College of William & Mary

Computer Science Department

Technical Report
WM-CS-2006-03
June 1, 2006

Challenges of DHT Design for a Public Communications System

David A. Bryan Marcia Zangrilli Bruce B. Lowekamp

Abstract

Communications systems, encompassing VoIP, IM, and other personal media, present different challenges for P2P environments than other P2P applications. In particular, reliable communication implies that each resource (person) is unique and must be reliably located, without false negatives. The system must function in the presence of NATs, which create non-transitive connectivity, and must be resilient against DoS attacks that attempt to disrupt its routing properties or DoS a particular person. We have designed and implemented a P2P communications system using an extension of the Chord algorithm as a resource location service. In this paper we present the design tradeoffs necessary to meet the requirements of a reliable communications system. In particular, the practical issues of non-transitive routing, NATs used by residential endpoints, and the prevention of DoS attacks are more critical than strict performance metrics in selecting DHT identifier, topology, and routing algorithms. Where a central authority exists, certificates can be stored in the overlay and allow more efficient DHT algorithms to be used, but securing an open network with NATs requires appropriate Node-ID, replica placement, and routing algorithms.

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

Challenges of DHT Design for a Public Communications System

Abstract

Communications systems, encompassing VoIP, IM, and other personal media, present different challenges for P2P environments than other P2P applications. In particular, reliable communication implies that each resource (person) is unique and must be reliably located, without false negatives. The system must function in the presence of NATs, which create non-transitive connectivity, and must be resilient against DoS attacks that attempt to disrupt its routing properties or DoS a particular person. We have designed and implemented a P2P communications system using an extension of the Chord algorithm as a resource location service. In this paper we present the design tradeoffs necessary to meet the requirements of a reliable communications system. In particular, the practical issues of non-transitive routing, NATs used by residential endpoints, and the prevention of DoS attacks are more critical than strict performance metrics in selecting DHT identifier, topology, and routing algorithms. Where a central authority exists, certificates can be stored in the overlay and allow more efficient DHT algorithms to be used, but securing an open network with NATs requires appropriate Node-ID, replica placement, and routing algorithms.

1. Introduction

Applying P2P technologies to communications systems has been gaining significant interest recently. Communications systems are, in some sense, a natural application for P2P because the dominant standard protocol for VoIP and IM, SIP [22], is already designed to support intelligent endpoints capable of end-to-end media connections. The challenge is to replace the client-server based registration and naming system [21] with a P2P-based distributed location service. However, because communications systems have different service requirements than other P2P applications, modifications to the algorithms are required.

1.1 P2P Communications

Whereas many P2P applications are built around anonymity, a fundamental requirement of communications systems is the ability to identify users and verify their identities. In filesharing applications there may be many nodes that contain replicas of the resource we are looking for. In communications systems, such as telephony or IM, there is usually a specific person who is being sought, and a close match just will not work.

The lack of anonymity and availability of the desired resource (person) at only a single location, coupled with the reliability requirements people expect out of their communications devices, provide the requirement that

- the P2P algorithm must report the correct location of a resource, if present. False negatives are unacceptable.

The requirement to avoid false negatives has implications that essentially require a DHT or other structured P2P algorithm to be used. Furthermore, the algorithm must contain sufficient protection from an attacker manipulating the overlay routing to gain control of a particular portion of identifier space to DoS attack against a particular user.

On the other hand, false positives (reporting that a person is present in the network, when they are not) are not a problem because the SIP user-agents can determine that the desired contact is not available. However, this is only one of many scenarios where positive identification of users is necessary

- A user-agent must be able to confirm the identity of the user contacted

This requirement is typically met through public-key cryptography, and may be assisted by the DHT, but does not rely on the DHT's integrity.

Deploying a P2P communications systems dictates that it must support the full range of last-mile networking technologies deployed on the Internet. Although file-sharing networks may be comprised of significant numbers of college students on relatively open networks, the anticipated market for communications services include both consumer broadband networks and small businesses, which are typically behind Network Address Translators. NATs introduce the problem of non-transitive connectivity. Freedman et al [9] explored some issues of Non-Transitivity in a Planet Lab based system. In such a system, the cause of these non-transitive connections is generally ephemeral problems or difficulties in Internet1 systems communicating with Internet2 systems. In contrast, in Internet deployed communications systems, particularly those targeted at consumers, a large fraction of the users are likely behind NATs, and non-transitive connectivity is the norm rather than the exception.

