Appears in AAA-IDEA 2005 ©IEEE

SOSIMPLE: A Serverless, Standards-based, P2P SIP Communication System

David A. Bryan and Bruce B. Lowekamp
Computer Science Department
College of William and Mary
Williamsburg, VA 23185
{bryan, lowekamp}@cs.wm.edu

Cullen Jennings
Cisco Systems, Inc.
170 West Tasman Drive, MS: SJC-21/3
San Jose, CA 95134
fluffy@cisco.com

Abstract

Voice over IP (VoIP) and Instant Messaging (IM) systems to date have either followed a client-server model or have required the use of clients that do not follow any VoIP or IM standard. We present SOSIMPLE—a fully decentralized, P2P, standards-based approach to communications. By building on the existing SIP/SIMPLE infrastructure for VoIP and IM, we support reuse of clients, network infrastructure, and open-source protocol stacks designed using current standards. By avoiding traditional centralized architectures, SOSIMPLE addresses corporate privacy concerns, eliminates dependency on constant Internet connectivity, and supports ad hoc groups. SOSIMPLE implements a DHT overlay based on Chord[24] using SIP messages, replicating location information for reliability. The DHT is used only for lookups, with actual communication passing directly between clients. We discuss important issues for security and authentication, as well as adaptations of conventional P2P routing for the social networks typical of personal communications. We are testing a prototype implementation of SOSIMPLE and anticipate a public release in the near future.

1. Introduction

Voice over IP (VoIP) and Instant Messaging (IM) are increasingly popular communications systems for private, corporate, and academic purposes. Despite their popularity, requirements for central servers (proxies), maintained locally or by third parties, limit growth and place a burden on users. For example, companies concerned about intellectual property typically ban the use of current, centralized IM services, instead deploying their own in-house services. These private IM realms address their security concerns but may not allow interaction with external customers and can be difficult for small organizations to deploy and maintain.

This work is partially supported by the Cisco University Research Program. David A. Bryan is partially supported by the Virginia Space Grant Consortium.

SOSIMPLE combines the SIP/SIMPLE [17] family of IETF standards for VoIP and IM with the Self Organizing properties of a Distributed Hash Table (DHT) P2P mechanism. The result is a standards-based, decentralized communications system. Our design allows for compatibility with and reuse of existing SIP/SIMPLE network elements. Because the implementation requires only slight modifications to the existing SIP protocol, we maintain compatibility with existing SIP infrastructure, such as SIP-enabled IP phones and gateways. Furthermore, modules can be added to existing IM clients to ensure functionality using a familiar interface. The P2P overlay is built using a DHT created through exchanging SIP messages, which are typically well-supported by existing firewalls, and therefore retain compatibility with current network infrastructure. We have submitted a draft for a P2P SIP protocol based on SOSIMPLE to the IETF [1].

The primary contributions of this work are:

- Creating a fully-distributed, open P2P system for VoIP and IM by extending existing standards.
- Identifying the security requirements for a decentralized communications system running on a non-secure network and proposing solutions that meet these requirements.
- Proposing P2P technology that reflects the connectivity patterns of personal communications systems.

This paper presents several scenarios in which existing IM/VoIP systems are unsatisfactory as motivation to derive a set of requirements for a new approach. We explain why a P2P SIP approach is best suited to meeting these requirements, and briefly cover SIP, P2P, and related work. We present the SOSIMPLE architecture and discuss the important research issues raised by our work.

2. Background and Challenges

2.1. IM and VoIP

IM/VoIP systems provide three basic functions. *Resource location* is responsible for identifying and locating other users so conversations can be established. *Session establishment and management* creates, controls, and terminates text or multimedia sessions between users. *Presence* is the ability of users to monitor other user's connection status and to be notified when they arrive or leave.

The majority of today's VoIP and IM systems are centralized. Users connect using a User Agent (UA), which may be a software application or a hardware device such as an IP or mobile phone. The UAs connect to a central sever, usually called a proxy, softswitch, or gatekeeper.

Several VoIP protocols are in wide use. Older protocols such as the ITU's H.323 place most intelligence in the central server, while SIP pushes most of the intelligence to the UAs but relies on central proxies for some functions. The devices supporting these protocols represent a significant investment.

