Peer to Peer Networks – A Review & Study on Load Balancing

E.Mahender, Mr.B.Hari Krishna, Mr.K. OBULESH, Mrs.M.Shireesha

Abstract - Peer-to-peer (P2P) systems increase the popularity and have become a dominant means for sharing resources. In these systems, load balancing is one of challenge issue because nodes are often heterogeneous. While several load-balancing schemes, these solutions are typically ad hoc, heuristic based, and localized. In the analysis a general framework, HiGLOB, for global load balancing in structured P2P systems, each node in HiGLOB has a histogram manager maintains a histogram that reflects a global view of the distribution of the load in the system, and a load-balancing manager that redistributes the load whenever the node becomes overloaded or then exploit the routing metadata to partition the P2P network into no overlapping regions corresponding to the histogram buckets, mechanisms to keep the cost of constructing and maintaining the histograms low. Comparison analysis shows that our scheme can control and bound the amount of load imbalance across the system.

Keywords **-** Histogram, P2P Networks, Load Balancing, BATON

INTRODUCTION I

Peer-to-Peer (P2P) technology enables the network-connected device to provide services to another network-connected device. A device in a P2P network can provide access to any type of resource that it has at its disposal, whether documents, storage capacity, computing power, or even its own human operator. The device in a P2P network could be anything ranging from a super computer to simple PDA. P2P technology is a robust and impressive extension of the Internet's philosophy of robustness through decentralization. The main advantage of P2P networks is that it distributes the responsibility of providing services among all peers on the network; this eliminates service outages due to a single point of failure and provides a more scalable solution for offering services. In addition, P2P networks exploit available bandwidth across the entire network by using a variety of communication channels and by filling bandwidth up to the brim of the Internet. Unlike traditional client/server communications, in which specific routes to popular destinations can become overloaded the route to google.com, P2P enables communication via a variety of network routes, thereby reducing network overloading. P2P has the capability of serving resources with high availability at a much lower cost while maximizing the use of resources from every peer connected to the P2P network, whereas client/server solutions rely on the addition of costly bandwidth, equipment, and co-location facilities to maintain a robust solution. P2P can offer a similar level of robustness by spreading network and resource demands across the P2P network. A peer to peer network is formed by nodes of equal capability and act both as client and server at the same time depending on the function performed. Peer to peer networks and applications have advantages over traditional client server, such as the elimination of single point of failure, balance the network load uniformly and to provide alternate path routing easily in case of link failures [3]. A peer to peer network forms an overlay network on top of the existing network infrastructure. The formation of this overlay network by the peer to peer nodes make the resulting network resilient to changes in the underlying network such as router and link failures and congestion. An overlay network is a computer network which is built on top of another network. Nodes in the overlay can be thought of as being connected by virtual or logical links, each of which corresponds to a path, perhaps through many physical links, in the underlying network. Usually overlay network run at the application layer of the TCP/IP stack making use of the underlying layer and independent of them [3]. Figure 1 shows the basic architecture of an overlay network. The nodes in an overlay network will form a network architecture of their own that may be totally different and independent of the underlying network.

Although, P2P networks provide a wonderful solution to a completely decentralized network there are still some issues that need some attention. One of the major drawbacks in P2P networks is the unsupervised uploading of content. This leaves a room for infecting the network with viruses and worms. This problem was addressed by a research group in Victoria University by supervising and controlling the upload on the network. The problem of piracy and copyright protection was addressed by attaching digital metadata to the uploaded materials [2]. These two implementations tackled the problem of network-infection and Digital Right Management (DRM) [1]. But still there is plenty of room for further improvement in making P2P networks even more useful.

Histogram-based Global Load Balancing (HiGLOB) to facilitate global load balancing in structured P2P systems. Each node P in HiGLOB has two key components. The first component is a histogram manager that maintains a histogram that reflects a global view of the distribution of the load in the system. The histogram stores statistical information that characterizes the average load of non-overlapping groups of nodes in the P2P network. These nodes are connected to P through its neighbor nodes. The histogram information can be used for two purposes. On one hand, it is used to determine if a node is normally loaded, overloaded, or under loaded. On the other hand, it is used to facilitate the discovery of a lightly loaded node or a heavily loaded node for the load-balancing process when it is needed. The second component of the system is a load balancing manager that takes actions to redistribute the load whenever a node becomes overloaded or under loaded. The load-balancing manager may redistribute the load both statically when a new node joins the system and dynamically when an existing node in the system becomes overloaded or under loaded.