- The DHT must provide connectivity between NATed nodes and make use of the in the overlay to the extent possible.

While super-nodes can be used to provide connectivity, a DHT achieves its best functionality and scalability when all nodes are able to provide services as many services as possible to other peers.

Finally, as with any other publicly deployed network, the network must be resilient against DoS attacks. In particular, a P2P communications network is subject to both general DoS attacks against the entire overlay and DoS attacks targeting specific users. Therefore,

- the DHT algorithm must not amplify DoS attacks and
- the DHT algorithm must prevent an attacker from gaining control of a particular portion of the identifier space

1.2 Contributions

In this paper we present the approaches we have developed to address the practical issues of deploying P2P communications systems across real-world mixes of residential, commercial, and academic networks. In particular:

- While standards are being developed for NAT hole-punching, securing a system using NATed nodes remains challenging, as traditional IP-address hashing can facilitate several attacks. In Section 3 we propose a secure scheme for Node-ID selection and a corresponding replication scheme to assure data integrity.
- The routing scheme used by a P2P algorithm is significant to both its performance and resistance against attacks. In Section 4 we review the characteristics of two recursive and one iterative routing algorithm and discuss the costs, benefits, and selection criteria of each.
- Whereas commercial P2P communications systems typically rely on central servers, Section 5 presents a scheme where certification authorities (CA) can achieve equivalent results without any centralized operations after a one-time registration.

A key observation we make throughout the paper is that there is no single right answer to any of these issues. Depending on the degree of trust in the participants (and corresponding openness of the network), control over the software, and availability of a central authority, different decisions must be made to preserve the integrity, scalability, and performance of the network.

2. Background

Our discussion will focus on structured P2P networks, as unstructured networks cannot generally meet our requirement of no false negatives. Although our paper focuses on extensions to the Chord algorithm [25], most discussions apply to any of the DHT-based solutions.

Most communications systems to-date have been based on client-server principles. The traditional phone network—the public switched telephone network (PSTN)—concentrated control in the core of the network. More recent communications protocols, such as the IETF SIP protocol for VoIP and instant messaging [6, 22], have moved more control to the endpoints. SIP allows endpoints to set up direct connections to exchange media (phone calls), although in practice the signalling is typically routed through proxies. Media often flows directly between nodes, but may also be proxied to solve a variety of routing, NAT penetration, and other issues. Furthermore, SIP clients require servers to act as registrars and proxies to register their presence and to locate other users.

2.1 P2P Communications Systems

Because SIP places much intelligence in the endpoints and media usually flows directly between the devices, bypassing intermediaries. SIP is often argued to be a P2P application. However, current work to create "P2PSIP," including this work and the efforts of the nascent IETF P2PSIP working group, use the term P2P in the traditional academic sense. Not only can devices that locate each other communicate directly, but the centralized resources needed at runtime are reduced or completely removed, and instead the endpoints rely on distributed resource location, NAT traversal, and possibly even security primitives.

The most well-known P2P communications system is Skype [24], which offers free computer-to-computer calls and charges for computer to PSTN calls. Skype is a completely closed system, and most of what is known about it has been discovered by reverse engineering [1, 3]. The protocol used between Skype devices is encrypted, and the binaries are obfuscated, making this more difficult. Skype is a hybrid system, relying on centralized authentication servers to establish initial identity, and possibly for resource location. Skype uses a super-node architecture with media relays to enable NAT traversal. Peers are occasionally elevated to this status (often much to the user's surprise when bandwidth suddenly increases), and it is thought that Skype provisions its own nodes to act as super-nodes as well. While there have been some offerings from licensed third parties, Skype's proprietary nature means that in general, third parities cannot create Skype clients.

