Publication I

Jouni Mäenpää and Gonzalo Camarillo. Study on Maintenance Operations in a Chord-based Peer-to-Peer Session Initiation Protocol Overlay Network. In *2009 IEEE International Symposium on Parallel and Distributed Processing (IPDPS '09), Seventh International Workshop on Hot Topics in Peer-to-Peer Systems (Hot-P2P)*, Rome, Italy, pp. 1-9, May 2009.

Study on Maintenance Operations in a Chord-based Peer-to-Peer Session Initiation Protocol Overlay Network

Jouni Mäenpää and Gonzalo Camarillo
Ericsson
{jouni.maenpaa, gonzalo.camarillo}@ericsson.com

Abstract

Peer-to-Peer Session Initiation Protocol (P2PSIP) is a new technology being standardized in the Internet Engineering Task Force. A P2PSIP network consists of a collection of nodes organized in a peer-to-peer fashion for the purpose of enabling real-time communication using the Session Initiation Protocol (SIP). In this paper, we present experimental results obtained by running a P2PSIP prototype in PlanetLab. Our prototype uses the Chord Distributed Hash Table (DHT) to organize the P2PSIP overlay and Peer-to-Peer Protocol (P2PP) as the protocol spoken between the peers. In the experiments, the performance of the system is studied under different churn rates and using different DHT maintenance intervals.

1. Introduction

Traditional architectures using the Session Initiation Protocol (SIP) [1] have a relatively fixed hierarchy of SIP proxies and user agents. These client-server systems typically employ a proxy-registrar server for every domain. In contrast, in a Peer-to-Peer Session Initiation Protocol (P2PSIP) system, SIP is used in a peer-to-peer environment where traditional proxy-registrar and message routing functions are replaced by a peer-to-peer overlay network [2]. Whereas in traditional client-server SIP, Address of Record (AoR) to contact Uniform Resource Identifier (URI) mappings are stored by a SIP proxy-registrar, in P2PSIP these mappings are distributed amongst the peers in the overlay.

In this paper, the performance of a P2PSIP system is studied through experiments in the PlanetLab network. The goal is to gain an understanding of the characteristics and requirements of P2PSIP. Gaining such an understanding is important, since the characteristics of P2PSIP, and person-to-person communication in general, are different from traditional P2P applications such as file sharing. The paper is structured as follows. In Section 2, an introduction is given to our P2PSIP prototype and to the mechanisms it uses. Section 3 summarizes the results of previous studies. Section 4 describes the experiment setup. In Section 5, the results of the measurements are presented. Finally, Section 6 concludes the paper.

2. P2PSIP prototype

2.1. Peer-to-peer protocol

Peers in a P2PSIP overlay need to use a protocol between them to maintain the overlay and to store and retrieve data [3]. This protocol, which is called the P2PSIP peer protocol, is currently being designed by the P2PSIP working group of the Internet Engineering Task Force (IETF). The peer protocol used in the experiments carried out for this paper is called the Peer-to-Peer Protocol (P2PP) [4]. P2PP is one of the peer protocol proposals submitted to the P2PSIP working group. It is an application-layer binary protocol for creating and maintaining an overlay network.

The peers in a P2PSIP overlay collectively run a distributed database algorithm, which allows data to be stored and retrieved in an efficient manner. This algorithm is used to implement a distributed location function providing the mapping between SIP AoRs and contact URIs. Distributed Hash Tables (DHTs) are one way to implement the distributed database. Both the Chord [5] and Bamboo [6] DHTs have been suggested as good choices for the distributed database algorithm to be used by P2PSIP [3]. The Resource Location and Discovery (RELOAD) [7] P2PSIP peer protocol proposal specifies the Chord DHT algorithm as mandatory to implement. For these reasons, we chose Chord as the DHT algorithm for our P2PSIP prototype.

2.2. Chord

Chord [5] is a structured P2P protocol that uses consistent hashing to build a DHT out of several independent nodes. Consistent hashing assigns each peer and data key an *m*-bit identifier using SHA-1 as the base hash function. These keys are ordered on an identifier circle of size 2^m. On the identifier circle, key *k* is assigned to the first peer whose identifier equals or follows the identifier of *k* in the identifier space. The identifier circle is called the Chord ring.

On the Chord ring, each peer *n* maintains a routing table with up to *m* entries, called the finger table. The *i*th entry in the table contains the identity of the first peer *s* that succeeds *n* by at least 2^{i-1} on the identifier circle. This peer is called the *i*th finger. In an *N*-node network, each node maintains information about only $O(\log N)$ other nodes in its finger table.

Chord needs the successor pointer of each node to be up to date in order to ensure that lookups produce correct results as the set of participating nodes changes. This is achieved by running a stabilization protocol in the background periodically. The stabilization protocol uses three operations: successor stabilization, finger stabilization and predecessor aliveness check. In the successor stabilization procedure, a node *n* asks its successor for the successor's predecessor *p*, and decides whether *p* should be *n*'s successor instead. The purpose of the finger stabilization procedure is to incorporate new nodes into the finger table. Finally, in the predecessor aliveness check, a node simply contacts its predecessor to learn whether the predecessor is still alive.

