Scaling Unstructured Peer-to-Peer Networks With Multi-Tier Capacity-Aware Overlay Topologies

Mudhakar Srivatsa, Buğra Gedik and Ling Liu
College of Computing, Georgia Institute of Technology
{mudhakar, bgedik, lingliu}@cc.gatech.edu

Abstract[1]
The peer to peer (P2P) file sharing systems such as Gnutella have been widely acknowledged as the fastest growing Internet applications ever. The P2P model has many potential advantages due to the design flexibility of overlay networks and the server-less management of cooperative sharing of information and resources. However, these systems suffer from the well-known performance mismatch between the randomly constructed overlay network topology and the underlying IP layer topology for packet routing. This paper proposes to structure the P2P overlay topology using a capacity-aware multi-tier topology to better balance load at peers with heterogeneous capacities and to prevent low capacity nodes from downgrading the performance of the system. To study the benefits and cost of the multi-tier capacity aware topology with respect to basic and advanced routing protocols, we also develop a probabilistic broadening scheme for efficient routing, which further utilizes capacity-awareness to enhance the P2P routing performance of the system. We evaluate our design through simulations. The results show that our multi-tier topologies alone can provide eight to ten times improvements in the messaging cost, two to three orders of magnitude improvement in terms of load balancing characteristics, and seven to eight times lower topology construction and maintenance costs when compared to Gnutellas random power-law topology.

1 Introduction

With applications such as Gnutella [6], KaZaA [8], and Freenet [5], the peer-to-peer (P2P) model is quickly emerging as a significant computing paradigm of the future Internet. Unlike traditional distributed computing, P2P networks aggregate large number of computers and possibly mobile or hand-held devices, which join and leave the network frequently. This new breed of systems creates application-level virtual networks with their own overlay topology and routing protocols. P2P systems can be broadly classified into two categories: Structured and Unstructured P2P systems. The primary focus of structured (distributed hash table based) P2P systems is on precisely locating a given object (identified by some unique ID) within a bounded number of hops. However, the main focus of unstructured P2P systems (Gnutella [6], KaZaA [8]) is to support approximate search and partial answers in a typical distributed file sharing application. Many agree that unstructured P2P systems are more suitable for mass-market file sharing applications and thus there are increasing interests for mechanisms that can make unstructured P2P systems scalable [2].

Most of the decentralized unstructured P2P overlay networks, such as Gnutella and KaZaA, share two unique characteristics. First, the topology of the overlay network and the placement of files within the network are largely unconstrained. Second, queries are flooded via a *broadcast-styled routing (bsr)* algorithm across the overlay network with limited scope. Upon receiving a query, each peer sends a list of all content matching the query to the query originator node. In the *bsr* scheme, a query Q is specified by a quadruplet: $\langle$*originator, keywords, ID, TTL*$\rangle$, where $Q.originator$ is the query originator, $Q.keywords$ is the list of user supplied keywords, $Q.ID$ is the unique query identifier and $Q.TTL$ is the Time-to-Live of the query. The query originator assigns the query Q a unique ID ($Q.ID$) and sets the scope of the query $Q.TTL$ to $initTTL$. When a peer p receives a query Q from any neighbor peer q, peer p checks if it has seen this query before by checking if it is a *duplicate* query (using $Q.ID$). If so, peer p drops the query Q; else peer p sends results from its local file index to peer q. If the query Q's TTL has not yet expired ($Q.TTL > 0$) then peer p forwards the query Q with its TTL decremented by one to all its neighbors (except peer q).

There are several serious performance problems with the use of such a random topology (RT) and its random broadcast based routing protocol. First, this approach does not distinguish peer heterogeneity in terms of their computing and communication capacity (such as access bandwidth, CPU, Disk I/O). It has been observed in [11] that peers are highly diverse in terms of their network resources and their participation times. Unfortunately, most decentralized P2P systems, construct an overlay network randomly resulting in unstable and less powerful peers hindering the system's performance. Further, a weak peer on a path between two powerful peers throttles the throughput between the powerful peers. Second, the random topology combined with flooding-based

[1]This research is partially supported by NSF CNS CCR, NSF ITR, DoE SciDAC, DARPA, CERCS Research Grant, IBM Faculty Award, IBM SUR grant, HP Equipment Grant, and LLNL LDRD. Any opinions, findings, and conclusions or recommendations expressed in the project material are those of the authors and do not necessarily reflect the views of the sponsors.

routing is clearly not scalable since the load on each peer grows linearly with the total number of queries in the network, which in turn grows with the size of the system.

In this paper, we propose capacity-based techniques to structure the overlay topology with the aim of improving the overall utilization and performance by developing a number of system-level facilities.

