Proximity-aware Superpeer Overlay Topologies*

Gian Paolo Jesi[1], Alberto Montresor[2], and Ozalp Babaoglu[1]

[1] Dept. of Computer Science, University of Bologna (Italy)
E-mail: {gjesi,babaoglu}@CS.UniBO.IT
[2] Dept. of Information and Communication Technology, University of Trento (Italy)
E-mail: alberto.montresor@dit.unitn.it

Abstract. The concept of *superpeer* has been introduced to improve the performance of popular P2P applications. A *superpeer* is a "powerful" node that acts as a server for a set of clients, and as an equal with respect to other superpeers. By exploiting heterogeneity, the superpeer paradigm can lead to improved efficiency, without compromising the decentralized nature of P2P networks. The main issues in the construction of superpeer-based overlays are the selection of superpeers, and the association between superpeers and clients. Generally, superpeers are either run voluntarily (without an explicit selection process), or chosen among the "best" nodes in the network, i.e. those equipped with the largest amount of resources, such as bandwidth or storage. In several contexts, however, shared resources are not the only factors; latency between clients and superpeers may play an important role, for example in *online games*. This paper presents SG-2, a novel protocol for building and maintaining a proximity-aware superpeer topology. SG-2 uses a gossip-based protocol to spread messages to nearby nodes and a biologically-inspired task allocation mechanism to promote the "best" nodes to the superpeer status. The paper includes extensive simulation experiments to prove the efficiency, scalability and robustness of SG-2.

1 Introduction

Modern P2P networks present several unique aspects that distinguish them from traditional distributed systems. Networks comprising hundreds of thousand of peers are not uncommon. A consequence of such scale is extreme dynamism, with a continuous flow of nodes joining or leaving. Such characteristics present several challenges to the developer. Neither a central authority nor a fixed communication topology can be employed to control the various components. Instead, a dynamically changing overlay topology is maintained and control is completely decentralized. The topology is defined by "cooperation" links among nodes, that are created and deleted based on the requirements of the particular application.

The choice of a particular topology is a crucial aspect of P2P design. Until recently, most deployed P2P applications were characterized by the absence of a specific mechanism for enforcing a given topology; for example, Gnutella nodes

* This work was partially supported by the FET unit of the European Commission through projects BISON(IST-38923), DELIS (IST-01907) and CASCADAS (IST-27807).

were free to accept/refuse connections at will [12]. The consequence of this choice was the adoption of inefficient communication schemes, such as flooding.

A distinct, but related problem regards roles that nodes may assume: original P2P systems were based on a complete "democracy" among nodes: "everyone is a peer". But physical hosts running P2P software are usually very heterogeneous in terms of computing, storage and communication resources, ranging from high-end servers to low-end desktop machines.

The superpeer paradigm is an answer to both issues [9, 12]. It is based on a two-level hierarchy: *superpeers* are nodes faster and/or more reliable than "normal" nodes and take on server-like responsibilities and provide services to a set of *clients*. For example, in the case of file sharing, a superpeer builds an index of the files shared by its clients and participates in the search protocol on their behalf. Clients are leveraged from taking part in costly protocols and the overall traffic is reduced by forwarding queries only among superpeers. Superpeers allow decentralized networks to run more efficiently by exploiting heterogeneity and distributing load to machines that can handle the burden. On the other hand, this architecture does not inherit the flaws of the client-server model, as it allows multiple, separate points of failure, increasing the health of the P2P network.

The superpeer paradigm is not limited to file sharing: it can be seen as a general approach for P2P networking. Yet, the structural details are strongly application-dependent, so we cannot identify a "standard" superpeer topology. Parameters to be considered include: how superpeers are linked together; how to arrange clients; how many superpeers are needed; etc.

In this paper, we focus our investigation on a specific aspect of the problem: *proximity*. Our goal is to build a topology where clients and superpeers are associated based on their distance (in terms of communication latency). The idea is to select superpeers among the most powerful nodes, and to associate them with clients whose round-trip time is bounded by a specified constant. This is a generic problem, whose solution can be beneficial to several P2P applications. Examples include P2P telephony networks such as Skype [2], streaming applications such as PeerCast [20], and online games such as Age of Empires [3]. In all these cases, communication latency is one of the main concerns.

Our solution, called SG-2, is a self-organizing, decentralized protocol capable to build and maintain a superpeer-based, proximity-aware overlay topology. SG-2 uses an epidemic protocol to spread messages to nearby nodes, and implements a task allocation protocol that mimics the behavior of social insects. These biological-inspired mechanisms are combined to promote the "best" nodes to the superpeer status, and to associate them to nearby clients.

To validate the results of our protocol, we considered a specific test case: *online games*. In these applications, a large number of players interact together (or against each other) in virtual worlds. Most online games follow a client-server model, where the only function of the client software is to present a graphic user interface to the player, while the state of the simulated persistent world is hosted on the server side. This approach is scalable only thanks to the deployment of high-end clusters of replicated servers. Just a few games have attempted a different approach. MiMaze [11] and Age of Empires [3] are completely decentralized, and the game state is a shared distributed object maintained by all participants. In this case, consistency requirements limit the number of players that may be involved in the same game.

