CHAPTER 4

Chapter-4. Bandwidth Aided Topology Awareness

<table>
<thead>
<tr>
<th>S.NO</th>
<th>Name of the Sub-Title</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Inefficient Bandwidth Utilization</td>
<td>67</td>
</tr>
<tr>
<td>4.2</td>
<td>Related work</td>
<td>68</td>
</tr>
<tr>
<td>4.3</td>
<td>Record Route</td>
<td>70</td>
</tr>
<tr>
<td>4.4</td>
<td>Capacity Utilization</td>
<td>71</td>
</tr>
<tr>
<td>4.5</td>
<td>Search Protocol</td>
<td>77</td>
</tr>
<tr>
<td>4.6</td>
<td>Proof of Correctness</td>
<td>79</td>
</tr>
<tr>
<td>4.7</td>
<td>Performance Evaluation</td>
<td>82</td>
</tr>
<tr>
<td>4.8</td>
<td>Summary</td>
<td>88</td>
</tr>
</tbody>
</table>
CHAPTER 4

Peer-to-Peer (P2P) networks are widely accepted and are growing at a rapid pace due to the ability of the peer-to-peer networks to sustain the major distributed system networking problems such as scalability, heterogeneity and the dynamic nature of the nodes present in the network. Although peer-to-peer networks provide many desirable properties for users like anonymity, low maintenance costs and fault tolerance, they have some inherent problems in the current peer-to-peer networks as follows.

4.1 Inefficient Bandwidth Utilization

The peer-to-peer applications like Gnutella use the mechanism of flooding or pure broadcast for a query search, which consumes a lot of bandwidth due to the number of query messages generated in the network. Bandwidth is a property between two nodes [55]. Here bandwidth between a peer and the rest of internet is determined by the nodes that directly link to the internet. As a result, the bandwidth is a considered as a property of the peer.

Studies in [5] describes that the average upstream bandwidth of nodes in a Gnutella network is around 1Mbps, while 22% of peers have bandwidth less than 0.1Mbps, 8% of peers holds the bandwidth more than 10Mbps, and the highest observed bandwidth capacities are around 100Mbps. This inbuilt heterogeneity in the Gnutella network affects the
performance of the system if all the nodes are treated equally.

The process illustrated here tries to solve the problems such as Topology mismatch as well as Inefficient Bandwidth Utilization with more focus on the bandwidth utilization. The bandwidth improvements in the system will have outstanding effects on the speed and performance of the system. The recent trends reveal that many organizations are more interested to improve the performance of the existing systems by monitoring and efficient utilization of bandwidth rather than increasing the amount of bandwidth due to high costs and resource consumption by the distributed devices [54]. This chapter presents the bandwidth aided topological awareness to improve the performance of the unstructured peer-to-peer systems like Gnutella. It is done by building a highly scalable system which avoids overloading any of the nodes by considering the capacity of nodes.

4.2 Related work

The older versions of Gnutella i.e. Gnutella 0.4 [2] uses the flooding based search to find files in the peer-to-peer network. To avoid the drawbacks of flooding mechanism, biased random walks are utilized in which the search query is biased towards high capacity nodes which are best to answer the queries. The biased random walks generate less number of messages compared to flooding which reduces the overhead on the network [52].
The present stable version of Gnutella i.e. Gnutella 0.6 [3] classifies the peers into ultra-peers and leaves based on the bandwidth and capacity of the peers. Unlike Gnutella 0.6, it is implemented without categorizing the peers, but the peers are assigned a capacity value which is used in connection establishment and query forwarding. This system allows for even load balancing on adapting the duties of the peers based on the available bandwidth of the peers.

Studies in [4] illustrates that the nodes in the Gnutella network show significant heterogeneity in terms of their capabilities such as bandwidth, online times, hard disk space and processing power. Gnutella 0.6 tries to solve this problem using the multilevel topology formation, but the problem is in the ultra-peer election process involves the decision of which peer to become the ultra-peer and configure the total number of ultra-peers for scalability which involves complex issues to deal with. The current work in this chapter considers the inherent heterogeneity and achieves better performance and scalability.

