Journal of Internet Services and Applications
a SpringerOpen Journal

RESEARCH Open Access

An efficient selection algorithm for building a super-peer overlay

Meirong Liu*, Erkki Harjula and Mika Ylianttila

Abstract

Super-peer overlay provides an efficient way to run applications by exploring the heterogeneity of nodes in a Peer-to-Peer overlay network. Identifying nodes with high capacity as super-peers plays an important role in improving the performance of P2P applications, such as live streaming. In this paper, we present a super-peer selection algorithm (SPS) to select super-peers for quickly building a super-peer overlay. In the SPS, each peer periodically builds its set of super-peer candidates through gossip communication with its neighbors, in order to select super-peers and client peers. Simulation results demonstrate that the SPS is efficient in selecting super-peers, and in quickly building a super-peer overlay. The proposed SPS also possesses good scalability and robustness to failure of super-peers.

Keywords: Peer-to-Peer, Super-peer, Overlay, Gossip

1. Introduction

Peer-to-Peer (P2P) overlay technologies have been widely applied for constructing large-scale network applications and services (*e.g.*, Skype [1], BitTorrent [2], Gnutella [3], and PPLive [4]) because of their inherent decentralization and redundant structures [5,6]. A lot of efforts have been made on P2P overlay construction (*e.g.*, [7-12]). Super-peer overlay (*e.g.*, Kazaa [13]) is an important type of P2P overlay. In a super-peer overlay, there are two types of peers: client peers and super-peers. Each client peer should connect to a super-peer in order to communicate with other peers in the overlay. A super-peer acts as the centralized server for its client peers and connects to other super-peers in the same way as in the pure P2P network [14]. Client peers with low capacity are shielded from massive query traffic by super-peers, which improves the scalability of the system and makes it feasible to connect *e.g.*, mobile devices to a P2P network. In the paper, the capacity of a peer refers to the combination of its available computational resource, network connections, and lifespan in the network.

Super-peer overlay enables applications to run more efficiently by exploring the heterogeneity of nodes in the overlay network. For example, in file sharing applications (*e.g.*, Gnutella [3]), Skype voice streaming, and live video streaming applications [15], the performance of these applications is improved through assigning nodes with high network bandwidth, long on-line time, or high processing capability as super-peers. Thus, super-peer selection in a given overlay is an important issue when building a super-peer overlay. On the other hand, particularly in dynamic network environment, it is common that peers join or leave (*e.g.*, a failure of a super-peer) a super-peer overlay [9,16,17]. To achieve a robust overlay, it should also be taken into account how to handle the failure of peers in the super-peer overlay construction. Therefore, efficient super-peer selection method in order to quickly build a robust super-peer overlay is an important research issue.

Many studies have been undertaken on building a super-peer overlay [7,8,10,12,14,18-22]. The principles and guidance of designing a super-peer overlay were addressed by Kirk [3] and Yang *et al.* [14], but no experimental results were reported. Some efforts utilized network proximity for building a super-peer overlay, *e.g.*, [8,12,18,21]. Client peers are connected with super-peers based on their distances. However, these studies focused on reducing communication latency between nodes by exploring network proximity rather than the efficiency of quickly building a super-peer overlay by exploring the

* Correspondence: meirong.liu@ee.oulu.fi
Media Team Oulu research group, Department of Computer Science and Engineering, University of Oulu, Oulu, Finland

capacity of nodes. Super-peer selection has to make a trade-off between reducing communication latency and selecting powerful super-peers. Other studies investigated the semantic similarity of peers when building a super-peer overlay, *e.g.*, [7,10,23]. Client peers that share the same interest are connected to same super-peers. However, these studies aimed to improve the search efficiency rather than the efficiency of quickly building a robust super-peer overlay. In addition, some super-peers could be overloaded because of popular content. The connections between super-peers and client peers in a super-peer overlay were also investigated [19,22,24,25]. However, these studies assumed that a super-peer overlay already exists and focused on managing an existing overlay instead of building an initial super-peer overlay. Wang *et al.* [15] presented a Labeled Tree to build a super-peer overlay. However, they aimed to achieve reliable high speed transmission in live stream rather than the efficiency of quickly building a super-peer overlay. In their study, super-peers are selected only based on the online-time, which did not take account of other information of nodes, *e.g.,* processing capacity, bandwidth. Montresor [11] proposed a gossip based algorithm SG-1 for the efficiency of quickly building a super-peer overlay. In SG-1, peers decide whether they should take a role of a super-peer by comparing their capacities with a randomly sampled neighbor peer, which is simple but takes long time to select needed super-peers. A super-peer searches and adds client peers only among its one-hop neighbors. All the peers in the overlay take the role of a super-peer in the beginning of the overlay construction.

In this paper, we focus on studying the efficiency of quickly building a robust super-peer overlay by taking account of capacity of nodes. Only peers that have high capacity (compared to neighbor peers) are selected as super-peers. The reason for this is that peers with a high capacity can contribute more to applications and enable applications to run more efficiently [3,15]. To this end, we present an efficient super-peer selection algorithm (SPS) to select peers with high capacity as super-peers for quickly building a robust super-peer overlay.

In the SPS algorithm, each peer maintains a set of super-peer candidates. Peers disseminate the information of super-peer candidates through a gossip method Newscast [26], which is efficient in information dissemination and enables peers to capture the dynamicity of a P2P overlay as well. Each peer periodically rebuilds its set of super-peer candidates and decides whether it takes the role of a super-peer based on its set of super-peer candidates, which differs from the super-peer selection method presented in SG-1 [11]. After that, peers execute the corresponding operations according to their roles: joining a super-peer or recruiting client peers. In the SPS, all the peers act as a client peer in the beginning of the overlay construction.

Overall, the proposed SPS algorithm has the following contributions:

(1) The SPS algorithm introduces a set of super-peer candidates for each peer to select super-peers, which enables the algorithm to select peers with high capacity as super-peers and consequently reduces the time of super-peer selection.
(2) The SPS algorithm employs a conditional two-top search method for super-peers to find and add client peers, which reduces both the time for building an overlay and the communication overhead.
(3) The SPS algorithm achieves a comparable robustness, and better performance in terms of convergence time, scalability, compared to related work SG-1 [11].

This paper extends our previous work [20]. In this paper, we provide a more in-depth development and analysis of our SPS algorithm. We also carry out additional experiments to evaluate the performance of our SPS algorithm compared to related work SG-1 [11]. The remainder of the paper is organised as follows: in Section 2, we briefly explain a communication method, called Newscast, utilized by the SPS, and present the SPS algorithm. In Section 3, we evaluate performance of the SPS in terms of convergence time, communication overhead, scalability, and robustness. In Section 4, we conclude the paper with future directions.

