Publication IV

Jouni Mäenpää and Gonzalo Camarillo. Estimating Operating Conditions in a Peer-to-Peer Session Initiation Protocol Overlay Network. In *2010 IEEE International Symposium on Parallel and Distributed Processing (IPDPS '10), Eight International Workshop on Hot Topics in Peer-to-Peer Systems (Hot-P2P)*, Atlanta, USA, pp. 1-6, April 2010.

Estimating Operating Conditions in a Peer-to-Peer Session Initiation Protocol Overlay Network

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

***Abstract*—Distributed Hash Table (DHT) based peer-to-peer overlays are decentralized, scalable, and fault tolerant. However, due to their decentralized nature, it is very hard to know the state and prevailing operating conditions of a running overlay. If the system could figure out the operating conditions, it would be easier to monitor the system and re-configure it in response to changing conditions. Many DHT-based system such as the Peer-to-Peer Session Initiation Protocol (P2PSIP) would benefit from the ability to accurately estimate the prevailing operating conditions of the overlay. In this paper, we evaluate mechanisms that can be used to do this. We focus on network size, join rate, and leave rate. We start from existing mechanisms and show that their accuracy is not sufficient. Next, we show how the mechanisms can be improved to achieve a higher level of accuracy. The improvements we study include various mechanisms improving the accuracy of leave rate estimation, use of a secondary network size estimate, sharing of estimates between peers, and statistical mechanisms to process shared estimates.**

I. Introduction

Real-world peer-to-peer systems need to deal with constantly changing operating conditions. As an example, the size of the overlay and the rates at which peers join and leave it typically vary with time. This makes it very difficult to configure the system in such a way that it can maintain its stability throughout its lifetime. Adaptive behavior would solve this problem but unfortunately, it requires knowledge about the prevailing state of the system. Such information is very difficult to obtain in a fully decentralized overlay.

Current overlays are usually configured statically. This works in situations where perfect information about present and future operating conditions is available. In such situations, it is possible to choose fixed optimal values for parameters such as stabilization rate, neighborhood set size, and routing table size. However, if the operating conditions of the overlay do not remain static but evolve with time, it is not possible to achieve both a low lookup failure rate and a low communication overhead by using fixed parameters. To achieve optimal performance, it is necessary to adapt the parameters of the overlay. For this, information about the current network size and churn rate is needed.

In this paper, we first evaluate the accuracy of existing size and churn rate estimation mechanisms for Distributed Hash Table (DHT) based overlays. We will show that the accuracy of these mechanisms is not satisfactory. Next, we study how the mechanisms can be improved to achieve a higher level of accuracy. The optimizations we study include various mechanisms improving the accuracy of leave rate estimation (including extended failure history, downloading of failure history, downloading of initial leave rate estimate, aging of failures, maintaining information on network sizes associated with old leave rate estimates, selective dropping of old failures, and improved failure detection), use of a secondary network size estimate, sharing of estimates, and statistical mechanisms to process shared estimates. The evaluation is performed in a simulated Peer-to-Peer Session Initiation Protocol (P2PSIP) overlay network. P2PSIP is a new decentralized person-to-person communication system that is currently being standardized in the Internet Engineering Task Force (IETF). P2PSIP uses the Session Initiation Protocol (SIP) to enable real-time communication in a peer-to-peer environment. P2PSIP uses the Chord DHT to implement the underlying lookup mechanism. The remainder of the paper is structured as follows: Section II describes the P2PSIP simulator used in the experiments. Section III presents related work. Section IV describes the experiments and the traffic model used. Section V presents the results of the experiments. Finally, Section VI concludes the paper.

II. P2PSIP simulator

The simulations were carried out using an event-driven, message-level P2PSIP overlay simulator. The simulator is implemented in the Java programming language. It is based on a real-world P2PSIP implementation we have used in previous work [1], [2]. We chose to use a simulator instead of carrying out the measurements in PlanetLab, as we did in our previous work, to be able to experiment with large-scale overlays. The simulator uses Peer-to-Peer Protocol (P2PP) [3] as the protocol between the peers in the overlay. P2PP is used to exchange overlay maintenance messages and to store and retrieve data. The RELOAD [4] peer protocol, which is currently being standardized in the IETF, is based on P2PP. An unreliable transport protocol was assumed. All peers except for the bootstrap peer were assumed to be located behind a Network Address Translator (NAT) that uses endpoint-independent mapping and filtering behavior. Since peers were located behind NATs, they were able to establish direct connections between each other only by routing connection establishment messages across the overlay. The simulator uses the Chord DHT algorithm [5] to organize the overlay. Chord was chosen since the P2PSIP working group of the IETF specifies it as the mandatory-to-implement DHT [4]. The simulator uses a topology generator that assigns peers randomly to 206 different locations around the world. These locations correspond to PlanetLab sites. The pairwise delays between peers were set based on real pairwise delays that we measured between nodes at these PlanetLab sites. The same seeds were used for the random number generators throughout the simulations, meaning that the same topology and same delays were used in all simulation runs. This allowed us to fairly compare the performance of different optimizations. However, we also ran a series of simulations using different seeds to confirm the general validity of the results. These simulations, whose results are not included in the paper for brevity, confirm that our results remain valid even when different seeds are used.

