Discovery of Stable Peers in a Self-organising Peer-to-Peer Gradient Topology

Jan Sacha, Jim Dowling, Raymond Cunningham, and René Meier

Distributed Systems Group, Trinity College, Dublin
{jsacha, jdowling, rcnnnghm, rmeier}@cs.tcd.ie

Abstract. Peer-to-peer (P2P) systems are characterised by a wide disparity in peer resources and capabilities. In particular, a number of measurements on deployed P2P systems show that peer stability (e.g. uptime) varies by several orders of magnitude between peers. In this paper, we introduce a peer utility metric and construct a self-organising P2P topology based on this metric that allows the efficient discovery of stable peers in the system. We propose and evaluate a search algorithm and we show that it achieves significantly better performance than random walking. Our approach can be used by certain classes of applications to improve the availability and performance of system services by placing them on the most stable peers, as well as to reduce the amount of network traffic required to discover and use these services. As a proof-of-concept, we demonstrate the design of a naming service on the gradient topology.[1]

1 Introduction

Recent measurements on peer-to-peer (P2P) systems show that the distribution of peer characteristics, such as their availability, bandwidth, or storage space, are highly skewed and often heavy-tailed or scale-free [1, 2, 3]. In particular, it has been shown that the uptime characteristics of peers are extremely diverse, and a large number of peers stay in the system for a relatively short time, which is commonly referred to as high "infant mortality" [4, 5].

At the same time, existing state-of-the-art P2P systems are often based on the assumption that all peers in the system have equal capabilities and that the distribution of resources between peers is uniform. Initial approaches using Distributed Hash Tables (DHTs), such as Chord [6], CAN [7] and Pastry [8], are examples of this assumption. These systems treat all peers as equals, and hence, low performance peers receive approximately the same amount of traffic as the highest performance peers.

A number of P2P systems address the heterogeneity of P2P environments by electing *super-peers* and assigning them extra responsibilities [9, 10, 11, 12]. However, these systems introduce the problem of super-peer election. Solutions based on flooding, random walking or other traditional election algorithms, potentially require communication with all peers in the network and thus do not

[1] This work was supported by the European Union funded "Digital Business Ecosystem" Project IST-507953.

F. Eliassen and A. Montresor (Eds.): DAIS 2006, LNCS 4025, pp. 70–83, 2006.

scale to large networks. Other solutions such as manual or static configuration of super-peers are inappropriate due to a lack of global knowledge of application characteristics.

This paper presents an approach where peers periodically measure their performance and stability properties, using a *utility* metric, exchange their measurements with neighbours, and construct a self-organising *gradient topology* that enables efficient searching for stable (high utility) peers in the network. We design and evaluate a search algorithm, called *gradient search*, that exploits the implicit information contained in the gradient topology and allows the efficient discovery of stable peers. We compare gradient search with random walking and with a probabilistic search strategy based on Boltzmann exploration, and we show that our approach provides superior performance. We also demonstrate that gradient search significantly reduces the message loss rate by preferentially forwarding messages through more stable peers.

The topology is designed to support certain classes of applications, such as P2P storage systems, or P2P registries, where system services are deployed on the most stable peers (super-peers), thus improving the stability and performance of these services. Our approach manages high rates of churn by exploiting stable peers to both provide system services and to route to these services using gradient search. As a proof-of-concept, we demonstrate the design of a sample naming service.

The remainder of the paper is organised as follows. Section 2 reviews related work. In section 3, we present an overview of the gradient topology and describe our neighbour selection algorithm that generates the topology. In section 4, we discuss search strategies for stable peer discovery in a gradient topology and we apply our approach to a sample naming service. In section 5, we describe our experimental setup and we analyse the results of the different search strategies. Section 6 concludes the paper.

2 Related Work

Recent research on P2P systems has been primarily focused on Distributed Hash Tables [6, 7, 8, 13], where the main goal is to provide efficient routing between any pair of peers. In our approach, we are focusing on searching for peers with particular properties in the system, and assuming that system services are placed on these peers, we provide a mechanism that allows the efficient discovery and consumption of these services.

A number of techniques have been developed for searching in unstructured P2P networks (e.g., Yang and Molina [14]). However, these techniques do not exploit any information contained in the underlying P2P topology, in contrast to our gradient search heuristic that takes advantage of the gradient topology structure to improve the searching performance. Morselli at al [15] proposed a routing algorithm for unstructured P2P networks that is similar to gradient searching, however, they address the problem of routing between any pair of peers rather than searching for reliable peers or services.