2. The SPS algorithm

In this section, we first give the background of a gossip based communication method utilized in the SPS (called Newscast [26]), and then present the design rationale. After that, we provide the general idea of the SPS algorithm. Finally, we depict the details of the SPS algorithm for super-peer overlay construction. We consider a set of nodes connected through an existing network, and assume that each node stores identifiers of its neighbors. Each node can directly or indirectly communicate with other nodes via its neighbors. In this paper, we consider a dynamic network environment and nodes may join or leave the overlay network at anytime. A node's information, such as its identifier, available resources, current role (*i.e.,* client peer or super-peer), neighbors, and lifespan are assumed to be disseminated through Newscast. A peer can capture the dynamicity of the network through message exchange of Newscast.

2.1. Background of a communication method utilized by the SPS algorithm

In our work, we use Newscast [26] (a gossip protocol that maintains a dynamic random topology) for information

dissemination between peers. Newscast has been used for P2P applications, such as broadcast [24] and aggregation [26] for its effective information dissemination. Furthermore, Newscast is designed for dynamic environment and enables peers to capture the dynamicity of an overlay, *e.g.,* joining of a new node or leaving of a node [26].

The general idea of Newscast is as follows: in Newscast, each node maintains a partial view that is constituted of a fixed-size set of peer descriptors. A peer descriptor is composed of the information of the address of a node, a timestamp identifying when the descriptor is created, and application specific information. Each node periodically exchanges and merges its partial view with a randomly selected node to get an up-to-date partial view and a better approximation of the target topology. More information can be found in Jelasity *et al.* [26].

2.2. Design rationale

We build a super-peer overlay as an additional overlay imposed on top of an existing connected topology (*i.e.,* a random graph), which is maintained by Newscast. In a random graph, peers are randomly connected with each other. The neighborhood of a node in the SPS algorithm is set as follows: if a node is a super-peer, it connects to a random sample of other super-peers and to the set of client peers that are managed by this super-peer. If a node is a client peer, it connects to only one super-peer. The initial role of all the nodes is a client peer. The SPS uses a node's capacity as a criterion for selecting a super-peer candidate and a super-peer. The SPS selects as few super-peers as possible when building a super-peer overlay, which aims to maximize the contribution of nodes with high capacity. Inspired by the method called VoRonoi Regions for mobile network [27], the SPS selects super-peers first during the super-peer overlay construction. Note that the reason for building a super-peer overlay on top of a connected overlay is to avoid a danger of disconnection if a large number of super-peers are failed.

The SPS overlay interacts with the Newscast overlay (*i.e.,* the connected overlay) to disseminate information of the set of super-peer candidates utilized by the SPS. Specifically, when a node disseminates the information of a set of super-peer candidates (which contains samples of super-peer candidates), this node needs to interact with Newscast overlay to get a random peer from the connected Newscast overlay. Then, this node sends its partial view to the randomly selected node to exchange and update their descriptors. Finally, the descriptor of this node contains not only an identifier, a time stamp, but also a set of super-peer candidates, which in its turn is used by the upper SPS overlay for selecting super-peers.

2.3. Overview of the SPS algorithm

The general idea of the SPS algorithm for selecting super-peers and client peers is as follows: all the nodes in the overlay periodically perform operations (1) and (2) described below until the super-peer overlay is built.

(1) A node n_i rebuilds its super-peer candidates CanSP(n_i) through communication with its neighbors for their super-peer candidates. Nodes with higher capacity (chosen from retrieved super-peer candidates) are promoted as super-peer candidates (*i.e.*, added into CanSP(n_i)).

(2) By checking whether there is a change in the rebuilt CanSP(n_i) retrieved in (1) (*i.e.*, whether new super-peer candidates are rebuilt into CanSP(n_i) or not), n_i will perform one of the operations (a) or (b) given below.

   (a) If there are new super-peer candidates added in CanSP(n_i), n_i notifies its neighbors of the new super-peer candidates and has its role determined again. Specifically, if n_i is a client peer and belongs to CanSP(n_i), n_i changes its role to become a super-peer. If n_i is a super-peer but it does not belong to CanSP(n_i), n_i changes its role to become a client peer and transfers its client peers (if any) to other super-peers.
   (b) If CanSP(n_i) is not changed, n_i proceeds as follows: if n_i is a super-peer, it searches and adds client peers until n_i is fully loaded or no more client peers can be found. If n_i is a client peer and belongs to CanSP(n_i), n_i changes its role to become a super-peer. If n_i is a client peer and does not belong to CanSP(n_i), and has not joined a super-peer, n_i searches and joins a super-peer.

Note that initially each node n_i takes the role of a client peer and sets its super-peer candidates CanSP(n_i) to be itself and its super-peer to be null. Then, each node starts to perform the SPS algorithm described above to build a super-peer overlay. The super-peer candidates of each node are rebuilt periodically and the role of each node could be changed dynamically during the super-peer overlay construction. When a new node n_i joins the overlay, it declares itself as a client peer, sets its super-peer candidates CanSP(n_i) to be itself and its super-peer to be null. Then, this node executes the SPS algorithm.

2.4. Detailed description of the SPS

Before presenting the details of the SPS algorithm, the notations used in the SPS are summarized as follows:

(1) n_i denotes a node in an N-node P2P overlay network. n_i has only one of the exclusive roles: client peer or super-peer. n_i has two optional states, *i.e.,* State(n_i) = {normal, failed}. The former denotes that n_i is part of the overlay without suffering from a failure, and the latter denotes that n_i is failed. Each client peer maintains three sets of data: its neighbors, its super-peer, and a set of super-peer candidates. Each super-peer in its turn maintains the data of its neighbors, a set of super-peer candidates, and a set of client peers.

(2) SP(n_i) denotes the super-peer of node n_i.

(3) C(n_i) represents the capacity of node n_i. C(n_i) is the aggregation of three resource metrics: computational resource (CPU cycles, storage), network bandwidth, and lifespan. It is represented as follows:

$$C(n_i) = \sum_{k=1}^{3} w_k * v_k, \quad (1)$$

where w_k is the weight of the k^{th} resource metric, and v_k is the value of the k^{th} resource metric. Each metric of C(n_i) has a different weight, which can be set according to specific applications. The value of C(n_i) is set in the beginning of building a super-peer overlay. For simplicity, we omit computation details of C(n_i) and assign the number of client peers that n_i can manage to C(n_i), which does not affect the presentation of the SPS.

