A Utility-Aware Middleware Architecture for Decentralized Group Communication Applications

Jianjun Zhang[1], Ling Liu[2], Lakshmish Ramaswamy[2], Gong Zhang[1], and Calton Pu[1]

[1] College of Computing, Georgia Institute of Technology,Atlanta, GA 30332,USA,
[2] Department of Computer Science, University of Georgia, Athens, GA30602, USA
{zhangjj,lingliu,gzhang3,calton}@cc.gatech.edu,laks@cs.uga.edu

Abstract. The success of Internet telephony services like Skype illustrates the feasibility of utilizing unstructured Peer-to-Peer (P2P) networks as an economical platform for supporting group communication applications. However, the ad-hoc nature of these networks poses significant challenges to the efficiency and scalability of the group communication services. This paper presents the design and implementation of GroupCast − a utility-aware middleware architecture for scalable and efficient P2P group communications. The GroupCast design is characterized by three unique features. First, we present the *utility function* for quantifying the role of unicast links in enhancing the scalability and efficiency of the group communication applications. This utility function provides a careful combination of the two most important performance factors, namely *relative network locations* and *resource capabilities* of the end hosts. Second, we develop a utility-aware distributed spanning tree construction algorithm for efficiently propagating group communication messages. It dynamically creates and maintains the group communication channels by optimizing the utility value of the group communication spanning trees. Third, we propose a utility-based overlay management protocol for constructing and maintaining low-diameter overlay networks to further enhance the performance of the group communication services. We evaluate the effectiveness of the GroupCast middleware architecture through analytical and experimental analysis of the costs and benefits of the proposed techniques. Our experimental results show that the GroupCast system can improve the scalability of wide-area group communication services by one to two orders of magnitude.

Key words: Middleware, peer to peer system

1 Introduction

Multi-party group communication applications such as multiplayer online games, online community based advertising, real-time conferencing [3], and instant messaging [2] have experienced a surge of popularity in the past few years. The

applications are characterized by exchanges of textual or multimedia contents among multiple participants. A large number of existing group communication systems achieve the required scalability through computer cluster-based server farms [3, 2]. These server farms are exclusively owned by service providers and they require significant investment. Subscribers are usually required to pay a premium for scaling the services [3]. Further, the features of server farm-based group communication applications are often limited by the capabilities of the respective service providers.

Decentralized Peer-to-Peer (P2P) networks have evolved as a promising paradigm for providing open and distributed information sharing services by harnessing widely distributed, loosely coupled, and inherently unreliable computer nodes (peers) at the edge of the Internet. The success of Skype [5] has demonstrated both the opportunity and the feasibility of utilizing P2P networks as economical infrastructures for providing wide-area group communication services. In Skype, widely distributed personal computers are interconnected by an unstructured P2P overlay network. Skype exploits the high flexibility and low maintenance cost of such an open system architecture to support value-added services like Internet-to-PSTN (Public Switched Telephone Network) calls at a fraction of the price of traditional telephone services. However, the overlay networks in Skype are used *only* for service lookup and control signaling. Under the multi-party conference settings, each node is required to send the payloads directly to other participants of the communication group through its IP unicast links [9]. This places severe limitations on the scalability of multi-party conference calls in Skype. The first release of Skype only supported group communication among 6 or less participants [9].

The natural questions that come up include: *Can P2P overlays be utilized for implementing scalable group communication services over wide-area networks? and if so what techniques and system level optimization are critical for enhancing the efficiency and scalability of decentralized wide area group communications?* Although several researchers have explored a related problem in the context of designing application-level multicasting or end-system multicasting (ESM) schemes [22, 24] over P2P overlays, surprisingly most of these works are designed to work in conjunction with structured P2P networks, and they rely on the distributed hash table (DHT) abstractions of the P2P network for inter-peer communication and routing [11, 23]. However, it is widely recognized that in environments that exhibit high *churn* rates maintaining DHT-based structures imposes severe overheads, which can affect the performance of the applications running on top of these networks to a considerable extent [13]. In contrast, *unstructured P2P networks* like Gnutella [31] are simple to implement, having low maintenance costs, and providing better resilience to network churn caused by peer entries, exits, and failures. To the best of our knowledge, very few group communication applications have been implemented on top of unstructured P2P networks. None, including Skype, has proposed a scalable group communication middleware architecture for supporting large number of participants in an unstructured P2P network. We hypothesize that common concerns about the

non-deterministic nature of communication and service lookup in unstructured overlay networks and their inefficient utilization of the underlying IP network resources are the main reasons for the lack of work in this area.

Designing scalable group communication services on top of unstructured P2P networks poses three main challenges. The first challenge is to translate wide-area group communication application requirements, such as communication efficiency, load balancing, and system scalability, into the metrics that can be used while designing the communication structures and managing the topologies of the overlay networks. Second, the communication and service lookup in unstructured P2P networks impose heavy messaging overheads. Further, these networks also suffer from high service lookup latencies. Although a number of optimizations have been proposed to improve the efficiency of unstructured P2P networks [13, 4], few of them are designed for scaling wide-area group communication services. The challenge is to devise low-cost service lookup mechanisms that are effective for both control signaling and communication group management. Furthermore, unstructured P2P overlay networks evolve in a random manner. The resilience of these networks to network churn is rooted in the fact that they do not use any global control mechanisms for regulating resource distribution and the network topology. The third challenge is to design overlay network management protocols such that we incorporate features critical to the performance of group communication applications without trading away the inherent randomness of the unstructured P2P networks.

Towards addressing these challenges, this paper presents *GroupCast* - a utility aware decentralized middleware architecture for scalable and efficient P2P group communication applications. In our system, the P2P network serves not only as a signaling overlay, but it also carries group communication payloads through spanning trees that are composed of unicast links interconnecting the participants nodes. In designing the GroupCast system, this paper makes three unique contributions:

- First, we propose a utility function to quantify the usefulness of unicast links to the efficiency of individual communication groups as well as to the scalability of the entire group communication infrastructure. This utility function provides a careful combination of the two most important factors that influence the performance of the system, namely *network proximities of the peers* and *resource availabilities at the end hosts*.
- Second, we design a *utility-aware* mechanism for constructing spanning trees required for disseminating group communication payloads. The objective of this scheme is to optimize the utility-values of the resultant spanning trees. Further, considering the decentralized nature of unstructured P2P networks, this scheme has been designed to operate in a completely distributed fashion, and it does not rely upon any global topological information.
- Third, we present a *utility-based* P2P overlay network management protocol that uses the proposed generic utility function for constructing low-diameter unstructured P2P overlays. This protocol generates overlay networks that are comparable to structured P2P network in their scalability and efficiency

while retaining the randomness and low maintenance overheads of generic unstructured P2P networks. We demonstrate that such a utility based overlay network is critical for group communication applications.

We have performed several experiments to evaluate the proposed utility-aware middleware architecture and its component techniques. The results show that our utility-aware spanning tree construction scheme and our utility-based overlay management protocols provide significant scalability and efficiency benefits for the group communication applications.

2 The Basic P2P-based Group Communication Framework

In this section, we describe our framework for group communication in unstructured P2P networks. *Spanning tree* forms the fundamental structure in most group communication schemes. The spanning tree is an acyclic overlay connecting all the participants of a communication group. The group communication messages (payload) are disseminated through the spanning tree so that they reach all the participants. The various group communication schemes differ in manner in which they construct and maintain the spanning trees. For instance, traditional client/server architectures can be viewed as a special spanning tree of height 1 with server forming the root of the tree. However, it is obvious that such an implementation has very poor scalability. Our system employs multi-level spanning trees for achieving the scalability needed for supporting group communication in large wide-area networks. The proposed framework includes completely distributed strategies for building and maintaining spanning trees of communication groups.

