SHP: A HIERARCHICAL PROTOCOL TO IMPROVE PERFORMANCE OF PEER-TO-PEER SYSTEMS

Guruprasad Khataniar[1] and Diganta Goswami[2]

[1]Department of Computer Engineering, Assam Engineering Institute Guwahati
drkhataniar@gmail.com
[2]Department of Computer Sc. & Engineering, Indian Institute of Technology Guwahati
dgoswami@iitg.ernet.in

ABSTRACT

Extensive application of Peer-to-Peer systems demands an effective solution for efficient query processing, handling of churn rate, load balancing and maintenance of healthy arrangement of nodes for the improved response of the system. Several key based systems offer an efficient solution for query processing but suffer from transient node population. We present a structured hierarchical Peer-to-Peer system to reduce overhead caused by churn rate. The simulation result shows better performance than existing structured systems while executes point query and range query.

KEYWORDS

Structured P2P, Hierarchical DHT, Churn rate, Point Query, Range Query

1. INTRODUCTION

Over the years Peer-to-Peer (p2p) systems have gained increased popularity because it can be used to exploit the computing power, resources and storage of a large population of networked computers in a cost-effective manner. A p2p network exploits diverse connectivity among participants in a network and makes efficient use of the collective bandwidth of network users in contrast to conventional centralized resources where a relatively low number of servers provide the core value to a service or application. Although peer-to-peer networking is mostly applied to file-sharing applications like Napster [6], BitTorrent [9], and KaZaA [6][14], a substantial amount of research projects have been launched overtime to exploit the new possibilities provided by this new networking paradigm. Services like video streaming (live and on-demand), Voice-over-IP, etc. and distributed applications like SETI@home [4] are some of the examples. According to several Internet service providers, more than 50% of Internet traffic is due to p2p users [30]. Sandvine [25] presented statistics of Internet traffic in North America in 2008. In aggregate 44% consumer broadband traffic was p2p, a slight increase from 2007 when 40.5% of traffic was p2p; web browsing was 27% and streaming was 15%. Web browsing, p2p, and streaming constituted much of the downstream traffic, while p2p dominated upstream with 75% of the upstream traffic.

The concept of p2p networking is not new and is started in the late 1960s with the starting of networking (ARPANET), which was intended for file sharing among users without the concept of client and server. Every host was being treated equally and one could therefore call this network a p2p network. But, only after 1999, with Napster [6], the modern p2p systems came to the picture. A p2p system normally uses the concept of overlay, a logical layer on top of the physical network, for message delivery between peers. In a client-server system, the server acts as the

DOI : 10.5121/ijp2p.2012.3501

central hub to provide services to the clients. Server is the central entity and only provider of service and content. Usually servers are supposed to have much higher performance/capacity compared to clients. But in p2p system no distinction is laid down for content provider (server) and content requester (client), where resources are shared between the peers and resources can be accessed directly from other peers. The first generation of p2p networking started with the centralized concept with a central server. However, contrary to the client-server approach this server only stores the indexes of peers where the actual content is available, thus greatly reducing the load of that server. Every peer is connected to the centralized lookup server, to which it can issue requests for content matching the keywords stated in the request. This concept was widely used and became well known due to Napster [6]. No file could be found in Napster if the central lookup table were not available. Only the file retrieval and the storage are decentralized. Thus, the server leads to bottleneck and single point of failure. The computing power and storage capabilities of the central lookup facility grow proportional to the number of users, which also affects the scalability of this approach. Thereafter, came the pure p2p without the concept of central server. The queries were flooded over the network. Gnutella 0.4 [33][20] and Freenet [8] are examples of pure p2p model. The major concern in this model is the high bandwidth consumption due to flooding of queries. An efficient way to reduce the consumption of bandwidth is the introduction of hierarchies. To avoid traffic overloading, schemes like Gnutella 0.6 [21][26], KaZaA [6], JXTA [18] etc. were introduced as second generation of p2p. Some of nodes are treated as special nodes known as superpeers with high capabilities. The superpeer acts as the dynamic central entity for joining and processing queries.

All these above discussed systems use unstructured overlays, which usually exhibit linear search complexity at its best. The unstructured overlays are easy to implement but have inefficient routing and inability to locate rare objects. In a structured overlay nodes are arranged in a systematic way and key ids are placed in specific location. It is the most recently stated overlay, where search is deterministic in nature. In a structured system, each data item is assigned an identifier, id, a unique value from the address space. Then it stores a file at the node responsible for the portion of the address space which contains the identifier. A data item can be retrieved from the system with a complexity of O(log n), where n is the number of nodes in the system. The underlying network and the number of peers in structured approaches can grow arbitrarily without impacting the efficiency of the distributed application. Structured systems like Chord [32][31], CAN [23], Pastry [27], Tapestry [37][36], P-Grid [1][2], etc. employ a globally consistent function to map a file name with key id, hence the searching is deterministic. But a structured p2p system assumes that all the peers have same priorities and features, which is infeasible in practice. From Gnutella analysis [35][3] it is found that all the peers in p2p systems do not contribute homogeneously. There may be various computers with heterogeneous specifications, different sharing capacities, and variable life-time in the system. Therefore, hierarchical system is introduced to develop a new category of p2p system [6]. The major advantages of a hierarchical system are scalability, efficient lookup, easy maintenance of nodes etc.

1.1 Defining Peer-to-Peer

The Internet based applications require scalability, security, reliability, flexibility and quality of service. The client-server based applications are unable to satisfy the requirement of Internet as it is growing exponentially in terms of users and resources. Peer-to-Peer has emerged as a new paradigm for communication on the Internet that can address some of these issues. Different definitions have been found in literature based on system architecture and configurations. Some of these are stated below.

- Distributed network architecture may be called a Peer-to-Peer (p2p) network, if the participants share a part of their own hardware resources (processing power, storage

capacity, network link capacity, printers, etc.). These shared resources are necessary to provide the service and content offered by the network (e.g. file sharing or shared workspaces for collaboration). They are accessible by other peers directly, without passing intermediary entities. The participants of such a network are thus resource (service and content) providers as well as resource requesters (servent-concept) [28].

- Peer-to-peer systems and applications are distributed systems without any centralized control or hierarchical organization, where the software running at each node is equivalent in functionality. A review of the features of p2p applications yields a long list: redundant storage, permanence, selection of nearby servers, anonymity, search, authentication, and hierarchical naming [31].
- Peer-to-peer systems can be characterized as distributed systems in which all nodes have identical capabilities and responsibilities and all communication is symmetric [27].
- Peer-to-Peer networking refers to a class of systems, applications and architectures that employ distributed resources to perform routing and networking tasks in a decentralized and self organizing way [39].
- Peer-to-Peer systems are distributed systems consisting of interconnected nodes able to self organize into network topologies with the purpose of sharing resources such as content, CPU cycles, storage and bandwidth, capable of adapting to failures and accommodating transient populations of nodes while maintaining acceptable connectivity and performance, without requiring the intermediation or support of a global centralized server or authority [17].
- A p2p is a self-organizing system of equal, autonomous entities aims for the shared usage of distributed resources in a networked environment avoiding central services [30].

From the above one can define P2P as *an autonomous, self-organized, scalable distributed system with shared resource pool without a single point of failure in which all nodes have identical capabilities and responsibilities and all communications are normally symmetric; and where main characteristics of the participants are decentralized resource usage and decentralized self-organization.*

1.2 Overlay

An overlay is a logical abstraction of the physical (underlying) network needed by distributed applications. The nodes for the overlay network and connections are decided by the application. The routing algorithm used in underlying network is different from that used by overlay network. Figure 1 illustrates the relation between the overlay and the underlay. Here if peer ‘A’ sends information to peer ‘C’. Then actual data transfer occurs from ‘a’ to ‘g’. Hence for a p2p, user actual data transfer is transparent in its application.

1.3 Classification

The p2p systems are classified according to logical arrangement of the nodes. The prime themes for classification of overlays are the relationships of overlay nodes and the topology used. Broadly two categories are proposed in different literatures: unstructured and structured. Ranjan et al. [22] investigate various decentralized resource discovery techniques primarily driven by p2p network model in 2006. They present a summary of grid resource discovery, resource taxonomy with focus on computational grid paradigm, p2p taxonomy with focus on extending the structured systems for indexing d-dimensional grid resource queries and classification of the surveyed approaches based on the proposed p2p taxonomy. The classification of p2p systems is as follows.

