Structured Peer-to-Peer Control Plane

Khashayar Khavari, Nadeem Abji, Ramy Farha, Chuen Liang, Ali Tizghadam, Farid Fadaie, Alberto Leon-Garcia
Dept. of Electrical and Computer Engineering
University of Toronto, Ontario, Canada
Email: {khashayar.khavari, nadeem.abji, ramy.farha, chuen.liang, ali.tizghadam, farid.fadaie, alberto.leongarcia}@utoronto.ca

***Abstract*— Peer-to-Peer (P2P) systems have witnessed an explosive growth in popularity due to their desirable characteristics (robustness, scalability, availability). In this paper, we present an approach to bring these characteristics into the control plane of IP networks, which mainly relies on signaling protocols such as SIP to setup multimedia and instant messaging sessions. We present a structured P2P control plane based on modifications to the original Chord P2P topology, resulting in a hierarchical overlay of SIP peers that replaces traditional client-server paradigms in control plane signaling protocols. Implementations were used to study the performance of the proposed structured P2P control plane, and its suitability for use in IP networks.**

I. Introduction

Next-generation service providers (SPs) offering an ever-increasing number of services need to find effective ways to reduce the costs of operating their networks, in order to become more profitable. Traditionally, telecommunication networks have been seen as consisting of three planes [1]: a data plane, a signaling / control plane, and a management plane. Although these three planes are logically separated, they can be in the same physical network, and their operations greatly determine the expenses incurred by SPs. The Session Initiation Protocol (SIP) [2] is currently expected to provide the signaling function in the control plane of next generation services that will be delivered over IP networks (voice, video, instant messaging). A new application of particular relevance to SPs is the use of Peer-to-peer (P2P) [3] technology to provide the "control plane" functionality that is central to many services. P2P systems are appropriate for control plane system implementation, since they exhibit many self-managing capabilities, which contribute in reducing expenses incurred by SPs to operate their networks.

In this paper, we propose the use of the P2P approach for control plane, by proposing a layered architecture and implementation of the SIP-based control / signaling plane over P2P. The approach is based on SIP over Chord [4], a structured P2P approach chosen because of its main characteristics (simplicity, provable performance, and provable correctness). However, and due to the observed benefits of the hierarchical P2P systems in unstructured approaches [5], we will use the notion of hierarchy in regular Chord. For the rest of this paper, we refer to this modification of Chord as hierarchical Chord (**h-Chord**), where some peers of the original Chord ring are elevated to the role of Super-Peers connected through a ring. The improvements that this approach brings to the P2P control plane are explored through detailed experiments using implementations performed. In addition, the validity of our approach to provide SIP signaling for the control plane over a P2P substrate is shown.

The rest of this paper is structured as follows. Section II briefly summarizes the related work and its relevance to this paper. Section III presents our design of SIP over P2P, justifying the decisions made. Section IV shows the detailed operation of SIP over P2P using **h-Chord**. Section V summarizes the experiments performed to evaluate the performance of **h-Chord**, along with some typical results obtained. Finally, Section VI concludes this paper.

II. Related Work

Prior to detailing our approach, we review some related work. Extensive amount of research has been done independently in the areas of P2P and SIP, but little work has been done on the capabilities of a combined P2P-SIP architecture.

P2P file-sharing applications have seen an explosive growth over the last few years. P2P technology, however, is not limited to file-sharing applications, but can also implement gaming, storage and processing applications. However, the main difference between the original P2P approaches for file sharing and those emerging for control plane implementations is that SPs require a more stringent performance level than that offered by widely available best-effort P2P applications. In P2P networks, the resulting interconnected set of peers forms an overlay network. P2P approaches can be categorized as: unstructured and structured. This depends on the placement of nodes in the overlay topology, and on how the lookup is performed to locate desired resources. In unstructured P2P approaches such as Freenet [6], the process of finding a match to a query for a resource is essentially a random search through the overlay network. In structured P2P approaches such as DHTs [4], [7], [8], the process of finding a match to a query for a resource has a more predictable performance since the overlay topology is tightly controlled and the placement of resources is in precise locations in the overlay. In structured P2P approaches, Pastry [7] and Tapestry [8] are alternatives that can be used as well, but Chord [4] was chosen because of its simplicity, provable correctness and performance.

A variant of the original flat P2P model, in which all peers were part od the overlay network, hierarchical P2P models have been developed. In hierarchical P2P models, Super-Peers are interconnected to form an overlay network. The hierarchical model recognizes the heterogeneity of peers in terms of communications and processing resources and adaptively elevates peers to the role of Super-Peer. Hierarchical

1-4244-0355-3/06/$20.00 (c) 2006 IEEE

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2006 proceedings.
Authorized licensed use limited to: The University of Toronto. Downloaded on December 9, 2009 at 02:04 from IEEE Xplore. Restrictions apply.