We introduce a few notations that would be used in the rest of the paper. The P2P network is conceptualized as a directed graph $G < V,\ E >$, where $V = \{p_0, p_1, p_2, \ldots, p_{N-1}\}$ represents the peers in the network and $E = \{e_0, e_1, \ldots e_{M-1}\}$ denotes the logical links in the network. The spanning tree $T_Pt < V_{Pt},\ E_{Pt} >$ could be defined as a connected, acyclic sub-graph of G, where the participant set $V_{Pt} \subseteq V$ and links set $E_{Pt} \subseteq E$. Each peer is aware of only its immediate neighbors. A global view of the network is not maintained in the system. Further, the network does not have any distributed hash table (DHT) abstractions.

2.1 Constructing a Distributed Spanning Tree

One of the challenges in developing group communication systems is to design a completely distributed scheme for building spanning tree. Several application level multicast (end-system multicast) systems have addressed a very similar problem (the problem of constructing multicast trees). However, as we explain below, none of them are directly applicable for building spanning trees on unstructured P2P networks.

The existing multicast tree construction schemes can be classified into three broad categories. In the first approach, the participants of a multicast group explicitly choose their parents in the multicast tree from a list of candidate nodes. Examples of such systems include NICE [7], Overcast [19], and Yoid [18]. Due to the complexity of those protocols, there are very few actual implementations of these systems. The second approach, which is adopted by systems like Narada [14] and Scattercast [12], constructs the spanning tree in two-steps. The first step constructs a well-connected mesh from the nodes in the network. The second step uses this mesh structure and constructs shortest path spanning trees through well-known distributed algorithms. These systems usually use extensive messaging to maintain the quality of the mesh network, which is critical to the performance of the multicast tree. Consequently, these systems do not scale well, especially when the underlying network experiences considerable churn. The third approach is represented by systems like CAN-multicast [23] and SCRIBE [11]. These systems assume that the nodes of the underlying network are organized as a structured P2P network [22, 24]. The multicast tree is constructed using the deterministic routing functionalities of these P2P networks. Because of the structured nature of the underlying network, these systems can incorporate techniques for optimizing various system parameters [32, 24, 35]. As we discussed in Section 1, DHT-based structured P2P networks are not suitable for scenarios wherein the peer populations are transient. Transient peer populations may degrade the performance of the applications built on top of structured P2P networks. In short, none of the current multicast tree construction approaches are applicable for the problem at hand.

We have developed completely decentralized scheme for building group communication spanning trees on a generic unstructured P2P network. We leverage techniques such as selective message forwarding for reducing the communication costs of spanning tree construction and maintenance. Our experiments (see Section 4) show that the quality of the resultant spanning trees are comparable to those built using the other three approaches.

2.2 Building the Communication Group

The objective of our communication group construction algorithm is to select the edges or the links in the P2P overlay to form the spanning tree that connects all the group participants. The implementation of communication group construction algorithms usually includes the implementation of two functionalities. First, participants should be aware of the existence of the communication group to which they will join. Second, a newly-joined participant should be able to setup a connection to the existing nodes in the chosen communication group for sending and receiving the communication payloads.

In the literature of end-system multicast (especially in the systems adopting two-step approaches that we discussed earlier) the first task is solved by appointing a node as the rendezvous point or the multicast source, and publishing its information at a well-known location such as a bulletin board system. The other participants will use the identity of this node as the keyword to establish the

multicast path, usually by leveraging the search interface provided by the P2P overlay network.

Two strategies have been proposed for implementing the second functionality. The first scheme is similar to the DVMRP [16] IP-multicast protocol. Instead of using the IP level network devices such as routers to implement the polling and pruning processes of multicast group management, this strategy uses overlay networks and peers. This strategy is adopted by the Scattercast system [12], in which the source node solely advertises route information, and each node in the overlay forwards this advertisement and builds the local routing table entries. To remove loops and to avoid the problem of counting-to-infinity, the full path information is embedded into the forwarded advertisement messages. For the purpose of comparison, we refer to this scheme as *Non-Selective Service Announcement*(NSSA) scheme in the rest of this chapter.

The second scheme is adopted by systems like SCRIBE [11]. The multicast source is mapped to a well-known node serving as the rendezvous point. Subscribers use the identifier of the rendezvous point as the keyword in their subscribing requests. The P2P overlay treats subscription requests in the same manner as the routing requests. The structured system topology and the deterministic routing algorithms decide the series of peers through which each subscription request would be forwarded so that it reaches the rendezvous point or an existing participant in the multicast group. The reverse of this path would be used for forwarding the multicast payloads down from the multicast source.

Two characteristics of our system prevent us from directly reusing these schemes. First, the nature of group communication applications is different from end-system multicast systems. In end-system multicast systems, communication payloads are forwarded in one direction in most of the cases (from the multicast source to all the other nodes), while in group communication systems, each participant may initiate messages in addition to receiving them. Second, the unstructured nature of our P2P overlay prevents us from directly using the reverse of the searching path as the payload communication path. Because of the lack of DHT abstractions and the random nature of the overlay topology, searching has to be carried out either by flooding the request or through random walks. The former approach results in heavy communication overheads, whereas the latter may generate very long search paths which would affect the communication latencies of the payloads.

We have proposed a scheme that combines the advantages of these two schemes, while avoiding their disadvantages. We call our scheme *Selective Service Announcement* (SSA) scheme. In this scheme, the spanning tree for a communication group is established in three steps.

Step 1: Choosing Rendezvous Point First, a peer in the P2P overlay is chosen as the rendezvous point. Unlike the rendezvous point in SCRIBE [11], to which all the multicast payloads are first forwarded, our rendezvous point serves as the source of the group advertisement messages and will behave as a normal node in the communication spanning tree. There are several ways to choose such a rendezvous point. It can be setup as a dedicated server donated by a service

provider who injects contents into the communication group. For groups that are setup for applications like online conferences, the first participant can initiate a random walk search to locate a node that has enough access network bandwidth and computational power to act as a rendezvous point.

Step 2: Advertising In the second phase, the rendezvous point advertises the group information to the potential participants of the communication group. The flooding scheme used for similar purposes in DVMRP [16] and Scattercast [12] will incur redundant messages in the overlay network, especially when the peer population is large and the communication group is relatively small. Our SSA scheme alleviates the communication overheads in the following manner. In our scheme, each peer that receives the advertisement message will forward it to a few of its neighbors, rather than flooding the message to all neighboring nodes. By filtering out the neighbors that will not likely be used in the communication spanning tree, we can reduce the number of messages by as much as 65%, compared to the NSSA scheme. Our basic group communication framework uses a very simple approach for selecting neighbors, namely random strategy. In this algorithm the rendezvous point randomly selects a pre-specified fraction of its neighbors and sends them the advertisement message. This process is repeated by every node which receives the message. At each step, the TTL is decremented and the message propagation terminates when the TTL becomes zero. However, this simple advertisement scheme suffers from two major drawbacks that can adversely impact the performance of group communication. We discuss these limitations later in the paper and present schemes for mitigating them.

Step 3: Subscribe Subscription activities are initiated when a peer i decides to join a communication group. Two scenarios need to be considered. First, if the potential service subscriber (peer i) has already received and routed the service advertisement, peer i is already on the message forwarding path of this communication group. All it need to do is to start the subscription process by sending the joining message in the reverse direction of incoming SSA message. However, note that the advertisement message might not reach all potential subscribers. In case the subscriber has never received the SSA message, a search method provided by the P2P overlay is triggered to look up the neighborhood of the peer for discovering nodes that might have received the SSA advertisement message.

The search method is implemented as a ripple search in standard Gnutella P2P network, with initial TTL (Time to Live) value set to a very low value. Because our advertisement mechanism would have already *pushed* the service information to different topological regions of the network, a potential subscriber can find a nearby neighbor that has received the SSA message with high probability. Our experiment reports that the average success rate of subscription search is as high as 100%, even when the TTL of the search messages are set to 2. Once such a node is discovered, the subscription message is sent to it, which then forwards in the reverse direction of the original SSA message.