A. Chord

Chord [5] is a DHT algorithm that uses SHA-1 as the base hash function to assign each peer and resource an *m*-bit identifier. The identifiers are ordered on an identifier circle of size 2^m, which is called the Chord ring. On the Chord ring, each peer maintains a routing table consisting of a finger table and neighbor table. In an *N*-node network, each peer maintains information about $O(\log N)$ peers in its finger table. The neighbor table consists of successor and predecessor lists. The successor list contains the peer's immediate successors on the Chord ring. In our Chord implementation, the size of the successor list was set to $\log N$. The predecessor list contains the immediate predecessors of the peer on the Chord ring. The size of the predecessor list was set to $\frac{1}{2} \times \log N$.

To ensure that the contents of the finger table, predecessor list, and successor list stay up-to-date with the constantly changing topology of the overlay, each peer runs a stabilization routine in the background periodically. As part of the stabilization routine, a peer synchronizes its predecessor list with its first predecessor and its successor list with its first successor, and incorporates new peers into its finger table.

B. P2PSIP operations

To join the P2PSIP overlay, a Joining Peer (JP) sends four different P2PP requests: Query, Join, Connect, and Publish. The Query request is used to obtain overlay-specific information. To join the overlay, the JP sends a Join request to an admitting peer (AP), which will become the JP's first successor on the Chord ring. Connect requests are used to establish direct connections between the JP and its neighbors in the overlay. The Publish request is used to store the JP's contact information in the overlay.

To initiate a call, the caller first sends a P2PP LookupObject request to fetch the contact address of the called user from the overlay. To leave the overlay, a peer first sends a P2PP RemoveObject request to remove its contact information from the overlay. When the RemoveObject transaction has finished, the peer sends two P2PP Leave requests in parallel to its first predecessor and its first successor.

C. Basic estimation mechanisms

This section describes the basic mechanisms that were used to calculate the network size, join rate, and leave rate estimates in the simulations. In later sections, we will describe how we modified these mechanisms to improve their accuracy.

1) Network size: As described in [6], the density of peer-IDs in Chord's neighbor table can be used to produce an estimate of the size of the overlay. To estimate the overlay size, a peer computes the average inter-peer distance *d* between the successive peers starting from the most distant predecessor and ending to the most distant successor. The network size estimate can then be calculated as:

$$N = \frac{2^m}{d},$$

where *m* is the number of bits in peer identifiers.

2) Join rate: A peer can estimate the join rate using the uptimes of the peers in its routing table [7]. An estimate of the global peer join rate λ can be calculated as:

$$\lambda = \frac{N}{4} \times \frac{1}{Ages[r/4]},$$

where *Ages* is an array containing the ages of the peers in the routing table sorted in increasing order and *r* is the size of the routing table.

3) Leave rate: The value of the leave rate μ can be estimated using peer failures in the routing table [6]. For this, each peer maintains a failure history with up to *K* entries. The leave rate estimate μ is calculated as follows:

$$\mu = \frac{k}{M \times T_k},$$

where *M* is the number of unique peer-IDs in the routing table, T_k is the time between the first and last failure in the history, and *k* is the number of failures in the history.

III. Related work

As described above, the estimation mechanisms presented in this paper use the work in [6], [7] as the starting point. In [8], we propose how these mechanisms can be used to extend the Chord-based DHT of the RELOAD [4] P2P signaling protocol to support adaptive stabilization.

A mechanism to determine various statistics from a running overlay, including network size and churn rate, is presented in [9]. The approach is active and is based on taking a snapshot of a running Chord system. Another active approach is presented in [10]. This mechanism is a gossip-based, aggregation-style algorithm that performs a random walk in the overlay. The general downside of active approaches is that peers need to actively collect data from the overlay by sending probe messages. Our goal is to avoid this extra probing.

In [11], a passive approach that is based on measuring density in the identifier space is presented. The approach is similar to [6]. In [12], a churn rate estimation mechanism that is, like [6], based on the changes a peer observes in its overlay neighbors, is presented. In [12], peers also share estimates with

TABLE I
JOIN AND DEPARTURE RATES

Period [s]	JR [1/s]	LR [1/s]	Max NS	Started
0-1800	0.1	0	180	180
1800-3600	0.2	0	540	540
3600-5400	0.5	0.1	1260	1440
5400-7200	1	0	3060	3240
7200-9000	1	0.25	4410	5040
9000-12600	2	0	11610	12240
12600-14400	2	1	13410	15840
14400-16200	0	2	9810	15840

TABLE II
TRAFFIC MODEL AND CHORD PARAMETERS

Parameter	Value
Network size (N)	1 - 10049
Duration	16200s (4.5h)
Busy hour call attemps	2.21 calls per user
% of calls to buddies	66.6
Mean size of buddy list	22
Finger pointers	14 $(logN, N = 10049)$
Successors	14 $(logN, N = 10049)$
Predecessors	7 $(0.5 * logN, N = 10049)$
% of leaves that are crashes	10
Stabilization interval	15s
Size of failure history	9
Keepalive interval	30s

TABLE III
OPTIMIZATIONS

Configuration	Optimizations
(1) Basic	No optimizations
(2) LR optimizations	Extended Failure History (FH), download FH, modified LR calculation
(3) Two NS estimates	Obtain secondary network size estimate
(4) Estimate sharing	Peers share their estimates
(5) All optimizations	Use percentiles instead of weighted averages, improved failure detection, modified calculation of initial LR estimate, old entries dropped from FH

their direct overlay neighbors. In [13], a localized size estimation mechanism is presented. However, since the accuracy of the estimator is within range $N/2..N^2$, it is not accurate enough for our purposes.

