

# Peer-to-Peer Single Hop Distributed Hash Tables

Luiz Monnerat <sup>\*,§</sup>

luiz.monnerat@petrobras.com.br

<sup>\*</sup>Petrobras

TIC/TIC-E&P/STEP

Claudio Amorim <sup>§</sup>

amorim@cos.ufrj.br

<sup>§</sup>COPPE - Computer and Systems Engineering  
Federal University of Rio de Janeiro (UFRJ)

**Abstract**—Efficiently locating information in large-scale distributed systems is a challenging problem to which Peer-to-Peer (P2P) Distributed Hash Tables (DHTs) can provide a highly scalable and cost-effective solution. However, there is very little experience on using DHTs in performance sensitive environments such as High Performance Computing (HPC) datacenters, and there is no published experimental comparison among low-latency DHTs. To fill this gap, we conducted an in-depth performance comparison of three proposed low-latency single-hop DHTs namely 1h-Calot, D1HT, and OneHop. Specifically, we compared experimentally the lookup latency and CPU use of D1HT with those of 1h-Calot by running each of them concurrently with the normal workload production for a subset of 1,800 nodes of a heavy-loaded HPC datacenter. In addition, we carried out an analytical performance comparison among the three single-hop DHTs for system sizes of up to 10 million nodes. The results showed that D1HT consistently had the smallest overhead and in most cases it required one order of magnitude less bandwidth than 1h-Calot and OneHop. Overall, the combination of our experimental and analytical results suggests that D1HT can provide a very effective solution for a broad range of environments, from large-scale HPC datacenters to widely deployed Internet P2P applications such as BitTorrent with up to one million peers. This ability to support such a wide range of environments may allow D1HT to be used as an inexpensive and scalable commodity software substrate for large-scale distributed applications<sup>1</sup>.

## I. INTRODUCTION

Efficiently locating information in distributed systems is becoming a challenging problem as those environments continuously and quickly grow in size, while the performance requirements steadily become tighter. Fortunately, Distributed Hash Tables (DHTs) are an inexpensive, self-organizing and highly scalable solution for this problem, so that they have been used in a plethora of systems ranging from Internet games [10] to databases [8].

DHT systems provide a hash-table abstraction on top of information dispersed in a distributed environment, and are typically implemented using a peer-to-peer (P2P) approach, where all nodes are able to perform the same functions. The information stored in a DHT is located through lookup requests, which are routed from its *origin* (the peer that issued the lookup) to its *target* (the peer in charge of the requested information). For this reason, DHTs implement overlay networks with the use of *routing tables* stored on all participant peers. The size of these routing tables represents a latency vs. bandwidth tradeoff, since the bigger are the routing

tables the faster should be the lookups (as the peers will have more options from where to choose the routes) but the higher should be the *maintenance traffic* (network traffic necessary to keep the routing tables up to date as nodes join and leave the system).

As similar tradeoffs between latency and bandwidth occur across several other technologies, in the long term the latency restrictions tend to be more critical, since it has been shown that ‘*over time bandwidth typically improves by more than the square of the latency reductions*’ [20]. In contrast, most DHT designs so far (e.g. [22], [26], [29], [32]) have opted for a midterm point in this tradeoff, where nodes use small routing tables but each lookup takes  $O(\log(n))$  hops to be solved ( $n$  is the system size), sacrificing performance and preventing the use of these multi-hop DHTs for performance sensitive applications. In contrast, *single-hop* DHTs are able to provide low latency access to information since each peer has a routing table with the IP addresses of all other peers in the system, and so the origin of a lookup is able to identify its target without the need to contact additional peers. Even though those full routing tables should demand more bandwidth in terms of maintenance traffic, it has been shown that in some cases single-hop DHTs may in fact reduce the *total* bandwidth consumption in relation to multi-hop DHTs, specially for systems with a high frequency of lookups [11], [30]. This happens because each lookup in a multi-hop DHT typically consumes  $O(\log(n))$  more bandwidth than a single-hop lookup, and this extra lookup overhead may offset the routing table maintenance traffic.

To address those issues, a number of low overhead single-hop DHTs have been proposed [7], [17], [25], [30], aiming to provide low latency with acceptable maintenance traffic. As a result, those systems may not only provide scalability, cost and manageability advantages that are inherent to DHTs, but are also able to achieve good performance coupled with low maintenance overheads, allowing their use not only by Internet wide systems, but also in performance sensitive distributed environments such as High Performance Computing (HPC) and Internet Service Providers (ISPs). HPC environments typically have several thousand stable nodes [31], where latency is usually a main concern and central directories may become a bottleneck (besides being expensive and requiring careful management and tuning), in a way that the conjunction of performance requirements and node stability make single-hop DHTs a very attractive option. In ISPs and several other

<sup>1</sup>This research was partially sponsored by Brazilian CNPq and FINEP.

enterprise data centers, such as AOL, Google and Yahoo among others [2], [4], the combination of thousands of stable nodes and high lookup rate makes single-hop DHTs an option with potentially less bandwidth overhead, lower cost and better performance than multi-hop DHTs and centralized directories [25].