At about the same time Skype was unveiled, work was beginning on removing the central servers from the SIP architecture. In particular, two projects, the SIPpeer project at Columbia University [23] and the SOSIMPLE project at William & Mary [4, 5] have emerged. The two projects

have taken somewhat similar approaches to the problem, and each has experimented with a number of implementations.

Earlier versions of the SIPpeer project and the current SOSIMPLE implementation take an approach often called "P2P over SIP." Both are based on a modification of Chord, and use SIP messages to convey the actual DHT routing information between nodes, as well as for establishing communications sessions between users. Since SIP was designed to be extensible, both take advantage of this and use a modified SIP REGISTER message, traditionally to store location information with the central server, to exchange the DHT information. SIPpeer and SOSIMPLE differ in a number of aspects, the most critical being that SIPpeer is a recursive algorithm, with each node acting as a proxy of messages, whereas SOSIMPLE uses an iterative algorithm, each node redirecting searches instead. We will discuss this difference in more detail in Section 4.

More recently, the researchers at Columbia have also begun to explore an alternate technique that uses an external DHT, rather than using SIP messages and creating a new DHT [2]. OpenDHT [16] is being used as the underlying DHT. While this has some disadvantages for a deployed system, most notably requiring a second stack (often problematic on small embedded communications devices) and not working in completely disconnected networks, it allows for rapid deployment and testing of new features and techniques for P2PSIP research.

Additionally, there have been a few attempts at bringing P2PSIP in the commercial space as well, although it is often unclear what is meant by "P2P" in commercial offerings. Some of these offerings mean P2P in the earlier sense, simply that media flows between the devices directly. However a few companies claim to offer P2P systems, including Avaya's one-X and Popular Telephony's Peerio system. Avaya's product is believed to be based on a broadcast and mirror approach, rather than a DHT. Neither the Avaya or Peerio protocols have been openly discussed, so how they work is something of a mystery, and the products do not interoperate with one another. None of these systems is intended for Internet-scale deployment, but rather for small office scenarios, where removing complexity of a server, rather than scalability, is the key motivation.

2.2 NATs and P2P

Despite the efforts of the IAB and IETF, current Internet architecture now prominently relies on NATs to establish a multi-level network where groups of nodes (in a single home or business) share a single (or small number of) globally routable IP address. Behind the NAT private, non-routable addresses are used. NATs create particular challenges for any applications that require connections placed to endpoints—such as SIP or P2P applications, because the internal nodes cannot use what they believe to be their IP address as an identity on the public Internet.

The most common technique for connecting machines behind NATs is hole-punching [8], which relies on the behavior of most NATs to allow external connections on a port on which an internal node has already sent an outgoing message. P2P systems generally support have used some combination of hole-punching and relaying to achieve full communication.

The IETF has recently begun focusing more attention on NATs, resulting in standards and Internet-Drafts for NAT behavior [11], STUN to support hole-punching [19] and TURN for relaying [20]. A recent Internet-Draft, ICE [18], specifies how to combine these protocols to establish the best connection possible between two peers. NUTSS [10] is a proposal from the P2P community to use these tools to allow for NAT traversal in generalized P2P systems.

3. Challenges with NATs for P2P

Although there are still many practical issues with applying the existing NAT-traversal standards to P2P connections, significant progress has been made in addressing the hole-punching issues. Once connectivity is assured, however, the issue of preserving the security of the overlay in the presence of nodes that do not have a readily identifiable global IP address becomes significant.

3.1 Alternate Node-ID Generation

Traditional Chord generates Node-IDs by hashing IP addresses into a 160-bit space using SHA-1. This approach fails in the presence of NATs in systems. A machine running behind a NAT will have a private IP address. If two nodes communicating with one another are both behind NATs, it is actually possible (and in fact if both are home users using the same brand of NAT, likely) that both sides of the conversation will have the same private IP address, and thus attempt to use the same Node-ID. External entities attempting to verify that the IP address and the hashed Node-ID agree will fail, as they will instead see the source IP address as the public address of the NAT.