To increase robustness in the event of node failures, each Chord node maintains a successor list of size *r*, containing the node's first *r* successors.

3. Related work

Previous studies of churn in DHTs include [6] and [17]. In [6], the performance of the Bamboo DHT is studied in an emulated 1000-node network. Three factors that affect the behavior of DHTs under churn were identified: reactive versus periodic recovery from failures, calculation of message timeouts, and choice of nearby over distant neighbors.

In [17], the impact of churn on four DHTs, Tapestry, Chord, Kelips and Kademlia is studied through simulations in a 1024-node overlay. The main finding is that with the right parameter settings, all four DHTs have similar overall performance.

Unlike [6] and [17], this paper focuses on P2PSIP. Our traffic model is specific to P2PSIP. Also, our results have been obtained by running experiments in PlanetLab instead of using a simulated or emulated network. In contrast to [6], which studies the performance of the Bamboo DHT, we used the Chord DHT. This paper also investigates three different churn rates, unlike [17], which uses only one churn rate. Finally, in addition to studying churn, we also experiment with different maintenance intervals.

4. Experiments

The results presented in this paper were obtained from a real P2PSIP overlay created by nodes running the P2PSIP prototype. The P2PSIP prototype was implemented in the Java programming language. The prototype uses P2PP as the peer protocol and Chord with recursive routing as the DHT. Peer protocol connections run over TCP. The reason TCP was chosen is that both the P2PP [4] and RELOAD [7] P2PSIP peer protocols prefer the use of TCP over UDP. The prototype uses SIP as the call control protocol. SIP uses the P2PSIP overlay as a lookup mechanism to map AoRs to contact URIs.

The experiments were carried out in the PlanetLab network [8]. To run each experiment, the P2PSIP prototype was uploaded to a set of PlanetLab nodes, which then created a global P2PSIP overlay consisting of 500 peers. For instance a small enterprise could have a 500-peer global internal P2PSIP telephony network. Ideally, we would have used larger overlays, but the number of simultaneously online PlanetLab nodes was a limiting factor. We believe that a 500-node overlay already demonstrates problems that will also affect larger overlays. Each PlanetLab node ran three peers. Peers join the overlay by contacting a bootstrap peer, a PlanetLab node located in France. During their uptime, peers collect measurement data and report the results to a server before leaving the overlay.

4.1. Traffic model

The arrival and departure of users (i.e., churn) is modeled as a Poisson process with three different mean arrival rates: 1/5, 1/10, and 1/30 peers/s. We chose 1/5 peers/s as the highest churn rate because at even higher churn rates, the stability of the network started to suffer. On the other hand, the choice of the lowest churn rate was motivated by the observation that in the Skype network, the median session time is several hours [9], [10]. Graceful node departures are used; before leaving the overlay, a peer sends a Leave message to its first successor and predecessor.

Table 1. Traffic model

Parameter	Value
Arrival rate of users	high 1/5s, medium 1/10s, low1/30s
Average network size	500 peers
Measurement duration	3600s
Busy hour call attempts	2.21 calls per user
% of calls to non-buddies	33.3%
Size of buddy list	22

In each experiment, a 500-node P2PSIP network was created from scratch. The actual data collection phase begins when the network size reaches 500 users and lasts for an hour. This one hour period is modeled as a busy hour, and during it the average network size stays at 500 peers.

The SIP traffic exchanged between the peers consists of instant messaging, presence and Voice over IP (VoIP) calls. Each user is assumed to make 13 VoIP calls per day, as suggested in [11]. Out of these 13 calls, 17% are used to represent the busy hour traffic [12], meaning that the number of busy hour call attempts per user is 2.21. The arrival of calls is modeled as a Poisson process with a mean rate of 2.21 calls per hour. It is assumed that 1/3 of calls are made to users not on the buddy list; users typically call their friends instead of strangers [13]. Only calls targeted to strangers require a P2PSIP lookup operation; in the case of buddies, the contact URI is learned already when making the initial lookup for the buddy to find her presence status. The size of the buddy list of each user is 22, based on the results obtained in [14]. After having joined the P2PSIP overlay, each peer initiates 22 lookup operations to find the contact addresses of the user's buddies. Periodic queries are not sent for offline buddies, but it is assumed that the overlay notifies the user when the buddy becomes online. The traffic model is summarized in Table 1.

4.2. Chord parameters

In Chord, each peer maintains on the order of $\log N$ fingers and successors [5]. In the experiments, nine fingers and ten successor pointers were maintained.