We believe that the superpeer paradigm could represent an interesting alternative to the two approaches above. We envision a system where a small number of powerful nodes act as state servers when needed, with the remaining ones acting as clients. All nodes run the same code and can switch from the first role to the second when needed. Thus, superpeers dynamically change over time, depending on the environment conditions.

2 System Model

We consider a network consisting of a large collection of *nodes*. The network is highly dynamic; new nodes may join at any time, and existing nodes may leave, either voluntarily or by *crashing*. Since voluntary leaves may be simply managed through "logout" protocols, in the following we consider only node crashes. Byzantine failures, with nodes behaving arbitrarily, are excluded from the present discussion. We assume nodes are connected through an existing routed network, such as the Internet, where every node can potentially communicate with every other node. To actually communicate with another node, however, a node must know its *identifier*, e.g. a pair ⟨IP address, port⟩.

The nodes known to a node are called its *neighbors*, and as a set are called its *view*. Together, the views of all nodes define the topology of the overlay network. Given the large scale and the dynamism of our envisioned system, views are typically limited to small subsets of the entire network. Views can change dynamically, and so the overlay topology.

Nodes are heterogenous: they differ in their computational and storage capabilities, and also (and more importantly) with respect to the bandwidth of their network connection. To discriminate between nodes that may act as superpeers and nodes that must be relegated to the role of clients, each node v is associated with a *capacity* value $cap(v)$, that represents the number of clients that can be handled by v. To simplify our simulations, we assume that each node knows its capacity. In reality, this parameter is strongly dependent on the specific application, and can be easily computed on-the-fly through on-line measurements.

Besides capacity associated to each single node ("how many"), another parameter to be considered is the end-to-end latency between nodes ("how well"). In our model, each pair of nodes (v, w) is associated with a *latency distance* $lat(v, w)$, representing the average round-trip time (RTT) experienced by communications between them. The latency distance between a specific pair of nodes may be measured directly and precisely through ping messages, or approximately estimated through a *virtual coordinate service* [7]; given the dynamic nature of our system and the large number of nodes to be evaluated as potential neighbors, we will adopt the latter approach.

3 The Problem

Generally speaking, our goal is to create a topology where the most powerful nodes (in terms of capacity) are promoted to the role of superpeers, and the association clients/superpeers is such that each client obtains a configurable *quality of service* (in terms of latency distance) from its superpeer.

More formally, we define the problem of building a proximity-aware, superpeer-based topology as follows. At any given time, the problem input is given by the current set of nodes $\mathcal{V}$, and the functions $cap()$ and $lat()$ defined over it. Furthermore, a global parameter tol expresses the maximum latency distance that can be tolerated between clients and superpeers. The constraints describing our target topology are the following:

- each node is either a superpeer or a client;
- each client c is associated to exactly one superpeer s (we write $super(c) = s$);
- the number of clients associated to superpeer s does not exceed $cap(s)$;
- given a superpeer s and one of its clients c, we require that $lat(s, c) \leq tol$.

To avoid to end up with a set of disconnected, star-shaped components rooted at each superpeer, we require that superpeers form another proximity-based overlay: two superpeers are connected if their latency distance is smaller than $tol + \delta$, where δ is another configuration parameter.

We aim at selecting as few superpeers as possible (otherwise, the problem could be trivially solved by each node acting as a superpeer, with no client/superpeer connections). This choice is motivated, once again, by the particular scenario we are considering: in online games, superpeers manage the distributed simulation state, so centralizing as many decisions as possible is important from the performance point of view. Note that given the dynamism of our environment, obtaining the minimum number of superpeers may be difficult, or even impossible. But even in a steady state, the resulting optimization problem is NP-complete.

4 The SG-2 Protocol

The architecture of SG-2 is shown in Figure 1; here, we briefly describe the rationale behind it, leaving implementation details to the following subsections.

Our solution to the problem described above is based on a fundamental observation: measuring precisely the RTT between all pairs of nodes (e.g., through pings) is extremely slow and costly, or even impossible due to topology dynamism. To circumvent this problem, and allow nodes to estimate their latency without direct communication, the concept of *virtual coordinate service* has been developed [7]. The aim of this service is to associate every node with a synthetic coordinate in a virtual, n-dimensional space. The Euclidean distance between the coordinates of two nodes can be used to predict, with good accuracy, the RTT between them; in other words, it is sufficient for two nodes to learn about their coordinates to estimate their latency, without direct measurements.