IV. EXPERIMENTS

A. Traffic model

Lookup traffic during the simulations consisted of lookups related to Voice over Internet Protocol (VoIP) calls and presence. Calls were modeled according to busy hour traffic volumes. Each user was assumed to initiate 13 VoIP calls per day, as suggested in [14]. Out of these 13 calls, 17% were used to represent busy hour traffic [15]. Thus, the number of busy hour call attempts per user was 2.21. This value was used as a mean rate for the arrival of calls, which was modeled as a Poisson process. Since users typically call their friends instead of strangers [16], it was assumed that 2/3 of the calls are placed to users on the buddy list.

To model churn, we used the join rates (JR) and leave rates (LR) listed in Table I. In the table, NS refers to network size. The last column contains the cumulative number of peers started before the end of each period. Max N refers to the theoretical maximum network size reached during each period assuming that all joins succeed. However, in practice, this size may not be reached since some of the joins fail due to instability caused by churn. The arrival and departure of users was modeled as a Poisson process. 10% of departures were crashes, whereas the rest were graceful departures. The number of buddies a user has was assumed to follow the power law distribution, as reported in [17], [18] with an average of 22. This size was chosen based on the results in [19]. After having joined the overlay, each user initiates lookups to fetch the contact information of her buddies from the overlay.

The size of Chord's successor list and finger table were set to $\log N$ using the maximum network size, N=10049. In Chord, roughly $\Omega(\log^2 N)$ rounds of stabilization should occur in the time it takes for N new peers to join or $N/2$ peers to leave the overlay [20]. The most challenging conditions in the simulation occur during the last 1800 seconds when the mean departure time between peers is 500ms. The above-mentioned rule can be used to calculate the appropriate stabilization interval for this period. The resulting value, 15s, was used throughout the simulations. The size of the failure history was set to 25% of routing table size [7] (i.e., to 9). Because peers are located behind NATs, they need to send periodic keepalive messages. A peer sends a keepalive request only if there has been no other traffic on a connection for 30 seconds. The traffic model and Chord parameters are summarized in Table II.

B. Optimizations

Table III lists the different configurations of optimizations used in the simulations. In configuration 1 (Basic), no optimizations were used. Instead, unmodified network size, join rate, and leave rate estimation algorithms were used. In configuration 2 (LR optimizations), various mechanisms were introduced to improve the leave rate estimate. In configuration 3 (Two NS estimates), each peer obtained a secondary network size estimate using locally available data to improve the accuracy of its network size estimate. In configuration 4 (Estimate sharing), peers shared their estimates. Finally, in configuration 5 (All optimizations), each peer used percentiles instead of weighted averages when forming its estimates from the shared values. Also, an improved failure detection mechanism was introduced, old failures were dropped from failure histories, and the calculation of the initial leave rate estimate was modified.

V. RESULTS

In this section, the results of the simulations are reported using two types of figures: figures showing estimates and figures showing average errors. The former types of figures plot the real value (i.e., real network size, join rate, or leave

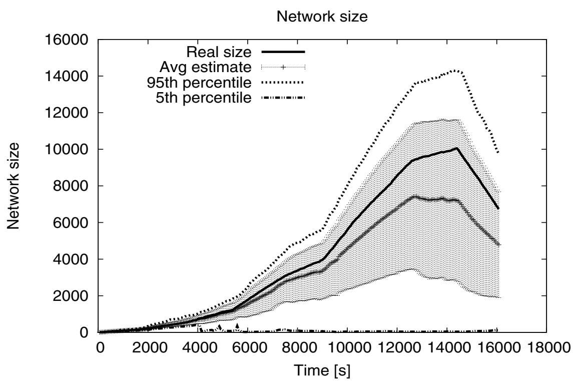

Fig. 1. Network size estimate, configuration 1 - basic

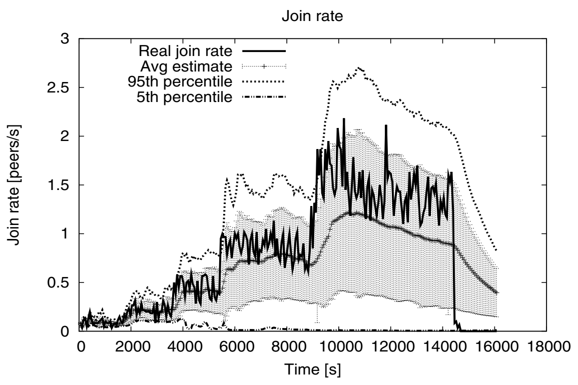

Fig. 2. Join rate estimate, configuration 1 - basic

rate), an average calculated over the individual estimates of the peers, and the 5th and 95th percentile estimates. The error bars shown for the average estimate represent the standard deviation. The samples were collected every 60 seconds. The average errors were calculated as follows for each 60s interval:

$$avg\ error = \frac{\Sigma_{i=0}^{n}|real - est_i|}{n},$$

where n is the total number of estimates, *real* is the real value of the variable being estimated, and est_i is the estimate of peer i. The error bars in the figures showing average errors represent 95% confidence intervals.