(4) CanSP(n_i) denotes the set of super-peer candidates of n_i. It is used to judge the role of n_i. Nodes with high capacity are promoted as super-peer candidates. The number of super-peer candidates included in CanSP(n_i) is computed as: $\frac{overlaysize}{super-peers' maximum\ capacity}$. Each node stores the overlay size and computes the number of super-peer candidates when building a super-peer overlay. It should be noted that in real life, the overlay size is retrieved by utilizing an underlying gossip aggregation protocol to compute the number of nodes in the overlay.

(5) Ld(n_i) denotes the workload of node n_i, which shows how many client peers is managed by n_i. If the workload Ld(n_i) is lower than C(n_i), n_i is set as under-loaded, otherwise n_i is set as full-loaded.

Figures 1 and 2 show the detailed actions of a client peer and a super-peer that are running the SPS algorithm for building a super-peer overlay. Table 1 summarizes all the basic operations used in Figures 1 and 2. According to Figure 1, a client peer n_i running the SPS algorithm acts as follows: (1) in the case when n_i has joined a super-peer SP(n_i): if SP(n_i) is failed, n_i calls a super-peer-failure handler, otherwise n_i does nothing. (2) In the case when n_i has not joined a super-peer, (2.1) n_i checks whether there is a message about a failed super-peer. If there is such a message, n_i removes the failed super-peer from CanSP(n_i). (2.2) n_i checks whether there is a message about a new set of super-peer candidates. If such a message exits, n_i updates its CanSP(n_i). (2.3) n_i checks whether the super-peer candidates of its neighbors have higher capacity than its CanSP(n_i). If the super-peer candidates of n_i's neighbors have higher capacity than CanSP(n_i), n_i retrieves a new CanSP(n_i) based on super-peer candidates of its neighbors and has its role judged again according to the new CanSP(n_i). Otherwise, n_i searches and joins a super-peer.

As shown in Figure 2, an under-loaded super-peer n_i running the SPS algorithm proceeds as follows: (1) similar to the operation of a client peer shown in Figure 1, super-peer n_i checks whether there is a message about a new set of super-peer candidates. If such a message exits, n_i updates its CanSP(n_i). (2) Similar to the operation of a client peer shown in Figure 1 again, super-peer n_i also checks whether super-peer candidates of its

```
Input: A client peer n_i
Operation:
if SP(n_i) ≠ null then
    if State(SP(n_i)) ==failed then
      SPFailureHandler(n_i).
    else Do nothing and return.
    end if
else
    RemoveFailedSPIfExist(n_i).
    if UpdSP = RetriveUpdateSPc(n_i) then
      UpdateSPCandidates(n_i , UpdSP).
    end if
    if ExistBiggerSPCandidate(n_i) ==true then
     CanSP(n_i) = RetriveNewSPCandidates(n_i).
      NotifyNewSPcandidates(n_i).
        if n_i ∈ CanSP(n_i) then
          ChangeRole(n_i).
        end if
    else
          if n_i ∉ CanSP(n_i) then
            Search&JoinUnderLoadedSP(n_i).
          else
            ChangeRole(n_i).
          end if
    end if
end if
```

Figure 1 The action of a client peer running the SPS.

Input: A super-peer n_i and n_i is under-loaded
Operation:
if *UpdSP* = RetriveUpdateSPc(n_i) **then**
UpdateSPCandidates(n_i , *UpdSP*).
end if
if ExistBiggerSPCandidate(n_i) == true **then**
CanSP(n_i) = RetriveNewSPCandidates(n_i).
NotifyNewSPcandidates(n_i).
if $n_i \notin$ CanSP(n_i) **then**
TransferClients(n_i).
ChangeRole(n_i).
end if
else
AddClientnodes(n_i).
end if

Figure 2 The action of a super-peer running the SPS.

neighbors have higher capacity than its CanSP(n_i). If the super-peer candidates of n_i's neighbors have higher capacity than CanSP(n_i), n_i retrieves a new CanSP(n_i) and has its role determined based on the new CanSP(n_i). If n_i cannot keep the role of super-peer, n_i transfers its client peers (if any) to other super-peers and changes its role to be a client peer. In contrast, if super-peer candidates of n_i's neighbors do not havehigher capacity than CanSP(n_i), the role of n_i is not changed. Then, n_i searches and adds client peers until n_i is fully loaded or no more client peers can be found.

Note that the operation AddClientnodes(n_i) (shown in Figure 2) employs a conditional two-hop search method for a super-peer to find client peers. That is, n_i increases search step from one hop to two hops in the condition of a worst case. Herein, the worst case is that a super-peer n_i manages some client peers and n_i finds that all neighbors of n_i have joined super-peers after it searches its neighbors. The worst case would increase the convergence time of building a super-peer overlay dramatically if only one-hop search method is used.

In the face of the worst case described above, if n_i changes its role to be a client peer and searches a super-peer, both the convergence time of the SPS algorithm and the network traffic would increase. Specifically, on one hand, n_i needs to transfer its client peers to other super-peers, which would increase the network traffic. On the other hand, after n_i changes its role to be a client peer, n_i needs to perform the SPS algorithm to find a super-peer, which in its turn would increase the

Table 1 The primitive operations used in the SPS algorithm

Operation	Description
AddClientnodes(n_i)	Node n_i adds client nodes until n_i is full-loaded or no more client nodes can be found. Specifically, n_i first searches its neighbors to find and add nodes. After that, if n_i is still under-loaded, n_i searches its neighbors' neighbors to find and add client nodes.
ChangeRole(n_i)	Node n_i changes its role.
ExistBiggerSPCandidate(n_i)	n_i sends messages to its neighbors to check whether its neighbors' super-peer candidates have higher capacity than CanSP(n_i). If that is the case, true is returned. Otherwise, false is returned.
NotifyNewSPcandidates (n_i)	n_i notifies its neighbors to update their sets of super-peer candidates to be CanSP(n_i).
RetriveUpdateSPc(n_i)	Node n_i checks whether there is a message about new super-peer candidates *UpdSP*. If such a message exists, *UpdSP* is returned. Otherwise, null is returned.
RetriveNewSPCandidates(n_i)	Node n_i computes its new set of super-peer candidates CanSP(n_i) according to its retrieved set of super-peer candidates through communication with its neighbors.
RemoveFailedSPIfExist(n_i).	Node n_i checks whether there is a message about super-peer failure. If such a message exists, the failed super-peer contained in the message is removed from the super-peer candidates of n_i.
Search&JoinUnderLoadedSP(n_i)	Node n_i sends query messages to super-peers contained in its super-peer candidates to check these nodes' workloads. If a super-peer that is under-loaded is found, n_i joins this super-peer.
SPFailureHander(n_i)	When the super-peer of n_i (*i.e.,* SP(n_i)) fails, n_i removes SP(n_i) from its set of super-peer candidates and empties its super-peer. n_i also notifies its neighbors of the failure of SP(n_i).
TransferClients(n_i)	When a super-peer n_i has to change its role to be a client peer, n_i transfers its client peers (if any) to super-peers that are randomly selected from its set of CanSP(n_i).
UpdateSPCandidates(n_i,*UpdSP*)	Node n_i updates its set of super-peer candidates to be *UpdSP,* which is an ordinary set to store updated super-peer candidates.