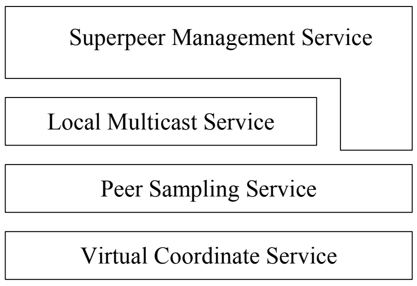


Fig. 1. The set of services composing the SG-2 architecture.

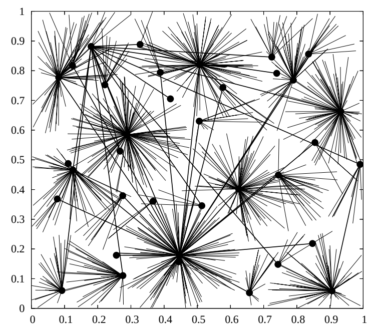


Fig. 2. A superpeer topology in a bi-dimensional virtual space, where Euclidean distance corresponds to latency.

Our problem may be redefined based on the concept of virtual coordinates. Nodes are represented by points in the virtual space; each of them is associated with an *influence zone*, described as a n-dimensional sphere of radius *tol* centered at the node. Our goal is to cover the virtual space with a small number of superpeers, in such a way that all nodes are either superpeers or are included in the influence zone of a superpeer. Figure 2 shows the topology resulting from the execution of SG-2 in a bi-dimensional virtual space.

Nodes communicate with each other using a *local broadcast service*, whose task is to efficiently disseminate messages to nodes included in the influence zone of the sender. This service is used by powerful nodes to advertise their availability to serve as superpeers, and by ordinary nodes to seek superpeers whose capacity has not been saturated yet.

The main component of SG-2 is the *superpeer management service*, which selects the superpeers and associates clients to them. The protocol is heavily inspired by the behavior of social insects [4], such as ants or bees, that have developed very sophisticated mechanisms for labor division. In summary, these mechanisms work as follows. In a totally decentralized fashion, specialized groups of individuals emerge, with each group aimed at performing some particular task. The task allocation process is dynamic and follows the community needs according to changes in the environment. The stimulus to perform some kind of task or to switch to another one can be given by many factors, but it is normally given by high concentrations of chemical signals, such as pheromones, that are released by other individuals and are spread in the environment. Each individual has its own response threshold to the stimulus and reacts accordingly.

The superpeer protocol mimics this general picture. Un-associated nodes diffuse a “request for superpeers” signal through local broadcasts; the signal concentration in the network may stochastically trigger a switch to the superpeer role in some nodes according to their response threshold, which is proportional to their capacity. On the other hand, powerful nodes covering the same area of the virtual space compete with each other to gain news clients, by signaling their availability through local broadcasts. Clients associate themselves to the most powerful superpeers, and superpeers with an empty client set switch back to the client role. The combination of these two trends (the creation of new superpeers to satisfy client request, and the removal of unnecessary superpeers) finds its equilibrium in a topology that approximates out target topology.

The last component to be described is the peer sampling service. The task of this layer is to provide each node with a view containing a random sample of nodes [13]. The motivation is twofold: first of all, the random sample is used by the local broadcast service to perform gossiping; second, the topology resulting from this layer can be described as a random graph composed of a single connected component among all nodes. This topology is extremely robust and present no central point of failure; it may be used to recover from catastrophic failures in the overlaying superpeer topology, for example due to a coordinated attack to the subset of superpeers.

4.1 Virtual Coordinate Service

In SG-2, the virtual coordinate service is provided by Vivaldi [7], which is a decentralized, scalable, and efficient protocol developed at MIT. Using Vivaldi, nodes may obtain good coordinates with few RTT probes directed to a small subset of nodes. More importantly, Vivaldi can exploit normal traffic produced by applications using it, without requiring further communication.

The *estimate* of the latency distance between v_i and v_j is denoted $est(v_i, v_j)$. Being estimates, these values may differ from the actual latency. The pairwise error between the estimate and the actual latency can be computed as:

$$\frac{|\, lat(v_i, v_j) - est(v_i, v_j)|}{\min\{est(v_i, v_j), lat(v_i, v_j)\}}$$

In our experiments, the number of dimensions of the virtual space is 5; measuring the error between all pairs of nodes, we found a median error of only 0.14, and a maximum error of 3.5. Note that latency distances that are "under-estimated" may pose a problem: if the actual latency is over *tol*, but the estimated latency is smaller, a superpeer may accept a client out of the tolerated range. For this reason, the maximum error must be considered when selecting parameter *tol*.

4.2 Peer Sampling Service

The sampling service is provided by Newscast [14], which has proven to be a valuable building block to implement several P2P protocols [15]. We provide here a brief description of the protocol and its characteristics.

Each Newscast node maintains a view containing c node descriptors, each of them composed of a remote node identifier and a logical time-stamp. Newscast is based on the gossip paradigm: periodically, each node (i) selects a random peer from its partial view; (ii) updates its local descriptor; and (iii) performs a *view exchange* with the selected peer, during which the two nodes send each other their views, merge them, and keep the c freshest descriptors.

