Peer-to-Peer Internet Telephony using SIP

Kundan Singh and Henning Schulzrinne
Department of Computer Science, Columbia University
{kns10,hgs}@cs.columbia.edu

Abstract

P2P systems inherently have high scalability, robustness and fault tolerance because there is no centralized server and the network self-organizes itself. This is achieved at the cost of higher latency for locating the resources of interest in the P2P overlay network. Internet telephony can be viewed as an application of P2P architecture where the participants form a self-organizing P2P overlay network to locate and communicate with other participants. We propose a pure P2P architecture for the Session Initiation Protocol (SIP)-based IP telephony systems. Our P2P-SIP architecture supports basic user registration and call setup as well as advanced services such as offline message delivery, voice/video mails and multi-party conferencing. We also provide an overview of practical challenges for P2P-SIP such as firewall, Network Address Translator (NAT) traversal and security.

(**Keywords:** System design, peer-to-peer, high availability, scalability, Internet telephony, SIP.)

Contents

1 Introduction

Existing Internet telephony client-server architecture based on IETF's Session Initiation Protocol (SIP [1, 2]) or ITU-T recommendation H.323 [3] typically employ a registration server for every domain. The user agents (or IP phones) of the users in the domain register their IP addresses with the server so that the other users can reach them. Scalability and reliability of such server-based systems are achieved using traditional redundancy and failover methods such as DNS, IP address takeover, MAC address takeover or application layer switches [4, 5]. The majority of the system cost is in maintenance and configuration, typically by a dedicated system administrator in the domain. It also means that quickly setting up the system in a small environment (e.g., for emergency communications or at a conference) is not easy.

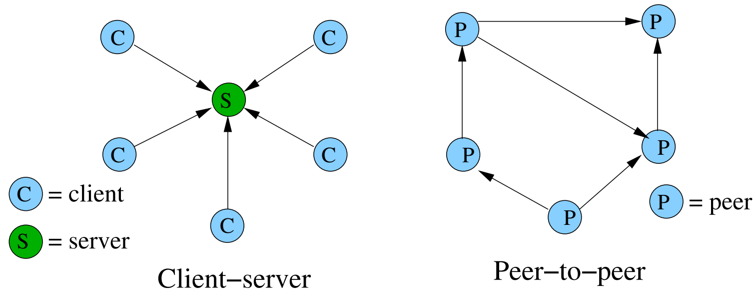

Figure 1: Client-server vs peer-to-peer distributed systems

On the other hand, peer-to-peer (P2P) systems [6] are inherently scalable and reliable because of the lack of a single point of failure. P2P systems, in the purest form, have no concept of servers as shown in Fig. 1. All participants are peers and communicate in distributed, potentially untrusted environment, to achieve a certain objective such as locating music files or users. Some file transfer systems with central index server such as the old Napster are hybrid P2P systems. However, for the purpose of this paper we use the definition of "pure" P2P systems that do not have any centralized control. Accordingly, existing SIP and H.323-based systems that have centralized user location lookup but end-to-end media transport are *not* P2P.

In this paper, we propose a P2P architecture for IP telephony using SIP. We identify differences between rendezvous systems such as Internet telephony and traditional file sharing systems in P2P context. We analyze various design alternatives and present a detailed design for our P2P-SIP endpoint using Chord [7] as the underlying distributed hash table (DHT). Our novel hybrid architecture allows both traditional SIP telephony as well as user lookup on P2P network if the local domain does not have a SIP server. We show that SIP can be used to implement various DHT functions in P2P-SIP such as peer discovery, user registration, node failure detection, user location and call setup by replacing DNS [8] with P2P for the next hop lookup in SIP. To our knowledge, our work is the first such attempt to apply P2P concepts to SIP-based systems.

Our architecture benefits from scalability and reliability offered by P2P. We believe that file sharing systems such as Kazaa [9] and Gnutella [10] are widely popular because they provide free music and video content without requiring maintenance of a content server, and they automatically detect NAT and firewall settings without any user intervention. Similarly, P2P-SIP has additional advantages over existing Internet telephony acrhitectures as follows:

No maintenance or configuration: The system works out-of-the-box without requiring any tedious server installation, including NAT and firewall configuration. Our work extends the goals of the IETF Zeroconf [11] Working Group to multimedia communication and collaboration systems.

Interoperability: Unlike other P2P systems such as Skype [12], our architecture uses SIP messages for communicating with other peers. This readily interworks with any existing IP telephony infrastructure such as SIP-PSTN gateways.

These advantages come at the cost of increased *resource lookup delay*, security threats and reliability issues. Unlike $O(1)$ lookup cost in a classical client-server based systems, the P2P lookup cost can be much higher (e.g., Chord [7] lookup latency is $O(logN)$ where N is the number of peer nodes in the system). A distributed P2P architecture makes the system more prone to *security* issues such as trust (privacy: how much information does the untrusted peer need to know about me? and confidentiality: what if the peer who knows my information misuses it?) and DoS attacks (were those thousands of call routing requests that I received, legitimate?). A reliable framework for authentication without centralized elements is a challenge. In addition, we may lose some of the traditional IP telephony services. For example, some of the programmable call routing techniques such as SIP-CGI available for SIP telephony can not work in the P2P-SIP system as we do not want to run potentially malicious script uploaded by some peer on our machines. Finally, the *reliability* of the IP telephony system is very important. People are unlikely to use it if the probability of successful call setup is not at par with that in the regular telephone network. The telephone switches are designed to be 99.999% available.

In addition to the basic call setup and registration, we also outline advanced services such as "missed call" notifications and multi-party conferencing in P2P-SIP. Most existing systems are studied and analyzed for file sharing type of applications. Rendezvous systems such as multi-party conferencing need to deal with some slightly different issues as we show in this paper.

We provide background on P2P and SIP related work in Section 2. Section 3 lists the goals for a P2P architecture for IP telephony. Section 4 presents various design alternatives. Section 5 gives an overview of the P2P-SIP architecture, user registration and call setup. Section 6 describes advanced services in P2P-SIP. Section 7 analyzes various security threats and their proposed solutions. Section 8 predicts performance of the system in terms of scalability, reliability and call setup latency. Finally, Section 9 presents our conclusions and future directions.

2 Background and related work

A number of studies have been done to analyze and understand different P2P systems [6, 13].

2.1 P2P systems: structured vs unstructured

P2P systems can be broadly classified into unstructured (such as Kazaa and Gnutella with no structure of how the nodes store the files) and structured (such as those using DHTs). The unstructured systems have concentrated on practical problems such as NAT and firewall traversal but search is typically performed by flooding the request to all the neighboring peers. On the other hand, structured systems such as Chord [7], Content Addressable Network (CAN) [14] and Pastry [15] focus on optimizing the P2P overlay for lookup latency and join or leave maintenance cost [16] instead of using inefficient blind search by flooding. NAT traversal has not been explored in detail for structured P2P.

We use an underlying DHT such as Chord in our architecture for lookup. Chord has a ring-based topology where each node stores at most $LogN$ entries (or state) in its *finger table* to point to other peers. Lookup is done in $O(LogN)$ time. The *iterative* and *recursive* lookup styles in Chord [7] directly map to the *redirect* and *proxy* behavior, respectively, in SIP. Research in DHT is complementary to our work, since our architecture can use new innovations or optimizations in the underlying DHT.

2.2 Skype and related systems

Skype [12] is a free P2P application based on Kazaa [9] architecture that allows making calls over the Internet to any other Skype user. Skype has the following problems:

1. The protocol is proprietary unlike open standards such as SIP.

2. It provides a single service which is making calls or instant messages and not a architecture for new services.

3. Most importantly, it has centralized elements for login authentication [**?**] which means that if this element fails the system may not work.

In a way, the Skype architecture is no different from the classical SIP telephony architecture, except that the Global Index Server assigns a *super-node* for a new joining node. The super-node, similar to the SIP registrar, proxy and presence server, maintains the presence information for this node, and locates other users by communicating with other super-nodes. A node that has enough capacity and availability can become a super-node. We believe that the lookup is based on some variation of flooding, similar to Kazaa, instead of using the more efficient DHT-based lookup.

The main advantage of Skype is that it implements the equivalent of STUN and TURN servers in the node itself to handle NAT [17], unlike explicit server configuration in existing SIP applications.

People have developed various P2P multimedia communication applications such as flooding-based text chat [18] or peer-to-peer collaboration systems [19, 20] for small groups with centralized components.

