SOSIMPLE: A SIP/SIMPLE Based P2P VoIP and IM System

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

***Abstract*— Voice over IP (VoIP) and Instant Messaging (IM) systems to date have been either highly centralized or non-standard in nature. SOSIMPLE is a fully decentralized, standards-based P2P communications systems that reuses existing clients. This design is based on currently available open-source software. In this paper we address the challenges of developing a distributed messaging system that preserves the advantages of centralized systems, including secure authentication. This approach opens up new opportunities for decentralized communications systems that are readily available and extensible.**

I. Introduction

Voice over IP (VoIP) and Instant Messaging (IM) are increasingly popular communications systems for private, corporate, and academic purposes. Despite this popularity, requirements for central proxies, maintained locally or by a third party, have limited growth and placed a burden on users. Efforts have been made to incorporate P2P technology, but to date these have utilized non-standard protocols or systems that are not fully decentralized.

Many network elements supporting the current VoIP and IM standard protocols are already in place. Because of the extensive investment of both application development time and equipment costs, a new system should allow for reuse of existing technology. Finally, a system utilizing standard protocols will allow many enhancements for these protocols, as well as the convergence of VoIP and IM, to be seamlessly integrated into a P2P based system.

We propose to combine well-documented, standards-based VoIP and IM technologies with the self-organizing properties of an existing P2P systems to forge an open, standards-driven solution that will allow us to leverage existing applications and equipment We combine the SIP/SIMPLE family of IETF standards for VoIP and IM with a Distributed Hash Table (DHT) P2P protocol to reach this objective. Throughout our design, we have kept existing SIP/SIMPLE clients in mind, and incorporated components to allow them to make use of the P2P approach with few or no modifications. As our design adds a self-organizing (SO) aspect to the existing SIP/SIMPLE family, we refer to our design as SOSIMPLE.

The primary contributions of this work are:

- Creating a fully-distributed, open, P2P VoIP and IM system—a Gnutella for VoIP and IM.
- Preserving many of the advantages of centralized approaches to VoIP and IM—compatibility and standardization, extensibility through hierarchical structure, and reliable and complete resource location—in a distributed design.
- Providing a mechanism to connect this system to existing SIP systems and the public telephone network.
- Offering a design that provides secure identities on an untrusted P2P overlay network.

Section II begins with a discussion of current VoIP/IM architectures and concepts. The basics of the SIP and SIMPLE protocols are presented, and existing P2P efforts in IM/VoIP are discussed. Section III presents the requirements for an open P2P approach to VoIP and IM. Section IV introduces the SOSIMPLE architecture, and discusses how our approach meets the requirements presented in section III. Finally, we conclude with some discussion of open questions and future work.

II. Existing Solutions

A. VoIP and IM Overview

Existing VoIP and IM systems need to provide capabilities for resource location, session establishment and management, and presence.

Resource location is primarily responsible for identifying and locating other users who are connected to the system so that conversations can be established. Session establishment and management allows a user to initiate

a multimedia or text session with another user or users, and manages that connection through to call completion. Presence is a term used in IM to refer to the ability of users to determine if other users are connected, and to be notified when they arrive or leave. For example, a user may have a list of individuals with whom they wish to communicate. These users are called "friends" or "buddies." Presence allows the user to see when a friend has entered, perhaps by highlighting that user on a list displayed on the user's GUI.

The majority of today's VoIP and IM systems are highly centralized. Users connect to the system using a User Agent (UA). UAs can take the form of a software application or a hardware device such as an IP telephone or mobile phone supporting IM. Typically, these devices are connected to a central sever, referred to as a proxy, softswitch, or gatekeeper. This central proxy is a more sophisticated network element than a web proxy. These proxies are typically responsible for registration of UAs, location of users within the system, and routing signaling traffic between UAs.

There are a number of VoIP protocols in wide use. Older protocols such as the ITU's H.323 place virtually all the intelligence in the central server, while SIP [9], a newer protocol developed by the IETF, is designed to push some, but not all, of the intelligence to the UAs. Users and developers have a large investment of capital and development time in the many physical or "hard" UAs in use today, and a large number of these support either H.323 or SIP.

Most commercial IM protocols, such as AOL's AIM, Microsoft's MSN Messenger, and Yahoo's Y! Messenger 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 [1, 2, 8, 10] allow a corporation or group to maintain their own internal server, but still require a centralized server. XMPP is an XML based protocol that emerged from the Jabber project. SIMPLE is a set of extensions to the SIP protocol intended to support instant messaging.