2.3 Limitations of the Basic Framework

The basic group communication framework has two important limitations which can affect its efficiency and scalability. The first limitation is the manifestation of the overlay-underlay mismatch problem. Since, in the advertisement phase of the scheme, a node receiving the advertisement forwards the message to a *randomly* chosen subset of neighbors, the resulting tree might not always be efficient in terms of the relative locations of its nodes on the physical network. For example, a node p_i located in New York might have a node p_j located in Australia as one of its children, which in-turn might have a child p_l located in Boston. This has a negative effect on the latencies experienced by the group communication messages.

For the very same reason of random advertisement forwarding, the capability (resource availability) of a node p_i in the spanning tree might be completely different from the capabilities of its parents or children. This mismatch among the capacities of the neighbors in the spanning tree can result in high packet losses. This again affects the performance of group communication.

We propose two middleware level techniques for overcoming the above drawbacks, namely a utility-aware spanning tree construction scheme and a utility-aware topology management scheme for the underlying P2P network. While the first technique addresses the question as to *how should the connections in the overlay be utilized for group communication applications?*, the second technique addresses the question of *how the peers should choose and maintain their neighbors in the overlay?* However, it is interesting to observe that these two questions are the manifestations of the same design issue, namely *given a list of nodes, say L, what are the metric(s) that dictate which of these nodes a peer p_i should connect to?* Both these techniques rely upon a unique utility-function, which assigns different preferences (rankings) to each peer in the list L. In the next section we explain the formulation of the utility function. We then describe how this utility function is utilized in the proposed techniques.

3 The Utility-Aware Middleware for P2P Group Communication

This section we present the design of GroupCast middleware architecture for decentralized group communication services. We focus on three main components of the GroupCast design. First, we describe the utility function we use to quantitatively model the critical performance metrics of wide area group communication applications. Second, we discuss how to employ our utility function to optimize the group communication channel construction and maintenance by developing a utility-aware distributed spanning tree construction algorithm that can efficiently propagate group communication messages. We show how it optimizes the utility value of the group communication spanning trees. Finally, we present our utility-based overlay management protocol which provides the capability for constructing and maintaining low-diameter overlay networks to further enhance the performance of the group communication services.

3.1 The Utility Function

The group communication in overlay networks essentially occurs by forwarding the communication payload through unicast IP network links. Hence, the properties of the unicast links interconnecting peers in the P2P overlay largely decide the performance and the efficiency of the group communication system. Our utility function considers the two important factors that determine the performance of unicast links, namely the network proximity of the end-nodes and the similarity between among the capacities of the peers. The network proximity between the end-hosts determines the latency of the unicast link. Similarly, it is known that mismatch between the packet-forwarding workloads and the capacities of peers introduces bottlenecks in the communication overlay and may result in high packet losses. We note that these two factors might sometimes be counteracting. For instance, a peer in the list L which is closest to p_i, might have completely different resource availabilities than p_i. Our utility function provides a careful combination of these two factors based on the utility preference of peer p_i, as well as the desired performance properties of the entire overlay.

Concretely, for each node p_j in the list L (recall that L represents a list on potential nodes from which the peer p_i chooses a subset), we assume that two types of information are available: the node capacity C_j, and the relative distance between peer i and peer j, denoted by $D(i, j)$. The capacity of a peer is measured in terms of its accessible network bandwidth, since the performance of a peer in a distributed environment like P2P networks is largely decided by its access network bandwidth available for forwarding communication payloads. The access network bandwidth can be specified by the end user in terms of the number of 64kbps connections the node is willing to support. Alternatively, it can also be estimated by network probing techniques. We use the network coordinates to estimate the relative distance any two peers between peer j and peer p_i. Vivaldi [15] and GNP [1] are some of the techniques proposed for measuring the network coordinates of nodes in wide area networks.

We define two utility-based preference metrics based on the two important performance factors namely network proximity and node capacity. Given a list of peers L, we define the $Distance\ Preference$ of peer p_i to peer $j \in L$ as the probability that peer p_i chooses peer p_j out of L, based on the network coordinate distance between them. The closer the peer p_j is to peer p_i, the more likely it is chosen. The $Distance\ Preference$ is computed as indicated in Equation 1.

$$DP_{p_i}(L, p_j) = \frac{\frac{1}{d_{p_i}(L,p_j)} - \alpha}{\sum_{p_k \in L} \frac{1}{p_{d_i}(L,p_k)} - \alpha} \tag{1}$$

where $\alpha \in (-\infty, 1)$ is a tunable parameter that indicates the degree to which p_i's prefers closer peers. Higher values of α indicates that p_i strongly prefers closer peers and vice-versa. We choose $\alpha < 1$ so that there is nonzero preference on each $p_j \in L$. The function $d_{p_i}(L, p_j)$ gives the *normalized distance* between

p_i and p_j. $d_{p_i}(L, p_j)$ is defined as follows:

$$d_i(L, j) = \frac{D(i, j)}{\text{MAX}_{k \in L} D(i, k)} \tag{2}$$

Note that $0 < d_{p_i}(L, p_k) \leq 1$ for each peer p_k in the list L.

Similarly, we define the *Capacity Preference* utility metric of peer p_i with respect to peer p_j as the probability that peer p_i chooses peer p_j out of L based on the node capacity of peer p_j. The goal is to utilize higher capacity nodes to relay group communication messages to larger number of peers. Equation 3 gives the formulation for the *Capacity Preference* utility metric.

$$PC_{p_i}(L, p_j) = \frac{C_{p_j} - \beta}{\sum_{p_k \in L} C_{p_k} - \beta} \tag{3}$$

Here C_{p_j} is the node capacity of the peer p_j. The parameter $\beta \in (-\infty, 1)$ plays a similar role as that of α in equation 1.

While the *Capacity Preference* and *Distance Preference* encapsulate the utility of nodes in L from two different perspectives, we need a means to combine these two utility parameters into a single utility function. In this regard, it is interesting to observe that the peer p_i which wants to select a subset of peers from L should also consider its own resource availability (capacity) while making its choices. If the peer p_i possesses more resources, we would like to use it as a forwarding hub in the overlay network and applications. Such a peer should be connected to those peers that have similar resources and play similar roles in the overlay network, which would make it a member of the "core" of the overlay network. On the contrary, if the resources of peer p_i are limited, it should not be placed into the "core" as that would easily exhaust its resources. A better choice for such a limited resource peer would be to connect to peers that are physically closer to it and use them to access the overlay network. Hence, the weightage given to the two utility metrics (*Capacity Preference* and *Distance Preference*) depends upon the capacity of peer p_i. Accordingly, we define the combined utility function *Selection Preference* of peer p_i to peer $p_j \in L$ as a weighted combination of *Capacity Preference* and *Distance Preference*.

$$P_{p_i}(L, j) = \gamma \cdot PC_{p_i}(L, p_j) + (1 - \gamma) \cdot PD_{p_i}(L, p_j) \tag{4}$$

Here γ is the weightage factor such that $0 \leq \gamma \leq 1$. Substituting the formulae for *Distance Preference* and *Capacity Preference* into Equation 4, we obtain:

The configurable parameters α, β, and γ gives us the flexibility to fine-tune the utility function for different application scenarios. For instance, in an overlay network supporting applications that are sensitive to network proximity, α can be set to higher values and γ to be set to lower values. This would ensure that that network proximity is the dominating factor when peers make their choices. On the contrary, for an overlay network that emphasizes more on load balancing, a higher value for β and a higher value for γ would be more preferable.