2.3 SIP-based telephony

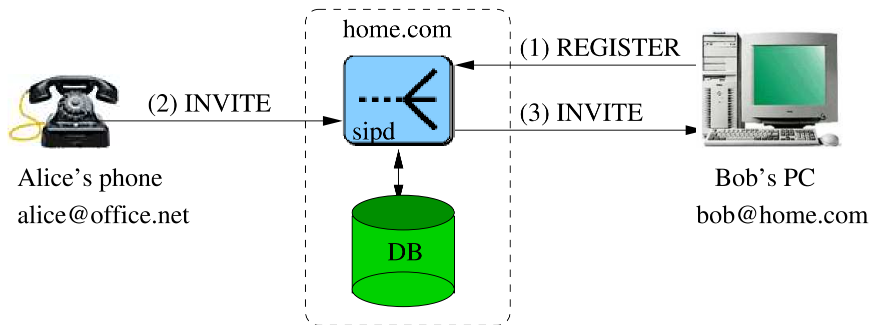

Figure 2: SIP call flow using proxy servers

Unlike P2P, existing SIP-based telephony has client-server architecture. SIP [2] is a signaling protocol for Internet conferencing, telephony, presence, event notification and instant messaging. As shown in Fig. 2, when a user, Bob, starts the SIP client on his PC, IP-phone or hand-held device, the client registers with the SIP server indicating the IP address of the device. The SIP server stores the mapping between the identifier *bob@home.com* and the IP address. When another user, Alice, makes a call or sends instant message for *bob@home.com* to the server in *home.com* domain, the server proxies the request to the current device of Bob. Further details of the protocol can be found in [2, 1]

SIP-based IP telephony can be treated as a P2P system with static set of super-nodes (SIP servers) where the lookup is based on DNS instead of a hash key. However, using a pure P2P architecture instead of static set of SIP servers improves the reliability and allows the system to dynamically adapt to node failures.

2.4 Difference between IP telephony and file sharing

There are three broad categories of P2P applications: file sharing, directory service and rendezvous systems. A rendezvous or meeting system initiates communication with users or groups of users and actively synchronizes different activities. For example, a user can send the SIP INVITE message to many potentially nomadic users to invite them in a conference by creating one-to-many binding. Table 1 summarizes the similarity and differences among these types. In

Table 1: Different applications of P2P

Properties/Types	File sharing	directory	rendezvous systems
Data storage	Yes	No	No
Caching	Yes	Yes	No
Delay sensitive	No	No	Yes
Reliability	Multiple copies		Does not work

particular, for rendezvous systems such as Internet conferencing, data storage is not an issue. A single P2P-SIP node can handle many more requests than a file sharing node due to low data volume. Caching of location information is not useful because compared to the file access pattern which follows the zipf distribution, the call access pattern is more uniformly distributed. Moreover, most residential users are likely to get new DHCP IP address every time they connect to the Internet making the cache entry for this user location stale. The file sharing and directory lookup-based systems can tolerate high lookup latency due to the fact that the user does not need to actively wait for the file to download, and the actual file download time tends to be larger than the lookup latency. On the other hand, an IP telephony caller actively waits for the phone on the other side to ring. For file sharing applications, multiple almost-exact copies of a popular file may be available (e.g., independently ripped by different peers). So node reliability does not matter. On the other hand, in the case of IP telephony, we want to talk to the right person, and not some similar person!

3 Design goals

Based on the review of existing P2P systems such as Skype [12] and Chord [7], we propose the following goals for our P2P-SIP telephony architecture.

Zero configuration: The system should be able to automatically configure itself [11], e.g., by detecting NAT and firewall settings, discovering neighboring peers and performing initial registration.

Heterogeneous nodes: It should be able to adapt to available resources and distinguish between peers with different capacity and availability constraints. This favors the distinction between nodes and super-nodes as in Kazaa.

Efficient lookup: Blind search based on flooding is inefficient [13]. The system should use an underlying DHT to optimize lookup. We choose Chord as the underlying DHT for our system because of its robustness and efficiency in the case of concurrent node joins and leaves [16].

Advanced services: It should support advanced telephony services such as offline voice messaging, multi-party conferencing, call transfer and call forwarding as well as advanced Internet services such as presence and instant messaging.

Interoperability: It should easily integrate with existing protocols and IP telephony infrastructure. We choose SIP [2] as the signaling protocol for interoperability.

Besides these explicit goals, there are some implicit scalability and reliability benefits in the P2P-SIP architecture compared to the client-server SIP architecture.

4 Design alternatives

In this section, we evaluate different design alternatives for user lookup and registration to meet above goals. We start with the simple call setup in SIP and incrementally extend it for reliability using P2P architecture.

4.1 Replicate registrations vs search on call setup

Going back to the simple call setup example of Fig. 2, the single server can become the bottleneck for reliability. It can be improved by having multiple redundant servers. There are two alternatives:

1. replicate all user location information to all the servers, as shown in Fig 3, or
2. search for the correct server holding the destination user location when a new incoming call is received, as shown in Fig 4.

In the first case, although Fig. 3 shows multiple registrations, one can alternatively do database replication to ensure consistent user records among multiple server databases in the cluster. In the second case, either the caller retries all the servers in some order or the first contacted server can do the search.

The disadvantage of the first approach is that it involves synchronization overhead for each registration. There is a danger of stale user location record on some servers for a brief interval after the update is done but before all the servers get the updated registration. With registration refreshes every hour per user, this architecture may limit the total number of users supported by the system as the synchronization traffic will soon become a bottleneck. In the second case, the call setup latency is higher due to the sequential search steps. A parallel search will increase the bandwidth requirement. Both the approaches of Fig. 3 and 4 tend to fail when the number of servers is very large. The first approach and its variations are described in [4]. In this paper we focus on the second case.

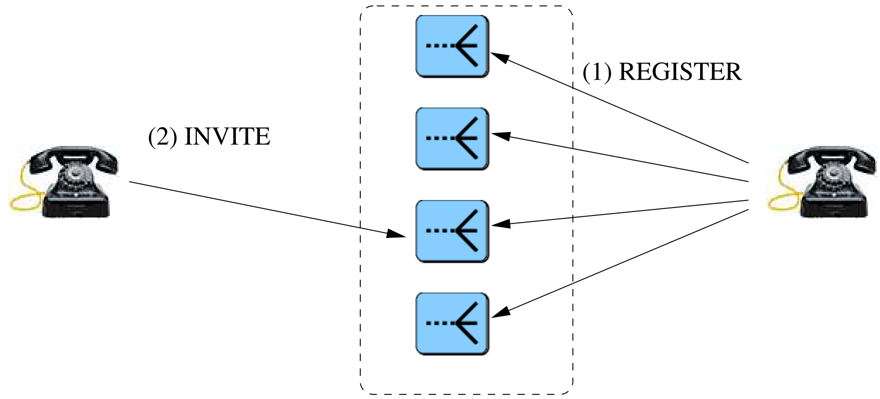

Figure 3: Design A: all servers store all user records on registration

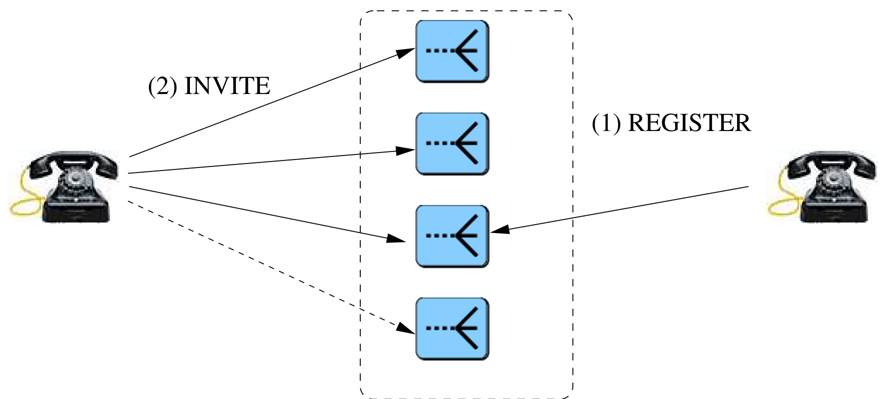

Figure 4: Design B: search for the server on call setup

4.2 What nodes form the DHT?

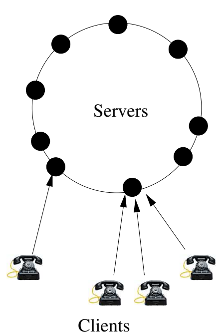

Figure 5: Option 1: Only servers in DHT

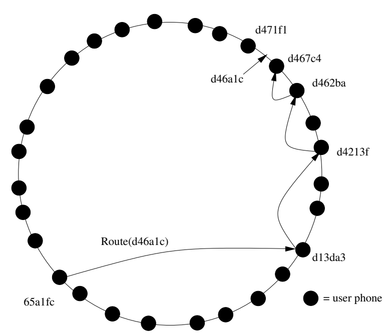

Figure 6: Option 2: Complete P2P overlay

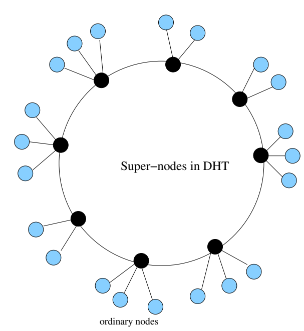

Figure 7: Option 3: Intermediate model

We can achieve some combination of the two designs using a DHT such as Chord [7] so that the registration is done on only $O(LogN)$ servers instead of all the N servers, and the search is done for only $O(LogN)$ servers instead of all the N servers. There can be three alternative designs for using a DHT. On one extreme, we can limit the DHT to the server farm as shown in Fig. 5. In this case, each client or phone connects to one of the servers. The servers implement a DHT or a scalable distributed data structure [21] to locate the correct user record. We assume ring-based Chord as the underlying DHT. The architecture is still client-server. The client needs to discover at least one server, preferably lightly loaded, and connect to it. On the other extreme (Fig. 6), a client also acts as a server and implement a "pure" P2P overlay with all the other clients. DHT is used to locate a user. The first option does not require modifying the clients, but provides a scalable and reliable server farm architecture. But it still has some of the server maintenance and configuration problems, unlike the second option.

