintenance of overlay topology in Gnutella, the connectivity status is periodically verified using the message ping-pong. Assume a peer \( p_i \) has a list of neighbours in \( P_i \) where \( P_i = \{p_1, p_2, ..., p_m\} \) in which “\( m \)” is the total number of neighbour peers.

To maintain the overlay, peer \( p_i \) sends information of its neighbour list to all the nodes in \( P_i \) to calculate the neighbour coefficient of the entire peer that are known to \( p_i \). Peer \( p_i \) also receives such information to make updates in their neighbour connectivity status. Given the information received by \( p_i \) from its neighbours, \( p_i \) executes algorithm shown in Figure 4.4 for topology maintenance in the overlay network. The algorithm is to provide the neighbour the status of neighbor stability and works as follows: for each \( p_j \in P_i \), \( p_i \) examines the change in its common neighbour list and decides to drop \( p_j \) from its neighbour statues. If the neighbour coefficient decreases from its threshold value, the link is severed, otherwise it is maintained. Therefore, \( p_i \) also attempts to connect to the nearest peers, which are not already its neighbours on delay criteria basis. In the end, to discover a better neighbour, \( p_i \) selects a random peer \( p_r \) that it has not involved in the above said process, and evaluates the neighbour coefficient for it. Once again, if it results in a higher neighbour coefficient, is added to \( P_i \). The process of updating allows the topology to be dynamically updated with the most recent position of the peers, providing resilience to node churn and node mobility. Periodic monitoring and assessing the link stability through the Neigh_Coff could meet the demand of the dynamic nature of the MANET on overlay network.
RTT : Round Trip Time

NT : Neighbour Table

LB : Lower Bound of neighbour’s size

RT: Routing Table

t₀ : initial time

t₁ : initial time + time period

Δt : time period

Input: P, i’s list of all possible neighbour nodes

Output: L, i’s list of eligible neighbour nodes

Require : neighbours list with their Neigh_Coff

If (sizeof (I) < LB) then

// Choose random node p from neighbours and request its neighbours list;

// Update common neighbours list and calculate neighbour coefficient Neigh_Coff between u and v

\[ t_1 = t_0 + \Delta t \]

for each \( p_j \not\in P_1 \) do

Calculate \( (\text{Neigh}_\text{Coff})_{t_1} \)

If \[ (\text{Neigh}_\text{Coff})_{t_1} - (\text{Neigh}_\text{Coff})_{t_0} \leq \epsilon \] then

Break

Else

Mark Dangerous peer

Calculate RTT

If \( (\text{RTT})_{t_0} > (\text{RTT})_{t_1} \)

Update neighbourlist
Else remove $p_j$ from $P_i$

End if;

End if

End for

if conn_status($p_i$) ≠ “full”

Select random peer $p_j$

For each peer $p_j \neq p_i$ do

Algorithm for delay-aware peer selection

End for

End if

End if

Figure 4.4 Algorithm for topology maintenance

4.4 OVERLAY WITH CROSS LAYER INTERACTION

The difficulty in designing a neighbour selection method is that each peer knows information only of a certain number (not all) of peers in the system. When peers join and leave it causes traffic overhead to transmit the information, which changes according to the movement of nodes. The peers are reshuffled by mobility and peer failure. Cross layer interaction is a potential solution to overcome the difficulties. Gnutella has not been designed for ad hoc networks, and suffers from node mobility, causing peers not to achieve minimum connectivity requirements. The protocol generates traffic bursts in correspondence of topological re-configurations. An alternative approach based on cross layering has been proposed to overcome the incompatibility between the overlay layer and network layer. To highlight the cross-layer interaction, the present research is based on the work of Eleonora
Borgia et al (2006) where the cross layer (XL) interface work has been introduced to expand the protocol’s interaction capabilities and possibly implemented in standard protocol stacks without spoiling the clean design.

Recently, Gnutella has been improved significantly so that it could handle the scalability problem via the introduction of Leaf and ultrapeer. Leaves are weak nodes with limited storage, processing power, and network bandwidth resources, while ultrapeers are powerful nodes. The number of pong messages generated in response to a ping message is the number of entries in the pong cache of the responding peer. Having random connections with the other peer, results in routing inefficiency. To address this problem, the system introduces a hierarchical structure with ultrapeers and leaves. A leaf keeps only a small number of connections with ultrapeers. On the other hand, an ultrapeer maintains many leaf connections as well as a small number of connections to the other ultrapeers. It acts as a gateway to the Gnutella network for the peers connected based on their content and shields these peers from the majority of message traffic.

The result of simulation of Schollmeier et al (2003) shows that a straight forward implementation of Gnutella on top of MANET is not promising, as lack of awareness of physical network topology increases network overhead. The success of the proposed architecture lies in its ability to create an efficient Gnutella overlay network using a cross-layer communication mechanism. Implementing a cross layer interaction between network layer and application layer in a standard protocol is a challenging task. To solve this problem, the work presented in Conti et al (2004) is referred, where a vertical stack component is introduced to extend protocols interaction capabilities beyond the standard layer interfaces. Using a cross-layer interface (XL-interface) inspired from this model, Gnutella peers could initiate peer discovery requests to an on-demand routing protocol, or ask
proactive routing peers to spread their own credentials around together with the control packets through the cross layer interaction. In the same way, routing peers could notify local peers about the reception of discovery requests and reply back when necessary, or, in the proactive case, notify local peers about the reception of fresh peer credentials along with incoming link state updates. The cross layer interaction is shown in Figure 4.5.

The MPRSP protocol stack offers a promising approach, by introducing a cross-layer communication channel between the physical network layer and the virtual P2P network layer. Thus, the Peer-to-Peer layer is well aware about the wireless network. Resulting MPRSP minimizes the signalling traffic overhead significantly and copes with frequent route breaks. Additionally the network layer is provided with the knowledge about the Peer-to-Peer application, to establish routes only to the best communication partners. As has been proved by means of simulations, MPRSP reduces significantly the messaging overhead and increases the search success rate.