Commercial IM protocols used by AOL, Yahoo and MSN require users to connect to a server hosted at the provider's site. A few commercial products, as well as the emerging IETF standards XMPP and SIMPLE [2, 14, 20] allow a corporation or group to maintain their own server, but these are still centralized.

2.2. SIP and SIMPLE

SIP and SIMPLE are text-based protocols derived from HTTP [4], therefore message types and traffic are similar to web traffic. SIP is a general protocol for establishing and controlling multimedia sessions, but is most widely used for VoIP. SIP embeds Session Description Protocol (SDP) [6] information to specify the media parameters. While SIP devices can be configured to communicate directly with each other, in systems larger than two UAs a central server or proxy[1] is usually used. A SIP proxy is a more sophisticated network element than a web proxy, maintaining location information about users and UAs, as well as routing signaling traffic between UAs. Proxies also provide authentication and user name administration.

The signaling portion of the call consists of SIP messages flowing through the central proxies connecting UAs. Media flows directly between the UAs for efficiency, typically in the form of Real Time Protocol (RTP) [21] packets. This separation is somewhat analogous to P2P file sharing, where the files are exchanged directly between nodes.

SIMPLE is a set of SIP extensions for IM. Most aspects of SIP and SIMPLE are identical, and we will note differences where appropriate. The most important difference is that IM messages are generally carried directly in the signaling path—there is typically no separate media stream when using IM.

2.3. Scenarios Requiring a New Approach

While the traditional IM/VoIP solutions have succeeded in many scenarios, there are a number of environments in which the current protocols fail for technical, financial, security, or social reasons. We will briefly describe some of these scenarios as motivation for SOSIMPLE's approach.

- **Security Conscious Small Organizations:** Services typically used by individuals and small organizations, such as AOL's AIM, Microsoft's Messenger, and the Vonage VoIP service, use servers located at the provider. Because all communication, even internal, passes through this external party, many organizations have banned the use of these services. While large organizations can install their own internal systems, the personnel and equipment requirements are too great for smaller organizations.
- **Limited or No Internet Connectivity:** If infrastructure is damaged in a large scale civil emergency, the user is located in a remote location in the developing world, or an ISP simply experiences a temporary failure, outside connectivity may be intermittent or nonexistent. Communication should be possible between connected nodes without requiring access to the Internet and without configuring a new server.
- **Ad-hoc and Ephemeral Groups:** Groups that collaborate may be ephemeral, such as in a meeting, classroom, or conference setting. These users may want to exchange text messages, establish voice or video sessions, or use collaborative programs. Such users should be able to easily establish a small network for the duration of the event without configuring or connecting to a server.
- **Censorship or Impeded Access:** Some users may not be allowed to access centralized, persistent servers because of government censorship or a carrier that prevents access to competing services (such as a telephone company prohibiting VoIP on a DSL line they provide).
- **Scalability:** While many users enjoy the functionality offered by services such as AIM or the Skype

1 In the SIP specification [17], the functions of a server are broken into several network elements, including a proxy and a registrar, often implemented together. We will refer to these elements collectively as a proxy here for convenience.

[23] VoIP service, the reliance on a central authority for availability and access is inherently unscalable and failure-prone. A decentralized system utilizing standards-based equipment should be more reliable, easier to join, and ultimately less expensive for the users.

3. Requirements

The following features are required to solve the problems we have outlined.

