# Unstructured Peer-to-Peer Session over IP using SIP

Khashayar Khavari, Chuen Liang, Ali Tizghadam, Farid Fadaie, Nadeem Abji, Ramy Farha, Alberto Leon-Garcia
Electrical and Computer Engineering Department
University of Toronto
Email: {khashayar.khavari, chuen.liang, ali.tizghadam, farid.fadaie, nadeem.abji, ramy.farha, alberto.leongarcia}@utoronto.ca

***Abstract*— Data and telephone service providers have started considering migration to an IP based environment. This paper presents an unstructured peer-to-peer approach to initiating and maintaining sessions over IP using session initiation protocol (SIP). The offered solution is completely modular and is compatible with traditional server-client based approaches using SIP. The peer-to-peer nature of our design allows it to be highly scalable and self managing with low maintenance costs, making it an attractive solution for service providers.**

## I. Introduction

SIP is a promising signaling protocol for the next generation network control plane. Current SIP implementations are typically based on the client-server architecture which has problems, such as limited scalability and the existence of a single point of failure. To address such failures redundant servers can be deployed at additional capital cost as well as increased system maintenance. In this paper, we propose a solution to these problems based on a peer-to-peer (P2P) architecture. Because of their self organized and self managed nature, P2P networks are low maintenance, highly efficient, cost effective and rather reliable. This makes P2P a good candidate for the underlying layer for signaling of IP-based applications. Our main objective for the P2P SIP system is that it be highly reliable and scalable. It has to achieve low cost by being self-organized and self-managed. Security is also an issue which has to be resolved but is not considered in the work presented here. In addition, the system must be compatible with the traditional client-server systems. Finally we expect the system to be robust, which means small changes in the system or the environment should not affect the system's performance.

The paper is organized as follows. In section II, we provide an overview of SIP and P2P networks as well as state of the art. Section III presents the details of the implementation as well as the rationale for the underlying design decisions. Section IV introduces a notion of stability that is used in this paper. In Section V, we present the results of the tests performed to support the validity of the approach. Finally Section VI provides a summary of the work as well as a discussion of future work.

## II. Background and related work

### A. SIP and current implementations

SIP is an Internet Engineering Task Force (IETF) protocol for initiating interactive user sessions that involve multimedia elements such as video, voice, chat, gaming, and virtual reality. SIP is a flexible and extensible protocol that can resolve traditional problems of mobility, presence and availability.

The IETF defines standard behavior for key SIP Server elements such as registration, redirect and proxy servers. Users who wish to participate in sessions initiated by SIP may first register with a registration server. The server stores the unique SIP address dedicated to that user (SIP ID). This registration process helps in creation of a service called *location service*, which allows users to find other users in the network and initiate sessions with them through the help of proxy servers. Each of these servers maintains session state information in a standard manner leading to inter-operability. However SIP's RFC [1] emphasizes that the term server refers to logical entities as opposed to physical ones. Nevertheless most of current implementations use dedicated physical machines to perform these tasks. Such designs result in high maintenance cost, limited scalability of the network and have a single point of failure. It is only recently that some work has been done to virtualize the server tasks using other means such as P2P networks [2], [3].

### B. P2P systems

A "peer-to-peer" (P2P) application relies on resources provided by participating peer computers [4]. Typically the peer computers are interconnected with each other in order to implement the application. In large P2P networks [5], all the nodes are not necessarily connected to each other. Instead each node has a group of neighbors that it is connected to directly and uses these connections to reach the other nodes in the network. Such a structure results in an overlay topology which is usually built and maintained by a protocol executed by the participating nodes. This is one of the features that makes P2P systems cost effective, scalable and self organizing [6].

There are two basic approaches to P2P applications, namely centralized and distributed. In centralized approaches such as Napster [7], a server is used to maintain the location of resources such as a file index. This approach cannot scale well and because of its server-client nature, it has a single point of failure and is vulnerable to denial of service attacks. In contrast with the centralized approach, in decentralized approaches every node acts like a server and a client at the same time, so that the notion of a server is eliminated.

*1) Structured vs unstructured:* There are two main types of decentralized systems, structured and unstructured. In struc-

tured systemss such as CAN [8], CHORD [9], PASTRY [10] and TAPESTRY [11] the construction of the overlay is controlled by a hash function and the location of each node in the network is predictable. On the other hand in unstructured systems such as KaZaA [12], [13] and Gnutella [14] although there is some control on how to connect the peer nodes, the placement of the nodes are rather random and unpredictable.