- We propose a overlay network scheme with *Multi-Tier Capacity Aware Topologies*, capacity aiming at improving the network utilization, reducing the search latency, and improving the fault tolerance of the P2P system.
- We develop *An analytical model* that enables us to construct multi-tier network-connection aware topologies such that the number of lookup queries served by a peer is commensurate with its capacity.
- *Capacity guided bootstrap and topology maintenance service* that enables us to efficiently construct and maintain the overlay topology.
- *A Probabilistic-Selective* routing technique, which employs a capacity-aware routing algorithm.

2 Multi-Tier Overlay Topologies

Unstructured P2P networks such as Gnutella face a common problem – nodes can quickly become overloaded in the presence of high aggregate query rate, and the situation may be aggravated as the number of nodes in the system increases. As a consequence, the system utilization decreases dramatically. Therefore, our first design objective is to make the unstructured P2P networks more scalable by promoting the Capacity-Aware Topologies (CAT), which exploits peer heterogeneity with respect to bandwidth connectivity, CPU, and Disk I/O. Our second design objective is to enable the multi-tier topologies to scale well with increasing number of nodes in the system. To achieve such scalability we strive to design the CAT construction and maintenance algorithms with two aims. First, we want to avoid overloading any of the nodes by explicitly taking into account of their capacity constraints. Second, we want to keep nodes with low capacities stay connected.

We organize this section as follows. First we introduce the concept of heterogeneity level (HL) and then formally introduce the three classes of multi-tier capacity-aware topologies. To provide a qualitative comparison among the three types of topologies, we develop a set of metrics to characterize their performance in terms of messaging bandwidth, load variance, query capacity, query latency, and fault-tolerance in terms of node connectivity. We then present an analytical model and algorithms for construction and maintenance of three classes of multi-tier topologies.

2.1 CAT: Capability-Aware Topologies

Before we describe the design of multi-tier capability-aware overlay topologies, we introduce the concept of Heterogeneity Levels, which is used as a guideline to construct connection aware topologies, and a set of Performance Metrics, which serve as the basic model to evaluate and compare different classes of overlay topologies.

We classify nodes into *Heterogeneity Levels* (*HLs*) based on their *capabilities* with level zero used to denote the *least powerful* set of nodes. Peers at the same heterogeneity level are assumed to be *almost* homogeneous in terms of their capability. The key idea is to ensure that two nodes whose HLs vary significantly are not directly connected in the overlay network. So, we *allow a connection between two nodes* i *and* j *only if* $|HL(i) - HL(j)| \leq 1$. In the following sections of this paper, we assume that the set of node classes are given to us, and that each node class is associated with a scalar capability C (presumably based on access network bandwidth).

Classifying nodes into capability classes is vital for the performance of the P2P system for at least the following reasons: (i) Weak nodes are prevented from becoming hot-spots or bottle-necks that throttle the performance of the system, (ii) Avoiding bottle-necks decreases the average query response time as perceived by an end user, and (iii) From the perspective of the P2P system designer, *load balanced architectures* improve the overall throughput of the system. For instance, we can use this notion of heterogeneity level to avoid a weak node from suffocating the performance of the system as follows. Given the fact that we construct the overlay network by only connecting nodes of comparable HLs, we can reduce the load on weak nodes by constructing topologies such that the connectivity of the nodes increases with their HL. Hence, a powerful node would maintain more connections and would receive more queries on an average when compared to a weak node. In short, our capability-aware overlay topologies achieve significant performance gains by getting more work done by the powerful nodes and reducing the workload on the weaker nodes.

2.2 Three Classes of Multi-Tier Topologies

We first formally define three classes of overlay topologies: *Hierarchical Topology*, *Layered Sparse Topology* and *Layered Dense Topology*. We characterize an overlay topology by the *type* of connections maintained by each node in the system.

We now formally define and analyze the performance characteristics of these three classes of overlay topologies.

Definition *Multi-Tier Topologies: Let* $HL(p)$ *denote the heterogeneity level of peer* p*. Let* $maxHL$ *denote the class of peers at the maximum heterogeneity level. A multi-tier topology is defined as a weighted, undirected graph G:*$\langle V, E\rangle$

where V is the set of nodes and E is the set of edges such that the following conditions are verified:

1. Connectivity Preserving Rule: $\forall p \in V$, if $HL(p) < HL(maxHL)$ then $\exists q \in V$ such that $HL(q) = HL(p)+1 \wedge (p,q) \in E$
2. $\forall (p,q) \in E$, assuming $HL(p) \leq HL(q)$ without loss of generality, we have
   (a) Hierarchical Topology: $p,q \in maxHL \vee \big(HL(q) = HL(p)+1 \wedge \nexists q' \neq q$ s.t. $(HL(q') \geq HL(p) \wedge (p,q') \in E)\big)$
   (b) Sparse Topology: $p,q \in maxHL \vee HL(q) = HL(p)+1$
   (c) Dense Topology: $p,q \in maxHL \vee HL(q) = HL(p)+1 \vee HL(q) = HL(p)$

Informally, a hierarchical topology is a pure *tree-like* topology, except that the nodes in the highest HL are strongly connected (but not completely connected). An example 3-tiered hierarchical topology is shown in Figure 1. A sparse topology improves the fault-tolerance and reduces the diameter of the hierarchical topology by permitting a node to maintain connections with more than one node at its immediate higher level, as illustrated in Figure 2. A dense topology is equivalent to the basic multi-tier topology that adds no further restrictions on the connections between nodes; see Figure 3. Note that constraint (1) in the definition of the topologies is very important to maintain the *connectivity* of the overlay network; while constraint (2) is responsible for the differences observed in the performance trends of these three classes of multi-tier overlay topologies.