Techniques such as STUN can be used to to identify the public IP address of the NAT, but even in this case there is a problem. If more than one machine is located behind the NAT, then both machines share the same public IP address. Multiple machines behind the NAT will still hash this address to produce the same Node-ID. It is clear that the approach taken by Chord to this problem won't work.

A naive solution would be to append the port number to the address before hashing, producing a different Node-ID

for each node behind the NAT, but this approach is problematic. If a system is built using this approach, an attacker wishing to place themselves in a particular location in the overlay in order to attack information stored there could hash the 64K host/port pairs for that NAT's global IP address in advance, and then select from the resulting Node-IDs the closest one to the information to be attacked. If we assume that an attacker could choose from approximately 64K port choices for each IP, a network with several hundred thousand nodes could be attacked with a high probability of success with only a few available IP addresses.

Our solution to this problem is to generate an initial Node-ID by hashing only the node's globally routable IP address. Then we replace the least significant 16 bits 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. In this approach, a single user wishing to mount a Sybil attack can still produce many nodes, again one for each port and IP, but the nodes within one IP address will all be contiguous on the overlay—the last 16 bits of a 160-bit address space are rarely needed to separate the Node-IDs of randomly picked IP addresses. The result of narrowing the nodes to this region is that the portion of the ring (and thus the registrations they are responsible for storing) they are able to attack is greatly reduced.

Our approach is not foolproof, but it greatly increases the difficulties of generating arbitrary Node-IDs. An attacker that manages a large ISP or other entity would have access to multiple IP addresses, but we believe such an attack to be sufficiently unlikely as to be ignorable for practical purposes. If IPv6 is deployed, we may need to reconsider how IP addresses are mapped to Node-IDs, because if ISPs give end-users (or their NIC MACs) control over significant portions of their IPv6 address, we may need to reevaluate the portion of the IP addressed used for the Node-ID.

For smaller networks, a stronger scheme is to generate Node-IDs using an HMAC-SHA-1 scheme, where a shared secret between all nodes is used as a seed together with the IP address to generate Node-IDs. This allows the load of the DHT to be distributed randomly across all participating nodes, but scales only as far as a single shared secret can be distributed and kept secret.

3.2 Redundancy with NAT Node-IDs

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 an arbitrary number of sequential nodes can be obtained simply by selecting different port numbers. While the user cannot control exactly what information they compromise, a rogue node would be able to disrupt the information stored for some user, including the replicas.

In place of this approach, we instead use a replica key. When data is hashed to be stored, the keyword is hashed to form the primary location, as is normally done in Chord. In addition, the keyword concatenated with one or more replica strings (or numbers), and hashed again to find the redundant locations. These replica strings 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 using a consensus algorithm can also be used to detect compromised data stored in the overlay. Comparison algorithms in DHTs have been discussed by others, most notably [15, 17]

4. Iterative vs. Recursive Queries

One major decision that needs to be made in the design of a P2P routing algorithm is a decision about whether information will be routed through the P2P network in an iterative or recursive fashion. In general, most designs have opted for recursive techniques, arguing that they are more efficient, usually in the sense of reducing the total number of messages transmitted and reducing the corresponding latency between sending the messages and obtaining a response.

Our work in developing protocols for large scale deployments has shown the decision is not so clear. Recursive algorithms may only offer the improvements in efficiency under certain circumstances, and these may not exist in practice in networks with a large number of NATs. Iterative algorithms offer robustness to a number of security attacks that recursive algorithms are vulnerable to. Finally, a somewhat less efficient variation of a recursive algorithm may still pose security risks, but offers advantages in traversing NATs.

4.1. Defining Different Algorithms

In a recursive query, the node initiating the query sends a message to the node it knows that is nearest to the target. If that contacted node does not have the resource, it will forward the query to the nearest node to the target it knows, and the process repeats until the target is reached.