In this paper we provide several contributions that will allow better understanding and use of single-hop DHTs in a wide range of environments. First, we implemented the D1HT [16] and 1h-Calot [30] pure P2P DHT systems, allowing us to perform the first experimental comparison among single-hop DHTs. Second, we made our D1HT implementation freely available for download and use [5]. Third and most important, we evaluated both D1HT and 1h-Calot in a network of 1,800 *physical* nodes, which is the most extensive experimental DHT evaluation made so far. Those experiments provided very important results, as they not only confirmed that both systems are able to solve a very large fraction of the lookups with a single hop, but also validated the published analyses for both systems [16], [30] as either precise or conservative. More interestingly, all experiments were made in a typical HPC data center with all the nodes under normal production (composed of CPU-hungry seismic processing applications), confirming that those DHTs have negligible CPU demands which allow their use even in heavy loaded nodes.

As our experiments validated the D1HT and 1h-Calot analyses, we further performed an analytical comparison between D1HT, 1h-Calot, and the OneHop DHT for system sizes of up to 10 million peers. Note that OneHop was the very first proposal for a single-hop DHT, and which had its analytical results already experimentally validated in a previous work [7]. Our analytical results showed that D1HT consistently had the lowest maintenance bandwidth requirements for different churn rates and system sizes ranging from 10 thousand to 10 million peers, with overhead reductions of up to one order of magnitude in relation to both OneHop and 1h-Calot. The results also indicated that D1HT is able to support vast distributed environments with dynamics similar to those of widely deployed P2P applications, such as Gnutella [28] and BitTorrent [21], with reasonable maintenance bandwidth demands.

As a consequence of our experimental and analytical results, we could conclude that D1HT consistently has the lowest overheads among the single-hop DHTs studied, and that it can potentially be used in a multitude of environments ranging from HPC data centers to huge P2P applications deployed over the Internet.

The remainder of this paper is organized as follows. In the next section we discuss the related work. Section III describes the features that are common to the systems studied in this work, while Sections IV, V, and VI present the particular features of D1HT, 1h-Calot, and OneHop, respectively. Sections VII and VIII present our experimental and analytical results, and we then conclude the paper.

## II. RELATED WORK

As far as we are aware, this is the first work to study and evaluate DHTs experimentally for performance-sensitive environments, and it will present an experimental evaluation of D1HT and 1h-Calot with up to 1,800 distinct physical nodes in an HPC datacenter. Since, up to now, the DHT evaluations with real implementations used a few hundred *physical* nodes at most [6]–[8], [23], [24], [32], then the present evaluation can be regarded as the broadest and most representative DHT experiments ever reported, besides being the first to compare two single-hop DHTs.

Although there are a number of published DHT analytical evaluations (e.g., [7], [11]–[13], [15], [16]), this work presents the first analytical comparison among three different single-hop DHTs. Besides comparing all single-hop DHTs designed to support dynamic environments (D1HT, 1h-Calot, and OneHop), our analytical comparison should also be valid for the 1HS system, once it has been proposed to use the 1h-Calot maintenance algorithm [25].

## III. SINGLE-HOP DHTS

We will now describe the features that are common to D1HT, 1h-Calot, and OneHop, while in Sections IV, V, and VI we will discuss the particularities of each system.

For D1HT, 1h-Calot, and OneHop, each DHT instance is composed of a set of  $n$  peers, and the keys (information) are mapped to peers using *consistent hashing* [9], where both peers and keys have IDs taken from the same identifier ring  $[0 : N]$ , with  $N \gg n$ . The key and peer IDs are respectively the SHA1 [18] hash of the key values and peer IP addresses.

As each peer has a full routing table, any lookup can be solved with just one hop, if the routing table is up to date. But if the origin peer is unaware of a membership change that had happened in the vicinity of the target peer (a node has joined or left the system), the lookup may be initially addressed to a wrong peer or to a peer that has already left the system. In both cases the lookup will eventually succeed, provided that each peer in the ring has a pointer to its correct successor [29], but it will take longer than initially expected. To completely avoid the occurrence of those *routing failures* (as the lookup will eventually succeed, we do consider it as a *routing failure* instead of a *lookup failure*), the DHT system would have to be able to immediately notify all its  $n$  peers about the occurrence of any *event* in the system (from now on we will refer to peer joins and leaves simply as *events*), which is simply infeasible. In practice, single-hop DHTs must try to keep the fraction of those routing failures below an acceptable maximum (e.g., the user may impose that at least 99% of the lookups must be solved with only one hop), by implementing mechanisms that are able to quickly notify all peers in the system about the events as they happen. These event dissemination mechanisms represent the primary distinction about the systems studied (and so they will be discussed in the next sections), specially in relation to the maintenance traffic required. We should also point out that while D1HT and 1h-Calot use a pure P2P architecture for both

the lookups and the maintenance algorithm, OneHop is pure P2P solely in relation to the lookups, as it uses a hierarchical approach for event dissemination.

DHT systems must provide a *joining protocol* to assure that any new peer is correctly inserted in the ID space and gets the appropriate keys and routing table, but this mechanism is usually orthogonal in relation to the DHT system itself. In this way, a DHT proposal may only briefly describe the joining protocol, leaving its details to be defined at the implementation level. To ensure fairness, in this work we will assume that the three systems studied use the same joining protocol, which is based on that of Chord [29] with some variations. In Section IV-B we will provide the details of the joining protocol implemented for D1HT and 1h-Calot.