Many existing P2P systems adopt a super-peer structure to exploit stable and/or high performance peers. Yang and Molina [9] investigate general principles of designing super-peer-based networks, however, they do not provide any specific super-peer election algorithm. OceanStore [16] proposed to elect a primary tier "consisting of a small number of replicas located in high-bandwidth, high connectivity regions of the network" for the purpose of handling updates, however, no specific algorithm for the election of such a tier is presented. Brocade [10] improves routing efficiency in a DHT by exploiting resource heterogeneity, but unlike our approach, it doesn't address the super-peer election problem.

In Chord [6, 17], it has been shown that the load between peers can be balanced by assigning multiple *virtual servers* to high performance physical hosts. Similarly, Mizrak et al [12] proposed the use of high capacity super-peers to improve routing performance. However, these systems focus on load balancing, and do not allow the selection of potential super-peers from the set of all peers in the system.

Montresor [11] proposes a protocol for super-peer overlay generation, however, unlike our gradient topology, his topology maintains a discrete (binary) distinction between super-peers and client peers. In contrast, our novel approach introduces a continuous peer utility spectrum and thus allows the identification of high utility (super)peers. Our neighbour selection algorithm can be seen as a special case of the T-Man protocol [18] that generates a gradient topology, where the ranking function is based on peer utility. The advantage of such a utility ranking function is that applications built on top of the gradient topology can exploit more stable peers in the system.

3 Self-organising Gradient Topology

In this section, we introduce the concept of a gradient P2P topology and we outline its main properties. We present a neighbour selection algorithm that generates the gradient topology and we show that it is self-organising. The topology is used in the later sections by a searching algorithm that enables the discovery of stable peers in the system.

The gradient topology is a P2P topology where the highest *utility* peers are connected with each other and form the so called *core* of the system, while lower utility peers are located gradually farther from the core. Peer utility is application specific and measures the ability of a peer to maintain services or provide resources to the system. The core, which clusters the highest utility peers in the system, is therefore most suitable for maintaining system services and system data. Figure 1 shows a visualisation of a gradient topology with the core visible at the centre. The position of each peer in the topology is determined by the peer's utility.

The definition of the utility function depends on the application built on top of the gradient topology and it captures domain specific knowledge about peers. It measures the properties of peers that are desired or required by the application. For example, in a P2P storage systems, the utility of a peer may be defined as a peer's

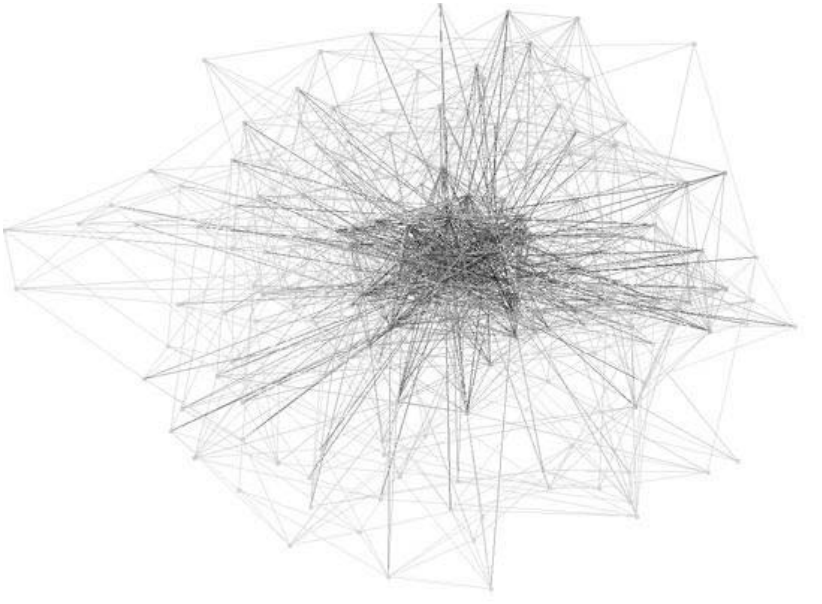

Fig. 1. Visualisation of a gradient topology

available bandwidth and local storage space. In a multi-media streaming application, the utility may be defined as a peer's latency and bandwidth, while in a grid computing system we may define the utility as a function of a peer's CPU load.

A very important property of the gradient topology is that the utility function is orthogonal to the neighbour selection algorithm, which generates the topology, and to the searching algorithm. The only assumption these algorithms make about the utility function is that every peer calculates some utility value.