According to [15], a Chord network in a ring-like state remains in a ring-like state as long as nodes send $\Omega(\log^2 N)$ messages before N new nodes join or $N/2$ nodes fail. Thus, in a 500-node network, at a churn rate of 1/5s, roughly $\Omega(\log^2 N)$ rounds of stabilization should occur in 1250 seconds (i.e., before 250 nodes leave). However, as we will see, making the maintenance operations too frequent has a negative impact on the performance of the overlay. In the experiments, we studied the performance of the P2PSIP overlay network at three different churn rates with maintenance intervals ranging from 15 to 360s.

Table 2. Chord parameters

Parameter	Value
Finger pointers	9
Successor pointers	10
Maintenance intervals	15s, 45s, 90s, 135s, 180s, 270s, 360s
Self-lookup interval	60s

Every measurement for a specific maintenance interval at a specific churn rate was repeated 15 times.

When the maintenance timer fires at peer *n*, peer *n* performs the successor stabilization, finger stabilization, and predecessor aliveness check operations. The successor stabilization operation of Chord is implemented by using the P2PP ExchangeTable request and response. The finger stabilization operation is implemented by using P2PP LookupPeer request. When the maintenance timer fires, peer *n* chooses the next finger interval *i* that should be fixed and sends a LookupPeer request for the identifier $n+2^{i-1}$. The peer that answers the request will be made the *i*th finger of peer *n*. Finally, the predecessor aliveness check operation is implemented by using P2PP KeepAlive request.

In addition to the stabilization operations described above, each peer performs a self-lookup operation [15] every 60 seconds. In the self-lookup, a node searches for itself in the network by sending a lookup to its first successor. The purpose of the self-lookup operation is to detect loops; in a loopy network a self-lookup initiated by node *n* may never reach node *n*, but is answered by some other node. The Chord parameters we used are summarized in Table 2.

5. Results

This section presents the results obtained in the measurements. The error bars shown in the figures of this section represent 95% confidence intervals.

5.1. Bandwidth consumption

P2PP traffic consists of maintenance traffic and lookup traffic. By maintenance traffic we refer to traffic used to maintain the overlay, that is, traffic generated by joins, departures, and stabilization operations. In contrast, lookup traffic consists of the P2P lookup operations done by users.

The average number of incoming bytes per second for the peers participating in the overlay is shown in Fig. 1. Due to the symmetric nature of P2PP traffic, the average bandwidth consumption of outgoing traffic is nearly identical to Fig. 1 and is thus not shown separately. The 95% confidence intervals are so small that they are not visible in the figure. Only the bandwidth consumption of P2PP messages is included in the figure; the overhead of lower layers in the

protocol stack has been omitted. The average bandwidth consumption is similar for all churn rates; since Chord uses periodic recovery from failures [6], the frequency of maintenance messages does not depend on the churn rate. However, although the amount of maintenance traffic is identical for all churn rates, at the higher churn rates there is more lookup traffic, as can be observed from Fig. 2, which depicts the percentage of maintenance traffic out of total traffic. For instance, at the highest churn rate, 720 peers join the overlay during the one hour measurement period. Each joining user initiates among other things 22 lookups for her buddies. These lookups are processed by 1220 peers that are active in the overlay during the measurement period. At the lowest churn rate, only 120 peers join during the one hour period, and the buddy lookups they initiate are processed by 620 peers. Thus, at the highest churn rate, an average peer receives more incoming lookup traffic than at the lowest churn rate. This explains why there is a small statistically significant increase in the bandwidth consumption for each maintenance interval when the churn rate is increased.

From Fig. 1, we can also observe that for all churn rates, there is a statistically significant reduction in the amount of traffic when the maintenance interval is increased from 15s to 45s or to 90s. As the maintenance interval is increased further, the amount of traffic each peer generates per second starts to level off. This is because e.g. increasing the interval from