The values of parameter α, β, and γ can be mathematically derived by using techniques similar to the ones used by Bu and Towlsey [10]. However, these techniques require information about the exact number of peers and the exact distributions of the various system-level parameters. In a decentralized environment like P2P networks where global statistical mechanisms are expensive to implement, it is unlikely that such information would be available. The GroupCast system adopts an approximation approach to address this problem. Specifically, we define *Resource Level* r_i as the fraction of peers that have less capacity than peer p_i in the overlay network. r_i can be estimated by sampling a few peers that are known to p_i. We use the resource levels of various peers to set the three parameters as $\alpha = 1 - r_i$, $\beta = r_i$, and $\gamma = r_i^{-ln(r_i)}$. Substituting the values for α, β, γ, PC, and PD into equation 4, we obtain:

$$P_i(L,j) = r_i^{-\ln(r_i)} \cdot \frac{C_j - r_i}{\sum_{k \in L} C_j - r_i} + (1 - r_i^{-\ln(r_i)}) \cdot \frac{\frac{1}{d_i(L,j)} - (1 - r_i)}{\sum_{k \in L} \frac{1}{d_i(L,k)} - (1 - r_i)} \quad (5)$$

We note that this configuration reflects our design rationale. The β and γ parameters assume higher values for peers with higher capacities. Hence, these peers would give preference more powerful peers while choosing a subset from L. In contrast, for peers with less resources α assumes higher values whereas β and γ become small. Thus, for these peers the subset selection is predominantly based upon the network proximities. In other words, the less powerful peers connect to nodes that are closer to them. Further, they avoid peers with large capacities, thereby shielding themselves from getting overloaded.

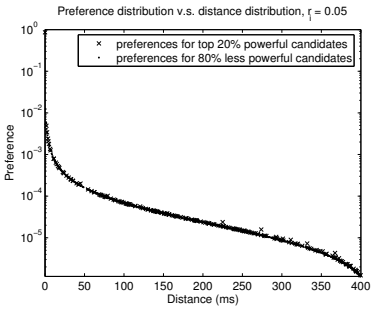

Fig. 1: Selection preference of low capacity peer vs. distance to other peers

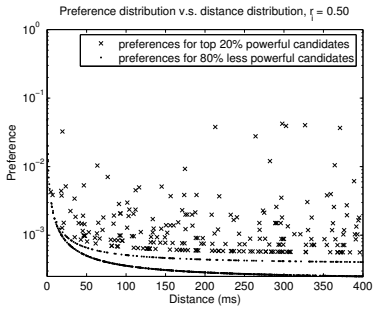

Fig. 2: Selection preference of medium capacity peer vs. distance to other peers

To evaluate the effectiveness of the selection preference metric, we simulate the selecting process of three peers, using a set of synthetic data. We assign each of them with different resource level value. The one with $r_i = 0.05$ represents a peer with low capacity. Similarly, the one with $r_i = 0.5$ simulates a peer with medium capacity, and the one with $r_i = 0.95$ represents a powerful peer. For each of them, we generated a list of 1×10^3 peers, each of which is assigned a

capacity value that follows a zipf distribution with parameter 2.0. We assume that the distance between each candidate peer and the peer evaluating them follows a uniform distribution Unif(0ms, 400ms).

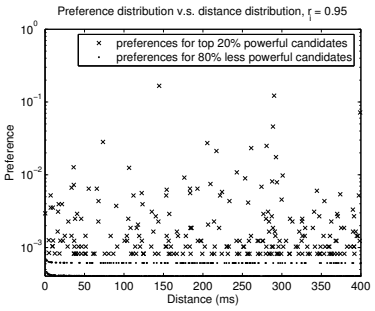

Fig. 3: Selection preference of high capacity peer vs. distance to other peers

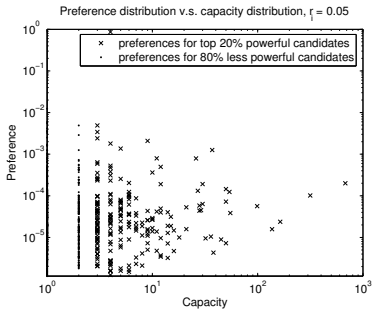

Fig. 4: Selection preference of low capacity peer vs. capacity of other peers

Figure 1 $\sim$ Figure 6 plot the simulation results, which exactly reflex our design rationale. For a weaker peer that has $r_i = 0.05$, its selection preference to other peers are dominantly decided by its distance to them, as plotted in Figure 1 and Figure 4. On the contrary, the selection preference of a powerful peer is largely decided by the node capacity of peers in the candidate set L, as shown in Figure 3 and Figure 6. For the peer that has medium amount of resources, it equally prefers powerful and nearby peers.

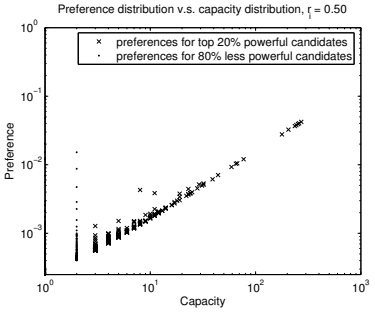

Fig. 5: Selection preference of medium capacity peer vs. capacity of other peers

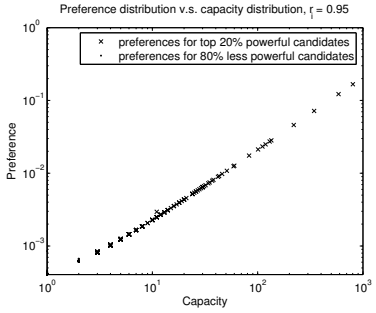

Fig. 6: Selection preference of high capacity peer vs. capacity of other peers

While the utility function quantifies the usefulness of unicast links of the overaly with respect to our design rationale, the question is *how can the utility function be utilized to construct efficient spanning trees for group communication?* We answer this question by describing the two unique features of the

GroupCast system, namely utility-aware spanning tree construction technique and utility-aware overlay management mechanism.

3.2 Utility-aware Spanning Tree Construction

In this section, we describe our technique for infusing utility-awareness into spanning tree construction for group communication. The central idea of this technique is to ensure that the edges in the spanning trees have very high utility values, thereby optimizing the overall group communication performance. If the topology of the P2P network and the utility values of all the unicast links in the network were to be available in a centralized location, we could have used one of the several optimization techniques for constructing utility-aware spanning tree. Unfortunately, due to the very nature of P2P systems collecting topological and utility information at a centralized location would be extremely expensive, if not impossible. Therefore, the challenge is to design a completely distributed spanning tree construction technique that is not only effective in ensuring high utility values for the edges in the tree but is also efficient and lightweight.

We observe that the basic spanning tree construction technique that we explained in Section 2 is indeed completely distributed, and it does not rely upon any centralized topological information. Therefore, the question is *whether it is possible to achieve high utility values while retaining the overall spanning tree construction framework?*

Our utility-aware spanning tree construction scheme is based upon the following crucial observation. Of the three phases of the basic spanning tree construction scheme, the advertisement phase has the most significant influence on the structure of the resultant spanning tree. In other words, the advertisement decisions made by various peers more or less determine the structure of the spanning tree. This is because, if a node p_l receiving an advertisement decides to participate in the group being advertised, the very links through which the advertisement was propagated to p_l from the rendezvous node would become a part of the corresponding spanning tree. However, in the basic group communication framework, each peer receiving the advertisement sends it to a randomly selected subset of its neighbors.

From the above observation, we conclude that the most natural way for injecting utility-awareness into the spanning tree construction process is to incorporate it at the advertisement phase. Accordingly, in the utility-based spanning tree construction technique, peer receiving the advertisement forwards it to a subset of its neighbors based on their utility values. Specifically, the probability of a neighbor being included in the subset selected for forwarding the advertisement is directly proportional to its utility value. Thus, a neighbor that has a higher utility value has a higher chance of being included in the subset of nodes to which the advertisement is forwarded. Due to space limit, we omit the pseudo-code of the utility-aware service announcement algorithm in this paper.

In this algorithm, rendezvous point rp first calls its local method $initiateAdvertising$ to start the SSA process. The parameter R_{rp} denotes the ranking of the rendezvous point, and is defined as the number of neighbors of rp that have more

capacity than rp. The utility function $P_{rp}(N_{rp}, j)$ is defined as Formula 5, and will help rp choose the peers either have similar capacities as rp or are physically close to rp, depending on the capacity of rp. Those peers will be the ones more useful to rp and will likely be included in the spanning tree.