A. *Configuration 1 - basic*

In this measurement, unmodified network size, join rate, and leave rate estimation algorithms were used. The network size estimate is shown in Figure 1. From the figure, we can observe that the standard deviation is very high. On the average, the size estimate is wrong by 36.0%.

Figure 2 shows the average join rate estimate. From the figure, we can observe that initially, between 0-9000s, the average estimate follows closely the real join rate. However, the standard deviation of the estimate is rather high as indicated by the error bars. On the average, the join rate estimate is wrong by 54.4%.

Figure 3 plots the average leave rate estimate. From the figure, we can observe that the standard deviation of the

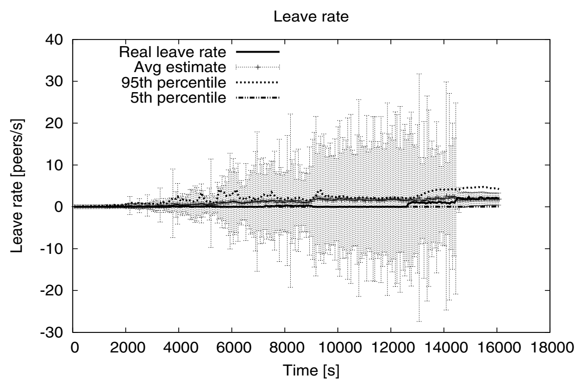

Fig. 3. Leave rate estimate, configuration 1 - basic

estimate is extremely high although it goes down for the last 30 minutes of the simulation. At that point, the join rate becomes zero for the first time in the simulation. This reduces the standard deviation, since the high variance is caused by the extremely poor leave rate estimates that peers with short uptimes produce. It takes time before newly joined peers gather enough failures to their failure histories to produce more accurate estimates. From the figure, we can see that most of the time, the average leave rate estimate completely fails to reflect changes in the real leave rate. On the average, the leave rate estimate is wrong by 293%.

B. *Configuration 2 - leave rate optimizations*

In this measurement, we focused only on improving the leave rate estimate since it was observed to be the most inaccurate estimate in the previous measurement. We applied several optimizations. First, the failure history was extended by maintaining information about the identities of failed peers and network size at the time when each failure occurred. Remembering the identities of failed peers is beneficial since a failure may be sometimes added to the history although the remote peer has not really failed. This can happen for instance if a timeout or routing error occurs due to temporary network instability caused by churn. Second, to address the problem with poor estimates produced by newly joined peers that was discussed in the previous section, joining peers downloaded the failure history of the admitting peer as part of the process of joining the overlay. Third, the leave rate calculation algorithm was modified in two ways. First, the entries in the failure history were allowed to age. The result of this change is that if new entries are not added to the history, the leave rate will eventually decrease. Second, rather than using the current network size estimate when calculating the leave rate, the size at the time when the failure occurred was used. These optimizations have an impact only on the leave rate estimate.

Figure 4 plots the resulting average leave rate estimate. From the figure, we can see that the standard deviation of the leave rate estimate has reduced considerably when compared to the previous measurement. However, the standard deviation is still unacceptably high. The average leave rate estimate now

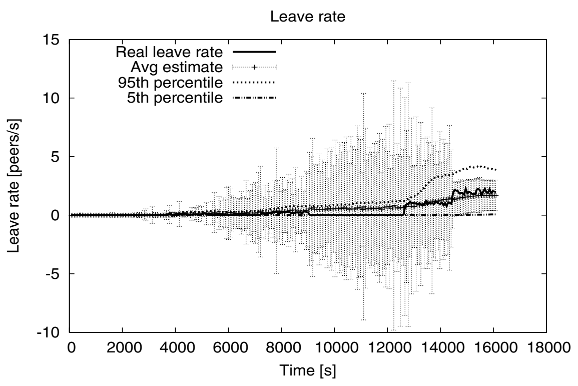

Fig. 4. Leave rate estimate, configuration 2 - leave rate optimizations

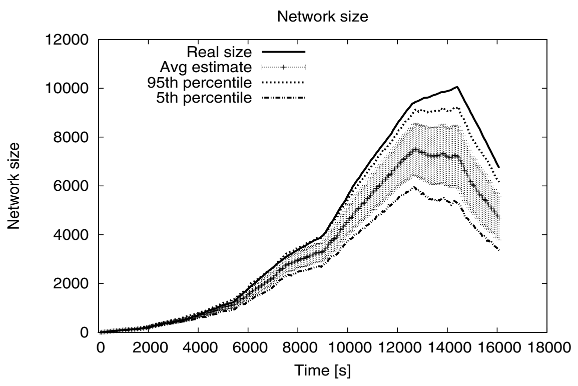

Fig. 5. Network size estimate, configuration 4 - estimate sharing

follows more closely the real leave rate. On the average, the estimate is still wrong by 116%.