This exchange mechanism has three effects: views are continuously shuffled, creating a topology that is close to a random graph with out-degree c; the resulting topology is strongly connected (according to experimental results, choosing $c = 20$ is already sufficient for very stable and robust connectivity); and finally, the overlay topology is self-repairing, since crashed nodes cannot inject new descriptors any more, so their information quickly disappears from the system.

The peer sampling service is a key component both during the initialization phase (*bootstrap*) of the other layers, and during the normal functioning of the protocol, when it allows the discovery of "distant" or newly joined peers from the entire network. Newscast is extremely inexpensive: messages are small, and the periodicity of view exchanges may be as low as one message per minute [14].

4.3 Local Multicast Service

Unlike previous layers, based on existing protocols, the local multicast service has adapted an existing protocol for the specific needs of SG-2 [8]. Each message m is associated with the sender identifier s_m and a radius parameter r_m. Message m is delivered only to those nodes that are within latency distance r_m from s_m, as estimated by Vivaldi. Hence, the name Spherecast.

The protocol may be described as follows. When a node either receives a message or wants to multicast a new one, it forwards it to its local *fan-out*. The fan-out of node v for message m is given by the subset of neighbors known to v that are potentially interested in the message, i.e. whose distance from s_m is not larger than r_m. Spherecast does not maintain its own topology; instead, it relies on the underlying overlay network provided by the peer sampling service.

When a message is originated locally, or it is received for the first time, it is forwarded immediately to all nodes in the fan-out. If a message has been already received, a node may stochastically decide to drop it (i.e., not forwarding it). This approach is used to avoid flooding the network. A strict deterministic approach such as dropping any multiple copy, would not work fine due to the nature of the underlying overlay. The actual clustering coefficient of the underlying topology and the continous rewiring process may stop the message spreading. The stochastic approach solves this issue in a straightforward manner.

The probability of dropping a message is given by the following formula: $p = 1 - e^{-s/\vartheta}$, where s is the number of times the node has seen this message and ϑ is a response threshold parameter. In this way, when a packet is received multiple times by a peer, it has less and less probability to be forwarded again. From an implementation point of view, digests of received messages are stored in a per-node table, together with the number of times that specific message has been received. This table is managed with a LRU policy, to avoid unbounded growth.

4.4 Superpeer Management Service

This layer is the core component of SG-2. Nodes participate in this protocol either as superpeers or as clients; a client c may be either associated to a superpeer

($\mathit{super}(c) = s$), or actively seeking a superpeer in its *tol* range ($\mathit{super}(c) = \perp$). At the beginning, all nodes start as clients; to converge to the target topology defined in Section 3, nodes may switch role at will, or change their client-superpeer relationship. The decision process is completely decentralized.

Each node v maintains the following local variables. *role* specifies the role currently adopted by v; $\mathit{role} = \textsc{sp}$ if v is a superpeer, $\mathit{role} = \textsc{cl}$ otherwise. *clv* and *spv* are two views, respectively containing the clients and the superpeers known to v. They are composed of node descriptors combining an identifier w and a logical time-stamp ts_w; the latter is used to purge obsolete identifiers, as in Newscast. When v acts as a superpeer, *clv* is populated with the clients currently associated to v; it is empty otherwise. The size of *clv* is limited by $\mathit{cap}(v)$. *spv* contains descriptors for the superpeers that are in $\mathit{tol} + \delta$ range; its size is not explicitly limited, but rather is bounded by the limited number of superpeers that can be found within $\mathit{tol} + \delta$ distance. When v acts as a client, one of the descriptors in *spv* may be the associated superpeer of v.

Two distinct kinds of messages are broadcasted using Spherecast: cl-bcast and sp-bcast. The former are sent while in client state and are characterized by a radius parameter r_m equal to *tol*, i.e. the maximum tolerated latency. The latter are used in superpeer state and their radius parameter is equal to $\mathit{tol} + \delta$; superpeers need a wider radius to get a chance to contact a larger number of superpeers; furthermore, nodes with overlapping influence zones can exchange clients if they find a better client allocation that reduces their latency.

At each node, two threads are executed, one active and one passive. The execution of active threads may be subdivided in periodic *cycles*: in each cycle, superpeers emit a sp-bcast signal which is broadcast in the surrounding area, to notify nodes about their presence and its residual capacity. Clients, on the other hand, periodically emit cl-bcast messages if and only if they are not associated to any superpeer. The shorter the cycle duration, the faster the system converge to the target topology; but clearly, the overhead grows proportionally. The passive threads react to incoming messages according to the message type and the current role. Four distinct cases are possible:

Superpeer v gets $\langle \textsc{sp-bcast}, s, \mathit{ts}_s, \mathit{cap}(s) \rangle$: the pair (s, ts_s) is inserted in *spv*. If s was already present, its time-stamp is updated. After that, the capacity of the two supernodes is compared: if $\mathit{cap}(v) > \mathit{cap}(s)$, then a migration process is started. Clients associated with s that are inside the influence zone of v migrate to v, until the capacity is exhausted. Each affected client is notified about the new superpeer (v) by the current superpeer s. Node s, if left with no clients, switches back to the client role; it associates itself to v, if $\mathit{est}(v, s) \leq \mathit{tol}$ and v has still residual capacity; otherwise, it starts emitting cl-bcast messages.