- **No Central Server:** In several scenarios, security or connectivity prohibit contacting an outside server, therefore we require none be present.
- **No Central Naming Authority:** Users should be allowed to select names, and a distributed system should mediate the assignment and verification of identity. No central authority should be involved in naming.
- **Simple System Discovery:** For scenarios where appropriate, a simple mechanism must be available for locating the system, for example a broadcast mechanism.
- **Scalable Number of Users:** The system should grow and scale as new users join. No new resources, other than those the new user brings, should be required.
- **Privacy:** Users should be able to locate other users without having to go outside the system. Messages between users must not leave the system or be interceptable by other users within the system.
- **Multiple Realms:** Users should be able to configure their own instance of a communication system, in which the participants are able to limit membership to those users they wish to allow to participate. We refer to each of these individual instances of a communication system as a *realm*. For example, a security conscious company would typically utilize a private realm for internal messages, with the option of interacting with external realms to communicate with external users.
- **Interconnection:** While traffic within a realm should be restricted to that realm, the design should support interconnecting different realms when desirable, to allow users to communicate. To preserve the requirement above, connectivity must not be achieved by merging realms.
- **User Mobility Support:** Users should be able to use different clients while maintaining access to user-specific information such as configuration and buddy lists. Users should be able to quickly move from one device to another, for example between home, office and mobile devices.
- **Compatibility and Reuse:** The system must be compatible with as much existing infrastructure as possible. Users should be able to use existing VoIP or IM clients to connect and should be able to leverage equipment such as VoIP gateways (used to connect to the public phone network). As much existing code as possible should be used in the implementation.

4. P2P SIP-Based Solution

Our solution takes a P2P approach to SIP, resulting in a distributed communications system with no central server.

Why SIP? SIP has emerged as the standard protocol for VoIP. Although some systems use older standards or proprietary mechanisms, the majority of new VoIP development uses SIP. SIP with SIMPLE extensions is also extensively used by existing IM clients, for example in Microsoft's MSN Messenger [11].

Using the open SIP/SIMPLE standard helps us meet our requirement for *Compatibility and Reuse*and allows the system to leverage open source stacks and applications [13, 26], as well as integrate with the tremendous number of existing SIP phones, SIMPLE IM clients, and SIP gateways. Additionally, using the SIP/SIMPLE family of protocols, gives us both IM and VoIP functionality—something not possible with protocols such as H.323 or XMPP.

Why P2P? Our requirements of *No Central Server, No Central Naming Authority, Scalable Number of Users,* and *User Mobility Support* led to our decision to use P2P. Dynamic DNS [25, 28] has been suggested as one approach to create a serverless communication system. While a dynamic DNS approach addresses some of the issues we raised, there are several scenarios in which such an approach is lacking. Dynamic DNS requires a central DNS server be available. In situations where there is no external connectivity, this is not possible. Additionally, unless users maintain their own DNS servers, this is not a service a typical end user has access to. Finally, if device mobility is important, the latency in making changes to DNS may be undesirable. The most obvious alternative, and the one we have chosen to pursue, is to take a P2P approach.

Why P2P over SIP? In order to provide the features of SIP IM/VoIP, applications will need to incorporate a SIP stack. To prevent applications from having to add a second stack, we use SIP messages to carry the P2P traffic. Additionally, there seems to be no clearly defined, standards-based approach to implementing the messages for a P2P system, while SIP is a mature, established protocol. Finally, since many border devices (firewalls, traffic shapers, etc.) are already able to recognize SIP traffic—as well as recognizing and often banning P2P traffic—using SIP as the underlying mechanism will ease deployment.

How scalable? An important question in evaluating SOSIMPLE is its scalability. Our design preserves the SIP architecture's approach of sending the signaling through the SIP infrastructure, while the media flows directly between UAs. Using this approach in our design ensures that the P2P overlay need only be used for resource location.

5. Related Work

WASTE was a short-lived P2P system for file sharing and IM. Traffic within the overlay was encrypted against outside snooping, but all traffic flooded all nodes, and all nodes shared the same encryption key. WASTE was non-standard, requiring a special client, and groups were limited to 50 users.

Skype [23] allows users to make free phone calls between Internet users and offers public phone connectivity for a fee. The system is proprietary and is not compatible with non-Skype devices. Skype provides VoIP and IM capabilities. While partially P2P, username control, authentication, and some other services are controlled by Skype's central servers.

A similar and also non-standard project, vop2p [27], is available open-source but is quite preliminary in nature and seems not to have been active for over a year.

The EarthLink R&D group has created an application called SIPshare [22], using SIP as the underlying transport protocol for a file-sharing P2P system. Similarly, NUTSS [5] is a project exploring how using SIP and related technologies might be useful as a general mechanism for transport in P2P networks and in particular for NAT traversal. In contrast, SOSIMPLE uses P2P to decentralize a SIP/SIMPLE communications system.