- Unstructured: deterministic and non-deterministic
- Structured: DHT-based and non-DHT-based

- Hybrid and
- Hierarchical

Unstructured deterministic systems are those where a look-up operation is successful within predefined bounds. Systems including Napster [6], BitTorrent [9], JXTA [18] fall into this category. In these systems, the object lookup operation is centralized while download is decentralized. In an unstructured non-deterministic system the query may not be successful even the required data is present in the system. The systems including Gnutella [33], Freenet [8], FastTrack [15] and KaZaA [6][14] offer non-deterministic query performance.

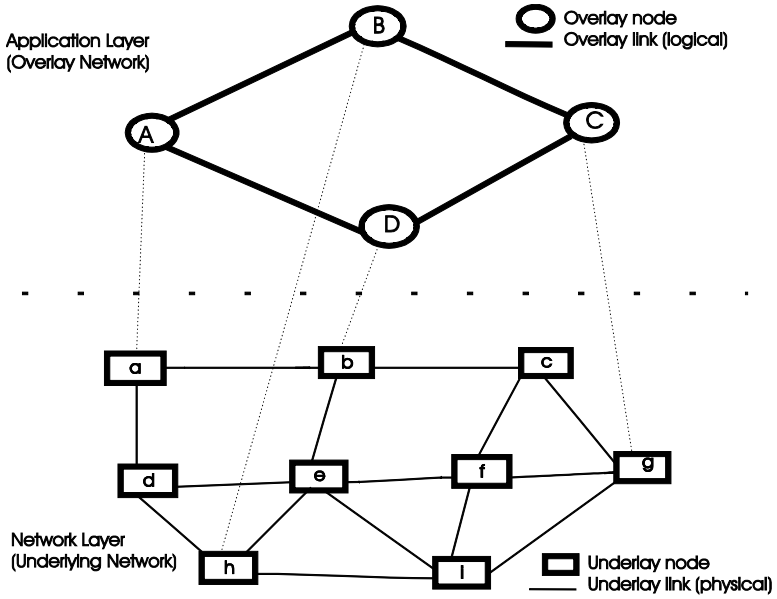

Figure 1: Illustrating Overlay and Underlay

Structured deterministic systems offer deterministic query search results within logarithmic bounds on network message complexity. Peers in DHTs maintain an index for O(log(n)) peers, where n is the total number of peers in the system, e.g. Chord [31] [32], CAN [23], Pastry [27], Tapestry [37] [36], P-Grid [1] [2] .

There are other structured overlays where a standard graph topology is used instead of applying randomizing hash functions for organizing data items and nodes. System like Mercury [5] organizes nodes into a circular overlay and places data contiguously on this ring.

In recent developments, new generation p2p systems have evolved to combine the advantages of both unstructured and structured p2p networks. This class of systems are referred as hybrid. Structella [7] is one such p2p system that replaces the random graph model of an unstructured overlay with a structured overlay, while still adopting the search and content placement mechanism of unstructured overlays to support complex queries.

With increasing pervasive deployments of various p2p overlay architectures, hierarchies are used to connect various nodes in the systems like RChord [16], DR-Chord [29], Chord2 [13], PChord [12], HPS [11], HIERAS [34]. The hierarchical system reduces the average number of peer hops in a lookup query and reduces the query latency when the peers in the same group are topologically close.

2. SYSTEM MODEL

The aim of this paper is to describe a Structured Hierarchical overlay Protocol (SHP) to improve the overall performance of the system. The overlay that is proposed should be able to handle both point and range queries and also should be capable of handling transient node population in the system with inherent hierarchical property. One of the major applications of p2p systems is content distribution. Millions of nodes may join a p2p network to share resources spread over the world. It is detrimental to the performance of the system if all the nodes are considered identical. Nodes in a network may differ in terms of available bandwidth, processing power, stability, privacy & security, reliability, amount of data shared and storage capacity.

It has been observed that all the nodes available in a p2p system are not homogeneous. An analysis [3] has shown that 70% of the Gnutella users share no files and 90% of users do not response to the query. Another Gnutella analysis [38] shows that 7% peers share more files than all other peers and 47% queries are responded by the top 1% peers. Definitely, temporary nodes have influenced the analysis due to their short span of life in p2p systems. For example, mobile devices have limited bandwidth and power, variable capacity and asymmetric links. These links are error-prone and hence network congestions are more, so it is difficult to achieve guaranteed quality of service (QoS). Eventually the presence of temporary nodes cannot be expected in p2p networks for long time. To give importance to nodes according to their capabilities, nodes are taxonomized into three categories as follows.

i. Temporary nodes
ii. Stable nodes or peers and
iii. Fully-stable nodes or superpeers

i. Temporary nodes: A newly joined node to the network is called a temporary node. It does not have own routing table and hence no corresponding entries in the DHT. These nodes connect at the last level of hierarchy and may be disconnected after accessing resources from p2p network. A temporary node knows the id of the contacted node and access information from the p2p network.

ii. Stable node: A temporary node becomes a stable node or peer after satisfying stability period T_{avg} as shown in Figure 2. T_{avg} is the observed average life-time of a node in the system. Since this parameter cannot be calculated beforehand so it is derived empirically at the time of instantiation of the network. Stable nodes have their own routing table known as Peer Routing Table (PRT). The structure of PRT is same as finger table used in Chord. Table 1 shows the PRT for a node (node $id = 0$) of the Group as shown in Figure 4. A stable node has information about its group, its successor and predecessor.

iii. Fully-stable node: A stable node gets promoted to fully-stable node or superpeer after completing the duration of T_{stab} and satisfying some other constraints like processing power, storage capacity, etc. T_{stab} is a empirically set initial parameter which determines the stability period of a fully-stable node. The value of T_{stab} is chosen sufficiently higher than T_{avg} so that network will not become a complete mesh. Hence, this can be used to control the number of fully-stable nodes in the system. These nodes are assumed to have ideal characteristics, higher availabilities and predicted to be available in future for long duration compared to other nodes. A fully-stable node contains PRT and Group Routing Table (GRT). GRT, as shown in Table 2, stores the addresses of all fully-stable nodes of all other groups in a network.

Therefore, inter-group lookup is reduced to *O(1)*. Fully-stable nodes act as gateways for inter-group communication.

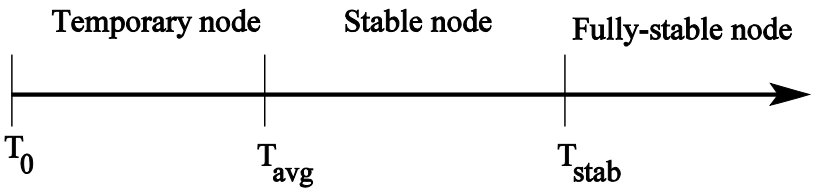

Figure 2: Categorization of nodes.

Table 1: PRT of node *id = 0* as shown in Figure 4

Start	Successor
0 + 1	5
0 + 2	5
0 + 4	5
0 +8	10
0+ 16	20

3. SHP ARCHITECTURE

Figure 3 shows the high level architecture of SHP where nodes are organized into groups. Number of nodes in a group may vary from one group to another. For example, some groups may have thousands of nodes while others may have only hundred. Each group maintains a ring structure using the well-known successor-maintenance algorithm [10] like Chord. Within a group, nodes are relatively close to each other and hence intra-group latency is sufficiently reduced. At the top level of the hierarchy, joining and leaving of nodes is rarely possible, because fully-stable nodes are relatively stable and expected to be available for long time. As it is evident from the Figure 3, only stable nodes are part of the Chord ring structure (which is considered to be the second level of the hierarchy) in a group. Temporary nodes are in the last level of the hierarchy. The first level of the hierarchy maintains a mesh structure which consists of one of the fully-stable node from each Chord ring. Temporary nodes may join or leave the network at their will without affecting the performance of the system.

