Innovations in Peer-to-Peer Communications

David A. Bryan
Bruce B. Lowekamp, advisor
Department of Computer Science
College of William and Mary

Abstract

In the last 2 years, Peer-to-Peer (P2P) communications has gone from a completely new topic to a popular mechanism for use in personal communications. We proposed the first standards-based P2P communications system, SOSIMPLE, and have been focused on addressing a number of issues for such systems. In particular, we have been exploring issues that have limited implementation and deployment of these systems, including security, distributed storage of offline messages, protocol standardization, NAT traversal, and defining P2P routing mechanisms optimized for communications, rather than file sharing. In this paper, we outline some of these issues, discuss possible solutions to address them, and discuss work in these areas to date.

1 Introduction

P2P technology has become an important part of communications technology. Skype[12] is a very popular program for communications, both instant message (IM) and voice, and is partially P2P in nature. We designed SOSIMPLE[2] as a fully standards compliant P2P communications system. The SOSIMPLE protocol[1] has been fully specified in an Internet Draft (ID) submitted to the IETF for consideration for standardization, and work toward making either SOSIMPLE or a similar protocol a standard is ongoing.

In this paper, we explore a number of issues related to deployment of such systems: security, storage of offline messages, standardization, NAT traversal, and optimization of P2P routing mechanisms for communications, rather than file sharing.

2 Security

2.1 Overlay and Routing Integrity

Security is an important consideration in P2P systems. These systems are vulnerable to attacks, just as client server systems are, but the attack mechanisms are often quite different. In addition to attacks against the server and DoS attacks, corrupt nodes joining and manipulating the overlay can be a risk for P2P systems.

In a P2P overlay network, peers are responsible for routing call information on behalf of other nodes. In the case of systems based on Chord[13] such as SOSIMPLE, the nodes are organized in a structured mathematical way. Nodes are organized in the form of a ring, with a node's location in the ring determined by a Node ID. Information is stored in the overlay by the nodes. Each piece of information to be stored is associated with a particular keyword, and the keyword is hashed to produce a key. The node with the Node ID nearest to the hashed key is responsible for storing the information.

One possible attack against a P2P system is called a Sybil attack[4]. In such an attack, a malicious attacker creates many identities, hoping that at least one will be assigned a Node ID that will map near the information they wish to attack. If they can become the node responsible for storing that information, they can deny access to or corrupt the information. In the case of communications systems, the information stored is usually the data needed to locate a particular user. Compromising this information can result in misdirection of calls or denial of service to a particular node, preventing that user from receiving calls.

Chord has partially dealt with this problem by hashing the IP address of a node to determine the Node ID. This limits the number of multiple identities that one can achieve to the number of IP addresses one has available. In practice, we have found this approach to be inadequate. Many users today are located behind network address translation devices (NATs), which allow many physical machines to share one IP address. This works well for many networks, and is commonly deployed today (although it introduces problems we will discuss in a later section), but precludes the one IP to one Node ID mapping used by Chord.

Our solution to this problem is to replace the last 5 places of the Node ID with the port number of the client. This allows multiple clients to be behind a NAT, each having a unique Node ID. A single user wishing to mount a Sybil attack may now produce many nodes, however these nodes will all be contiguous, meaning the portion of the ring (and thus the registrations they are responsible for storing) is reduced.

With this change, the approach to redundancy taken by Chord must be revisited. Chord places redundant storage of information on sequential nodes. That is, in addition to storing on the nearest node, some number of the nearest node's successors also store the information to improve reliability. This approach is less desirable with our solution, since sequential nodes are likely to be owned by the same user or group of users. Instead, we have adopted a strategy using a replica number. When data is hashed to be stored, the keyword is hashed to form the primary location, and the keyword concatenated with one or more replica numbers is hashed to find the redundant locations. These replica numbers are widely known to all nodes in the overlay, but adding them to the hashed value ensures that the replicas are distributed around the ring, limiting the ability of one user to attack the data and its replicas. A strategy of comparing the results returned from both the primary location and the replicas can also be used to detect compromised data stored in the overlay. If the replica numbers are kept private to the authorized nodes, this can serve as a very primitive access control mechanism for the network.