Singh and Schulzrinne at Columbia University have recently begun exploring a concept similar to ours [9]. Some preliminary information about their research is available on their webpage. Key differences between these approaches include having different SIP mechanisms underlying the system and taking a recursive, rather than iterative approach to P2P search. In addition, their approach does not currently use user subscriptions and the resulting information for routing, but instead relies exclusively on the traditional Chord approach to choosing fingers.

6. The SOSIMPLE P2P SIP Architecture

Because our design has no central servers and nodes communicate directly with one another, there are some differences between our approach and traditional SIP. Peers connect to a few other peers in the overlay network and use these nodes as the destination for most SIP messages. Nodes act both as UAs and as proxies simultaneously. Collectively, the peers replace the functionality of SIP registrars and proxies—each node being responsible for those roles for some portion of the overlay.

6.1. Structure and Messages

Nodes are organized in a DHT based on Chord. Each node is assigned a Node-ID created by hashing (using SHA-1) the real IP address (found using a mechanism such as STUN [18] if the node is behind a NAT) and appending a port. We use the same algorithms used by Chord to maintain our overlay network, but as we discuss below, we use SIP messages to implement the DHT operations. We implement redundancy in the way as Chord, storing resources on the node responsible for a particular resource as well as $k = 4$ successor nodes.

SOSIMPLE nodes maintain a smaller number of finger table entries than in Chord. Chord attempts to provide rapid resource location and assumes that users will query many times for many distinct items. To support this, Chord keeps 160 entries in their finger tables. While we desire reasonable resource location time, users often message or call the same user many times (for example co-workers, friends or family members). Additionally, delays of even a full second are insignificant when setting up a phone call or IM session. As such, we maintain only 16 finger table entries for routing but prefetch entries for contacts in the user's buddy list as well as caching previously contacted nodes for future use.

All messages for maintaining the DHT, registering users, locating resources, and establishing sessions are SIP messages. While SIP was not designed with supporting P2P in mind, it was designed to be extensible by developers. We have not needed to add new methods to SIP to implement P2P functionality but rather have only needed to define a number of new SIP headers. Defining new headers is a common way of extending SIP to support new functionality while preserving compatibility with existing applications.

We use the SIP REGISTER message to pass the overlay information between nodes, as well as the original use of sending user location information to a registrar (peer). To distinguish between the two ways in which we user REGISTER messages, we refer to the traditional use for registering users as *User Registration.* When using REGISTER for DHT operations such as entering, leaving, or maintaining the overlay, we refer to this as *Node Registration.* Both of these uses are discussed below.

6.2. Node-level Operations

New nodes join the system by exchanging a number of REGISTER messages. The new node uses its IP address, as discussed above, to calculate a Node-ID. Once the new

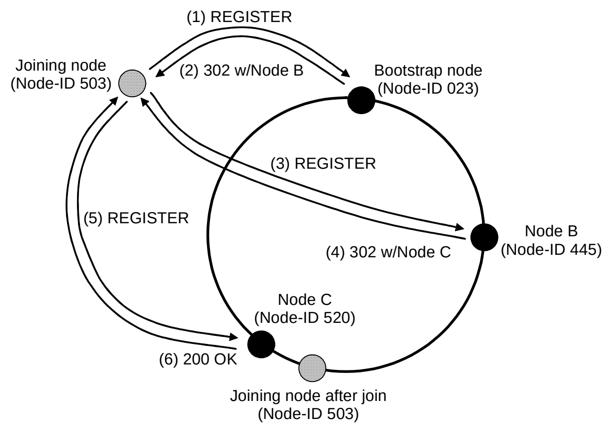

Figure 1. An example of a new node with Node-ID 503 joining the overlay.