*2) Pure vs hybrid:* In a pure P2P system, all the nodes are treated equally and there is no distinction between the nodes. However in a hybrid system, nodes with desirable features, such as greater resources, are part of a group called superpeers while the rest of the nodes are categorized as ordinary nodes. Hybrid designs help to increase the efficiency of the overlay without compromising its decentralized nature. The hierarchical nature of these architecture makes them a suitable candidate to meet the reliability and efficiency requirements of our design.

### C. Skype and other peer to peer applications

Napster [7] is the first application which was based on the P2P structure. Many subsequent file sharing applications such as Gnutella [14], KaZaA [13] and BitTorrent [15] are also based on a P2P structure. Collaborative web caching systems, distributed data sharing and continual queries are some applications in ongoing P2P projects.

Skype [16] is a free voice over IP application that demonstrates how cost effective P2P systems can be. Skype uses its own proprietary protocol to initiate and modify sessions, instead of SIP. Since there are no published papers on the details of how Skype works one can only speculate that the designers were encouraged by their previous work, KaZaA [13], and used a similar structure, i.e. a decentralized unstructured architecture. Some insights into the operation of Skype are provided in [16].

## III. Design architecture

### A. Integration with SIP

In order to keep our approach compatible with other designs supporting the SIP protocol, mainly the server-client approaches, all SIP messages are encapsulated using extra headers as they enter the overlay topology (Figure 1). If a message needs to leave the overlay topology and enter another network, the extra headers are removed and the original SIP message is transferred to the new network. The benefit of this approach is that unlike solutions proposed in [2], [3], [17] our design is independent of the underlying signaling protocol. This will also expand the potential of our approach to be used with any P2P protocol and provide a stable resource sharing and lookup system.

Each node in the P2P overlay topology plays the role of the registration server for the SIP user agents connecting to it. This allows the connection of multiple user agents to a single node in the P2P overlay topology. Once the user agent contacts a node for registration, the user's SIP ID is stored at that node. Further if the current node is an ordinary node, the registration message is also passed to this node's superpeer. This results in limiting the scope of searches to superpeers only. Once user agents are registered in the overlay topology, initiating a call from a user agent to another involves three steps. First the INVITE message is encapsulated as it leaves the user agent initiating the session and enters the P2P module. Then a lookup for the destination user agent based on SIP user ID is performed by searching at the overlay topology layer. Finally the INVITE message is forwarded to the destination user agent, through the node at which this user agent is registered and after removing the P2P headers from the message. At each stage of the signaling, necessary responses, such as ACK and RINGING, are sent to the user agents by the nodes.

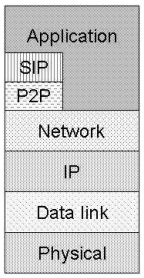


Fig. 1. Integration of SIP with P2P

One can see that handling the SIP messages as explained above is similar to the implementations based on server-client approaches. The major difference is in the virtualizing of the lookup through the use of *location service* [1], which performs the search through the P2P overlay topology. This results in the elimination of dedicated registration and proxy servers that are usually part of the traditional server-client based approaches to SIP. As is normally the case for SIP once the destination peer is found and the session is established the data messages are exchanged between the two peers directly with no involvement of the overlay topology.

To improve the search performance every time a session is initiated the node responsible for the user agent that initiated the call caches the result of the search to speed up future calls. When searching for a user this cache is consulted first. If there is a previous result stored at this location, the node confirms the validity by contacting the destination node as indicated by the cache table. Upon success the search is completed. Only in case of a failure a search through the overlay topology is necessary. This approach helps to reduce the number of searches required for frequent contact list members of a user.

### B. P2P overlay topology

The P2P architecture used in this paper is based on a gossiping protocol for constructing unstructured, hierarchical superpeer overlay topologies proposed by Alberto Montresor [18]. There are four different sets of neighbors maintained at three different layers to achieve this overlay topology. At the first layer, nodes are members of an underlying randomly connected overlay topology and use their connections to transfer information regarding their status to their neighbors. The next two sets of neighbors include the set of all superpeers and the

set of superpeers that are not loaded to capacity with clients (the underloaded set). Finally each superpeer maintains a list of ordinary nodes for which it is currently responsible [18]. Throughout the rest of this paper we will use the term client to refer to ordinary nodes.