A variant of DSR by name LSDSR could create an on-demand routing protocol by taking link quality into account. As ad hoc nodes are autonomous nodes, the discovery of other nodes in the system is based on the reactive routing protocol of LSDSR which has the mechanism to discover nodes based on their requirement and fill their routing tables based on nodes position. LSDSR is an extension of the DSR routing protocol that implements all the functionalities of DSR. LSDSR uses a link cache instead of path cache to efficiently maintain link information. The key idea behind calculating link-state information was to calculate the physical link delay. This link delay was used the present research as the metric for judging the link quality. Moreover, in LSDSR the resource information is added to the hello and response messages, to know the particular service offered by each peer. Hence, LSDSR provides the route for accessing the desired resources. LSDSR could reduce
the routing down time and flooding and hence LSDSR not only improves the routing discovery time and link load balance, but also optimizes the routing request message and reduces the packet loss rate.

![Diagram of cross-layer (XL) Interaction](image)

**Figure 4.5 Cross-layer (XL) Interaction**

Keeping the need for compatibility between both overlay and underlay layers, a Gnutella peer could exploit the extension on routing protocol and perform peer discovery in conjunction with route discovery at the network layer. Gnutella peers could make their demand through cross layer events along with the routing peer. They spread their resource list together with control packets. From the routing peers the application on peers learns about the link quality such as delay, signal strength and liveliness of the link. To outline the link quality of a wireless link, latency is used to account for each link. Figure 4.5 shows the way the cross layer interaction takes place between overlay and underlay networks. Whenever a peer needs to find a resource peer it makes use of the LSDSR which in turn makes use of the LinkInfo (link information) events, to which the routing peer subscribes. The key idea behind calculating link-state information is to calculate the physical link delay. This link delay is used as the metric for judging the link quality.
**ContInfo** (content information) events to which the local Gnutella peer subscribes is used to inform about the resources to the other peers. Ping messages generated by the peers that are accompanied with the **ContInfo** event help the peers to know the resource content list available with the other peers. These events received from the other peers are used to notify the local peer about the resources of these peers, received together with a routing control message. The Gnutella overlay formation method proposed could minimize the overhead since the LSDSR initiate the route search reactively. When mobility is high, the cached routes may not reflect the current topological status. LSDSR routing handles this situation by obtaining fresh routes for each search. Figure 4.6 explains the information passing from the application layer to the network layer and vice versa.

![Cross-layer communication](image)

**Figure 4.6 Cross-layer communication**
Ping, pong, query, and query hit are the crucial messages for Gnutella operation. Ping is used to discover peers on the network. A peer after receiving a ping message sends one or more pong messages. A pong message contains information on a peer. When a peer receives a pong message, it stores the obtained peer information in its pong cache and tries to make connection to the peer. Each entry in the pong cache corresponds to a pong message. Table 4.1 explains the node’s status framed by their neighbour connectivity. Gnutella logical network formation defines four node states (‘idle, connecting, stable and full’) that would help to maintain the connecting degree within the bound. Table 4.2 explains the four possible states in which a node may exist. Each peer maintains a neighbouring table to store the peer to peer connection in the overlay network. This scheme of the present research enforces limit on the connections established with the neighbours.

A pong cache mechanism and link policy selection are integrated in this schema to achieve better performance in the Gnutella logical network formation. The term “node”, as used in the present research, is a peer. Apart from the overlay formation and the resource discovery processes, rest of the Gnutella protocol is left unaltered.
Table 4.2 Neighbour state table

<table>
<thead>
<tr>
<th>Neighbor Table Size</th>
<th>State</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Idle</td>
<td>Must find new neighbors</td>
</tr>
<tr>
<td>UL</td>
<td>full</td>
<td>No acceptance of new connection or initiate a peer discovery process.</td>
</tr>
<tr>
<td>1 to LL</td>
<td>Connecting</td>
<td>Runs peer discovery process and accepting the new connections</td>
</tr>
<tr>
<td>LL to UL</td>
<td>stable</td>
<td>Runs peer discovery process if needed</td>
</tr>
</tbody>
</table>

4.5 PERFORMANCE EVALUATION

In this section, the results of simulations that were conducted accordingly as described in the previous subsection are presented, and the obtained results are evaluated to see the performance of the proposed system over Gnutella and P2P system using only P2P metric (Choi et al 2006). As a measure for the performance, the followings have been investigated:

- **Average query success rate**: Ratio of queries that are replied by one or more query hits over the total initiated queries

- **Average query response time**: Time duration from the time when query is sent to the time when the corresponding query hit is received at the query

- **Average control overhead**: Number of non-data packets sent per peer per second.

4.5.1 Simulation Environment

In this section, the simulation environments on which the simulations are executed are described. The Network Simulator (ns-2 version
2.26) is chosen as a simulation tool. Each run of the simulator accepts a scenario file as an input that describes the initial location and mobility pattern of each node in the network. The experiment consists of defining an order with which the nodes start running the overlay, and no message generation is required from the application. In this way the overhead introduced for the overlay construction and management can be measured. Table 4.3 shows the parameters chosen in performing the simulations.

**Table 4.3 Simulation parameters**

<table>
<thead>
<tr>
<th>Description</th>
<th>Default Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation Area</td>
<td>1000*1000</td>
</tr>
<tr>
<td>Simulation Duration</td>
<td>600s</td>
</tr>
<tr>
<td>Packet Size</td>
<td>1024 byte</td>
</tr>
<tr>
<td>TTL</td>
<td>7</td>
</tr>
<tr>
<td>Number of Session</td>
<td>2</td>
</tr>
<tr>
<td>Node state connections</td>
<td></td>
</tr>
<tr>
<td>Full</td>
<td>7</td>
</tr>
<tr>
<td>Stable</td>
<td>5</td>
</tr>
<tr>
<td>Connection</td>
<td>&lt; 5</td>
</tr>
<tr>
<td>Idle</td>
<td>0</td>
</tr>
<tr>
<td>Low mobility mode</td>
<td></td>
</tr>
<tr>
<td>Pause time</td>
<td>1</td>
</tr>
<tr>
<td>Speed</td>
<td>0-2 m/s</td>
</tr>
<tr>
<td>High mobility mode</td>
<td></td>
</tr>
<tr>
<td>Pause time</td>
<td>10s</td>
</tr>
<tr>
<td>Speed</td>
<td>5-30 m/s</td>
</tr>
<tr>
<td>Transmission range</td>
<td>150m</td>
</tr>
</tbody>
</table>

As seen in Table 4.3, 100 seconds is allowed for the peer application to stabilize before starting to get the information. Two situations slow and fast, representing the movements of nodes is created. Nodes mobility pattern has been modeled based on two models:
Random way point (PalChaudhuri et al 2005) is used to place and move node in the network. In this model, it is assumed that nodes are in cars and they can move up to a speed of 30 m/s (Wikipedia 2009.). Nodes inside big trucks are not moving that much, thus, the node speed is almost slow.