2.2 Using Limited Central Servers

One problem remains. A user is still able to create as many identities as they wish. Our strategy of concatenating the port to the Node ID limits the damage that can be done in a traditional IPv4 NAT implementation, but in an IPv6 scenario, the user may have access to many IP addresses. Similarly, in the case of an attack from a very large enterprise or government facility in IPv4, there may be many available addresses, and our method may still prove inadequate. Because of this problem, we have proposed that a central server, used only to issue public key certificates, should be employed.

The server is contacted only the very first time a user wishes to join, and issues a certificate to the new user. The certificate may have a minimal cost associated with it to prevent the user from repeatedly requesting new certificates. The entire certificate chain is stored in the overlay, ensuring that other nodes can verify the authenticity of the key. Because the central server limits the number of keys one can obtain, this can severely limit the risk of Sybil attacks in such a system. This server is only contacted when a user first joins. Subsequent actions require only the certificate chain, stored in the overlay, but do not require contacting the central certificate server.

These central authorities serve to address another very difficult problem in such a system. How does one obtain a unique username and ensure it remains theirs? Since resources, in this case information about where a user is located, are stored based on keyword, we need to be able to determine if the entity storing a location is really authorized and associated with that username. Additionally, how can one ensure that the user reached is the same user that was reached before?

Solving the repeated identity problem, that is, ensuring that repeated conversations with a user are with the same user each time, is relatively simple to address. If the user creates a public key certificate, and stores the public portion in the overlay, a caller can verify that the certificate did not change and that the person called has the private portion of the key, ensuring they are the same party as before.

Preventing collisions where users attempt to use the same username (either intentionally or accidently) is far more difficult. Here again, having a central server eliminates this concern. The server can issue a certificate for a given name to only one individual, and preclude others from obtaining such a certificate. If all operations to add information to the overlay require authentication with this certificate, it will be impossible to have name space collisions or attacks.

2.3 Securing Conversations

If a system is built on top of an overlay with reliable routing, securing conversations become much easier. The centrally issued certificate for each user can also be used to encrypt the media or text messages being exchanged.

The public portion of the user's key can be stored in the overlay as would any other item of information. Before contacting a user, this certificate can be retrieved and used to encode information flowing between parties. Because the entire certificate chain is stored, and because all nodes are aware of the root certificate, the authenticity of the certificate can be verified.

Particularly in systems using media relays, as discussed in the NAT traversal section of this document, we recommend all media be encrypted, so that only the user for whom the communications is intended can decrypt and read the messages.

3 Offline Message Storage

3.1 Using Centralized Security Servers

One important feature for a communications system is storing offline information, either voice mail messages in the case of a media based system, or offline text messages for an IM system. We explored this in a paper on using strong central security servers to secure distributed file storage[3]. The servers proposed in this section differ from those presented above, in that they are contacted for each transaction.

In the approach presented in the paper, voice mail messages and away text messages are stored in the overlay. A peer is selected to store the information using a heuristic to determine the best possible node for storage (this approach differs from selecting a location using a hash). The sender generates an encryption key and encrypts the message, then stores the message on the selected node. The sender then transmits to the central server the location of the node storing the message, the encryption key and the name of the recipient. This information is all encrypted using a pre-shared key between the sender and the central server.

When the recipient logs in to the system, they authenticate with the central server. The server checks if any messages are waiting, and if so, passes the location as well as the key to decrypt these messages back to the joining node over an encrypted channel.

Since the messages are encrypted when stored, and all communication with the central server is encrypted during transit, this mechanism preserves voice mail messages integrity, so long as the central security server is not compromised.

3.2 Using Certificates for Offline Storage

The system for offline message security discussed above has a shortcoming. It requires a strong security server be available at all times. In the absence of a strong security server, but with a join-time certificate server as described earlier in the Overlay and Routing Integrity Section, one can also use the PKI to secure offline messages.

In such a system, each message is associated with a recipient. The sender hashes the recipient name to determine the node responsible for storing the offline message, and uses the public portion of that user's certificate (also stored in the overlay) to encrypt the message.