2.3 Performance Characterization of Three Multi-Tier Topologies

2.3.1 Performance Metrics

We characterize the goodness of an overlay topology using the following five metrics.

Amortized messaging bandwidth: Amortized messaging bandwidth is measured as the ratio of the total *capacity-aware bandwidth (CAB)* consumed by a query to the number of results obtained. Capacity-Aware Bandwidth (CAB) is defined as the ratio of the total number of messages sent/received by a node to the node's capacity.

Load Variance: Load on a node is measured as the ratio of the total number of messages sent/received by a node to its capacity, i.e., the amount of CAB expended by a node in the system. We characterize load distribution of a heterogeneous P2P system by the variance of load over all nodes in the system.

Coverage: Number of nodes that receive a broadcast query whose initial TTL was set to $initTTL$.

Amortized Latency: Ratio of total response time for a query to the number of results obtained.

Fault-Tolerance: Fault-tolerance is measured as the fraction of the number of results obtained when a random set of nodes fail.

2.3.2 Performance Trends

Having formally defined the overlay topologies, we now discuss in depth their performance trends with respect to various performance metrics listed in section 2.3.1.

Amortized messaging bandwidth: In a hierarchical topology, the tree structure ensures that there is no duplication of queries (except for the nodes at the strongly connected highest HL). However the increase in the number of duplicate queries in the sparse and the dense topology (due to redundant paths in the overlay topology) increases their amortized messaging bandwidth requirement. In comparison to a random topology, we achieve a huge bandwidth savings because our capability-aware overlay topologies ensure that the number of messages required to be handled by a node is commensurate with its capability.

Load Variance: In a hierarchical topology, due to its *tree-like* structure, the nodes at the highest HL are likely to be more heavily loaded (relative to their capability) than the nodes at lower levels, thereby making the load distribution somewhat skewed. Nevertheless, our multi-tier topologies, by their very structure, ensure that the load on a node is commensurate with its capability, and hence show remarkably better load distribution when compared to a random topology that is agnostic to peer heterogeneity.

Coverage: A hierarchical topology's tree-like structure increases the diameter of the overlay network, and tends to bring down the coverage when compared to a random topology. Also, by the structured nature of sparse and dense topologies, they reduce the number of duplicate queries, and hence achieve higher coverage, when compared to a random topology.

Amortized Latency: Latency is much lower in the multi-tier topologies primarily because they largely avoid weak nodes from appearing on a overlay path between two powerful nodes. Assuming that the communication latency between two nodes is constrained by the weaker of the two nodes, the response time (and consequently amortized latency) in a random topology would be much higher than a multi-tier topology.

Fault-Tolerance: In a hierarchical topology, if a node fails, then the entire *sub-tree* rooted at that node gets disconnected from the P2P network temporarily. The sparse and dense topologies show increasingly higher fault tolerance, but still

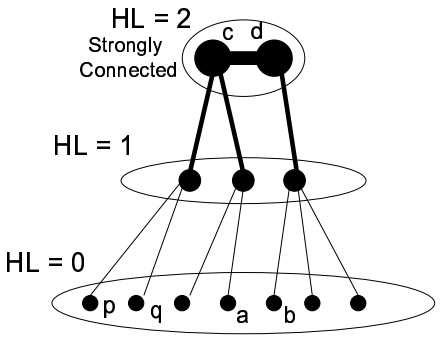

Figure 1: Hierarchical Topology

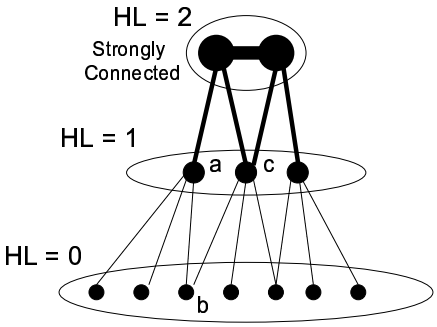

Figure 2: Sparse Topology

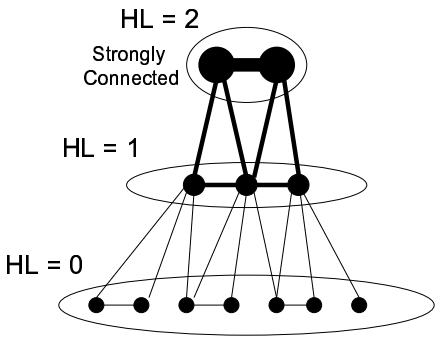

Figure 3: Dense Topology

they assign *higher responsibility* to higher level nodes thereby making them marginally weaker than the random topology.