C. *Configuration 3 - two network size estimates*

In this measurement, we studied whether the network size estimate can be improved if each peer obtains two independent estimates. The first, est_n, is calculated in the same way as in configuration 1 (Basic). The second estimate, est_f, is obtained by using the average distance of finger pointers from their ideal positions. The actual estimate is then calculated as a weighted average of these two estimates as follows:

$$NS = \frac{size_{nt}}{size_{nt} + size_{ft}} \times est_n + \frac{size_{ft}}{size_{nt} + size_{ft}} \times est_f,$$

where $size_{ft}$ is the number of peers in the finger table and $size_{nt}$ is the number of peers in the neighbor table.

The average errors of this mechanism and configuration 2 (LR optimizations) are shown in Figure 11 for the network size. The figure includes also the average errors of other configurations. Although statistically significant, the improvement obtained by using two estimates is very small; the average error decreases only by 3.3%. Consequently, in the remainder of the simulations, we did not use this optimization. However, it should be noted that using a secondary estimate not based on information in the neighbor table is useful if the neighbor table does not contain enough entries for producing an accurate size estimate.

D. *Configuration 4 - estimate sharing*

In this measurement, peers were allowed to share their network size, join rate, and leave rate estimates. This was done by piggybacking the estimates in all overlay stabilization messages. Every peer initiating or forwarding such a request or response added its estimates to the message. At the maximum, estimates of ten peers were allowed in each message to keep the additional bandwidth consumption low. Piggybacking a maximum of ten estimates was observed to result in 12% increase in bandwidth consumption, which we consider to be reasonable. It should be noted that the increase in bandwidth consumption can be controlled by limiting the number of estimates each message can carry. Weighted averages were used to form the actual estimates. The network size and join rate estimates were calculated as follows:

$$est = \Sigma_{i=0}^{n} \frac{u_i}{u_{all}} \times est_i,$$

where n is the number of received estimates, u_i is the number of unique peer-IDs in the routing table of peer *i*, u_{all} is the sum of received u_i values, and est_i is the estimate of peer *i*. The leave rate estimate was calculated as follows:

$$LR = \Sigma_{i=0}^{n} \frac{u_i + f_i}{u_{all} + f_{all}} \times LR_i,$$

where f_i is the number of failures in the history of peer *i* and f_{all} is the sum of received f_i values.

Figure 5 shows the average network size estimate for this configuration. The first improvement one can observe is that the standard deviation of the estimate is considerably lower. Also the average error is significantly lower than for the previous mechanisms, as can be seen from Figure 11. However, there is still room for improvement, since on average, the size estimate is wrong by 22.5%. Interestingly, nearly all peers seem to underestimate the network size; after roughly 10000s, the 95th percentile estimate is always lower than the real network size.

Figure 6 shows the average join rate estimate. From the figure, we can observe that the standard deviation is considerably lower than in the previous measurements. However, during the period when the join rate is highest (i.e., between 9000 and 14400 seconds), peers seem to slightly underestimate the join rate. On the average, the join rate estimate is wrong by 35.7%.

Figure 7 shows the average leave rate estimate. Compared to Figure 4, we can observe that the standard deviation has reduced considerably, but is still rather high. It seems that the optimizations done in this configuration and configuration 2 (LR optimizations) are not enough to remove the problem caused by poor estimates of newly joined peers. A joining peer initializes its failure history by downloading the admitting peer's failure history. In some cases, the joining peer does not receive any estimates from other peers before calculating its first leave rate estimate. Thus, it forms its estimate using only the downloaded failure history. If the admitting peer has experienced an unusually high number of failures (or if it

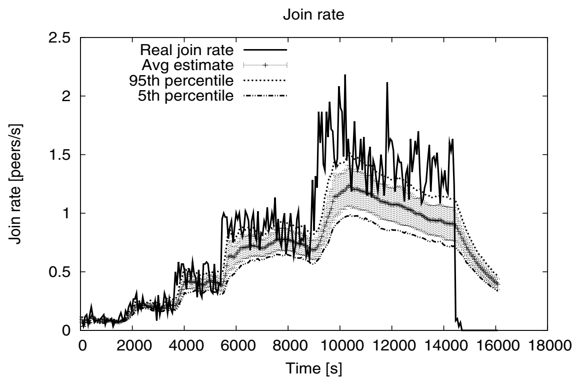

Fig. 6. Join rate estimate, configuration 4 - estimate sharing

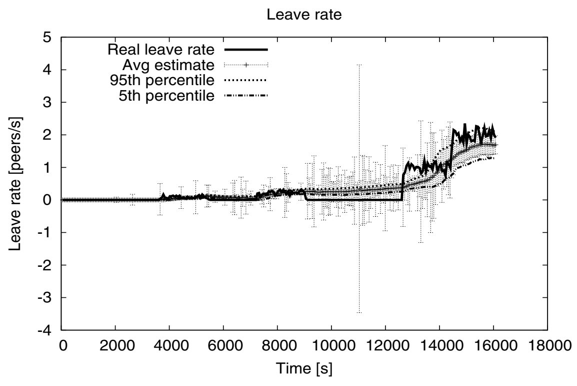

Fig. 7. Leave rate estimate, configuration 4 - estimate sharing

has not experienced any failures at all, or experienced a few failures in a short period of time), the joining peer's first leave rate estimate may be rather high or low (the admitting peer's estimate will not be as bad because the admitting peer can adjust its estimate using the leave rate estimates received from other peers). On the average, the leave rate estimate is wrong by 51.2%. The high standard deviation at 11040s is caused by three newly joined peers whose leave rate estimates are exceptionally large.