In the experiments described later in this paper, we define peer utility as the peer's current *uptime*, and we use the uptime as a metric to measure peer stability. Our metric is based on the observation that high stability peers, on average, have a higher uptime value than low stability peers. More elaborate peer stability models will be studied in the future, in particular, metrics based on peer uptime history.

We also assume that the system runs in a cooperative environment, and that every peer is able to calculate its own utility. Future work will investigate the use of the gradient topology in an untrusted environment where malicious peers may provide incorrect utility information. We expect that this issue can be addressed by adopting one of the existing decentralised approaches to reputation or trust management.

3.1 Building a Gradient Topology

We have designed and evaluated a neighbour selection algorithm that generates the gradient topology in a completely decentralised P2P environment. Each peer

maintains two sets of neighbours, a *similarity-based* set and a *random* set. Peers periodically gossip with each other and exchange their sets. On receiving both sets from a neighbour, a gossipping peer selects one entry whose utility level is closest to its own utility and replaces an entry in its similarity-based set. This behaviour clusters peers with similar utility characteristics and generates the core of the network surrounded by peers with gradually decreasing utility. In addition, a gossipping peer randomly selects an entry from the received random set and replaces a random entry in its random set. Connections to random peers allow peers to explore the network in order to discover other potentially similar neighbours. This greatly reduces the probability of more than one cluster of high utility peers forming in the network. Random connections also massively reduce the probability of the gradient topology partitioning due to excessive clustering. Figure 2 shows a neighbour selection algorithm being used by two peers, A and B, to exchange neighbour information during one round of gossipping.

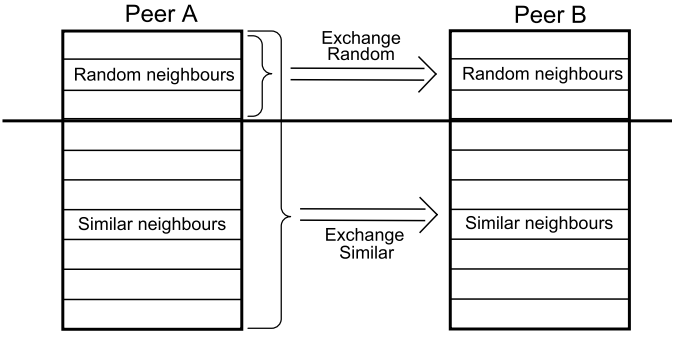

Fig. 2. Neighbourhood set exchange from Peer A to Peer B

In addition to the neighbour sets, each peer maintains a cache that stores estimated utility values of current neighbours. This cache is updated whenever a peer gossips with a neighbour.

Our initial evaluation of the neighbour selection algorithm, described in a separate paper [19], shows that the algorithm generates a P2P topology that has a gradient structure and a very small diameter (an order of 5-6 hops for 100,000 peers). The algorithm works well for a relatively small number of neighbours per peer (an order of 20). Figure 1 above shows a visualisation of a sample gradient topology created using the described neighbour selection algorithm.

The emergence of the gradient topology is an example of self-organisation. Peers are independent, have limited knowledge about the system and interact with a limited number of neighbours. There are no centralised components. The peers estimate their microscopic properties, i.e., their utility, and through the exchange of information with neighbours the peers build a P2P topology that has global, macroscopic properties, i.e., the gradient structure. The resultant topology is based on peer utility characteristics, which contrasts with many other P2P systems where the topology is based on random peer identifiers.

4 Discovery of Stable Peers

In this section, we present a heuristic search algorithm, which we call gradient search, based on the gradient topology, that enables the discovery of high utility peers in the system. The algorithm exploits the information contained in the gradient topology to limit the search space to a relatively small subset of peers and to achieve a significantly better search performance than traditional search techniques, such as random walking, which require the communication with potentially all peers in the system.

The goal of the search algorithm is to deliver a message from any peer in the system to a high utility peer in the core, i.e., to a peer with utility above a *threshold.* The value of the threshold is assigned by a peer that initiates the search and is included in the search message. The threshold values are application specific and can vary between services. Peers below the specified utility threshold forward messages to their neighbours. Each message is associated with a time-to-live (TTL) value that determines the maximum number of hops the message can be propagated. The messages are never duplicated.