Superpeer v gets $\langle \textsc{cl-bcast}, c, \mathit{ts}_c \rangle$: if $|\,\mathit{clv}(v)| < \mathit{cap}(v)$ (the capacity of v has not been exhausted), the client node is associated to v (unless, given the asynchrony of messages, it has been already associated with another superpeer).

Client v **gets** $\langle$SP-BCAST$, s, ts_s, cap(s)\rangle$: the pair (s, ts_s) is inserted in *spv*. If s was already present, its time-stamp is updated. If v is not client of any superpeer, it sends a request to s asking to be associated with it. The response may be negative, if s has exhausted its capacity in the period between the sending of the message and its receipt by v. On the other hand, if v is already client of another superpeer s' and $cap(s) > cap(s')$, then it tries to migrate to the more powerful superpeer. This strategy promotes the emergence of a small set of high-capacity superpeers.

Client c **gets** $\langle$CL-BCAST$, c, ts_c\rangle$: This kind of messages can trigger a role change from client to superpeer; it is the cornerstone of our approach. The willingness of becoming a superpeer is a function of a node threshold parameter and the signal concentration perceived by a node in its influence area. The switching probability can be modeled by the following function:

$$P(role(v) = \text{CL} \rightarrow \text{SP}) = \frac{s^2}{s^2 + \theta_v^2}$$

where s is the signal magnitude and θ_v is the response threshold of node v. This function is such that the probability of performing a switch is close to 1, if $s \gg \theta$, and it is close to 0 if $s \ll \theta$. If c_{max} is the maximum capacity, θ_v is initialized with a value which is $c_{max} - cap(v)$; in this way, nodes with higher capacity have a larger probability of becoming superpeers. The maximum capacity may be either known, or it can be easily computed by an *aggregation* protocol in a robust and distributed fashion [15].

After the initialization, in order to make the topology more stable and avoid fluctuations, the response threshold is modified in such a way that time reinforces the peer role: the more time spent as a client, the less probable is to change role. Once again, the inspiration for this approach comes from biology: it has been observed, for example, that the time spent by an individual insect on a particular task produces important changes in some brain areas. Due to these changes, the probability of a task change (e.g., from foraging to nursing) is a decreasing function of the time spent on the current task [4]. For this reason, θ_v is reinforced as follows:

$$\theta_v(t) = \theta_v(t-1) + (\alpha * (t - t'_v))$$

Where t is the current cycle and t'_v is the last cycle in which v became a superpeer; α is a parameter to limit or increase the time influence. The peer normal responsiveness is re-initialized based on its local capacity if its superpeer crashes or if it becomes a superpeer node.

The reaction to CL-BCAST messages is the only mechanism to allow a client to become a superpeer. A superpeer can switch back to the client role only when other higher capacity superpeers have drained its client set. The θ adaptation process is only active when a node is in the client state.

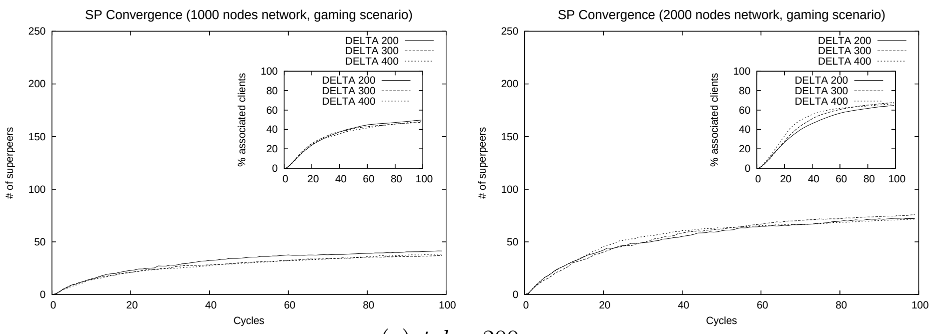
5 Experimental results

We performed a large number of experiments based on simulation to validate the effectiveness of our approach. The goal of our experiments was twofold: first of all, we measured the speed of convergence in a stable overlay, in the absence of failures; second, we measured the robustness of our approach in a dynamic environment, where a fixed percentage of nodes is substituted with fresh ones periodically. Any node in the network can be affected by substitution, regardless of its role. Unlike the real world, where a superpeer is supposed to be more reliable, our choice is stricter and more "catastrophic". Finally, communication overhead has been measured. The experiments have been performed using Peersim [21].