Many organizations have banned the use of commercial IM products because private conversations flow through a third party location. The overhead of an organization maintaining their own IM servers in-house often exceeds the benefits gained from IM. Individual users who wish to communicate with other VoIP users, but don't need to make calls to the public phone system are often frustrated to find they must either use a public server or maintain their own central server. We feel that freeing VoIP and IM from the need for a central server makes the technology more attractive.

B. SIP and SIMPLE

We choose to work with the SIP and SIMPLE protocols because they are open, widely implemented, and offer open-source implementations ranging from simple protocol stacks to full fledged telephone systems [7, 16]. MSN Messenger, although designed to use Microsoft's central server, is a SIP and SIMPLE client that can be configured to use systems other than Microsoft's. Additionally, a large fraction of the existing hard clients implement SIP. Before we continue, we explore briefly how SIP and SIMPLE function.

SIP and SIMPLE are text-based protocols derived from HTTP [3], therefore message types and traffic are similar to web traffic. SIP is a general protocol for establishing and controlling multimedia sessions, and is widely used for VoIP. SIP is a protocol for negotiating the session, and embeds Session Description Protocol (SDP) [4] information to specify the media parameters.

Two SIP devices can be configured to communicate directly to each other, but in all systems of greater than two UAs a central proxy is used. Messages are sent from the calling UA to the proxy, which is responsible for locating the other UA. SIP messages are passed from the proxy to the destination UA, possibly through additional intermediary proxies. Proxies are also responsible for authenticating UAs and administering user names. Proxies are located either on site or at a central service provider.

SIP messages flowing through the central proxies make up the signaling portion of a VoIP call. For efficency, the actual media packets that flow between the UAs do not pass through proxies or intermediaries. This media stream travels directly between the endpoints, usually in the form of Real Time Protocol (RTP) [11] packets. This separation of signaling and content, and the P2P nature of media exchange is analogous to existing P2P file-sharing systems that search with an overlay and exchange data directly.

SIMPLE is a set of extensions to SIP designed to carry IM traffic. The underlying technology to locate resources, establish sessions, and route messages is identical in SIP and SIMPLE. We will note differences where appropriate. Perhaps the most fundamental difference is that there is no separate media stream. The text of the messages is carried in a SIP packet, and so passes through the intermediaries that make up the signaling path.

Users are identified by an address of record, which uniquely defines an individual within a SIP system and is referred to as a SIP URI, for example sip:bryan@cs.wm.edu. Users make themselves available

to receive calls at a particular UA by sending a REGISTER message to the registrar (usually the proxy). The registrar maintains a mapping between an address of record and the location where this address is currently registered. REGISTER messages also includes an expiry time. Registrations will expire after this time unless refreshed to remove clients which are no longer valid. Clients can remove themselves immediately by registering with an expiry time of zero.

Once a client is registered, incoming message intended for the user will be resolved by the proxy and forwarded to the client. For unknown or unregistered users, local rules may be consulted to determine how to route the call, the call may be rejected, or the URI may be examined and the request proxied to a different domain to be processed. A client placing an outbound call communicates with their proxy/registrar, leaving resolution of the address to their proxy. Session establishment in SIP (for a media session) and message passing in SIMPLE can be performed without a proxy, so long as the endpoints know the addresses where each can be reached. Our design uses an element connected to the P2P network to locate resources, rather than leaving this job to a conventional proxy.

Access to and from the public phone network, sometimes referred to as the Public Switched Telephone Network (PSTN), is implemented with SIP gateways. These devices except incoming calls intended for addresses that resemble phone numbers. Typically, this is either a pay service provided by an ISP, or an organization supports their own gateways connected to their own phones. As the PSTN is a closed, for-profit system, SOSIMPLE cannot support connections directly with the P2P system. Users wishing to have access to the PSTN will still require access, usually for a fee, to a gateway. Some attempts have been made at forming distributed networks of users willing to allow access to their phone lines in exchange for access to others—trading local access between users.

The approach taken in SOSIMPLE is to leave the SIP call establishment and SIMPLE messaging mechanisms largely untouched. The registration, resource location, and subscription features are implemented with a combination of SIP/SIMPLE messages and P2P queries.

C. P2P Approaches

While this is a very new area of research, there have been a few efforts to date in this area.