The gossiping information transferred through the underlying topology is used by the neighbors to learn about other nodes in the network and decide how to reorient themselves to minimize the number of superpeers. This is achieved by each superpeer trying to transfer as many of its clients as possible to underloaded superpeers with higher capacity and available room. Once a superpeer has transferred all its clients to other nodes, it can itself be accepted as a client by underloaded superpeers to approach the *target topology* that has minimum number of superpeers.

*1) Modifications to Montresor's algorithm:* The algorithm proposed by Montresor in [18] was modified to achieve better performance under highly dynamic conditions, where nodes join and leave the network frequently and their capacity can vary over time. The first modification involves the ability of a superpeer to switch its role with one of its clients who has a higher capacity (can handle more clients).

The original protocol by Montresor, only allows a client to become a superpeer if the superpeer receiving clients during a transfer has two properties. First, it does not have enough capacity to accept the superpeer which is transferring its clients as a client. Second it has a client with higher capacity than the superpeer which is transferring its clients. In such a case the client with greater capacity switches its place and role with the superpeer which is transferring its clients.

Although at first this may seem sufficient to reach the target topology, the following example (Figure 2) demonstrates that the protocol as stated in [18] does not guarantee reaching minimum number of superpeers. In this example nodes A and B are superpeers while all the other nodes are ordinary nodes. The numbers presented inside each node indicate their capacity. At stage I, superpeer A learns about superpeer B through gossiping and since it finds node B to have higher capacity it tries to transfer as many of its clients to node B, picking these clients randomly. After accepting as many clients as it can, node B can no longer accept the remaining client of A (stage II). At this point B analyzes its clients and cannot find a node with higher capacity than itself in its clients list and no further actions are taken. If the capacity of all nodes remains constant there will be no more transactions and the system remains with two superpeers.

However one can clearly see that if node A had checked its own clients list before the transaction it could have switched its role with the node whose capacity is 10 which is enough to accept all the nodes in the network as clients and reach the target topology. Figure 3, demonstrates the new behavior of the protocol. The new protocol can overcome the problem explained above and it guarantees reaching the target topology (the topology with a minimum number of superpeers).

We introduce two additional modifications to improve performance in the presence of failures. The second modification

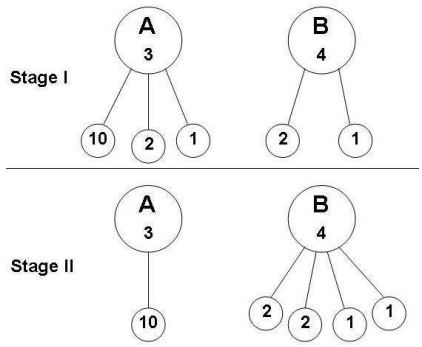


Fig. 2. The protocol as presented by Montresor in [18] does not guarantee reaching the target topology.

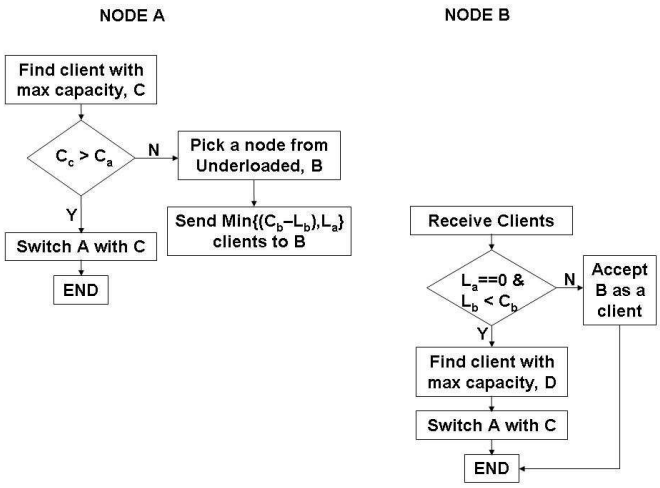


Fig. 3. The modified switching protocol

to the Montresor algorithm allows the clients of a superpeer that leaves the network unexpectedly, to attempt to join other superpeers that they had learnt about through gossiping. A final modification introduces a threshold value smaller than the real capacity of a node for use in gossip messages. This modification results in superpeers not being completely loaded, which is a departure form the original goal of the protocol by Montresor [18]. However, superpeers can now use this remaining capacity to accept clients that lose their superpeers in failures. This will reduce the number of clients that become underloaded superpeers once they lose their superpeer. The latter two modifications result in faster recovery from large failures in the network and avoids explosion in number of the underloaded superpeers. Experiments to test these modifications have been performed and the results are presented in section V.