Gnutella Systems are not scalable when a large number of queries are introduced into the network. In such cases, the node in the network gets overloaded and system performance is affected. This problem gets even worse when the system size increases. Thus the system should be able to handle the higher rate of queries with increasing system size by avoiding the overload at any of the nodes in the network.
4.3 Record Route

In order to solve the topology mismatch problem, the proposed technique in this chapter uses the record route option of IP protocol. The record route is the option 7 of the internet protocol, which makes the nodes traversed during communication to add their IP addresses to record route data. The pointer field in record route will point to the position to insert the next IP address. The length field defines the number of IP addresses to be inserted in the record route data field. If the route data area is full i.e., pointer exceeds the length, an ICMP parameter problem message will be sent to the sender host. This length field is used to restrict the scope of choice of neighbors for a node. The format of record route option is as shown in Figure.4.1.

<table>
<thead>
<tr>
<th>OPTION TYPE (0-7)</th>
<th>LENGTH (8-15)</th>
<th>POINTER (16-23)</th>
<th>ROUTE DATA (24-31)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>hop length</td>
<td>Points to next node</td>
<td>IP address of current node.</td>
</tr>
</tbody>
</table>

Fig: 4.1, Record Route Format

Let us consider an underlay network with 24 nodes with random connections and with 20 peer nodes with capacity values having 5 levels. Since it is a small network we set the length field of record route to 3 (00000011) during the connection establishment phase. The Underlay and overlay representation of the network is as shown in Table 4.1.
Table 4.1: Underlay Network Capacity Levels

<table>
<thead>
<tr>
<th>Capacity level</th>
<th>No of Nodes</th>
<th>Degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>100</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>1000</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>10000</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Most of the neighbor connections are within 3 hop distances of the underlay network. After attaining initial number connections, the node will calculate the stability value. For example, for node H

\[
\text{Stability of } H = \frac{N\text{'s Capacity}}{N\text{'s neighbors}} + \frac{E\text{'s Capacity}}{E\text{'s neighbors}} + \frac{B\text{'s Capacity}}{B\text{'s neighbors}}
\]

\[
= \frac{(1000/4) + (100/2) + (100/3)}{100}
\]

\[
= \frac{(250 + 50 + 30.5)}{100}
\]

\[
= 3.5
\]

Fig: 4.2, calculating the stability of node H

4.4 Capacity Utilization

In general the capacity of a node is characterized by its capability in handling number of queries per second. The nodes with the low
bandwidth, query processing capacity are determined by available link-bandwidth of the node. On the other hand, the nodes with high speed connection, together with the bandwidth, other properties such as speed of the CPU, disk latency etc. will affect the node’s capacity. The bandwidth aided system implementation ignores the effects of CPU speed, disk latencies and the capacity is assumed as a function of available bandwidth, the overlay topology formation of the system is described as follows. When a node enters the network it uses the bootstrapping mechanism similar to the one found in Gnutella to find the other nodes of the network. Each peer maintains a cache containing the list of all other peers present in the network. The cache entries will contain the IP address, port number and the capacity values of the other peers. These caches are exchanged periodically with the neighbors and updated accordingly. The old and stale entries are marked as dead.

The main aim of the overlay topology formation is to ensure that high capacity nodes have higher degree and the lower capacity nodes are at short reach of lower capacity nodes. To achieve this, each node computes a stability level. This quantity is a function of the given nodes current set of neighbors, their capacities, degree and preconfigured maximum and minimum number of neighbors. This measure takes values between 0 and 1 that signifies how stable the node is with the current set of neighbors. As long as the node is not stable, it will continuously configure its neighbors and improves its level of stability.
The stability value is given as follows:

\[ S(X) = \frac{\sum_{i \in Nbr_X} \frac{C_i}{Nbr_i}}{C_X} \quad \text{Eqn.(4.1)} \]

The measure \( S(X) \) returns a value between 0 and 1. If the value is below 1, the node \( X \) is not stable and continues to search for neighbors with better capacity values. If the value is more than 1, the node \( X \) is stable and only tries to calculate the stability value periodically to keep the node stable.