E. Configuration 5 - all optimizations

In this configuration, peers used the 75th percentile of the received estimates as the actual estimate. We used the 75th percentile rather than the median to be able to capture increases in network size, join rate, and leave rate faster. Further, the failure detection mechanism was improved so that it handles differently timeouts that occur because of a routing error and timeouts that occur because a peer has left the overlay or crashed. The mechanism was also modified to handle differently connections that are terminated because a peer departs from the overlay and connections that are terminated since the remote peer is no longer within the appropriate range in the routing table. Also, the process of calculating the initial leave rate estimate was modified by taking into account the estimate of the admitting peer in addition to relying on the received failure history. Finally, if

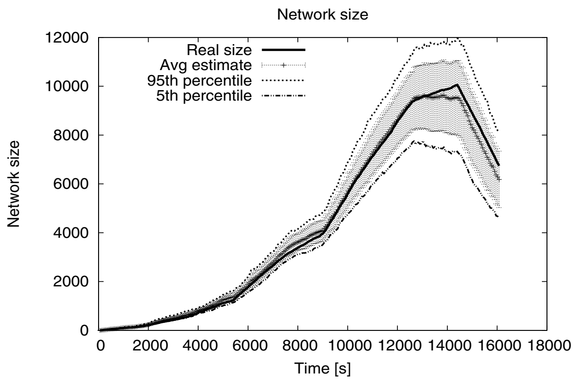

Fig. 8. Network size estimate, configuration 5 - all optimizations

before adding a new failure, the newest entry in the history was older than T seconds, all entries except for the newest one were removed. T was set to 1800s. The purpose of this modification is to speed up detection of increases in leave rate after long periods with no departures.

Figure 8 plots the average network size estimate. When comparing the figure to Figure 5, we can see that the situation has improved; the average network size estimate now follows very closely the real network size. Further, from Figure 11, we can observe that the average error is significantly lower for this configuration. On the average, the size estimate is wrong by only 10.6%.

Figure 9 shows the average join rate estimate. When compared to the results of the previous measurement shown in Figure 6, one can observe that the average estimate no longer underestimates the real join rate (in fact, the estimate now slightly overestimates it). The estimate also detects the sudden increases in join rate at 3600s, 5400s, and 9000s much faster. Further, the average error is lower than for the other configurations. The average errors of the worst (Configuration 1 - Basic) and best (Configuration 5 - All optimizations) join rate estimation mechanisms are plotted in Figure 12. The average error of configuration 5 (All optimizations) is clearly lower except for the last 1800 seconds of the simulation. The higher average error during this interval is explained by the fact that the estimate is higher at the beginning of the interval, which is also the moment when the join rate drops to zero. Since calculation of the join rate estimate is based on the average age of peers in the routing table, the estimate starts to slowly decrease with time. The rate at which this happens is the same for both of the configurations. Since the decrease starts from a higher level for configuration 5 (All optimizations) (because configuration 5 had a more accurate estimate), the error is higher for configuration 5. On the average, the estimate is wrong by 35.2%.

Figure 10 plots the average leave rate estimate. From the figure, we can see that the standard deviation is dramatically lower than for configuration 4 (Estimate sharing) (see Figure 7). On the average, the estimate is accurate within 47.6% of the real leave rate. From the figure, we can also see that the