## IV. STABILITY

The classic definition of stability states that a system is stable if and only if any bounded input results in a bounded output. In order to apply this definition to our system the inputs and outputs should be clearly defined.

The P2P overlay network can be viewed as a black-box which accepts vector

$X = [n, C_1, C_2, ..., C_n, C'_1, C'_2, ..., C'_n, LT_1, LT_2, ..., LT_n, S, S']$

as the input, where $n$ is the number of nodes in the network, $C_1 ... C_n$ are the capacities of the nodes, $C'_1, ..., C'_n$ are the advertised capacities in gossip messages and $LT_1 ... LT_n$ are the lift-time of superpeers. $S$ is the maximum number of virtual links to and from a superpeer and $S'$ is the maximum number of links to and from any node.

We define the output of the system, $Y$, to be the time which it takes for any node in the topology to find any other node in the network. If the network is connected the output is a finite value implying that the network is stable. As a result the system is considered to be unstable only if a search for a node, that exists in the network, requires an infinite amount of time to complete. Searching can be used by a node to find underloaded superpeers in the network. If the system becomes unstable and the topology becomes disconnected the number of underloaded superpeers counted by this node will not be correct since the corresponding search will not be completed in a finite amount of time. As a result one can define stability of the system based on the ability of the topology to find the underloaded superpeers and to reach the target topology.

## V. Experimental results

A number of simulations have been performed to study the behavior of this proposed overlay topology as a function of its variables and to asses cost as a function of network size [18]. We are mainly concerned with the performance of this approach to initiate, manage and terminate sessions using SIP in a real environment, so the system was implemented and assessed experimentally. The algorithm was programmed (using the C programming language) and the resulting program was installed on a cluster of IBM blade servers each with two 2.8GHz Xeon processors and 2GB of RAM. The final testing environment consisted of 1000 nodes running the P2P-SIP application, allowing any node to initiate sessions with any other node. In the initial set of tests, Cisco 7960 IP phones as well as Xten's Xlite softphones were used to validate the functionality of the encapsulation method used to transfer SIP messages.

The next set of tests concentrated on the stability and performance of the substrate P2P topology. In all of the tests, when a node is added to the network, it is given the address of one other node currently part of the topology to connect to. The assignment is made by using a uniform random variable at bootstrapping servers. In order to test the worst case scenario, nodes are removed from the network using ungraceful shutdown. In other words the node is simply disconnected from the topology without informing any of its neighbors about its removal. In all the tests, unless stated otherwise, the number of neighbors at the underlying layer is limited to 20, the number of neighboring superpeers for each superpeer is set to 5 and the capacity of each node is

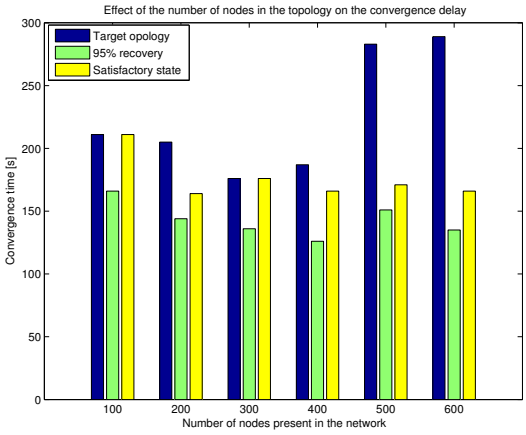


Fig. 4. Effect of the number of nodes currently present in the topology on the stabilization delay as nodes are added to the network. In each case 50% of the network size was added to it.

5. These numbers were chosen to allow a network of limited number of nodes grow large at the superpeer layer.

The tests are categorized into four groups. First we study the behavior of the overlay topology as different number of nodes are added and removed from it. Second, we measure how the systems performance varies with its size. Third, we asses the benefits of our modifications to the Montresor protocol. Finally we study the search performance of the overlay topology as it is the main functionality provided to the signaling layer. We note that in our experiments the overlay topology is only used as a control plane that provides a location service. All data messages are transfered between the two end points without the interference of the overlay topology.