2.4 Constructing CATs: Analytical Model

Having presented three classes of multi-tier capability-aware overlay topologies and analyzed their performance trends with respect to various performance metrics, we now turn our attention to constructing such overlay topologies. We utilize an analytical model described in [12] that provides directives to construct an overlay topology that minimize the variance of CAB over all nodes in the system. This not only ensures that all nodes contribute equal amounts of CAB to the P2P system (thereby, eliminating bottle-necks in the system), but also significantly reduces the amortized messaging bandwidth requirement.

3 CAR: Capability-Aware Routing

3.1 Design Ideas

Recall the broadcast styled routing (*bsr*) technique used in most Gnutella-like P2P systems today, query originator initializes $Q.TTL$ to $maxTTL$ and consequently the query Q reaches all peers that are at most $maxTTL$ hops from the originator. Several threshold based iterative algorithms have been proposed to improve the performance of the routing algorithm. For example, the restricted depth first search ($rdfs$) or the iterative deepening technique attempts to be *lucky* by *satisfying* the query within a smaller scope (fewer hops) [13]. Each query is associated with another parameter $Q.threshold$, specifying the number of results required. The query originator iterates over query's $initTTL$, starting from $Q.initTTL$ = $minTTL$ to $maxTTL$ ($minTTL$ and $maxTTL$ are system parameters) until the query Q is satisfied. However, all these routing techniques do not exploit the huge variances observed in terms of both the number of documents shared by each peer and the bandwidth capability of different peers. In an open P2P system like Gnutella in which a large number of non-cooperating peers are present, it has been observed that a large number of peers (70%) are free-riders [1] and that about 90% of documents is shared by about 10% of the nodes [11]. This means that most of the peers do not contribute to the peer community but merely utilize the resources provided by a small subset of peers in the community.

The key idea behind our *probabilistic selective routing (psr)* algorithm is to promote a focused search through selective broadcast. The selection takes into account the peer heterogeneity and the huge variances in the number of documents shared by different peers. Our routing algorithm iterates over the number of neighbors to whom the query is forwarded at each step. This is accomplished by adding a breadth parameter B to the query. The query originator iterates over the breadth parameter $Q.B$ from $minB$ to $maxB$ ($minB$ and $maxB$ is a system defined parameter). In each iteration the query Q takes $maxTTL$ hops but is forwarded only to $Q.B$ neighbor peers at each hop. A main distinction of our algorithm, in contrast to other existing search schemes, is that, in a given iteration when the query is required to be forwarded to say n neighbors, conscious effort is put forth to ensure that the query is sent to the *best* subset of n neighbors, rather than *any* or all of the n neighbors. We use a capability-aware ranking algorithm to select the best B neighbors from a set of neighbor peers. The ranking algorithm also takes into account the performance of peers in the recent past and dynamically updates the rankings as peers join or leave the system.

The probabilistic selective routing algorithm comprises of three major components: *Ranking neighbor peers*, *Processing a query using the ranking information*, and *Maintaining the rankings up-to-date*. We below describe the ideas of the algorithmic design of these three components. Readers may refer to [12] for the algorithm details.

3.2 Ranking Neighbor Peers

In a P2P system, peers are constantly interacting with their neighbors as a part of their query forwarding responsibility. Also, the results of a query retrace the path to the query originator. Note that the results forwarded to peer p by its neighbor peer q not only includes the results from the local file index of peer q, but also includes the results from other peers which received the query via peer q. A peer p could build the following metrics to rank the *goodness* of a neighboring peer q based on its interactions with peer q:

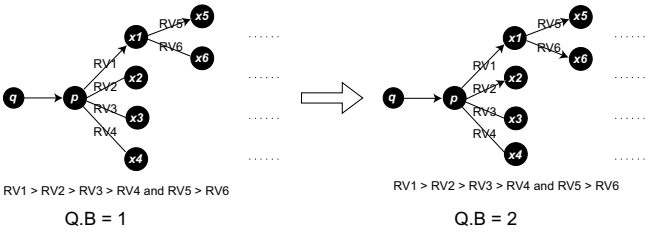

Figure 4: Probabilistic Selective Algorithm: Illustration

- Max Degree: Degree refers to the number of neighbors to which a peer is connected. Peers maintaining large degree can be expected to have high processing power and access network bandwidth.
- Max Results: The number of results returned per query by a neighbor over the last T time units.
- Max Rate: The ratio of the number of results returned per query to the time taken to return them over the last T time units.