3.1 Bootstrapping

It is assumed that first g nodes that join the p2p network trivially qualify for fully-stable nodes, where *g* is the initial number of groups. Fully-stable nodes maintain the time-stamp of newly joined nodes and form a group. Number of fully-stable nodes in a group may increase further to G_{max}, where G_{max} is the maximum number of nodes in a group. When number of nodes in a group becomes more than G_{max}, the group is divided into two groups choosing one of the stable nodes as fully-stable node.

4. JOINING AND LEAVING OF NODES

Any node that wants to join a p2p network joins as a temporary node in the last level of hierarchy. To reduce the intra-group lookup, a node first sends k ping messages to different landmarks and

finds the average Round Trip Time (RTT) [24]. It contacts the group which is closest to it based on the RTT. As the node joins as a temporary node, it does not have any routing table up to T_{avg} time units. When a temporary node crosses the T_{avg} time duration, it is expected that it has high probability to remain active in the network. So, data shared by a stable node is available in the p2p system with high probability. Hence, joining of a stable node can smoothly be handled by the following three steps.

- Initialization of PRT of newly upgraded stable node.
- Update PRTs of existing stable nodes.
- Transfer of *keys* from its successor.

Leaving of a stable node can be handled by the following two steps.

- Transfer of *keys* from leaving node to its successor node.
- Update PRTs of existing stable nodes.

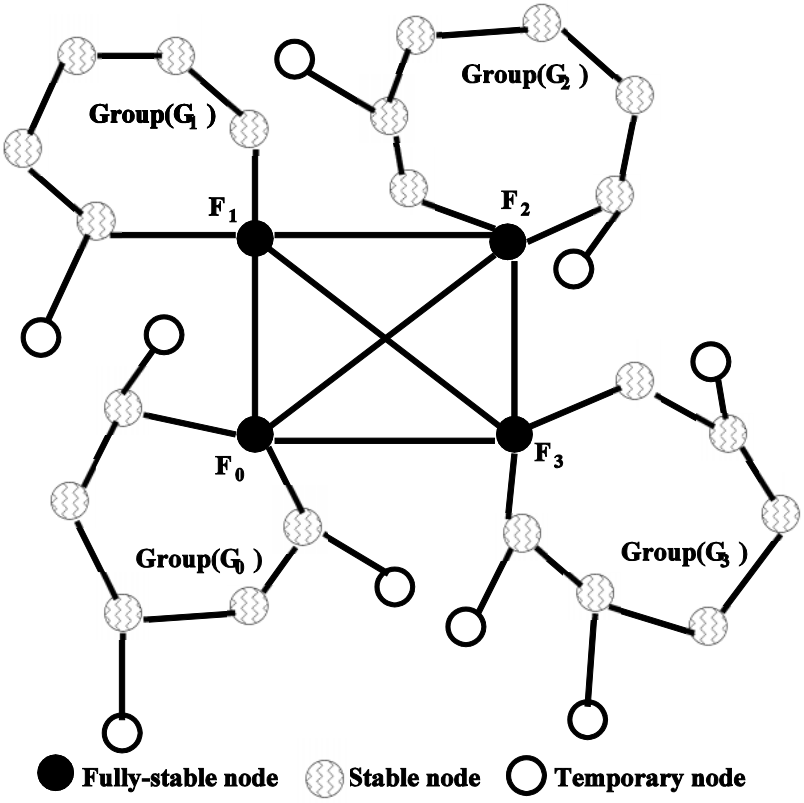

Figure 3: Illustrating High-level architecture of SHP

4.1 Effect of Temporary Node

Since a temporary node maintains no routing table, it may easily leave the p2p network without informing any other nodes.

Let n be the number of nodes in the system. Hence, (n/g) is the average number of nodes in a group where g is the number of groups in the system. Let λ_j and λ_l (where $\lambda_j \geq \lambda_l$ and λ_j , $\lambda_l \geq 0$) be the rate of joining and leaving of temporary nodes, respectively. Let M_{ccp} and M_{shp} be the number

of updates required in time span T_{avg} to repair the routing table in conventional Chord protocol and SHP, respectively.

Chord handles the joining or leaving event of nodes with no more than $O(\log n)^2$ updates with high probability. Hence, it is assumed that $(\log n)^2$ updates are required to handle joining / leaving operation. Therefore,

$$M_{ccp} = (\log n)^2 X\ \lambda_j X\ T_{avg} + (\log n)^2 X\ \lambda_l\ X\ T_{avg}$$

$$= (\lambda_j + \lambda_l) X\ (\log n)^2\ X\ T_{avg}$$

In SHP, temporary nodes do not have entries in routing table. Since nodes which are available beyond T_{avg} time affect the routing tables of existing stable nodes and fully-stable nodes, the number of updates required to repair the routing table in time span T_{avg} is given by

$$M_{shp} = (\lambda_j - \lambda_l) X\ (\log((n/g) - ((\lambda_j - \lambda_l) X\ T_{avg})/g)^2\ X\ T_{avg}$$

If the rate of leaving nodes is too frequent i.e. $\lambda_j \approx \lambda_l$, SHP saves $2\lambda_j\ X\ (\log n)^2\ X\ T_{avg}$ updates in time span T_{avg}.

Table 2: Group Routing Table.
Here [id_a – id_b] denotes the key interval values in the group G_0 and so on.

Group	Key interval	Fully-stable node vector
G_0	$id_a - id_b$	F_0
G_1	$id_c - id_d$	F_1
G_2	$id_e - id_f$	F_2
G_3	$id_g - id_h$	F_3

Now considering the average case scenario as leaving of half of the nodes from the system in T_{avg} i.e. $\lambda_l = (\lambda_j)/2$. Then

$$(M_{shp}/ M_{ccp}) = [(\lambda_j/2) X\ (\log((n/g) - ((\lambda_j/2) X\ T_{avg})/g)^2\ X\ T_{avg}]/[(3\lambda_j/2)\ X\ (\log n)^2\ X\ T_{avg}]$$

$$= [(\log((n/g) - ((\lambda_j X\ T_{avg})/2g))^2]/[3\ X\ (\log n)^2]$$

Where $g \geq 1$ and $T_{avg} \geq 0$

Hence $(M_{shp}/ M_{ccp}) \leq (1/3)$

Therefore, SHP saves at least 66.67% updates in average case to repair the routing table due to frequent joining and leaving of temporary nodes.

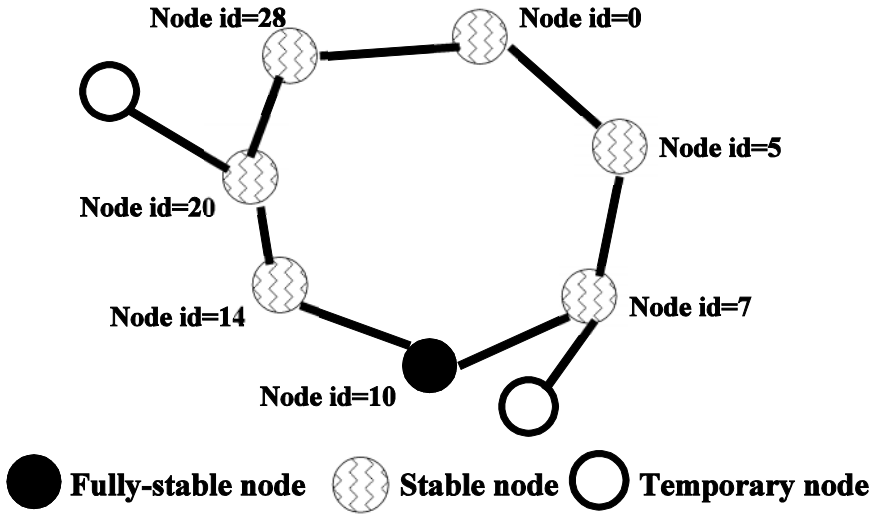

Figure 4: A Group with 5-bit node id.

4.2 Effect of Stable Node

The joining/leaving of a stable node in a group needs $O((log(n_s + n_f))^2)$ updates to repair the routing table in contrast to $O((log(n_t + n_s + n_f))^2)$ in Chord protocol, where n_t, n_s and n_f are the number of temporary, stable and fully-stable nodes in a group, respectively.