Figure 4 shows the behavior of the topology as nodes are added to networks to increase the size by 50%. In every case the test starts with $n \in \{100, 200, 300, 400, 500, 600\}$ nodes already part of the topology. The network is then allowed to evolve until it reaches the target topology (where the number of superpeers is minimized). Next 50% of the current size of the network is calculated and that many additional nodes are added to the network and the time required for the topology of 1.5n nodes to reach stabilization is measured. In each case three different measurements were carried out. The first bar, represents the time required for the topology to reach the target topology with a minimum number of superpeers. The middle bar, represents the time that was required until the removal of 95% of the total number of underloaded superpeers caused by addition of the new nodes. Finally the third bar indicates the time needed for the number of underloaded superpeers to be less than 5% of the total number of superpeers. Unless specified otherwise, in the remainder of this paper we will use the last criterion to define when a "satisfactory state" has been reached.

It is clear from Figure 4, with the exception of the case with 100 nodes in the initial network, the time to reach the satisfactory state is almost independent of the size of the

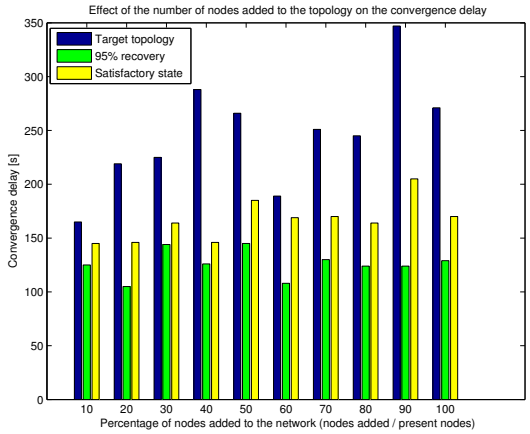


Fig. 5. Effect of the number of nodes added to the topology on the stabilization delay

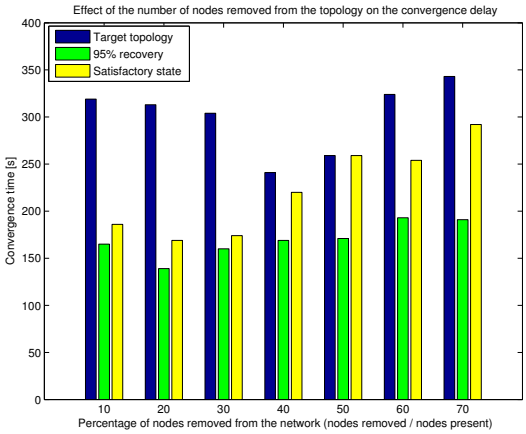


Fig. 6. Effect of the number of nodes removed from the topology on the stabilization delay.

network. This agrees with the simulation results obtained by Montresor [18] and demonstrates the ability of the network to adjust itself to changes rapidly and independently of the number of users participating in the overlay topology.

Figure 5 represents the behavior of the network as the number of nodes added to the topology varies from 10% to 100% of the original size of the network (300 nodes). The bars in the graph correspond to the same definitions given for Figure 4. Again although the time required to reach the target topology grows as the number of added nodes increases, the time to reach the satisfactory state is bounded between 140 and 200 seconds, increasing slightly as the percentage of added nodes increases.

The results in Figures 4 and 5 together indicate an interesting feature of the Montresor algorithm. The addition of new nodes causes a departure from the satisfactory state due to the creation of underloaded superpeers. However the addition of nodes also provides new opportunities for creation of loaded superpeers, and this packing of superpeers proceeds largely in parallel, so that the time to convergence to satisfactory state basically remains constant.

In Figure 6 we analyze the effect of nodes leaving a 900 node network ungracefully. Since failure of each superpeer may result in some of its clients having to become underloaded superpeers, we expect to have a higher number of underloaded superpeers in this case compared to the tests involving addition of nodes to the network. This results in the pattern observed in Figure 6, where the delay increases as the number of nodes removed from the topology is increased past the 40% limit. In such cases the probability of superpeer failures increases resulting in more and more clients having to look for a new location in the network. The system however, remains stable (as defined in section IV), finds the underloaded superpeers and reaches the target topology in a finite amount of time.