convergence time of the SPS. In contrast, when n_i increases the search step conditionally to two hops, a faster convergence time at the expense of network traffic is achieved by the SPS. Specifically, n_i searches the neighbors of its neighbors to find more client nodes in the face of the worse case, which reduces the convergence time of building a super-peer overlay. On the other hand, only a little more communication overhead is generated when increasing the search step conditionally to two hops, because super-peers that perform the two-hop search take only a very small portion of the whole peers. More importantly, our work aims to quickly build a super-peer overlay. Thus, we employ a conditional two-hop search method for a super-peer to find client peers in AddClientnodes(n_i).

3. Performance evaluation

In this section, we describe simulations conducted for evaluating the feasibility and performance of the SPS algorithm. First, we introduce experimental settings. Then, we evaluate performance of the SPS from four aspects: convergence time, communication overhead, scalability, and robustness, respectively.

3.1. Experimental setup

We use PeerSim [28] to carry out simulations. In PeerSim, one simulation round means that all the nodes finish performing deployed protocols once. Four performance metrics are emphasized in the experiments: (1) convergence time of the SPS and the impact of parameters (*e.g.*, the maximum capacity of super-peers) on the SPS's convergence time, compared to related work SG-1 [11]; (2) communication overhead compared to SG-1; (3) scalability in comparison to SG-1, and (4) the SPS's robustness to failure of super-peers compared to SG-1. The overlay size for simulations is set as 10^5 unless separately specified. All the peers take the role of client peer in the beginning of simulations. The initial overlay topology adopted in the simulation is a random graph, where all the peers are randomly connected with each other. The initial random graph topology provides a good chance to verify the efficiency of the SPS because the initial overlay is far from the converged super-peer overlay.

3.2. Evaluation of the convergence time

In this section, we evaluate how fast the SPS can converge, and how parameters affect the convergence time of the SPS. Two types of distributions for nodes' capacity are evaluated: the uniform distribution and the power-law distribution. Simulation results are depicted in Figures 3 and 4.

Figure 3 shows the convergence of the SPS, *i.e.,* the variation of the number of client peers that have joined super-peers as the simulation goes on. It should be noted that simulations converge in the condition that no more client peers will join a super-peer. Specifically, when the capacity of peers follows the uniform distribution, it takes about 7 simulation rounds for all the client peers to finish selecting and joining super-peers (*i.e.,* building a super-peer overlay). When the capacity of peers follows the power-law distribution, it takes about 4 rounds. The results show that the SPS performs well when measured with the convergence time for building a super-peer overlay. At simulation round 4, client peers that have joined super-peer in uniform distribution are almost the same as those in power-law distribution. However, in uniform distribution, the SPS algorithm continues running until by simulation round 7 because there are still nodes that have not joined super-peers at simulation round 4. This result shows that different distributions of peers' capacity lead to different convergence times (*i.e.,* different numbers of needed simulation rounds). In other words, distribution of peers has an impact on the convergence time (*i.e.,* simulation rounds) of the SPS algorithm [29]. The number of selected super-peers in power-law distribution is 370 according to the SPS algorithm. In contrast, the number of selected super-peers in uniform distribution is only 201. The reason is that in power-law distribution, only a small portion of peers have a relatively high capacity, and more super-peers are selected for managing client-peers when building the super-peer overlay. The result shows that the number of selected super-peers is related to the distribution of peers' capacity.

Figure 4 shows the impact of super-peers' maximum capacity on the SPS's convergence time compared to SG-1 [11]. Figure 4a depicts the number of selected super-peers in the target super-peer overlay as maximum capacity of super-peers increases. Figure 4b illustrates the number of needed simulation rounds for convergence as maximum capacity of super-peers increases.

The result shown in Figure 4a is as expected: the larger the maximum capacity of super-peers (*i.e.,* a super-peer can manage more client peers according to the capacity defined in Section 2.2), the fewer super-peers are selected in the converged super-peer overlay.

According to Figure 4b, when the maximum capacity of super-peers increases, more simulation rounds are needed for convergence. Specifically, the SPS takes a few more simulation rounds to converge (*i.e.,* from 8 to 9 rounds), but SG-1 gains an obvious increase in simulation rounds for convergence (*i.e.,* from 8 to 14 rounds). This result shows that the SPS is less affected by the variation of super-peers' maximum capacity compared to SG-1.

The reason for different convergence times between the SPS and SG-1 is that different super-peer selection and search methods are used in the SPS and SG-1.

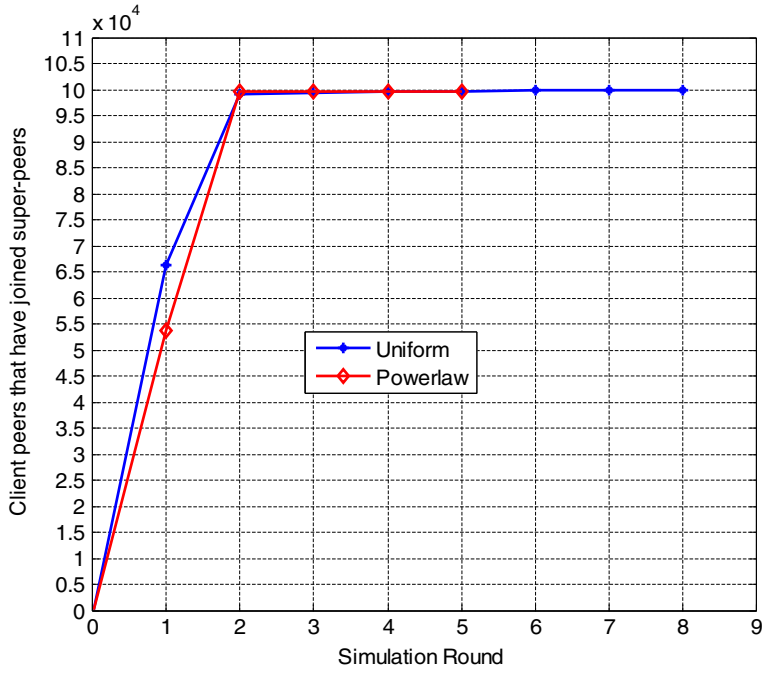

Figure 3 The convergence of the SPS during the simulation.