SECTION II

2.1. Average Service Capacity Approach in P2P Networks: We here illustrate limitations of the approach based on averaged quantities in a P2P network by considering the following examples. Suppose that a downloading peer wants to download a file of size F from N possible source peers. Let ci be the average end-to-end available capacity between the downloading peer and the ith source peer (i = 1, 2, . . . ,N). Notice that the actual value of ci is unknown before the downloading peer actually connects to the source peer i.

The average service capacity of the network, $\bar{C}$, is given by $\bar{C} = \sum_{i=1}^{N} c_i/N$

Intuitively, the average download time, T, for a file of size F would be $T = F/\bar{C}$.

In reality, however, (1) is far different from the true average download time for each user in the network. The two main reasons to cause the difference are (i) the spatial heterogeneity in the available service capacities of different end-to-end paths and (ii) the temporal correlations in the service capacity of a given source peer. We first consider the impact of heterogeneity. Suppose that there are two source peers with service capacities of c1 = 100kbps and c2 = 150kbps, respectively, and there is only one downloading peer in the network. Because the downloading peer does not know the service capacity of each source peer 1 prior to its connection, the best choice that the downloading peer can make to minimize the risk is to choose the source peers with equal probability. In such a setting, the average capacity that the downloading peer expects from the network is (100+150)/2 = 125kbps. If the file size F is 1MB, we predict that the average download time is 64 seconds. However, the actual average download time is 1/2(1MB/100kbps)+1/2(1MB/150Mbps) = 66.7 seconds! Hence, we see that the spatial heterogeneity actually makes the average download time longer. Suppose now that the average service capacity can be known before the downloading peer makes the connection. Then, an
obvious solution to the problem of minimizing the average download time is to find the peer with the maximum average capacity, i.e., to choose peer j with the average capacity cj (cj ≥ ci for all i), as the average download time Ti over source peer i would be given by F/ci. We assume that each peer can find the service capacity of its source peers via packet-level measurements or short-term in-band probing. Consider again the previous two-source peer example with
c1 = 100kbps and c2 = 150kbps. As we want to minimize the download time, an obvious choice would be to choose source peer 2 as its average capacity is higher. Now, let us assume that the service capacity of source peer 2 is not a constant, but is given by a stochastic process C2(t) taking values 50 or 250kbps with equal probability, thus giving E{C2(t)} = c2 = 150kbps. If the process C2(t) is strongly correlated over time such that the service capacity for a file F is likely to be the same throughout the session duration, it takes on average
(1MB/50kbps + 1MB/250kps)/2 = 96 seconds, while it takes only 80 seconds to download the file from source peer 1.

In other words, it may take longer to complete the download when we simply choose the source peer with the maximum average capacity! It is thus evident that the impact of correlations (second-order statistics) or higher-order statistics associated with the capacity fluctuation in time will need to be taken into account, even for finding a source peer with minimum *average* download time. In practice, most P2P applications try to reduce the download time by minimizing the risk of getting stuck with a 'bad' source peer (the connection with small service capacity) by using smaller file sizes and/or having them downloaded over different source peers (e.g., parallel download).2 In other words, they try to reduce the download time by minimizing the *bytes* transferred from the source peer with small capacity. However, we show in this paper that this approach cannot effectively remove the negative impact of both the correlations in the available capacity of a source peer and the heterogeneity in different source peers. This approach may help to reduce average download time in some cases but not always. Rather, a simple and distributed algorithm that limits the amount of *time* each peer spends on a bad source peer can minimize the average download time for each user almost in all cases. Through extensive simulations, we also verify that the simple download strategy outperforms all other schemes widely used in practice under various network configurations. In particular, both the average download time and the variation in download time of our scheme are smaller than any other scheme when the network is heterogeneous (possibly correlated) and many downloading peers coexist with source peers, as is the case in reality.