As in previous works [7], [16], we will assume that the systems will be churned with an event rate (or *churn rate*) that is directly proportional to the system size according to Equation III.1, where  $S_{avg}$  is the peer average session length. Here, we refer to *session length* as the amount of time between a peer join and its subsequent leave, and so Equation III.1 simply assumes that, as expected, each peer generates two events per session (one join and one leave).

$$r = 2 \cdot n / S_{avg} \quad (III.1)$$

We should point out that Equation III.1 does not assume that  $r$  is fixed, as  $S_{avg}$  can vary to address systems where the peer behavior changes over time (e.g., during nights or weekends). Each of our experiments and comparative analysis presented in Sections VII and VIII will assume a single value for  $S_{avg}$ , but we performed experiments and analyses with different session lengths in order to evaluate its effect on the systems behaviors.

With the exception of the 1h-Calot heartbeat messages (more details on Section V), any message should be acknowledged to allow for retransmissions in the case of failures, which can be done implicitly by a protocol like TCP, or be explicitly implemented over a non reliable protocol like UDP. We will assume that all maintenance messages are sent using UDP in order to minimize their network impact.

#### IV. D1HT

In this section we will briefly describe the D1HT system design and analysis (more details can be found in [16]), and introduce our D1HT implementation.

##### A. System Design

In D1HT, the events are detected and propagated according to the proposed *Event Detection and Report Algorithm* (EDRA), which is able to notify any peer join or leave to the whole system with low overhead and good load-balance properties by means of logarithmic dissemination trees. Besides, EDRA groups several events per message in order to save bandwidth.

At first glance, grouping events in a single message seems to be an obvious and easy way to save bandwidth, since any peer can locally buffer the events happened during a period of time and forward them in a single message. But such

mechanism imposes delays in the propagation of the events, which in turn will result in more stale entries in the routing tables. So, the hard question is: *For how long the DHT system may buffer events while assuring that the vast majority of the lookups (e.g. 99%) will be solved with just one hop?* This problem is especially difficult since the answer depends on a number of factors that may vary unpredictably, including the system size and churn rate. EDRA address this issue based on formally proven properties of a theorem introduced in [17], which allowed it to dynamically calculate how long it can buffer events while assuring that at least a fraction  $1-f$  of the lookups will be solved with a single hop ( $f$  can be defined according to the application needs, and it is typically 1%).

The original D1HT paper formally describes EDRA by means of eight well defined rules, and introduces two theorems that provided the basis for the D1HT analysis, including conservative equations for calculating the average number of maintenance messages and bytes transmitted per peer, as follows (more details about the equations below can be found in [16]):

$$\Theta = (2 \cdot f \cdot S_{avg} - 2 \cdot \rho \cdot \delta_{avg}) / (8 + \rho) \text{ secs} \quad (IV.1)$$

$$N_{msgs} = 1 + \sum_{i=1}^{\rho-1} 1 - (1 - 2\Theta/S_{avg})^i, \text{ where } i = 2^{\rho-i-1} \quad (IV.2)$$

$$B_{D1HT} = (N_{msgs} \cdot (v_m + v_a) + 2 \cdot n \cdot m \cdot \Theta / S_{avg}) / \Theta \text{ bps} \quad (IV.3)$$

where  $B_{D1HT}$  is the average maintenance bandwidth demand by peer,  $m$  is the number of bits to describe an event,  $v_m$  and  $v_a$  are respectively the bit overheads (i.e. headers) per maintenance and acknowledgment message (as shown in Figure 1),  $S_{avg}$  is the average session length,  $\rho$  is  $\log_2 n$ , and  $\delta_{avg}$  is the average message delay. In these equations,  $\Theta$  is the maximum period of time a peer can buffer events and, as a consequence, it is also the maximum interval between two maintenance messages sent by a peer.

Equations IV.1, IV.2 and IV.3 were derived after several conservative assumptions, including that each peer event would be detected after  $2\Theta$  secs *on average* (i.e.,  $T_{detect}=2\Theta$ ). But in fact, this assumption does not reflect the average but the worst case of ungraceful departures (i.e., peer failures), where after one missing maintenance message from its predecessor ( $\Theta$  secs), a peer would probe it for other  $\Theta$  secs in order to confirm its leave. Since all joins and graceful leaves are immediately detected, a more realistic assumption would be given by the equation below (which was validated by the experimental results presented in Section VII), where  $k$  is the fraction of ungraceful peer leaves (i.e.,  $k \leq 1$ ).

$$T_{detect} = k \cdot \Theta \quad (IV.4)$$

We should point out that, since the results presented in [16] showed that  $\Theta$  should vary from 2 secs (for systems with  $n=10^7$  and  $S_{avg}=1$  hour) to 16 secs (for  $n=10^4$  and  $S_{avg}=13$  hours), D1HT is able to detect ungraceful leaves much faster than 1h-Calot, which relies on heartbeat messages sent at one minute intervals to detect peer failures (more details in Section V).

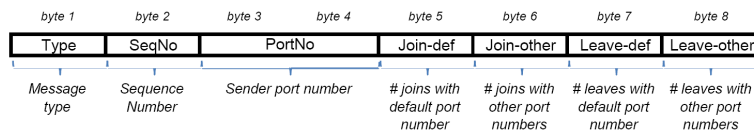


Fig 1.a: D1HT and OneHop message headers

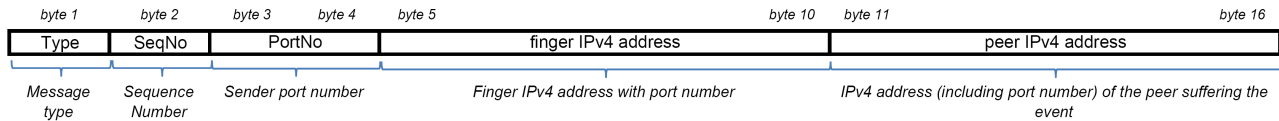


Fig 1.b: 1h-Calot message headers.