One problem with the pure P2P overlay of all nodes is that not all nodes have equal capacity and availability. For example, a node with low bandwidth connection to the Internet or those behind a firewall or NAT may not be able to fully function in a DHT because it may need in-bound connections, significant bandwidth for forwarding P2P messages or significant memory or CPU for maintaining DHT state. This problem can be solved by adopting an intermediate design as shown in Fig. 7. Some of the nodes with high capacity (bandwidth, CPU, memory) and availability (uptime, public address) are made super-nodes. Only the super-nodes form a DHT. An ordinary node just connects to one of the available super-nodes, similar to Kazaa. This is similar to the first option except that there is no distinction between clients and servers, and any node can be a super-node or ordinary node, depending on the capacity and availability. Our goal is to allow a P2P-SIP node to work in any of the above configurations.

The decision to become an ordinary node or a super node is usually local. When a node starts up it will become an ordinary node. When the ordinary node detects enough capacity and availability (public address and uptime), then it can transition to a super-node. A node with enough capacity and availability may be forced to become a super-node when an existing super-node is leaving or has reached the capacity limit. However, some nodes that are known to have enough capacity and availability can immediately transition to super-node upon startup. Existing peers can influence the neighbors to become a super-node.

Having two levels, super-nodes and ordinary nodes, does not improve the search latency bounds. The search latency is still $O(LogN)$. However, it improves the performance in practice because the DHT maintenance traffic is reduced if the nodes in the DHT are more stable. Effect of more than two levels on efficiency and maintenance cost is for further study.

4.3 Why REGISTER?

Users register their identifiers with the system so that other users can locate them. As shown in Fig. 8 when the user starts her client application and indicates her "screen name" as *alice@office.com*, the node computes the DHT key (e.g., using SHA1 as in Chord) from the name and joins the DHT using this user key as the node key. Alice's key is 42 in the example. When another user, say Bob, wants to locate Alice, Bob's node uses the same hash function to calculate the same key, 42, for Alice, and invokes the `find(42)` method on the DHT. The DHT algorithm locates node 42 and then, Bob's application can talk to Alice's application.

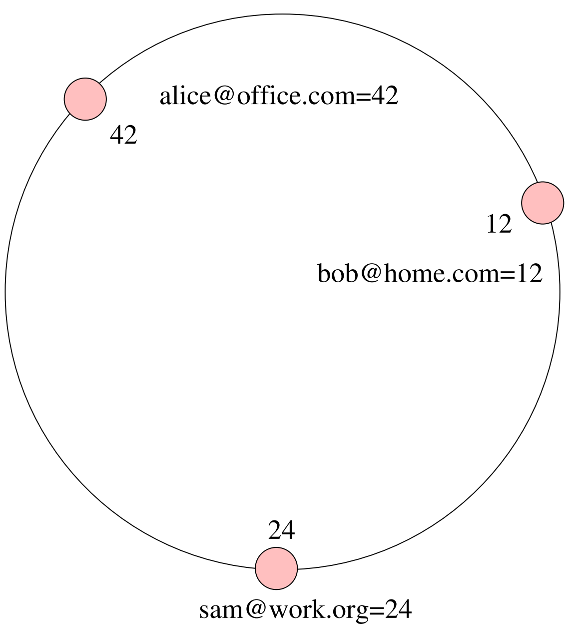

Figure 8: No REGISTER

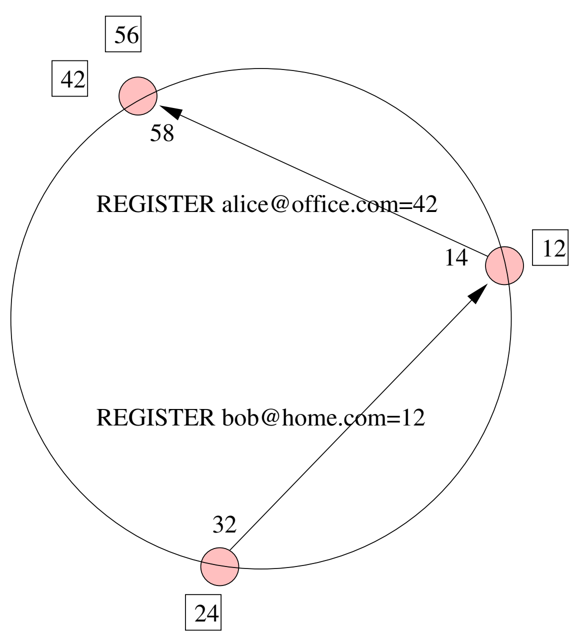

Figure 9: With REGISTER

This scheme can not support offline messages or multiple clients registered for the same SIP user identity, *sip:alice@home.com*. For example, if Alice is not present then Bob cannot leave a message for her. On the other hand, in Fig. 9, the node key and user key are computed separately. The SIP REGISTER message is used for inserting a node as well as registering a user identity in the DHT. Each node in the DHT acts as a registrar. When Alice starts her application, the node uses its IP address to compute the node key, 14. (In other DHT algorithms such as CAN [14], it can randomly choose a key). It then inserts itself into the DHT based on its node key by sending one or more SIP REGISTER messages. The node then computes the key on Alice's name and sends a SIP REGISTER message to the other node, with key 58, that is responsible for the user key 42. For example, Alice's node has a node key of 14 where as Alice's user key is 42, so the node 14 sends a REGISTER message for key 42. The node 58 that is responsible for key 42 accepts the registration and maintains the state that user Alice can be found at node 14's IP address. Even if Alice's application (node 14) is not available, Bob can still leave offline message with node 58 that can later be delivered to Alice when she comes online. Similarly, there can be multiple registrations for the same user key 42, if Alice has multiple active clients.

As an alternative to the SIP REGISTER message, one can use the SIP PUBLISH message to publish the user location and online status [22]. Both the messages are handled in the similar way for the purpose of this paper, so the choice does not affect the overall architecture.

4.4 Alternatives to P2P

We notice that the classical client-server architecture of SIP and the P2P-SIP architecture are two extremes. For example, in the former case, there is a per-domain server to locate the user in the domain and DNS is used to locate the server. In the latter case, P2P overlay is used to locate the node holding the user location. There can be an intermediate architecture that can use DNS to locate the server but the servers can dynamically join and leave the system using dynamic DNS. This gives rise to the service provider model where the provider sells the SIP service by becoming part of another provider's SIP server pool. DotSlash [23] explores this option in the context of web "hot spots" and uses service location protocol (SLP) to locate the backup servers. Such approaches need explicit synchronization of registration records among the participating servers similar to join and leave maintenance in P2P.

5 Architecture

Based on the different alternatives mentioned in the previous section, we propose our P2P-SIP architecture in this section. Fig. 10 shows the block diagram of the different components in the P2P-SIP node as follows:

Registration: When the node starts up and the user signs-in with her identifier, the registration module is activated to initiate NAT and firewall detection, peer discovery and SIP registration (Section 5.1).

Firewall and NAT detection: See Section 6.1.

User interface: This module interacts with the user, keeps track of her "buddy list" and invokes the user location module to locate the buddies.

User location: This module can invoke the SIP module or, if this node is a super-node, the DHT module to locate the user (Section 5.4).

DHT: This module is used when the node is a super-node to maintain DHT peers information (e.g., Chord *finger table*) and perform DHT logic. The API includes find, join and leave methods.

SIP: SIP is used as the underlying protocol for locating another user, registering the user, call setup and instant messaging. Once the user location is done, the call setup or instant messages can be sent directly via the SIP module. The API allows sending and receiving SIP requests and responses, and maintaining call and registration states.

Media path: The media path (audio device, codecs and transport) is largely independent of the P2P-SIP operation.