When the user logs on, they hash their own username to determine where their registration is stored, and contact that node. When connecting, the node providing service to the joining node checks for any waiting voice mail or IM messages, and notifies the joining node of any that are available. The user can then retrieve the messages and decrypt them. Other users would be unable to replay the contents of the messages, since they are encrypted with the user's encryption key.

3.3 Open Issues for Offline Storage

There are a number of open issues for offline storage. While it is straightforward to ensure that messages cannot be read by unauthorized parties, determining when messages can be deleted from the node that is storing the message can be difficult. If the node allows any node to delete the messages, users can delete other user's messages. If the node verifies the identity of the person requesting the message, it can "authorize" deletion, but problems remain.

A malicious node could sending large or a large number of messages with invalid recipients, causing other nodes to store the information, leading to a DoS attack. In the hash based scheme, nodes responsible for popular parties receiving many offline messages may have high loads. These problems remain to be solved.

4 Standardization

In telecommunications, having protocols standardized is critical to acceptance. Imagine a world where one phone was unable to communicate with another. Because of this, groups such as the IETF and ITU, which standardize telephony protocols, are critical to deploying any new telecommunications technology.

We have worked diligently to advance a standard for P2P SIP communication through the IETF. We are currently advocating SOSIMPLE as a possible standard, but there are a number of different protocol options that have emerged, all of which offer some advantages.

One key question is on the issue of SIP over P2P, or P2P SIP. In SOSIMPLE, we have proposed using SIP to convey the information needed to transport P2P messages such as maintenance of the overlay, now referred to as a P2P SIP mechanism. In contrast, the advocates of SIP over P2P instead advocate using an existing library for P2P, such as the widely publicized Open DHT project[8] as the location mechanism and then using SIP only to establish communications sessions between nodes locating using this mechanism. Both approaches have traction. The SIP over P2P advocates have a major procedural issue to overcome in that most standards bodies standardize protocols, rather than operations or interfaces, and are likely to be unwilling to standardize a mechanism that depends on an implementation, rather than a protocol as the underpinning.

Another critical issue that is currently being debated in the standards groups is the appropriate DHT to be standardized. The SOSIMPLE protocol specifies Chord, but many other protocols have been proposed, most notably Kademlia[7]. Modifying or creating a new DHT that is optimized for communications is also being considered, and is discussed in a later section. Determining one or more DHT algorithms to use is likely to be a critical aspect of any standardization effort.

5 NAT Traversal

An early issue raised in discussions about P2P communications is how NAT traversal will be implemented in the systems. Currently, two major proposals seem to be leading the debate.

One is to use existing methods defined by SIP and the related family of protocols to enable NAT traversals. These include STUN, TURN, and ICE[11;10;9]. These protocols have been widely advocated in the P2P literature[5] as a general mechanism for P2P NAT traversal, thus their use in an area more closely related to their intended purpose, use in communications, seems reasonable. Each node participating would implement one or more of these protocols, allowing the nodes to communicate in the presence of NATs.

Another proposal enjoying wide support is to use ordinary peers and super peers to form such a network. The ordinary peers could be behind NATs, and do not participate directly in the overlay. Instead, only super peers participate, and a direct connection is formed between ordinary peers and super peers. One super peer may participate in the overlay on the behalf of many ordinary peers. Connecting directly between a client and server through a NAT is a well understood problem, it is only when one needs to contact many arbitrary nodes behind NATs that the problem arises. Having ordinary peers communicate directly with the super peers eliminates this problem. The biggest objection to such a system is that those peers that are not behind the NATs—the super peers, are forced to shoulder a disproportionate portion of the traffic or storage responsibility for the network.

In either approach, other nodes may occasionally be called upon to relay traffic, either signaling or the more bandwidth intensive media. This leads to suggestions that all media be encrypted end to end, prohibiting intermediate nodes from intercepting the messages.

In general, we feel this problem has gone from a research problem when this architcture was first proposed to a simple engineering exercise. It is no longer a matter of if a reasonable solution can be found, but which one to select.

6 Optimal Routing for Communications

An open area for research that has been extensively discussed but largely unexplored is defining routing techniques that are optimal for communications systems, rather than file sharing.

