Using Aggregation for Adaptive Super-Peer Discovery on the 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 environments exhibit a very high diversity in individual peer characteristics ranging by orders of magnitude in terms of uptime, available bandwidth, and storage space. Many systems attempt to exploit this resource heterogeneity by using the best performing and most reliable peers, called super-peers, for hosting system services. However, due to inherent decentralisation, scale, dynamism, and complexity of P2P environments, self-managing super-peer selection is a challenging problem. In this paper, decentralised aggregation techniques are used to reduce the uncertainty about system properties by approximating the peer utility distribution allowing peers to calculate adaptive thresholds in order to discover appropriate super-peers. Furthermore, a heuristic search algorithm is described that allows super-peers, above a certain utility threshold, to be efficiently discovered and utilised by any peer in the system.[1]

1 Introduction

Measurements on deployed peer-to-peer (P2P) systems show that the distributions of peer characteristics such as the uptime, bandwidth, or available storage space, are highly skewed and often follow heavy-tailed distribution [1,2,3]. Researchers have also reported that the use of low stability or low performance peers can lead to poor performance in a P2P system [4,5]. Consequently, in order to improve the system efficiency, many P2P systems attempt to assign most important services to selected, high capability peers, called *super-peers* [6,7,8,9,10].

However, super-peer election in P2P environments poses a number of difficult problems. Due to the massive scale, dynamism, and complexity of P2P systems, it is not feasible for a peer or any other entity to maintain a global view of the system. Inherent decentralisation of P2P environments introduces uncertainty in decision making. Traditional election algorithms, such as the Bully algorithm [11], and other classical approaches to group communication [12], potentially require communication with all peers in the system, and can only be applied to small clusters of peers. Other approaches to super-peer discovery, based on

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

flooding or random walking [13], are difficult in large P2P systems due to high communication overhead that increases as the size of the P2P system grows. Solutions based on manual or static configuration of super-peers are inappropriate due to a lack of system-wide knowledge of peer properties.

This paper proposes a decentralised and self-managing approach to super-peer discovery based on gossipping. The paper introduces a peer *utility* metric in section 3 and applies a decentralised *aggregation* algorithm, covered in section 4, that generates a utility histogram in order to estimate the distribution of peer utility in the system. In section 5, a self-organising *gradient topology* is constructed based on the utility metric that allows peers to apply an efficient search heuristic for super-peer discovery described in section 6. The utility histogram is used for adaptive super-peer criteria selection. The presented approach is evaluated in section 7 and this evaluation shows that the aggregation algorithm provides a good approximation of peer utility distribution and that the search heuristic based on the utility metric achieves high search performance (significantly better than random walking). Section 8 concludes the paper.

2 Related Work

Recent research on P2P systems has been primarily focused on Distributed Hash Tables [14,15,16,17], 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 (high utility), and assuming that system services are placed on these peers, we provide a mechanism that allows the efficient discovery and consumption of these services. Furthermore, DHTs assume that peer identifiers are unique and relatively static, uniformly distributed in a key space. In our approach, the utility is dynamic and may follow any distribution with multiple peers potentially having the same utility value.

A number of P2P systems based on super-peers have been proposed. Yang and Molina [6] investigate general principles of designing super-peer-based networks, however, they do not provide any specific super-peer election algorithm. OceanStore [18] proposes 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 [8] improves routing efficiency in a DHT by exploiting resource heterogeneity, but unlike our approach, it does not address the super-peer election problem.

Chord [14,10] shows that the load between peers can be balanced by assigning multiple *virtual servers* to high performance physical hosts. The DHT structure may be used for the discovery of under- or over-loaded peers using *random sampling,* distributed *directories,* and other similar techniques. Mizrak et al [9] proposes the use of high capacity super-peers to improve routing performance in a DHT. However, these systems focus on load balancing in a DHT rather than the selection of potential super-peers from the set of all peers in the system.

Montresor [7] 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 approach introduces a continuous peer utility spectrum and approximates the distribution of peer utility in the system in order to discover peers above an adaptive utility threshold. Our neighbour selection algorithm can be seen as a special case of the T-Man protocol [19] that generates a gradient topology, where the ranking function is based on our peer utility metric. The advantage of such a utility ranking function is that applications can exploit the constructed topology in order to elect appropriate super-peers.