P2P systems can scale to very large size, but are vulnerable to faults in Super-Peers. Super-Peers have mainly been presented for unstructured P2P models [9], but have rarely been used for structured P2P models [5].

In control plane protocols used for signaling, SIP [2], backed by IETF, has recently been gaining ground because of its extensibility, flexibility, and text-based format. SIP provides support for new services, operates with low complexity, and possesses an efficient addressing scheme based on Uniform Resource Identifiers (URIs). The combination of SIP and P2P has received little attention until recently. In [10], SIP with DHTs was presented, where P2P messages are sent over the SIP infrastructure, which is the major difference between this approach and the one we present in this paper. The major disadvantage with [10], is that SIP messages are used for P2P operations, hence restricting the use of the approach to SIP. Since new headers in SIP messages have to be supported by any SIP UA attached to the network, adaptors are needed to make any translation in an environment where new SIP User Agents (UAs) are to be introduced. In [11], a P2P approach to SIP registration is presented, by adding headers and options to the original SIP messages. Again, the major disadvantage with this approach is that it modifies SIP semantics. In this paper, we aim to use P2P capabilities to complement SIP. SIP exhibits several P2P capabilities already, since the media delivery is performed in a P2P fashion. The bottleneck is in the SIP control plane, where central servers are used which can become overloaded and even fail. Hence, we aim to make SIP benefit from P2P's characteristics without any changes at the SIP layer, so our solution can potentially be used with all SIP-compliant UAs while remaining compatible with client-server solutions. Furthermore, our design can be easily modified for use with any other signaling protocol.

III. Structured P2P Control Plane Design

The structured P2P control plane architecture is intended to be modular. Unlike the previous attempts to integrate SIP and P2P [10], we aim to achieve modularity, where both the signaling protocol of the control plane (SIP) and the P2P topology can be replaced without affecting the overall design. Thus, we adopt a layered architecture, where the "SIP" layer has an underlying "P2P substrate" layer replacing the traditional client-server communication model. The SIP servers' functions (Proxy, Redirect, Registrar) are virtualized, distributed, and performed collectively by the peers in the P2P substrate. For the structured P2P substrate, we use an approach based on Chord [4], mainly because of its simplicity, provable performance, and correctness.

A major design choice to make is on the topology of the P2P substrate to use for the control plane. It is clear that there are several alternatives to the traditional client-server paradigm. Namely, the three possibilities are shown in Fig. 1:

1. SIP Servers Structured P2P: This is the most straightforward method to add some of P2P's benefits to traditional client-server control planes. In this design, the SIP servers form a structured P2P ring, and communicate directly with

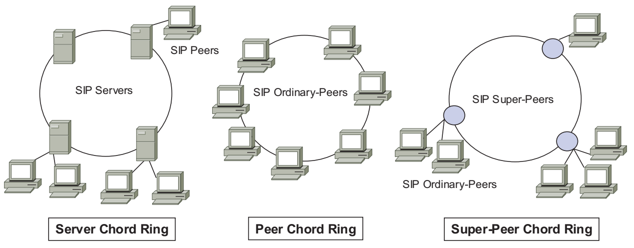


Fig. 1. Design Alternatives

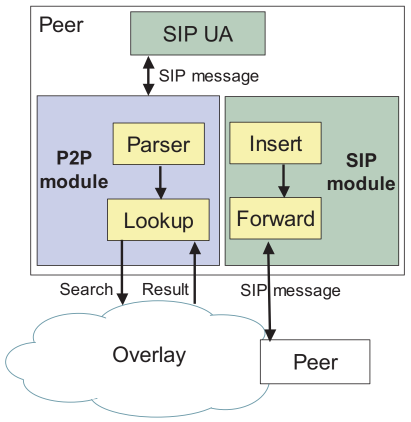


Fig. 2. Typical Node Architecture

the peers connecting to them. The problem with this model is that SPs are not harnessing the wealth of resources at some of their SIP peers.

2. SIP Peers Structured P2P: This is the method that benefits the most from the P2P capabilities since it involves all SIP Peers in the control plane. This model harnesses the wealth of resources at all clients, however, the problem is that this model is not likely to be used by SPs for commercial purposes, mainly due to security reasons and to the little control SPs exert over the operations.

3. SIP Super-Peers Structured P2P: This is the method used in this paper to design the P2P control plane. The Super-Peers form the structured P2P ring, to which peers connect directly, each of which have a capacity regulating how many peers can connect to them before the need arises to promote new peers to the role of Super-Peers. SPs control which Ordinary-Peers are allowed promotion to the role of Super-Peers based on predefined criteria. This method is based on the original Chord [4] topology, with modifications to benefit from the advantages shown by the notion of Super-Peers, leading to our approach, which we refer to as hierarchical Chord (**h-Chord**).