Random trip walking with reflection is used to navigate node in the overlay network. Most time, the disaster sites are organized with mobile device users. On an average, the walking speed of human is 1.4 m/s with a pause time equal to 10 seconds.

For P2P systems, the number of P2P nodes is set to 60 among 100 mobile nodes network area of 1000m X 1000m for the simulation. Non-P2P nodes have been used just to form an ad-hoc network. The ratio of ultrapeers to leaf nodes is set to 1:3 among the P2P nodes (Qiu et al 2007). For each P2P node, ping timeout has been set to 30 seconds. For each Gnutella messages, initial TTL value has been set to 7 as recommended in Klingberg & Manfredi (2002). The proposed scheme, has been compared with two other schemes, namely, Gnutella over AODV, and LSDSR. Many experiments have been done and the average across all the experiments drawn as results. Additionally, situations where the overlays had undergone to bursts of increasing number of node churns (i.e., node replacements) has been proposed, to study the reaction of the three protocols. Two types of topologies, physical topology and logical topology, have to be generated in the simulation. The physical topology should represent the real topology with Internet characteristics. The logical topology represents the overlay P2P topology built on top of the physical topology. All P2P nodes are in the node subset of the physical topology. The communication cost between the two logical neighbors is calculated based on the physical shortest path between this pair of nodes. To simulate the performance of different search
mechanisms in a more realistic environment, the two topologies have been simulated by Ns-2 simulator.

### 4.5.2 Result Analysis

In this section, the results of the simulation for the overlay formation mechanism have been shown and discussed.

The number of nodes has been varied without changing the terrain size that is varying the network density. The number of queries per peer is fixed. For networks under 60 nodes, the three protocol query success rate varies according to the situation they involved. Cross-layer with LSDSR performs evenly without lot of changes. The protocol used in the present research is not largely affected by the increase in network density due to the cross layer interaction.

Cross-layer Overlay for Multimedia Environment on wireless ad hoc P2P (COME-P2P) (Kuo et al., 2012) motivates for high data rate and time sensitivity on mobile network. It integrates both enhanced DHT (EDHT) lookup and IPv6 route to improve the delivery efficiency. The hop-by-hop routing path for overlay proximity is derived via the path information, which is altered to accommodate the mobility and the changing topology according to the latency. CL-MCHORD (Che-Liang Liu et.al, 2010) has Cross-Layer Mobile Chord P2P Protocol Design for VANET to provide scalable content distribution in vehicular networks. Aggressive table, try to use any available information to update finger table and overlay table for Mobile Chord.
Figure 4.7  (a) and (b) Average query success rate for node speed 2 m/s and 30m/s respectively

Figure 4.7(a) indicates that cross-layer LSDSR and Gnutella with LSDSR perform competitively in terms of query success ratio at lower density. As the number of nodes increase, the number of ultrapeer and content accessibility also increases. Cross-layer design becomes easier to search files and the request success ratio grows steadily. It is observed from Figure 4.7 (b) that the request success ratio of Gnutella with LSDSR reaches its peak around 50
nodes, and then falls. It is due to the utilization of flooding for a file search, which becomes the performance bottleneck as the size of the network grows. As far as Gnutella with AODV is concerned it is almost stable throughout the nodes increase. Since routes are established on demand in Gnutella over AODV it more or less behaves evenly when the density increases.

COME-P2P can use cross-layer information to map the latency-based topology to detect the peer’s mobility and derive a routing path like LSDSR in advance. For that reason, the Query success rate of COME-P2P is best among all compared schemes in slow moving speed. However, when there are increases in network size and speed, the aforementioned routing path cannot work well because EDHT changes too frequently. Hence our scheme and CL-MCHORD work better than COME-P2P.
Figure 4.8  (a) and (b) Average response time for node speed 2 m/s and 30 m/s respectively

**Average response time:** Figure 4.8 shows the average query response time with the increase in overlay nodes. The proposed system results about 8% and 4% lesser query response time than Gnutella over AODV when the maximum node speed is 2 m/sec and 30 m/sec respectively. Slow mobility and higher pause time are the contributing factors for the improvement in average query response time in the slow speed environment.

In low mobility the already found routes are valid for much longer time period. This means that the found routes can be used for more packets. From the graph above, it would be able to conclude that Gnutella over AODV established routes on demand and destination sequence numbers are used to find the latest route to the destination. The connection setup delay is less. Cross layer interaction with LSDSR reacts with the information obtained through cross layer, whenever the topology of the network changes. On the other hand, plain LSDSR does not have the mechanism to inform the overlay of its underlying activity like which route in the cache is stale, which link is
broken. Hence, the cross layer and the AODV based Gnutella works better than the plain Gnutella over LSDSR.

As Figure 4.7 illustrated, the proposed cross layer has the shortest completion time among the compared schemes. Because CL-MCHORD and COME-P2P inherit the DHT query, the querying latency of is longer than that of CL-LSDSR. However, the routing path can be derived from CL interaction due to the integration of overlay and LSDSR, so this shortens the response time. The response time is lengthened gracefully with the increasing speed, so the proposed our scheme is demonstrated with scalability. The update of path information not only recovers the routing path but also improves the overlay proximity, which avoids the far routing problem. As Figure 4.8 illustrated, although the response time is lengthened with the increasing moving speed, the query success ratio is still high, so the proposed scheme is demonstrated with mobility. Although the peers periodically probe their neighbors in Gnutella over AODV and Gnutella over LSDSR, the far routing problem cannot be avoidable. The routing path may be disconnected, and AODV reestablishes the routing discovery, which lengthens the service interruption latency.

**Control overhead:** As illustrated in the Figure 4.9, the control overhead of cross-layer over LSDSR linearly increases when the churn rate rises. In the proposed scheme, when a peer joins, its neighbors inform each other to update their node status and derive the routing path, and when a peer leaves, it proactively informs its neighbours.
As a result, cross-layer over LSDSR generates more control overhead when the peer churn rate is high. As illustrated in figure 4.9, the proposed scheme has more overheads for calculating latency and path information than the AODV based Gnutella design and it is unable to derive the routing path in advance to economize signaling overhead. On the other hand, Gnutella over AODV scheme, the signaling over head increases with the peers count but not with the churn rate. Figure 4.9 shown at pause time 0 seconds (low to high mobility) environment, AODV outperforms cross layer LSDSR and plain LSDSR in high mobility environment, by adapting to the changes quickly since it only maintains one route that is actively used. LSDSR have higher overhead as the need for periodic updates to the nodes mobility arises when compared with AODV. Due to the rapid change in topology it sends lot of packets to adapt the route changes in updating its table.