Upon receiving an SSA message, peer k performs two tasks as shown in method *advertise* of the utility-aware service announcement algorithm to forward the advertisement message. First, peer k uses a local hashing table $receivedAdvertising$ to check and record if it has already received the same message from any other neighbors. The message will be dropped if it is a duplicated one. Otherwise, the same mechanism we used to initiate the service announcement process on rp will be used to select neighbors of peer k for further propagating the SSA messages.

In effect, when a peer receives an advertisement, it is more likely that the advertisement traversed a path in which each link had a high utility value. If this peer decides to participate in the group being advertised, the path of the advertisement becomes a part of the corresponding multicast tree. Thus, our scheme seamlessly incorporates utility awareness into the spanning tree construction process.

3.3 Utility-aware Topology Construction and Management

In this section, we describe our second technique for enhancing the efficiency of group communication. Studies have shown that the topologies of many natural and social networks including several real world P2P networks like Gnutella exhibit power-law characteristics [28, **?**,**?**]. In a power-law graph, the total number of node pairs within h hops (represented as $P(h)$), is proportional to the number of hops (h) raised to the power of a constant $\hbar$, $P(h) \propto h^{\hbar}$, where $h \ll \delta$, and δ denotes the diameter of the graph. Several studies have reported that unlike most P2P networks, the Gnutella P2P network suffers from large network diameters. It is interesting to note that Gnutella topology contradicts the properties of the power-law networks, which should be scale free [17]. Large diameters cause the search operations in such networks to be extremely expensive. A node needs to set the search scopes to very high values, if it wants the queries to reach substantial fraction of the peers in the network. More importantly, P2P networks with large diameters are not appropriate for group communication applications as they result in very deep spanning trees thereby affecting the latency of group message propagation.

In order to alleviate the above concern, we propose a distributed utility-aware algorithm to construct an unstructured power-law network. The objective of this technique is to create P2P networks in which the neighbors of an arbitrary node p_i have reasonably high utility values with respect to p_i. Unlike many P2P networks that are based on the concept of super nodes, our technique inserts both high-capacity and low-capacity peers into the same overlay. Our technique essentially works as follows: When a peer p_i joins the overlay, it gathers the information of a number of existing peers as its neighbor candidates. The new peer calculates the probability of connecting to each candidate by using the utility

function defined in Equation 5. These probabilities and the total number of connections that the p_i intends to maintain determine whether p_i would establish a connection with an arbitrary neighbor candidate peer.

Topology Construction Algorithm Our protocol extends the current version of the Gnutella protocol [31]. Recall that an arbitrary peer in our overlay is uniquely identified by a tuple of four attributes:

$$< IP\ address,\ port\ number,\ coordinate,\ capacity >$$

where *coordinate* represents its network coordinate and *capacity* is its capacity.

Preferential attachment has been widely used in centralized network topology generators to generate power-law networks. Algorithms like [10, **?**] add links between nodes with probability in direct proportion to the incident link degrees of the nodes. To build a power-law network for wide-area group communication applications, each node needs two types of information to decide its neighbors in the overlay network. First, it needs the degree information of the other nodes to preferentially connect to highly connected peers. Second, it needs the network proximity information to connect itself to a set of physical network neighbors and a few randomly selected ones as its routing "shortcuts". However, in a distributed environment like P2P networks, such information cannot be explicitly obtained.

In GroupCast, a joining peer i obtains a list of existing peers either using its local cache which contains its P2P network neighbors carried from the last session of activities or by contacting a host cache server. The host cache server is an extension of Gnucleus [30], which caches the information of a list of peers that are currently active in the P2P network. The joining peer i attaches its 4-tuple identification to the query request sent to the host cache server. Upon receiving a query request from peer i, the host cache sorts its cached entries in an ascending order by their network coordinate distances to peer i. From the top of this sorted list, the host cache selects a list of peers BD_i. They are returned to peer p_i together with a list of randomly selected peers BR_i. We set $|BR_i| = |BD_i|$ and use B_i to denote $BR_i \bigcup BD_i$. Similar to Gnutella, we set $5 \leq |B_i| \leq 8$.

Starting from the subset B_i of bootstrapping peers received upon its entry, Peer p_i sends a probing message M_{prob} to each peer $p_k \in B_i$:

$$M_{prob} =_{def} \langle\ source = i,\ type = prob,\ TTL = 0,\ hops = 0\ \rangle$$

Each peer p_k that receives this probing message sends back a responding message M_{prob_resp}, which is augmented with a list of p_k's P2P network neighbors $Nbr(p_k)$.

$$M_{prob\_resp} =_{def} \langle\ source = k,\ type = prob\_resp,\ TTL = 0,\ hops = 0\rangle$$

Peer p_i assembles all the neighbor information contained in the probing replies and compiles them into a candidate list LC_i. For each unique peer

$j \in LC_i$, peer p_i computes two types of information: (1) The *occurrence frequency* of peer p_j, which records the number of appearances of peer p_j in LC_i, denoted as $f_i(p_j)$. As LC_i serves as a sampling of the peers in the network, $f_i(p_j)$ is the sample of the degree of peer p_j. (2) The estimation of the physical network distance between peer p_i and peer p_j, denoted by $d_i(LC_i, p_j)$, as defined in Equation 2.

Based on these two sets of information, the peer p_i computes the utility value of each peer in LC_i using the equation 6. Depending upon its own its own node capacity, peer p_i selects a certain number of peers from the list LC_i and adds them into its neighbor list ($Nbr(p_i)$). The chances a peer $p_k \in LC_i$ being added to the neighbor list of p_i is directly proportional to p_ks utility values. Concretely, the probability of p_k being selected as a neighbor of p_i is given by the following equation.

$$P_i(LC_i, j) = r_i^{-\ln(r_i)} \cdot \frac{f_i(j) - r_i}{\sum_{k \in LC_i} f_i(j) - r_i} + (1 - r_i^{-\ln(r_i)}) \cdot \frac{\frac{1}{d_i(LC_i,j)} - (1 - r_i)}{\sum_{k \in LC_i} \frac{1}{d_i(LC_i,k)} - (1 - r_i)} \quad (6)$$

Note that this equation is equivalent to Equation 5. However, we substitute the node capacity information C_j with $f_i(j)$ and instantiate the candidate list L with LC_i. As mentioned in Section 3.1, the resource level r_i of peer p_i can be estimated through sampling of the network. The possible way of obtaining resource-levels of peers would be to derive it using statistical information [25]. However, our approach avoids the problems that arise when the statistical information is outdated.

The peer p_i now sets up its outgoing edges (forwarding connections) to each node in its neighbor list. It then initiates the process to setup the incoming edges (back links to p_i) by sending a backward connection request to each peer $p_k \in Nbr(p_i)$. The request is augmented with the capacity information C_i of peer p_i and its network coordinates.

Upon receiving a backward connection request, the peer p_k compares the capacity and distance information of peer i against those of its existing neighbors and decides whether to add p_i to its neighbor list. Specifically, this function takes three sets of information as inputs and returns the probability with which the peer p_k should add p_i to its neighbor list. The three sets of information are: the capacity ranking $rc_k(Nbr(p_k))$ of peer p_k amongst its neighbors $Nbr(p_k)$, which is defined as $rc_k(Nbr(p_k)) = \frac{|\{p_j | p_j \in Nbr(p_k), C_j \leq C_k\}|}{|Nbr(p_k)|}$; the capacity ranking $rc_i(Nbr(p_k))$ of peer p_i amongst the neighbor of peer p_k, which is defined as $rc_i(Nbr(p_k)) = \frac{|\{p_j | p_j \in Nbr(p_k), C_j \leq C_i\}|}{|Nbr(p_k)|}$; the distance ranking $rd_i(Nbr(p_k))$ of peer p_i amongst $Nbr(p_k)$, which is defined as $rd_i(Nbr(p_k)) = \frac{|\{p_j | p_j \in Nbr(p_k), D(p_j,p_k) \geq D(p_i,p_k)\}|}{|Nbr(p_k)|}$, where $D(p_i, p_k)$ denotes the network coordinate distance between peer p_i and peer p_k. The probability with which peer p_k accepts the back connection request is defined as follows:

$$PB_k(Nbr(p_k), p_i) = rc_k(Nbr(p_k))^2 \cdot rc_i(Nbr(p_k)) + (1 - rc_k(Nbr(p_k))^2) \cdot rd_i(Nbr(p_k))$$

Peer k generates a random number following uniform distribution Unif(0, 1). If this number is smaller than $PB_K(Nbr(p_k), p_i)$, peer p_i is accepted by peer p_k as a new neighbor, and a back connection acknowledgement is sent back. If this number is larger than $PB_K(Nbr(p_k), p_i)$, only with probability p_b, a backward connection from peer k to peer i will be set up. The value of p_b controls the ratio between the number of outgoing links and the number of incoming links of each peer. In our implementation, we set it with a value 0.5.