2.2. Service Capacity in P2P Heterogeneous: In a P2P network, just like any other network, the service capacities from different source peers are different. There are many reasons for this heterogeneity. On each peer side, physical connection speeds at different peers vary over a wide range (e.g., DSL, Cable, T1, etc). Also, it is reasonable to assume that most peers in a typical P2P network are just personal computers, whose processing powers are also widely different. The limitation in the processing power can limit how fast a peer can service others and hence limits the service capacity.

On the network side, peers are geographically located over a large area and each logical connection consists of multiple hops. The distance between two peers and the number of hops surely affect its round-trip-time (RTT), which in turns affects the throughput due to the TCP congestion control. Moreover, in a typical P2P network, this information is usually ‘hidden’ when a user simply gets a list of available source peers that have contents the user is looking for. Note that the aforementioned factors do not change over the timescale of any typical P2P session (days or a week). Hence, we assume that those factors mainly determine the long-term average of the service capacity over a given source peers.

2.2.1. Correlations in Service Capacity

While the long-term average of the service capacity is mainly governed by topological parameters, the actual service capacity during a typical session is never constant, but always fluctuates over time. There are many factors causing this fluctuation. First, the number of connection a source peer allows is changing over time, which creates a fluctuation in the service capacity for *each user*. Second, some user applications running on a source peer (usually a PC), such as online games, may throttle the CPU and impact the amount of capacity it can offer. Third, temporary congestion at any link in the network can also reduce the service capacity of all users utilizing that link.

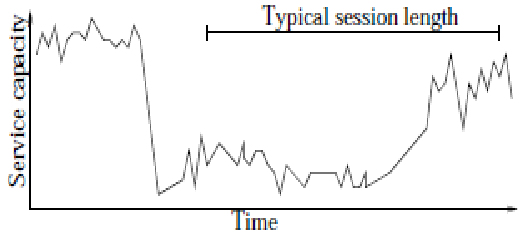


Figure. 1. Typical variation in ene-to-end available bandwidth .Drastic changes usually occur in the scale of minutes.

Figure 1 shows a typical available end-to-end capacity fluctuation. The time scale for the figure in the measurement study is on the order of minutes. We know that a typical file download session can last from minutes to hours for a file size of several megabytes. This implies that the service capacity over the timescale of one download session is stochastic and correlated. In Figure 1, the short-term variations in the capacity are mainly due to the window size fluctuation in TCP, while the long-term variations are due to network congestion, changes in workload or the number of connecting users at the source peer, etc. The long-term fluctuation typically lasts over a longer time scale, say, few minutes up to several hours. As illustrated in the introduction, both the heterogeneity over different source peers and the correlations of the capacity in a given source peer have significant impact on the average download time. To the best of our knowledge, however, there has been no result available in the literature addressing these issues. All the existing studies have simply assumed that the service capacity is given by a constant (its average value) for the duration of a download. Consequently, the download time of a file of size F is simply given by F/c, where c is the average service capacity. As will be seen later on, however, this is true only when the service capacity is constant or i.i.d. over time, neither of them is true in reality.

2.3. Charactizing the Download Time in P2P Networks: We consider our network as a discrete-time system with each time slot of length _. For notational simplicity, throughout the paper, we will assume that the length of a time slot is normalized to one, i.e., _ = 1. Let C(t) denote the time-varying service capacity (available end-to-end bandwidth) of a given source peer at time slot t (t = 1, 2, . . .) over the duration of a download. Then, the download time T for a file of size is defined as

$$T = \min\left\{ s > 0 \;\middle|\; \sum_{t=1}^{s} C(t) \geq F \right\}.$$

Note that T is a stopping time or the first hitting time of a process C(t) to a fixed level F.

If C(t), t = 1, 2, . . . are independent and identically distributed (i.i.d.), then by assuming an equality in (2), we obtain from Wald's equation that

$$F = \mathbb{E}\left\{ \sum_{t=1}^{T} C(t) \right\} = \mathbb{E}\{C(t)\}\mathbb{E}\{T\}.$$

The expected download time, measured in slots, then becomes E{T} = F/E{C(t)}. Note that (3) also holds if C(t) is constant (over t). Thus, when the service capacity is i.i.d. over time or constant, there exists a direct relationship between the average service capacity and the average download time, as has typically been assumed in the survey.