Specifically, when the maximum capacity of super-peers increases (*i.e.,* a super-peer can manage more client peers), the number of the required super-peers decreases. In the SPS, a set of super-peer candidates is built for selecting peers with very high capacity as super-peers, and a conditional two-hop search method is employed for super-peers to quickly find client peers. Even when the number of the required super-peers decreases, most of the super-peers can still be selected through super-peer candidates during the first few simulation rounds. As soon as super-peers are selected, client peers can quickly join a super-peer and super-peers can quickly find and add client peers with a conditional two-hop search method (which makes the SPS converge even faster). However, for SG-1, when the number of the required super-peers decreases, more super-peers need to change their role to be client peers, since the initial role of all the peers is a super-peer. Moreover, super-peers only compare their capacities with one of their neighbors to determine their role and search client peers among its one-hop neighbors. Thus, it requires more simulation rounds of message exchanges to finish

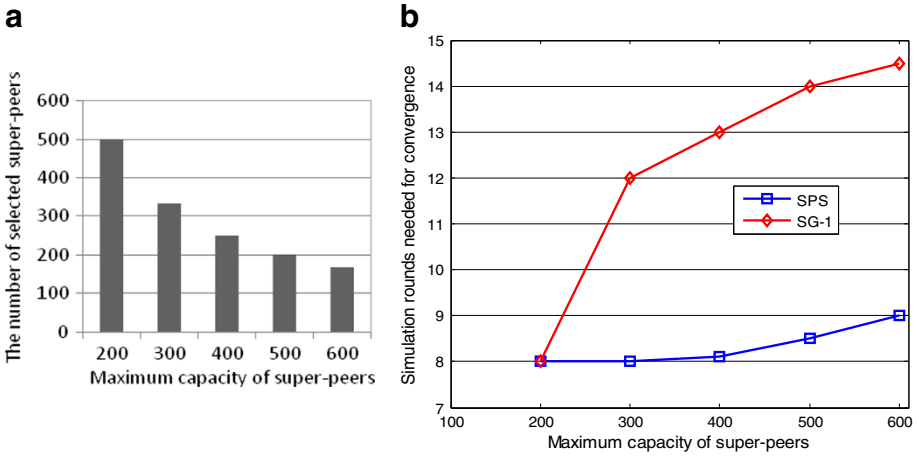

Figure 4 (a) Selected super-peers with varied maximum capacity of super-peers. (b) The impact of super-peers' maximum capacity on the convergence time.

selecting super- peers. Therefore, it takes longer for SG-1 to converge compared to the SPS when the maximum capacity of super-peers increases.

3.3. Evaluation of the communication overhead

In this section, we evaluate the communication overhead, *i.e.,* the number of messages that are transmitted between peers during a super-peer overlay construction. Three types of communication overhead are evaluated: (1) the total number of probes per node for query about the load of neighbor super-peers, (2) the number of gossip messages per node for building super-peer candidates, and (3) the number of client peer transfers per node. Herein, the term client peer transfer means that client peers are transferred to other super-peers when their super-peers change their roles to be a client peer. For simplicity, we use the average value of these three types of communication overhead for presenting results. Figure 5 shows the results and the comparison between the SPS and SG-1.

According to Figure 5, one can find out that: (1) for the SPS algorithm, the number of probes is independent from the overlay size and approximately one probe per node is sent for query about workload of super-peer (as shown in Figure 5a. (2) Taking account of the total communication overhead (*i.e.,* the sum of the three types of communication overhead), as shown in Figure 5a and b, the SPS is comparable to SG-1. Specifically, the number of communication messages generated in the SPS for building super-peer candidates per node is a little bit large. However, the number of client peer transfers in the SPS (*i.e.,* 0.04 on average) is much smaller than that of SG-1 (*i.e.,* 9.5 on average). In other words, maintaining the set of super-peer candidates in the SPS generates more communication overhead; nevertheless it makes a significant positive effect on reducing the number of client peer transfers (as shown in Figure 5b).

The reason for the much lower number of client transfers in the SPS compared to SG-1 is as follows: the SPS selects peers with the highest capacity among its neighbors to be super-peer candidates and then picks up super-peers from the super-peer candidates. Thus, only a small portion of super-peers change their roles to be a client peer and the number of transferred client peers (because of role change of these super-peers) is low. However, in SG-1, one peer exchanges gossip messages with randomly selected neighbors to decide whether it keeps the role of super-peer or changes its role to be a client peer. Thus, one peer could frequently change its role when it compares its capacity with different neighbors. Consequently, client peers managed by these super-peers generate a large number of client peer transfers.

3.4. Evaluation of the scalability

In this section, we verify the scalability of the SPS in terms of convergence time. In other words, we examine the variation of the SPS's convergence time while the number of peers increases from 1,000 to 100,000. Two types of distributions for peers' capacity are examined:

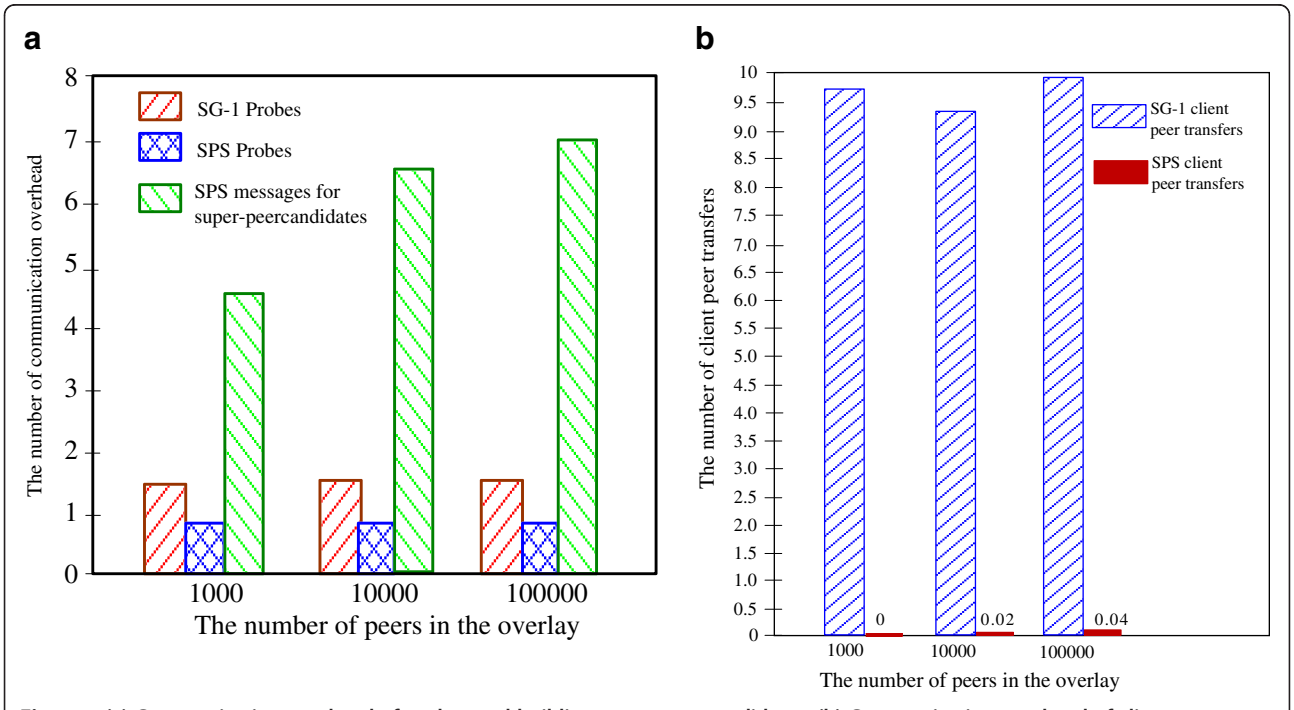

Figure 5 (a) Communication overhead of probes and building super-peer candidates. (b) Communication overhead of client peer transfers.

the power-law distribution and the uniform distribution. In addition, we compare the scalability of the SPS with that of SG-1. Figure 6 and Table 2 show the results and the comparison between SPS and SG-1.

According to Figure 6 and Table 2, one can find out that: (1) the SPS scales well and the number of needed simulation rounds for building a super-peer overlay grows gradually when the number of peers increases from 1,000 to 100,000. Specifically, the number of simulation rounds increases from 2.0 to 7.0 rounds for the uniform distribution, and from 3.0 to 12.0 rounds for the power-law distribution. (2) The deviation on the simulation rounds for convergence in the SPS is smaller than that in SG-1 when the total number of peers in the overlay increases (as shown in Table 2). For example, for the uniform distribution, the number of simulation rounds in the SPS increases from 2.0 to 7.0 rounds with a standard deviation of 2.5, and it increases from 7.0 to 19.0 rounds with a standard deviation of 6.0 in SG-1, as shown in Table 2. This result shows that the SPS has better scalability than SG-1.

The rationale behind result (1) mentioned above is that although there is a huge increase in the number of nodes in the overlay, most of the required super-peers can still be selected during the first few simulation rounds by all the peers executing the SPS algorithm. The rest of the required super-peers can be selected in the following simulation rounds. After that, client peers join super-peers, and super-peers search and add client peers using a conditional two-hop search (as explained in the end of Section 2). Therefore, the increase in the overlay size has only a little impact on the needed simulation rounds for building a super-peer overlay.

The reason for result (2) mentioned above, is as follows: in the SPS, most of the required super-peers can be selected in the first few simulation rounds using the set of super-peer candidates. As soon as super-peers are identified, client peers can join super-peers and super-peers can search for client peers. However, for SG-1, when the overlay size increases, firstly, more super-peers need to change their roles to be client peers since the initial role of all the nodes is a client peer. Hence, more simulation rounds are inflicted on the convergence time. Secondly and more importantly, in SG-1, super-peers only compare their capacities with one of their neighbors to determine their role, which increases the time of selecting peers with high capacity as the target super-peers and the convergence time. In addition, in SG-1, a super-peer n_i only searches client peers among its one-hop neighbors. When n_i is still under-loaded after searching all its one-hop neighbors, n_i has to wait for client peers to join it. Thus, it takes longer for n_i to find client peers. As the overlay size increases, nodes like n_i increase and much more time is required for SG-1 to converge. Taking into account the three factors mentioned above, it takes longer for SG-1 to build a super-peer overlay when the overlay size increases.

3.5. Evaluation of the robustness

In this section, we verify the robustness of the SPS in the face of super-peers' failure and compare the result with that of SG-1. We examine three catastrophic

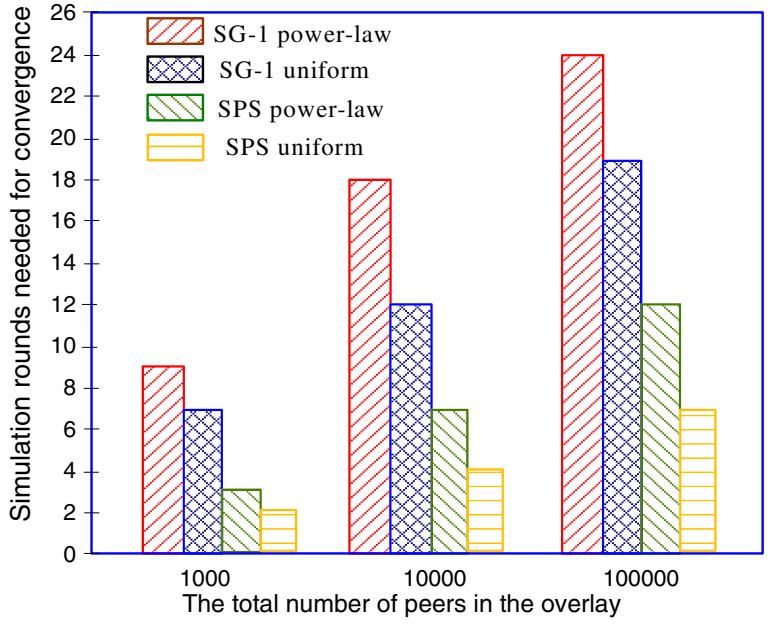

Figure 6 The scalability in terms of convergence time.

Table 2 Scalability of the SPS compared to SG-1

Algorithm	Capacity distribution	Simulation rounds needed with 1,000 peers	Simulation rounds needed with 10,000 peers	Simulation rounds needed with 10,0000 peers	Standard deviation of the simulation rounds needed as the overlay size increases
The SPS	Uniform	2.0	4.0	7.0	2.5
	Power-law	3.0	7.0	12.0	4.5
SG-1	Uniform	7.0	12.0	19.0	6,0
	Power-law	9.0	18.0	24.0	7.6

scenarios: (a) 10% of super-peers are removed at the sixth simulation round, (b) 20% of super-peers are removed at the sixth simulation round, and (c) 30% of super-peers are removed at the sixth simulation round. Results are shown in Figure 7 and Table 3.