When a node starts up with minimum preconfigured set of connections based on its bandwidth it is assigned with maximum number of connections. Now the node starts gathering more neighbors and improves its level of stability. The node will continuously reconfigure its neighbors until the node will have optimal set of neighbors. This process is described in Algorithm 4.1.

To add a new neighbor, a node randomly selects a small set of entries from its own cache that are not already neighbors or dead entries. From these entries, the node selects those nodes with greater capacity than itself and sends the connection request message. If no such eligible entry exists the node randomly selects one entry from the cache and sends the connection message.

During the connection establishment phase, each node makes a decision of whether to accept the other node as neighbor or not based on
its capacity with the existing neighbors and with the new one. To accept the new node, an existing neighbor will be dropped based on the capacity value the neighbor acceptance and the maximum number of neighbors that the node can accept.

Step 1. \( C_i \leftarrow \text{Capacity of Node } i \)

Step 2. \( \text{Nbr}_A \leftarrow \text{Number of Neighbors of } A \)

Step 3. \( \text{Max}_\text{Nbr}_A \leftarrow \text{Maximum neighbors Node } A \text{ can have} \)

Step 4. \( \text{Min}_\text{Nbr}_A \leftarrow \text{Minimum Neighbors Node } A \text{ should have} \)

Step 5. \( S \leftarrow \text{Stability Value} \)

Step 6. If \( \text{Nbr}_A < \text{Min}_\text{Nbr}_A \) then

Step 7. Return 0

Step 8. Value \( \leftarrow 0 \)

Step 9. \( \forall X \in \text{Nbr}_X \text{ do} \)

Step 10. Value \( \leftarrow \text{Value} + \frac{C_x}{\text{Nbr}_x} \)

Step 11. \( S \leftarrow \frac{\text{Value}}{C_x} \)

Step 12. If \( S > 1 \) or \( \text{Nbr}_A > \text{Max}_\text{Nbr}_A \) then

Step 13. \( S \leftarrow 1 \)

Step 14. Return \( S \)

**Algorithm 4.1: Level of Stability**

While dropping the neighbors, the degree of the dropping node is also taken into consideration in order to avoid dropping connections of poorly connected nodes in favor of highly connected ones. The process of
neighbor adaptation will continue until a stable state is reached at each node. The neighbor elimination process is described in Algorithm 4.2. The idea behind use of capacity is that a node with capacity $C$ will forward approximately $C$ messages per second at its full load and it needs adequate outgoing capacity in all its neighbors to maintain that load. Additionally, several other techniques may be used to compute the stability levels such as calculating the load on the neighbors network locality etc.

For simplicity, this system implementation uses the capacity of the nodes and degree to compute the stability level. Once the node is fully stable, it will still continue to perform the neighbor reconfiguration process checking its stability levels for every $T$ seconds by keeping in mind the mobility of nodes.

**Step 1.** $C_i \leftarrow$ Capacity of Node $i$

**Step 2.** $Nbr_A \leftarrow$ Number of Neighbors of $A$

**Step 3.** $\text{Max}_Nbr_A \leftarrow$ Maximum neighbors Node $A$ can have

**Step 4.** $\text{Min}_Nbr_A \leftarrow$ Minimum Neighbors Node $A$ should have

**Step 5.** If $Nbr_A + 1 \leq \text{Max}_Nbr_A$ then

**Step 6.** Accept $B$; return

Else

**Step 7.** Candidates $\leftarrow x \forall X \in Nbr_X$ such that $C_x \leq C_B$

**Step 8.** If NotExists then

**Step 9.** Reject $B$
Step 10. Return.

Step 11. $M \leftarrow$ Neighbor with highest degree from candidates

Step 12. If ($C_B > \max(C_X \forall X \in \text{Nbr}_X))$ Or

Step 13. $(\text{Nbr}_A > \text{Nbr}_B)$ then

Step 14. Drop $M$; Accept $B$

Else Reject $B$.