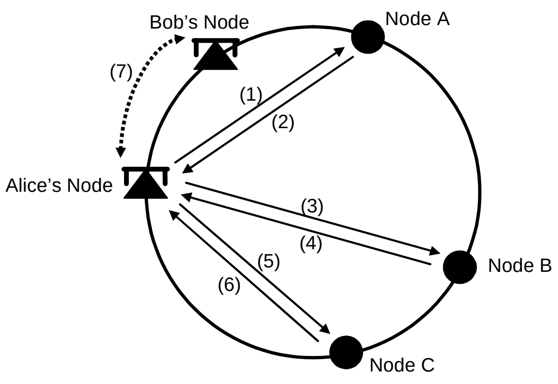

Figure 2. Alice locates user Bob and establishes a communications session.

node has joined the overlay, it will be responsible for storing information (user registrations) associated with the portion of the overlay mapped to that calculated Node-ID. The joining node must find the node currently responsible for that region, insert itself into the overlay, and transfer the information for the region it will be responsible for from the node presently storing that data.

We provide an example of a node join in Figure 1. When a new node wishes to join the overlay it first must locate some node already in the overlay, referred to as a bootstrap node. Presently this node is located via out-of-band mechanisms. The joining node calculates its Node-ID, 503 in our example, and sends it in a REGISTER to the bootstrap node, which has Node-ID 023 (1). Assuming that the bootstrap node is not the node currently responsible for this region, it responds with information about the nodes it knows nearest to where the joining node will be placed in the overlay—in this case, Node B, with Node-ID 445. This information is passed back in our new headers within a SIP 302 Moved Temporarily response (2). The joining node repeats the process, using this nearer node as the new bootstrap node (3-4). Finally, the joining node reaches the node that is currently responsible for maintaining the appropriate section of the overlay, In this case Node C, with Node-ID 520. Node C responds with a SIP 200 OK response including detailed information about nearby neighbors in the headers (5-6), allowing the joining node to insert itself into the overlay.

Further messages, not shown, are sent between the joining node and the node previously responsible to exchange the actual information (registrations) in the overlay the new node must store. Additionally, the new node will send messages to other nodes in the overlay to update their finger tables. Please see our IETF Internet-draft [1] for details of the exact format of these messages. At this point the node has joined the overlay and node registration is complete. Any users associated with the node are not yet registered at this point, but the user operations can now take place.

6.3. User Operations

User registration and resource location (finding users) are decentralized to meet our requirements for *No Central Server* and *No Central Naming Authority* . In order to register, or to contact another user, the node responsible for storing that user's information must be located. We will discuss registration and establishing a session individually, but first will address resource location in general. We illustrate the resource location process in Figure 2.

When a node wishes to find the node responsible for a particular user—either themselves or another user they wish to contact—they begin by hashing the user name to produce a Resource-ID. Because the user's node is already properly in the overlay (node registration has already occurred), the node has a number of finger table entries pointing to nodes within the overlay. In our example, Alice attempts to look up a resource (Bob in our example). Alice's node looks in its finger table and finds the node with Node-ID nearest the Resource-ID to be located. In this case, Node A. Alice's node sends a message to Node A (1). The exact type depends on why Alice wants to locate the resource. Node A is not responsible for that Resource-ID, so it sends a SIP 302 Moved Temporarily reply, including the node it thinks is closest, Node B, in the headers (2). Alice's node tries Node B and again receives a reply with another node to try, in this case Node C (3-4). Finally, Alice's node tries Node C, which is responsible for that resource (5). The result of that request varies depending on why Alice wished to locate the resource, so we now explore those alternatives.

If Alice was registering herself as a user within the overlay (user registration), the messages Alice's UA sent (1, 3, 5) were of type REGISTER. When Node C receives this message, it realizes that it is responsible for storing the registration on Alice's behalf. Node C enters a mapping between Alice's username and her IP address into its registration table and replies to Alice's node with a 200 OK (6). At this point, Alice is registered with the system.

If instead Alice was trying to call Bob, the messages Alice's UA sent (1, 3, 5) were either SIP INVITE messages to establish a VoIP call or SIP MESSAGE messages containing IM data. When these messages reach the desired node, in this case Node C, it looks in its registration table, determines it is responsible for Bob's registration, and looks to see if a registration is present. If there is not, a 404 Not Found is returned (6), indicating the user is not in the system. If Bob is registered, Node C sends a 302 Moved Temporarily reply, providing a contact directly for Bob (6).