3.3 Probabilistic-Selective Routing

Query processing using the probabilistic selective routing technique consists of three steps: (i) The query originator iterates through the breadth parameter $Q.B$ from $minB$ to $maxB$ until the query Q is satisfied. In each iteration (if the query is not yet satisfied), the query originator issues the query with the query's TTL set to $maxTTL$. (ii) When a peer p receives a query Q from peer q, peer p chooses the best $Q.B$ neighbors (excluding peer q) and forwards the query only to them. (iii) When a peer q receives the results for a query Q from its neighbor peer p it updates the goodness metric for peer p. Figure 4 illustrates our algorithm at peer p for $Q.B = 1$ and $Q.B = 2$ with ranking values $RV1 > RV2 > RV3 > RV4$ and $RV5 > RV6$.

3.3.1 Maintaining Rankings Up-To-Date

The key problem in maintaining the rankings up-to-date is the following: Say at time instant t, a peer p ranks its neighbors $X = \{x_1, x_2, \cdots, x_n\}$ as $x_1 \succ x_2 \succ \cdots \succ x_n$. Let the average breadth parameter ($Q.B$) at which a typical query is satisfied be b. So, peer p sends most of its queries to peers $x_1, x_2, \cdots, x_b$ at time t' ($t' > t$). Consequently, the ranking measures of peers $x_{b+1}, x_{b+2}, \cdots, x_n$ are not updated by peer p since peer p did not forward queries to (and thus not get results from) these peers. Hence, changes to the file indices of peers in the overlay network that are *accessible* through peers $x_{b+1}, x_{b+2}, \cdots, x_n$ are not considered subsequently.

To capture the dynamics of the P2P network we modify the *neighbor selection step* as follows: Instead of *deterministically* selecting the best $Q.B$ peers, peer p selects $Q.B$ peers *probabilistically*, that is, each of the neighbor peer x_j is selected with a probability proportional to its ranking value $RV(x_j)$. Hence, most of the queries get routed through the neighbors who have performed well before; yet, by probabilistically sending the query to some inferior neighbors, peer p can figure out if they can provide us better results *now*.

4 Experimental Results

We have performed three sets of simulation-based experiments. The first set compares the three classes of multi-tier topologies against a random topology using our performance metrics. The second set evaluates our model for constructing multi-tier capability-aware topologies against a random topology. The third set studies the effectiveness of probabilistic routing in handling vastly heterogeneous nodes and trade-offs in probabilistic-selective routing algorithm.

4.1 Simulation Set Up

We implemented our simulator using a discrete event simulation [4] model. The end users send queries to the system at an exponentially distributed rate λ. The latency of a link from node n to node m is modeled as being inversely proportional to $min(C_{HL(n)}, C_{HL(m)})$. The processing time of a query at a node is modeled as a constant; especially since the amount of computation power available to nodes is almost homogeneous. Our simulation setup comprises of three main modules: the Network Generator, the Document Generator, and the Routing Simulator. Tables 1, 2 and 3 present the major tunable parameters used in these modules. In all these modules we chose the parameter settings from several observations made on the real Gnutella network [11, 2].

Network Generator. We use our multi-tier bootstrap algorithm and construct the overlay network incrementally by adding one node at a time. For our simulation we use 20%, 70% and 10% of the peers for the corresponding HL = 0, 1 and 2.

Document Generator. We use Zipf-like distribution wherein the number of documents that match the i^{th} most popular query is proportional to $1/i^{\alpha}$ (with $\alpha = 1$). We use *Document Bias (Db)* to specify non-uniform distribution of documents amongst peers (Db of 20%-80% means that 20% of the peers hold about 80% of the documents).

Routing Simulator. The routing simulator implements one of the following routing algorithms: broadcast styled routing and probabilistic-selective routing.

4.2 Evaluation of CATs

We compare the performance of a hierarchical, sparse, dense and random overlay topology using five performance metrics: Amortized messaging bandwidth, Load Distribution, Coverage, Amortized Latency, and Fault Tolerance using the broadcast style routing algorithm with $initTTL$ equal to 7. We conclude this section with an experiment that shows the importance of identifying the correct number of heterogeneity levels (peer classes) in obtaining good performance.

Amortized messaging bandwidth. Figure 5 presents the

Parameter	*Description*	*Default*
N	Number of Nodes	10,000
HL	Number of Peer Classes	3
C	Node capability	8:4:1
$Maxd$	Maximum Peer Degree	10
$Mind$	Minimum Peer Degree	1
Dd	Degree Distribution (Random/Power Law)	Power Law

Table 1: Network Generator

Parameter	*Description*	*Default*
Nd	Number of Documents	100,000
Ndd	Number of Distinct Documents	10,000
Db	Document Bias	20%-80%

Table 2: Document Generator

Parameter	*Description*	*Default*
THR	Result threshold	32

Table 3: Search Simulator

amortized messaging bandwidth consumption on various topologies. For $N = 10,000$ nodes, the hierarchical topology consumes the least amortized messaging bandwidth since its tree-like structure minimizes the number of duplicate queries. Also, the multi-tiered sparse and dense topologies reduces the CAB requirement on nodes by about **6 to 10 times** when compared to a random topology because they require that the weak nodes send/receive far fewer messages when compared to strong (more capable) nodes. This clearly demonstrates our claim that capability-aware topologies indeed reduce the amortized messaging bandwidth requirement significantly.