After sending the request forward, recursive algorithms can route the response in two ways. In the first, the response is symmetrically routed back through the overlay following the same path, a mechanism we call Recursive Overlay Response Routing (RORR). Alternatively, once the destination is reached, the response can be sent directly back to the

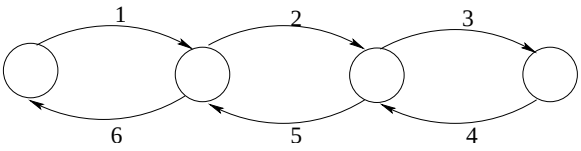

Figure 1. An RORR Query Flow

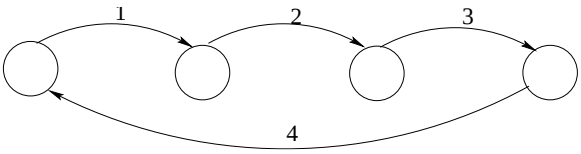

Figure 2. An RDRR Query Flow

originating node, which we call Recursive Direct Response Routing (RDRR). In Figures 1 and 2, we illustrate examples of the two recursive queries.

In contrast, an iterative query as shown in Figure 3 has the initiating node sending the query to an intermediary node. Rather than forwarding the message along, the intermediary instead replies directly to the initiator node with a nearer location. The initiator then sends a new request to the nearer node recommended. The process continues until the target node is reached.

Table 1 lists the total messages passed in a query involving i nodes (including the source and target). The RDRR technique is the most efficient in this case, but there are some drawbacks.

4.2. Implications for NAT Traversal

As discussed earlier, the introduction of NATs to the system lead to systemic non-transitivity in the network. Recursive algorithms offer a number of advantages when NATs are present. For each other node with which a node must communicate, NAT hole punching techniques must be used to establish to path between the nodes, and in many cases, keep alive messages need to be exchanged if the hole is to be maintained. With an RORR algorithm, a node establishes a connection with only the neighbor nodes, and has no need to establish new connections on the fly, since responses are routed back through the same connections.

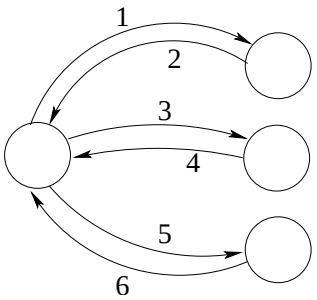

Figure 3. An Iterative Query Flow

In Table 1, we show the total number of new NAT punching connections that must be made by all nodes for the query, assuming every node is behind a NAT. We assume that the connections to immediate neighbors have already been established during periodic DHT maintenance, and do not include these in our count. We further assume that the first query is always sent to a neighbor, as is the case in systems such as Chord.

For RORR, the fact that routing occurs over the overlay and is proxied from node to node via closest neighbor means no new connections must be created at all. As long as nodes maintain connections to their neighboring nodes, there is no need to punch additional holes. With RDRR, the situation is similar, however in order to directly route the return message, a new connection will need to be established (except in the case that query source and target are direct neighbors), one additional connection must be established, and the cost of this hole-punched connection (which may greatly exceed the cost of a connection to an established neighbor) must be included in the cost of the RDRR routing.

For an iterative algorithm, many more connections must be established. While the first node where a message is sent is the direct neighbor of the query source, all subsequent connections are with non-neighbors, and a NAT connection must be opened. In all, $i - 1$ new connections must be established.

Clearly in networks where NATs are the norm, a recursive technique reduces the number of new connections that must be made through the NATs to enable communications, with RORR being the most efficient.

4.3 DoS Attacks and Parallelization

Recursive routing simplifies NAT traversal, but introduces the hazard of DoS amplification attacks. An Internet-scale P2P communications system may have hundreds of thousands of nodes, but because each resource is unique, searches must be completed all the way to the node containing the registration, rather than only to the nearest replica as in other P2P applications. The result is that each node in the DHT expects to route many searches on behalf of nodes with which they have had no previous contact, and those searches will ultimately terminate at nodes with which they have never contacted.