SECTION III

3. Problem Definition: Resource sharing is rapidly grown in popularity in peer-to-peer networks load balancing is a key challenge because nodes are heterogeneous to avoid all these we proposed a framework HIGLOB to enable global load balance for structured p2p systems, each node in HIGLOB maintains the load information of nodes in the systems using histograms and partition the system into non-overlapping groups of nodes and maintain the average load of them in the histogram at a node.

3.1. Histogram Construction: Based on the histogram structure described above, we now design the algorithms for constructing and maintaining histograms in Skip Graph. A new node needs all of its neighbor nodes to send the necessary information to it for constructing the histogram. The question now is what information neighbor nodes should send to the new node. It is impossible for a neighbor node to send the load of every node in the non-overlapping group connected by it because there is no way to get such information from a histogram. The neighbor node cannot send the average load of the non-overlapping group either because it does not know which node is the next neighbor node of the new node at a higher level. Instead, a neighbor node x needs to send to the new node the summary of load and capacity of all nodes following x on the opposite side with the new node together with the load and capacity of x itself, these

Values can be calculated from histogram values of non-overlapping groups representing these nodes. Note that for the computation to be done, the histogram value of a non-overlapping group must be represented by a pair of the total load and the total capacity of all nodes in that group.

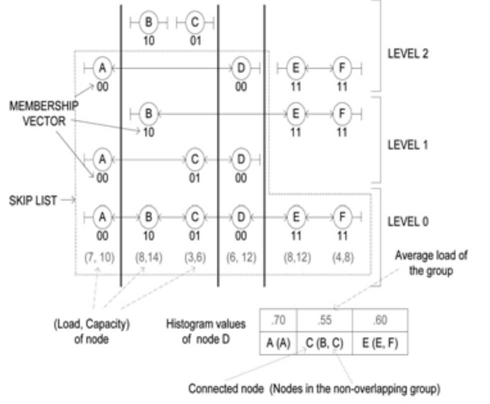

LEVEL 2
LEVEL 1
LEVEL 0
MEMBERSHIP VECTOR
SKIP LIST
Average load of the group
(Load, Capacity) of node
Histogram values of node D
Connected node (Nodes in the non-overlapping group)

Figure 2 Histogram structure in Skip Graph

For example, as in Fig., the stored value of a group of nodes connected through node C at D should be (11, 20) instead of the average load result 0.55. From now on, we always imply the average load as a pair of such information even though we only show the calculated average loads in the figures to simplify the presentation. When a new node receives all the necessary information from its neighbor nodes, it can construct its histogram. The average load of a group connected by a node y is calculated from the received information from y and the next neighbor node z at the nearest higher level on the same side.

Assume that a new node G joins the network at a position between E and F and it has a membership vector with prefix "01" as in below figure after that joining G has four different neighbor nodes C,D,E and F. As above C needs to send of G the load and capacity of A, B, and C which is (18,30) the first parameter is the total load while the second parameter is the total capacity.

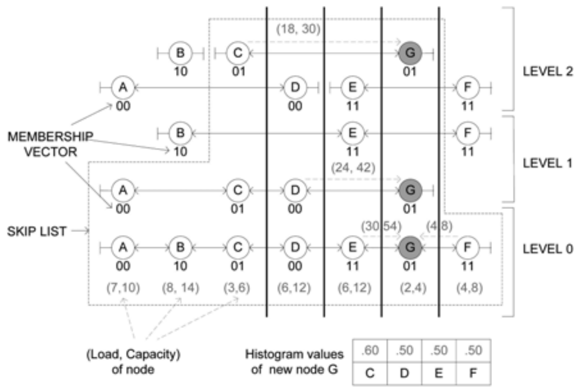
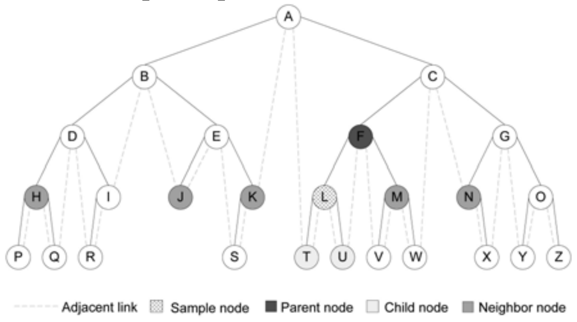