4.3 Effect of Fully-stable Node

The joining of a fully-stable node needs update of GRT. Thus, number of updates require during joining of a fully-stable node is $O(N_f)$, where N_f is the total number of fully-stable nodes in the system. The assumption is that leaving of a fully-stable node is very rare as this node is most powerful and highly stable. Even if a fully-stable node leaves the system, the cost to repair the routing table is $O((log(n_s + n_f))^2)$ and $O(N_f)$ for PRT and GRT, respectively.

5. QUERY PROCESSING

The SHP supports point query and range query. A single data item is searched in point query, where the key is hashed to find the identifier for the file. The node responsible for that file is located and the file is retrieved. A data item may be found within the group (intra-group) or outside the group (inter-group). A range of data items are searched in range query, where corresponding node is found for each data item and retrieved.

5.1 Range Query

DHT-based system like Chord provides an efficient solution for point queries. But uniform distribution of hash function disturbs the data relationship and does not have support for range query.

To handle range queries in SHP, complete data space is divided into some regions. Each group is associated with a part of the data space. At the time of startup number of regions in the data space is equal to the number of groups. A fully-stable node holds information about the group and hence it knows the part of data space corresponding to particular group.

Let the complete key space be [AAA - DDD] with three groups in the system. The division of key space to groups is [AAA - BBB), [BBB - CCC), [CCC - DDD). The format of the Group Routing Table is shown in Table 3.

Table 3: A GRT showing key space AAA – DDD

Group	Key interval	Fully-stable node vector
G_1	*AAA - BBB*	F_1
G_2	*BBB - CCC*	F_2
G_3	*CCC - DDD*	F_3

Internal to the group, key space is divided into intervals and each interval is assigned to one node. A node is allowed to hold more than one interval but an interval is never mapped to more than one node. Each node maintains the following information apart from the routing table which is used for routing.

- Predecessor Range Node: A node which contains the immediate previous key range is called as predecessor range node. For instance let A, B, C be three nodes which hold key lying in the range [AAA - BBB), [BBB - CCC), [CCC - DDD) respectively. Then A is called the predecessor range node of B.
- Successor Range Node: A node which contains the immediate next key range is called as successor range node. In the above example, C is the successor range node of B.
- Key Interval: The key space can be arranged in ordered manner and divided into several intervals. Each interval is known as key interval.

Table 4 shows the range table of node B as stated above.

Table 4: Range Table of node B

Key interval	Predecessor Range node	Successor Range node
BBB – CCC	*A*	*C*

5.2 Range Query Processing

Let the range of data maintained by a group be X_1 to X_2. Let the query range be from p to q. When a node gets a query, it checks whether the query falls in the region maintained by the group. If the query does not fit in the interval maintained by the group, then it is forwarded to one of the fully-stable nodes of that group which in turn forwards it to the respective group, where a desired data space is available. The following steps are used to process a range query.

i. If (p to q) falls in the range (X_1 to X_2) then the node runs the regular Chord algorithm to find the node holding data p. Then the query is forwarded to the successor range node till the range of data is obtained.

ii. If ($p < X_1$) and q falls in the range of the data maintained by the group, then the range [p-X_1) is forwarded to one of the fully-stable nodes for further processing. Step i is used for the range [X_1- q].

iii. If p falls in between the range and q is out of the data range maintained by the group, then for the range [p-X_2) step i is executed. For the range [X_2-q] the message is sent to the fully-stable node of the group for locating the desired group.

The message format for range query is as follows.

Query(hash(L_{key}), k_1, k_2, NodeID(id)), where L_{key} is the starting point of the interval containing the desired data range i.e. $L_{Key} = k_1 - (k_1 \% j)$; k_1 is lower bound of range of data; k_2 is upper bound of range of data; *id* is identifier of the source node and *j* is key interval size.

Assuming uniform distribution of nodes to all groups, lookup complexity of range query is given by $O(\log(n/g) + R/j)$, where *g* is the number of groups; *n* is the total number of nodes and *R* is the length of range query. The lookup complexity can be simplified as $O(\log(n/g))$.

5.3 Point Query

Suppose a node P_i of group G_i makes a query for key *k*. The fully-stable node F_i finds the responsible group for key *k*. If *k* is found within G_i, then it returns with complexity $O(\log n_i)$, where n_i is the number of nodes in group G_i. Otherwise F_i sends request to the fully-stable node where the key *k* belongs. This takes $O(1)$ lookup step, since every fully-stable node knows about fully-stable nodes of other groups. Suppose F_l of group G_l receives the request, then key *k* is retrieved with complexity $O(\log n_l)$, where n_l is the number of nodes in group G_l. Therefore, it is found that effective complexity to execute a point query is $O(\log (n/g))$.
The message format for point query is shown below.

Query(hash(L_{key}), k, NULL, NodeID(id)), where L_{key} is the starting point of the interval containing the desired data range i.e. $L_{key} = k - (k \% j)$; *k* is the key of data object; *id* is identifier of the source node; *j* is key interval size.

The first field generates the key *id* for key interval in which key *k* lies. The second field specifies exact key within that key interval. *NULL* in the third field indicates that it is a point query. Finally, the last field specifies the node *id* of the querier node (the node which makes query).

For example, let the key intervals be *[0-10), [10-20), [20-30), . . .* and a querier node makes query for the key *15*. Then $j = 10$ and $L_{key} = 15 - (15 \% 10) = 10$. Hence, *hash(10)* gives the key *id* for key interval *[10-20)* within which key *15* lies.

6 EXPERIMENTAL RESULTS

In this section, first we have given a brief outline of a simulator and its importance. Then we have shown the observations of experiments along with discussions.

6.1 Simulator

Any scientific community including the computer science community requires that the results of the models developed for specific purposes are tested so that reproducible results are provided. To achieve this, analytical solutions, simulations and experiments are the methods that are used. However, in the area of p2p, they give rise to a number of challenges.

The Analytical approach is one where a mathematical model of the system is examined. It can provide an analytical solution, but only if the model is simple. While this approach is being used in p2p research, as it requires a simple model, many of the complexities of real-world p2p systems have to be ignored.

In cases where applying analytical methods prove too complex, an alternative is to run experiments with the actual system. Peer-to-peer systems can consist of a large number of nodes

and any experiment on even a relatively small scale of a few thousand nodes would be impractical, or even impossible. Even it may be feasible to acquire the resources necessary to run such an experiment; there are other impracticalities that must be considered. Applying changes to an experiment such as modification of the protocol running at each node and the topology of the network would be difficult and certainly time consuming if the experiment were performed on a large scale.

Simulation techniques are used in most cases which provide a flexible framework to test the developed applications. Even though simulators have some disadvantages, they do address the impracticalities of mathematical analysis and experimentation. However simulations are not seen as strictly independent from the analytical approach and experimentation. If possible analytical approaches should be utilized and proved using a simulator, similarly simulation results should if possible be validated by experiments with the actual system.

S. Naicken et al. [19] discuss about DHTSim, P2PSim, Overlay Weaver, PlanetSim, PeerSim, GPS and Neurogrid. Most of these simulators are developed out of the necessity to fulfil a particular requirement. In general none of the simulators give a common platform to design and test any p2p structure. In particular, none of the available simulator provides any classification of nodes considering the heterogeneity that is inevitable in any computer network. Therefore it is difficult to implement the SHP model using these simulators which is based on the taxonomy of nodes. Therefore, it is decided to develop a new simulator (it is named as *shpsim*) to simulate SHP and Chord together and a relative comparison is done between two by observing the simulation results. A comparison of *shpsim* with some existing simulators is given in Table 5.

Table 5: Comparison of Simulators

	Oversim	OpenChord	PlanetSim	*shpsim*
GUI	Yes	No	Yes	Yes
Source Code	Open	Open	Open	Open
Network supported	Many	One	Few	Two
Finger Table	Viewable	Viewable	Viewable	Viewable
Real time simulation	Yes	No	No	No
Manual events possible	No	Yes	Yes	No

An *event file* is generated as input to the simulator. An event file consists of records for different events with the following fields.

Time: The time at which the event is generated.
Event: Nature of event i.e. joining /leaving of a node, searching of item(s) using point query and range query.

NodeID: Identifier of the node which has generated the event.
A few sample records in the event file is shown in Figure 5.