Figure 7 compares the original protocol with the case that superpeers advertise a smaller capacity in gossiping messages than their real capacity. As expected the maximum number

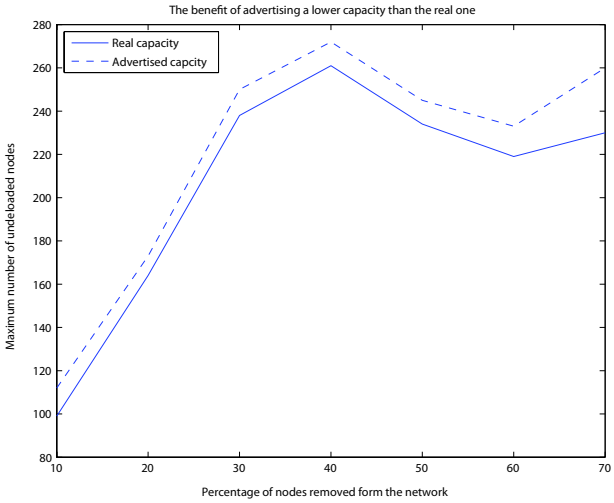


Fig. 7. Benefit of advertising a lower capacity than the real capacity of a node.

of underloaded superpeers generated after node failures in the topology is reduced by up to 12%.

We investigate the behavior of the topology as nodes are added and removed based on a random distribution and comment on the stability of the system. Prior simulations have shown the behavior of the network when the capacity of nodes changes based on a uniform and power law distributions [18]. Here we would like to show that the network remains stable and reaches the target topology when nodes join and depart according to an exponential distribution.

In Figure 8 we analyze the probability distribution of the time to from when the topology reaches a satisfactory state to the time of the latest insertion. The node interarrival time is specified by an exponential random distribution with $\lambda = 0.15$. The test was performed 20 times, each time adding 100 nodes to a network of size 300 nodes and the resulting points are presented in Figure 8. The fitted curve suggests that the distribution of the time between the last node join and the

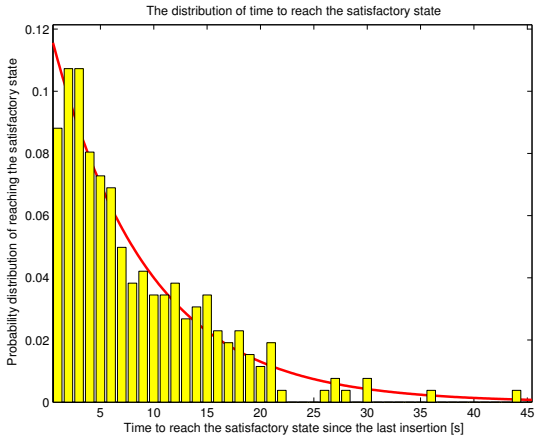


Fig. 8. The distribution for the time it takes for the system to stabilize since last insertion of a node, if the delay between adding two nodes to the network is an exponential random variable with $\lambda = 0.15$

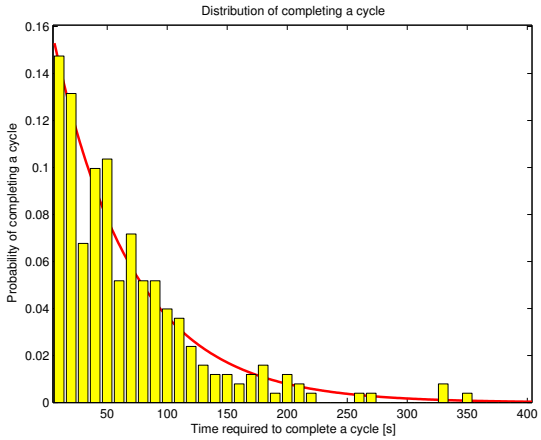


Fig. 9. The distribution for the time it takes for the system to return to the satisfactory state, after it was forced out of it by the addition of nodes. The delay between adding two nodes to the network is an exponential random variable with $\lambda = 0.15$.

system reaching a satisfactory state is also exponential.

We measure the distribution between instants when the topology reaches the satisfactory state when the network is subjected to exponentially-distributed node interarrivals. Since there appears to be a natural, network-size-independent time to converge to the satisfactory state, we conjecture that as long as the node arrival rate is bellow some threshold, then the topology will converge to a satisfactory state in finite time. Equivalently, the number of underloaded superpeers should not tend to infinity bellow this threshold. Figure 9 shows that for arrival rate 0.15 this is indeed the case. Figure 10 shows the distribution of time to reach a satisfactory state when $\lambda = 0.25$. It is obvious that the number of the underloaded nodes goes to infinity. Figure 10 shows that the time to achieve satisfactory state increases without bound for arrival rates 0.25.