Figure 3 Histogram construction for a new node in Skip Graph



Similarly D,E and F respectively send to G (24,42) (30,54) and (4,8). Finally from the received information G can calculate the average load of these four non-overlapping groups connected by C, D,E and F to build its histogram.

3.2. BATON: Balanced Tree Overlay Network (BATON) [7] is a structure based on a binary balanced tree in which each peer in the network maintains a node of the tree. A node has connections to other nodes by four different kinds of links a parent link pointing to the parent node; child links pointing to child nodes; adjacent links pointing to adjacent nodes, which maintain adjacent ranges of values and neighbor links pointing to selected neighbor nodes, which are nodes at the same level, having distances equal to a power of two from the node. In BATON, data stored at nodes is ordered increasingly from the left to the right of the tree. Example of BATON is shown in above Figure.

In BATON, when a node x processes a query, if the searched key does not fall into the range of values managed by x, x forwards the query to the farthest neighbor node in the routing table, which is nearer to the searched key. If there is no such neighbor node, x forwards the query to either a child (if it exists) or an adjacent node of x in the search direction. In particular, if x is a leaf node without a full routing table on the search direction, x always forwards the query to its parent node for processing.

3.3. Histogram Construction and Maintenance: The way histogram is constructed and maintained in BATON is similar to that of Skip Graph. When a new node joins the system, all of its neighbor nodes have to calculate and send the summary of load and capacity of themselves and all nodes following their position on the same side of the new node from their histogram values. However, in BATON, the histogram values are not recalculated right away if the new node does not have full routing tables. These values are kept until the node has full routing tables. At that time, histogram values are recalculated. The histogram values of non-overlapping groups, which contain nodes falling between the node and the first neighbor node, are calculated with additional information from histogram values of the parent node. When the load of a node is changed, it sends updated information, which is calculated in the same way as in the process of histogram construction, to all of its neighbor nodes. The following update propagation is also similar to that of Skip Graph.

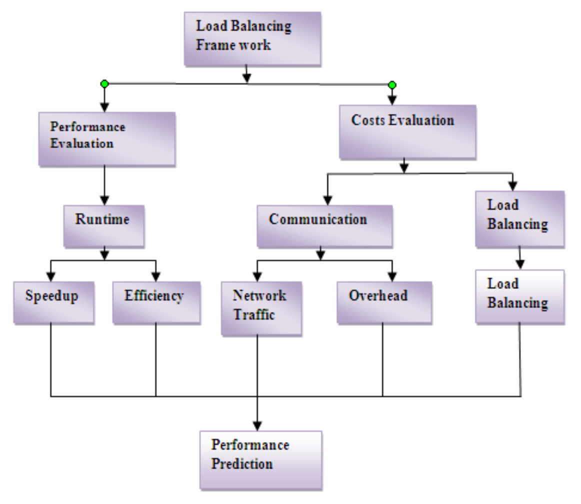
SECTION IV

4.1. Experimental Study: Evaluation of performance proposal, built a simulator for our framework on which we implemented the three structured P2P systems: Skip Graph [21], BATON [7]. The simulator is used to test the storage load distribution on large-scale networks from 1,000 to 10,000 nodes over Planet Lab, a testbed for large-scale distributed systems. For each network size N, we insert a data set of 100 _ N data items each of which has the same size. In this way, the work load of a node is simply the number of data item stored at that node. Analysis in the system is three different network types:

1. Type-I is a homogeneous system with skewed data distribution,
2. Type-II is a heterogeneous system with uniform data distribution.
3. Type-III is a heterogeneous system with skewed data distribution.