Time: 0; Event: Joining of a new node; NodeID:2

Time: 0; Event: Joining of a new node; NodeID:55

Time: 2; Event: Searching of a data item (point query);
Search key: 228; Source NodeID:81

Time: 3; Event: Joining of a new node; NodeID: 163

Time: 7; Event: Searching range of data item (range query);
Range key: 94 – 135; Source NodeID: 218

Time: 7; Event: Leaving of an existing node; NodeID: 46

Figure 5: A snapshot of records in the event file

6.2 Joining and Leaving of Nodes

When a node wants to join the system, it first contacts any fully-stable or stable node and joins as a temporary node. After time-stamp T_{avg} it executes the JOIN algorithm of Chord and joins the group at its specific location (as determined by the hash function). The joining and leaving process is implemented as in Chord, considering the group as a Chord ring. Since the number of nodes in a group is less than the total number of nodes in the system, the number of messages required for joining and leaving is less than Chord.

Figure 6 shows the comparison of messages required for joining of nodes in Chord and SHP for various sizes of network. For our experiment numbers of nodes are taken as 256, 512, 1024 and 2048. It is seen from the Figure 6 that a node in SHP takes less number of messages to join in the system in comparison to Chord and the same has been shown for different sizes of network.

Figure 7 shows effect of joining of nodes with different group sizes in SHP. The number of nodes in the system is taken as 2048 and number of groups considered as 8, 16, 32 and 64. It is found form Figure 7 that performance of SHP is better as the number of nodes in a group is more than the total number of groups in the system. But the efficiency of SHP is reducing when the total number of groups in the system is nearing or greater than the number of nodes in a group, even smaller than the Chord as shown in Figure 7-[C] and Figure 7-[D].

Figure 8 represents joining of nodes with varying T_{avg} value keeping size of network and T_{stab} value fixed. As the T_{avg} value increases the number of temporary nodes in the system increases, so messages required to join a node in SHP decreases as depicted in Figure 8.

Figure 9 represents joining of nodes with varying T_{stab} value keeping size of network and T_{avg} value fixed. As the T_{stab} value approaches T_{avg} value, the nodes in the system frequently upgrade to fully-stable node, which need to build Group Routing Table (GRT) and Peer Routing Table (PRT). It is observed that in all cases SHP takes less messages to join nodes as compared to Chord as shown in Figure 9-[A], Figure 9-[B], Figure 9-[C] and Figure 9-[D].

Graphs shown in Figure 10 indicate less updates required for leaving of nodes in SHP and Chord. Since leaving of a temporary node does not affect the SHP overlay, hence it is seen that for some

initial values leaving of nodes take zero message pass. Therefore, the leaving of nodes in SHP takes fewer messages in comparison to Chord which supports theoretical assumptions.

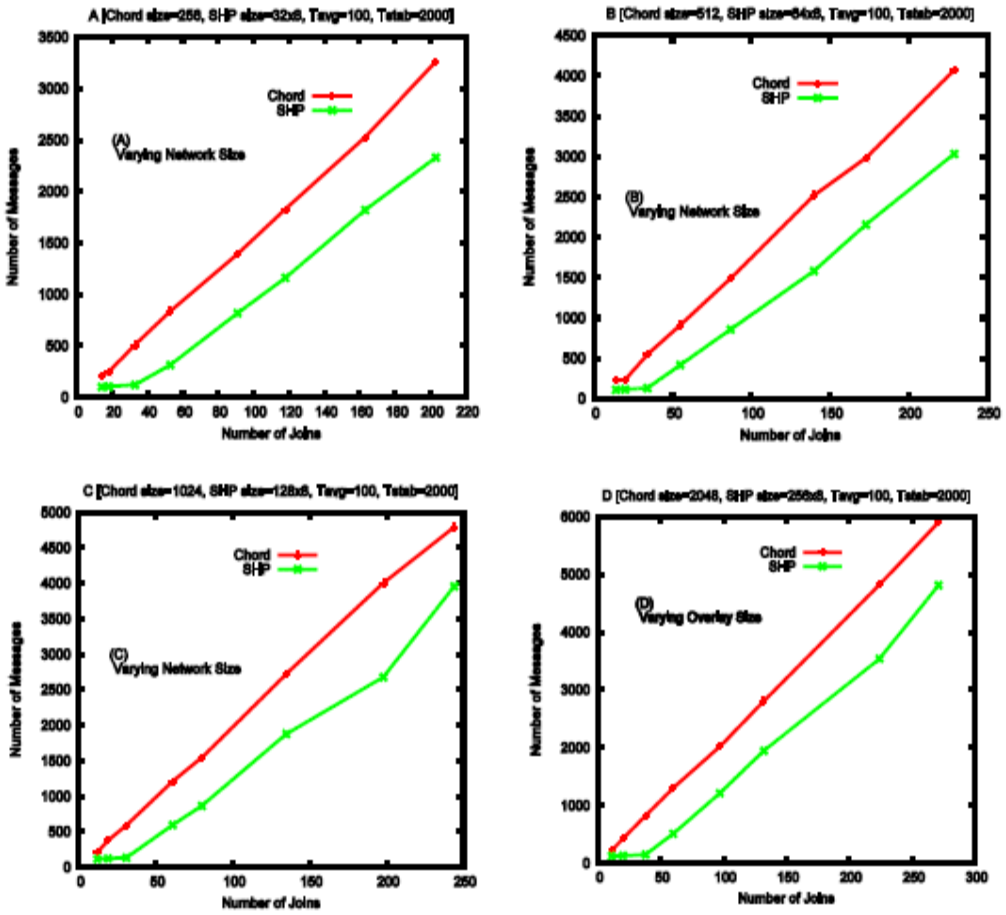

Figure 6: Graph showing number of messages required to join nodes in Chord and SHP with different network size

6.3 Query Processing

In SHP more than one key interval can be mapped to a node but a key interval cannot be partially mapped to more than one node. It is observed that number of hops required for point and range queries is less for SHP in comparison to Chord. In SHP, if a data item is found within the same group then it is found within *O(log (n/g))* steps; otherwise, it needs another *O(log(n/g))* steps. For range query difference in number of messages required is more since Chord doesn't support range query (i.e. each data item within a range is retrieved using individual point query), whereas in SHP it executes the query with first key of the range as discussed in section 5.1.

Figure 11 illustrates the number of hops required to execute point query for different network sizes. It is observed that execution of point query in SHP has improvement than in Chord.

Figure 12 illustrates the number of hops needed to execute range query for different network sizes. In all cases performance improvement is observed significantly for range query in SHP.