An additional set of experiments investigated the case where the nodes are removed from the network with exponentially-

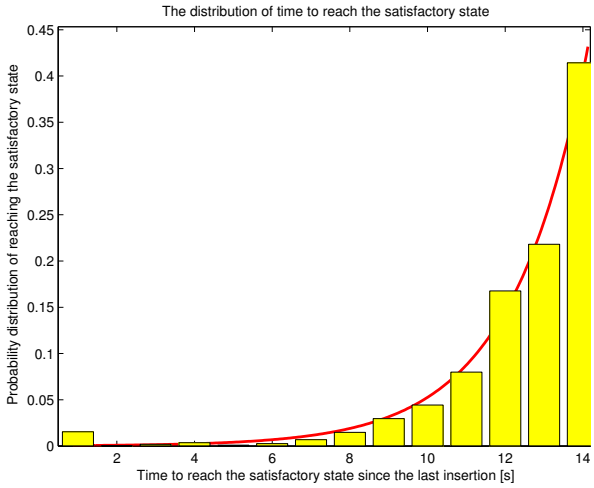


Fig. 10. An example which shows if $\lambda = 0.25$, the number of the underloaded nodes grows exponentially.

distributed inter-removal times. Again as long as the node removal rate is below a threshold ($\lambda = 0.12$) the system remains sable and returns to a satisfactory state in a finite amount of time.

A final set of experiments considered the performance of the overlay topology in terms of lookup delay. In these tests, a randomly selected node in the topology searches for all the other nodes in the network. The test was performed 5 times and the results for the average lookup cost are presented in Figure 11. In order to keep the results independent of the physical location of the nodes, instead of measuring the absolute time taken by a search we count the number of hops traversed by a query message. Since the search is performed at the superpeer layer and since each superpeer connects to m (5 in this experiment) other random superpeers, one can expect the lookup cost to increase as the $m^{th}$ root function of the total number of superpeers. This notion is supported by the fitted curve in Figure 11 which shows behavior according to the $5^th$ root. One might also expect the search performance to improve as the number of connections between the superpeers increases. To test this conjecture the experiment was repeated for networks of different connectivity degrees to produce performance graphs of the type shown in Figure 11. A curve was fitted to each performance to identify the degree root that best fit the data. Figure 12 shows the resulting best degree root from the fitting as a function of the connectivity degree. It's clear that the performance increase slows down as the connectivity degree increases past a threshold value. This behavior is the result of the randomness in the connections. In other words the resulting topology is not guaranteed to expand completely in all dimensions and the probability of loopback connections increases as the degree of connectivity increases.

## VI. Conclusion

This paper presents a new approach to implementing SIP over P2P networks. To the best of our knowledge this is

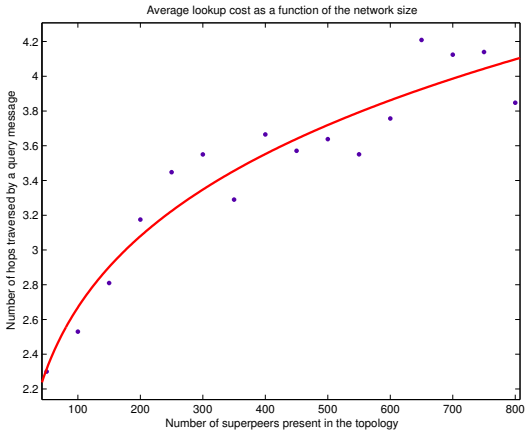


Fig. 11. Average number of hops traversed by a search query approximated by a root function.

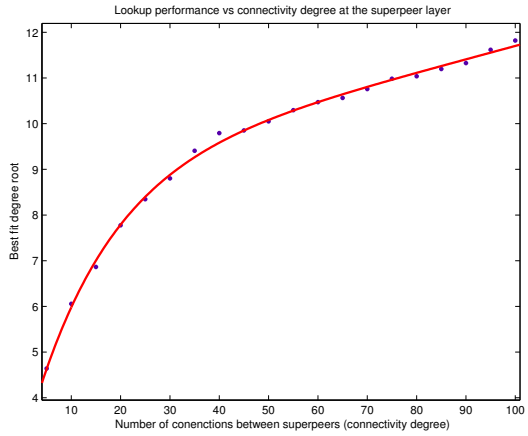


Fig. 12. Performance improvement achieved by increasing connectivity degree of the topology.

the first time SIP is integrated with a fully unstructured P2P network. The modifications we have added to the Montresor algorithm provide significant performance improvement and enhance the robustness of the protocol. We have investigated the stability and scalability of the proposed system and shown that it has robust behavior in a wide range of scenarios.