Load Distribution. Figure 6 shows the variation of load distribution among the four overlay topologies that are normalized with respect to a $N = 1,000$ node random topology. For $N = 10,000$ the sparse and the dense topology show **80 to 100 times** lower variance, while a hierarchical topology shows 10 times lower variance.

Coverage. We measured the distribution of the coverage with respect to the heterogeneity level of peers. Table 4 shows the variation of coverage with different HLs of peers that are normalized with respect to $HL = 0$ peers. One key conclusion drawn from this table is that the extra work done by the higher HL peers rewards them with larger coverage (and thus more results). Further, this property of *fairness* acts as an important motivation for the high capability nodes to do more work for the system.

Amortized Latency. Figure 7 shows the amortized latency (equivalently, the response time experienced by an end-user) of the four overlay topologies. Recall that we had, for simplicity, modeled the latency of a link from node n to node m to be inversely proportional to $min(C_{HL(n)}, C_{HL(m)})$. In a random topology, the presence of many weak nodes on the paths between two powerful nodes is mainly responsible for its latency to be about twice as large as that of capability-aware topologies. Further, among the capability-aware topologies, the hierarchical topology shows about 5% and 12% lower latency than the sparse and the dense topologies; one could attribute this trend to the increasing *randomness* in the topology from a hierarchical topology to the sparse and the dense topologies.

Fault Tolerance. We study the fault-tolerance of the four topologies under two conditions: *Uniform Faults* and *Non-Uniform Faults*. Figure 8 shows the quality of the results obtained when a random 10% of the peers fail under uniform faults. Quality of results is expressed as the ratio of the number of results obtained under faulty conditions to that obtained when all the peers were functional. For $N = 10,000$ nodes, the hierarchical topology shows about 50% lower fault-tolerance than the random topology because of its delicate tree-like structure; while the sparse and the dense topologies exhibit only 2-3% lower fault-tolerance because they assign more responsibility to higher level nodes; recall that in a random topology all nodes are treated as the same and hence are assigned equal responsibilities.

Figure 9 shows the fault-tolerance of the topologies under non-uniform faults with 10% of failed peers, where the probability of peer failure at HL = 0, 1 and 2 are in ratio 4:3:1 [2, 11]. For $N = 10,000$ nodes, the sparse and dense topology show about 4-5% more fault-tolerance than the random topology and the hierarchical topology shows 20% improvement as against the uniform faults case. This improvement is primarily because the multi-tiered topologies assign *more responsibility* to more capable nodes which are *less likely to fail* (higher level nodes).

Importance of Choosing Correct *numHL*. To demonstrate the importance of determining the correct value of $numHL$, we compare the performance of a system with an incorrect value for $numHL$ to that of a system which used the right value. Assume that the number of genuine peer classes in the system is three. We constructed a hierarchical, a sparse, and a dense topology using these peers for $numHL = 2$ (by merging the top two peer classes) and $numHL = 3$. Table 5 shows ratios of the performance measures obtained for $numHL = 2$ to those obtained for $numHL = 3$. For instance, a sparse topology with $numHL = 2$ shows 1% lesser coverage, **3.5 times** more bandwidth (CAB), 16% more latency, 18% lesser fault-tolerant, and **11 times** more variance in load distribution than a sparse topology with $numHL = 3$. Nevertheless, observe that the results obtained for $numHL = 2$ is better than a random topology because the former is *more* capability-aware.

4.3 Evaluation of CAR

In this section we present a detailed evaluation of our capability-aware routing algorithm. We first show the benefits of CAR on both a random topology and a multi-tier topology. Second, we compare our routing algorithm with other routing algorithms popularly known in literature, such as, iterative deepening (*rdfs*) [13] and broadcast-styled routing (*bsr*). Third, we show the ability of our routing algorithm to adapt itself to the dynamics of the P2P network.