Another major design choice is to decide on the mechanism to use in order to combine SIP with the P2P substrate. A peer in the structured P2P control plane has both a "SIP module", and a "DHT module". Note that our approach is valid even if the "SIP module" is in a separate peer (different physical peer). Fig. 2 shows the internal architecture of a typical peer.

In the "DHT module", each peer (Super or Ordinary) has a

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2006 proceedings.
Authorized licensed use limited to: The University of Toronto. Downloaded on December 9, 2009 at 02:04 from IEEE Xplore. Restrictions apply.

unique m-bit Peer ID. Thus, such an approach allows for up to 2^m peers to be included in the ring. The Peer ID is generated using the SHA-1 [12] consistent hashing function (modulo 2^m), using the IP address as an input to the hash function to generate a 160-bit identifier. The Resource ID is calculated by hashing the SIP URI of the SIP UA running on the peer, again using the SHA-1 hashing algorithm resulting in the same hash space as that of Peer ID. As in the original Chord, each Super-Peer in **h-Chord** maintains data for Resource IDs close to its Peer ID. Specifically, Resource ID k will be stored at the first node with Peer ID greater than or equal to k (clockwise around the circle). This peer is referred to as the successor of k. Each peer maintains a finger table. The i^{th} entry in the finger table of a peer with Peer ID n is the successor peer to the Resource ID $(n + 2^{i-1})$. Peers in the ring use their finger tables to forward query messages to their destination. With each forwarded message, a peer can advance a query halfway around the ring, thus resulting in an $O(log(n))$ bound on the number of hops needed for queries. An advantage of **h-Chord** is that each peer must only maintain information about a small subset of the other peers in the network. More details on **h-Chord** will be presented in the next section.

The interaction between the "DHT module" and the "SIP module" is regulated by the following sequence of events:

1. The SIP User Agent (UA) on a peer generates a SIP message (REGISTER, INVITE, MESSAGE), which it sends to a "virtual" proxy, located on the same peer, listening on a well-known port. This preserves SIP transparency to the SIP UA at the sender. Note that due to the modularity of the design, the SIP UA does not have to be on the same peer as the "DHT module". This step is the same regardless of whether the peer is a Super or an Ordinary peer.

2. The "DHT module" receives the SIP message, and the Parser component parses the contents of the header of the SIP message, in order to identify the destination of this SIP message, normally identified using the **To:** field in the SIP message's header. This step is the same regardless of whether the peer is a Super or an Ordinary peer.

3. The Lookup component performs the lookup function, by hashing the SIP URI obtained from the **To:** field to obtain the Resource ID to find. If the peer is a Super-Peer, the lookup operation is similar to that of a regular Chord lookup. If the peer is an Ordinary-Peer, the lookup function is performed by its Super-Peer. As a result, the DHT module receives information about the Super-Peer storing the Resource ID and hands it over to the "SIP module".

4. The "SIP module" inserts a **Via:** field in the header of the original message, to ensure the return path of future SIP messages traverses the "virtual" proxy, hence preserving SIP transparency to SIP endpoints. The modified SIP message is forwarded directly to the SIP UA peer.

Hence, messaging can be seen as consisting of two main types: DHT Messaging, which consist of the messages needed to maintain the P2P substrate and perform lookups, and SIP Messaging, which consist of the SIP messages sent over the P2P substrate.

TABLE I
PSEUDO-CODE NOTATIONS OF STRUCTURED P2P CONTROL PLANE

`Finger[x].start`	Entry in the finger table with index x peer p: $(p + 2^{k-1}))mod2^m, 1 \leq k \leq m$
`Finger[x].peer`	First node greater than or equal to entry x in the finger table of the peer
`High_child (Super-Peer)`	Returns the child of the Super-Peer with the highest capacity
`Peer.capacity`	Denotes the capacity of the peer, i.e. the number of connections that it can support
`Peer.role`	Either Super-Peer or Ordinary-Peer
`Peer.add(child)`	Add child to peer's child list, when peer is a Super-Peer to which child connects
`Super-Peer(Peer)`	Super-Peer to which peer is connected, where peer is an Ordinary-Peer
`Predecessor(Peer)`	Predecessor of peer
`Successor(Peer)`	Successor of peer

IV. STRUCTURED P2P CONTROL PLANE OPERATIONS

The operations needed to make SIP over P2P possible can be categorized in three main categories: Peer Registration, User Registration, and Periodic Maintenance. In this section, we will explain in details each of the three, and show pseudo-codes for the main functions needed. Table I shows some definitions that will be used in the rest of the paper.