Algorithm 4.2: Neighbor elimination

The addition of new neighbors is depicted in Figure 4.2. The existing network contains the nodes A, C, D, E and F. Now, node B sends a request to A to join as neighbor and then A will add B as its neighbor if the room exists. If there is no room, then it checks all the capacities and degrees of its neighbors then decides the request of new node. Among the neighbors of A if there exists a node with lower capacity than B and with higher degree, then A will add B as its neighbor and drop the lower capacity node. In the Figure 4.3, if the capacity of Node D is lower than B, then B will be accepted as neighbor since B has higher degree to continue its operations.

Fig: 4.3, Adding New Neighbor
The nodes maintain an index for the data stored in each of its neighbor nodes. The indices are exchanged, when neighbors establish connections to each other. These indices are useful for query response for not only its data but also its neighboring data. When the neighbor leaves, the system or the topology is updated and the index information for that neighbor node is removed.

4.5 Search Protocol

Unlike Gnutella, this system uses the biased random walk for searching the data items in the P2P network. Biased random walk is the method of forwarding the incoming query to highest capacity neighbor instead of forwarding to a random neighbor of all the neighbors.

In order to limit the random walk at some point of time, the TTL (Time to Leave) values are used there by limiting the number of hops a query can progress. To avoid redundant paths, the nodes store the QID (Query Identifier) of the query and the neighbors to which it has already forwarded the query, which reduces the chance that a query is forwarded twice in the same path.

The query is resent and forwarded until the minimum number of responses to the queries that are received from the neighboring nodes. Since a node can respond to not only its own data but also its neighbors, the response is made with the identification of the node having the data. The search protocol is depicted in the Figure 4.3 and it shows how the
query is forwarded using the biased random walk where the bias is towards the high capacity nodes, which are represented using larger circles.

Fig: 4.4, Query Search

In the Figure 4.4, Node X will generate the query, which is forwarded to the neighbor with higher capacity, i.e. D. Node D intern will forward the query to its highest capacity neighbor i.e., C. In this manner, the query is forwarded to A and then to B. Since the query is reached to many high capacity nodes, there is a high probability that many responses will be generated by these nodes. And also since the nodes can also reply to the data present in its neighbors, the high capacity nodes can provide many responses compared to other nodes.
4.6 Proof of Correctness

**Theorem 4.1:** For the given underlay network $U(N,E)$ an Overlay network $O(N,E)$ can be formed that chooses an optimal set of neighbors taking hop count into account.

**Proof:** We prove this by contradiction. Assume that the set of neighbors chosen by the nodes are optimal when hop count is not taken into account. Without considering the hop count, the random selection of neighbors for a node in Gnutella, that only considers the traditional distance measure RTT (Round Trip Time) result in nodes with large distances.

Let us consider distance estimation for the given nodes $a$ and $b$ without taking hop count into account and considering RTT be $d_{\text{random}}(a,b) = x$, where $x = \{1, 2, 3, ..., N\}$, where $N$ is the longest hop distance, which can occur between two nodes in the network.

Since the nodes are chosen randomly the hop distances can be in the range $1: N$, it will not produce optimal neighbor selection. Thus we consider hop count between the nodes taking RTT value from one node to another reducing the distance and making the nodes optimal in the network i.e., $d_{\text{hop}}(a,b) \propto \text{hop limit}$. This therefore proves that the nodes are dependent on the hop limit rather than the RTT distance. This contradicts our assumption. Hence the theorem is proved.
**Theorem 4.2:** Given an underlay network \( U(N,E) \) during the overlay network \( O(N,E) \) formation, each node chooses the optimal set of neighbors where the hop count between the nodes is 3.