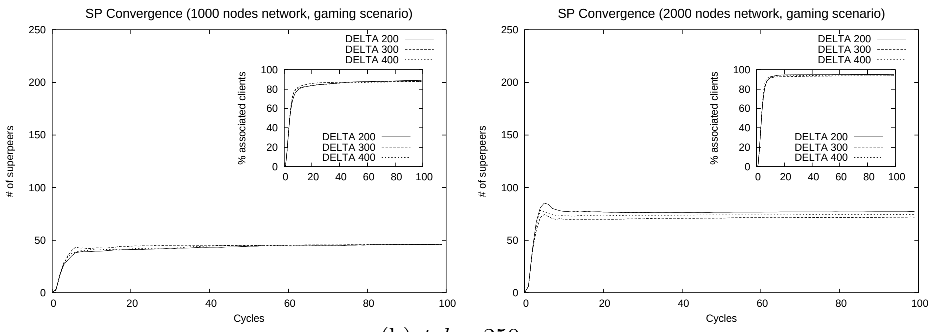
In our experiments, network size is fixed at 1000 and 2000 nodes. Several kinds of networks have been considered, but here, due to space restrictions, the focus is on *gaming-oriented* scenario [23, 28]. Other scenarios present similar results. For each pair of nodes v, w, the latency distance $lat(v, w)$ among them has been generated using a normal distribution with average value $\mu = 250\,ms$ and variance $\sigma = 0.1$ [28]. Then, we have run VIVALDI on this network, obtaining the corresponding function $est(v, w)$. In the corresponding virtual space, we have considered *tol* values of 200 ms, 250 ms and 300 ms, which are typical of strategy and role-playing games. We have experimented with δ values of 200 ms, 300 ms and 400 ms, corresponding to typical round-trip time that can be accepted for superpeer communication. The capacity function $cap()$, i.e. the maximum number of clients that can be served, is generated through an uniform distribution in the range $[1 : 500]$. The time during the simulations is measured in *cycles*; we define a cycle to be the time period in which each node in the network has performed a gossip exchange. All the results are averaged over 10 experiments.

Figure 3 illustrates the behavior of the protocol over time. All the figures in the left column are obtained in networks whose size is 1000 nodes, while the figures in the right column are relative to networks with size equal to 2000 nodes. The content of each sub-figure is divided in two parts; in the main plot, the number of superpeer active at each cycle is shown; in the small frame inside the main plot, the percentage of clients that are already associated is shown. In these experiments, the network is static; no nodes are removed or added.

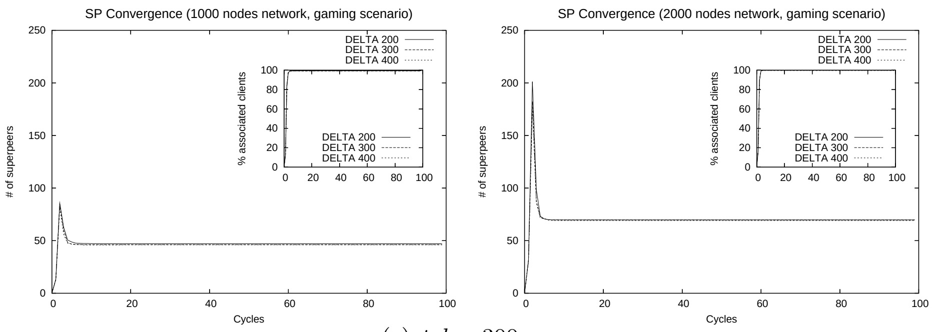
Figure 3(a) depicts a rather bad situation: in both network sizes, the convergence is extremely slow, and the number of nodes that are satisfied is low. This bad performance is motivated by the characteristics of the latency distributions [23, 28] and the tolerance value selected; most of the node pairs have a higher latency than 200 ms, and thus SG-2 cannot help much. Figure 3(b) shows a much better situation: a large percentage of clients (between 94% and 100% depending on size and parameter δ) have been associated after only few cycles (20-30). The number of superpeers is also very small, after an initial peak due to a large number of clients reacting to the signal. Almost every client can reach the required latency because 250 ms is the average pairwise latency in our game-like coordinates distribution. However, some nodes lies outside the 250 ms border and it is challenging for SG-2 to accommodate those nodes. The node density plays an important role for SG-2. In fact, the bigger network can be fully orga-



(a) $tol = 200\,ms$



(b) $tol = 250\,ms$



(c) $tol = 300\,ms$

Fig. 3. Convergence time. Three *tol* values are considered: $200\,ms$ (a), $250\,ms$ (b), $300\,ms$ (c). The main figures show the number of active superpeer at each cycle, while the small sub-figures show the number of clients that are in *tol* range. Three different δ values are shown in each figure.

nized in a latency-aware fashion using the wider superpeer communication range ($\delta = 400\,ms$). Figure 3(c) shows the performance for $tol = 300\,ms$: a response time that is perfectly acceptable in a strategic/role playing game scenario. The