A. Peer Registration

When a SIP peer wishes to join the network, it needs an attachment point, i.e. a bootstrap. This bootstrap should be a peer that already belongs to the network, and that will provide the incoming peer with the capability to reach other peers in this network. Several methods exist to allow bootstrapping [10]. In this paper, when a SIP peer joins the network for the first time, it will use the bootstrap to provide it with an attachment point. After the SIP peer joins and is assigned a location in the ring (if Super-Peer), or a Super-Peer to connect to (if Ordinary-Peer), this information is cached at the SIP peer, so that for future joins, it can attempt to join by attaching to the location where it was positioned last time. We expect this approach to reduce delay between the time a SIP peer joins the network, and that at which it is placed in the appropriate position in the structured P2P control plane.

When a new SIP peer joins the network, it will be an Ordinary Peer. If that SIP peer is the first peer in the network, then it is promoted to the role of Super-Peer. Otherwise, the SIP peer uses a bootstrap as an attachment point. If the bootstrap is an Ordinary Peer, then it refers the new SIP peer to its Super-Peer. The hash result of the new SIP peer's IP address is used to find its Peer ID, so that its position in the structured P2P control plane topology is determined. Once its Super-Peer is found, the new SIP peer is added as a child.

However, if the capacity of the SIP peer's chosen Super-Peer is exceeded, then one of its children needs to be promoted to the role of Super-Peer. The promoted Super-Peer's finger table, its predecessor and successor information, need to be

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2006 proceedings.
Authorized licensed use limited to: The University of Toronto. Downloaded on December 9, 2009 at 02:04 from IEEE Xplore. Restrictions apply.

```
Result: Join
// SIP Peer p joins the network
// SIP Peer p' is an arbitrary peer serving as bootstrap
p.join(p')
if p' then
    // SIP Peer p is not the first, so p is an Ordinary Peer
    if p' is Ordinary-Peer then
        // Refer to the Super-Peer of SIP Peer p'
        p' = Super-Peer(p');
        // If SIP Peer p' is a Super-Node, continue
        Super-Peer(p) = p'.find_successor(p);
        // If capacity of p's successor (Super-Peer) reached
        if Super-Peer(p).capacity = max then
            // Promote highest capacity child
            Promote(High_child(Super-Peer(p),
            Super-Peer(p)));
            // If successor capacity is less than the
            maximum, continue
        else
            // If capacity of p's successor (Super-Peer) not
            reached
            Super-Peer(p).add(p);
        end
    end
else
    // If SIP peer p is the first
    p.role=Super-Peer;
    // SIP Peer p has to become a Super-Peer
    for i=1 to m do
        // Fill in the finger table entries of p
        Finger[i].peer = p;
    end
    // Set the predecessor of SIP Peer p to p
    predecessor(p) = p;
end
```

Algorithm 1: Pseudo Code for SIP Peer Joining

```
Result: Promote
// Promote SIP peer p knowing its successor
promote(p,successor)
// Initialize finger table of p using successor's finger table as
starting point
p.initialize_finger_table(successor);
// Update other finger tables
p.update_others();
move Ordinary-Peers in (predecessor, p]
from successor;
```

Algorithm 2: Pseudo Code for SIP Ordinary-Peer Promotion

```
Result: Find_successor
// Ask SIP peer p to find successor of a SIP peer with ID $p_{id}$
p.find_successor($p_{id}$)
p'=find_predecessor(p_id);
return successor(p')
// Ask SIP peer p to find p_id's predecessor
p.find_predecessor($p_{id}$)
p'=p;
while $p_{id} \in$ (p', successor(p')] do
    p' = p'.closest_preceding_finger ($p_{id}$);
end
return p';
// Return closest finger preceding $p_{id}$
p.closest_preceding_finger($p_{id}$)
for i=m to 1 do
    if finger[i].peer $\in$ (p, $p_{id}$) then
    end
    return finger[i].peer;
    return p;
end
```

Algorithm 3: Pseudo Code for Finding Successor

```
Result: Stabilize
// Periodically verify peer p's immediate successor and tell
the successor about p
p.stabilize ()
x = predecessor (successor(p));
if x $\in$ (p, successor(p) then
    successor(p) = x;
end
successor(p).notify(p);
// Peer p' thinks it might be the predecessor
p.notify (p')
if predecessor (p) is nil or p' $\in$ (predecessor(p) , p) then
    predecessor(p) = p';
end
// Periodically refresh finger table entries
p.fix_fingers ()
i = random index > 1 into finger[];
finger[i].peer = find_successor
(finger[i].start);
```

Algorithm 4: Pseudo Code to Stabilize

initialized. Fingers and predecessors of existing peers are updated to reflect addition of the newly Super-Peer. The SIP peers associated with keys that the newly promoted Super-Peer is now responsible for, are transferred to him. Algorithm 1 shows the pseudo-code for the operation of a new SIP peer joining, while Algorithm 2 shows the pseudo-code for the promotion of an Ordinary-Peer.

B. User Registration

When a SIP User Agent (UA) is added to the P2P substrate, the SIP URI is hashed to obtain the Resource ID so that the Super-Peer storing the user's registration is determined. The obtained Resource ID is stored at the appropriate Super-Peer in the ring, by finding its successor. The SIP user registration (successor finding) operation is shown in Algorithm 3.