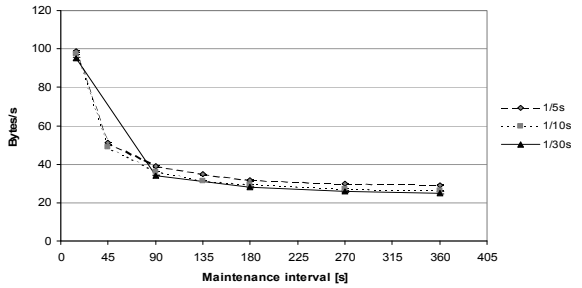

Figure 1. Incoming bytes per second

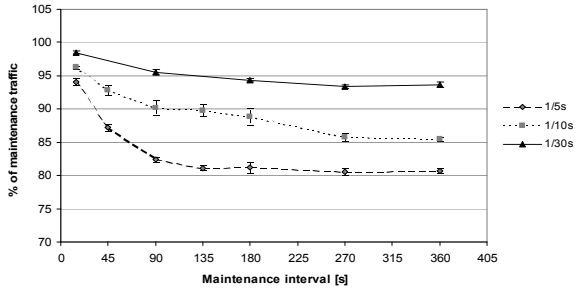

Figure 2. Percentage of maintenance traffic

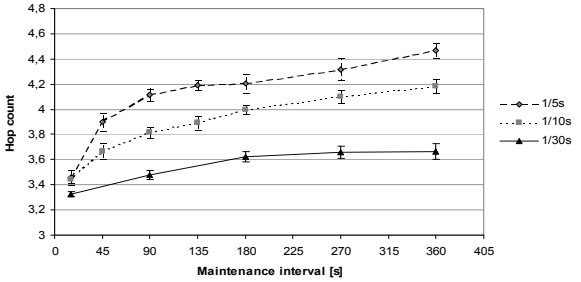

Figure 3. Average hop count

270s to 360s reduces the number of stabilization rounds occurring during the one hour period only by three and thus does not reduce the total amount of traffic significantly, whereas increasing the interval from 15s to 45s causes the number of stabilization rounds to drop by 160 from 240 to 80.

From Fig. 2, one can observe that even at the highest churn rate, 80-95% of the traffic is maintenance traffic, depending on the frequency of maintenance operations. Thus, maintenance traffic clearly dominates over lookup traffic. This is especially clear at the lowest churn rate, for which 93-98% of the total traffic is maintenance traffic.

5.2. Hop count

Fig. 3 shows the average hop count for each churn rate. We can observe that churn rate has a clear impact on hop count: the average number of hops is always significantly (statistically at the 95% confidence level) larger for the same maintenance interval at higher churn rates than at the lower churn rates. For instance, for the maintenance interval of 90s, the average hop count is 0.5 hops larger at the highest churn rate (1/5s) than it is at the lowest churn rate (1/30s). The reason for the hop count being larger at higher churn rates is that a high churn rate causes the finger table of each node to be less optimal. Each finger that leaves the network introduces a hole to the finger table of at least one other node in the network. Since the finger stabilization process is run periodically, it takes time before the empty finger table entry can be filled with a new finger. The higher the churn rate, the more frequently holes are introduced to the finger tables. And the more holes the finger table of a peer has, the less optimal are the routing decisions the peer makes, since the peer is forced to forward the request being routed to a finger that may be a rather distant predecessor of the target identifier. This has the effect of increasing the average hop count.

By observing the graph of each churn rate, we can also see that increasing the maintenance interval has the effect of increasing the hop count. As an example,

for the high and medium churn rates, lengthening the maintenance interval from 15s to 45s introduces a statistically significant increase in the hop count. Hop count grows as the maintenance operations become less frequent since at longer maintenance intervals, it takes longer before old fingers are replaced with more optimal fingers, and before holes in the finger table are filled. The highest hop count values are observed when a long maintenance interval is used in a high-churn scenario. In that case, holes are introduced to finger tables frequently, and it takes a long time before the empty positions are filled with new fingers.

5.3. Lookup Delay

Fig. 4 shows the average lookup delay for all churn rates. Delays of self-lookup requests are not included in the figure; since self-lookups requests make a 360 degree turn around the Chord ring, their delays are considerably longer than for other lookups. In Fig. 4, the x-axis has been shifted to right by 5s for the medium (1/10s) churn rate, since the confidence intervals of that churn rate overlap with those of the other churn rates. The difference in delay between the medium and high churn rates and the medium and low churn rates is in many cases statistically insignificant. However, we can still see from Fig. 4 that regardless of the frequency of maintenance operations, there is always a statistically significant difference between the average delay at the high and low churn rates. The reason why the delay is greater for the higher churn rates is explained first of all by the fact that the higher churn rates have higher hop counts. Second of all, at the higher churn rates also the average load (queue size) at each peer is higher, meaning that the queuing delay is higher. As an example, for the maintenance interval of 15s, the average forwarding delay at each hop is roughly 40ms more for the highest churn rate than for the lowest churn rate. Thirdly, path failures are more common at higher churn rates. The detection of a path failure requires a timeout at an intermediate node, which has the effect of increasing the average delay.

From Fig. 4, we can also observe that for each individual churn rate, the average lookup delay grows as the maintenance operations become less frequent. As an example, for the 1/10s churn rate, the average lookup delay is significantly higher for the 180s maintenance interval than for the 90s interval. The delay grows first of all because the average hop count grows. Second of all, also the amount of requests experiencing path failures grows, which has the effect of increasing the delay.