Kempe et al [20] describes a push-based gossip algorithm for the computations of sums, averages, random samples, and quantiles, and provides a theoretical analysis of the algorithm. Montresor, Jelasity and Babaoglu [21,22] introduce a push-pull-based approach for aggregate computation, however, their model assumes that message exchange between any two peers is atomic and that the clocks of peers are synchronised. We have extended Kempe's algorithm to calculate histograms, and we have added a peer leave procedure that improves the behaviour of the algorithm in the face of peer churn. We are using the aggregates for adaptive super-peer threshold calculation.

3 Peer Utility

This section introduces peer *utility* as a metric that captures the application-specific requirements and measures the capability of a peer to become a super-peer. Depending on the domain, the utility metric may involve a number of parameters. For example, in a P2P storage system, the utility may place greater emphasis on a peer's available local storage space and bandwidth. In a multimedia streaming application, the utility may combine a peer's latency and bandwidth, while in a grid computing system the utility may be a function of a peer's CPU load and availability.

A simple approach to utility calculation would be for each peer to individually compute their own utility. A more sophisticated utility metric may involve feedback from neighbouring peers. In either case, the utility of a peer is a local or microscopic property of a peer (or neighbourhood of peers). In an untrusted environment, a decentralised approach to trust may be adopted to prevent malicious peers from providing fake utility information about themselves.

Given that the utility of each peer in the topology can be calculated by a common function $U()$, the selection of super peers then becomes a question of how can an individual peer discover a high utility peer. In one possible approach, a peer may search for a super-peer above an absolute utility value. However, in many other applications, before a peer attempts to discover a high utility peer, the peer needs to estimate the distribution of peer utility in the system in order to know what constitutes high utility in a running system. For example, if an application requires the selection of the most stable peers in the system, it needs to learn the peer stability characteristics before it can decide on the stability threshold for a super-peer. Otherwise, if the super-peer threshold is static (hardwired), it may happen that no peer in the system satisfies the criteria, or that the threshold is very low and hence sub-optimal. Moreover, due to the system's dynamism, the super-peer selection criteria has to be continuously adapted to the system's current state and peer availability.

In the remainder of this paper, we describe a set of algorithms that provide solutions to the problems highlighted above. A decentralised aggregation technique is shown that allows peers to estimate the distribution of peer utility in the system and from this to identify an adaptive super-peer selection threshold. The gradient topology and gradient search heuristic are shown that enable the efficient discovery of (super)peers above a given utility threshold.

4 Aggregation Algorithm

Our approach to aggregation is based on the algorithms described by Kempe [20] and Montresor [21]. We adopt a push-based gossip model, since it can be implemented using asynchronous message passing and does not require synchronisation between peers.

In our approach, each peer continuously maintains estimates of a number of system properties by gossipping with neighbours. A peer, p , has an estimate of the current number of peers in the system, N_p , the maximum peer utility in the system, Max_p, and a cumulative histogram of peer utility values, H_p. Each of these values approximate the true system properties N^*, Max^*, and H^*. The cumulative utility histogram with B bins of width w is defined as

$$H(i) = \Big|\{p \mid U(p) \geq i \cdot w\}\Big| \tag{1}$$

for $1 \leq i \leq B$. Parameter B is also called the histogram resolution. The histogram is a discrete approximation of the peer utility distribution in B points, where each bin corresponds to a single point of the distribution function.

Peers joining the network contact any peer already in the system and obtain an initial set of neighbours and a current approximation of N^*, Max^*, and H^*. A newly joining peer has minimum utility, which is zero, and the maximum utility of any peer is unbounded. The number of histogram bins, B, is constant in the algorithm.

Peers periodically execute a gossip-based algorithm, where at one *step* (or *round*) of the algorithm a peer can send (*push*) messages to a number of neighbours and receive messages sent by its neighbours in the previous round. A sequence of steps that leads to a new approximation of N^*, Max^*, and H^* is called an aggregation *epoch*. An epoch can be potentially initiated by any peer at any time step, and the information about the newly created epoch is gradually propagated to all peers in the system. In order to distinguish between different, possibly overlapping, epochs, each epoch is tagged by a unique identifier selected by the initiating peer. Every peer p maintains a cache $cache_p$ that stores the identifiers of aggregation epochs that this peer has participated in.