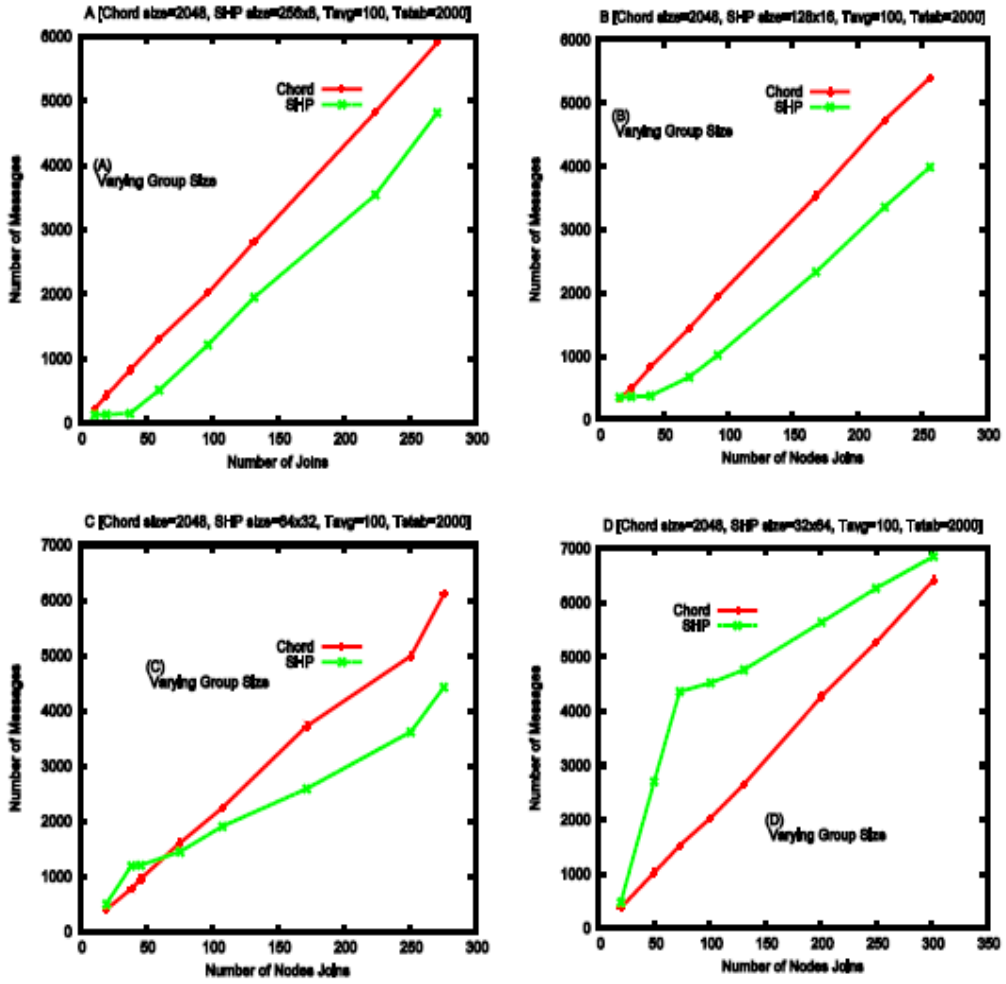

Figure 7: Graph showing number of messages required to join nodes in Chord and SHP with different group size of same network size

7 CONCLUSION

The primary goal of this paper is to propose some solutions to improve performance of peer-to-peer systems. Many peer-to-peer systems have rapidly become basic platforms for users to share information and resources over the Internet. A new overlay called structured hierarchical protocol is introduced in this paper. It is necessary to introduce hierarchies in the p2p system to improve overall performance. We have classified nodes according to their capabilities and placed them in different hierarchies. Hence the system becomes capable of handling high churn rate. Another important feature of this system is that unlike existing structured systems, it can efficiently execute both point and range queries. The Chord is considered as basis for comparison since Chord is well-defined and its functionalities has already been proved.

ACKNOWLEDGEMENT

We would also like to thank All India Council for Technical Education (AICTE) for the grant under Research Promotion Scheme No. 8023/RID/RPS-5/(NER)/2011-12 which motivated us to carry out this work.

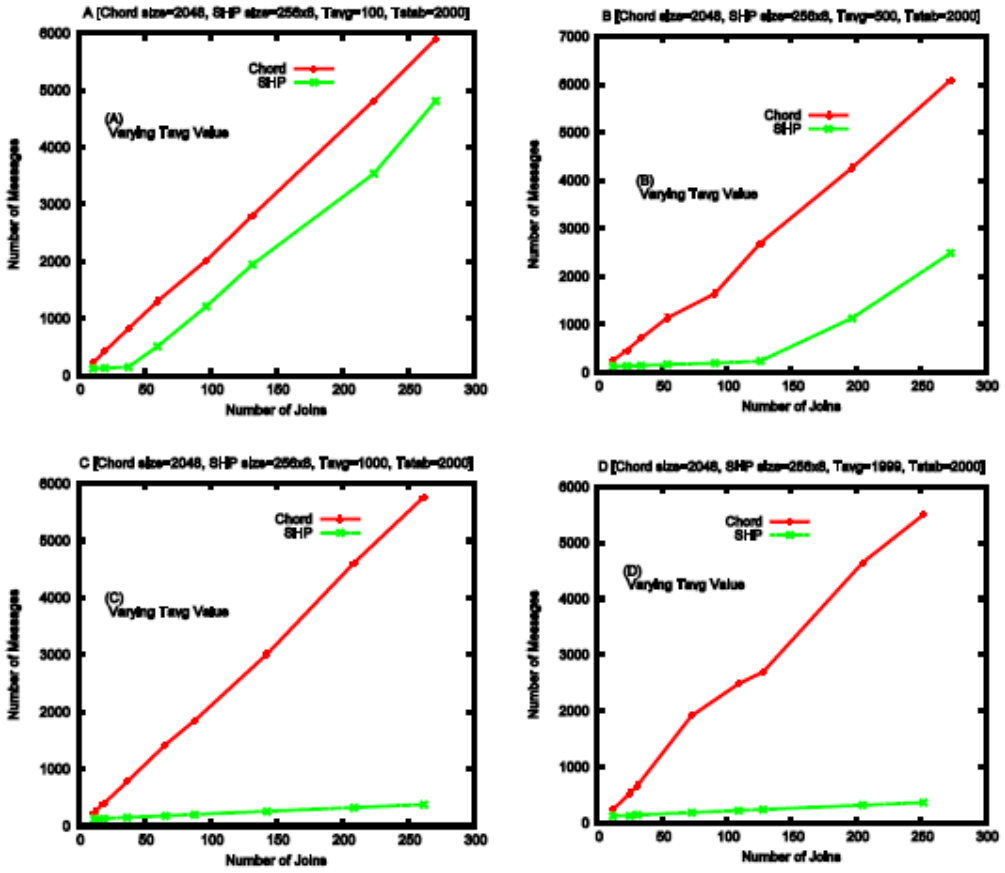

Figure 8: Graph showing number of messages required to join nodes in Chord and SHP with different T_{avg} value of same network size

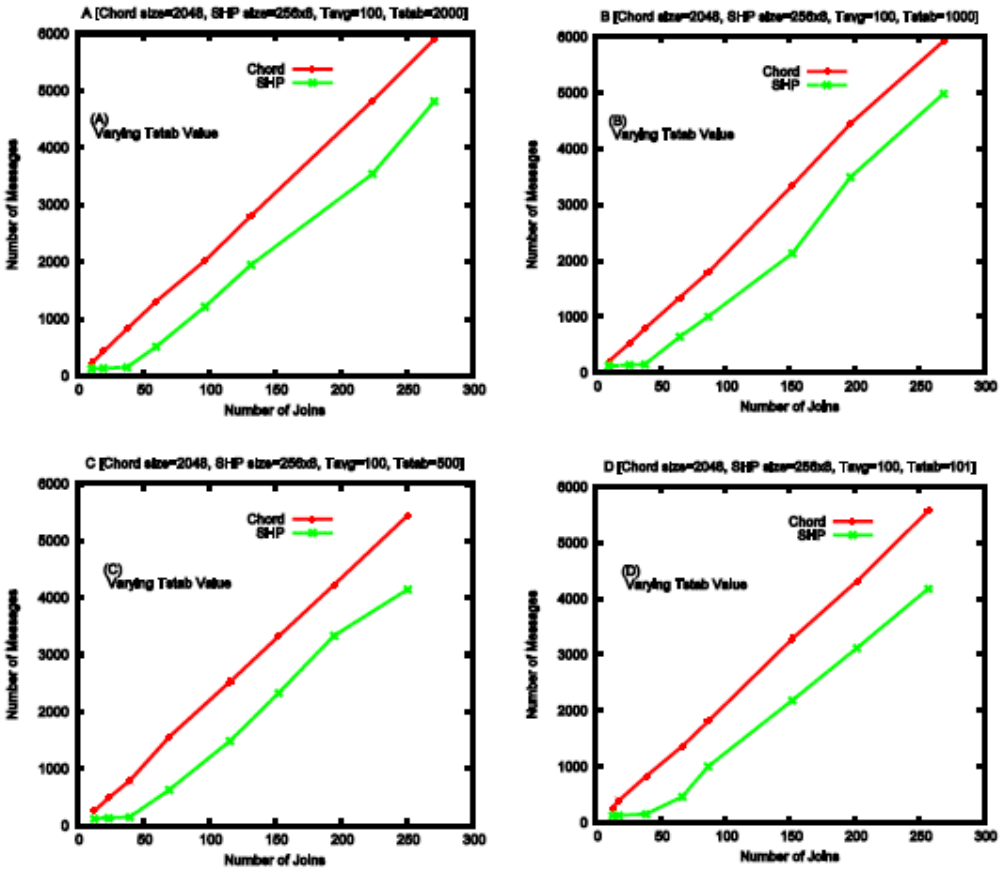

Figure 9: Graph showing number of messages required to join nodes in Chord and SHP with different T_{stab} value of same network size

REFERENCES

[1] K. Aberer. P-grid: A self-organizing access structure for p2p information systems. In Proceedings of the 9th International Conference on Cooperative Information Systems, pages 179–194, Trento, Italy, 2001. Springer-Verlag.