The DHT module uses the SIP messages to communicate with other peer nodes as we describe in this section.

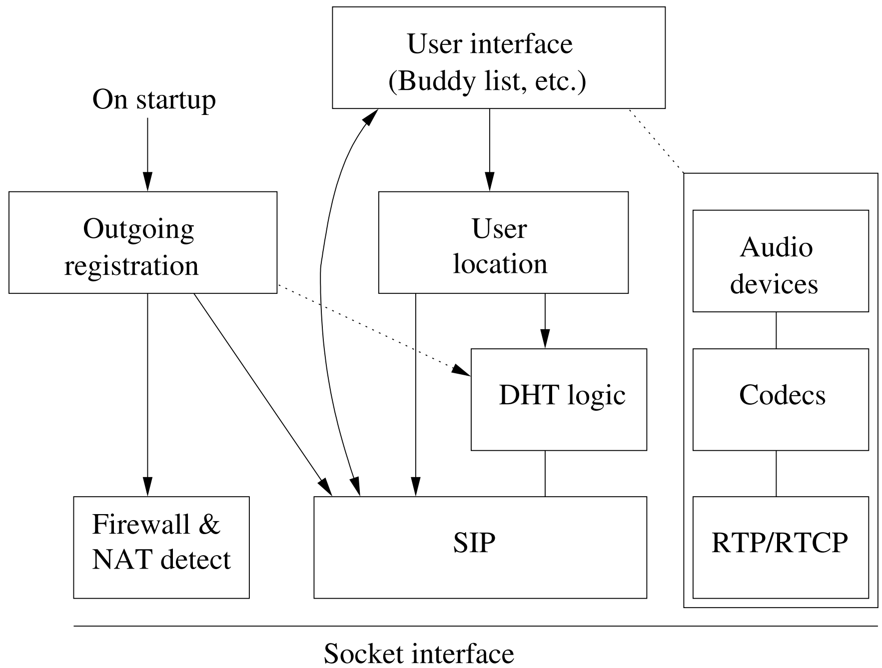

Figure 10: Block diagram of a P2P-SIP node

5.1 Node startup and peer discovery

In practice, the client node can try to use both P2P overlay and the SIP-based user lookup. When the node starts up and the user enters her identifier such as *alice@example.com*, the node finds the possible SIP server addresses using DNS [8] and sends a SIP REGISTER message as shown in Fig. 11. If the SIP registration succeeds, the node can be reachable using standard SIP mechanism in addition to the P2P mechanism.

The node also tries to discover possible super-nodes so that it can join the P2P overlay. Once you know one node in Chord, you can join the Chord DHT based on the node key. A number of approaches can be re-used from various existing proposals as follows:

- Multicast with very small time-to-live (TTL) value (e.g., within a LAN) can be used to discover local peers and get more super-node information from these peers. SIP defines multicast registration address for IPv4 as 224.0.1.75. Multicast-based node discovery may result in many disconnected DHT components. To prevent this, only existing DHT nodes (super-nodes) should respond to multicast discovery requests (i.e., ordinary nodes should not get discovered). Limited multicast on wide-area means the system can not rely on multicast alone.
- Some sort of service discovery can be used, e.g., SLP, to locate super-nodes [24].
- If the peer addresses are cached, then more super-node information can be obtained from those peers assuming the peers are still active and have not changed their locations since last seen.
- As the last resort, some pre-configured bootstrap peers can be obtained from DNS query to a well known domain, e.g., *sip-p2p.net*, or can be pre-configured in the application software (e.g., as implemented in Skype).

The super-node information is cached for subsequent registrations when the user logs out and logs in again. Hence, the discovery is going to be a one time affair for most installations unless all the cached super-nodes are found to have moved or disappeared.

5.2 User registration

Once it detects a set of super-nodes, it picks two and sends SIP REGISTER messages to register with them. Two super-nodes are used for redundancy. The To and From headers in the message correspond to the local user identifier, e.g., *sip:alice@home.com*. The Request-URI corresponds to the super-node's address, e.g., *sip:192.2.1.2:5060*.

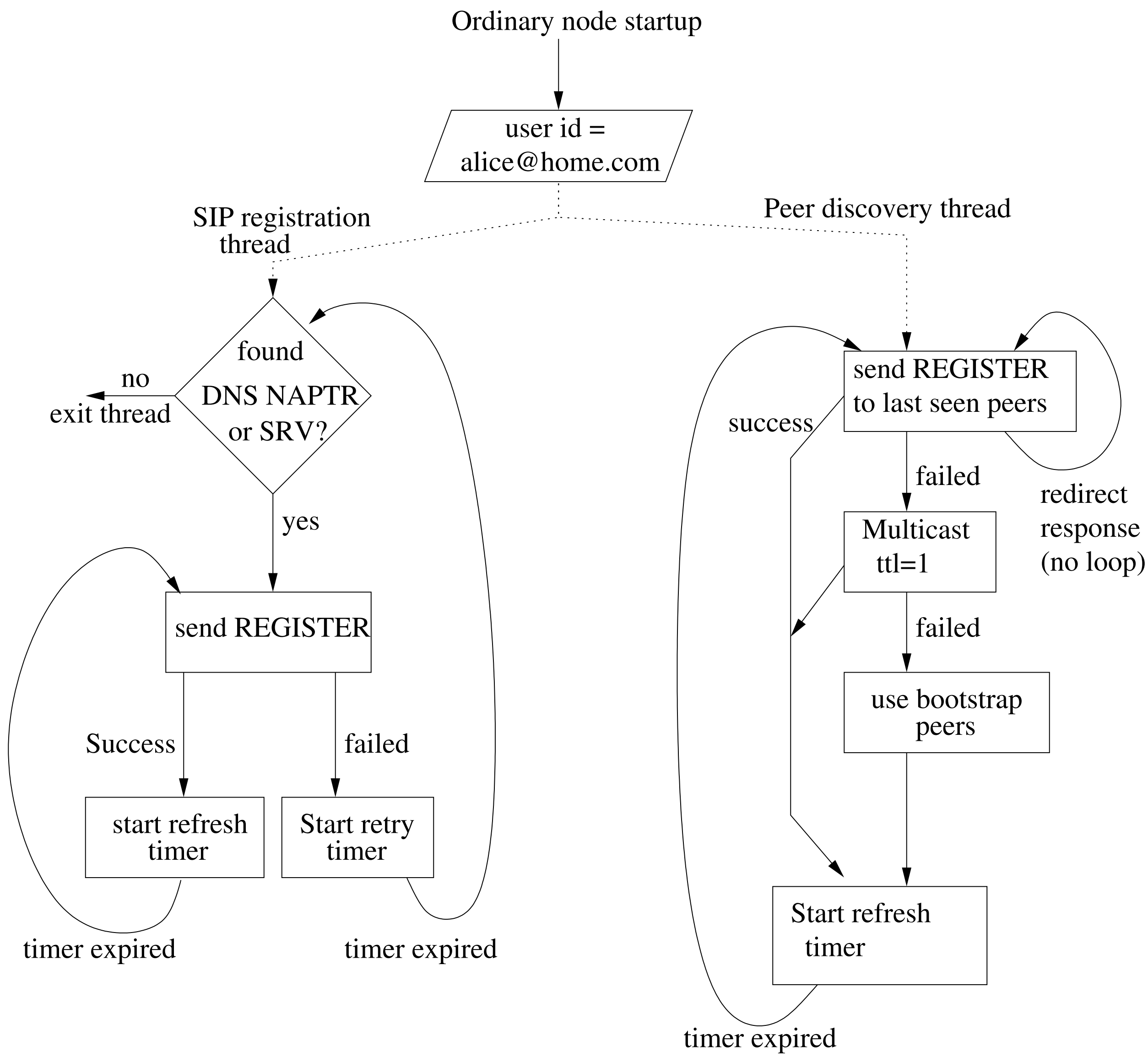

Figure 11: Node startup and outgoing registration

An ordinary node is just a SIP user agent, whereas a super-node serves as the SIP user agent as well as registrar for other nodes. A super-node sends the REGISTER messages on behalf of the attached nodes to the destination super-nodes in the DHT. It also joins the DHT with other super-nodes and actively takes part in user location lookup.

Ordinary nodes periodically send REGISTER refreshes to detect any super-node failures. Super-nodes can periodically send the SIP OPTIONS message among themselves or to the attached nodes to monitor liveness. The refresh interval can be adjusted based on the system load. The OPTIONS message is not sent if this node communicated with the destination node in the past interval.

When an ordinary node receives a REGISTER message, it sends the SIP redirect response to redirect the sender to its own super-nodes as shown in Fig. 12. When a super-node receives a REGISTER message and the sender is part of its attached nodes, it proxies the message to the appropriate nodes in the DHT as per user key of the sender. If the sender is not part of this super-node's attached nodes, it can decide either to accept the new node or reject it. If it wants to reject it, it redirects it to some other super nodes which may be less loaded than this super-node. The sender does loop detection to avoid getting into redirection loop. The detailed analysis of loop prevention is for further study.