File sharing systems, the bulk of P2P systems to date, have very different operational constraints than communications systems. In file sharing systems, many copies of a particular resource (file) often exist. A user searching for a particular music file generally does not care which particular instance they find—a copy of a song is good enough. In contrast, in communications systems there is typically only one valid user to be reached for a particular call. Systems must be deterministic in nature, reporting either success or failure to locate a user in a communications system. In a file sharing system this is not always the case. A user may search only a portion of the network, and failing to find a particular file within that sub section of the network, report failure. Such an approach is unacceptable for a communications network.

Another critical difference between file sharing and communications systems is anonymization. In file sharing systems, the action being performed is frequently illegal, since the users are generally involved in copyright violation. Anonymity is a desirable property. If the source of a file can be obfuscated, this is almost universally considered a positive.

In communications systems, the decision is not so clear. Some systems have been proposed where anonymity is a desirable property. Take for example a communications system intended to be used by citizens in an oppressive regime that limits access to the Internet using firewalls or similar mechanisms. Anonymity in such systems is highly desirable, as it can allow political dissidents to converse. On the other hand, this can make lawful intercept difficult.

In contrast, in a traditional telephony system, anonymity is usually considered a negative. Users do not want to receive calls from unknown parties, and certainly do not want to place them to unknown parties! It is clear that this metric differs in communications systems relative to file sharing systems, although different approaches can be considered optimal for communications, depending on the purpose of the system.

Finally, communications systems frequently have an additional information source that file sharing systems cannot rely on. IM or voice communications systems have either a list of buddies or a phone book of contacts. Routing techniques that capitalize on this additional information can greatly increase the efficiency of searches by searching for users via buddies, rather than the network as a whole. Since many people have many friends with similar interests, statistically speaking people they want to contact are more likely to be buddies of their buddies than buddies of random nodes. The creators of the SPROUT[6] first proposed such a system, and the authors are working to extend this work to apply to communications systems.

7 Conclusion

Many advances have occurred in the field of P2P communications in recent months and years. We feel these advances have reduced a number of previously difficult problems to simple implementation exercises, but that a number of open, difficult, problems remain.

As standardization efforts move forward, we feel that more of these problems will be resolved, and P2P systems will see increasingly wide deployment.

8 Acknowledgments

This work was supported by the Virginia Space Grant Consortium (VSGC), through their support of David A. Bryan through a Graduate Student Fellowship. Additionally, this work is sponsored by Cisco Systems, Inc., through their University Research Program (URP).

Thanks to Cullen Jennings and Feng Cao at Cisco, who have contributed to the work presented in this document. Thanks also to the many people participating in the IETF standardization efforts on this technology—their comments have been invaluable in this work.

References

[1] D. A. Bryan and C. Jennings. draft-bryan-sipping-p2p-02. `http://www.ietf.org/internet-drafts/draft-bryan-sipping-p2p-00.txt`, Mar. 2006.

[2] D. A. Bryan, B. B. Lowekamp, and C. Jennings. Sosimple: A serverless, standards-based, p2p sip communication system. In *Proceedings of the 2005 International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications (AAA-IDEA 2005)*. IEEE, 2005.

[3] F. Cao, D. A. Bryan, and B. B. Lowekamp. Providing secure services in peer-to-peer communications networks with central security servers. In *Proceedings of the 2006 International Conference on Internet and Web Applications and Services (ICIW'06)*, 2006.

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

[5] S. Guha, Y. Takeda, and P. Francis. Nutss: A sip-based approach to udp and tcp network connectivity, Aug 2004.

[6] 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.

[7] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric, 2002.

[8] Open DHT. `http://www.opendht.org`.

[9] J. Rosenberg. draft-rosenberg-mmusic-ice-08. `http://www.ietf.org/internet-drafts/draft-ietf-mmusic-ice-06.txt`, Mar. 2006.

[10] J. Rosenberg, R. Mahy, and C. Huitema. draft-ietf-behave-turn-00. `http://www.ietf.org/internet-drafts/draft-ietf-behave-turn-00.txt`, Feb. 2006.

[11] 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.

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

[13] I. Stoica, R. Morris, D. Karger, M. Kaashock, , F. Dabek, and H. Balakrishman. Chord: A scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on Neworking*, 2003.