Performance Enhancements by CAR. Figure 10 shows the

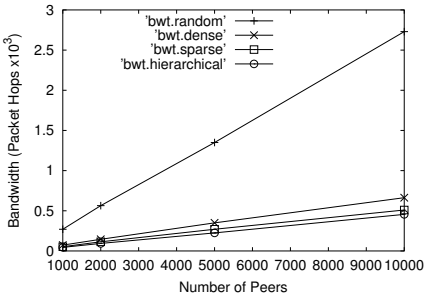

Figure 5: Amortized messaging bandwidth

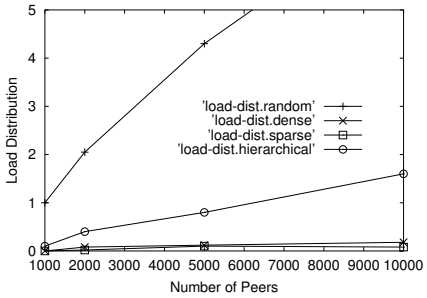

Figure 6: Variance of load/capability

Topology	$HL = 0$	$HL = 1$	$HL = 2$
Hierarchical	1.0	5.2	15.3
Layered Sparse	1.0	3.6	7.7
Layered Dense	1.0	2.9	6.0

Table 4: Coverage vs HL for $N = 10,000$

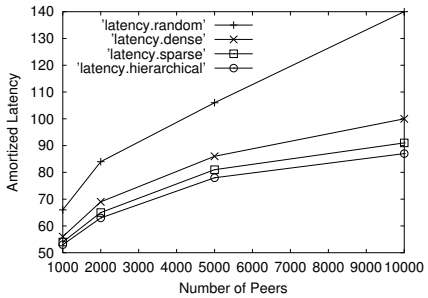

Figure 7: Amortized Latency

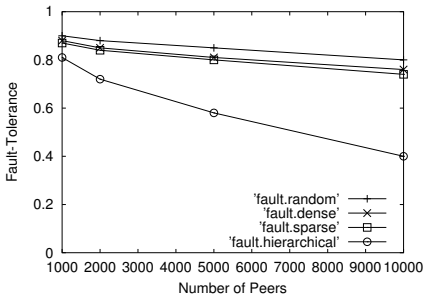

Figure 8: Fault Tolerance with uniform faults

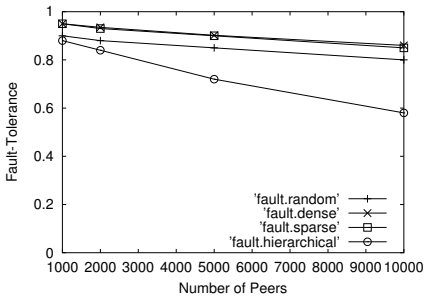

Figure 9: Fault Tolerance with non-uniform faults

mean bandwidth (CAB) expended for the execution of a query. For a random topology, CAR shows about 30% reduction in bandwidth, while for a capability-aware multi-tier topology, CAR shows about 35% reduction in network bandwidth.

Comparison of Routing Algorithms. Table 6 shows the performance of several routing algorithms over a random topology and a three-tiered sparse topology for several values of the search threshold. The performances of the routing algorithms are measured in terms of amortized messaging bandwidth (based on CAB) and amortized latency (shown within brackets). Note that PSR denotes our probabilistic-selective routing algorithm, ITR-DEEP denotes the iterative-deepening technique [13], BIASED-RW denotes a biased random-walk algorithm [9] and BSR denotes the broadcast-styled routing algorithm. In general the capability-aware topologies (CAT) consume about 5-7 times lower amortized messaging bandwidth than the random topologies (RT).

5 Related Work

In the past few years research on P2P systems has received a lot of attention. Several research efforts have been targeted at improving the performance of P2P search [13, 3, 9]. These papers suggest enhancements over the naive flooding-based algorithm by fine-tuning their search schemes based on measurement studies conducted on user characteristics, distribution of files among peers, etc. For example *Routing Indices* in [3] exploits the locality of the queries and the uneven distribution of different categories of files among peers.

Several have pointed out that peer heterogeneity would be a major stumbling block for Gnutella [10, 11]. Solutions based on super-peer architectures have been proposed in [7] to alleviate the problem of peer heterogeneity. The super-peer architecture can be viewed as a hierarchical topology with $numHL = 2$. Our work not only generalizes $numHL$ to arbitrary values, promoting multi-tiered sparse topology over the hierarchical topology, but also provides an analytical model that yields the desired degree information to precisely construct capability-aware overlay topologies.

Chawathe *et al* [2] suggest the use of dynamic topology adaptation that puts most nodes within short reach of high capacity nodes. Their topology adaptation scheme defines a level of satisfaction and ensures that high capacity nodes are indeed the ones with high degree and that low capacity nodes are within short reach of higher capacity ones. However, it constructs and maintains topology using ad hoc mechanisms which not only increases the topology maintenance cost (to maintain the level of satisfaction for every node) but also runs into the risk of building disconnected topologies, each of whose components are like the *small world groups* observed in Gnutella. Also, the performance gains reported in [2] are primarily due to the fact that they use one hop replication and allow nodes to maintain as many as 40-60 neighbors. From well-known observations on Gnutella, it is observed that even the powerful nodes maintain only about 10 neighbors. Hence, we believe that maintaining such a large number of connections on behalf of one application is unreasonable.