The duration of an epoch is delimited by a time-to-live value. At the end of an epoch, every peer p updates its estimates N_p, Max_p, and H_p.

Algorithm 1: Aggregation algorithm at a peer p at round t.

```
with probability $\frac{1}{F \cdot N_p}$ send message $\big(rand(), TTL, 1, 0, 0, \frac{Max_p}{B}, 0\big)$ to self
forall epoch identifiers $id$ do
    let $\{m_i\}_{id}$ be messages received at round $t-1$ with epoch identifier $id$
    $M \leftarrow |\{m_i\}_{id}|$
    let $(id_i, ttl_i, w_i, n_i, max_i, hw_i, h_i)$ be $m_i$
    $ttl \leftarrow \sum_{i=1}^{M} \frac{ttl_i}{M}$
    $w \leftarrow \sum_{i=1}^{M} w_i$
    $n \leftarrow \sum_{i=1}^{M} n_i$
    $max \leftarrow \max(max_i)$
    for $1 \leq j \leq B$ do
        $h(j) \leftarrow \sum_{i=1}^{M} h_i(j)$
    end
    if $id \notin cache_p$ then
        $n \leftarrow n + 1$
        $max \leftarrow \max(U(p), max)$
        for $1 \leq j \leq \lfloor \frac{U(p)}{hw} \rfloor$ do
            $h(j) \leftarrow h(j) + 1$
        end
    end
    $cache_p \leftarrow cache_p \cup \{id\}$
    if $ttl < 1$ then
        $N_p \leftarrow \frac{n}{w}$
        $Max_p \leftarrow max$
        $H_p(j) \leftarrow \frac{h(j)}{w}$
    end
    else
        $m \leftarrow (id, ttl - 1, \frac{w}{2}, \frac{n}{2}, max, hw, h)$
        send $m$ to a random neighbours and to self
    end
end
```

Fig. 1. Aggregation algorithm at peer p at time t.

The algorithm performed at each step by a peer p is shown in Figure 1. In line 1, peer p starts a new aggregation epoch with probability $\frac{1}{F \cdot N_p}$. Thus, a new epoch is started by the system with average frequency $\frac{1}{F}$ (every F time steps). The epoch is initiated by creating an aggregation message with a new epoch id and a weight $w = 1$, as specified by Kempe. The ttl field is initialised with an $O(log(N_p))$ value, since informally speaking, the propagation speed of push-based epidemics is exponential and requires only $O(log(N_p))$ steps with high probability to reach all N peers in the system [20]. The histogram bin width is calculated as $hw = \frac{Max_p}{B}$. Furthermore, aggregation messages include a field used to estimate N^* labelled n, a field used to estimate Max^* labelled max, and finally, a histogram, h, consisting of B entries representing individual histogram bins. By combining all aggregation information in one message, the algorithm reduces the total number of messages generated, and thus limits the network traffic generated. For a 100-bin histogram, the aggregation message size is below 1KB.

In lines 2-12 of Figure 1, a peer performs the aggregation of received messages. A peer that receives an aggregation message with a new epoch identifier, i.e., with an *id* field that is not stored in the cache, joins this new aggregation (lines 13-20) by adding the value of 1 to its n field and to all histogram bins according to formula (1). If the *ttl* value is less than 1 (indicating the end of the epoch), a peer updates its current estimates of the system properties (lines 21-25). Otherwise, the peer emits a message to a random neighbour and to itself so that this peer will continue to participate during the next aggregation round (lines 26-30).

The algorithm exhibits the property of *mass conservation* defined by Kempe [20] provided that no peers fail during an aggregation epoch. At any time step, for each aggregation epoch, the sum of the weights of all aggregation messages in the system is always equal to one, i.e., $\sum_{i=1}^{N} w_i = 1$. Furthermore, the sum of n fields of all messages is equal to the number of peers participating in the aggregation, the maximum of max fields is equal to the maximum utility among peers participating in the aggregation, the average value of *ttl* fields of all messages at subsequent rounds decreases by one, and for $1 \leq j \leq B$, $\sum_{i=1}^{N} h_i(j) = H^*(j)$, where H^* is the utility histogram for peers participating in the aggregation.