Once Alice has the IP address of Bob's UA, a call between these UAs can be established directly between the nodes using conventional (non-P2P) SIP mechanisms without involving the overlay (7). Alice's node caches the information received from Node C and can use that for future communications with Bob. Unless Bob or Alice log off or move their nodes, possibly prompting a new search, the overlay is not involved in any further messages or conversations between Alice and Bob. Once this location information is passed to the UA, the call establishment is unmodified SIP—no P2P functions are required. The resource location portion can be implemented in a thin adapter shim, leaving the SIP UA unmodified. As in traditional SIP, the media also flows directly between the endpoints.

Note that because Alice and Bob's registration data is likely to be stored on different nodes (i.e, their usernames are likely to hash to Resource-IDs that map to different nodes), "Node C" in the register and session establishment example above would most likely be different nodes.

When a P2P UA starts, the user's buddy or contact list is consulted. The SIP SUBSCRIBE/NOTIFY mechanism (again, using P2P mechanisms to locate the resources) is used to attempt to establish subscriptions with these users. These subscriptions track the availability of the buddies. If the buddy is not connected, the subscription will be attempted later. The address of the node responsible for the user information (not the address of the UA itself) will also be added to the node's finger table, as described above. The buddy list is stored as an encrypted file within the overlay, to meet our requirement for *User Mobility Support* . As the SIP PUBLISH draft [12] comes into wider use, we will support this standard.

7. Current Research, Future Work and Open Issues

There are several key areas of research that arise from designing a P2P SIP-based communications system.

7.1. Security and User Authentication

Perhaps the largest open question is how to properly secure a distributed P2P communications system. SIP/SIMPLE offers private-key mechanisms for authentication between UAs and proxies, and for end-to-end encryption [8]. A P2P architecture, however, requires a system that assumes no trust between the user and the system with which they are authenticating.

We are currently evaluating a number of possible approaches. Ultimately, we expect that more than one approach will prove desirable, and different instances of P2P communications systems will weigh the relative strengths and weaknesses of different approaches and choose the one most appropriate for the deployment. In the case of ad-hoc or transient networks, no verification may be required or desirable. The mechanisms we have explored include:

- Simply storing the public portion of a user's key in the overlay. This mechanism provides no guarantee as to the identity of a user, but once the certificate has been obtained one can use it to ensure that future conversations are with the same individual as previous conversations. This mechanism allows communication between nodes to be encrypted using this certificate. No name enforcement is used in this scheme—multiple users could use the same name, and one would need to use certificates and previous conversations to distinguish users. This technique is best when combined with others below.

- Email verification of user identities requires user names be valid email addresses. A user attempting to register is challenged by the node handling the registration to provide a key, which the node sends to the user in email. While this approach ensures the uniqueness of usernames, it has a number of shortcomings. A corrupted node in a single email challenge system can mount a DOS attack against users for which they are responsible. Note however, that because a Node-ID is tied to an IP address, attacks on a specific user are difficult, even with a single party email challenge. A more serious drawback is that this system will not work without email access. Additionally, we would need to solve the problem of trusting other nodes to have properly verified IDs, which is a major security shortcoming.

- The PGP web of trust model is also an alternative. In such a model, a user wishing to join the overlay would create a certificate and ask one or more other users to "verify" the new user's identity by signing the new certificate. By following this path of signed certificates, the node receiving the register can ascertain that the registering user really is who they say they are. The system could be constructed so that only signatures from nodes inside the overlay are considered, or conventional PGP mechanisms could be used, which could require access to systems outside the overlay.
- Centralized certificate stores could also be used. In such a scenario a central authority issues a certificate to a user. Registrations are confirmed by verifying the identity using standard certificate-chain verification mechanisms, such as X.509. Again, this idea requires some centralized authority to be consulted. This is very similar to the way that email signatures and encryption are handled today.
- In many cases, attacks such as the Sybil attack [3] can be used against the approaches defined above. Computational puzzles can be used as a rate limiting mechanism against this. Registrations must be accompanied by a solved puzzle, which is associated with the user ID that is being created. Many alternatives for such puzzles exist, one need only ensure that they can be verified trivially. Jennings has proposed such a standard for SIP clients in a recent Internet-draft [7].