Note that the above back link set up process is based upon the same rationale we followed in devising the peer selection mechanism, i.e., powerful peers are easier to be accepted by other powerful peers as their neighbors, and weaker ones are good candidates only when they are close enough.

Neighborhood Link Maintenance The GroupCast system uses an epoch-based scheme to maintain the structure of the P2P overlay. The peers regularly exchange heartbeat messages with their neighbors. The peer p_i attaches its identifier quadruplet to each heartbeat message it sends to the neighbors. A neighboring peer p_k receiving this message responds back with its own identifier quadruplet. A neighbor that has failed to respond to two consecutive heartbeat messages is assumed to have failed. Further, a peer that *gracefully departs* sends a departure message to its neighbors. The peer p_i records the neighbor failures that have occurred in the current epoch. At the end of the epoch, the peer p_i attempts to repair its neighbor list establish a set of new links to peers that are currently not its neighbors. New peers are chosen according to their utility values. The process for choosing new neighbors is similar to that of bootstrapping. Further, the epoch duration is dynamically adjusted depending upon the network churn so that overall overlay network can agilely adapt to current churn pattern. Due to space limitations we omit further discussions on this topic. A detailed discussion can be found in our technical report.

4 Experimental Evaluation

we have implemented a discrete event simulation system to evaluate GroupCast. This system is an extended Java version of p-sim [20] system. We used the Transit-Stub graph model from the GT-ITM topology generator [34] to simulate the underlying IP networks. Peers are randomly attached to the stub domain routers and organized into overlay networks using the algorithm presented in Section 3.3. The capacity of peers is based on the distribution gathered in [25], as shown in Table 1. We use the algorithm of [1] to assign network coordinate to each peer. Each experiment is repeated over 10 IP network topologies. Each IP network supports 10 overlays, and each overlay network has provided services for 10 communication groups.

Table 1: Capacity distribution of peers

Capacity level	Percentage of peers
1x	20%
10x	45%
100x	30%
1000x	4.9%
10000x	0.1%

4.1 Power-law Overlay and Network Proximity

We first simulated the construction of P2P overlay networks. Peers join with intervals following an exponential distribution Expo(1s). They choose their overlay network neighbors using the utility-aware algorithm described in Section 3.3. Figure 7 shows the log-log degree distribution of a GroupCast overlay network of 5×10^3 peers. For comparison purposes, in Figure 8 we show the the degree distribution of a power-law network generated using the centralized PLOD algorithm [21]. The results show that our utility-aware overlay construction scheme preserves the power-law distribution property that has been observed in many real-world P2P networks. However, it is interesting to note that the node degree distribution of the GroupCast network does not have a long tail. This is because our bootstrapping algorithm are to some extent is conservative in replacing existing peers with new ones. This property results in a lower value of network clustering coefficient compared to the random power-law overlay. As our experiments show this characteristic of the GroupCast network reduces the messaging overhead without being detrimental to the performance of the overlay or the applications supported by it.

One of the objectives of the utility-aware overlay construction mechanism is to alleviate the mismatch between the topologies of the overlay and the underlying physical network. The next experiment evaluates the effectiveness of our overlay construction mechanism in minimizing the overlay-underlay mismatch. We simulated the joining processes of 1×10^3 peers and compared the overlay networks constructed using our algorithm with the ones that are generated using centralized PLOD algorithm [21]. Figure 9 and Figure 10 show the average distance of peers to their neighbors in the GroupCast overlay network and the random power-law networks respectively. The results show that in the GroupCast overlay network, the neighbors are much closer to one another than the random power-law network. Thus, the utility-aware overlay construction scheme significantly reduces the overlay-underlay mismatch. Nevertheless, the results show the existence of a few long unicast links. These links belong to the powerful peers for which network proximity is a secondary criterion for neighbor selections. The powerful peers use these long links to connect to other powerful peers and thereby creating forwarding backbone of the overlay network.

Next, we study benefits of the utility-aware overlays in reducing the latency of service lookup requests. Figure 13 shows that the subscription response time

in GroupCast overlay network is reduced by 74% to 84%, compared to that of the random power-law overlay networks. This benefits peers entering the GroupCast overlay, since they can subscribe to the group communication services much faster than their counterparts in the random power-law overlay networks.

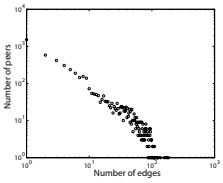

Fig. 7: Log-log degree distribution of GroupCast overlay network with 5000 peers

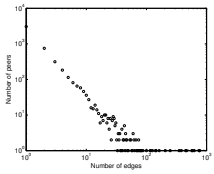

Fig. 8: Log-log degree distribution of random power-law overlay network, 5000 peers, $\alpha = 1.8$

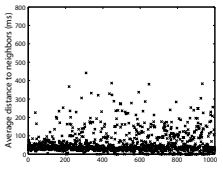

Fig. 9: Average distance to neighbors of GroupCast overlay network with 1000 peers

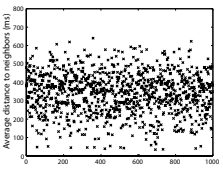

Fig. 10: Average distance to neighbors of a random power-law overlay network with 1000 peers

4.2 Evaluating the GroupCast Service Lookup Mechanism

In the second set of experiments, we evaluate the utility-aware spanning tree construction and group communication mechanisms of the GroupCast system. Most unstructured use either scoped flooding (broadcast) or random walk as their communication paradigm. However, flooding-based mechanisms are expensive in terms of message loads they impose, whereas random walks result in longer delays. The GroupCast system includes a selective service announcement (SSA) mechanism for efficient and low-cost service lookups.

Our next experiment evaluates the effectiveness and efficiency of the SSA scheme by simulating the service announcement processes in a number of overlay networks that are generated using either our utility-aware overlay construction mechanism or the centralized PLOD algorithm. In order to gain a better understanding, we compare the SSA mechanism with the non-selective service announcement (NSSA) scheme (see Section 2.1). For each overlay network, we randomly select 10 peers as rendezvous points, and initiate the selective service announcement (SSA) process and the non-selective service announcement (NSSA) process from each of them. For both SSA and NSSA, we first record the fraction of peers in the overlay that have received the service announcement. As we mentioned earlier, when these peers want to subscribe for the group communication service, they can circumvent the service searching process. For peers that have not received the service announcement message, subscription process involves searching its neighborhood for peers that have received the service announcement message. In our simulator, these peers use a ripple flooding search scheme for this purpose with TTL being set to 2. We measure the success rates of service lookups for both SSA and NSSA schemes. We also record the total number of messages generated by these two schemes.

The results in Figure 11 show that the SSA scheme reduces the total number of messages generated in both GroupCast and random power-law overlay networks. The SSA scheme limits the number of subscription messages sent to neighbors that are not likely to be a part of the communication group. This reduces the message load by 63% to 70% when compared with NSSA scheme for the GroupCast overlay. The reduction is 35% to 44% for the random power-law overlay. We notice that the number of subscription messages of SSA scheme in random power-law overlays is almost negligible. This is because GroupCast overlays have lower cluster coefficient values than the random power-law topologies generated using PLOD. Thus, SSA messages reach fewer peers. The results also show that the SSA scheme performs better for networks with higher connectivity value.