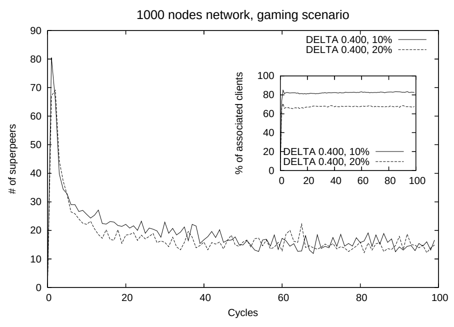


Fig. 4. Experiments with churn. Network size is 1000; at each cycle, 10% or 20% of the nodes are substituted with new ones.

latency aware topology in the figure is very good with any δ value. We obtain 100% of in range clients with about 50 superpeers in the small network and about 63 in the bigger network, in less than 10 cycles.

Figure 4 is aimed at illustrating the robustness of our protocol. The size of the network is fixed at 1000 nodes. Its composition, however, is dynamic: at each cycle, 10% or 20% of the nodes crashes and are substituted with new ones. The figure shows that the number of superpeers oscillates over time, as expected, and that up to 80% and 70% of the clients are associated to superpeers. The nodes that are not associated are those that have been recently created and are trying to find a position in the topology.

Finally, we discuss message overhead; due to space limitations, we provide summary data instead of plots. We have measured the number of broadcast messages, including both CL-BCAST and SP-BCAST. Since the former type of message is broadcast only in case of lack of satisfaction, only a small number of them are generated: on average, less than 2 messages every thousand nodes. Superpeers, on the other hand, send one message per cycle.

6 Related work

The superpeer approach to organize a P2P overlay is a trade-off solution that merges the client-server model relative simplicity and the P2P autonomy and resilience to crashes. The need for a superpeer network is mainly motivated by the fact to overcome the heterogeneity of peers deployed on the Internet.

Yang and Garcia Molina [27] proposed some design guidelines. A mechanism to split node clusters is proposed and evaluated analytically, but no experimental results are presented.

Superpeer solutions proved to be effective solutions in the real world: Kazaa / Fasttrack [9] and Skype [25] are two outstanding examples. However, their actual protocols are not publicly available and they cannot be compared with any other solution or idea. At the time of writing, only a few works [2, 17] describe some low-level networking details.

The SG-2 protocol can be considered as a natural evolution of the SG-1 [18] protocol, however the two solutions cannot be directly compared from a performance point of view because their goals are pretty different. SG-1 focuses on optimizing the available bandwidth in the system, while SG-2 introduces the notion of latency between peer pairs and poses a QOS limit on it. The definition of the target topology is straightforward in SG-1 (e.g., the minimum number of superpeers to accommodate all the peers according to the superpeer capacities), while it is a NP-problem in the SG-2 case. From the architectural point of view, they both rely on the existence of an underlying random overlay. The superpeer overlay is generated on top of it. The superpeer election process in SG-2 is strongly bio-inspired and much more randomized than approach used in SG-1.

In [24], the authors propose a socio-economic inspiration based on T.Shelling's model to create a variation of the super-peer topology. Such variation allows the ordinary peers to be connects with each other and to be connected to more than one super peer at the same time. This topology focuses on efficient search. As in our case, the superpeers are connected to each other to form a network of hubs and both solutions are suited for unstructured networks. However, they do not address the problem of the superpeer election.

The basic problem of finding the best peer, having the required characteristics, to accomplish some task (e.i., the superpeer task) is addressed in a more general form in [1]. The problem is referred as "optimal peer selection" in P2P downloading and streaming scenarios. The authors use an economics inspired method to solve the optimization problem; the developed methodologies are general and applicable to a variety of P2P resource economy problems. The proposed solution is analytically strong, but no experimental results are shown especially regarding a large and dynamic scenario as the one the authors are addressing.

Our Virtual Coordinate Service (see section 4.1) is based on VIVALDI but it is not tied to any particular implementation. Other architectures can be adopted, such as IDMaps [10] and GNP [19] or PIC [6] and PCoord [16]. The former pair rely on deployment of infrastructures nodes, while the latter pair provide latency estimates gathered only between end-hosts, as VIVALDI does. We opted for VIVALDI because of its fully distributed nature and simple implementation.

In less strict latency context, the hop-count is usually preferred in contrast to the millisecond latency to provide distance estimation. Pastry [5,22], for example, uses a hop distance metric to optimize its time response.

7 Conclusions

This paper presented SG-2, a fully decentralized, self-organizing protocol for the construction of proximity-aware, superpeer-based overlay topologies. The protocol produces an overlay in which almost all nodes (99.5%) are in range with a *tol* latency of 300 *ms*. The number of generated superpeers is small with respect to the network size (only 3-5%). The protocol shows also an acceptable robustness to churn. We believe that these results can be profitably adopted to implement several classes of applications, including strategy and role-playing games. Other classes of games, such as first-person shooter, are probably not suitable given their extremely strict latency requirements (inferior to 100 ms). These results are an improvement over existing decentralized games [3], that are based on strong replication [26] or low-level facilities such as IP-multicast [11]).