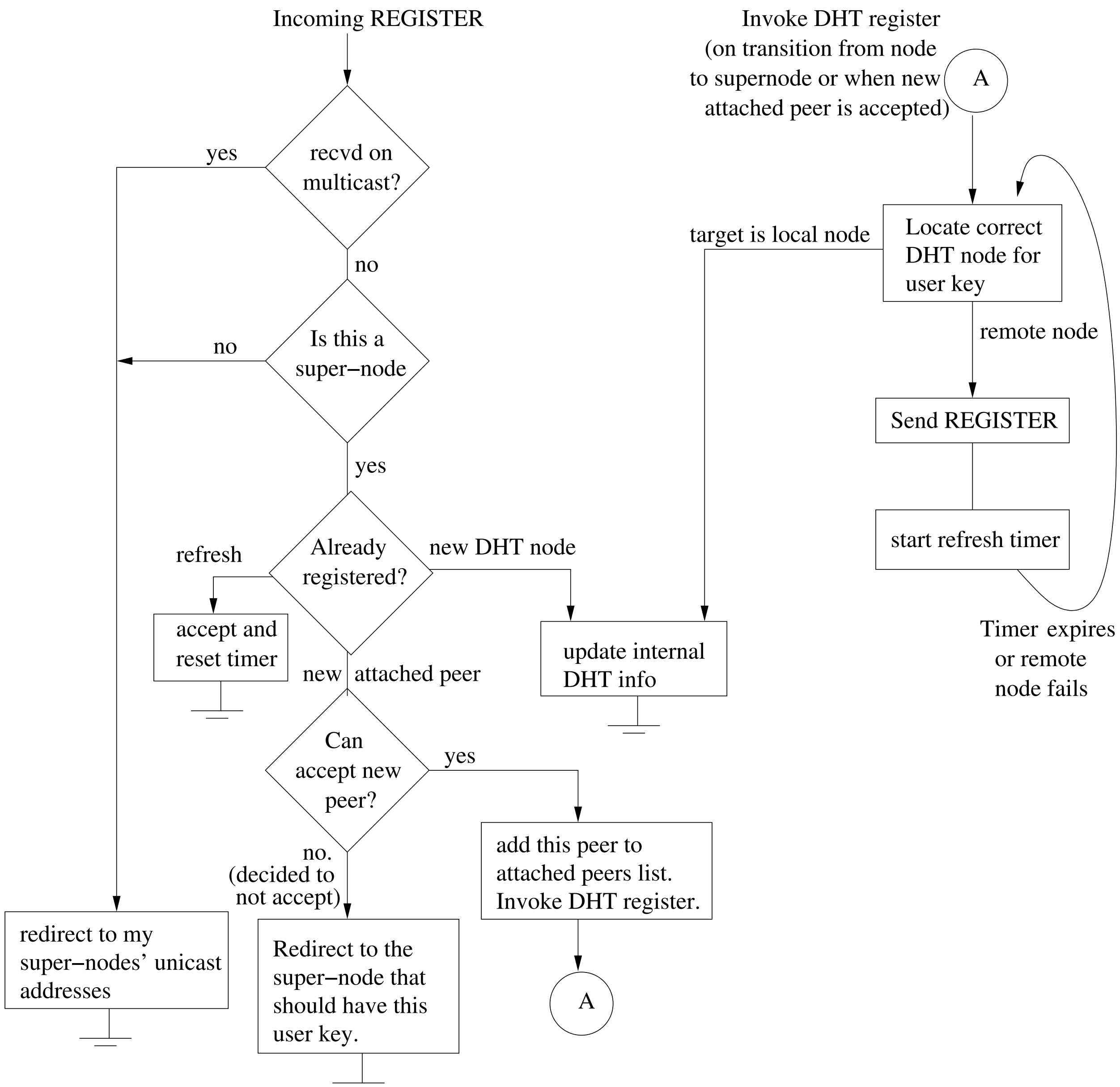

Figure 12: Incoming registration

5.3 Node shutdown or failure

When an ordinary node leaves the system it can just un-REGISTER with the attached super-node which in turn can propagate the un-registration to the corresponding nodes holding this node's key. A failure of an ordinary node does not affect the rest of the system. In any case, the attached super-node can detect the failure by the absence of periodic refresh. It can further confirm the failure by sending an OPTIONS message to the failed node to see if there is any response.

When a super-node leaves, the state needs to be updated in the attached ordinary nodes as well as the other super-nodes in the DHT that are neighbors of this node. If a super-node is shutting down, it gracefully transfers the user records that it holds to the other nodes in the P2P overlay. This guarantees that others users can locate the record when the DHT node is gracefully shutting down. It sends the SIP REGISTER message to the DHT nodes that will be holding the user records after this node leaves. It does not need to inform the attached ordinary nodes. The attached nodes will detect the failure on the next registration refresh and try to discover and connect to other super-node that

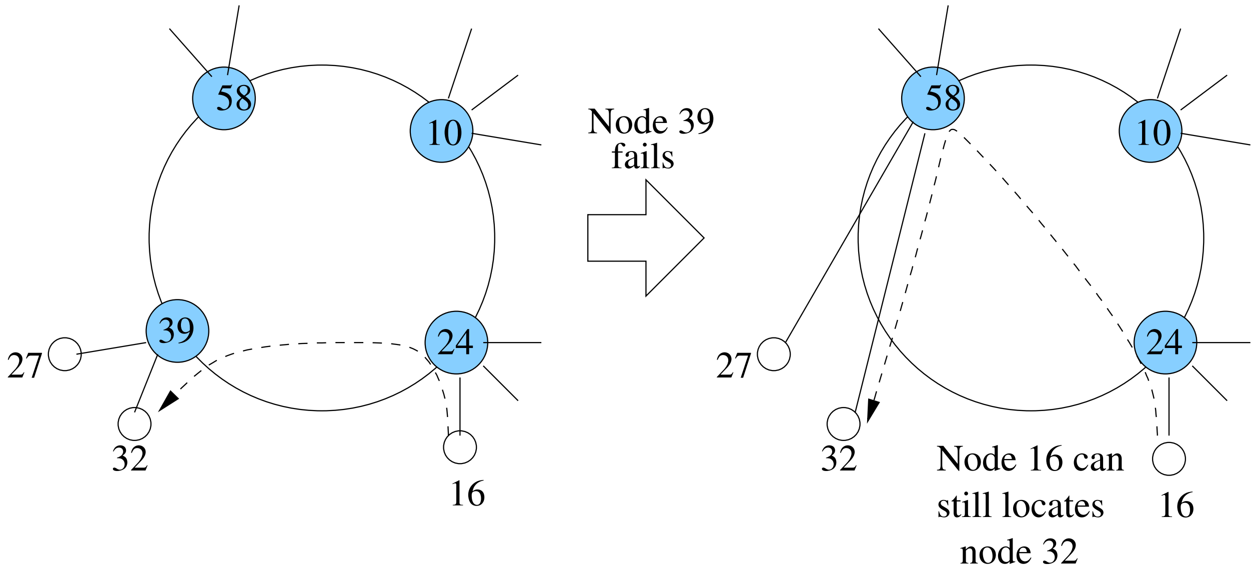

Figure 13: Failure of a super-node in the DHT

holds the record.

When a super-node fails abnormally (Fig. 13), the neighboring DHT nodes detect the failure and adjust the DHT to accommodate for the keys that were held by the failed nodes. However, the mapping is lost unless the originating node sends REGISTER refresh. The REGISTER refresh goes to the new super-node that handles the corresponding key in the DHT. This make some services such as offline messaging temporarily unavailable (Section 6).

To distinguish a SIP-only application with a SIP/P2P application, we can use the Require header in the OPTIONS or REGISTER messages.

5.4 User location and call setup

User can watch the presence status of other users by specifying their identities in his "friends" list. If the user already has a friends list, the node tries to locate those friends on startup. Initially we assume that the friends list is stored in the local computer for this user. Later we extend this in Section 6 to store any user information (including the friends list) on the P2P network to provide device independence to the user. The IP addresses of all the friends are cached for future use.

The only important step for the purpose of this paper is locating the node that has the user location record for the destination user. Once the call setup is complete, media packets are sent end-to-end. A node sends the SIP MESSAGE or INVITE message for instant message or multimedia call, respectively. If the destination address is cached because, for example, this node made a recent call or instant message to that destination, then the cached address is used. If the client at the cached IP address does not respond (because there is no client running or the client is not a SIP/P2P node), then the cache entry is removed and discovery is restarted.