**Proof:** Let us assume the neighbor selection for a node is made by considering the hop count as 2. The hop count for distance estimation is \( x_{\text{random}}(i,j) = a \), where \( a = \{1,2,3,\ldots,N\} \), where \( N \) is the longest hop distance, which can occur between two end nodes in the network. Since the nodes are chosen with hop length 2, many isolated nodes remain in the network failing to produce optimal neighbor selection. Hence the given overlay network \( O(N,E) \) chooses the neighbors based on hop count and average distance between the nodes. Average Distance in an overlay with \( p \) nodes can be equated as follows.

\[
AD(G) = \frac{1}{p(p-1)} \sum_{i=1}^{p} \sum_{j=1, j \neq 1}^{p} v_i v_j \ldots \text{ Eqn. (4.2)}
\]

Where \( v_i v_j \in V \) denote the shortest distance between the nodes \( V_i \) and \( V_j \).

In [63] the hop limit considered as 2 i.e., any node can reach other node in an average of 2 hops, which cannot cover all the nodes. If the hop count is assumed as 4, the distance between nodes will become higher. Hence the delay will be higher, which can reduce the system performance and also increases the Average Query Response time. Consequently in between 2 & 4 we choose the hop length as 3. Hence the theorem is proved.
**Theorem 4.3:** For an overlay network $O(N,E)$ the usage of stability metric of a node can balance the load in the network effectively.

**Proof:** Let’s assume that the overlay network $O(N,E)$ formed without considering the stability level produces a network which balances the node efficiently. Without taking the stability metric into the account, each node can have both lower capacity nodes and higher capacity nodes as neighbors.

For instance, consider the following network containing nodes with different capacity values. Imagine the capacity on the node D is 10, if a lower capacity node is connected to large number of higher capacity nodes, then the lower capacity node cannot forward and receive the packets at the same rate. By considering the stability metric, a lower capacity node will not be connected to too many higher capacity nodes.

![Network Capacity Values](image)

Fig: 4.5, Network Capacity Values
The node capacities in the network shown in Figure 4.6 will be dispersed to the nodes with higher capacity levels. By using the stability values the neighbors of the nodes will be concentrated on their own level. One level higher to itself and the other level lower. A 100 capacity node will have most of its neighbors from 10 capacity, 100 capacity and 1000 capacity nodes.

When nodes are organized in this manner, the load will be dispersed from lower capacity to higher capacity nodes. Unlike Gnutella the query search process in this system uses biased random walks, which guided toward higher capacity nodes so, the lower capacity nodes do not have much stress.

### 4.7 Performance Evaluation

This section provides the details of the simulation setup and the results that support the improvements of performance of the system. First, each node in the system is assigned a capacity value which represents the number of messages it can accept per unit time. The capacities are assigned based on Gnutella distributions derived from [5].
In addition the nodes are given a query generation rate value which represents the number of queries that the node generates per unit time.

In order to evaluate the effectiveness of this technique, we compare these simulation results with that of Gnutella 0.4 and Gnutella 0.6. Each node in the Gnutella 0.4 system is modeled with 4 neighboring nodes. In the Gnutella 0.6 each ordinary peer has 3 ultra-peer connections and each ultra-peer has 10 peers as neighbors. The node in this technique is configured with minimum of 3 neighbors and the maximum is based on the capacity of nodes. On an average, the nodes in the bandwidth aided systems have 20 neighbors. The metrics taken into account for evaluation are as follows:

1. Maintenance overhead
2. Query delay
3. Query success rate

These metrics are considered for all the three networks and the results are represented in the following graphs.