None of these mechanisms using public key authentication are standard SIP. Therefore, they must be implemented between our modified nodes, and public key methods or no security must still be used between the adapters/proxies and non-modified UAs. There are efforts underway within the IETF to standardize end-to-end authentication and we will attempt to incorporate these as they emerge.

7.2. NAT Traversal

While NAT traversal is a topic that has been well explored in the IETF, for example with the STUN, TURN, and ICE drafts [15, 16, 18], how this is best implemented in a P2P system is an open question. STUN and TURN both require access to a public server to determine public IP addresses. We are currently evaluating what portion of users would likely have nodes on publicly routable addresses. If a sufficient number of nodes are likely to be publicly routable, incorporating STUN and TURN servers directly into the P2P nodes would be an attractive option. If few of the nodes will have public addresses, centralized STUN/TURN servers will need to be used instead. In addition, anycast may be a useful mechanism for locating the servers needed for NAT traversal.

7.3. Routing

There are a number of issues relating to routing we are currently exploring.

- Finger table design is a key parameter for SOSIMPLE. We are currently working with simulations and experiments using our implementation to help determine reasonable parameters. Because communications session are established with a new entity less frequently than a file-sharing application searches for files, early results indicate that fewer fingers are needed than are used in Chord, particularly if fingers are kept pointing to buddy/contact list entries and recent contacts. Current work focuses on defining guidelines for the number of fingers based on overlay size, as well as algorithms to adaptively change the finger table size based on the number and distribution of buddy/contact fingers.
- Social routing is another concept being explored. SPROUT [10] explored using social routing for P2P. We are investigating how this applies to communications systems. Because users frequently have buddies on their lists that have similar interests, a communications system should be an ideal candidate for a P2P approach that uses not only DHT mechanisms but also social connections to route messages. We are investigating the costs of routing using traditional P2P mechanisms, P2P mechanics with fingers to contacts (our current approach), and systems that use the overlay as well as queries using social networking.
- The underlying P2P architecture is also an area that we are currently evaluating. While we presently use a system very much like Chord and are evaluating social network extensions to this, we also are working to evaluate other P2P mechanisms, such as those used in Pastry or Tapestry [19, 29].
- Hierarchical routing involves interfacing multiple independent P2P SIP realms. For example, different enterprises and organizations will be running different realms to keep their communications internal. These users may still wish to communicate with users in other realms. Bridging these will be critical and is a research area we have not yet begun to explore.
- Interfacing with existing SIP systems also merits more work. Our current design allows reuse of unmodified SIP clients within our system, but we also need to finalize how calls to non-P2P traditional endpoints are handled. We don't see this as a research problem, but it is a technical issue remaining to be solved.

7.4. Persistent Storage and Abuse Prevention

One final area of future research is investigating persistent storage. Persistent storage is needed for a number of purposes—storing public keys, storing buddy lists, voicemail, and storing configuration information. These functions must be supported without allowing the use of SOSIMPLE as a mechanism for illicit file sharing.

8. Conclusions and Implementation Status

We have worked carefully to ensure that SOSIMPLE meets the requirements for a practical P2P VoIP/IM system. We have removed dependence on a central proxy server, while preserving most of the advantages of a proxy-based system. SOSIMPLE offers compatibility in terms of reuse of existing SIP clients as well as the ability to interface to established SIP systems. Adapting existing technologies reduces barriers to learning the technology, as well as reducing the effort needed to create SOSIMPLE applications. We have taken an approach to security that allows for authentication without requiring a fully secure P2P system. In short, SOSIMPLE meets the need for a more distributed communication system.

We have implemented the SOSIMPLE architecture described here for experimentation and use for internal communications within our research group.

References

[1] D. A. Bryan and C. Jennings. draft-bryan-sipping-p2p-01. http://www.ietf.org/internet-drafts/draft-bryan-sipping-p2p-01.txt, July 2005.

[2] B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, and D. Gurle. RFC 3428 - session initiation protocol (SIP) extension for instant messaging. http://www.ietf.org/rfc/rfc3428.txt, Dec. 2002.