According to Figure 7 and Table 3, one can find out that: the robustness of SPS is comparable to SG-1 in the face of super-peer failure when taking into account both the convergence time and the impact of super-peers' failure on client peers. For example, in the case of the

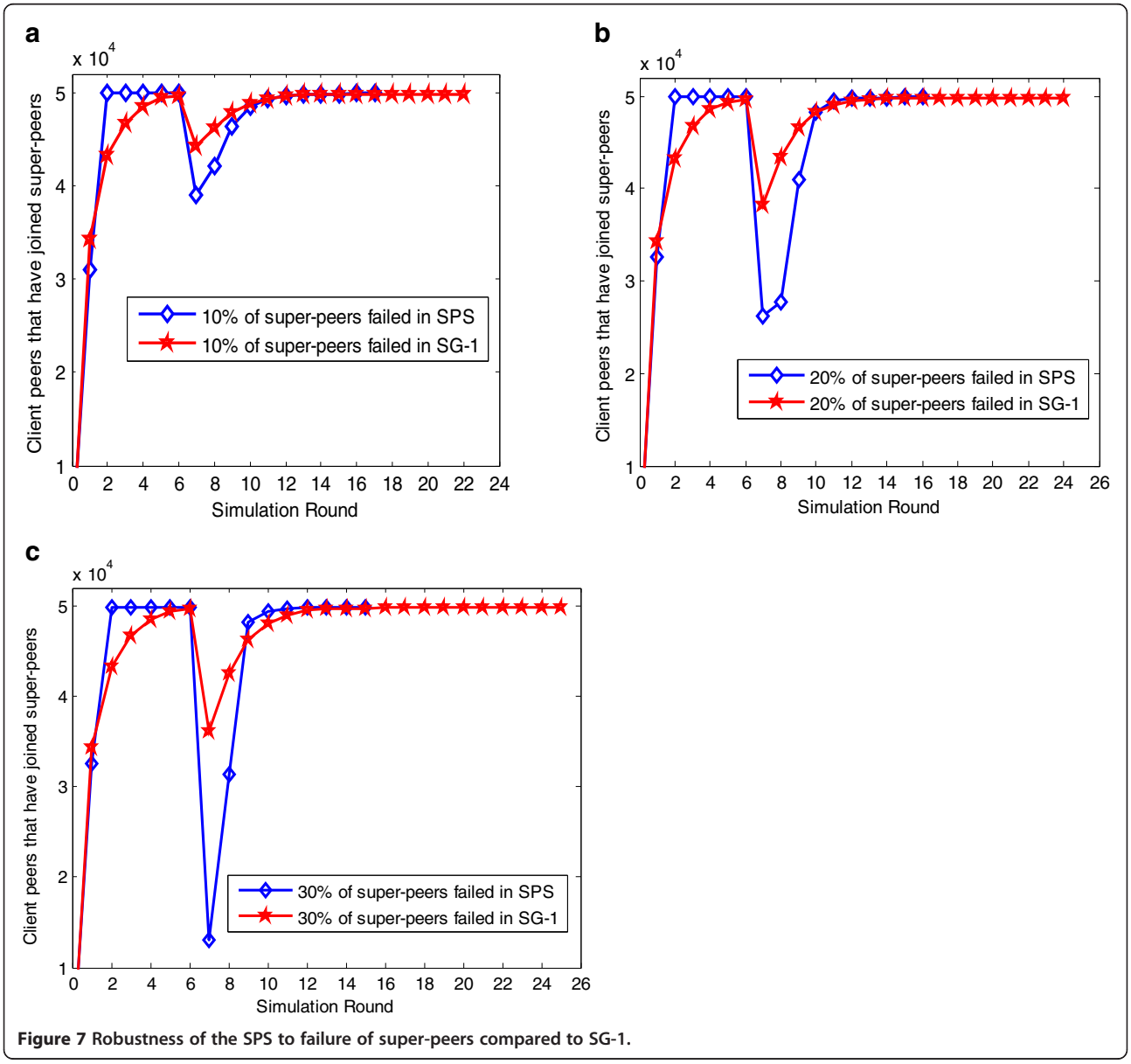

Figure 7 Robustness of the SPS to failure of super-peers compared to SG-1.

Table 3 Robustness of the SPS to failure of super-peers compared to SG-1

Algorithm	Simulation rounds needed when 10% of super-peers are failed	Simulation rounds needed when 20% of super-peers are failed	Simulation rounds needed when 30% of super-peers are failed	Standard deviation of the simulation rounds
The SPS	17.0	16.0	15.0	1.0
SG-1	22.0	24.0	25.0	1.5

failure of 30% super-peers as shown in Figure 7 (c), the number of client peers without a super-peer (because of super-peer failure) in SPS (around 3/5 of client peers) is larger than that of SG-1 (around 1/5 of client peers). That is, there is a sharp decrease in the number of client peers that have joined a super-peer in our SPS compared to a slight decrease in SG-1. In other words, the super-peer failure in SPS makes worse impact on client peers than that in SG-1. However, the needed simulation rounds for restoring stable state in our SPS (restored at round 10) are fewer than those in SG-1 (restored at round 12). In summary, our SPS takes fewer simulation rounds to restore in the face of larger number client peers without a super-peer (because of super-peer failure) compared to SG-1.

The rationale behind result (1) mentioned above is as follows: when some super-peers are failed at the sixth simulation round, they are removed from the overlay. On one hand, the client peers, whose super-peers have crashed, remove the failed super-peers from their sets of super-peer candidates and rebuild their sets of super-peer candidates by executing the SPS algorithm. Then, these client peers select and join new super-peers. On the other hand, most of the required super-peers can be selected during the first few simulation rounds. Thus, even more super-peers fail; there is only a little variation in the number of the required simulation rounds for the overlay to converge again. Based on the simulation results above, we can conclude that our SPS is robust to failure of super-peers and our SPS is efficient in re-organizing a super-peer overlay.

4. Conclusion and future work

In this paper, we have presented a gossip-based super-peer selection algorithm (SPS) for quickly building a super-peer overlay upon a connected overlay. In the SPS, each peer periodically rebuilds its set of super-peer candidates through gossip communication. Peers with high capacity are promoted to be super-peer candidates. The decision whether a peer takes the role of a super-peer is made based on its set of super-peer candidates. Once the roles of peers are determined, peers join a super-peer, or search and add client peers with a conditional two-hop search method according to their roles.

The conducted simulations show that the proposed SPS is efficient in both selecting super-peers and quick building a super-peer overlay. Furthermore, our SPS achieves a comparable robustness and better performance in convergence time, scalability compared to SG-1 [11].