Fig. 1. Headers for maintenance messages used in our implementations and analytical evaluations. The SeqNo field is necessary to assure message delivery over UDP. Each 1h-Calot maintenance message has a fixed size of 44 bytes ( $v_c=352$  bits, including 28 bytes for the IPv4 and UDP headers). Each D1HT and OneHop message has a fixed part with 36 bytes ( $v_m=288$  bits, including IPv4 and UDP headers), followed by the IPv4 addresses (without port numbers) of the peers that have joined and left the system in the default port ( $m=32$  bits), and the IPv4 addresses (with port numbers) of the peers that have joined and left the system using others ports ( $m=48$  bits). All acknowledgment and heartbeat messages for the three systems have just the three first fields shown (Type, SeqNo and PortNo), and so  $v_a=v_h=256$  bits (including IPv4 and UDP headers).

## B. D1HT Implementation

As part of this work, we implemented D1HT from scratch, resulting in more than 8.000 lines of dense C++ code (without comments), even though we did not implement Quarantine [16]. This implementation is fully functional and was tested on thousands of nodes running Linux 2.4 and 2.6. Since the original D1HT proposal [16] did not detail the joining mechanism, leaving it to be fully defined at the implementation level, we decided to use a variant of the Chord joining protocol [29] with a few but important differences. First, any join is notified to the whole system by EDRA. Second, the new peer  $p$  gets the full routing table from its successor  $p_s$ . Third, in order to avoid  $p$  from missing events while its joining is notified to the system,  $p_s$  will forward to  $p$  any event it knows until  $p$  receives maintenance messages with all different TTLs.

As in many other systems (e.g., [7], [13], [14]), in order to improve the accuracy of the routing tables without additional overheads, each D1HT peer is able to learn from routing failures and messages received. For example, a maintenance message or a lookup received from an unknown peer will imply in its insertion in the local routing table.

For the implementation, we extended the original D1HT analysis in order to allow each peer to calculate the number of events it may buffer ( $E$ ), according to Equation IV.5. The equation below was derived from the analysis presented in [16] with the additional assumption that peers in a D1HT system observe similar event rates, as a consequence of the fact that all peers are notified by any event, and so we assumed that  $r = E/\Theta$ .

$$E = (8 \cdot f \cdot n) / (16 + 3 \cdot \rho) \quad \text{events} \quad (\text{IV.5})$$

In our implementation, each D1HT instance has a default IPv4 port (e.g., 8000), but any peer is able to choose an alternative port when joining the system. As a consequence, most events will be identified only by the peer's four bytes IPv4 address (as most peers should use the default port), which led us to propose the message header layout as shown in Figure 1. As a consequence, for Equation IV.3 we expect that  $m$  will be

near 32 bits on average.

Each peer stores its routing table as a local hash table indexed by the peer IDs, in a way that any peer needs only to store the IPv4 addresses of the participant peers (including port number), leading to a memory overhead of about  $6n$  bytes per peer (plus some additional space to treat eventual hash collisions). In this way, for environments such as HPC and corporate data centers, each routing table will require a few hundred KBs at most. For a huge one million Internet wide D1HT deployment, each routing table would require around 6 MB, which is negligible for domestic PCs, while being acceptable even for small devices such as modern cell phones and media players.

## V. 1H-CALOT

As D1HT, the 1h-Calot system [30] uses a logarithmic tree to propagate events, but it has a number of fundamental differences. First, 1h-Calot builds its propagation trees based on peer ID intervals, while D1HT uses message TTLs. Second, 1h-Calot uses explicit heartbeat messages to detect node failures, while D1HT relies on the maintenance messages. Third and most important, 1h-Calot is not able to buffer events, and so the propagation of each event requires  $2n$  messages (including acknowledgments).

As each 1h-Calot maintenance message carries just one event, it does not make sense to include counters in its message headers, which will then have the format shown in Figure 1, where each event will require the full IPv4 peer address (including port number) to be described. As each peer sends two heartbeats per minute (one to its predecessor and the other to its successor), and those heartbeats are not acknowledged, we have that the analytical average 1h-Calot peer maintenance bandwidth is given by:

$$B_{Calot} = (r \cdot (v_c + v_a) + 2 \cdot n \cdot v_h / 60) \quad \text{bps} \quad (\text{V.1})$$

where  $v_c$ ,  $v_a$ , and  $v_h$  are, respectively, the sizes of the maintenance, acknowledgment, and heartbeat messages (as shown in Figure 1).

Cluster	# nodes	CPU	OS
A	731	Intel Xeon 3.06GHz	Linux 2.6
B	924	AMD Opteron 270	Linux 2.6
C	128	AMD Opteron 244	Linux 2.6
D	99	AMD Opteron 250	Linux 2.6

TABLE I

CLUSTERS USED IN OUR EXPERIMENTS. EACH NODE HAS TWO CPUS, WHICH ARE DUAL-CORE IN CLUSTER B AND SINGLE-CORE OTHERWISE.