1) Waste: WASTE was a P2P system for file sharing, IM, and chat conferencing, briefly released as open-source but later rescinded. Traffic is encrypted to prevent snooping, but nodes share the encryption key. Some node in the overlay must agree for a new node to join.

All messages are sent using a Gnutella-like flood mechanism. Periodic beacon broadcasts and flood poling are used by nodes to identify other nodes in the overlay. Text and group chat messages and search requests are also sent using broadcast, and pass through all nodes.

WASTE has a number of shortcomings. The IM protocol used is non-standard, so existing clients cannot connect to WASTE. All nodes must be trusted, since all WASTE nodes route and see the content of each message, and can monitor all conversations. IM and broadcast traffic explode as size increases, limiting WASTE to 50 nodes. There are no mechanisms to verify individual sender identity or ensure nodes forward messages.

2) Skype and vop2p: Skype [14] allows users to make free phone calls between internet users, and offers connection to the outside world for a fee. The system is available only in binary form—source code is not available, and the system is not standards based. A Skype user cannot communicate with a non-Skype user nor use a non-Skype endpoint.

Skype provides both VoIP and IM capabilities, and does so in a P2P fashion. Users run the Skype client to connect to the Skype overlay. The nodes organize themselves into a peer-to-peer overlay, using a supernode architecture. Supernodes are operated by the Skype corporation, which also controls user names and authorization. All end-to-end communication, both voice and IM, is encrypted for security.

In many ways, Skype is similar to what we propose, but has a number of serious limitations. As the technology is closed and non-standard, users must use Skype clients. No existing equipment can be used with Skype. Additionally, Skype controls the network in a centralized way. We see the Skype solution as analogous to the early Napster P2P systems. What is needed is a Gnutella approach.

A similar non-standard project, vop2p [17], is available open-source, but appears to be quite preliminary in nature, and seems not to have been active for over a year.

3) EarthLink SIPshare: The EarthLink R&D group has created an application called SIPshare [13], using SIP as the underlying protocol for a conventional file-sharing P2P system. Essentially, SIPshare is the opposite of SOSIMPLE. SOSIMPLE uses P2P to decentralize a SIP/SIMPLE system, while SIPshare uses SIP as a protocol for P2P.

SIPshare uses UDP for the transfer of files, analogous to the way media sessions in SIP are streamed using UDP. Messages for search, as well as requesting the

particular content requested are sent using SIP messages with specialized content.

III. Requirements for a P2P approach to VoIP and IM

We now present requirements pertaining to the P2P system, as well as general security requirements. Additionally, we discuss requirements affecting the overall architecture and configuration of the system.

A. P2P Requirements

1) Completeness: We require that P2P queries always complete and return valid responses. If a resource is present (the user is registered), we return the location of this user, and return a failure if the user is not present. Most DHT based P2P protocols satisfy this requirement.

2) Loss Intolerance: The system requires redundancy. Registrations need to be stored on several nodes, and searches must query multiple locations to ensure that node failure does not cause registrations to be lost.

3) Topological Awareness: Although it is not essential, we would topological awareness to be built into the system. Ideally, the system should attempt to group resources located in similar places in the DHT space on topologically close machines. This is particularly desirable in the case where the usernames are of an email user@domain format, as users who are within a similar name space, and presumably communicate frequently, will be located on nearby machines. Similarly, some systems may want to use geographical proximity to group machines because people make the majority of phone calls to local destinations.

B. Security Requirements

1) Identity Enforcement: We require some mechanism for verifying that users are who they say they are. We must try prevent one user from "spoofing" another by sending forged messages, or from registering as another user divert their messages. We require some way to verify a user. Identity verification is a difficult problem [6], and we have chosen to initially guarantee only that a user is the same user one has communicated with before. We feel this does not require a trusted location service, but can be met using end-to-end security principles.

C. Architecture and Configuration Requirements

1) Persistence: The overlay must be able to store arbitrary information in a replicated, distributed fashion. The information must persist as long as some node is still active. This is required to meet some of the earlier requirements, particularly being able to identify returning users, This store should also be available to clients to store configuration information. This configuration may include an encrypted version of the user's friend list, or their public key.

2) Compatibility: The system should be designed in such a way that unmodified, existing clients, whether software or physical, can be used with the SOSIMPLE system. Additionally, the system should be designed so that common, modular clients can easily be modified to connect directly to the SOSIMPLE system.