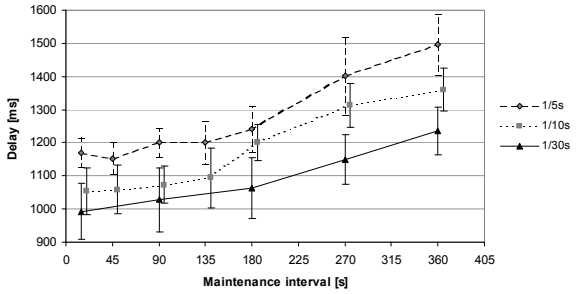

Figure 4. Average lookup delay

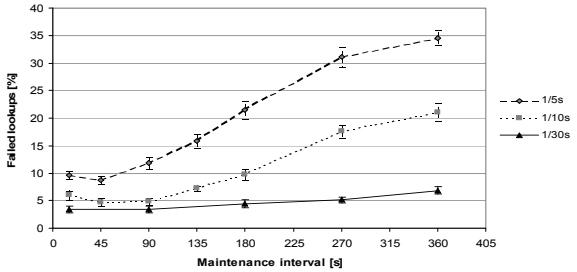

Figure 5. All failed lookups

5.4. Failed lookups

Fig. 5 shows the percentage of failed lookups for the three churn rates. By lookups we mean P2PP LookupObject, Self-Lookup and LookupPeer requests. The differences between different churn rates are statistically significant. At the highest churn rate, the best performance is achieved when the maintenance interval is 15s or 45s (the difference in lookup failure rates for these two maintenance intervals is statistically insignificant), in which case roughly nine percent of all lookups fail. At the 1/10s churn rate, the best performance is achieved when the maintenance interval is 15-90s. In this case, roughly five percent of lookups fail. The lowest churn rate also has the lowest lookup failure ratio. The best performance is achieved when the maintenance interval is 15-180s, in which case at the minimum, 3.5 percent of all lookups fail. A lookup is considered a failure if it either times out, exceeds the hop count limit or encounters a path error.

From Fig. 5, we can observe for the two highest churn rates, that as the maintenance interval is increased beyond 90s, the number of failed lookups starts to grow steeply. The differences between maintenance intervals are statistically significant. At the lowest churn rate, the performance does not degrade as dramatically, since the number of peers joining and leaving the network between any two consecutive rounds of maintenance operations is considerably smaller than for the higher churn rates.

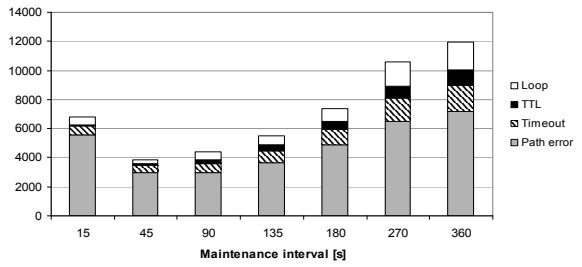

Figure 6. Self-lookup failures

This results in a smaller amount of path errors, loops, timeouts and Time-to-Live (TTL) exceeded errors.

The best level of performance at the highest churn rate is achieved when the maintenance interval is 15-45s; but even in this case roughly one out of ten lookups fails. This can hardly be considered satisfactory. To help understanding the reasons behind failed lookups, Fig. 6 depicts the causes of failures for self-lookup requests at the highest churn rate. The figure shows only self-lookup requests because the rate of self-lookup operations does not depend on the maintenance interval, which makes comparison between different maintenance intervals easy. From the figure, we can observe that for all maintenance intervals, the most common failure type is a path error. A path error occurs whenever the peer to which a message is being routed cannot be contacted, for instance because the peer has already left the network. When the maintenance interval is 45s, roughly 75% of self-lookup failures are caused by path errors. Nearly all path errors occur because the next hop node a peer *n* selects from its finger or neighbor table for a request has left the network, meaning that the peer cannot establish a connection to the next hop node. Nearly all path errors are forward path errors; path errors that occur because of a reverse path failure, that is, because a response cannot be returned to the previous hop intermediary since that node has left the network, are very rare. For instance, at the highest churn rate, less than 1.5 percent of all path failures are reverse path failures regardless of the frequency of maintenance operations. This result supports the use of symmetric recursive routing in P2PSIP overlays [16].

Forward path errors occur whenever a finger to which a request is being routed has already left the overlay. Since Chord uses periodic recovery, a finger that leaves the overlay does not inform a remote peer *n* having it as a finger of its departure. Instead, peer *n* is responsible for detecting the departure of the finger by itself. Whenever peer *n* detects that a finger has left while trying to unsuccessfully forward a request to it, a forward path error occurs. However, since the P2PSIP prototype uses TCP as the transport protocol, the departure of fingers that leave the overlay can be detected by peer *n* from the fact that the TCP connection is closed, provided of course that peer *n* has established a TCP connection with the finger. However, if the finger leaves the overlay before peer *n* has established a TCP connection with it, peer *n* will not learn that the finger has left until a path error occurs. We used a TCP socket connect timeout of 30s.

5.5. Timeouts

Fig. 7 plots the percentage of requests that experience a timeout. Also all other request types in addition to lookup requests are included. From the figure, we can observe that timeouts are more common at the higher churn rates. The differences between the churn rates are statistically significant except in the case of the 90s maintenance interval for which the difference between the low and medium churn rates is statistically insignificant. We can also observe that for each churn rate, a reduction in the rate of maintenance messages results in a higher amount of timeouts. As an example, for all churn rates, the percentage of timed out requests is significantly higher for the 180s interval than for the 15s interval. The majority of timeouts are caused by path errors. At the time when an intermediate node along the path of a message detects that the next hop node is unreachable, it will send an error response back to the initiator of the request. However, if it took a long time for the intermediate node to realize that the next hop node is gone, then the request may already have timed out at the initiator and also at intermediate nodes along the path.