In such a system, recursive algorithms allow other nodes to serve as a surrogates for DoS attacks against a particular node. Assume an attacker A can send requests to intermediate nodes I searching for target node T. In an iterative case, the intermediate node I replies with a close node to use to

Table 1. Comparison of costs associated with various routing algorithms

Algorithm	Total Messages Passed	Number of NAT Holes to be Punched	Messages Processed by Interior Nodes	DoS Risk?
RORR	$2(i-1)$	0	4	Yes
RDRR	i	1	2	Yes
Iterative	$2(i-1)$	$i-1$	2	No

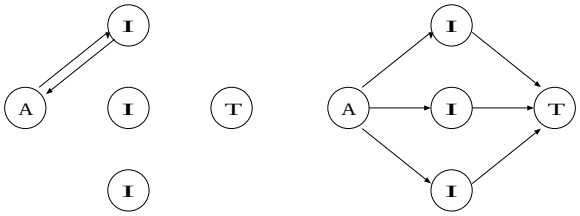

Figure 4. Surrogate Attackers in a Recursive Approach

attempt to reach T. The attacker is still forced to directly send to T if they wish to interfere with that node. In the recursive approach, attacker A can send to many intermediate nodes I, all of which will relay the message, eventually reaching T. In effect, the intermediate nodes have served as a DoS surrogate for A and may assist in obfuscating the source of the attack. The ability of A to multiply the attack is limited only by the number of intermediate nodes I that are reachable from A.

If we take Chord as an example, each node has $lg(m)$ immediate neighbors, where m is the size of the address space. If the attacker sends to every neighbor, $lg(m)$ messages will reach the target. For Chord, m is 160. All of these messages will appear to come from separate sources, requiring additional network processing and making blocking the attack more difficult than if the messages come directly from one attacking node.

Many algorithms, such as EpiChord or Kademlia [13, 14], improve average case performance by doing parallel searches (although these are iterative for a number of reasons, including the one outlined below). Parallelizing in a recursive algorithm would lead to an immense DoS multiplication factor, and tremendous message overhead. If a node sends the request to k neighbors, and a recursive algorithm parallelizes by sending to on average p nodes at each level of the search, and further assuming that queries have a worst case depth of $lg(N)$, the number of messages M_T arriving at the target T can in the worst case be:

$$M_T = k\, p^{lg(N)} \tag{1}$$

For even small values of k and small branching factors such as the $p = 3$ used by EpiChord, taking a recursive approach is impractical. If we assume a network of 100,000, and a non-hostile query with $k = 3$, we have:

$$M_T = 3\, 3^{16} \approx 129 \text{ million} \tag{2}$$

Clearly parallelization cannot be combined with a recursive algorithm.

4.4 Fairness

Additionally, recursive algorithms present a fairness problem. As shown in Table 1, and can be seen by examining Figures 1–3, intermediate nodes in an RORR network either send or receive 4 messages each, while the source and target only deal with 2. Both the RDRR and Iterative algorithms require intermediate nodes to handle only two messages, reducing the burden on uninvolved parties. This is particularly important in communications networks, since these devices (often phones) have limited processing power.

In short, neither iterative nor recursive routing is a universal solution. In a small or closed system, where attacking nodes are prevented by other means, the most efficient scheme should be chosen, which is RORR or RDRR depending on whether the searching node is behind a NAT or firewall (that information can easily be indicated in the query). In open Internet-scale systems, despite its cost, iterative routing must be used to defeat DoS amplification attacks. A more advanced system might determine which approach to use at run-time depending on its current load.

5. Using an Offline CA for Security

One technique we have been exploring for securing an overlay network involves having a centralized certificate authority (CA) issue a certificate to each user and node in the overlay. Without a CA, public-key cryptography can be used to establish repeated identity (once you communicate with a user for the first time or exchange keys outside the system, you can be certain you contact the same user), but it cannot be used to establish initial identity, nor does it effectively address node security. The central CA, even when used completely offline, helps to solve these issues.

5.1 Problems Certificates Can Address

In communications systems, the source and destination of information is not arbitrary. In many system, an individual usually doesn't care what the origin of information (usually a file) is, or which users wish to obtain a copy. Multiple copies of a particular piece of information may exist in the system, and in generally all are of equivalent value. In a communications system, communication is directly between two individuals, who need to be reachable in a deterministic way.