In the future work, we will reduce the communication overhead for building the set of super-peer candidates, and take account of the stability of a super-peer overlay using a local search method. Moreover, it would also be interesting to apply a greedy approach in selecting super-peers in our SPS algorithm in order to improve the performance of quickly building a super-peer overlay.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

ML had the initial idea of building a super-peer overlay (SPS), carried out simulations, and worked on formulating the paper. EH provided suggestions on the paper finalization and MY is the supervisor. All authors read and approved the final manuscript.

Acknowledgements

This work was supported by the ITEA2 Expeshare project, funded by the Finnish Funding Agency for Technology and Innovation (TEKES), the project of SOPSCC (Pervasive Service Computing: A Solution Based on Web Services), funded by the Academy of Finland, and the DECICOM project, funded by TEKES Ericsson, Nokia, and NetHawk. The authors would like to thank Dr. Vidyasagar Potdar from Curtin University in Australia and Dr. Jiehan Zhou for their valuable comments on improving the quality of the paper.

Received: 5 February 2013 Accepted: 5 February 2013
Published: 27 February 2013

References

1. Baset S, Schulzrinne H (2004) An analysis of the Skype peer-to-peer internet telephony protocol. Technical Report CUCS-039-04. Columbia University, Department of Computer Science, New York
2. Cohen B BitTorrent, http://www.bittorrent.com/btusers/guides/bittorrent-user-manual/chapter-02-basic-guides/basics-bittorrent (last accessed 17-12-2009)
3. Kirk P RFC-Gnutella 0.6. http://rfc-gnutella.sourceforge.net/index.html (last accessed 17-12-2009)
4. PPLive, http://www.pplive.com, (last accessed 17-12-2009)
5. Milojicic DS, Kalogeraki V, Lukose R, Nagaraja K, Pruyne J, Richard B, Rollins S, Xu Z (2002) Peer-to-Peer Computing. Technical Report HPL-2002-57. HP Labs, Palo Alto
6. Oram A (ed) (2001) Peer-to-Peer: Harnessing the Power of Disruptive Technologies. O'Reilly & Associates, Inc, 101 Morris Street Sebastopol, CA 95472, USA
7. Garbacki P, Epema DHJ, Steen M (2010) The design and evaluation of a self-organizing superpeer network. IEEE Trans Comput 59(3):317–331
8. Jesi GP, Montresor A, Babaoglu O (2007) Proximity-aware superpeer overlay topologies. IEEE Trans Network Serv Manag 4(2):74–83
9. Lua K, Crowcroft J, Pias M, Sharma R, Lim S (2005) A survey and comparison of peer-to-peer overlay network schemes. IEEE Comm Surv Tutorials 7(2):72–93
10. Löser A, Naumann F, Siberski W, Nejdl W, Thaden U (2004) Semantic Overlay Clusters within Super-Peer Networks. In: Aberer K et al (eds) VLDB 2003 Ws DBISP2P, LNCS 2944. Springer, Berlin Heidelberg, Germany, pp 33–47

11. Montresor A (2004) A robust protocol for building super-peer overlay topologies. In: Proc. of International Conference on Peer-to-Peer Computing., pp 202–209
12. Yu J, Li M (2008) CBT: a proximity-aware peer clustering system in large scale BitTorrent-like Peer-to-Peer networks. Comput Comm 31(3):591–602
13. Kazaa, URL: http://www.kazaa.com/us/help/glossary/p2p-.htm (last accessed 17-12-2009)
14. Yang B, Garcia-Molina H (2003) Designing a super-peer network. Proc ICDE:49–60
15. Wang F, Liu J, Xiong Y (2008) Stable peers, existence, importance, and application in Peer-To-Peer live video streaming. Proc IEEE INFOCOM:1364–1372
16. Stutzbach D, Rejaie R (2005) Characterizing the two-tier Gnutella topology. Proc ACM SIGMETRICS:402–403
17. Voulgaris S, Gavidia D, Steen M (2005) YCLON, Inexpensive Membership Management for Unstructured P2P Overlays. J Netw Syst Manag 13(2):197–217
18. Jelasity M, Montresor A, Babaoglu O (2009) T-Man: gossip-based fast overlay topology construction. Comput Netw Elsevier 53(13):2321–2339
19. Li X, Zhuang Z, Liu Y (2005) Dynamic layer management in superpeer architectures. IEEE Trans Parallel Distr Syst 16(11):1078–1091
20. Liu M, Zhou J, Koskela T, Ylianttila M (2009) A robust algorithm for the membership management of super-peer overlay. Proc of 12th IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services:132–143
21. Lua EK, Zhou X, Crowcroft J, Mieghem PV (2008) Scalable multicasting with network-aware geometric overlay. J Comput Comm Elsevier 31(3):464–488
22. Sachez-Artigas M, Garcia-Lopez P, Skarmeta AFG (2008) On the Feasibility of Dynamic Super-peer Ratio Maintenance. In: Proc of International Conference on Peer-to-Peer Computing., pp 333–342
23. Nejdl W, Wolpers M, Siberski W, Schmitz C, Schlosser M, Brunkhorst I, Löser A (2004) Super-peer-based routing strategies for RDF-based peer-to-peer networks. J Web Semant: Sci Serv Agents World Wide Web 1(2):177–186
24. Ganesh AJ, Kermarrec AM, Massoulie L (2003) Peer-to-peer membership management for gossip-based protocols. IEEE Trans Comput 52(2):139–149
25. Min S, Holliday J, Cho DS (2006) Optimal Super-peer Selection for Large-scale P2P System. Proc Hybrid Inform Tech:588–593
26. Jelasity M, Kowalczyk W, Van Steen M (2003) Newscast Computing. Technical Report, IRCS-006. Dept. of Computer Science, Vrije Universiteit, Amsterdam
27. Yuan Q, Wu J (2008) DRIP: A Dynamic VoRonoi Reglons-Based Publish/ Subscribe Protocol in Mobile Networks. In: proc. IEEE INFOCOM., pp 2110–2118
28. Jelasity M, Montresor A, Jesi GP, Voulgaris S (2009) PeerSim: P2P Simulator., http://peersim.sourceforge.net/
29. Laoutaris N, Smaragdakis G, Oikonomou K, Stavrakakis I, Bestavros (2007) A distributed placement of service facilities in large-scale networks. Proc IEEE INFOCOM:2144–2152

doi:10.1186/1869-0238-4-4
Cite this article as: Liu *et al.*: An efficient selection algorithm for building a super-peer overlay. *Journal of Internet Services and Applications* 2013 **4**:4.