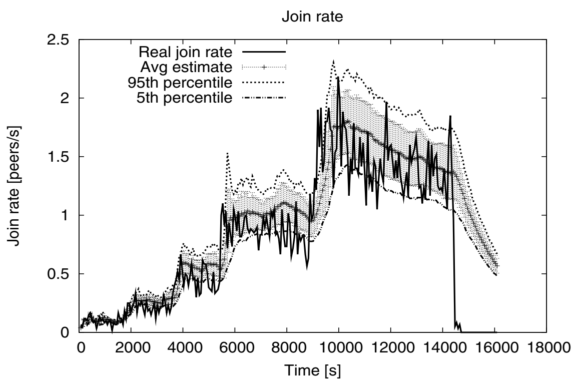

Fig. 9. Join rate estimate, configuration 5 - all optimizations

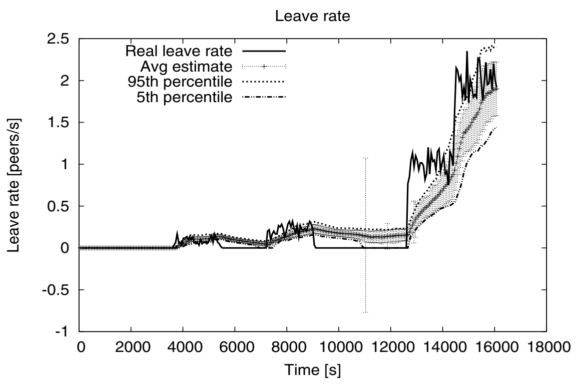

Fig. 10. Leave rate estimate, configuration 5 - all optimizations

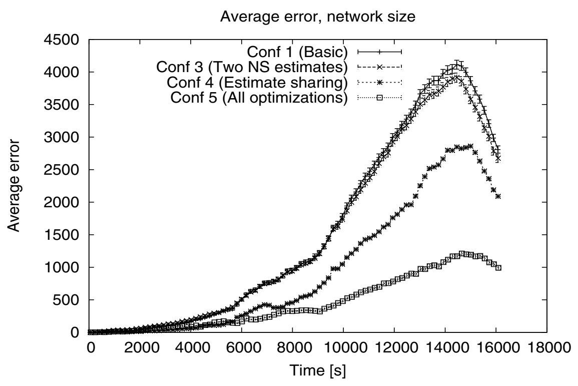

Fig. 11. Average errors of network size estimates

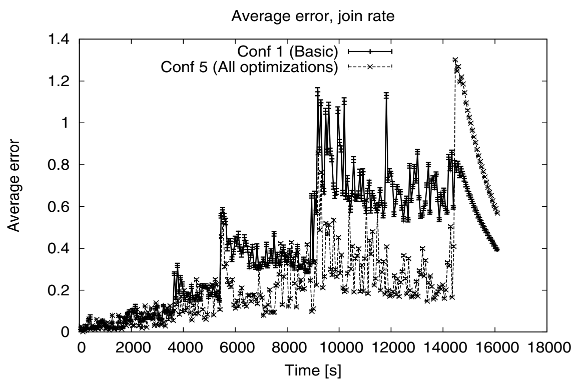

Fig. 12. Average errors of join rate estimates

TABLE IV
AVERAGE INACCURACY

Configuration	NS [%]	JR [%]	LR [%]
1 (Basic)	36.0	54.4	293
2 (LR optimizations)	36.0	54.4	116
3 (Two NS estimates)	34.8	53.9	121
4 (Estimate sharing)	22.5	35.7	51.9
5 (All optimizations)	10.6	35.2	47.6

average estimate reacts rather slowly to the sudden increases in leave rate at 12600s and 14400s.

F. Comparison

The average inaccuracy of the network size, join rate, and leave rate estimates of each configuration are shown in Table IV. As an example, the inaccuracy of the network size estimate of configuration 1 (Basic) is 36.0%, meaning that on average, the estimate is accurate within 36.0% of the real network size. The average inaccuracy was calculated by dividing an average calculated over the distances of all estimates during the simulation from the corresponding real value by the average of the real value.

Figure 11 plots the average errors of the different size estimation mechanisms. The error bars are so small that they are not visible. Configuration 2 (LR optimizations) is not shown separately since the estimate it produces is identical to configuration 1 (Basic). From the figure and from Table IV, we can conclude that configuration 5 (All optimizations) clearly outperforms the other configurations. The difference is statistically significant after roughly 6000s.

Figure 12 plots the average errors of the worst (Configuration 1 - Basic) and best (Configuration 5 - All optimizations) join rate estimation mechanisms. Based on the figure and Table IV, we can conclude that configuration 5 (All optimizations) outperforms other configurations. If looking only at the average inaccuracy, the improvement over configuration 4 (Estimate sharing) is very small. However, the most important benefit of configuration 5 (All optimizations) is that it detects sudden increases in the join rate much faster than configuration 4 (Estimate sharing).

The average errors of the worst (Configuration 1 - Basic) and best (Configuration 5 - All optimizations) leave rate estimation mechanisms are plotted in Figure 13. Based on the figure and Table IV, we can conclude that configuration 5 (All optimizations) outperforms the other configurations.

VI. CONCLUSION

In this paper, we studied how to improve the accuracy of mechanisms estimating the operating conditions of a running P2P overlay network. We started from existing overlay size, join rate, and leave rate estimation mechanisms and showed that their level of accuracy is not satisfactory. We proposed several optimizations to improve the mechanisms, including

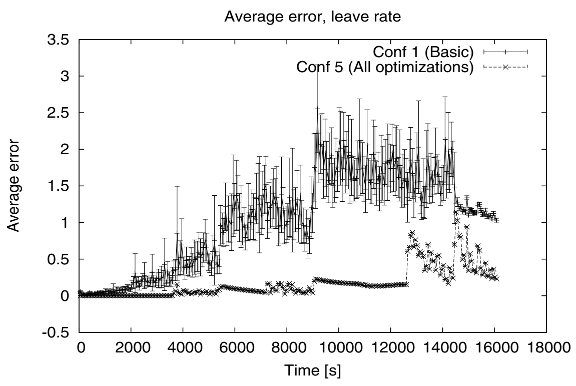

Fig. 13. Average errors of leave rate estimates

extended failure histories, downloading of failure histories, aging of leave rate estimates, secondary network size estimates, estimate sharing, statistical mechanisms to process shared estimates, periodic flushing of failure histories, and copying of leave rate estimates. When all the optimizations are applied, the improvement is 239% for network size, 55% for join rate, and 515% for the leave rate estimate.