C. Periodic Maintenance

In addition, maintenance actions are performed regularly by running a stabilization algorithm to ensure the finger tables at the Super-Peers are correctly maintained, and keeping the Super-Peers' successor pointers up to date. The stabilization operation is shown in Algorithm 4.

Authorized licensed use limited to: The University of Toronto. Downloaded on December 9, 2009 at 02:04 from IEEE Xplore. Restrictions apply.

```
Result: Initialize finger table
// Initialize finger table of local SIP peer
// SIP peer p' is an arbitrary peer serving as bootstrap
p.initialize_finger_table(p')
finger[1].peer =
p'.find_successor(finger[1].start);
// Insert SIP peer p
predecessor(p) = predecessor(successor(p));
// Set SIP peer p as predecessor of successor
predecessor(successor(p)) = p;
for i=1 to m-1 do
    if finger[i+1].start ∈ [p , finger[i].peer) then
        finger[i+1].peer = finger[i].peer;
    else
        finger[i+1].peer =
        p'.find_successor(finger[i+1].start);
    end
end
Result: Update others
// Update all SIP peers whose finger tables should refer to p
p.update_others()
for i = 1 to m do
    // Find last SIP peer p" whose $i^{th}$ finger might be p
    p''=find_predecessor($p - 2^{i-1}$);
    p''.update_finger_table(p,i);
end
Result: Update finger table
// If s $i^{th}$ finger of SIP peer p, update p's finger table with s
p.update_finger_table (s, i)
if s ∈ [p, finger[i].peer) then
    finger[i].peer = s;
    // Get first SIP peer preceding p
    p = predecessor(p);
    p.update_finger_table (s, i);
end
```

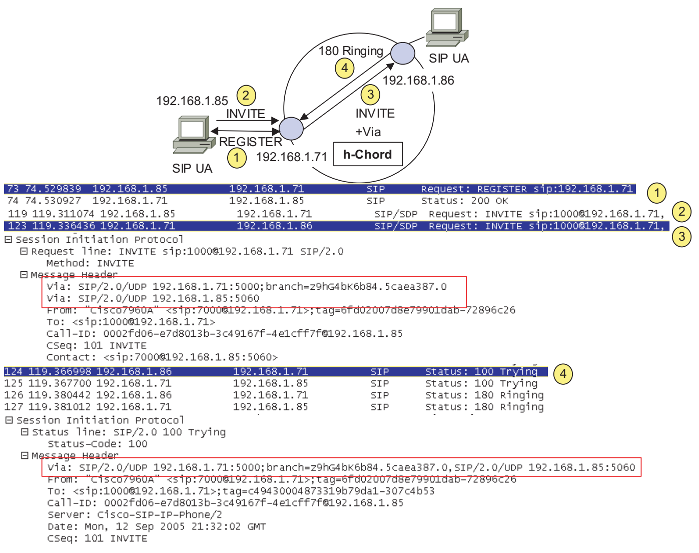
Algorithm 5: Pseudo Code for other operations needed by Join, Promote, and Stabilize

V. Experimental Results

Several experiments were performed to check the validity and to quantify the performance of the proposed structured P2P control plane approach. We will use actual implementations in the Network Architecture Labs at the University of Toronto, using IBM's BladeCenter with 56 blades consisting of 2 Xeon 2.8GHz processors each with 2GB of RAM. SIP peers using **h-Chord** for the "DHT module" are implemented using the C language. In the following experiments, we do not measure the absolute delay for the search and stabilization, since these numbers are not meaningful given SIP peers used for our experiments are collocated and not distributed over the Internet, as will be the case for a typical application. Instead we use normalized metrics such as number of hops.

In the first experiment, we tested whether the proposed SIP over P2P approach results in successful call establishment. For this purpose, we use any SIP phone (for instance, Cisco IP Phone 7960) and attempt to establish a call over the P2P substrate. Fig. 3 shows a snapshot of the setup, and the call establishment procedure using the **h-Chord** substrate, with Ethereal [13] captures to show the packets exchanged.