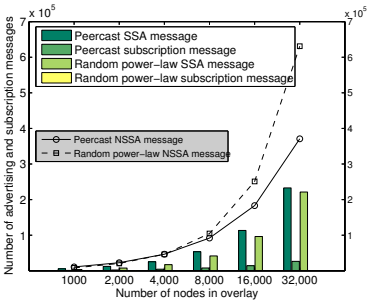

Fig. 11: Number of messages generated by service lookup schemes

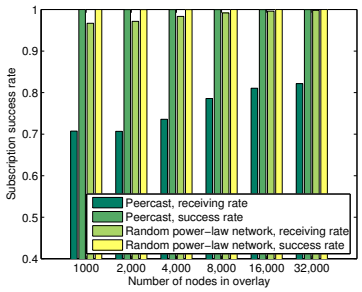

Fig. 12: Success rate of service lookup in GroupCast overlay networks and random power-law overlay networks using selective service announcement

Figure 12 leads to two interesting observations. First, fewer peers in GroupCast receive the SSA messages compared to random power-law topology. Second, all subscribers can locate their services with 100% success rate even when the initial TTL of the subscription messages is set to *two*. This is essentially because, in the GroupCast overlay, the neighbors of individual peers are likely to have higher utility values. Hence, at each step of the SSA process, more candidate peers meet utility-aware selection criterion. This is also the reason for the relatively large number of service announcement messages in the GroupCast overlay when compared with random power-law network. However, the peers chosen by our utility-aware selection mechanisms are more suitable to the group communication spanning trees and they ensure high subscription success rates even at very small TTL values.

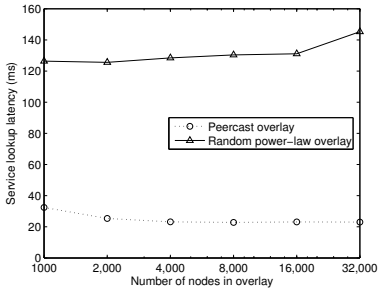

Fig. 13: Latency of service lookup in GroupCast overlay networks and random power-law overlay networks using selective service announcement

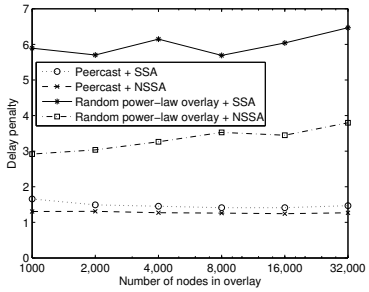

Fig. 14: Delay penalty of group communication applications

4.3 Improvement of Application Performance

Our next of experiments study the effects of the proposed techniques on a group communication application. The group communication application we consider is that of end-system multicasting (ESM). ESM has been proposed as an alternative for IP multicast, which has suffered from lack of wide acceptance and deployment. In this approach, peers form overlay networks and implement multicast functionality. Multicast data are replicated on peers and propagated through unicast edges of the overlay networks. ESM is inherently less efficient than IP multicast, as ESM may send packets with same payload multiple times over the same physical network link. Moreover, the ESM workload distribution among heterogeneous peers affects the overall system performance.

We simulated P2P overlay networks consisting of 1×10^3 to 3.2×10^4 peers. P2P overlay networks are constructed using our utility-aware mechanism as well as the centralized PLOD algorithm. We used the routing weights generated by the GT-ITM package to simulate the IP unicast routing. IP multicast systems are simulated by merging the unicast routes into shortest path trees. We use both SSA and NSSA for service announcement and subscription management.

We quantify the performance of the schemes using *Relative Delay Penalty* and *Link Stress* parameters, which are the two popular metrics for evaluating the efficiency of ESM systems. Relative delay penalty is defined as the ratio of the average ESM delay to the average IP multicast delay. Link stress is defined as the ratio of the number of IP messages generated by an ESM tree to the number of IP messages generated by an IP multicast tree interconnecting the same set of subscribers.

Figure 14 shows the relative delay penalties when multicasting is implemented through various combinations of the two overlay management schemes (utility-aware (Peecast) and random power-law) and the two spanning tree construction schemes (SSA and NSSA). Figure 15 shows the respective link stress values. The results show that ESM implemented on GroupCast overlays yield

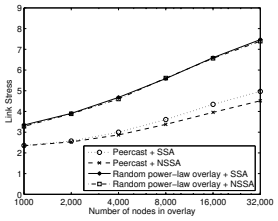

Fig. 15: Link stress of group communication applications

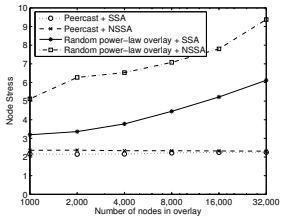

Fig. 16: Node stress of group communication applications

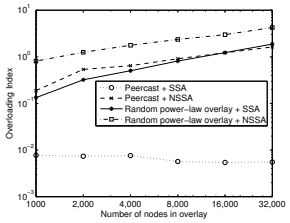

Fig. 17: Overload index

significant improvements in terms of both metrics when compared with their counterparts implemented on random power-law networks. The delay penalty of ESM implemented on GroupCast overlay is around 1.5, which is close to the theoretical lower bound of 1. The link stresses of ESM implemented on GroupCast is about 2/3 of the link stresses of ESM implemented on top of random power-law network. The improvements are due to the network proximity awareness of the GroupCast overlay networks. Multicast payloads are forwarded through shorter paths (recall the result in Figure 9 and Figure 10), thus generating fewer IP packets in the underlying IP network.

It is interesting to note that the impact of the SSA scheme on application performance is almost negligible in GroupCast overlay networks, whereas the impact in random power-law networks is significant. We attribute this behavior to the fact that GroupCast overlay networks are already aware of the network proximity of peers. Thus, the peers chosen by the SSA scheme are most likely be the ones that are actually used in the information dissemination spanning tree.

4.4 Load Balancing in Group Communication Systems

Next, we study the impacts of the different schemes on the load distributions of the group communication applications. We use the *node stress* metric to quantify the average multicast workloads on peers. The node stress is defined as the average number of children that a non-leaf peer handles in the end-system multicast tree. To measure the load distribution in the overlay network, we define a metric called *overload index*. This metric quantifies the mismatch between the a node's capacity and the communication load it encounters. Specifically, we define it as

the product of the fraction of peers overloaded and the average workloads that exceed the capacities of those peers.

Figure 16 shows the node stress values of the four combinations and Figure 17 indicates their overload indices. The results show that our utility-aware communication mechanism can improve the load distribution in both random power-law overlay and GroupCast overlay. However, the GroupCast overlay considers node capacities during the overlay construction process, which makes it more scalable in terms of node stress values. When the system is scaled to accommodate more subscribers, the node stress in GroupCast is almost constant. We also note that SSA scheme can effectively reduce overloading in overlay networks. By considering node capacity of candidates in the choosing the recipients of the service announcement messages at each hop, the SSA scheme reduces the overloading in the random power-law overlay by an order of magnitude. Similarly, the overloading in the GroupCast overlay is reduced by one to two orders of magnitude when compared with random power-law overaly. Combining the SSA with GroupCast utility-aware overlay bootstrapping scheme yields even better performance. The overloading is reduced by two to three orders of magnitude compared to the random power-law network. We also notice that in Figure 17 the curve of GroupCast overlay coupled NSSA and the line of random power-law overlay with SSA crosses each other when the overlay size is around 1.6×10^4. This indicates that optimization at the overlay level is better than at the group communication application level for larger overlay networks.

To summarize, the proposed utility-aware spanning tree construction scheme and the utility-aware topology management mechanism offer significant benefits in terms of scalability, efficiency and load distribution.

5 Related Work

The work on group communication in P2P networks has mainly focused on structured P2P networks. Researchers have proposed several application-level multicasting schemes for DHT-based structured overlay networks [24, ?,?,?,?]. However, structured P2P networks have high maintenance costs, especially in highly dynamic environments. In contrast the GroupCast system does not require any DHT abstractions from the overlay. Instead, Our techniques are completely distributed, and they rely only on local information. Further, our system makes no assumptions on the underlying IP network or the knowledge of peer activity patterns.