We conclude noting that the results presented in this paper are only a first step toward the implementation of P2P games; several other problems have to be solved, including security, state replication, state distribution, etc.

References

1. M. Adler, R. Kumar, K. W. Ross, D. Rubenstein, T. Suel, and D. D. Yao. Optimal peer selection for p2p downloading and streaming. In *Proceedings of IEEE Infocom*, Miami, FL, March 2005.
2. S. Baset and H. Schulzrinne. An analysis of the skype peer-to-peer internet telephony protocol. Technical Report CUCS-039-04, Columbia University, Department of Computer Science, New York, NY, Sept. 20004.
3. P. Bettner and M. Terrano. 1500 Archers on a 28.8: Network Programming in Age Of Empires and Beyond. In *Proc. of the GDC'01*, Mar. 2001.
4. E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence: from natural to artificial systems*. Oxford University Press, Inc., New York, NY, USA, 1999.
5. M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron. Exploiting network proximity in distributed hash tables. In O. Babaoglu, K. Birman, and K. Marzullo, editors, *International Workshop on Future Directions in Distributed Computing (FuDiCo)*, pages 52–55, June 2002.
6. M. Costa, M.Castro, A.Rowstron, and P.Key. Pic: Practical internet coordinates for distance estimation. In *Proc. of ICDCS'04*, 2004.
7. F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *Proc. of the SIGCOMM '04*, Portland, Oregon, August 2004.
8. P. Eugster, R. Guerraoui, S. B. Handurukande, A.-M. Kermarrec, and L. Massoulié. Lightweight probabilistic broadcast. *ACM Transactions on Computer Systems*, 21(4):341–374, 2003.
9. Fasttrack Home Page. http://www.fasttrack.nu.
10. P. Francis, S. Jamin, C. Jin, Y. Jin, V. Paxson, D. Raz, Y. Shavitt, and L. Zhang. IDMaps: a global internet host distance estimation service. In *Proc. of IEEE Infocom '99*, 1999.
11. L. Gautier and C. Diot. MiMaze, a Multiuser Game on the Internet. `http://citeseer.ist.psu.edu/gautier97mimaze.html`, 1997.
12. Gnutella web site. http://gnutella.wego.com.
13. M. Jelasity, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. The peer sampling service: Experimental evaluation of unstructured gossip-based implementations. In *Proc. of the 5th Int. Middleware Conf.*, Toronto, Canada, Oct. 2004.
14. M. Jelasity, W. Kowalczyk, and M. van Steen. Newscast computing. Technical Report IR-CS-006, Vrije Universiteit Amsterdam, Department of Computer Science, Amsterdam, The Netherlands, November 2003.
15. M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.*, 23(1):219–252, 2005.

16. L. Lehman and S. Lerman. Pcoord: network position estimation using peer-to-peer measurements. In *Proc. of the 3rd IEEE International Symposium on Network Computing and Applications (NCA'04*, 2004.
17. N. Leibowitz, M. Ripeanu, and A. Wierzbicki. Deconstructing the kazaa network, 2003.
18. A. Montresor. A Robust Protocol for Building Superpeer Overlay Topologies. In *Proc. of the 4th Int. Conf. on Peer-to-Peer Computing*, Zurich, Switzerland, August 2004. IEEE. To appear.
19. T. Ng and H. Zhang. Predicting internet network distance with coordinates-based approaches. In *Proc. of IEEE Infocom*, 2002.
20. Peercast P2P Radio. `http://www.peercast.org`.
21. Peersim Peer-to-Peer Simulator. `http://peersim.sf.net`.
22. A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, Nov. 2001.
23. N. Sheldon, E. Girard, S. Borg, M. Claypool, and E. Agu. The effect of latency on user performance in Warcraft 3. In *Proc. of the 2nd Workshop on Network and System Support for Games*, pages 3–14, New York, NY, USA, 2003. ACM Press.
24. A. Singh and M. Haahr. Creating an adaptive network of hubs using schelling's model. *Commun. ACM*, 49(3):69–73, 2006.
25. Skype: Free internet telephony that just works. http://www.skype.com.
26. Unreal networking protocol notes by tim sweeney. http://unreal.epicgames.com/Network.htm.
27. B. Yang and H. Garcia-Molina. Designing a super-peer network. In *IEEE International Conference on Data Engineering, 2003*, 2003. `http://dbpubs.stanford.edu/pub/showDoc.Fulltext?lang=en&doc=2003-33&for%mat=pdf&compression=`.
28. S. Zanikolas and R. Sakellariou. Towards a monitoring framework for worldwide grid information services. In *Euro-Par*, pages 417–422, 2004.