As seen in Fig. 3, the REGISTER message is first sent by the SIP UA @192.168.1.85. The SIP URI is extracted, hashed using SHA-1, and stored in its corresponding successor in the ring. Then, when a call is to be initiated over P2P from the SIP UA @192.168.1.85 to the SIP UA @192.168.1.86, the INVITE message is sent to the Super-Peer @192.168.1.71. This Super-Peer extracts the **To:** field in the INVITE header, and using its hash value, inserts the **Via:** field in the header, as seen in the Ethereal captures. The INVITE message is forwarded to the destination, and the 180 Ringing SIP message is returned, indicating successful call establishment.



```
73 74.529839  192.168.1.85          192.168.1.71         SIP       Request: REGISTER sip:192.168.1.71
74 74.530927  192.168.1.71          192.168.1.85         SIP       Status: 200 OK
119 119.311074 192.168.1.85          192.168.1.71         SIP/SDP  Request: INVITE sip:1000@192.168.1.71,
123 119.336436 192.168.1.71          192.168.1.86         SIP/SDP  Request: INVITE sip:1000@192.168.1.71,
⊟ Session Initiation Protocol
  ⊟ Request line: INVITE sip:1000@192.168.1.71 SIP/2.0
      Method: INVITE
  ⊟ Message Header
      Via: SIP/2.0/UDP 192.168.1.71:5000;branch=z9hG4bK6b84.5caea387.0
      Via: SIP/2.0/UDP 192.168.1.85:5060
      From: "Cisco7960A" <sip:7000@192.168.1.71>;tag=6fd02007d8e79901dab-72896c26
      To: <sip:1000@192.168.1.71>
      Call-ID: 0002fd06-e7d8013b-3c49167f-4e1cff7f@192.168.1.85
      CSeq: 101 INVITE
      Contact: <sip:7000@192.168.1.85:5060>
124 119.366998 192.168.1.86          192.168.1.71         SIP       Status: 100 Trying
125 119.367700 192.168.1.71          192.168.1.85         SIP       Status: 100 Trying
126 119.380442 192.168.1.86          192.168.1.71         SIP       Status: 180 Ringing
127 119.381012 192.168.1.71          192.168.1.85         SIP       Status: 180 Ringing
⊟ Session Initiation Protocol
  ⊟ Status line: SIP/2.0 100 Trying
      Status-Code: 100
  ⊟ Message Header
      Via: SIP/2.0/UDP 192.168.1.71:5000;branch=z9hG4bK6b84.5caea387.0,SIP/2.0/UDP 192.168.1.85:5060
      From: "Cisco7960A" <sip:7000@192.168.1.71>;tag=6fd02007d8e79901dab-72896c26
      To: <sip:1000@192.168.1.71>;tag=c49430004873319b79da1-307c4b53
      Call-ID: 0002fd06-e7d8013b-3c49167f-4e1cff7f@192.168.1.85
      Server: Cisco-SIP-IP-Phone/2
      Date: Mon, 12 Sep 2005 21:32:02 GMT
      CSeq: 101 INVITE
```

Fig. 3. Call establishment over Structured P2P Control Plane

Next, we performed experiments to quantize the performance of the structured P2P control plane. In these experiments, the maximum capacity of SIP peers is set to 5, unless stated otherwise. Since the P2P control plane provides a location service for the signaling plane we are interested with the following three major categories of measurements:

1. **Lookup Cost:** Quantized using the number of hops (both average and maximal) needed by a SIP peer to look for a particular resource and to successfully find it using **h-Chord**.

2. **Stabilization Cost:** Quantized using the number of messages exchanged by a new SIP peer to reach stability when SIP peers join / leave. Stability is defined as the state in which all Super-Peers have the correct successors, thus guaranteeing correct lookup using **h-Chord**.

3. **Load Balancing Capability:** Quantized using the number of SIP peers connected to a Super-Peer, and how this number changes when peers join / leave. Even though the structured P2P control plane was not initially designed to balance the load between the Super-Peers, we attempt to characterize its load balancing capabilities.

A. Structured P2P Control Plane Lookup Cost

To obtain the lookup cost, we performed the following experiments. For different network sizes (100 to 600 peers), a random SIP peer is chosen, and this SIP peer sequentially

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2006 proceedings.
Authorized licensed use limited to: The University of Toronto. Downloaded on December 9, 2009 at 02:04 from IEEE Xplore. Restrictions apply.