In order to ensure mass conservation, each peer leaving the system is required to perform a leave procedure shown in Figure 2. In lines 1-11 of this figure, a peer aggregates currently buffered messages (as in lines 2-12 of Figure 1). In lines 12-15 of Figure 2, the peer subtracts the value of 1 from the n field and from the histogram bins. Finally, in lines 16-18, the peer sends a message containing the aggregated values to a random neighbour.

During one round of the aggregation algorithm, each peer participating in an epoch generates one aggregation message. The epochs are initiated on average every F rounds (frequency $\frac{1}{F}$), and since each epoch lasts on average TTL rounds, the average number of aggregation messages generated and received by each peer in one round is bounded by $O(\frac{TTL}{F})$, or $O(\frac{1}{F}log(N))$ if TTL is $O(log(N))$.

5 Gradient Topology

In this section, we introduce the self-organising gradient P2P topology and we outline its main properties. We briefly discuss the neighbour selection algorithm that generates the gradient topology. The topology is exploited by the gradient search heuristic.

Algorithm 2: Leave procedure at peer p

```
forall epoch identifiers $id$ do
    let $\{m_i\}_{id}$ be all currently buffered messages with epoch identifier $id$
    $M \leftarrow |\{m_i\}_{id}|$
    let $(id_i, ttl_i, w_i, n_i, max_i, hw_i, h_i)$ be $m_i$
    $ttl \leftarrow \sum_{i=1}^{M} \frac{ttl_i}{M}$
    $w \leftarrow \sum_{i=1}^{M} w_i$
    $n \leftarrow \sum_{i=1}^{M} n_i$
    $max \leftarrow \max(max_i)$
    for $1 \leq j \leq B$ do
        $h(j) \leftarrow \sum_{i=1}^{M} h_i(j)$
    end
    $n \leftarrow n - 1$
    for $1 \leq j \leq \lfloor \frac{U(p)}{hw} \rfloor$ do
        $h(j) \leftarrow h(j) - 1$
    end
    $m \leftarrow (id, ttl - 1, w, n, max, hw, h)$
    send $m$ to a random neighbour
end
```

Fig. 2. Leave procedure at peer p.

The gradient topology is a P2P topology where the highest utility peers are connected with each other and form a *core* in the system, while lower utility peers are located gradually farther from the core. The core, which clusters the highest utility peers in the system, corresponds to a set of super-peers in the system. Figure 3 shows a visualisation of a gradient topology. The position of each peer in the topology is determined by the peer's utility.

We have designed and evaluated a self-organising neighbour selection algorithm that generates the gradient topology in a completely decentralised P2P environment. Each peer p maintains two sets of neighbours, a *similarity-based* set, S_p, and a *random* set, R_p. 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 gradient structure of the topology. 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 significantly reduces the possibility of more than one cluster of high utility peers forming in the network. Random connections also reduce the possibility of the gradient topology partitioning due to excessive clustering. Moreover, random connections between peers are used by the aggregation algorithm described in section 4. Peer p removes a random entry from R_p or S_p

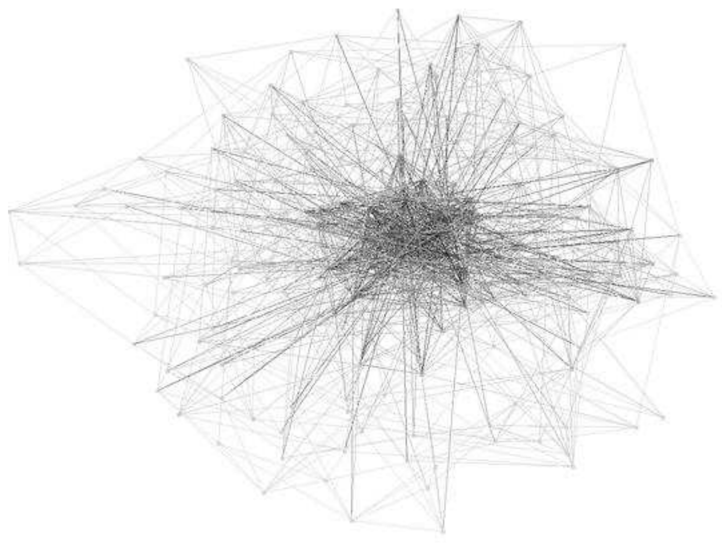


Fig. 3. Visualisation of a gradient topology.

if the number of entries in the sets exceeds the maximum allowed number of connections.