As Figure 4.9 illustrated, the signaling overhead of cross-layer over LSDSR linearly increases when churn rate rises. As a result, cross-layer over LSDSR generates the high signaling overhead at high churn rate. COME-P2P
has similar over-head due to the similar cross-layer design, the periodical updation also adds overhead. Similarly, CL-MCHORD faces the difficulty of peer churn, but it is unable to derive routing path in advance to economize signaling overhead. On the other hand, AODV sends its information within its packets with the increase in the packet size rather than the packet counts. That reflects in the graph shown in Figure 4.9 implying that AODV has less overhead than the other two protocols.

The overhead and average response time is always inversely propositional to each other. It is not normally possible to satisfy both the metrics for an efficient algorithm. If a discovery process takes low response time to find content, it will suffer from high traffic cost and vice versa. MPRSP is designed to keep the fair tradeoffs between the total traffic and average response time, thus obtaining a better search performance. It has been attempted to go for a reduction of large amount of traffic by increasing a little more response time. Measuring the overall performance and balancing both the metrics is an issue. Figures 4.10 and 4.11 show the control overhead per query and average response time versus the time.
The value $h$ is a time period for calculating neigh_coefficient, which is a deciding factor for the overlay formation. The impact of different values of $h$ ranging from 80s to 260s has been looked into. Figures 4.10 and 4.11 show the results of query response time and control overheads incurred with some of $h$ at 80s, 160s, 200s, and 260s, respectively, where the x-axis
indicates the time elapsed since the first query is sent. A small $h$ means the updation time is less and leads to a fast convergent speed. However, if $h$ is too small, peers will conduct the updating operations too often, making the overhead to keep growing. On the other side, if $h$ is too large, e.g., $h \geq 200$, the frequency of the updating operations is not being according to the changes of peers’ churn rate. Thus, the convergent speed is slow and the reduction of overhead or response time is limited. The figures 4.10 and 4.11 suggest that at the time intervals of $h = 80s$ and $h = 160s$ the performances is very close on both the metrics. However, when the $h = 200s$ and $h = 260s$, there is heavy compromise between overhead and response time. Considering this it is concluded that the optimistic updating period falls for 160 seconds, which means each single peer will probe its neighbors and update its neighbor table every 2.5 minutes.

4.6 CONTENT-AWARE GROUP MANAGEMENT

In this section, we present a multicast management protocol is presented for maintaining the groups and efficiently routing the data within that group in the overlay network. The multicast management protocol establishes and maintains neighbor peer information for each multicast group. the multicast scheme of the present research is based on a receiver-initiated group joining scheme to operate and pre-assign cores to groups. Groups are based on the types of resource and each peer may be a part of more than one group according to its willingness to share many types of resources. The multicast members are communicated within the group in a flooded fashion. Each member has to find the parent based on the link quality which has been collected during periodic updating of link status.

In brief, the working of the multicast algorithm is as follows: A core based distributed tree is formed and the implicit information contained in multicast announcements is used for message distribution. Application level
overlay is the topology created using latency as the metric. It uses heuristics to adapt its topology to the state of the underlying network. On top of the base overlay, it conducts forming groups based on content similarity. The topology is created as the members/hosts join the multicast group. Topology building algorithms define a definite relationship among the participating members and thereby create mesh topology. The relationship is a host-neighbours, or cluster member – cluster leader. In the present research, application level multicast protocol arranges the hosts into groups with periodic updates using the information from the network layer and distributes the data over it.

A part of multicast protocol called Protocol for Unified Multicasting through Announcement (PUMA) (Vaishampayan & Garcia-Luna-Aceves 2004) has been adopted in the work for multicast routing. PUMA-based protocol has been chosen as multicast routing protocol due to its better performance than the other representative multicast routing protocols. It implements a distributed algorithm to elect one of the receivers of a group as core of the group. It uses a receiver-initiated approach in which the receivers join the multicast group by using address of a special node, without the need for network-wide flooding of control or data packets from all the sources of the group. This protocol uses the shared mesh based multicast topology and eliminate the need for a unicast routing protocol and pre-assignment of cores to multicast groups. Its use of multicast announcements is to accomplish all the functions needed in the creation and maintenance of a multicast routing structure in a MANET. Multicast announcements are used to dynamically elect cores, determine the routes for sources outside a multicast group to unicast multicast data packets towards the group, join and leave the mesh of a group, and maintain the mesh of the group. PUMA uses the soft state approach for multicast group maintenance where multicast group membership and its associated routes are refreshed periodically by flooding its multicast announcement packet. In general soft state approach is preferred
over hard state approach and hence PUMA has been preferred for the present research.

Osamah et al (2009) show through their simulations that PUMA minimizes data packet overhead, using the core node to flood the group. In addition, it tends to concentrate mesh redundancy in the region where receivers exist by including all the shortest paths from the receivers to the core, which is also a receiver. Hence, PUMA protocol has been adopted to manage the multicast activity.

Each node member has a maximum number of children depending on its state of connectivity. When a node reaches the maximum number of connection (full state) it stops accepting new member connections. Core peer’s periodic broadcast announcement helps the peers to receive the multicast messages with control packets. The ultrapeer which has been connected to the group has the updates of the core. Ultraceers can be part of the Gnutella network and act as Gateway peers which play a gateway role for mobile device in cluster management and improve the quality of search application. An ultrapeer forwards a query to a cluster member only if it knows that the group is based on content for which the query has been raised. Peers never reply back the queries to ultrapeers. In sections 4.6.3 and 4.6.4, the join and leave method which would stabilize the node during node churn have been explained.

The findings of the source initiated route forms the shortest path tree for the overlay multicasting. The core keeps track of the multicast tree state using their Childnode Tables. These tables are used to localize multicast group membership management protocol to reduce the maintenance overheads and the impact of reconstructing the multicast tree due to node churn. When the node churn occurs, they rely on the local topology maintenance control in the overlay network to repair the overlay. A local
recovery protocol is invoked to localize repair and self-healing of the multicast tree. In this manner, group membership operations are decentralized and managed efficiently.