Our 1h-Calot implementation was developed after our D1HT C++ code. To allow for a fair experimental comparison between 1h-Calot and D1HT, both systems share most of the code, assuring that any difference in the results was not due to implementation issues.

## VI. ONEHOP

In OneHop [7], the dissemination of events is based on a hierarchy, where the nodes are grouped in *units*, which in turn are grouped in *slices*. As each unit and slice has a *leader*, this hierarchy divides the nodes in three levels: *slice leaders*, *unit leaders*, and *ordinary nodes*, but this topology incurs in high levels of load imbalance among the different type of nodes. Additionally, to achieve its best performance, all nodes in an OneHop system must agree in relation to some system wide topological parameters [30], which should be difficult to implement in practice, especially as the best parameters may change over time according to the system size and behavior.

The OneHop analysis is available from [7], where it was considered that each maintenance message had a fixed part with 20 bytes (while only the IPv4 and UDP headers require 28 bytes), and each event required 10 bytes to be described (but our D1HT implementation has shown that we need only the peer IP address). In this way, for our OneHop analysis we will consider the same message formats used in our D1HT implementation, as shown in Figure 1, which have been shown to be realistic in practice.

## VII. EXPERIMENTAL EVALUATION

In this section, we will present the experimental results obtained with our D1HT and 1h-Calot implementations.

### A. Methodology

The evaluation of the two systems presented in this section is based on their real implementation developed as part of this work and described in Sections IV-B and V. The test environment, as detailed in Table I, was comprised of 1,882 nodes distributed among four clusters used for Seismic Processing jobs at a Petrobras data center, which is very representative of an HPC environment [19]. In this network, each node has a Gigabit Ethernet connection to an edge switch. Each edge switch concentrates 16 to 24 nodes and has a 2 Gbps or 4 Gbps Ethernet trunk connection to a non-blocking core switch.

As already discussed in Section II, this environment allowed us to develop the broadest and most representative DHT evaluation ever reported. Besides, the experiments were performed with all the clusters under normal production, where typically

most of the nodes are experimenting 100% CPU use, as imposed by the Seismic Processing parallel jobs. Nevertheless, we were able to run all our experiments smoothly, without any single interference in the normal data center production, due to the very low overheads of D1HT and 1h-Calot on the network, memory, and CPU. In this way, those experiments showed that it is absolutely feasible to run these DHT systems in heavy loaded environments such as HPC data centers. In fact, to support the environment with 1,800 peers, we defined the routing tables with 2,000 entries each, and so the routing table memory overhead per peer was a mere 2KB.

Each experiment evaluated one DHT system (D1HT or 1h-Calot) with a given network size (600, 1200 or 1800 physical nodes) and different churn rates as given by Equation III.1 according to the average session length used (60 or 174 minutes). These average session lengths were used in other studies (e.g., [7], [16]), and the 174 minutes session length is representative of Gnutella [28]. We ran one DHT peer per node, which allowed all nodes to use the default IP port. We ran each experiment three times and reported the average results. For all systems we computed only the traffic for routing table maintenance and peer failures detection (e.g., heartbeats), as the other overheads involved, such as lookup traffic and routing table transference, should be the same for all single-hop systems.

Each experiment had two phases, where the first phase was used to grow the system up to the target size and the second phase was used for the measurements. In the first phase each system started with just eight peers, and we joined one peer per second until reaching the target size, which resulted in a very steep growth rate (the systems doubled in size in just eight seconds, with an eight fold growth in less than one minute). In order to assure a really thorough test of the joining and maintenance algorithms, the systems did not perform lookups during the first phase, as the lookups would help peers to detect stale entries in their routing tables, and so they could mitigate problems associated with the joining and maintenance algorithms. The second phase always lasted for 30 minutes, while each peer performed one random lookup per second.

In both phases of all experiments, the systems were churned according to Equation III.1 and the chosen  $S_{avg}$  (60 or 174 minutes), and all peer leaves were randomly distributed along the ID ring. Half of the peer leaves were forced with a SIGKILL signal, which does neither allow the leaving peer to warn its neighbors nor to flush any event buffered, in a way that we should assume  $k=0.5$  for Equation IV.4. In order to maintain the system size, any leaving peer rejoined the system in one minute with the same IP and ID, which also allowed us to evaluate both systems in a scenario with concurrent joins and leaves.

### B. Comparative Experiments

In this section, we will present our measurements comparing D1HT and 1h-Calot, with special attention to the network maintenance overheads, but the first results to report are in relation to CPU use and fraction of the routing failures.