There is no statistically significant difference between the percentages of failed requests for the medium and low churn rates at the 90s maintenance interval. As we will see later, in the case of the medium churn rate, the 90s interval also has a lower amount of detected loops and path failures than the other intervals. The low number of timeouts for this interval at the medium churn rate is thus explained by the small amount of path errors (as we discussed above, path errors cause the majority of timeouts).

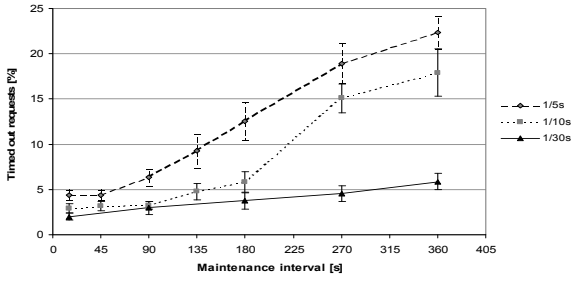

Figure 7. Percentage of timed out requests

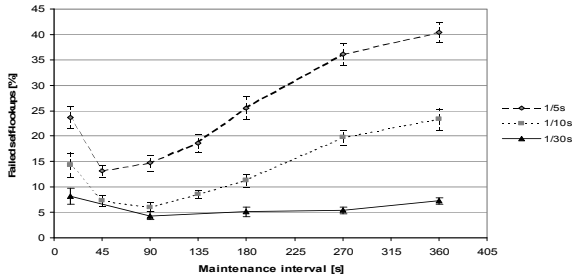

Figure 8. Percentage of failed self-lookups

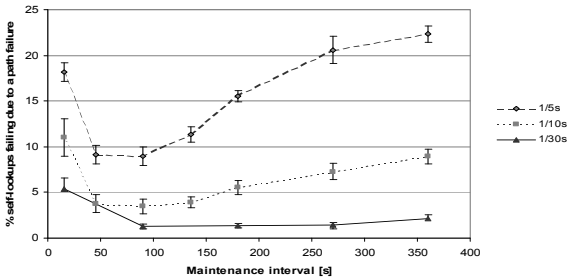

Figure 9. Self-lookup path errors

5.6. Failed self-lookups

Fig. 8 shows the percentage of failed self-lookups for each churn rate. As can be expected, we can observe that at the higher churn rates, significantly more self-lookups fail. However, when considering the frequency of maintenance operations, we can observe that it does not seem to pay off to make the maintenance interval very small; for each churn rate, a statistically significant reduction in the percentage of failed self-lookups can be achieved by using an interval longer than 15s. The most common reason why a self-lookup request fails is a path error. The high amount of path errors at short maintenance intervals is explained by the fact that when the interval is short, a peer switches its fingers and successors too frequently. If the interval is 15s, each of the nine fingers may change every 135s. When switching to a new finger, the connection to the old finger is closed and the new finger is inserted into the finger table. The disadvantage of changing fingers frequently is that a peer may choose as the new finger a peer that will leave the network shortly. If the new finger leaves the network before the peer has established a TCP connection with it (to deliver the first message), a path error will occur. The benefit of using slightly longer maintenance intervals is that since fingers change less frequently, there are less chances of switching to a finger that will leave the network shortly.

Another reason for the high number of self-lookup path failures at the 15s maintenance interval is that due to the large amount of traffic, some of the peers become overloaded, as will be discussed in Section 5.7. If the successor *s* of peer *n* is overloaded, the messages *n* sends to *s* may time out, which causes *n* to remove the successor. The removed successor is replaced by the next successor on the successor list. However, if no TCP connection has yet been established with the new successor, a path error will occur if the new successor leaves the overlay before the first messages gets routed to it.

The percentage self-lookup requests experiencing a path failure is depicted in Fig. 9. Although initially, the percentage of path failures decreases significantly as the maintenance interval is lengthened, it starts to grow again after a certain point. This is especially visible for the two highest churn rates; for both of them the increase in the percentage of path failures is statistically significant when the maintenance interval is increased from 135 to 270s. In our Chord implementation, a peer does not establish a connection to a new finger or to a new neighbor until it has the first message that should be forwarded to that peer. This works well at small maintenance intervals since due to the high amount of traffic, the gap between adding a peer to the routing table and sending the first message to it is very small. However, at longer maintenance intervals, there is less traffic and it will take longer before the first message is routed to a new finger. Thus, it will take longer before the connection is established. Consequently, the probability that the new finger or neighbor has left the network before the moment when it is contacted for the first time grows as the maintenance interval becomes longer. The effect is more pronounced at the higher churn rates.