Using simulations in ns-2, the multicast routing efficiency is compared with Progressively Adapted Sub-Tree in Dynamic Mesh (PAST-DM) (Gui & Mohapatra 2003) and Application Layer Multicast Algorithm (ALMA) (Ge et al 2006). The results of these experiments show that, for those conditions in which PAST-DM and ALMA perform at their best, MPRSP attains the same or better packet delivery ratios as PAST-DM and ALMA, while incurring the same or far less overhead per packet delivered. Furthermore, the results also show that the savings in control overhead in PUMA can be orders of magnitude compared to the overhead of PAST-DM and ALMA, depending on the mobility of nodes, group size, and number of sources per group.

4.6.1 Formation of Multicast Group

In this section, the construction of multicast group structure is elaborated. Figure 4.12 captures the hierarchical overlay formation in the algorithm. The ideal topology is defined as a collection of groups of nodes, where the nodes in a group are related based on resources which they willing to share. The aim of the present research is to come as close to the content availability as possible using only content information. Same content holding nodes are organized in a multicast group, labeled as member nodes ‘Mnodes’. Gnutella v.06 is used as a representative of unstructured protocol.

Open platform like Gnutella distribute data without the need for a central server as in Napster. With this context, adhoc networks can be considered in synergy with traditional P2P systems which organize themselves into logical network and co-operate independently with each other
for the availability of the digital content. Hence, Gnutella has been chosen to meet the various requirements, such as the ability to search and retrieve the resources and work in dynamic environment which are the basic activities of digital retrieval in movement. The Gnutella V6 network provides peer hierarchy classification as Leaves and Ultraceers. Leaves are weak nodes with limited storage, processing power, and network bandwidth, while ultrapeers are powerful nodes.

Leaf node: This is the node with reduced storage capacity, CPU processing time, and network bandwidth.

Ultraceer node: This is the node with large storage capacity, high processing power, and high network bandwidth. Ultraceers maintain many leaf connections, as well as a small number of connections to other ultrapeers.

Figure 4.12 Multicast group formations
**Grouping Criterion:** A peer share digital resources are classified into several categories like Text, Music and Movies. Based on this information, peers self-organize into clusters by joining or leaving different parts of the overlay. A physical node may join several clusters as a logical node. A logical node represents the presence of a physical node inside a cluster. First, the locality of related content has to be increased by connecting the resource content peer to form a cluster. The group is framed with the same type of digital resource to be shared. Forming the group based on certain resources can help to minimize the search latency and reduce the overhead incurred by maintaining the overall network peers.

Each group has a group leader called core peer of that group which would keep group information and guide the search process. High amount of files holding node, is labelled as core peer and attached to physically close Mnode, which are member nodes of that group. The core peer is being selected based on the number of files available for that resource type. The peer holding maximum number of the same type of resources is the Core peer and if the two or more have the same count then the smallest ID peer is selected as the core peer. Obviously, a core peer and the Mnodes attached to it constitute a cluster and the core peer acts as a rendezvous point of the cluster. In order to avoid overloading core peers and to relieve the single point of failure problem, an upper bound on the number of nodes that a peer can be attached is imposed. It is denoted as the upper limit. To achieve a good scalability, the default value is set according to Gnutella limitations.
As shown in Figure 4.13, each group with a special ID along with their content ID can select a core peer for its group. With the information contained in such announcements, nodes elect cores, and each node in the network learns one or more routes to the core. Selected core sends this CS (Core Selection) message to the ultrapeers for further contact. In case the ultrapeers receive two or more core for the same content type, it will select the least ID Core peer. In order to resolve group leader conflicts, ultrapeers send CAcc (core accept) message with the selected core ID back to the core. It shows confirmed way of administrating the whole dynamic network.

4.6.2 Effective Management of Multicast Announcements

In a dynamic network, the receivers elect an intermediary peer called core peer, to serve as the point of contact between the group and non-
members, and the intermediaries must flood the news about their existence to the rest of the nodes. The analysis shows that the control overhead of MPRSP is fairly independent from such factors as mobility, number of receivers, and the number of multicast groups.

The multicast algorithm used is receiver-initiated approach in which the receivers join a multicast group using the address of a special node called core (group header) without flooding the network for all the source nodes involved. It is implemented as a distributed algorithm to elect one of the members of a group as the core of the group, and to inform each ultrapeer in the network. In order to reach the farthest core each ultrapeer has one or multiple paths to the core. Each core is in contact with one or more nearest ultrapeer. All the nodes form the mesh overlay and MPRSP chooses a Core peer per multicast group in each connected component of the network. When a receiver needs to join a multicast group, it first determines whether the ultrapeer has the address of that Core peer. If it has, the receiver then collects the address of the Core peer, and tries to join the same group. Otherwise, it considers itself the core of the group and starts transmitting multicast announcements periodically to its nearby ultrapeer stating itself as the core of the group. Core peers propagate multicast announcements (Osamah et al 2009) based on their proximity to the ultrapeer selected from their neighbors’ table. The format of the multicast announcement is shown in Figure 4.14.

<table>
<thead>
<tr>
<th>Sequence number</th>
<th>Address of group ID</th>
<th>Address of core ID</th>
<th>Distance to the core</th>
<th>Flag</th>
<th>Parent ID</th>
</tr>
</thead>
</table>

**Figure 4.14 Multicast announcements format**

A node that believes to be the core of a group transmits multicast announcements periodically within that group. Multicast announcements is a
single control message for all the activities like core select information, leaving or joining of group information and maintenance of group. Every member node connects to the elected core along all the shortest paths between the member and the core. When the data packet reaches a Core node, it is flooded within the group, and the nodes maintain a packet ID cache to drop the duplicate data packets. Each multicast announcement specifies a sequence number, the address of the group (group ID), the address of the core (core ID), the distance to the core, and a parent that states the preferred neighbour to reach the core. Successive multicast announcements have a higher sequence number than the previous multicast announcements sent by the same core. With the information contained in such announcements, nodes elect cores, notify others about joining or leaving the mesh of a group, and maintains the mesh of the group. For the same core ID, only multicast announcements with the highest sequence number are considered valid. Hence, all the valid entries in the connectivity list at any point of time have the same core ID and sequence number. Among these valid entries, the entries with the shortest distance to core qualify as the best entries and the neighbours corresponding to these entries are called parents.