In gradient search, a peer greedily forwards messages to its highest utility neighbour. Thus, messages are forwarded along the utility gradient, as in hill climbing and other similar techniques. It is important to note that the gradient search strategy is generally applicable only to a gradient topology. It relies on a certain structure of the P2P overlay, in particular, it assumes that a higher utility peer is closer to the core in terms of the number of hops than a lower utility peer. The maintenance of the gradient structure introduces extra overhead, however, the cost of topology maintenance is generally constant per peer, since the neighbour selection algorithm is performed periodically, and potentially can be lower than the cost of frequent and expensive searches.

In a greedy search strategy, where messages are always forwarded to the highest utility peer, messages may oscillate around peers with a locally maximal utility. To prevent message looping, we append a list of visited peers to each message, and we add a constraint that messages are never forwarded to already visited peers. Local maxima should never occur in an idealised gradient topology, however, every P2P system is under constant churn and the topology can always contain local deviations.

We compare gradient search with a probabilistic search strategy where a peer, x, selects the next-hop destination for a message with probability, P_x, given by the Boltzmann exploration formula [20]:

$$P_x(a) = \frac{e^{(U(a)/T)}}{\sum_{i \in N_x} e^{(U(i)/T)}}$$

where $P_x(a)$ is the probability that x selects neighbour a, $U(a)$ is the estimated utility of peer a, N_x is the set of x's available neighbours, and T is a parameter of the algorithm called the temperature. Setting T close to zero causes the algorithm to be more greedy and deterministic, as in gradient search, while if T grows to infinity, all neighbours are selected with equal probabilities similar to random

walking. Thus, the temperature enables a trade-off between exploitative (and deterministic) routing of messages towards the core, and random exploration that enables searches to escape local maxima.

Routing messages steeply towards the core, as in the gradient search, or Boltzmann search with a low temperature value, has the advantage over random walking that subsequent peers on a message's path are more and more stable, and therefore, the probability of message loss decreases.

4.1 Applying the Gradient Search

We demonstrate the use of the gradient topology by sketching out the design of a sample naming service. The naming service supports registration, unregistration, and querying of names. We show that all of these operations can be implemented easily using gradient searching.

One of the first decisions when designing a naming service is where to store the name entries, i.e., the mapping between the names and the objects associated with them. We assume that a centralised solution, where all information is stored by one peer is unacceptable due to reliability and performance reasons. Another extreme, a fully decentralised solution, where all peers participate in the replication of the naming service, has a drawback that the use of low performance peers may degrade the performance of the entire system. For this reason, it is preferable to choose a group of super-peers and use them to host the naming service. However, this introduces the problem of super-peer selection from the set of all peers in the system.

This problem can be solved using the gradient topology and gradient search. We define the utility function so that it describes the requirements for our super-peers, for example as some peer stability metric (e.g. uptime). If we expect that the size of the naming service is significantly large, we may extend the utility metric to include the peer storage space, and potentially, network bandwidth. Given the utility function, the gradient topology can then be generated by the peers in the system.

In order to create the first replica of the service, the owner of the service decides on the utility threshold required for the naming service replica, searches for a peer above the threshold, and requests a replica placement. Subsequent replicas are created in a similar way. Whenever a peer updates the naming service, either by inserting, removing, or modifying an entry, the update request is routed to the core using the gradient search and the update is performed on one of the replicas of the naming service. The replicas need to be synchronised after they are modified. The synchronisation method depends on the replication scheme used, and exact details of the synchronisation algorithm are beyond the scope of this paper. However, we assume that some probabilistic gossip-based approach can be efficiently adopted, since in the gradient topology replicas are located close to each other in the core, and hence, the update messages do not need to be propagated to low-utility peers outside of the core.

The query operation is perhaps the most important for the performance of the system since we expect that most naming services are much more frequently

queried than updated, similarly as the update can be implemented using gradient searching. A query is routed to the core where it can be resolved by any naming service replica (see Figure 3). No synchronisation is needed. For subsequent requests, peers may cache known replica addresses and contact them directly.