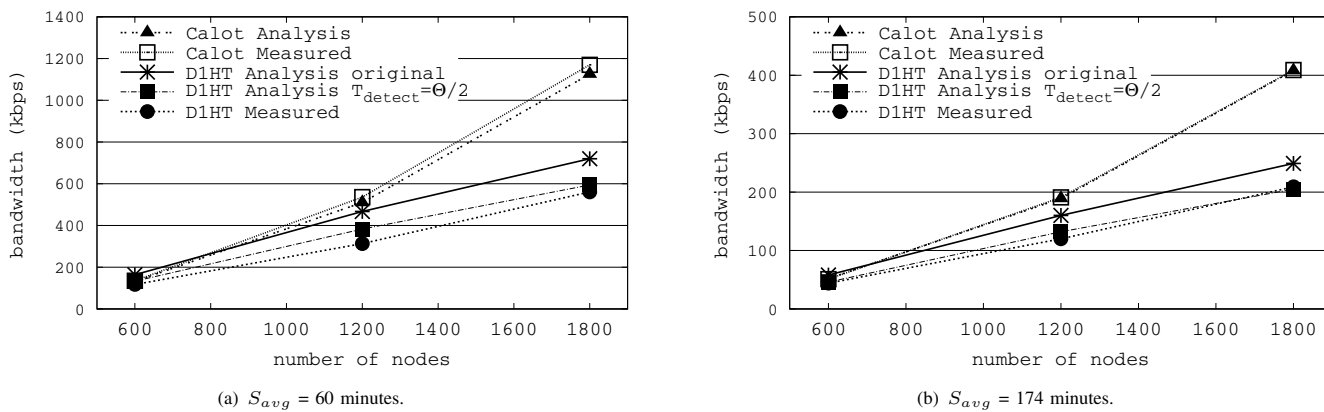


Fig. 2. Analytical and experimental outgoing maintenance bandwidth demands for DIHT and 1h-Calot.

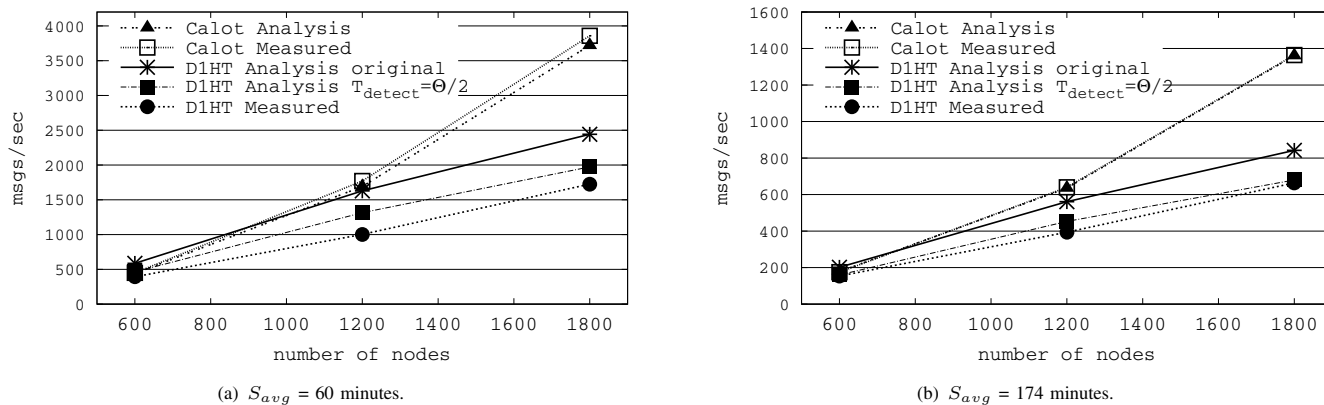


Fig. 3. Number of messages sent for DIHT and 1h-Calot as measured in our experiments and predicted by their analyses.

Both systems were able to solve more than 99% of the lookups with average CPU use below 0.02% in both phases of all experiments. This CPU use is negligible, and it already incorporates the cycles used by the joining mechanism as well as to perform and solve the lookups. Those are very important results, especially if we consider that all experiments were performed with the nodes under normal seismic processing production, confirming that DIHT and 1h-Calot are able to provide low latency access to information even when used in heavy loaded data center production environments.

Figures 2(a) and 2(b) show the sum of the outgoing bandwidth requirements of all peers for each of the systems studied for different churn rates  $r$  (as given by Equation III.1 as a function of  $S_{avg}$ ). We show both the measured (obtained with our experiments) and the analytical (calculated with the equations presented in Sections IV and V) requirements. The measured results included the overheads caused by message retransmissions, which were below 2% in all experiments. For DIHT we plotted the analytical results according to its original analysis parameterization (as conservatively suggested in [16]), as well as with the alternative parameterization provided by Equation IV.4 ( $T_{detect}=\Theta/2$ ). The figure shows that the 1h-Calot analytical results are very close to those measured in practice (the differences were basically due to the message

retransmissions), while the original DIHT analysis showed to be very conservative (the DIHT analytical predictions were on average 25% above the results measured). So, our experiments indicated that both analyses can be used to predict those systems behaviors, but the original DIHT analysis should lead to conservative predictions. We investigated this issue further and, as already discussed in Section IV-A, among several conservative assumptions, the original DIHT analysis assumed that the events would take *on average*  $2\Theta$  seconds to be detected, while Equation IV.4 reflects a more realistic assumption. To illustrate this conservative effect, Figure 2 also plots the DIHT analytical results according to Equation IV.4 with  $k=0.5$  (since in our experiments half of the peer leaves were forced with a SIGKILL signal), which resulted in  $T_{detect}=\Theta/2$ . We can see from the figure that the revised parameterization provided results much closer to those measured in practice, while still being slightly conservative.

The measured results plotted in Figure 2 also show that DIHT had less bandwidth requirements in all cases studied, confirming in practice that this system can provide a more lightweight DHT implementation than 1h-Calot.