A user needs to be able to repeatably identify a user of the system. A user does not want conversations with "Alice" to be with two different individuals depending on who is logged in and claiming that identity. This means that a user joining the system must have some way of uniquely asserting over a prolonged time and spanning entering and leaving the overlay that they are a particular individual. Ensuring such identity in the absence of any centralized servers is a very difficult problem.

Similarly, communications systems need to have the ability to leave messages (text or other media) for offline users. Since the messages will be stored by arbitrary nodes, it is highly desirable to be able to encrypt the messages. Encrypting media flows between parties is also desirable.

While not a problem that is unique to communications systems, Sybil attacks [7] involve one entity attempting to masquerade as multiple nodes in the overlay. Certificates issued by a central authority can help detect and mitigate these attacks

5.2 Certificate Mechanism Structure

To solve both of these problems, we introduce a centralized certificate authority (CA). This server is not consulted on every transaction, but only when a new user (or node) wishes to join the overlay. The CA can issue certificates certifying several things (either separately or in a combined certificate):

- That the person possessing the certificate is a valid user with the specified username. The server tracks and ensures all usernames are unique. That is, it never issues certificates to two users with the same username.
- That the node bearing the certificate is authorized to join the system, and has the Node-ID specified in the certificate. This Node-ID is a random number, not tied to IP as in Chord. The Node-IDs are guaranteed unique by the server.
- Certificate chains can be supported if the CA grants a certificate to intermediaries responsible for certifying nodes and users that belong to a particular domain, such as the names of users of the form user@domain.edu, and for nodes in the address space controlled by that university. In this way, each administrative domain can be operated separately from the central CA.

A user must obtain a certificate for their username and their node (this could be one certificate asserting both) before joining the system. For architectural reasons, we allow the certificates to be separate. A particular device, for example an IP phone, might have multiple users or "lines," requiring several User-IDs to be asserted, but only one Node-ID. A user may be present on multiple devices, requiring multiple Node-IDs but only one User-ID. Finally, devices such as gateways, which allow VoIP calls to terminate onto the PSTN might not be associated with a user, but might still be a peer in the network. The certificates need to be obtained from the CA before joining the network.

In addition to having their own certificates, each node in the system has a copy of the public key for the CA, obtained directly from the CA at the time the CA issued the certificate. Any new node or user wishing to join the system will be expected to present a certificate signed by the CA, and any node will be able to validate the authenticity of the certificate using their copy of the CA's public key.

The public portion of the certificate for each user is stored as a file in the overlay. These certificates persist, being transferred from one node to another as nodes enter and leave the system. These public certificates are available at all times, even when the user is offline, which allows secure messages (voicemail) to be left for the user.

5.3 Securing the System with Certificates

These certificates allow a user to uniquely assert that they are entitled to a particular user name within the system. This helps to eliminate the namespace collision problem, and allows users to ensure the individual they are communicating with is in fact the asserted user. Communication between users (online or offline) and nodes can be secured using the certificates.

This approach can also be used to protect against Sybil attacks. When a node tries to join, it must present a valid certificate for the Node-ID it is attempting to use, and the node currently responsible for that region of the overlay must verify the certificate matches the Node-ID presented. The CA can rate limit Sybil attacks using a number of mechanisms. Perhaps the simplest mechanism would be to charge a minimal cost for a certificate. Others might include requiring a valid credit card number, that while not charged, would not be allowed to obtain more certificates for a given period, perhaps 24 hours. While this would be inconvenient, it would effectively prevent Sybil attacks. While the certificates may be very low in cost, obtaining the hundreds, thousands, or even millions required to corrupt an overlay

would be very costly. Because the Node-IDs are randomly selected by the CA, it is very difficult to place a node in a particular location.

This approach is arguably a hybrid approach. The server involved in this case only has to be involved at the time the certificate is issued (presumably when a new user joins the system and some periodic period after certificates expire). The CA doesn't need to be consulted at any other time during the interaction of the nodes of the system. Unlike a centralized authentication server such as used in Skype [1], this doesn't represent a central point of failure, although it is arguable that such a system does have a central entity in charge, which might go against some ideas about P2P.