3) Hierarchical Structure: Current SIP systems allow disparate domains to be connected to each other using proxies that are aware of other domains. The design should not break this functionality—one should be able to introduce a proxy back into the P2P system and use it to interface to existing SIP systems and other SOSIMPLE systems. For example, an organization should be able to have a self-contained SOSIMPLE network, but use a proxy to connect to a larger public SOSIMPLE network or an external conventional SIP network.

IV. Design of the SOSIMPLE System

We next discuss specifically how messages are exchanged, and implementation decisions for SOSIMPLE.

A. System Architecture

Users select their usernames, we suggest a valid email address, owing to the unique nature of email addresses. It is very difficult to ensure that the user has not selected a duplicate name. We discuss some approaches to the *identity enforcement* requirement in the security section. Users should search for a username before using it, and clients should not allow a user to use a name that already exists in the overlay.

To meet our requirements for *compatibility* and *hierarchical structure*, SOSIMPLE allows for the P2P aspect of the system to be inserted at one of several locations in the system.

UAs can be modified to directly join the overlay. There are few changes needed. Instead of always using the outbound proxy for new calls, these UAs instead consult the overlay to locate resources, and connect directly. UAs joining directly must support SIP register messages and store mappings between address of records and contacts, and maintain expirations for them. SIP plug-ins for gaim already exist, and can be easily adapted for SOSIMPLE.

Several alternatives exist to allow for *compatibility* with existing clients. An adaptor can be constructed to act as a "mini-proxy"—receiving queries from an unmodified UA and translating these into search requests in

the overlay. The adaptor then returns a redirect message to the client with the location of the resource. To a UA, this adaptor will appear to behave like a conventional SIP proxy. This adaptor must also support registrations.

The behavior discussed above could also be added to a full SIP proxy. The overlay would then be one of many alternatives queried when resolving a SIP URI. Such proxies fulfill our requirement for *hierarchical structure.*

B. Registration, Location and Presence

Resource location is modified to remove the central proxy and meet several of our requirements. The registration process is modified by changing where registration messages are sent. The UA constructs a SIP REGISTER message containing their contact information. The endpoint hashes their username, and sends the SIP message embedded in a P2P message using the overlay. Upon arriving, the message is extracted and a reply is sent.

To meet our requirements for *loss intolerance* and *persistence*, the REGISTER is also sent to the k successor nodes in the overlay. Each node now serves the function of registrar, and knows where some users can be contacted. New nodes joining the system contact their neighbors and replicate the registrations and expiration times.

Figure 1 illustrates an example of an endpoint joining the SOSIMPLE network. Alice starts a SOSIMPLE enabled UA at 1.1.1.1:5060 and connects to the overlay. Assume her username, alice@alice.com hashes to node a, and the k=2 successor nodes are nodes b and c. Alice embeds a SIP REGISTER in a P2P message and transmits it to each of these nodes using the overlay. These 3 nodes now store a mapping from alice@alice.com to 1.1.1.1:5060.

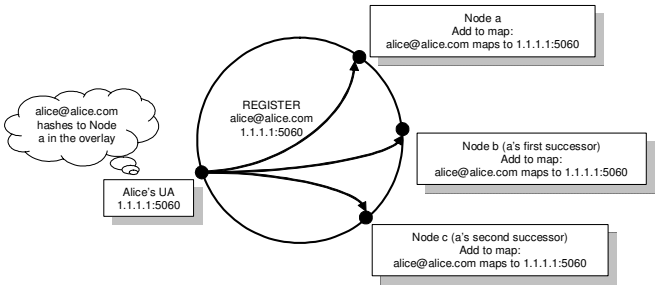

Fig. 1. Registration Example

Message routing is similarly modified to use the overlay. Rather than sending the SIP messages to a proxy, the UA will hash the destination user name. The message is embedded in a P2P message and sent to this location using the overlay.

Returning to our example, assume Bob wishes to send a SIMPLE MESSAGE to Alice. Bob hashes Alice's ID and sends the message using the overlay. A SIP redirect, including Alice's address, is embedded in a P2P message and sent back to Bob using the overlay. Bob sends the message (and future messages) to that address using a direct SIP message. Alice's responses are also sent directly using standard SIP messages. Meeting our *completeness* requirement, if Alice was not registered, a SIP 404 Not Found would have been embedded in the P2P response Bob received instead. Similarly, to satisfy the *loss intolerance* requirement, Bob can contact one of the k successors in the event node a fails.