As the multicast announcement travels through the group, it establishes an Mpeer list at every node in the group. A node stores the data from all the multicast announcements it receives from its neighbours in the connectivity list. Mpeer list contains information about each neighbour’s ID along with their parent ID and their distance to the core. If a higher sequence number multicast announcement comes from a neighbour the node overwrites entries with lower sequence numbers for the same group. In each multicast group, a node keeps only the latest information of its neighbor in its Mpeer list. Using Mpeer lists, the nodes are able to establish a mesh, and route data packets from the senders to their receivers. Each entry in the Mpeer list, in addition to storing data from the multicast announcement, also stores the time
when it was received, and the neighbour from which it was received. The node then generates its own multicast announcement based on the entries in its Mpeer list.

Each node maintains a soft-state that records the round-trip time to each of its neighbors in the node. The table is maintained by exchanging alive messages occasionally, and is useful in selecting parents and tree partition recovery. A node chooses one of its parents to set its parent field, which is referred to as the chosen parent. The manner in which this selection is made determines the amount of redundancy in the mesh. A node sets its membership code field based on whether it is a non-member, only a receiver or both. After receiving a multicast announcement with a fresh sequence number, nodes wait for a short period to collect multicast announcements from multiple neighbours before generating their own multicast announcement.

Each core peer keeps track of the multicast tree state using their Childnode_Tables to reduce the maintenance overheads and helps in node churn. The core keeps track of the multicast tree state using their Childnode_Tables. The Childnode_Tables contains ID of their child peers and their predecessor peer list. These tables are used to localize multicast group member ship management protocol to reduce the maintenance overheads and the impact of reconstructing the multicast tree due to node churn. Node churn problem refers to the continuous process of node arrival and departure in distributed application. When node churn occurs, they rely on the local topology maintenance control in the overlay network to repair the overlay. The Childnode_Tables information has been used to decide with whom to establish connection during churn. A Recuperation process has been invoked to localize repair and self-healing of the multicast tree. In this manner, group membership operations are decentralized and managed efficiently.
4.6.3 Join Process

Leaving and joining processes involve the message exchange that happens between the peer and the group members. Here, when the peer joins the multicast group, the content group has been reached by having the group ID collected from the ultrapeers. When a node wants to join the multicast tree, it sends a join message to the nearest member. It gets a list of its children and learns RTT by cross layer interaction event. Every peer has Childnode_Tables and distances to these children.

![Diagram of Join Process]

Figure 4.15 Join process

When a node wants to join to the multicast tree, it sends a join message to the nearest member. The receiving peer replies with GAct (Group Accept) or GDcen (Group decline) message. If it is GAct the peer gets list of the receiving peer’s children list and learns RTT by cross layer interaction event. Every peer has Childnode_Tables and distances to these children.
For instance, P in Figure 4.15 compares the delay values and decides the way to make a move. We aim to find the most appropriate parent for a peer so that the data travels the minimum possible path. The key idea is to connect the nodes which are reached with less latency, so that the source-destination path length for the overall structure is minimized. An iterative approach has been used for selecting a child or a parent. Suppose that there is a source (S) and an existing node (E) which is already in overlay network. There is a new node (P) which is going to join to overlay tree. The new node measures the distances among these three nodes. In Figure 4.15, S and E which are already in overlay network keeps their position while P could be in there in different position.

When a node wants to join the overlay tree, it sends a Join message to the core. It gets a list of its children and learns RTT by probing. Every node has a children list and distances to these children. For instance, P in Figure 4.15 compares these values and decides the way to continue. Now P has four children of S for selection along with their delay. It selects the least delay peer as their potential parent. Peer P then selects the least delay parent which is E_1 in this case. When another node P_1 wants to join, it will repeat the process, and it selects the same way P has selected its parent, that is P in this case. If the potential parent is the leaf node that does not have any children, P will connect directly to this node without further query. This case is valid when P is the first node connecting to the multicast tree. Due to churn, if P finds a better node, it connects and becomes a child. A node can accept connections up to its maximum limit, which is called “Upper limit”. Each node has a predefined connectivity limit which has been explained in section 4.3. If node E_1 reaches its limit, P contacts children of E_1, which can accept connection without breaking its degree limit.
A pseudo-code for the Join procedure is given in Figure 4.16 nodes store some state information regarding their children in Childnode_Tables. They also store their predecessor list. For a join process, when P gets pong responses from parent and all children, it first find out their delay values. First, it is assumed that P is in position between S and E1 position. That is the new node is between two existing nodes a parent (S) and currently checked child (E1). The delays among these three nodes are then calculated. Now P decides that whichever node distance is small is apt for a parent candidate. Now proper connections are made, and join process is done. Whichever peers’ delay is minimum that peer is selected as their parent. In this case, P selects S as its parent and the child of S that is E1 becomes P’s child. If the new peer P lies away from the node S and near its child node E1 the E1 becomes its parent node. If the potential parent doesn’t have free degree slot to accommodate a new connection, new node connects to the closest free child of the potential parent.

// Peer joins Multicast Group

S = Potential parent
P = New node
E1 = member node
D (AB) = delay between A and B peers
Join (P)
while
P pings S and all its children of S
    If E1 is NOT FULL
        If P is between S and E1
            If D (PS) < D (PE1) then
                S becomes parent of P
                P becomes parent of E1
            Else


E_i becomes parent of P
End if
Else if E_i is between S and P
If D(PE_i) < D(PS) then
E_i becomes parent of P
End if
Else
If S is a leaf node
P connects to S
Else
P connects to free child of S
End if
End if
End while

Figure. 4.16 Algorithm for join process

If the peer reached the threshold of the upper bound level (which has been explained in section 4.3), it sends GDcen message and Childnode_Tables to the new peer. If the new peer receives GDcen message, it moves to the next preferred link to repeat the process. If the requesting peer is interested, it could make use of the child list for establishing connectivity with any of their child. Otherwise it starts the join process with the other peers, as mentioned in Figure 4.17 to be part of the tree. All the messages are sent in unicast way to reduce unnecessary traffic. There is a possibility that the new peer may not receive any messages from ultrapeer. This situation conveys two possibilities. First is that there could be no group formed based on that particular content. Another is that there could be a break in the multicast tree which could not be reached by the ultrapeer. In the first case the peer itself forms a group based on that content and in the second case the new
peer will wait for some multiple RTT time and then try to find other route to reach the ultrapeers