For consistent hashing, which distributes data uniformly this version can only be used to test with Type-II network. To generate heterogeneous nodes, we used a real data set from that in [8], which measures the amount of shared data at nodes in the Gnutella system (we assume that the storage capability of nodes is proportional to the amount of data they share or store in the data set). To generate skewed data distribution, we used the Zipfian method with parameter 1.0. The default values of the network size and 10,000 and 2.0.

4.2. Load Balancing Framework: In a simulation framework of a load balancing algorithms proposed. Furthermore, the authors stated that any load balancing algorithm depends on some important factors including communication delay, topology, work load and negotiation protocol. In another hand the research concluded that for any simulation study framework of a load balancing algorithm, there are some factors that must take in to account. In real state two factors of those factors must take into account that is the performance evaluation and the cost evaluation. Depending on the above conclusion, the framework isolates the performance factors of the system in to three main components.



4.3. Comparative Study: In Existing system if the number queried nodes is large enough the node can approximate the average load of the system and hence it can determine if it overloaded. The system is overloaded it redistributes its load with the queried node having the lightest or heaviest load since that node should be lightly loaded node problem with this method is that it can only guaranteed the global load balance of the system with some probability. By using this technique we have many disadvantages listed here it can only guaranteed the global load balance of the system with some probability. Suggest a use of a separate DHT such as Skip Graph to maintain the nodes load distribution nevertheless this solution still have a problem it incurs a substantial cost of maintaining complete information about the load at every node in the system. Our work proposes that nodes are connected to P through its neighbor nodes histogram information can be used for two purposes one it is used to determine if a node is normally loaded overloaded and the on the other hand it is used to facilitate the discovery of a likely loaded node or a heavily loaded node for the load-balancing process when it is needed, our framework that uses histograms to maintain a global view of the load distribution on structured P2P systems. This histogram enables efficient load-balancing algorithms that can effectively control the amount of load imbalance across the system to globally balance the load that reduces the cost of constructing and maintaining the histograms.

CONCLUSION V

A framework, HiGLOB, to enable global load balance for structured P2P systems. Each node in HiGLOB maintains the load information of nodes in the systems using histograms. Enables the system to have a global view of the load distribution and hence facilitates global load balancing. Partition the system into non-overlapping groups of nodes and maintain the average load of them in the histogram at a node. Techniques to reduce the overhead of maintaining and constructing histograms. Even though the proposal is a general framework, it is possible to deploy different kinds of P2P systems on it. Comparison shows that our HiGLOB enabled systems are superior over other methods.

Reference -

[1] Shi H., Zhang Y., Zhang J. and Beal E. "Managing Digital Copyright Licences in Peer-to- Peer Collaborative Research Network", *International Multi-Symposiums on Computer and Computational Sciences (IMSCCS)*, The University of Iowa, Iowa City, Iowa, USA , August 13 - 15, 2007, pp. 198-203.

[2] M Gupta, P Judge, M Ammar. "A reputation system for peer-to-peer networks". Proc. 13th international workshop Network and operating systems support for digital audio and video, 2003.

[3] R. Zhou, K. Hwang. "PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to- Peer Computing". IEEE Trans. Parallel Distributed Systems, 18(4), pp. 460-473, April, 2007.

[4] Selcuk, A.A. Uzun, E. Pariente, M.R "A reputation-based trust management system for P2P networks". Proc. IEEE Int. Symposium on Cluster Computing and the Grid, 2004.

[5] S3: Scalable, Shareable and Secure P2P Based Data Management System, http://www.comp.nus.edu.sg/~s3p2p, 2008.

[6] Gnutella, http://www.gnutella.com/, 2008.

[7] H.V. Jagadish, B.C. Ooi, and Q.H. Vu, "Baton: A Balanced Tree Structure for Peer-to-Peer Networks," Proc. Very Large Databases Conf. (VLDB '05), pp. 661-672, 2005.

[8] S. Saroiu, P.K. Gummadi, and S.D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," Proc. Multimedia Computing and Networking Conf. (MMCN), 2002.

[9] M. Mitzenmacher, "The Power of Two Choices in Randomized
Load Balancing," IEEE Trans. Parallel and Distributed System,
vol. 12, no. 10, pp. 1094-1104, Oct. 2001.