6. Conclusion

We have presented solutions to several of the problems we have encountered moving DHT-based P2P communications systems from the lab to a real-world setting with combinations of residential, commercial, and academic systems. In short, the P2P algorithms must be chosen based on the control exerted over participation in the network.

- If a CA-based system can be used to establish identity and monitor node additions, then routing algorithms can be chosen based on efficiency (generally based on the prevalence of NATs and firewalls in the environment) and Node-ID selection can be assigned with a simple uniform random distribution. However, if policing of node behavior is unavailable or nodes are easy to add, iterative routing will still be needed.
- For a more open system, the choices are more restricted. Node-IDs must be chosen using a limited scheme such as we present in Section 3 to control the ability of nodes behind NATs to attack the DHT. In such an environment, routing must be done iteratively to avoid DoS amplification attacks.

References

[1] S. Baset and H. Shulzrinne. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. In *INFOCOM2006*, Apr. 2006.

[2] S. Baset and H. Shulzrinne. Using an external DHT as a SIP location service. Technical report, Feb. 2006.

[3] P. Biondi and F. Desclaux. Silver Needle in the Skype. Presentation at Black Hat Europe 2006, Feb. 2006.

[4] 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.

[5] D. A. Bryan, B. B. Lowekamp, and C. Jennings. draft-bryan-sipping-p2p-02, Mar. 2006.

[6] B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, and D. Gurle. RFC 3428 - session initiation protocol (SIP) extension for instant messaging, Dec. 2002.

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

[8] B. Ford, P. Srisuresh, and D. Kegel. Peer-to-peer communication across network address translators. In *Proceedings of USENIX 2005*, pages 179–192, Anaheim, CA, April 2005.

[9] M. Freedman, K. Lakshminarayanan, S. Rhea, and I. Stoica. Non-Transitive Connectivity and DHTs. In *Proceedings of the 2005 Usenix Workshop on Real, Large-scale Distributed Systems (WORLDS05)*, 2005.

[10] S. Guha, Y. Takeda, and P. Francis. NUTSS: A SIP-based Approach to UDP and TCP Network Connectivity. In *Future Directions in Network Architecture (FDNA-04)*, SIGCOMM '04 Workshops, Portland, OR, Aug. 2004.

[11] IETF behave Working Group. `http://www.ietf.org/html.charters/behave-charter.html`.

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

[13] B. Leong, B. Liskov, and E. Demaine. Epichord: Parallelizing the chord lookup algorithm with reactive routing state management. In *Proceedings of the 12th International Conference on Networks (ICON 2004)*, Nov. 2004.

[14] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the XOR metric. In *Proceedings of IPTPS02*, Mar. 2002.

[15] A. Muthitacharoen, S. Gilbert, and R. Morris. Etna: A fault-tolerant Algorithm for Atomic Mutable DHT Data. Technical Report MIT-LCS-TR-993, MIT-LCS, June 2005.

[16] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, and J. Hellerstein. OpenDHT: A Public DHT Service and Its Uses. In *Proceedings of ACM SIGCOMM 2005*, 2005.

[17] R. Rodriques and B. Liskov. Rosebud: A scalable byzantine-fault-tolerant storage architecture. Technical Report TR/932, MIT CSAIL, Dec. 2003.

[18] J. Rosenberg. draft-ietf-mmusic-ice-08, Mar. 2006.

[19] J. Rosenberg, C. Huitema, R. Mahy, and D. Wing. RFC 3489 - Simple Traversal of UDP Through Network Address Translators (NAT) (STUN), Feb. 2006.

[20] J. Rosenberg, R. Mahy, and C. Huitema. draft-rosenberg-midcom-turn-07, Feb. 2005.

[21] J. Rosenberg and H. Schulzrinne. RFC 3263 - Session Initiation Protocol (SIP): Locating SIP Servers, June 2002.

[22] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. RFC 3261 - SIP : Session initiation protocol, June 2002.

[23] K. Singh and H. Schulzrinne. Peer-to-peer internet telephony using sip. In *Proceedings of the 2005 Network and Operating System Support for Digital Audio and Video (NOSSDAV 2005)*, 2005.

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

[25] 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*, 11(1), 2003.