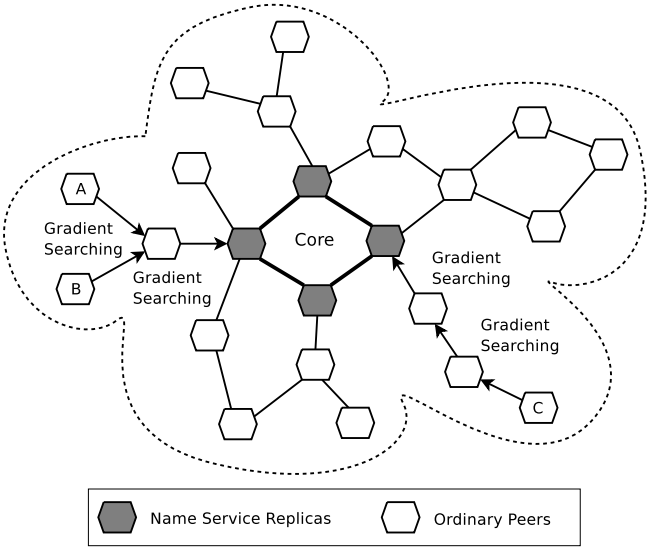

Fig. 3. Sample Naming Service built on top of the Gradient Topology. Gradient searching is used by Peers A, B, and C to discover and access instances of the naming service.

We believe that our approach based on the gradient topology and gradient searching can be used to build other classes of applications, such as a P2P storage system, a P2P distributed database, or a P2P multicast application.

5 Experimental Evaluation

In this section, we describe our experimental setup and present the results of search experiments on the gradient topology. In the experiments we compare the performance of gradient search, random walking, and Boltzmann search by measuring the three properties for each of the three search algorithms. Firstly, we calculate the average number of hops in which the algorithm delivers a message from a random peer in the network to the core, i.e., to a peer above a certain utility threshold. Next, we measure the average message loss rate of the sent messages. Finally, we calculate the average utility of peers that are used as hops when forwarding messages.

We perform three experiments to simulate the gradient topology. In the first experiment, we increase the number of peers in the system while keeping a constant peer churn rate. In the second experiment, we keep the network size

constant, however, we increase the peer churn rate over time. In the last experiment, the number of peers and the churn rate are fixed, but we increase the message TTL.

We ran our experiments on a Pentium 4 machine with a 3GHz processor and 3GB RAM under Debian Linux. We evaluate the search algorithms in a Java-based discrete event simulator. An individual experiment consists of a set of peers, connections between peers, and messages passed between peers. We assume all peers are mutually reachable, i.e., any pair of peers can establish a connection. We also assume that it takes exactly one time step to pass a message between a pair of connected peers. We do not model network congestion, however, we limit the maximum number of concurrent connections per peer. In order to reflect network heterogeneity, we limit the number of peer connections according to the Pareto distribution with an exponent of 1.5 and a mean of 20 connections per peer.

The simulated P2P network is under constant churn. Every new peer is assigned a session duration, measured in simulation steps, according to the Pareto distribution with an exponent of 1.5. The session duration determines the time step when a peer leaves the system. We calculate the churn rate as the fraction of peers that leave (and join) the system at one step of the simulation. Over the lifetime of a running system, the average churn rate is equal to the inverse of the expected peer session time.

We use a central bootstrap server that stores 1000 addresses of peers that have recently joined the network. The list includes "dangling references" to peers that may have already left the system. Every joining peer receives an initial random set of neighbours from the bootstrap server. A peer discovers other peers in the system by gossipping with neighbours at every step of the simulation. If a peer becomes isolated from the network (i.e., has no neighbours), it is bootstrapped again. Our experience shows that if a peer maintains 10 random connections, the possibility of isolation is extremely low.

We start each individual experiment from a network consisting of a single peer. The number of peers is increased by one percent at each time step, until the network grows to the size required by the experiment. Afterwards, the network is still under continuous churn, however, the rate of arrivals is equal to the rate of departures and the number of peers in the system remains constant.

At each turn, a number of randomly selected peers emit messages, and all peers attempt to either deliver or forward messages that they hold in their buffers. If a peer's utility is higher than a defined utility threshold, all messages in its buffer are delivered. Otherwise, each message is forwarded to one of the peer's neighbours selected by the current search policy. When the TTL value of a message drops to zero, the message is discarded. Additionally, if a peer leaves the system, all messages that it currently stores in its buffer are lost.

We examine peer churn rates between 0 and 0.1, where the value of 0.1 corresponds to a configuration where 10% of all peers leave the system at every step of the simulation. In physical time, if a discrete time step was 10 seconds, such churn rate would correspond to roughly 1000 peer departures per second

for a 100,000 peer network. We have observed that for extreme churn rates, such as 0.1 and higher, the network topology depends heavily on the bootstrapping method.

For the purpose of the simulation, in all experiments, we set the utility threshold to a value that corresponds to 1% of highest utility peers. In Boltzmann searching we compare two temperatures: 1000 and 100. The TTL value is set to 100 hops if not stated otherwise.