On the other hand, e.g. in the case of the longest maintenance interval, the successor stabilization procedure is only run every six minutes. If the churn rate is high, the chances are that some of the successors leave the network between two consecutive successor stabilization rounds. This will result in more path errors, since messages will get routed to successors that are no longer present in the overlay.

To avoid the case that the departure of a peer is not detected until the moment when routing the first message to it, peers could as an optimization establish TCP connections to new neighbors and fingers immediately after they have learned about the existence of the new neighbors and fingers, even before they have any traffic to send to them. Of course, detecting node failures from closed connections is TCP-specific and does not work if a connectionless transport such as UDP is used. We plan to evaluate this approach as part of future work.

5.7. Detected loops

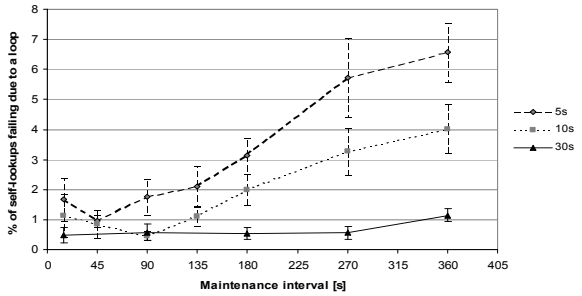

Figure 10. Percentage of looping self-lookups

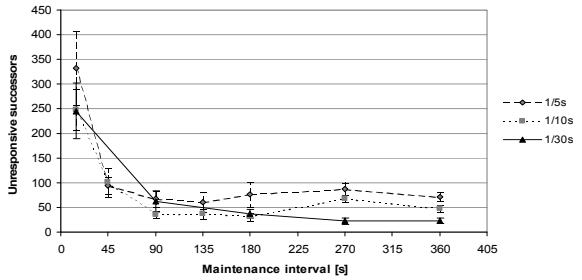

Figure 11. Unresponsive successors

The percentage of self-lookups failing due to a loop is plotted in Fig. 10. The amount of loops reveals how stable the Chord ring is. We consider a loop to exist in the network if a self-lookup request initiated by peer *n* is answered by some other peer than peer *n* itself. From the figure, we can observe that the network is clearly less stable at higher churn rates. As the maintenance interval is increased beyond 135s, the difference in the percentage of self-lookups encountering a loop for the three churn rates is statistically significant. We can also see that for the high and medium churn rates, the amount of instability increases rapidly as the maintenance interval grows. However, initially the percentage of self-lookups encountering a loop decreases for the highest churn rate; as the maintenance interval grows from 15s to 45s, there is a statistically significant drop in the percentage of self-lookups failing due to a loop. The same phenomenon can be observed for the medium churn rate as the interval is increased from 15s to 90s.

Fig. 11 shows the number of successors that fail to respond to ExchangeTable requests that are used to implement the successor stabilization procedure. A successor is removed if two consecutive ExchangeTable requests sent to it time out. From the figure, we can see that for each churn rate, there is a large number of unresponsive successors in the case of the shortest maintenance interval. In the case of the highest churn rate, the amount of unresponsive successors starts to level off beginning from the maintenance interval of 45s. For the medium churn rate, this happens starting from the interval of 90s, and for the lowest churn rate, from the interval of 270s onwards. For the medium and high churn rates, there is a statistically significant drop in the number of unresponsive successors when increasing the maintenance interval from 15s to 45s. E.g. in the case of the highest churn rate, the number of unresponsive successors drops by almost 73%. For the low churn rate, there is a statistically significant drop in the number of unresponsive successors when increasing the maintenance interval from 15s to 90s. Removal of unresponsive successors creates a lot of instability in the network and causes messages to be incorrectly routed. The high amount of unresponsive successors explains the reason why the amount of loops decreases initially in Fig. 10 when moving away from the shortest maintenance intervals. The unresponsiveness of the successors is explained by the fact that at the 15s maintenance interval, the queue size of the message dispatcher thread of the P2PSIP prototype becomes very large in some PlanetLab nodes. Such overloaded nodes create a lot of instability in the network.

6. Conclusion

In this paper, the performance of a P2PSIP VoIP and instant messaging system was studied through measurements carried out in PlanetLab.

The average lookup delay in a 500-node P2PSIP overlay was found to be between 1.0s and 1.5s, depending on the churn rate and maintenance interval. Based on these results we can conclude that the extra delay caused by replacing the centralized location service of SIP with a distributed mechanism is not excessive.

In a P2PSIP overlay where P2PP lookup traffic consists of lookups related to presence, instant messaging and VoIP calls, DHT maintenance traffic clearly dominates over lookup traffic.

As churn increases, hop count, lookup delay, and amount of timeouts, detected loops and failed lookups increase. The same happens when the rate of maintenance messages is decreased. However, what is perhaps more surprising is that making maintenance operations too frequent clearly has a negative impact on the performance of the overlay. This is because too short a maintenance interval results in some peers becoming overloaded and in too frequently changing finger and neighbor pointers.