Many distributed group communication systems rely on the services of overlay networks for operation [7, 11, 14]. The properties of overlay networks, such as communication efficiency, system scalability, and fault resilience, largely decide the performance of those systems. Usually, end-hosts in the communication groups use the unicast links of overlay networks to exchange application and management messages. Researchers have explored various techniques to optimize the system performance at the application level with the objective of designing efficient and scalable query processing mechanisms [13].

A number of P2P systems have been proposed to construct overlay networks with power-law degree distribution [29, **?**]. However, none of these works have considered combining node capacity and network proximity metrics in constructing overlay networks and the effects such a combination may have on the overlay network performance. Our work shows that a careful combination of node capacity and network proximity metrics can enhance the scalability and performance of applications that run on top of these overlays.

Another approach to improving P2P networks is to utilize the ranking of different peers in terms of their node capacity and organize them into different hierarchical layers [31, 4, 33]. However, such predetermined hierarchical structures can introduce system vulnerabilities. For example, when super nodes are attacked or overloaded, the overlay network can easily become fragmented if the normal peers depend upon a few peers for their services. Further, for efficiency purposes, the supernodes maintain the state information of the normal peers they serve. However, such state information is generally tied to the application, and it is hard to design a supernode overlay layer that can serve as a generic middleware to support different services. Such systems are also vulnerable to attacks from malicious peers that assume the role of super peers.

Adaptation mechanisms have been studied in the context of application-layer multicasting [8, 36]. Our research is complimentary to these works. These systems can utilize the GroupCast protocols for construct well-regulated spanning trees. Our protocol can help reduce the number of adaptations by ensuring the efficiency of initial spanning trees. Techniques such as RON [6] have been designed for building generic overlays that are independent of the applications built on top of them. The GroupCast system differs from these works in two specific ways. First, our system construct overlay networks that incorporate network proximity information. Second, our algorithm builds *scale-free* power-law topologies and assigns connections according to the peers' capacities.

In short, the work presented in this paper has several unique features and our system addresses a problem that is crucial for the success of several multi-party collaborative applications.

6 Conclusion

It is widely recognized that unstructured P2P networks provide a decentralized and economical platform for implementing group communication services. However, the ad-hoc communication and the non-deterministic service lookups in these networks poses several performance challenges. This paper presents the design and evaluation of *GroupCast* − a utility-aware decentralized middleware architecture for scalable and efficient wide-area group communications. The GroupCast design incorporates three novel features: (a) A *utility function* that measures the usefulness of unicast links to the scalability and efficiency of the group communication application; (b) A distributed utility-aware scheme for constructing efficient spanning trees for disseminating group communication payloads; and (c) A utility-based overlay management protocol that generates

and maintains low-diameter overlay networks for supporting scalable wide-area group communication applications. Our experiments show that GroupCast can improve the scalability of wide-area group communications by one to two orders of magnitude.

Our work on *GroupCast* continues along several dimensions. First, we are currently incorporating reliability, trust, and security solutions into the *GroupCast* system. Concretely, the GroupCast system can be augmented with mechanisms such as dynamic replication [35], TrustGuards [27] and Event Guards [26] to enhance its failure resilience, its node-level trust, and its middleware level security. Second, the GroupCast system can be easily adapted for "supernode" or multi-layer overlay architectures.

Acknowledgement

This work is partially supported by grants from NSF CSR, NSF SGER, NSF CyberTrust, and grants from IBM SUR and IBM Faculty Award, DoE SciDAC, and AFOSR.

References

1. E. Ng. GNP software. http://www.cs.rice.edu/ eugeneng/research/gnp/software.html, July 2006.
2. Google Talk. www.google.com/talk/, July 2006.
3. Live Meeting. http://www.microsoft.com/office/livemeeting, July 2006.
4. Sharman networks LTD. KaZaA media desktop. http://www.kazaa.com, July 2006.
5. Skype. http://www.skype.com, July 2006.
6. D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resillient overlay networks. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP)*, 2001.
7. S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *Proceedings of the 2002 ACM SIGCOMM Conference*, 2002.
8. S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller. Construction of an efficient overlay multicast infrastructure for real-time applications. In *Proceedings of INFOCOM*, 2003.
9. S. A. Baset and H. Schulzrinne. An analysis of the skype peer-to-peer internet telephony protocol. Technical Report cucs-039-04, Department of Computer Science, Columbia University, September 2004.
10. T. Bu and D. Towsley. On distinguishing between internet power law topology generators. In *IEEE INFOCOM*, New York, NY, June 2002. IEEE.
11. M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in communications (JSAC)*, 2002.
12. Y. Chawathe. *Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service*. PhD thesis, University of California, Berkeley, 2000.
13. Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making Gnutella-like P2P systems scalable. In *ACM SIGCOMM*, Karlsruhe, Germany, August 2003. ACM.

14. Y.-H. Chu, S. G. Rao, and H. Zhang. A case for end system multicast. In *ACM SIGMETRICS 2000*, pages 1–12. ACM, 2000.
15. F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *ACM SIGCOMM*, Portland, Oregon, USA, August 2004. ACM.
16. S. Deering and D. Cheriton. Multicast routing in datagram internetworks and extended lans. *ACM Transactions on Computer Systems*, 8(2), May 1990.
17. M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *ACM SIGCOMM*, pages 251–262. ACM, 1999.
18. P. Francis. Yoid: Extending the multicast internet architecture. 1999.
19. J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole, Jr. Overcast: Reliable multicasting with an overlay network. In *Proceedings of Symposium on Operating System Design and Implementation (OSDI)*, pages 197–212, 2000.
20. S. Merugu, S. Srinivasan, and E. Zegura. p-sim: A simulator for peer-to-peer networks. In *Proc. of the 11th IEEE Intl. Symp. on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'03)*, 2003.
21. C. R. Palmer and J. G. Steffan. Generating network topologies that obey power laws. In *Proceedings of GLOBECOM '2000*.
22. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. In *Proceedings of SIGCOMM*. ACM, 2001.
23. S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Application-level multicast using content-addressable networks. *Lecture Notes in Computer Science*, 2233, 2001.
24. A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329–??, 2001.
25. S. Saroiu, P. Gummadi, and S. Gribble. A measurement study of Peer-to-Peer file sharing systems. In *Proceedings of MMCN*, San Jose, CA, August 2002.
26. M. Srivatsa and L. Liu. Securing Publish-Subscribe overlay services with EventGuard. In *Proceedings of ACM Computer and Communication Security (CCS 2005)*, 2005.
27. M. Srivatsa, L. Xiong, and L. Liu. Trustguard: Countering vulnerabilities in reputation management for decentralized overlay networks. In *WWW 2005*.
28. D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.
29. R. H. Wouhaybi and A. T. Campbell. Phenix: Supporting resilient low-diameter peer-to-peer topologies. In *IEEE INFOCOM*, Hong Kong, China, March 2004. IEEE.
30. WWW. Gnucleus. The Gnutella web caching system. *http://gnucleus.sourceforge.net*, July 2006.
31. WWW. The Gnutella RFC. *http://rfc-gnutella.sourceforge.net*, July 2006.
32. Z. Xu, M. Mahalingam, and M. Karlsson. Turning heterogeneity into an advantage in overlay routing. In *Proceedings of INFOCOM*, 2003.
33. Z. Xu, C. Tang, and Z. Zhang. Building topology-aware overlays using global soft-state. In *Proceedings of ICDCS*, 2003.
34. E. W. Zegura, K. L. Calvert, and S. Bhattacharjee. How to model an internetwork. In *IEEE Infocom*, volume 2, pages 594–602. IEEE, March 1996.
35. J. Zhang, L. Liu, C. Pu, and M. Ammar. Reliable peer-to-peer end system multicasting through replication. In *Proceedings of IEEE P2P*, 2004.

36. Y. Zhou and et al. Adaptive reorganization of conherency-preserving dissemination tree for streaming data. In *ICDE*, 2006.