5.1 Evaluation Results

The experimental results reveal that gradient search exhibits better performance than Boltzmann searching and random walking, in terms of number of hops and message loss rate, when routing messages from a random peer to the core.

Figure 4(a) shows the average hop count for delivered messages as a function of the network size. The churn rate was fixed at 0.01. We can see that gradient search performs better than other search strategies, and that the message hop count increases together with the Boltzmann temperature. For the random walk, the hop count grows more slowly than for gradient search with increasing network size. This can be explained by the fact that the average number of high utility peers is a fixed percentage of the network size, and hence, the probability of high utility peer discovery by random walking is a function of this percentage. For gradient search, the hop count increases with the network size, since the average distance from a peer to the core increases. We can also see that the two Boltzmann approaches, with different temperatures, converge as the network size grows. This is due to the growing average utility (uptime) of peers in the system, which results in a decreasing relative difference between the Boltzmann temperatures.

Figure 4(b) shows the average message loss rate as a function of the network size with a churn rate of 0.01, and Figure 5(a) shows the average message loss rate as a function of the churn rate for a network of 10,000 peers. Both figures demonstrate that the message loss rate is lowest for the gradient search, and that it grows as the Boltzmann temperature is increased.

Better performance of the gradient search results from two facts. First, as shown in Figure 4(a), the message path is shorter in gradient searching than in other search strategies, and therefore, the probability that a message is lost by forwarding peers, or that the message exceeds its TTL value, is lower. Second, as confirmed by measurements reported below, the stability of peers used for forwarding messages in the gradient search is higher, which additionally reduces the message loss probability. For random walking the message loss rate is nearly equal for all network sizes, which is due to the fixed percentage of high utility peers in the system.

Figure 5(b) presents the message loss rate as a function of the churn rate with a distinction between message loss caused by exceeded message TTL and message loss caused by peers leaving the system. The total message loss rate is calculated as a sum of the two mentioned loss rates. The figure shows that for random walking the message loss rate attributed to peers leaving the system

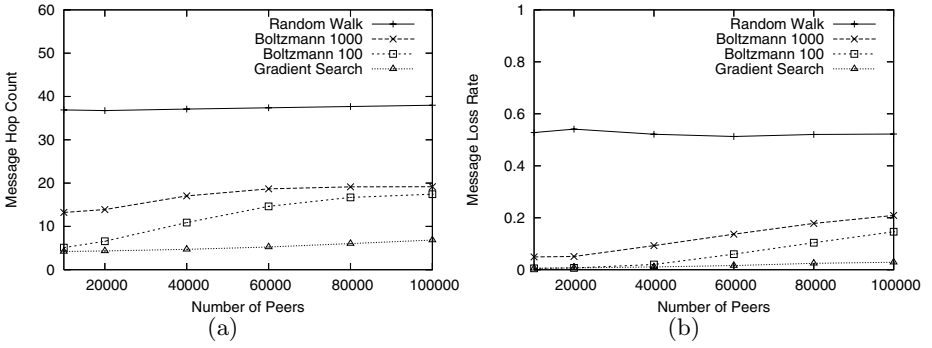

Fig. 4. Average hop count of delivered messages (a) and average message loss rate (b) as function of network size with a churn rate of 0.01 and TTL set to 100

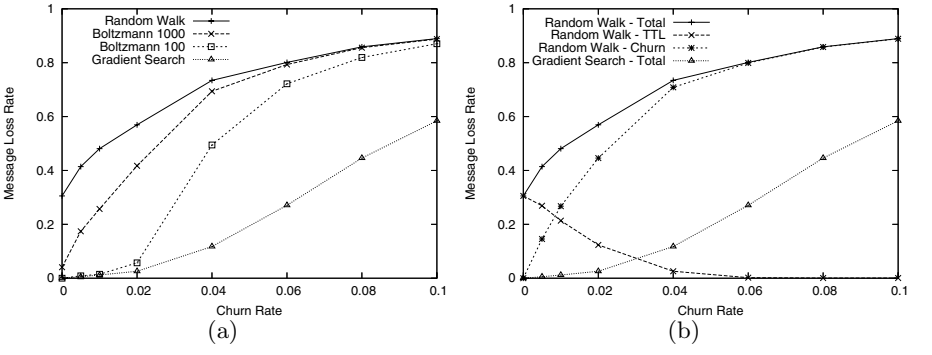

Fig. 5. Message loss rate as a function of peer churn rate. (a) Comparison between random walk, Boltzmann search, and gradient search. (b) Comparison between message loss rates attributed to exceeded message TTL and message loss attributed to peer churn. For gradient search, nearly 100% of the observed message loss is caused by peer churn (network size is 10,000 and TTL=100).

grows together with the churn rate. At the same time, the message loss rate attributed to exceeded TTL decreases with growing churn, which means that for higher churn rates messages are more likely to be lost by leaving peers than by exceeding their TTL values. For gradient search, nearly 100% of the total message loss is caused by churn.