The most common cause of failed requests is a path error. Path errors are nearly always forward path failures. Reverse path failures were found to be rather rare. This result supports the use of symmetric

recursive routing in P2PSIP overlays, since it shows that reverse path failures, which have been considered to be a major disadvantage for symmetric recursive routing, do not in fact constitute a problem.

The number of failed lookups is rather high for all of the churn rates. Since the most common cause for a failed lookup is a path error, an improvement can be achieved by retrying a failed lookup. However, if the lookup is related to e.g. a VoIP call, such retries are undesirable since they increase the call setup delay.

At the highest churn rate and highest rate of maintenance messages, each peer sends and receives 355 kilobytes of data during one hour. Thus, we can conclude that the bandwidth consumption of a P2PSIP system is rather low compared to traditional P2P applications such as file sharing.

7. References

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley and E. Schooler: "SIP: Session Initiation Protocol", RFC 3261, Jun. 2002.

[2] K. Singh and H. Schulzrinne: "Peer-to-peer internet telephony using SIP," in *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video*, Washington, USA, 2005, pp. 63-68.

[3] D. Bryan, P. Matthews, E. Shim, D. Willis and S. Dawkins: "Concepts and Terminology for Peer-to-Peer SIP", Internet-Draft, work in progress, Jul. 2008.

[4] S. Baset, H. Schulzrinne and M. Matuszewski: "Peer-to-Peer Protocol (P2PP)", Internet-Draft draft-baset-p2psip-p2pp-01, work in progress, Nov. 2007.

[5] I. Stoica, R. Morris, D. Liben-Nowell, D.R. Karger, M. F. Kaashoek, F. Dabek and H. Balakrishnan: "Chord: A scalable peer-to-peer lookup protocol for Internet applications," *IEEE/ACM Trans. Networking*, vol. 11, no. 1, pp. 17-32, Feb. 2003.

[6] S. Rhea, D. Geels, T. Roscoe and J. Kubiatowicz: "Handling churn in a DHT," in *Proc. annual conference on USENIX*, Boston, MA, 2004.

[7] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset and H. Schulzrinne: "Resource Location and Discovery (RELOAD) Base Protocol", Internet-Draft draft-ietf-p2psip-base-01, work in progress, Dec. 2008.

[8] L. Peterson, A. Bavier, M.E. Fiuczynski and S. Muir: "Experiences Building PlanetLab," in *Proc. 7th USENIX Symp. on Operating Systems Design and Implementation*, 2006.

[9] S. Guha, N. Daswani, and R. Jain: "An experimental stydy of the Skype peer-to-peer VoIP system," in *Proc. 5th International Workshop on Peer-to-Peer Systems (IPTPS)*, Santa Barbara, CA, Feb. 2006.

[10] D. Rossi, M. Mellia, and M. Meo: "A detailed measurement of Skype Network Traffic," in *Proc 7th International Workshop on Peer-to-Peer Systems (IPTPS'08)*, Tampa Bay, Florida, Feb. 2008.

[11] B. Athwal, F.C. Harmatzis and V.P. Tanguturi: "Replacing Centric Voice Services with Hosted VoIP Services: An Application of Real Options Approach," in *Proc. 16th International Telecommunications Society (ITS) European Regional Conference*, Porto, Portugal, Sept. 2006.

[12] "Traffic Analysis for Voice over IP", http://www.cisco.com/en/US/docs/ios/solutions_docs/voip_solutions/TA_ISD.html

[13] C. Cheng, S. Tsao and J. Chou: "Unstructured Peer-to-Peer Session Initiation Protocol for Mobile Environment," in *Proc. 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'07)*, 3-7, Sept 2007, pp. 1-5.

[14] B. A. Nardi, S. Wittaker and E. Bradner: "Interaction and Outeraction: Instant Messaging in Action," in *2000 Proc ACM conference on computer supported cooperative work*, Philadelphia, Pennsylvania, United States, 2000, pp. 79-88.

[15] D. Liben-Nowell, H. Balakrishnan and D. Karger: "Observations on the dynamic evolution of peer-to-peer networks," in *Proc. First International Workshop on Peer-to-Peer Systems (IPTPS'02)*, Cambridge, MA, Mar. 2002.

[16] D.A. Bryan, B.B. Lowekamp and M. Zangrilli: "The Design of a Versatile, Secure P2PSIP Communications Architecture for the Public Internet," in *IEEE International Symposium on Parallel and Distributed Processing (IPDPS 2008)*, April 2008, pp. 1-8

[17] J. Li, J. Stribling, T.M. Gil, R. Morris and M.F. Kaashoek: "Comparing the performance of distributed hash tables under churn," in *Proc. 3rd International Workshop on Peer-to-Peer Systems (IPTPS)*, San Diego, CA, Feb. 2004.