All of the proposed optimizations use passive techniques to produce the estimates. One important finding is that if peers rely only on local information, a subset of the peers will produce very bad estimates. Thus, in practice, peers need to share their estimates by piggybacking them to overlay stablization messages and use statistical mechanisms to determine the actual estimate from the shared data. The additional bandwidth consumption caused by piggybacking of estimates can be kept low by limiting the number of estimates a single message can carry. We showed that with the proposed optimizations, passive mechanisms can achieve a satisfactory level of accuracy.

The main use cases for estimation of operating conditions include performance monitoring and adaptive behavior. Performance monitoring is useful for operators of real-world overlays (e.g., P2PSIP overlays). Adaptive behavior is needed to make structured overlays scalable, that is, to allow them to adapt to changing operating conditions. Even with all the optimizations, the estimates cannot achieve a zero error rate. However, they are still accurate enough to be valuable in the above-mentioned use cases.

REFERENCES

[1] J. Mäenpää and G. Camarillo, "Study on maintenance operations in a peer-to-peer session initiation protocol overlay network," in *Proc. of IPDPS*, 2009.

[2] J. Mäenpää and G. Camarillo, "Analysis of delays in a Peer-to-Peer session initiation protocol overlay network," in *Proc. of 2010 IEEE Consumer Communications and Networking Conference (CCNC)*, Las Vegas, USA, 1 2010.

[3] S. Baset, H. Schulzrinne, and M. Matuszewski, "Peer-to-peer protocol (p2pp)," IETF, Internet Draft – work in progress, November 2007.

[4] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, and H. Schulzrinne, "Resource location and discovery (reload) base protocol," IETF, Internet Draft – work in progress, March 2009.

[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. Netw.*, vol. 11, no. 1, pp. 17–32, 2003.

[6] R. Mahajan, M. Castro, and A. I. T. Rowstron, "Controlling the cost of reliability in peer-to-peer overlays," in *IPTPS*, 2003, pp. 21–32.

[7] G. Ghinita and Y. M. Teo, "An adaptive stabilization framework for distributed hash tables," in *Parallel and Distributed Processing Symposium, International*, vol. 0. Los Alamitos, CA, USA: IEEE Computer Society, 2006, p. 12.

[8] J. Mäenpää, G. Camarillo, and J. Hautakorpi, "A self-tuning distributed hash table (dht) for resource location and discovery (reload)," IETF, Internet Draft – work in progress 01, October 2009.

[9] G. K. A. Binzenhofer and R. Henjes, "A scalable algorithm to monitor chord-based p2p systems at runtime," in *Proc. of the 20th International Parallel and Distributed Processing Symposium (IPDPS 2006)*, Rhodes Island, 2006.

[10] T. M. Shafaat, A. Ghodsi, and S. Haridi, "A practical approach to network size estimation for structured overlays," in *IWSOS '08: Proceedings of the 3rd International Workshop on Self-Organizing Systems*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 71–83.

[11] D. Kostoulas, D. Psaltoulis, I. Gupta, K. Birman, and A. Demers, "Decentralized schemes for size estimation in large and dynamic groups," in *NCA '05: Proceedings of the Fourth IEEE International Symposium on Network Computing and Applications*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 41–48.

[12] A. Binzenhöfer and K. Leibnitz, "Estimating churn in structured p2p networks," in *International Teletraffic Congress*, 2007, pp. 630–641.

[13] K. Horowitz and D. Malkhi, "Estimating network size from local information," *Inf. Process. Lett.*, vol. 88, no. 5, pp. 237–243, 2003.

[14] 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. of the 16th international telecommunications society (ITS) European regional conference*, 2006.

[15] "Traffic analysis for voice over ip," 2001. [Online]. Available: http://www.cisco.com

[16] C. Cheng, S. Tsao, and J. Chou, "Unstructured peer-to-peer session initiation protocol for mobile environment," in *Proc. of the 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC '07)*, 2007, pp. 1–5.

[17] J. Leskovec and E. Horvitz, "Planetary-scale views on a large instant-messaging network," in *WWW '08: Proceeding of the 17th international conference on World Wide Web*. New York, NY, USA: ACM, 2008, pp. 915–924.

[18] C. D. Morse and H. Wang, "The structure of an instant messenger network and its vulnerability to malicious codes," in *Proc. of ACM SIGCOMM*, 2005.

[19] B. A. Nardi, S. Whittaker, and E. Bradner, "Interaction and outeraction: instant messaging in action," in *Proc. of the 2000 ACM conference on Computer supported cooperative work (CSCW '00)*. New York, NY, USA: ACM, 2000, pp. 79–88.

[20] D. Liben-Nowell, H. Balakrishnan, and D. Karger, "Observations on the dynamic evolution of peer-to-peer networks," in *Proc. of the First International Workshop on Peer-to-Peer Systems (IPTPS '01)*. London, UK: Springer-Verlag, 2002, pp. 22–33.