Figure 6(a) shows the message loss rate as a function of message TTL. We can see that the overall message loss decreases when TTL grows. For random walking, as the TTL is increased, messages are lost more often due to churn, i.e., because of peers leaving the system. As a consequence, the message loss rate does not converge to zero. On the contrary, for the gradient search, the message loss rate becomes negligible for TTL values above approximately 50 hops.

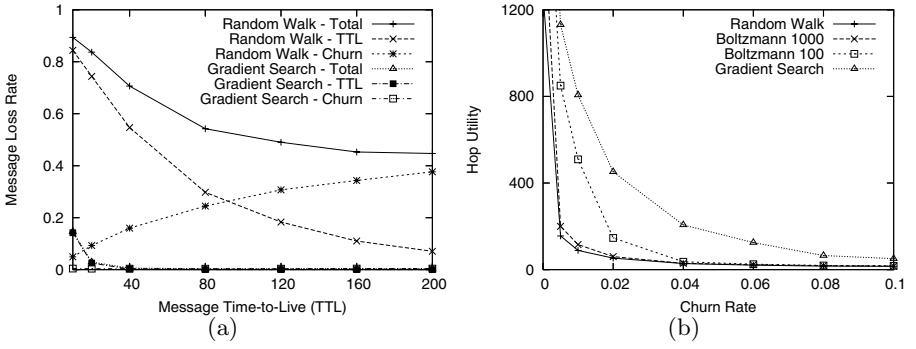

Fig. 6. (a) Message loss rate as a function of message TTL for 10,000 peers and 0.01 churn rate. The graph shows a distinction between message loss caused by exceeded message TTL and message loss caused by peers leaving the system. (b) Average utility of peers forwarding messages (hop utility). The average utility of all peers in the system, measured as uptime, decreases with the churn rate. Gradient search achieves better hop utility by forwarding messages to the highest utility peers (network size is 10,000, TTL=100).

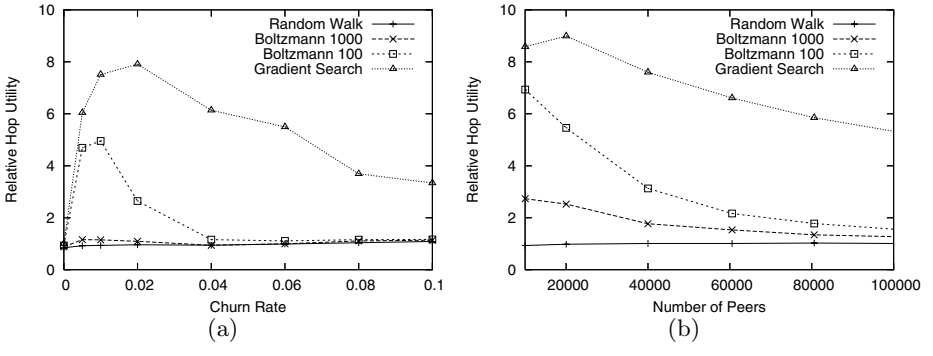

Fig. 7. Average relative utility of peers forwarding messages (relative hop utility) as a function of churn rate (a) and network size (b). The utility is scaled so that the value of 1 corresponds to the average utility among all peers in the system.

Figures 6(b), 7(a), and 7(b) demonstrate the average utility of peers used for forwarding messages in different searching strategies. In all cases we can see that the average hop utility is highest for gradient search and lowest for random walks. This result is consistent with the observation that for gradient search the message loss rate is lower than for the other strategies. In Figures 7(a) and 7(b) the utility is scaled in such a way that the average utility over all peers in the system is 1. As expected, for random walking the average path utility is 1. Figure 6(b) shows also that the average peer utility (measured as uptime) grows steeply when the churn rate approaches zero.