In future work, it would be beneficial to provide a fully distributed search algorithm to optimize the search time in very large scale networks. In addition, the behavior of the system and service delivery guarantees under high volume of traffic (data, voice and SIP messaging) needs to be studied. Support for bypassing firewalls and security issues are among the other challenges that will be addressed.

## REFERENCES


[1] J. Rosenberg and H. Schulzrinne and G. Camarillo and A. Johnston and J. Peterson and R. Sparks and M. Handley and E. Schooler, "SIP: Session Initiation Protocol." RFC 3261, Jun 2002.

[2] Kundan Singh and Henning Schulzrinne, "Peer-to-peer Internet Telephony using SIP." New York Metro Area Networking Workshop, Sep 2004.

[3] D. Bryan and C. Jennings, "A P2P Approach to SIP Registration and Resource Location." IETF Draft (draft-bryan-sipping-p2p-01), Jul 2005.

[4] Rudiger Schollmeier, "A definition of peer-to-peer networking for the clasification of peer-to-peer architectures and applications," in *Proceedings of the 1st IEEE International Conference on Peer-to-Peer Computing*, Aug 2001.

[5] Shelley Q. Zhuang and Ben Y. Zhao and Anthony D. Joseph and Randy H. Katz and John D. Kubiatowicz, "Bayeux: An Architecture for Scalable and Fault-tolerant Wide-area Data Dissemination," in *Proceedings of NOSSDAV*, June 2001.

[6] R. Schollmeier, "Why peer-to-peer (P2P) does scale: an analysis of P2P traffic patterns," in *Proceedings of the Second IEE International Conference*, Sep 2002.

[7] Lechner, U. and Schmid, B.F., "Communities-business models and system architectures: the blueprint of MP3.com, Napster and Gnutella revisited," in *Proceedings of the 34th IEEE Annual Hawaii International Conference*, Jan 2001.

[8] Sylvia Ratnasamy and Mark Handley and Richard Karp and Scott Shenker, "Application-Level Multicast Using Content-Addressable Networks," *Lecture Notes in Computer Science*, vol. 2233, 2001.

[9] Ion Stoica and Robert Morris and David Karger and M. Frans Kaashoek and Hari Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*, pp. 149–160, ACM Press, 2001.

[10] A.Rowstron and P.Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," *Proceedings of 18th IFIP/ACM International Conference on Distributed Systems Platforms*, pp. 329–350, Nov 2001.

[11] B.Zhao and J.Kubiatowicz and A.Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing." Berkeley: Computer Science Department, University of California, Technical Report, UCB/CSD-01-1141, 2001.

[12] N. Leibowitz and M. Ripeanu and A. Wierzbicki, "Deconstructing the Kazaa Network," in *Proceedings of the 3rd IEEE Workshop on Internet Applications (WIAPP'03)*, Jun 2003.

[13] Beverly Yang and H. Garcia-Molina, "Designing a super-peer network," in *Proceedings of the 19th IEEE International Conference*, pp. 49–60, Mar 2003.

[14] M. Ripeanu, "Peer-to-Peer Architecture Case Study: Gnutella Network," in *Proceedings of the 1st IEEE International Conference on Peer-to-Peer Computing*, Aug 2001.

[15] D. Hales and S. Patarin, "Computational Sociology for Systems "In the Wild": The Case of BitTorrent," *Distributed Systems Online, IEEE*, vol. 6, pp. 94–97, Jul 2005.

[16] J. Lennox and H. Schulzrinne, "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol." Computer Science Department, University of Columbia, Technical Report, cucs-039-04, Sep 2004.

[17] David A. Bryan and Bruce B. Lowekamp, "SOSIMPLE: A SIP/SIMPLE Based P2P VoIP and IM System." College of William and Mary, Computer Science Department, Technical Report, Nov 2005.

[18] Alberto Montresor, "A Robust Protocol for Building Superpeer Overlay Topologies," in *Proceedings of the 4th IEEE International Conference on Peer-to-Peer Computing*, pp. 202–209, August 2004.