The SIP-based lookup [8] and P2P lookup is done simultaneously as shown in Fig. 14. For P2P lookup, an ordinary node sends a INVITE or MESSAGE to the attached super node, which acts as a SIP proxy. A super-node locates the destination node holding the key in the underlying DHT. Once the mapping is obtained, it can either proxy or redirect the message. Redirection is the preferred way as it takes the super-node out of the call loop. However, in some cases such as those involving firewall and NAT, proxy is the only option as we show in Section 6.1.

Some DHTs (e.g., CAN) may allow parallel search to multiple peers, unlike sequential search of Chord. In this case the super-node may act as a back-to-back user agent (B2BUA) and propagate the SIP message to the neighboring peers. However, parallel search should be avoided to prevent flooding the network, except possibly in the case of emergency call routing, such as 911 calls in the United States.

Other SIP functions such as third-party-call control and call-transfer are implemented in the similar way. For example, the SIP REFER message for call transfer is routed similar to INVITE on the P2P overlay. Most of the messages are handled end-to-end directly by the communicating nodes without going over the P2P overlay. Only

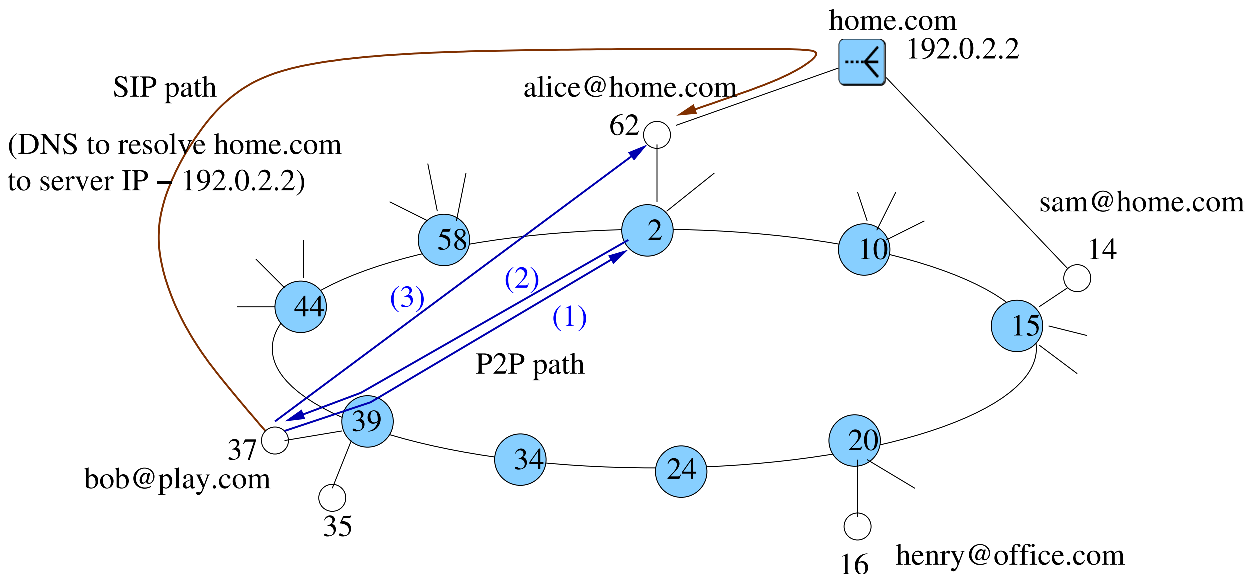

Figure 14: User location and call setup

dialog initiating messages such as INVITE or out of dialog messages such as first MESSAGE for instant messages need to use the P2P lookup service.

6 Advanced services

Basic call setup is not enough to be competitive in Internet telephony. This section describes some of the advanced services such as NAT and firewall traversal, offline message storage and multi-party conferencing.

6.1 NAT and firewall traversal

In an ideal world, ISPs and corporate system administrators should enable their NAT and firewall devices with SIP proxies or application level gateways (ALG). However, in practice, this is rarely done. This forces the application developers to write customized kludges to work around the NAT and firewall [17, 25].

There are two aspects to NAT and firewall traversal: automatic detection of the type of NAT and firewall and tunneling though the NAT and firewall devices for inbound or outbound messages. The detection is done at the application startup when the node connects to a super-node. The node implements the Interactive Connectivity Establishment (ICE) algorithm [17] for NAT traversal. UDP is preferred mode of communication. However, if UDP messages can not be received (e.g., the firewall blocked UDP), then a persistent TCP tunnel presumably to port 80, initiated from the internal node to the external super-node can be used for both inbound and outbound messages.

6.2 Offline messages

This section describes the problems with offline messaging. When Alice calls Bob or leaves an instant message for Bob, and Bob is not online, the message should be stored reliably by the system and delivered to Bob when he comes online.

There are three places where we can store the offline messages: the source, the destination or some intermediate node in the P2P overlay. The classical PSTN voice mails are stored in the destination answering machine attached to the callee's phone, or in some cases in centralized voice mail server attached to the destination PBX. Similarly, the P2P-SIP client running on the destination user's machine can store the message if the destination user did not pick up the phone. The problem comes when the destination phone itself is not active or the user has not started her client.

One way to achieve this is by having the DHT peer that is responsible for storing location of Bob, also store the offline multimedia messages for Bob as shown in Fig. 15. In the case of node failure, the offline messages become

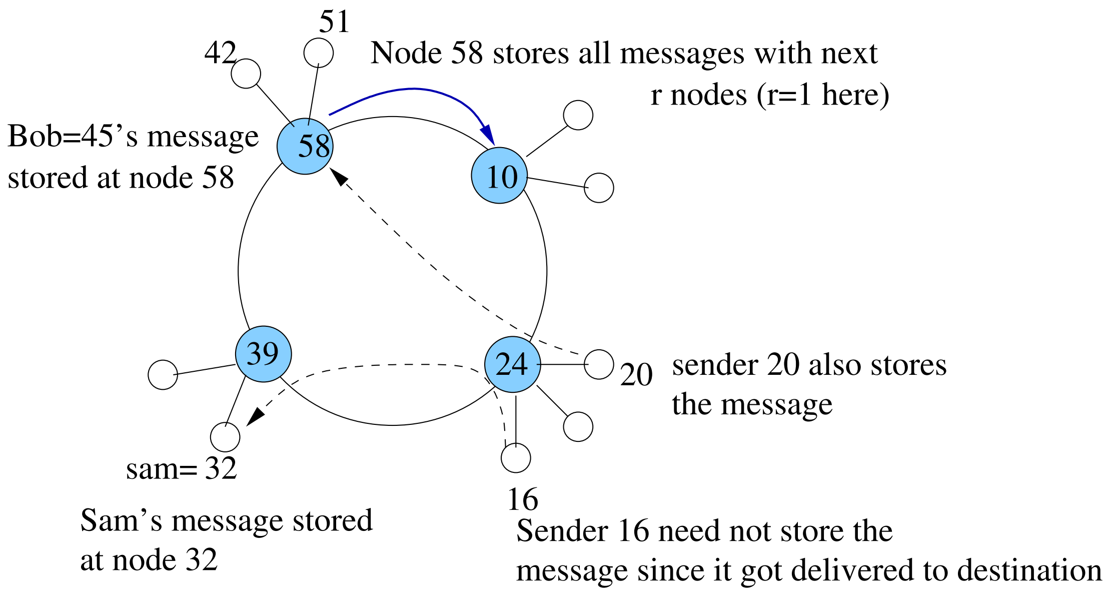

Figure 15: Offline message storage

unavailable until the storage node becomes online again. To solve this, the node can store the message in multiple places and keep them consistent similar to the Oceanstore architecture [26]. A P2P file storage system with message waiting indication is sufficient to implement offline message storage. POST [27] is a P2P messaging system that can also be used for offline messages.

Another option is for the caller node to cache the message locally and deliver it to the destination node when the destination becomes available.

The message delivery notification is reliably sent back to the caller. If the message is not delivered or the storage node fails, then the caller node finds the new storage node and records the message again without any user intervention. When a node starts up, it checks for any undelivered message from past boot cycle, and tries to re-send them upon bandwidth and CPU availability. This has certain security issues if the same machine is used by many users as in an Internet kiosk.

Unlike email system, where the intermediate Mail Transfer Agents (MTAs) are reliable and delivery confirmation from an MTA is sufficient, in P2P-SIP an end-to-end confirmation is desirable. Alternatively a third party storage server can take the ownership of the message for the subscribed user, relieving the sender from keeping a copy.

Some nodes may just cache a summary of undelivered messages (such as subject, date, headers) instead of the complete multimedia content to save on bandwidth and disk space. Some nodes may attempt to send the message by alternative means such as email if the email identity can be cryptographically verified to belong to the destination user.