Figure 4.7 depicts the maintenance overhead in the network. It shows that the Bandwidth aided system has higher maintenance overhead at the node, as the nodes have to maintain the cache and indices for all the neighbors in the network. Table 4.2 shows the comparison of existing and proposed techniques with respect to the messages per second, per node in terms of time period.
Table 4.2: Maintenance overhead

<table>
<thead>
<tr>
<th>Time(hours)</th>
<th>Messages/Second/Node</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gnutella 4</td>
</tr>
<tr>
<td>0.5</td>
<td>0.3</td>
</tr>
<tr>
<td>10</td>
<td>0.2</td>
</tr>
<tr>
<td>20</td>
<td>0.18</td>
</tr>
<tr>
<td>30</td>
<td>0.17</td>
</tr>
<tr>
<td>40</td>
<td>0.16</td>
</tr>
<tr>
<td>50</td>
<td>0.15</td>
</tr>
<tr>
<td>60</td>
<td>0.14</td>
</tr>
</tbody>
</table>

Fig: 4.7, Maintenance Overhead
Table 4.3 shows comparison of Gnutella 4 and Gnutella 6 with proposed technique Bandwidth-Aided topology with respect to the Query delay.

<table>
<thead>
<tr>
<th>Time(hours)</th>
<th>Query Delay (ms)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gnutella 4</td>
<td>Gnutella 6</td>
<td>Bandwidth-Aided</td>
</tr>
<tr>
<td>0.5</td>
<td>25000</td>
<td>21000</td>
<td>19950</td>
</tr>
<tr>
<td>10</td>
<td>26000</td>
<td>22000</td>
<td>20000</td>
</tr>
<tr>
<td>20</td>
<td>26500</td>
<td>22000</td>
<td>17000</td>
</tr>
<tr>
<td>30</td>
<td>27000</td>
<td>23000</td>
<td>15500</td>
</tr>
<tr>
<td>40</td>
<td>27000</td>
<td>24000</td>
<td>14500</td>
</tr>
<tr>
<td>50</td>
<td>27000</td>
<td>23000</td>
<td>13000</td>
</tr>
<tr>
<td>60</td>
<td>27500</td>
<td>22000</td>
<td>18000</td>
</tr>
</tbody>
</table>

The drawback of higher maintenance overhead is compensated in the form of faster query results, as the nodes respond to the queries faster due to the presence of indices of neighbors, as a result the delay for queries is less which is depicted in Figure 4.8.
Table 4.4 and Figure 4.9 show the comparison of existing and proposed techniques with respect to Query success rate and time.

**Table 4.4: Query Success Rate**

<table>
<thead>
<tr>
<th>Time(hours)</th>
<th>Query Success Rate (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gnutella 4</td>
</tr>
<tr>
<td>0.5</td>
<td>0.4</td>
</tr>
<tr>
<td>10</td>
<td>0.3</td>
</tr>
<tr>
<td>20</td>
<td>0.25</td>
</tr>
<tr>
<td>30</td>
<td>0.25</td>
</tr>
<tr>
<td>40</td>
<td>0.25</td>
</tr>
<tr>
<td>50</td>
<td>0.3</td>
</tr>
<tr>
<td>60</td>
<td>0.3</td>
</tr>
</tbody>
</table>
To understand the distribution of load among the nodes, we consider the load of the nodes for each of capacity values. This is depicted in Figure 4.10 in which the node capacity of 100 nodes introduces higher overhead when compared to capacity 1 and 10 nodes but less than the capacity of 1000 nodes.

Since the nodes of bandwidth-aided network have higher in-degree, these will be often visited by the random walk which increases the success rate and in turn reduces the average message overhead on the network. A bandwidth-aided system overhead without queries is nearly twice that of Gnutella 0.6, but these are comparable with queries due to shorter random walks.
4.8 Summary

The Bandwidth-Aided topology awareness system is a highly scalable unstructured P2P system which improves the performance of the Gnutella network by considering the peer bandwidth for the topology formation and distributing the load on the high capacity nodes. The system makes the search more scalable and reduces the query response time which resulted in drastic improvement in the query success rate with small overhead. The future work for this system would be considering other underlay properties of the peer while constructing the overlay.