[2] K. Aberer, P. Cudr-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, and R. Schmidt. P-Grid: A self-organizing structured p2p system. ACM SIGMOD Record, 32(3), 2003.

[3] E. Adar and B. A. Huberman. Free riding on gnutella. First Monday, 5, 2000.

[4] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. Seti@home: an experiment in public-resource computing. Communication, ACM, 45(11):56–61, 2002.

[5] A. R. Bharambe, M. Agrawal, and S. Seshan. Mercury: supporting scalable multi-attribute range queries. SIGCOMM Computer Communication Review, 34(4):353–366, 2004.

[6] J. F. Buford, H. Yu, and E. K. Lua. P2P Networking and Applications. Morgan Kaufmann, 2009.

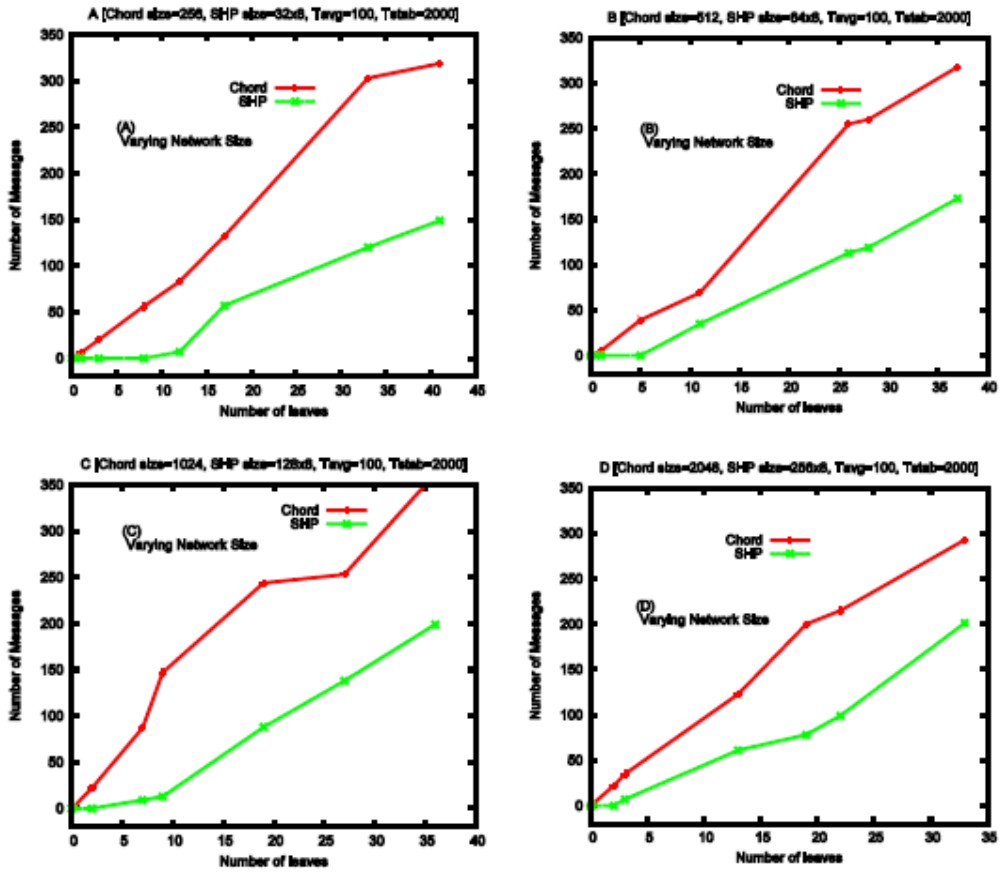

Figure 10: Graph showing number of messages required to leave nodes in Chord and SHP with different network size

[7] M. Castro, M. Costa, and A. Rowstron. Peer-to-peer overlays: structured, unstructured, or both? Technical report, Microsoft Research, USA, 2004.

[8] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In Lecture Notes in Computer Science, pages 46–66, 2000.

[9] B. Cohen. Incentives build robustness in bittorrent. Technical report, bittorrent.org, 2003.

[10] A. Crainiceanu, P. Linga, J. Gehrke, and J. Shanmugasundaram. Querying peer-to-peer networks using p-trees. In Proceedings of the 7th International Workshop on the Web and Databases (WebDB '04), pages 25–30, New York, NY, USA, 2004. ACM.

[11] L. Garces-Erce, E. Biersack, P. Felber, K. Ross, and G. Urvoy-Keller. Hierarchical peer-to-peer systems. In Proceedings of the ACM/IFIP International Conference on Parallel and Distributed Computing (Euro-Par), pages 643–657, 2003.

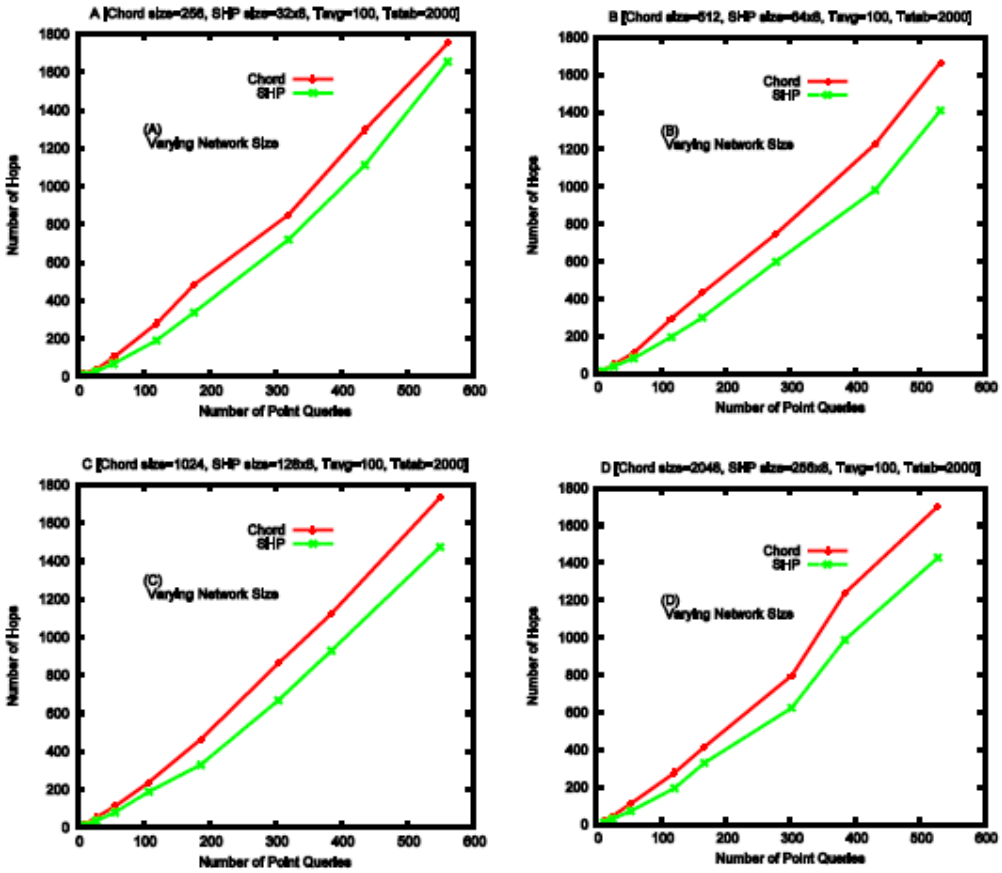

Figure 11: Graph showing number of hops required for point query in Chord and SHP with different network size

[12] F. Hong, M. Li, J. Yu, and Y. Wang. Pchord: Improvement on chord to achieve better routing efficiency by exploiting proximity. In Proceedings of the First International Workshop on Mobility in Peer-to-Peer Systems, pages 806–811, Washington, DC, USA, 2005. IEEE Computer Society.