Figure 3 presents the rate of messages sent for both systems, showing again that the 1h-Calot analysis is precise and the original DIHT analytical results are conservative, but we can

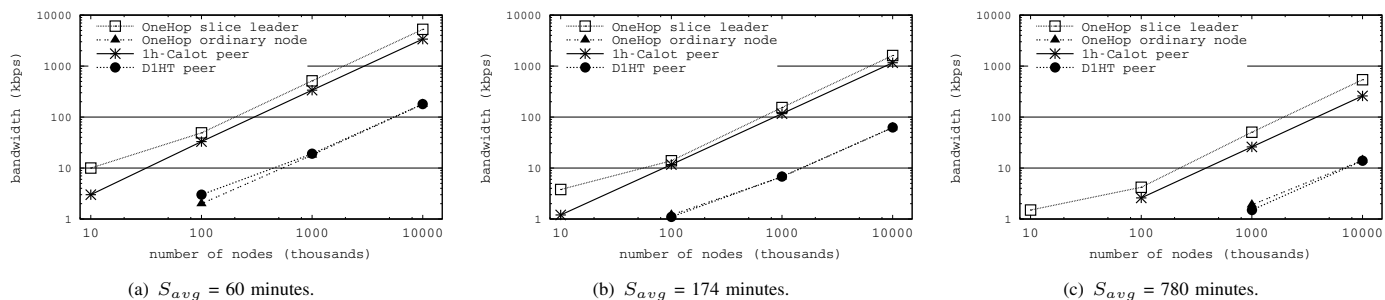


Fig. 4. Log-log plots showing the outgoing maintenance bandwidth demands for all systems studied with different average session lengths and system sizes varying from  $10^4$  to  $10^7$ .

see that our revised D1HT analysis based on Equation IV.4 is more precise while still being slightly conservative. The figure shows that D1HT always sends fewer messages than 1h-Calot, which is a consequence of its ability to group several events in the same message. From Figures 2 and 3 we can see that for both systems the bandwidth demands are highly related to the number of messages sent.

### VIII. ANALYTICAL RESULTS

Once our experiments validated the 1h-Calot and D1HT analyses, and since the OneHop analysis had already been validated [7], in this section we used analyses to compare those three systems with network sizes of up to 10 million nodes. As discussed in Section II, this comparison should also be valid for the 1HS DHT [25]. Please note that in a previous work [16] we have already provided an extended D1HT analysis, studying the duration of the  $\Theta$  intervals and the variation of the D1HT bandwidth demands for different values of  $f$ , churn rates and system sizes, and so here we focus on comparing D1HT, 1h-Calot and OneHop.

The results presented in this section have special value since they are based on analyses validated in practice, and allowed us to estimate the systems behavior with up to 10 million peers, which would be infeasible to obtain even through simulations. In fact, so far most DHT evaluations based on real implementations used a hundred *physical* nodes at most (e.g. [6]–[8], [23], [24], [32]), while the DHT simulations have been restricted to 20K nodes (e.g. [6]–[8], [11], [12], [15], [27], [29], [32]), with a few exceptions (e.g. [3]).

#### A. Methodology

The results presented in this section are based on the analyses discussed in Sections IV, V and VI. All D1HT results were obtained without the Quarantine mechanism introduced in [16]. The OneHop results always considered the optimal topological parameters and did not consider the failure of slice and unit leaders. For OneHop and D1HT we used  $f=1\%$ . The D1HT results were obtained with the use of Equation IV.4, but we conservatively assumed that all peer leaves are ungraceful (i.e.,  $k=1$ ).

The sizes of messages used in our analyses were derived from the formats shown in Figure 1, which are realistic since they were used in our D1HT and 1h-Calot implementations.

For D1HT we used 0.25 sec for the average message delay  $\delta_{avg}$ , which is conservative in relation to the results presented in [28]. In contrast, the OneHop and 1h-Calot results do not consider message delays. In the same way as we did in our experiments, our analytical results compute only the traffic for routing table maintenance, assume that the events and lookup targets are randomly distributed along the ID ring, and the event rates were obtained according to Equation III.1.

We varied the system sizes from  $10^4$  to  $10^7$ , which include sizes that are representative of ISP (e.g. [2]) and HPC (e.g. [19]) sites, as well as sizes that are coherent with huge P2P Internet applications such as BitTorrent and Gnutella. We studied systems with average sessions of 60, 174 and 780 minutes, where the 174 and 780 minutes session lengths were measured in Gnutella [28] and BitTorrent [1] studies, while the 60 minutes length tries to evaluate the systems under more severe scenarios. Besides being representative of widely deployed P2P applications, this range of session lengths is more comprehensive than those used in most DHT evaluations (e.g. [7], [11]–[13], [15]).

#### B. Comparative Analysis

In this section, we will compare the analytical bandwidth demands of D1HT and 1h-Calot peers against those of the best (ordinary nodes) and worst (slice leaders) OneHop cases for different churn rates and system sizes, as shown in Figure 4. Each log-log sub-figure plots the bandwidth overheads for a specific average session length, and system sizes varying from  $10^4$  to  $10^7$ . The requirements for a D1HT peer in systems with  $n=10^5$  and average sessions of 60, 174 and 780 min are respectively 3.1 kbps, 1.1 kbps, and 0.3 kbps, growing to 20 kbps, 6.8 kbps, and 1.5 kbps for  $n=10^6$ . Considering that back in 2004 the average peer download speeds of popular P2P systems like BitTorrent were already around 240kbps [21], those results show that the D1HT maintenance overheads should be acceptable even for systems with one million domestic nodes with behavior similar to those of BitTorrent peers.