### 4.6.4 Leave Process

To leave the multicast group the member sends a Leave message to the parent and all the group membership information of this peer are removed from the parent’s Childnode Tables. The parent node updates the information, removes the node from the multicast data, and informs the core peer. The leave process of three cases has been explained through Figures 4.17 (a) to 4.17 (c).
Figure 4.17 (a) Child $S_i$ leaves process (b) Parent $P$ leave process (c) Parent $P$ and grandparent $GP$ leave process

Figure 4.17 (a) shows the methods involved when a child $s_i$ leaves the tree. As the child is a leaf node the modification carried out would not affect any other node except the parent node. The child node inform only to the parent about his/her leave activity and it is the parent’s responsibility to inform its leave to the core. However, the same could not be done in the process of parent leaving. Here, the parent has to inform all the children and its parent that is grandparent of its children. This process is depicted in Figure 4.17 (b). This process leaves the children without a link with the tree. This is rectified by using the predecessor list available with each node. The child nodes start reconnection process with the help of predecessor instead of core, to speed up updation on connectivity.

The affected children respond to this change by making a choice from its predecessor list available in Childnode_Tables. They find the closest neighbouring peers from the predecessor list to rejoin. These neighbouring predecessors provide the information about their children from their Childnode_Tables. Since all the leaf nodes are close to each other, they have common parent, it is expected that parent-child relationships will be
established among them after rejoin. Figure 4.17 (c) illustrates the case of parent and grandparent of a child leave at the same time which is expected to happen very occasionally. The information about their departure has been properly informed by their respective parents to the core.

With the latency information about the predecessor and their children, the closest one is chosen as a parent. The child nodes allow to start the reconnection process at core if it is not able to find any predecessor. The parent Childnode_Tables is updated with the affected children information and the children update the predecessor list accordingly. This process helps the children and the parent peer to recover quickly from the node churn in a highly dynamic environment.

$S_{(i,...,i)}$ : leaf peer of 1to i  
P: parent  
PG: Grandparent  
Childnode_Table : table stored at each node about their children and their predecessor  
For leave ()  
If leaf child ($S_i$) leaves  
$S_i \notin P$. Childnode_Table  
P informs core  
ElseIf Parent (P) leaves  
P $\notin$ PG. Childnode_Table  
P informs children  
GP informs core  
For each $s_i$ in P. Childnode_Table  
Join process()  
End for
ElseIf Parent (P) and grandparent (P) leave

\[ P \notin PG. \text{Childnode\_Table} \]

P informs children

\[ P \notin S_{(1..i)}\text{-predecessor\_Table} \]

GP informs core

For each \( S_i \) in P. Childnode\_Table

Join process ()

End for

End if

End for

**Figure. 4.18 Algorithm for leave process**

In Figure 4.18 the action taken by every node during the leave process is explained. The group members are periodically updated. The parent node periodically sends Hello message to their children. If there is no response for a period of time the parent assumes that the child has moved out or dead. That subsequently initiates the repair process with the Leave operation of the child and the repair is done by the execution of Recuperation process. Whenever there is a change in the member connection, only few Join and Leave message would reach the core. The Join and Leave messages are accounted for the stability of the tree at that particular peer. If this number exceeds certain threshold the multicast tree sourced at the peer is reconstructed. This is because some peers may be struck in an unfair and unfavorable situation.

**4.6.5 Performance Analysis**

In section 4.6.5 MPRSP has been analyzed by varying the number of group size and speed. Considering that only multicast has been used to deliver the content, in this section the multicast management efficiency has
been analyzed using the metrics: the tree cost, the link stress and peer stretch. However, the metrics such as the tree cost, the link stress and peer stretch are expected to vary proportionally with the number of nodes in a group. A group with up to a maximum of 50 nodes is considered. Among these nodes, one or more nodes are randomly selected as the content provider (i.e., owner node). Content size is defined as 100 KB. At some point, some requesting nodes start the content discovery process. Each simulation is run with mobility (with maximum speed set to 2 m/s) and without mobility. Each experiment is run for a period of 4 hours. This period is sufficient to test the downloading process of various content types by various peers. Using simulations in ns-2, the multicast routing efficiency is compared with PAST-DM (Gui & Mohapatra 2003) and ALMA (Ge et al 2006), which are representatives of mesh-based and tree-based multicast routing in overlay networks. ALMA is a receiver-driven of tree-based multicast routing and PAST-DM is source based mesh multicast routing on overlay networks. They both use link metric as a deciding criterion to update their link connectivity. Also, as each one represents different tree forming techniques, these two have been adopted for the comparison. The results from a wide range of scenarios of varying mobility, group members, and number of multicast groups show that the MPRSP algorithm incurs less control overhead than the other protocols.

Multicast networks must effectively handle the dynamic behaviour of the participating overlay nodes. New members may join the multicast group and, at the same time, some other members may leave the multicast group. Such changes may happen at different time points and the overlay multicast algorithm should allow for such changes in a seamless fashion and with minimum changes in the routing tables. Group reconstruction in the case of joining nodes and recovery from node leaving must therefore be done. A joining node may graft a new branch onto the specific tree and a leaving node may cause an existing branch to be pruned from the tree.
Ns-2 simulator has been used to conduct simulation experiments for evaluating the protocol. The core peers are assumed to be alive during the entire simulation time, and is known by other peers. Randomly selected 100 of 250 nodes join to the overlay multicast tree. 400s has been taken as a time interval and churn has been generated based on that interval. Based on the churn rate, a number of nodes join and leave the overlay. For example, if the churn rate is 10%, then 10 new nodes join and 10 of the existing nodes leave in each time interval. The number of nodes in the overlay is retained at 100 by the end of the 400s time interval. At the end of every time interval, there is a pause time of 50s, and the measurement is done. The tree to churn again in the next time interval is exposed after the measurement. For the experiments a default size of 15 group members is specified.

This process is repeated until the end of the entire simulation time. For instance, the nodes are renewed almost twice over lifetime under 10% churn. Some nodes may join and leave several times while some never join. There is 10 % of ultrapeer and they are all considered equal. Degree limits of the nodes are uniformly distributed within the range from 4 to 7. The protocols have been simulated under different churn rates from 1% to 20%. The simulation experiments have been repeated 20 times for each churn rate, and the results have been reported at 90% confidence intervals.