In addition to the neighbour sets, a peer p maintains a cache U_p that stores an estimated utility value, $U_p(q)$, for each neighbour q. Entries in the cache are timestamped and peers exchange these entries whenever they gossip.

Our initial evaluation of the neighbour selection algorithm, described in [23], shows that the algorithm generates a P2P topology with a very small diameter (an order of 5-6 hops for 100,000 peers) and that it has a global gradient structure.

The emergence of a gradient topology is a result of the system's self-organisation. Peers are independent, have limited knowledge about the system and interact with a limited number of neighbours. Utility can be considered as a microscopic property of a peer which enables through peer interaction the construction of the macroscopic gradient structure.

6 Gradient Search

The gradient structure of the topology allows us to develop an efficient search heuristic, called *gradient search*, that enables the discovery of high utility peers in the system. The algorithm exploits the information contained in the 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 [24].

The goal of the search algorithm is to deliver a message from any peer in the system to a super-peer in the core, i.e., to a peer with utility above a certain *threshold.* The value of the threshold is assigned by a peer that initiates the search and is calculated using the utility histogram generated by the aggregation algorithm described in section 4. The threshold is included in the search message. A peer below the specified utility threshold forwards search messages to higher utility peers until a peer is found whose utility is above the threshold. Each message is associated with a time-to-live (TTL) value that determines the maximum number of hops the message can be propagated.

In gradient search, each peer greedily forwards messages to its highest utility neighbour, i.e., to a neighbour q whose utility is equal to

$$max_{x \in S_p \cup R_p}\big(U_p(x)\big) \tag{2}$$

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 assumes that a higher utility peer is closer to the core in terms of the number of hops than a lower utility peer.

Local maxima should never occur in an idealised gradient topology, however, every P2P system is under constant churn and the gradient topology may undergo local perturbations from the idealised structure. In order to prevent message looping in the presence of such local maxima, a list of visited peers is appended to each search message, and a constraint is imposed that messages are never forwarded to already visited peers.

7 Experimental Evaluation

In this section, we describe our experimental setup and present the results of our experiments. The experiments evaluate the precision of the aggregation algorithm and the performance of gradient search.

The following notation and metrics are used. We measure the average error in histogram estimation, Err_H, defined as

$$Err_H(T) = \frac{\sum_{t=1}^{T}\sum_{p=1}^{N_t^*} D(H_t^*, H_{p,t})}{T \cdot N_t^*} \tag{3}$$

where N_t^*, $N_{p,t}$, Max_t^*, $Max_{p,t}$, $H_t^*$and $H_{p,t}$ correspond to N^*, N_p, Max^*, Max_p, $H^*$and H_p at time t of the experiment, T is the duration of the experiment, and D is a histogram distance function defined as

$$D(H_t^*, H_{p,t}) = \frac{\sum_{i=1}^{B} |H_t^*(i) - H_{p,t}(i)|}{\sum_{i=1}^{B} H_t^*(i)} \tag{4}$$

Similarly, we define Err_N as the average error in the estimation of N_t^*, and Err_{Max} as the average error in the estimation of Max_t^* over the course of the experiment.

We compare the performance of gradient search with random walking by measuring two properties of both algorithms. We calculate the average number of hops in which the algorithms deliver a search message from a random peer in the network to a super-peer in the core, i.e., to a peer above a certain utility threshold, and we measure the search failure rate, i.e., the percentage of search messages that are never delivered to super-peers.

The super-peer utility threshold is determined by each peer individually using the utility histogram calculated by the aggregation algorithm. A peer, p, sets the threshold, t_p, to a value that corresponds to 1% of highest utility peers. This value is approximated using the following formula

$$t_p = w \cdot max_{1 \leq i \leq B}(H_p(i) \geq 0.01 \cdot N_p) \tag{5}$$

where w is the histogram width and B is the number of bins in the histogram.

We evaluate the aggregation and search algorithms in a 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 (power law) with an exponent of 1.5 and a mean of 24 connections per peer.