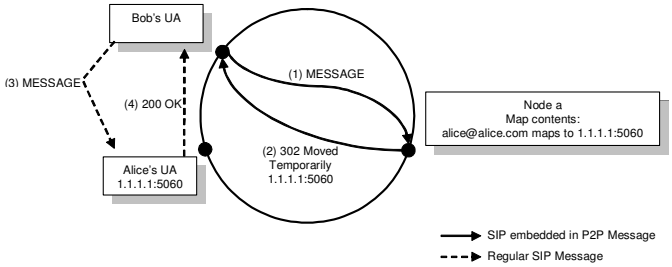

Fig. 2. Message Example

To improve performance, UAs should cache contacts and attempt to use these in the future. If subsequent messages fail (no response is received), the UA should hash in the overlay again.

C. Transmitting Content

As discussed and shown in the examples above, IM traffic is sent between the nodes at the SIP/SIMPLE level using the MESSAGE method. The overlay need only be involved when a message is first sent or if a cached contact fails. Since SIMPLE messages are not correlated into a session, there is no need to maintain state for IM other than the cache of user contacts.

Advanced instant messaging, such as sending simultaneously to multiple clients, has not yet been explored. Traditionally, this has been handled at the proxy level in SIMPLE, but could likely be handled here by maintaining multiple entries in the cache for a given contact.

Voice conversations are established in essentially the same way. The initial signaling messages will be embedded and use the overlay. Media flows directly between the UAs, just as in an ordinary SIP call. For these calls, the UA must maintain some state for the session. The call can last for a period of time, after which a BYE message is sent. It is possible that the user has switched to a different UA and their contact has changed since the call began. We need to send the BYE to the original

client. In practice, this problem is easy to solve. Most SIP stacks today maintain this state for the user, and the size of the state required is very small.

D. Security

SIP/SIMPLE offers private-key mechanisms for authentication between UAs and proxies and for end-to-end encryption [5]. We instead require a public key mechanism.

The public portion of a user's key is stored in the overlay. When a user attempts to join the overlay, a cryptographic challenge using this PKI system is used to verify that the user is the same one we saw in the past. This doesn't guarantee the user is who they say they are—only that they are the same user we saw before. All messages between nodes can be similarly protected. In this way, we meet or *identity enforcement* requirement. By retaining this information as data in the overlay, the *persistence* requirement is also satisfied.

Mechanisms using public key authentication are not standard SIP. Therefore, this must be implemented between our modified nodes, and public key methods or no security must still be used between the adaptors/proxies and non-modified UAs. There are efforts underway within the IETF to standardize end-to-end authentication and these should be used when they emerge.

E. Implementation

We have chosen to use the Chord [15] P2P library for our initial implementation. For a SIP stack, we have opted to use the ReSIProcate project stack, a fully compliant SIP/SIMPLE stack that incorporates most of the new features introduced into SIP. Additionally, ReSIProcate is one of the few SIP stacks that includes fully tested SIMPLE support. There is a ReSIProcate based plug-in for the GAIM client as well as an all new client called SIPImp [12].

At present, several of the components of SOSIMPLE have been completed, and we are in the process of completing an implementation. We expect to have a working implementation completed by January, 2005.

V. Future Work and Conclusion

Changes to our P2P mechanism could include a broadcast approach to locate the initial node in the overlay. At present, we require the user locate the first node through an out of band mechanism. While this is a desirable improvement, it might introduce security and fragmentation risks. Additionally, we would like to explore using an underlying search mechanism similar to EarthLink's SIPshare as the P2P component of SOSIMPLE, leading to an all SIP approach.

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, we feel SOSIMPLE meets the need for a more distributed communication system.

References

[1] Campbell, B., Mahy, R., and Jennings, C. draft-ietf-simple-message-sessions-08.txt. http://www.ietf.org/internet-drafts/draft-ietf-simple-message-sessions-%08.txt, Aug. 2004.

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

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

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

[5] Jennings, C., Peterson, J., and Watson, M. 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.

[6] Peterson, J. draft-peterson-message-identity-00.txt. http://www.ietf.org/internet-drafts/draft-peterson-message-identity-00.%txt, Oct. 2004.

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

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

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

[10] Saint-Andre, P. RFC3920-3923 – IETF XMPP RFCs . http://www.ietf.org, Oct. 2004.

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

[12] SIPImp Project. http://sourceforge.net/projects/sipimp/.

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

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

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

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

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