[3] J. R. Douceur. The sybil attack. In *Proceedings of the IPTPS02 Workshop*, Cambridge, MA, USA, Mar 2002.

[4] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. RFC 2616 - hypertext transfer protocol - HTTP/1.1. http://www.ietf.org/rfc/rfc2616.txt, June 1999.

[5] S. Guha, Y. Takeda, and P. Francis. NUTSS: A SIP-based approach to UDP and TCP network connectivity. In *SIGCOMM '04 Workshops*, Portland, OR, Aug 2004.

[6] H. Handley and V. Jacobson. RFC 2327 - SDP: session description protocol. http://www.ietf.org/rfc/rfc2327.txt, Apr. 1998.

[7] C. Jennings. draft-jennings-sip-hashcash-01. http://www.ietf.org/internet-drafts/draft-jennings-sip-hashcash-01.txt, Feb. 2005.

[8] C. Jennings, J. Peterson, and M. Watson. RFC 3325 - private extensions to the session initiaion protocol (SIP) for asserted identity within trusted networks. http://www.ietf.org/rfc/rfc3325.txt, Nov. 2002.

[9] K. Singh and H. Shulzrinne. http://www1.cs.columbia.edu/~kns10/research/p2p-sip.

[10] S. Marti, P. Ganesan, and H. Garcia-Molina. Sprout: P2p routing with social networks. In *First International Workshop on Peer-to-Peer Computing and Databases (P2P&DB 2004)*, Mar 2004.

[11] Microsoft MSN Messenger. http://messenger.msn.com.

[12] A. Niemi. draft-ietf-sip-publish-04. http://www.ietf.org/internet-drafts/draft-ietf-sip-publish-04.txt, May 2004.

[13] ReSIProcate Project. http://www.resiprocate.org.

[14] J. Rosenberg. RFC 3856 - a presence event package for the session initiation protocol (SIP). http://www.ietf.org/rfc/rfc3856.txt, Aug. 2004.

[15] J. Rosenberg. draft-rosenberg-mmusic-ice-04. http://www.ietf.org/internet-drafts/draft-ietf-mmusic-ice-04.txt, Feb. 2005.

[16] J. Rosenberg, R. Mahy, and C. Huitema. draft-rosenberg-midcom-turn-07. http://www.ietf.org/internet-drafts/draft-rosenberg-midcom-turn-07.txt, Feb. 2005.

[17] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. RFC 3261 - SIP : Session initiation protocol. http://www.ietf.org/rfc/rfc3261.txt, June 2002.

[18] J. Rosenberg, J. Winberger, C. Huitema, and R. Mahy. RFC 3489 - STUN : Simple traversal of user datagram protocol (UDP). http://www.ietf.org/rfc/rfc3489.txt, Mar. 2003.

[19] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, 2001.

[20] P. Saint-Andre. RFC 3920-3923 – IETF XMPP RFCs . http://www.ietf.org, Oct. 2004.

[21] H. Shulzrinne, S. Casner, R. Frederick, and V. Jacobson. RFC 1889 - RTP: a transport protocol for real-time applications. http://www.ietf.org/rfc/rfc1889.txt, Jan. 1996.

[22] SIPShare Project. http://www.research.earthlink.net/p2p/.

[23] Skype Technologies, S.A. http://www.skype.org/.

[24] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM 2001*, pages 149–160. ACM Press, 2001.

[25] P. Vixie, S. Thomson, Y. Rekhter, and J. Bound. RFC 2136 - dynamic updates in the domain name system DNS UPDATE. http://www.ietf.org/rfc/rfc2136.txt, Apr. 1997.

[26] VOCAL Project. http://www.vovida.org.

[27] vop2p Project. http://vop2p.jxta.org.

[28] B. Wellington. RFC 3007 - secure domain name system (DNS) dynamic update. http://www.ietf.org/rfc/rfc3007.txt, Nov. 2000.

[29] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A global-scale overlay for rapid service deployment. *IEEE Journal on Selected Areas in Communications*, 2003. Special Issue on Service Overlay Networks.