The simulated P2P network is under constant churn. Every new peer p is assigned a session duration, s_p, according to the Pareto distribution with an exponent of $\gamma = 1.5$ and minimum value s_{min}. Thus, the expected session duration is given by the formula $E(s) = \frac{\gamma \ s_{min}}{\gamma - 1}$. We calculate the churn rate as the fraction of peers that leave (or join) the system at one step of the simulation. Over the lifetime of a running system, the average churn rate, $E(c)$, is equal to the inverse of the expected peer session time $E(s)$, therefore, in order to simulate a churn rate, c, in the system, we set s_{min} to

$$s_{min} = \frac{\gamma - 1}{\gamma \cdot c} \tag{6}$$

We assume that 10% of peers leave the system without performing the leave procedure of the aggregation algorithm (i.e., they crash).

A central bootstrap server is used that stores the addresses of peers that have most 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 20 neighbours from the bootstrap server. If a peer becomes isolated from the network (i.e., has no neighbours), it is bootstrapped again. The bootstrap server executes the aggregation algorithm and provides initial estimates of N^*, Max^*, and H^*, for peers entering the system.

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. Each peer continuously performs the neighbour selection and aggregation algorithms at every time step after it is bootstrapped. Additionally, at each turn, a number of randomly selected peers emit search messages that are routed using gradient search or random walking.

For the purpose of the simulation, in all experiments, the number of bins in the utility histogram is 100 , the aggregation frequency parameter F is 10 (except figure 4), and TTL is set to $3 \cdot log(N) + 10$ hops. The utility function of a peer p with uptime u and d maximum connections with neighbours is defined as $U(p) = d \cdot log(u + 1)$.

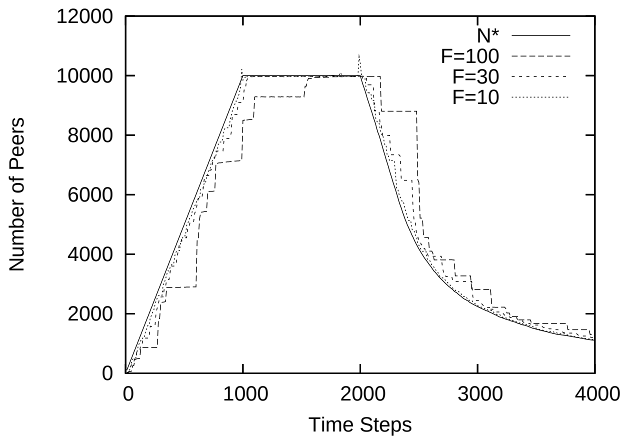
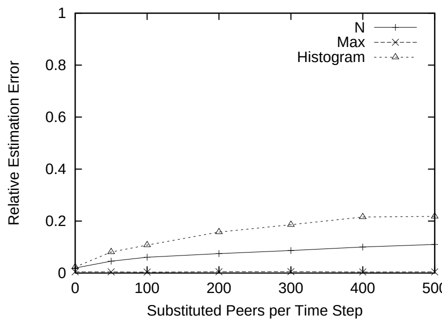


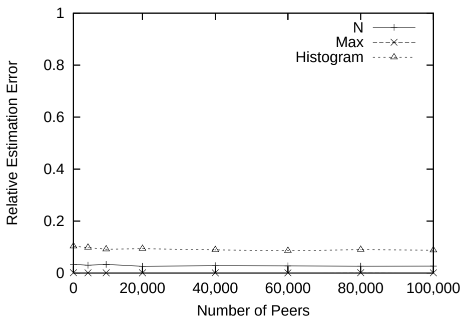
Fig. 4. Average estimation of the number of peers in the system (N) as a function of time. Three experiments are compared, with the frequency of aggregation (F) set to 100, 30, and 10 time steps.

Figure 4 shows the average precision of N^* estimation as a function of time and compares the results obtained for three different values of F. The best approximation, close to N^*, is obtained for $F = 10$. Random fluctuations are visible.