[13] Yuh-Jzer Joung and Jiaw-Chang Wang. Chord2: A two-layer chord for reducing maintenance overhead via heterogeneity. Computer Networks, 51(3):712–731, 2007.

[14] N. Leibowitz, M. Ripeanu, and A. Wierzbicki. Deconstructing the kazaa network. In Proceedings of the Third IEEE Workshop on Internet Applications, pages 112–120, 2003.

[15] J. Liang, R. Kumar, and K. Ross. The fasttrack overlay: a measurement study. Computer Networks, 50:842–858, 2006.

[16] J. Liu and H. Zhuge. A semantic-based p2p resource organization model r-chord. Journal of Systems and Software, 79(11):1619–1634, 2006.

[17] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. Communications Surveys Tutorials, IEEE, 7(2):72–93, Feb 2005.

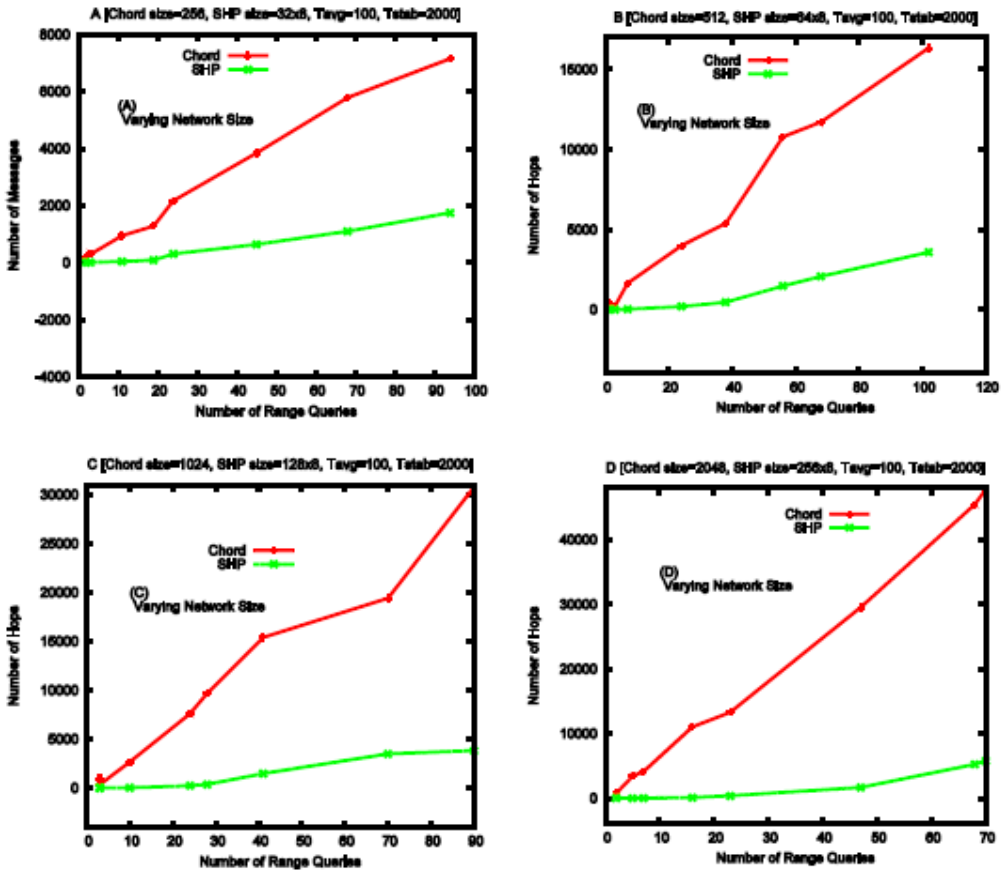

Figure 12: Graph showing number of hops required for range query in Chord and SHP with different network size

[18] N. Maibaum and T. Mundt. Jxta: A technology facilitating mobile peer-to-peer networks. In Proceedings of the International Mobility and Wireless Access Workshop, pages 7+, 2002.

[19] S. Naicken, A. Basu, B. Livingston, and S. Rodhetbhai. A survey of peer-to-peer network simulators. In Proceedings of The Seventh Annual Postgraduate Symposium, 2006.

[20] Gnutella protocol v0.4. http://dss.clip2.com/ gnutellaprotocol04.pdf, 2007.

[21] Gnutella protocol v0.6. http://rfc-gnutella.sourceforge.net/src/rfc-0 6-draft.html, 2007.

[22] R. Ranjan, A. Harwood, and R. Buyya. Peer-to-peer-based resource discovery in global grids: a tutorial. Communications Surveys Tutorials, IEEE, 10:6–33, Feb 2008.

[23] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content- addressable network. In Proceedings of the ACM SIGCOMM, pages 161–172, 2001.

[24] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. In Proceedings of the INFOCOM 2002, volume 3, pages 1190–1199, 2002.

[25] Sandvine Report. 2008 Analysis of traffic demographics in north-american broadband networks. Technical report, sandvine.com, 2008.

[26] M. Ripeanu. Peer-to-peer architecture case study: Gnutella network. In Proceedings of the IEEE 1st International Conference on Peer-to-Peer Computing, pages 99–100, Linkoping, Sweden, Aug 2001.

[27] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for largescale peer-to-peer systems. In Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), pages 329–350, Heidelberg, Germany, Nov 2001.

[28] R. Schollmeier. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In Proceedings of the First International Conference on Peer-to-Peer Computing, pages 101–102, Aug 2001.

[29] Y. Shao-Shan, Y. Jiong, K. Kamil, and S. Qi-Gang. Dr-chord - an efficient double-ring chord protocol. In Proceedings of the Sixth International Conference on Grid and Cooperative Computing, pages 197–202, Washington, DC, USA, 2007. IEEE Computer Society.
[30] R. Steinmetz and K. Wehrle (Eds.). P2P Systems and Applications. Springer-Verlag Berlin Heidelberg, 2005.
[31] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications, pages 149–160, 2001. ACM.
[32] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. IEEE Transactions on Networking, 11(1):17–32, Feb 2003.
[33] D. Stutzbach and R. Rejaie. Capturing accurate snapshots of the gnutella network. In Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies, volume 4, pages 2825–2830, 2005.
[34] Zhiyong Xu, Rui Min, and Yiming Hu. Hieras: a dht based hierarchical p2p routing algorithm. In Proceedings of the International Conference on Parallel Processing, pages 187 –194, 2003.
[35] D. Zeinalipour-yazti and T. Folias. A quantitative analysis of the gnutella network traffic. Technical report, Department of Computer Science University of California, 2002.
[36] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. IEEE Journal on Selected Areas in Communications, 22:41–53, 2004.
[37] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical report, Berkeley, CA, USA, 2001.
[38] S. Zhao, D. Stutzbach, and R. Rejaie. Characterizing files in the modern gnutella network: A measurement study. In Proc of the ACM Multimedia Computing and Networking, pages 1–13, 2006.
[39] S. Zols, R. Schollmeier, W. Kellerer, and A. Tarlano. The hybrid chord protocol: A peer-to-peer lookup service for context-aware mobile applications. In Proceedings of ICN, pages 781–792, 2005.

Authors

Dr.Guruprasad Khataniar: Received the BE degree in 1992 from Dibrugarh University, Assam and M.Tech degree in 1999 from Indian Institute of Technology, Delhi. He has received PhD in Computer Science and Engineering from Indian Institute of Technology, Guwahati, Assam. He is working as Lecturer (Selection Grade) in Computer Engineering Department at Assam Engineering Institute, Guwahati, Assam. His research interest is in peer-to-peer networking, Cloud Computing and Distributed System.

Dr.Diganta Goswami: Received the BE degree in 1986 from Gauhati University, Assam and M.E. degree in 1992 from Indian Institute of Science, Bangalore. He has done his PhD in Computer Science and Engineering from Indian Institute of Technology, Kharagpur in 2001. He is currently working as a Professor of Computer Science and Engineering at Indian Institute of Technology, Guwahati, Assam. His research interest is in peer-to-peer networking and distributed system. He is a member of IEEE.