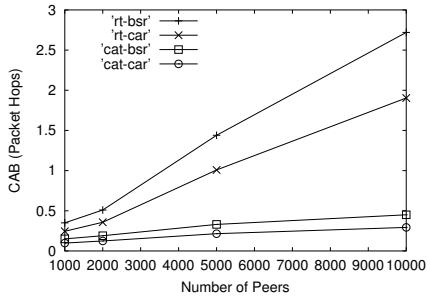

Figure 10: CAR Benefits

Topology Type	*Coverage*	*Amortized Bandwidth*	*Amortized Latency*	*Fault Tolerance*	*Load Dist*
Hierarchical	0.97	3.06	0.96	0.96	21.0
Sparse	0.99	3.49	1.16	0.82	11.0
Dense	0.99	3.33	1.12	0.81	9.0

Table 5: Importance of choosing correct $numHL$

Search THR	CAT-PSR	CAT-ITR-DEEP	CAT-BIASED-RW	CAT-BSR	RT-PSR	RT-ITR-DEEP	RT-BIASED-RW	RT-BSR
1	2.11 (7.40)	1.75 (1.75)	1.45 (1.82)	5242 (24.03)	7.53 (10.53)	6.33 (2.54)	4.62 (3.69)	9031 (35.87)
5	7.50 (8.41)	8.53 (8.87)	7.03 (7.95)	5242 (24.03)	35.15 (12.65)	42.17 (13.74)	20.53 (15.07)	9031 (35.87)
10	15.63 (12.73)	17.94 (13.73)	14.00 (20.83)	5242 (24.03)	68.39 (18.21)	78.97 (19.83)	56.74 (30.85)	9031 (35.87)
20	31.01 (18.23)	36.34 (20.01)	27.05 (50.34)	5242 (24.03)	130.63 (26.74)	150.33 (30.15)	104.98 (61.32)	9031 (35.87)
30	50.93 (23.06)	62.93 (24.62)	40.83 (100.53)	5242 (24.03)	200.73 (32.53)	240.93 (33.83)	154.35 (123.65)	9031 (35.87)

Table 6: Comparison of Routing Algorithms: $N = 10,000$, $numHL = 3$ and $initTTL = 7$

6 Conclusion

The key problem that has plagued a Gnutella like P2P systems is *Peer Heterogeneity*. We have proposed simple yet effective multi-tier capacity-aware topologies for improving the P2P search performance. There are three main contributions in this paper. First, we propose techniques to structure overlay topologies taking peer heterogeneity into account; such capacity aware topologies ensure that the performance of the P2P system is not hindered by less powerful peers. Second, we developed an analytical model to enable us to construct and maintain capacity-aware overlay topologies with good node connectivity and better load balance. Third, we proposed a probabilistic routing algorithm that further reduces the bandwidth consumption of our P2P system. We used extensive simulation-based experiments and mathematical analysis to construct and evaluate the goodness of capacity-aware topologies and routing algorithms over random topologies and broadcast-styled routing algorithms. Finally, our design and techniques being simple and pragmatic can be easily incorporated into existing systems like Gnutella.

References

[1] E. Adar and B. A. Huberman. Free riding on gnutella. http://www.firstmonday.dk/issues/issue5_10/adar, 2003.

[2] Y. Chawathe, S. Ratnaswamy, L. Breslau, N. Lanham, and S. Shenker. Making gnutella-like p2p systems scalable. In *ACM SIGCOMM 2003*, 2003.

[3] A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *Proceedings of International Conference on Distributed Computing Systems*, July 2002.

[4] G. S. Fishman. Discrete-event simulation. Springer Series in Operations Research.

[5] Freenet. The free network project. http://freenet.sourceforge.net/, 2000.

[6] Gnutella. The gnutella home page. http://gnutella.wego.com/, 2002.

[7] F. S. Inc. Super-peer architectures for distributed computing. http://www.fiorano.com/whitepapers/ superpeer.pdf, 2002.

[8] Kazaa. Kazaa home page. http://www.kazaa.com/, 2003.

[9] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *16th annual ACM International Conference on supercomputing*, 2002.

[10] S. R. Qin Lv and S. Shenker. Can heterogeneity make gnutella scalable? In *Proceedings of the first International Workshop on Peer-to-Peer Systems*, 2002.

[11] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. Technical Report UW-CSE-01-06-02, University of Washington, 2001.

[12] M. Srivatsa, B. Gedik, and L. Liu. Improving peer to peer search with multi-tier capability-aware overlay topologies. Technical report, Georgia Institute of Technology, 2003.

[13] B. Yang and H. Garcia-Molina. Improving search in peer-to-peer networks. In *22nd International Conference on Distributed Computing Systems (ICDCS'03)*, July 2002.