To receive the offline message, the destination node subscribes to the message waiting indication (MWI) event with the P2P network and gets notified on startup when a new offline message is available. The node can then fetch the message using file transfer or real-time multimedia call to a special URI such as *sip:bob-vmail@server*. Alternatively, the user can buy MWI service from some centralized service provider that registers with the P2P-SIP network on behalf of the user to receive her calls.

6.3 Multi-party conferencing

In classical telephony, multi-party conferencing is done via pre-arranged dial-in conference bridges (or conference servers). These conference servers can register the intended conference addresses such as "staff-meet@columbia.edu" with the P2P overlay. However, the mixing is done by a centralized server which can become the potential bottleneck for large conferences.

For small scale ad hoc conferencing among the participants, one of the participant who has good capacity (CPU, memory, bandwidth) can become the mixer and mix audio from other participants. Since audio mixing requires access to the un-encrypted audio samples from all the speakers, one cannot pick an untrusted peer as the mixer. One viable alternative is to pick an existing conference participant as the mixer.

Completely decentralized conferencing [28] can be used to establish a full-mesh signaling and media relationship among the participating members. The protocol works for concurrent join and leave of members in the conference. This prevents dependency on a single peer node that does mixing.

Instead of a full-mesh media, a multicast media distribution tree can be used. It assumes that a small number of members (say one or two) will be speaking at any instant, and the receiving node can select or mix the audio samples from multiple streams in the session. Several P2P application layer multicast schemes have been proposed [29, 30], some of which can use the proximity information available in the underlying DHT [15, 31].

6.4 Device independence

So far we assumed that a user logs in from a particular node and all the user profile information such as friends list or privacy policy are stored in the local node. However, similar to file storage systems or storing offline messages in P2P-SIP, the node can store the encrypted user profile information also in the P2P overlay network [26]. On startup when the user signs in her identifier, the node fetches the profile information reliably and uses that.

7 Security

Security is one of the most important problem to be solved for any P2P system because of potentially untrusted peers [32]. The problems include: (1) authentication (to prevent unauthorized calls from spammers), (2) encryption (to prevent others not in the call setup path knowing about the call information), (3) privacy and confidentiality (to prevent sending information to untrusted entity and to prevent misuse of information) and (4) dealing with malicious nodes (what if a peer node happily accepts the call requests but drops them without forwarding to the appropriate node). The first two problems (authentication and encryption) can be solved using mechanisms similar to those proposed for SIP telephony. For example, end-to-end digest authentication, hop-by-hop transport layer security (TLS) or end-to-end S/MIME can be used.

A number of "untrusted" peers may be involved in user location lookup for a call, unlike the "trusted" servers in the classical SIP telephony. In the classical server based telephony, as long as both caller and callee can trust the server for privacy and confidentiality of the call information, there is no problem. Secondly, the peers may be acting correctly but secretly logging all the call requests which may later be misused.

Freenet [6] solves this problem by hop-by-hop routing of request and responses where each hop (peer) changes the source identifier. This prevents any peer in the request path to know the original sender of the request. Similar techniques can be used in P2P-SIP architecture assuming absence of collusion (i.e., multiple malicious peers collaborating to know the call information).

A number of reputation systems have been proposed for P2P [33, 34, 35, 36]. However, they focus on file sharing systems (not real-time), have centralized components, assume co-operating peers or have problems of collusion and multiple identities. Further study is needed to detect the peers who are known to drop calls or do other malicious behavior so that they are not used in the call routing path and not allowed to become part of the underlying DHT.

There is another kind of threat to P2P systems, called "free riding" [37]. Some nodes may want to use the P2P services for making and receiving calls but refuse to serve in the user location lookup process by becoming a super-node. The system should enforce some policy to discourage such peers. For example, peers can earn some credit for doing services which can later be used for using the services. Every peer can start with an initial amount of credit. Peers behind a NAT and firewall may have to pay for the service if they can not serve by becoming a super-node. Nodes that run out of credits and refuse to pay are declined the service.

If the user identity is easy to obtain (e.g., yahoo.com email addresses), then people can always acquire new identities to make outbound calls. However, they won't be able to advertise their identity for incoming calls for long period, if they do not serve in the P2P overlay or pay credits to other serving peers. Moreover, making outbound calls does not entitle them to free gateway access or free PSTN calls.

Finally, if automatic software updates are incorporated in P2P-SIP nodes, then it must be done in a reliable, secure and decentralized manner.

8 Performance prediction

8.1 Scalability

Scalability of the P2P-SIP network depends on the capacity (bandwidth, CPU, memory) of the individual participating super-nodes. Suppose there are N super-nodes in the Chord ring, identifier space is m-bit long (i.e., the identifier range is 0 to $2^m - 1$), number of registered users in the system is n (such that number of keys stored per node is approximately k=$\frac{n}{N}$), REGISTER refresh rate to successor and predecessor to keep the Chord ring correct is r_s, refresh rate for finger table entry is r_f, call arrival is poisson distribited with mean c per node, user registration is uniformly distributed with mean interval t per user, and node joining and leaving are poisson distributed with mean λ. Because average lookup in Chord travels through $O(log(N))$ nodes [7], the finger refresh messages, call arrival messages and user registration refresh messages travel $O(log(N))$ hops. There are $O(log(N))$ finger table entries per node. Node join and leave generate $O((log(N))^2)$ messages. The average message rate per node is sum of the message rates due to refresh, call arrival, user registration and node join or leave, which can be given as:

$$M = \{r_s + r_f(log(N))^2\} + c.log(N) + \frac{k}{t}log(N) + \frac{\lambda(log(N))^2}{N}$$

The message rate in the node determines the bandwidth and CPU utilization for the node. If each node can handle C requests per second, then the equation $C = M$ gives the maximum possible number of nodes, N_{max}, in the system, which roughly translates to $N_{max} = 2^{\frac{C}{r+c}}$ for large N, where r is the refresh rate and c is the call rate. Note that λ is low because nodes which often join and leave are not made super-nodes.

Suppose the node supports 10 requests per second (which is much less than the typical SIP proxy performance [38]) with minimum refresh interval of one minute (r = $\frac{1}{60}$) and call rate of one call per minute per node, then the maximum number of nodes in the system can be 2^{10*30}. If more nodes join the system, the super-nodes become overloaded and may deny some incoming call, registration or proxy requests. However, large values of N also increases the call setup latency as we describe below.

8.2 Reliability

When a node fails the user registrations stored on that node are lost. To achieve reliability, the refresh rate can be increased (so that node failure detection happens quickly), the user registration refresh rate can be increased (so the the user record is unavailable only for a brief period of time) or the user registration record can be replicated at multiple nodes (e.g., store the user registrations at $log(N)$ successive nodes in Chord). We plan to quantify the effect of each factor on mean time to recover (MTTR) from node failures for a given user record. The equation for average message rate does not change if λ includes failure rate along with node join and leave rates.

8.3 Call setup latency

The P2P advantages come at the cost of increased call setup latency. For example, with 10,000 nodes in Chord, the average lookup path length is six hops [7], so P2P call setup will take about six times more than traditional client-server call setup in SIP. With good network condition, single lookup (INVITE response) in SIP is expected to take less than 200 ms. So one or two seconds delay before the phone rings in P2P-SIP is tolerable given that on an average the phone will ring for much longer before the callee picks up.

Due to P2P synchronization latency which depends on refresh rate and node join, leave and failure rates, there may be delay in updating the user records. In this case, it may take multiple retransmissions before call setup is complete. This further increases the call setup latency. Successful user location in Skype takes about three to eight seconds [**?**].

Some kind of hybrid system may be implemented that takes the advantages of many different structured and unstructured P2P algorithms to further reduce the latency and maintenance cost. For example, there has been recent proposal on one hop lookups for P2P [39] assuming large storage space in the peer nodes.

9 Conclusions and future work

We have described a pure P2P architecture for SIP telephony. The architecture provides reliability and scalability inherent in P2P systems, in additional to interoperability with existing SIP infrastructure. The advantages come at the cost of increased call setup latency.

We analyze various design alternatives, propose a P2P-SIP architecture using Chord as the underlying DHT, and describe various user location and registration steps in detail. We also present an overview of various advanced services such as offline messaging, conferencing, NAT and firewall traversal and security issues.

We are implementing a P2P-SIP node for multimedia communication using our SIP C++ library. We will be doing performance measurement for reliability and scalability on our actual system instead of using simulations. Since the implementation is based on Chord [7], more simulations may not add any research value to the existing simulation results.

More work is needed in advanced services such as large scale application level multicast conferencing using P2P, distributed reputation system for peers, and PSTN interworking related issues such as authentication and accounting. There should be a reasonable incentive to become a super-node to provide services to other peers.

We are working on allowing an internal node inside a firewall and NAT to become a super-node. This reduces the load on public super-nodes, since most of the users typically will be behind some firewall and NAT. Alternatively, the private nodes in a domain can form a secondary P2P overlay connected to the public DHT via a few external connections to reduce the port utilization on the NAT device. .