6 Conclusions

In this paper, we have described the gradient topology and the gradient search algorithm that allow peers to efficiently discover peers with particular attributes in the system, i.e., high utility peers. The topology can be used to improve the availability and performance of system services by placing them on the highest utility peers, as well as to reduce the amount of network traffic required to discover and use these services. The topology enables a trade-off between centralisation and decentralisation, in the sense that it allows the selection of a subset of high utility peers for supporting application services rather than distributing the services equally between all peers. We demonstrate the usability of our approach by designing a sample naming service on top of the gradient topology.

The evaluation of our work shows that gradient search achieves significantly better performance than random walking. Our results agree with the no-free lunch theorem for search [21], that states that no generalised search algorithm, such as random walking, can out-perform a specific search algorithm that makes use of suitable domain knowledge. The gradient topology contains implicit knowledge of peers' utilities and this knowledge is exploited by our gradient search algorithm, enabling its significant performance gains over random walking.

References

1. Sen, S., Wong, J.: Analyzing peer-to-peer traffic across large networks. Transactions on Networking **12** (2004) 219–232
2. Gummadi, K.P., Dunn, R.J., Saroiu, S., Gribble, S.D., Levy, H.M., Zahorjan, J.: Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In: Proceedings of Symposium on Operating Systems Principles. (2003) 314–329
3. Pouwelse, J., Garbacki, P., Epema, D., Sips, H.: The bittorrent p2p file-sharing system: Measurements and analysis. In: the 4th International Workshop on Peer-To-Peer Systems. (2005)
4. Bhagwan, R., Savage, S., Voelker, G.M.: Understanding availability. In: the 2nd International Workshop on Peer-to-Peer Systems. (2003)
5. Rhea, S., Geels, D., Roscoe, T., Kubiatowicz, J.: Handling churn in a dht. In: Proceedings of the USENIX 2004 Annual Technical Conference. (2004) 127–140
6. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. SIGCOMM Computer Communication Review **31**(4) (2001) 149–160
7. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S.: A scalable content-addressable network. In: Proceedings of the Conference on Applications, Technologies, Trchitectures, and Protocols for Computer Communications. (2001) 161–172
8. Rowstron, A.I.T., Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: Proceedings of the 18th International Conference on Distributed Systems Platforms. (2001) 329–350
9. Yang, B., Garcia-Molina, H.: Designing a super-peer network. In: Proceedings of the 19th International Conference on Data Engineering. (2003) 49–60

10. Zhao, B.Y., Duan, Y., Huang, L., Joseph, A.D., Kubiatowicz, J.D.: Brocade: Landmark routing on overlay networks. In: Proceedings of the 1st International Workshop on Peer-to-Peer Systems. (2002) 34–44
11. Montresor, A.: A robust protocol for building superpeer overlay topologies. In: Proceedings of the 4th International Conference on Peer-to-Peer Computing. (2004) 202–209
12. Mizrak, A.T., Cheng, Y., Kumar, V., Savage, S.: Structured superpeers: Leveraging heterogeneity to provide constant-time lookup. In: Proceedings of the 3rd IEEE Workshop on Internet Applications. (2003) 104–111
13. Manku, G.S., Bawa, M., Raghavan, P.: Symphony: Distributed hashing in a small world. In: Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems. (2003) 127–140
14. Yang, B., Garcia-Molina, H.: Improving search in peer-to-peer networks. In: Proceedings of the 22nd International Conference on Distributed Computing Systems. (2002) 5–14
15. Morselli, R., Bhattacharjee, B., Srinivasan, A., Marsh, M.A.: Efficient lookup on unstructured topologies. In: Proceedings of 24th ACM Symposium on Principles of Distributed Computing. (2005) 77–86
16. Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C., Zhao, B.: Oceanstore: An architecture for global-scale persistent storage. In: Proceedings of the 9th international Conference on Architectural Support for Programming Languages and Operating Systems. (2000) 190–201
17. Rao, A., Lakshminarayanan, K., Surana, S., Karp, R., Stoica, I.: Load balancing in structured p2p systems. In: the 2nd International Workshop on Peer-to-Peer Systems. (2003)
18. Jelasity, M., Babaoglu, O.: T-man: Gossip-based overlay topology management. In: the 3rd International Workshop on Engineering Self-Organising Applications. (2005)
19. Sacha, J., Dowling, J.: A self-organising topology for master-slave replication in p2p environments. In: Proceedings of the 3rd International Workshop on Databases, Information Systems and Peer-to-Peer Computing. (2005) 52–64
20. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA (1998)
21. Wolpert, D.H., Macready, W.G.: No free lunch theorems for search. IEEE Transactions on Evolutionary Computation **1**(1) (1997) 67–82