The following is the scenario of the present research. The system initially contained small amount of peers and reach the peers strength by joining sequentially. Afterwards, a simulation for 500s has been run, which is a loop of 20 runs, each run letting a peer to leave or a new peer to join. The probability that a peer leaves is $\rho$ (ranging between 0.2 and 0.8), thus a new peer joins with a probability $1 - \rho$. After the simulation stopped, statistics have been collected on the trees generated by MPRSP, PAST-DM and ALMA, respectively. PAST-DM, is an overlay multicast protocol defined for
MANET that tries to eliminate redundant physical links so that the overall bandwidth consumption of the multicast session is reduced. The virtual mesh in PAST-DM constantly adapts to the changes in the underlying network topology. Each node implements a neighbour discovery protocol using the extended ring search algorithm. The nodes periodically exchange link state information with their neighbors in a non-flooding manner. Thus, by looking at the link state of each node, a node gets a view of the entire topology. The information is used to build a source-based tree. PAST-DM yields a stable tree quality at a cost of higher overhead, which increases with the periodicity of the link state updates. PAST-DM constructs a logical mesh connecting all the group members like MPRSP protocol. A Source-Based Steiner tree is then constructed upon this mesh. The tree is then periodically refreshed. Both the logical mesh topology and the multicast tree are the main reason for the selection of this protocol for comparison.

ALMA constructs a multicast tree in a decentralized and incremental way. This approach is based on RTT measurements to detect and manage nodes' mobility. When periodic RTT measurements towards its parent exceed the threshold, a node has to perform a reconfiguration procedure of its delivery tree. ALMA is also based on an expanded ring search technique limited by a maximum hop count, to detect neighbors. This makes ALMA running over costly positioning systems, that incur considerable amount of control traffic, and thus is more likely to contribute to the overall congestion in the network. To investigate its performance under various scenarios, a simulation-based study has been carried out. The multicasting protocol of the present research has been evaluated by using the performance metrics namely, Peer Degree, Peer Stretch, Link Stress and Message Overhead, which are defined by Chu et al (2000).

The definitions of the metrics are given below:
- **Tree cost:** The total number of the physical links that make up the logical links in the multicast delivery tree. This metric represents the “goodness” of the structure created by the overlay multicast.

- **Link Stress:** The stress of a physical link is the number of identical copies of a multicast packet that need to traverse the link. This metric quantifies the efficiency of the overlay multicast scheme.

- **Path Stretch:** It is defined as the ratio of the number of hops the packets has passed through since its emission from the source and of the shortest path length from the source.

The results of the simulation are described in detail and observed behavior is explained.

![Graph](image)

**Figure 4.19 Tree cost for different group size**

**MPRSP creates a less expensive tree than PAST-DM and ALMA**

The multicast tree constructed from the research protocol comes with a lower cost in terms of the latency than PAST-DM and ALMA. Graph
shown in Figure 4.19 has been plotted by observing the tree cost as the group size increases. The decision at the source to create a logical Steiner tree in somewhat centralized manner affects the tree construction in PAST-DM. Compared to ALMA there is a 20-25% reduction in the research protocol. MPRSP is less susceptible to link breakages because it is a mesh based protocol. Even when a link breaks, a node does not need to inject control packets to rebuild it. It is able to lookup an alternate route using its connectivity list. The algorithm exploits the underlying link-state information between the members to update the overlay nodes and the peer starts the connectivity to the tree. PAST-DM creates to some extent a logical Steiner tree in a centralized way. Hence, the decisions at the source affect the creation of the tree globally and only the source starts the construction of the tree. In ALMA, the reconfigurations are handled by the receivers, and the local decisions turn out to respond to more tree cost as the nodes size increases. The MPRSP is shown to possess good scalability properties, as far as the tree create procedures are concerned, and good overall efficiency values.

**Figure 4.20 Average link stress for different group size**

**MPRSP creates a tree with lesser link stress:** The average stress observed is much smaller than that with PAST-DM and ALMA (with update period of 20). In Figure 4.20, the average stress has been plotted versus the group size. The possibility of bottlenecks is much lower than in ALMA and PAST-DM. This behavior is attributed to the fact that ALMA uses RTT, and PAST-DM
uses a hop count while the research’s algorithm uses link latency to estimate the quality of a logical link. When nodes choose parents, or decide to re-configure they utilize the neighbor updating algorithm to check the link liveliness. The algorithm of the present research makes the peers to maintain the connection within the limit or pass it to their children. Hence, in the present research algorithm, it is unlikely for a new node to choose a peer whose status is full as the neighbor. This makes the selection process based on latency and load status of the link. ALMA is based on an ‘expanded ring search’ technique limited by a maximum hop count, to detect neighbors. This makes ALMA running over costly positioning systems, that incur considerable amount of control traffic, and thus is more likely to contribute to the overall congestion in the network. When the group size increases, the number of transmission increases and so does the network load, and the stress.

![Figure 4.21 Average path stretch for different group size](image)

**Figure 4.21 Average path stretch for different group size**

**MPRSP creates a tree with lesser path stretch:** Figure 4.21 shows the results of average peer stretch. The peer-stretch metric is defined as the ratio of path-length from the requester to the sender along the overlay to the direct unicast path. The overlay formation uses the cross layer interaction from the network layer to keep the path gap between the overlay networks and
underlay network narrow. It also helps to form the tree and their parent selection with less delay. Hence, the stretch is less than PAST-DM which goes for hop count of the path. Results show that stress ratio falls between 1.45 and 1.52. It shows that the overlay stress comes under 1.52 and the overlay path is 1.5 times higher than the shortest path. When nodes choose parents, or decide to reconfigure, ALMA is actually responding not only to the path length but also to the congestion. Hence, there is a slight raise from MPRSP.

4.7 SUMMARY

In this chapter the concepts behind the MPRSP protocol and the corresponding components involved were introduced. The main areas of the research protocol, namely, the overlay formation and the overlay maintenance have been explained with implementation results. Here, the system keeps track of the link information together with the local resource information, which is fully distributed. To achieve this, the technique of cross-layer scheme has been adopted between peer to peer of network and MANET. A multicast scheme with appropriate operation has been applied to reduce the search scope. The results of the simulation reveal that the system employed in the research enables efficient access to shared files.