Some of the open questions described in [40] are relevant to P2P-SIP architecture also. Some kind of hybrid system may be implemented that takes the advantages of many different structured P2P algorithms to further reduce the latency and maintenance cost. For example, there has been recent proposal on one hop lookups for P2P [39] assuming large storage space. Applying this in P2P-SIP is for further study.

Finally, we conclude on a note that unless the SIP servers (proxies, registrars) are widely deployed, we will need P2P based IP telephony tools so that everyone can use the system. Such P2P-SIP architecture can be extended to other protocols such as H.323.

10 Acknowledgment

Salman Abdul Baset helped in understanding the Skype architecture. The work is supported by a grant from SIPquest, Inc.

References

[1] Henning Schulzrinne and J. Rosenberg, "Internet telephony: Architecture and protocols – an IETF perspective," *Computer Networks and ISDN Systems*, vol. 31, no. 3, pp. 237–255, Feb. 1999.

[2] J. Rosenberg, Henning Schulzrinne, G. Camarillo, A. R. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: session initiation protocol," RFC 3261, Internet Engineering Task Force, June 2002.

[3] James Toga and Jörg Ott, "ITU-T standardization activities for interactive multimedia communications on packet-based networks: H.323 and related recommendations," *Computer Networks and ISDN Systems*, vol. 31, no. 3, pp. 205–223, Feb. 1999.

[4] Kundan Singh and Henning Schulzrinne, "Failover and load sharing in SIP telephony," Tech. Rep. CUCS-011-04, Columbia University, Computer Science Department, New York, NY, USA, Mar. 2004.

[5] Haakon Bryhni, Espen Klovning, and Øivind Kure, "A comparison of load balancing techniques for scalable web servers," *IEEE Network*, vol. 14, no. 4, July 2000.

[6] Dejan Milojicic, Vana Kalogeraki, Rajan M Lukose, Kiran Nagaraja, Jim Pruyne, Bruno Richard, Sami Rollins, and Zhichen Xu, "Peer-to-peer computing," technical report HPL-2002-57 20020315, Technical Publications Department, HP Labs Research Library, Mar. 2002, http://www.hpl.hp.com/techreports/2002/HPL-2002-57.html.

[7] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *SIGCOMM*, San Diego, CA, USA, Aug 2001.

[8] J. Rosenberg and Henning Schulzrinne, "Session initiation protocol (SIP): locating SIP servers," RFC 3263, Internet Engineering Task Force, June 2002.

[9] "Kazaa: peer-to-peer file sharing software application," http://www.kazaa.com.

[10] “Gnutella: peer-to-peer file sharing software application,” http://www.gnutella.com.

[11] “Zero configuration networking (zeroconf),” http://www.ietf.org/html.charters/zeroconf-charter.html.

[12] “Skype: Free internet telephony that just works,” http://www.skype.com.

[13] Zihui Ge, Daniel R. Figueiredo, Sharad Jaiswal, James F. Kurose, and Don Towsley, “Modeling peer-peer file sharing systems,” in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, Mar. 2003.

[14] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker, “A scalable content-addressable network,” in *SIGCOMM Symposium on Communications Architectures and Protocols*, San Diego, CA, USA, Aug. 2001, ACM.

[15] Antony Rowstron and Peter Druschel, “Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems,” in *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Heidelberg, Germany, Nov. 2001, pp. 329–350.

[16] David Liben-Nowell, Hari Balakrishnan, and David Karger, “Analysis of the evolution of peer-to-peer systems,” in *ACM Conf. on Principles of Distributed Computing (PODC)*, Monterey, CA, USA, July 2002, ACM.

[17] Jonathan Rosenberg, “Interactive connectivity establishment (ICE): a methodology for nettwork address translator (NAT) traversal for the session initiation protocol (SIP),” Internet draft, Internet Engineering Task Force, July 2003, Work in progress.

[18] Frank Strauss and S. Schmidt, “P2P CHAT - a peer-to-peer chat protocol,” Internet draft, Internet Engineering Task Force, June 2003, Work in progress.

[19] “Groove workspace software,” http://www.groove.net.

[20] “Magi p2p technology being adopted across vertical industries,” http://www.endeavors.com/PressReleases/partners1.htm.

[21] Steven D. Gribble, Eric Brewer, Joseph Hellerstein, and David Culler, “Scalable, distributed data structures for Internet service construction,” in *Operating Systems Design and Implementation*, San Diego, CA, USA, Oct. 2000, Usenix.

[22] A. Niemi, “Session initiation protocol (SIP) extension for event state publication,” Internet Draft draft-ietf-sip-publish-02, Internet Engineering Task Force, Jan. 2004, Work in progress.

[23] Weibin Zhao and Henning Schulzrinne, “Dotslash: A scalable and efficient rescue system for handling web hotspots,” Tech. Rep. CUCS-007-04, Department of Computer Science, Columbia University, New York, New York, Feb. 2004.

[24] Weibin Zhao, Henning Schulzrinne, and Erik Guttman, “Mesh-enhanced service location protocol (mslp),” RFC 3528, Internet Engineering Task Force, Apr. 2003.

[25] B. Ford, P. Srisuresh, and D. Kegel, “Peer-to-peer communication across middleboxes,” Internet Draft draft-ford-midcom-p2p-01, Internet Engineering Task Force, Oct. 2003, Work in progress.

[26] John Kubiatowicz, David Bindel, Yan Chen, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, and Hakim Weatherspoon, “Oceanstore: An extremely wide-area storage system,” technical report UCB//CSD-00-1102, U.C. Berkeley, CA, USA, May 1999.

[27] Alan Mislove, Ansley Post, Charles Reis, Paul Willmann, Peter Druschel, Dan Wallach, Xavier Bonnaire, Pierre Sens, Jean-Michel Busca, and Luciana Arantes-Benzerra, “Post: A secure, resilient, cooperative messaging system,” in *HotOS IX: The 9th workshop on hot topics in operating systems*, Lihue, Hawaii, USA, May, USENIX.

[28] Jonathan Lennox and Henning Schulzrinne, “A protocol for reliable decentralized conferencing,” in *ACM NOSSDAV 2003*, June 2003.

[29] Miguel Castro, Michael B. Jones, Anne-Marie Kermarrec, Antony Rowstron, Marvin Theimer, Helen Wang, and Alec Wolman, “An evaluation of scalable application-level multicast built using peer-to-peer overlays,” in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, Mar. 2003.

[30] Suman Banerjee, Bobby Bhattacharjee, and Christopher Kommareddy, “Scalable application layer multicast,” in *SIGCOMM Symposium on Communications Architectures and Protocols*, Pittsburgh,PA, Aug. 2002, p. 13.

[31] Miguel Castro, Peter Druschel, Y. Hu, and Antony Rowstron, “Proximity neighbor selection in tree-based structured peer-to-peer overlays,” technical report MSR-TR-2003-52, Microsoft Research, 2003.

[32] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan Wallach, “Security for structured peer-to-peer overlay networks,” in *Operating Systems Design and Implementation*, Boston, MA, Dec. 2002, Usenix.

[33] Minaxi Gupta, Paul Q. Judge, and Mostafa Ammar, “A reputation system for peer-to-peer networks,” in *ACM NOSSDAV 2003*, June 2003.

[34] Seungjoon Lee, Rob Sherwood, and Samrat Bhattacharjee, “Cooperative peer groups in NICE,” in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, Mar. 2003.

[35] Sepandar Kamvar, Mario Schlosser, and Hector Garcia-Molina, “The eigentrust algorithm for reputation management in P2P networks,” in *International World Wide Web Conference (WWW)*, Budapest, Hungary, May 2003, International World Wide Web Conference Committee.

[36] Li Xiong and Ling Liu, “Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 7, pp. 843–857, July 2004.

[37] Eytan Adar and Bernardo A. Huberman, “Free riding on gnutella,” *First Monday*, vol. 5, no. 10, Oct. 2000.

[38] Jonathan Lennox, “Services for internet telephony,” PhD. thesis, Department of Computer Science, Columbia University, New York, New York, Jan. 2004, http://www.cs.columbia.edu/˜lennox/thesis.pdf.

[39] Anjali Gupta, Barbara Liskov, and Rodrigo Rodrigues, “One hop lookups for peer-to-peer overlays,” in *HotOS IX: The 9th workshop on hot topics in operating systems*, Lihue, Hawaii, USA, May, USENIX.

[40] Sylvia Ratnasamy, Scott Shenker, and Ion Stoica, “Routing algorithms for DHTs: some open questions,” in *International Workshop on Peer-to-Peer Systems (IPTPS)*, Cambridge, MA, USA, Mar. 2002, IEEE.