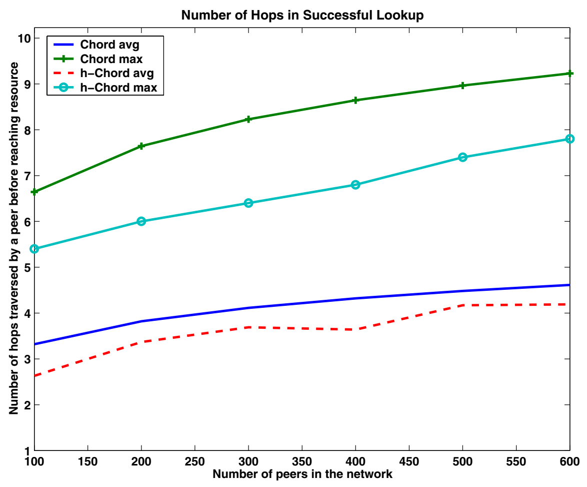


Fig. 4. Avg. and Max. number of hops needed for a successful lookup

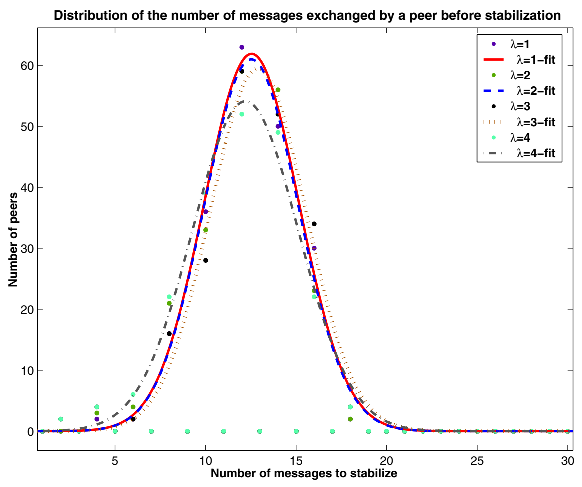


Fig. 5. Distribution of the number of messages exchanged by a peer before stabilization for different arrival rates

searches for all other SIP peers in the network, and records the average and maximum lookup cost in terms of the number of hops traversed before the lookup succeeds. To avoid biased results, we repeat the experiment for 5 different randomly chosen SIP peers, and average the results.

Fig. 4 shows the number of hops needed to perform a successful lookup using **h-Chord**. The number of hops needed for a successful lookup depends on the number of SIP peers in the network. In these measurements, we compare **h-Chord** to regular Chord, in order to quantify the effects of introducing the notion of Super-Peers into the structured P2P control plane. In Chord, the maximum lookup cost is $log(n)$ and the average lookup cost is $\frac{log(n)}{2}$, where n is the number of SIP peers in the Chord ring [4]. On the other hand, **h-Chord** shows improvements over Chord for all network sizes. For 400 peers, **h-Chord** has an average and maximum lookup costs of 3.64 and 7.4 respectively.

The explanation for the improvement shown by **h-Chord** is that the number of SIP peers in the ring (Super-Peers) is lowered from that in the Chord ring by a factor proportional to the average load of a Super-Peer, where load refers to the number of Ordinary-Peers connected to a given Super-Peer. For example, for 100 SIP peers, **h-Chord** resulted in 68 Ordinary-Peers, and 32 Super-Peers, hence an average load of 2.125 per Super-Peer. In a lookup, the maximum number of hops is proportional to $log(n)$, and the average number of hops is proportional to $\frac{log(n)}{2}$, with n being the number of SIP peers in the ring. Thus, the expected improvements from **h-Chord** are around $log(2.125)$, which is 1.0875, and $\frac{log(2.125)}{2}$, which is 0.5437, respectively for maximum and average lookup costs. Comparing Chord and **h-Chord** results obtained in the experiment for 100 SIP peers, we have improvements of 1.2439 and 0.6919 hops respectively for the maximum and average number of hops needed for correct lookup.

B. Structured P2P Control Plane Stabilization Cost

To obtain the stabilization cost, we performed the following experiments. For a given network size (500 SIP peers), 200 SIP peers join with different arrival rates, following an exponential distribution, with an average rate of λ SIP peers per second. For each new SIP peer, we count the number of messages exchanged before stability, as defined defined, is reached.

Fig. 5 shows the distribution of the number of messages exchanged by a SIP peer entering the network, before stability is reached, for different arrival rates λ, increased from 1 to 4 SIP peers per second. As the SIP peer arrival rates increase, a bound on the number of messages needed by the new SIP peers to reach stabilization is observed. This indicates that the structured P2P control plane system tends to stabilize with a finite number of messages exchanged. Furthermore, the distribution of the number of messages exchanged by a new SIP peer entering the network is Gaussian, with means and standard deviations shown in Table II.

One possible explanation can be found using the central limit theorem [14]. In fact, this number is a random variable that depends on the hash function used by **h-Chord** which determines the location in the ring at which the arriving SIP peer should be placed. This number is independent from a SIP peer to another, and follows an arbitrary distribution for each SIP peer, hence for several SIP peers arriving successively as in our experiment, this number has a limiting cumulative distribution function which approaches a Gaussian distribution. Furthermore, by observing Table II, we note the following: The mean and standard deviations are almost the same for different SIP peers arrival rates, hence the structured P2P control plane seems to be stabilizing. The coefficient of variation (ratio of standard deviation to mean) [14], which measures the dispersion of a probability distribution is close to 0.2. This shows that the dispersion of the number of messages needed by new SIP peers to reach stability do not drastically vary regardless of the arrival rate.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2006 proceedings.
Authorized licensed use limited to: The University of Toronto. Downloaded on December 9, 2009 at 02:04 from IEEE Xplore. Restrictions apply.