[10] A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, "Load Balancing in Structured P2P Systems," Proc. Int'l Workshop Peer-to-Peer Systems (IPTPS), 2003.

[19] D. Karger and M. Ruhl, "Simple Efficient Load Balancing Algorithms for Peer-to-Peer Systems," Proc. ACM Symp. Parallelism in Algorithms and Architectures (SPAA), 2004.

[20] P. Ganesan, M. Bawa, and H. Garcia-Molina, "Online Balancing of Range-Partitioned Data with Applications to Peer-to-Peer Systems," Proc. Very Large Databases Conf. (VLDB '04), pp. 444-455, 2004.

[21] J. Aspnes and G. Shah, "Skip Graphs," Proc. 14th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA '03), pp. 384-393, 2003.

[22] D. Karger and M. Ruhl, "Simple Efficient Load Balancing Algorithms for

Peer-to-Peer Systems," Proc. Int'l Workshop Peerto- Peer Systems (IPTPS), 2004.
[23] B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, "Load Balancing in Dynamic Structured P2P Systems," Proc. INFOCOM, 2004.
[24] K. Kenthapadi and G.S. Manku, "Decentralized Algorithms Using Both Local and Random Probes for P2P Load Balancing," Proc. ACM Symp. Parallelism in Algorithms and Architectures (SPAA), 2005.
[25] G. Giakkoupis and V. Hadzilacos, "A Scheme for Load Balancing in Heterogenous Distributed Hash Tables," Proc. ACM Symp. Principles of Distributed Computing Conf. (PODC), 2005.
[26] J. Ledlie and M. Seltzer, "Distributed, Secure Load Balancing with Skew, Heterogeneity, and Churn," Proc. INFOCOM, 2005.
[27] S. Surana, B. Godfrey, K. Lakshminarayanan, R. Karp, and I. Stoica, "Load Balancing in Dynamic Structured P2P Systems," Performance Evaluation, vol. 63, no. 6, pp. 217-240, 2006.
[28] M. Adler, E. Halperin, R.M. Karp, and V.V. Vazirani, "A Stochastic Process on the Hypercube with Applications to Peerto- Peer Networks," Proc. 35th ACM Symp. Theory of Computing (STOC '03), pp. 575-584, 2003.

E.Mahender. He holds B.Tech degree in Computer Science and Engineering, and M.Tech degree in Software Engineering from JNTU, Hyderabad. At present, he is working as **Assoc. Professor & Head, Dept of Computer Science and Engineering** in Avanthi Institute of Engineering & Technology, Hyderabad, Andhra Pradesh, India. He is a member of Computer Society of India and founder branch counselor of CSI student branch in the college. His areas of interests are Network security, Data Ware Housing and Data mining, Image Processing.

Mr.B.Hari Krishna Assoc.prof, He has done his B.Tech in Information Technology from JNTU Hyderabad, and completed his M.Tech (ALCCS) in CSE (Part Time) from IETE Hyderabad, Currently he is working in CSE dept and Chief Exam Branch in charge at Avanthi Institute of Engineering and Technology, Hyderabad, Andhra Pradesh, India . He is a member of Computer Society of India. His areas of interests are Data Ware Housing and Data mining, Image Processing, Mobile Computing Network security.

Mr.K. OBULESH Asst.prof, He has done his B.C.A from S.V.University .MSc Information Technology from Osmania University M.Tech Geo-informatics from JNTU, Hyderabad.Now he is working in CSE Dept at Avanthi Institute of Engineering and Technology, Hyderabad, Andhra Pradesh. Having 1 + year's software industry experience in SAP as an Analyst. His areas of interests are Geo-informatics, Digital Image Processing, Cloud Computing, Network security.

Mrs.M.Shireesha Assoc.prof, Completed B.Tech in Computer Science & Information Technology & M.Tech in Computer Science & Engineering from JNTU, Hyderabad. She is a member of Computer Society of India**.** Now she is working in CSE Dept at Avanthi Institute of Engineering and Technology, Hyderabad, Andhra Pradesh. Her areas of interests are Image Processing, ,Network security Data Ware Housing and Data mining