Figure 5 shows the average error of the aggregation algorithm, Err_N, Err_{Max}, and Err_H, as a function of the churn rate and as a function of the network size. The variance is not shown as it is approximately two orders of magnitude lower than the plotted values. The churn rate is measured as the number of substituted peers per time step. The estimation of Max^* is the most precise as the algorithm for the maximum calculation is simpler compared to the algorithm for H^* and N^* estimation. H^* approximation is less accurate than N^* since the histogram changes more dynamically than the number of peers. The relative error

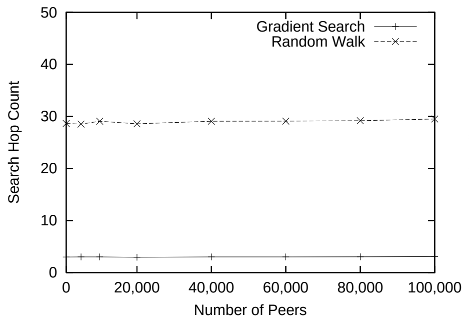


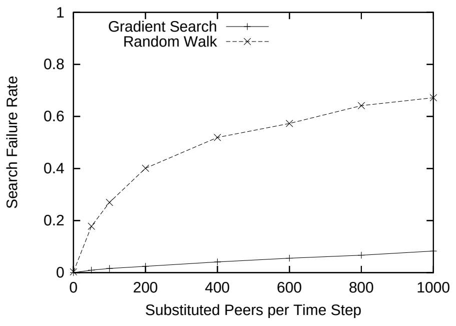
(a)



(b)

Fig. 5. Average estimation error of the number of peers in the system (N), maximum utility (Max), and the utility distribution (Histogram) as a function of peer churn rate (a) and network size (b).





(a)

Fig. 6. Searching for peers above super-peer utility threshold. Gradient search is compared with random walking. Average route length (hop count) of search messages as a function of the number of peers in the system (a) and average search failure rate as a function of peer churn rate (b).

as a function of the number of peers is bounded as the number of rounds in the epoch is proportional to $log(N)$, which corresponds to the theoretical analysis of Kempe.

Figure 6(a) shows the average hop count of search messages delivered to peers above the utility threshold as a function of the number of peers in the system. The hop count is nearly constant since the percentage of high utility peers in the system is fixed (1% of the system size). Figure 6(b) shows the average failure rate when searching for peers above the utility threshold as a function of peer churn rate. Both figures demonstrate superior performance of gradient search over random walk.

8 Conclusions

In this paper we have shown that the combination of a peer utility metric, aggregation techniques, and gradient topology with gradient searching allows the discovery of super-peers in peer-to-peer environments. Decentralised aggregation techniques reduce the uncertainty about the system by approximating peer utility distribution, and enable the decentralised and adaptive calculation of super-peer utility thresholds. The neighbour selection algorithm used in the gradient topology allows peers to self-organise themselves and to create a system-level gradient structure based on a local peer utility metric. The information contained in the topology enables the efficient searching for (super)peers above a given utility threshold.

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. Leibowitz, N., Ripeanu, M., Wierzbicki, A.: Deconstructing the kazaa network. In: Proceedings of the 3rd International Workshop on Internet Applications. (2003) 112–120
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. Yang, B., Garcia-Molina, H.: Designing a super-peer network. In: Proceedings of the 19th International Conference on Data Engineering. (2003) 49–60
7. 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
8. 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

9. 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
10. 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)
11. Garcia-Molina, H.: Elections in a distributed computing system. IEEE Transactions on Computers **31**(1) (1982) 48–59
12. Robbert van Renesse, K.P.B., Maffeis, S.: Horus, a flexible group communication system. Communications of the ACM **39**(4) (1996) 76–83
13. 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
14. 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
15. 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
16. 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
17. 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
18. 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
19. Jelasity, M., Babaoglu, O.: T-man: Gossip-based overlay topology management. In: the 3rd International Workshop on Engineering Self-Organising Applications. (2005)
20. Kempe, D., Dobra, A., Gehrke, J.: Gossip-based computation of aggregate information. In: Proceedings of the 44th IEEE Symposium on Foundations of Computer Science. (2003) 482–491
21. Montresor, A., Jelasity, M., Babaoglu, O.: Robust aggregation protocols for large-scale overlay networks. In: Proceedings of the International Conference on Dependable Systems and Networks. (2004) 19–28
22. Jelasity, M., Montresor, A.: Epidemic-style proactive aggregation in large overlay networks. In: Proceedings of the 24th International Conference on Distributed Computing Systems. (2004) 102–109
23. 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
24. Sacha, J., Dowling, J., Cunningham, R., Meier, R.: Discovery of stable peers in a self-organising peer-to-peer gradient topology. In: Proceedings of the 6th IFIP International Conference on Distributed Applications and Interoperable Systems. Number 4025 in LNCS, Springer-Verlag (2006) 70–83