TABLE II
STABILIZATION COST

Arrival Rate (Peers per second)	Average	Standard Deviation	Coefficient of Variation
1	12.54	2.6099	0.2081
2	12.52	2.6000	0.2077
3	12.9	2.6474	0.2052
4	12.2	2.9338	0.2405

TABLE III
LOAD BALANCING CAPABILITIES

Network Size (Peers)	Average	Standard Deviation	Coefficient of Variation
100	2.1875	1.2113	0.5537
200	2.3333	1.2576	0.539
300	2.6429	1.3206	0.4997
400	2.5455	1.6699	0.5077

C. Structured P2P Control Plane Load Balancing Capabilities

To explore the load balancing capabilities of the structured P2P control plane, we performed the following experiments. For different network sizes, we measured the load on each Super-Peer in the ring. We found the distribution of this load, and calculated its first-order statistics.

Table III shows the results for network sizes of 100, 200, 300, and 400 SIP peers. The maximal capacity per Super-Peer is set to an arbitrary number (5 in this experiment) to obtain values for average and standard deviation. Although the distribution itself does not exhibit any particular properties, the interesting observations are in the results obtained for the coefficient of variation (which is independent of the maximal capacity value). While the values are very similar for different network sizes, they are close to 0.5. This means that the dispersion of the load on the Super-Peers is large, and that the load is not equally distributed among Super-Peers. While some are lightly loaded, others have a higher load. This observation is not surprising, given that **h-Chord**'s objective was not to equally distribute the load among Super-Peers, but rather to have a hierarchical structured P2P control plane topology with bounds on lookup and stabilization costs. In addition, the location of a SIP peer in the ring is random, depending on the hash function obtained. This aspect of **h-Chord** will be revisited in the future in order to propose solutions that could provide a better load balance in the control plane.

VI. CONCLUSION

In this paper, we introduced a hierarchical structured P2P overlay topology approach as the underlying substrate for the control plane in IP networks. This design attempts to combine the advantages of structured overlay topologies with the notion of Super-Peers introduced in unstructured overlay topologies, to deliver SIP functionalities over a P2P substrate. The functionalities offered by SIP servers are "virtualized" over the P2P substrate built using **h-Chord**. Experimental results show that **h-Chord** improves on the number of hops needed for successful lookup, and stabilizes quickly when SIP peers join the network at a high rate. However, the load balancing capabilities of **h-Chord** need further studying. This is part of our future work, as well as examining NAT / Firewall traversal issues when SIP endpoints are in private networks. We will also study the effect of the underlying topology on the the structured P2P control plane, and whether a cross-layer optimization of performance is possible by using feedback information from the underlying network.

REFERENCES

[1] A. Leon-Garcia and I. Widjaja, *Communication Networks*. Mc Graw Hill, 2004.
[2] J. Rosenberg et. al., "Session Initiation Protocol." RFC 3261, June 2002.
[3] D. Liben-Nowell, H.Balakrishnan, and D. R.Karger, "Analysis of the evolution of peer-to-peer systems," *Proceedings of 21st ACM Symp. Principles of Distributed Computing (PODC)*, pp. 233–242, July 2002.
[4] I.Stoica et. al., "A scalable peer-to-peer lookup service for internet applications." MIT, Tech. Rep. TR-819, 2001.
[5] L. G.-E. et. al., "Hierarchical p2p systems," *Proceedings of ACM/IFIP International Conference on Parallel and Distributed Computing (Euro-Par)*, pp. 1230–1239, Aug. 2003.
[6] I.Clarke, O.Sandberg, B.Wiley, and T. W.Hong, "Freenet: A distributed anonymous information storage and retrieval system," *Proceedings ICSI Workshop Design Issues in Anonymity and Unobservability*, June 2000.
[7] A.Rowstron and P.Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," *Proceedings of 18th IFIP/ACM International Conference on Distributed Systems Platforms*, pp. 329–350, Nov. 2001.
[8] B.Zhao and J.Kubiatowicz and A.Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing." Berkeley: Comput. Sci. Div., Univ. California, Tech. Rep. UCB/CSD-01-1141, 2001.
[9] A. Montresor, "A robust protocol for building superpeer overlay topologies," in *Proceedings of the 4th IEEE International Conference on Peer-to-Peer Computing*, pp. 202–209, August 2004.
[10] K. Singh and H. Schulzrinne, "Peer-to-peer internet telephony using sip." New York Metro Area Networking Workshop, Sept. 2004.
[11] "P2p sip." http://www.p2psip.org/.
[12] "Secure Hash Standard." Springfield, VA: U.S. Dept. Commerce/NIST, National Technical Information Service, FIPS 180-1, Apr. 1995.
[13] "Ethereal network packet analyzer." www.ethereal.com.
[14] A. Leon-Garcia, *Probability and Random Processes for Electrical Engineering (2nd Edition)*. Prentice Hall, 1993.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2006 proceedings.
Authorized licensed use limited to: The University of Toronto. Downloaded on December 9, 2009 at 02:04 from IEEE Xplore. Restrictions apply.