We can see from Figure 4 that the OneHop hierarchical approach imposes high levels of load imbalance among slice leaders and ordinary nodes. Besides, a D1HT peer typically has one order of magnitude smaller maintenance requirements than OneHop slice leaders, while attaining similar overheads

when compared to ordinary nodes, even though the OneHop results assume the best topological parameters and do not consider message delays and the failure of slice and unit leaders. When comparing 1h-Calot to DIHT, we can see that DIHT has much smaller overheads for all cases studied, as the 1h-Calot maintenance demands were at least 200% and typically one order of magnitude higher.

## IX. CONCLUSION

In this work, we evaluated the impact of low-latency DHTs on performance-sensitive environments, such as HPC datacenters. To this end, we implemented two single-hop DHTs (DIHT and 1h-Calot) and performed the most representative experimental DHT evaluation to date, running both DHTs in up to 1,800 physical nodes in a heavy-loaded HPC datacenter. These experiments validated the analyses of both systems and confirmed that DIHT consistently achieved the lowest overheads. More interesting, these experiments showed that both DHTs are able to solve more than 99% of the lookups with low latency and very little CPU overhead, even when running on high loaded nodes.

We also performed an analytical comparison of DIHT and 1h-Calot against the very first single-hop DHT, the OneHop system, with system sizes of up to 10 million peers. The results showed again that DIHT consistently had the lowest maintenance bandwidth requirements, achieving overhead reductions of up to one order of magnitude in relation to both OneHop and 1h-Calot, in most cases. Our results also showed that DIHT is able to support vast distributed environments with dynamics similar to those of widely deployed P2P applications, while using reasonable maintenance bandwidth demands.

As a consequence of our experimental and analytical results, we may conclude that DIHT consistently had the lowest overheads among the single-hop DHTs studied, and that it can potentially be used for locating information in a multitude of environments, ranging from HPC datacenters to huge P2P applications deployed over the Internet. This ability to support such a wide range of environments may allow DIHT to be used as an inexpensive and scalable commodity software substrate for distributed applications. As one step in that direction, we made our DIHT source code available for free download [5].

## ACKNOWLEDGMENTS

We would like to thank Petrobras for providing access to the clusters used in the experiments, as well as for authorizing the public dissemination of the results.

## REFERENCES

- [1] N. Andrade, M. Mowbray, A. Lima, G. Wagner, and M. Ripeanu, "Influences on cooperation in BitTorrent communities," in *Proc. of the 3rd SIGCOMM Workshop on Economics of P2P Systems*, Aug 2005.
- [2] A. Barroso, J. Dean, and U. Holzle, "Web search for a planet: The Google cluster architecture," *IEEE Micro*, vol. 23, no. 2, 2003.
- [3] I. Baumgart, B. Heep, and S. Krause, "OverSim: A flexible overlay network simulation framework," in *Proc. of the IEEE Global Internet Symposium*, May 2007.
- [4] E. Brewer, "Lessons from giant-scale services," *IEEE Internet Computing*, vol. 5, no. 4, 2001.
- [5] DIHT source code available from <http://www.lcp.coppe.ufrj.br/DIHT>, Nov 2009.
- [6] F. Dabek, M. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-Area cooperative storage with CFS," in *SOSP*, 2001.
- [7] P. Fonseca, R. Rodrigues, A. Gupta, and B. Liskov, "Full information lookups for peer-to-peer overlays," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 9, Sep 2009.
- [8] R. Huebsch, J. Hellerstein, N. Boon, T. Loo, S. Shenker, and I. Stoica, "Querying the Internet with PIER," in *Proc. of VLDB*, Sep 2003.
- [9] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy, "Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web," in *Proc. of the Symposium on Theory of Computing*, May 1997.
- [10] B. Knutsson, H. Lu, W. Xu, and B. Hopkins, "Peer-to-Peer support for massively multiplayer games," in *Proc. of INFOCOM*, Mar 2004.
- [11] J. Li, J. Stribling, T. Gil, R. Morris, and F. Kaashoek, "Comparing the performance of distributed hash tables under churn," in *IPTPS*, 2004.
- [12] J. Li, J. Stribling, R. Morris, and M. Frans, "A performance vs. cost framework for evaluating DHT design tradeoffs," in *Proc. of INFOCOM*, Mar 2005.
- [13] J. Li, J. Stribling, R. Morris, and M. Kaashoek, "Bandwidth-efficient management of DHT routing tables," in *Proc. of NSDI*, May 2005.
- [14] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the XOR metric," in *Proc. of IPTPS*, Mar 2002.
- [15] A. Mizrak, Y. Cheng, V. Kumar, and S. Savage, "Structured Superpeers: Leveraging heterogeneity to provide constant-time lookup," in *Proc. of the 3rd Workshop on Internet Applications*, Jun 2003.
- [16] L. Monnerat and C. Amorim, "DIHT: A Distributed One Hop Hash Table," in *Proc. of IPDPS*, Apr 2006. [Online]. Available: <http://www.lcp.coppe.ufrj.br>
- [17] L. R. Monnerat and C. Amorim, "DIHT: A Distributed One Hop Hash Table," UFRJ, Tech. Rep. ES-676/05, May 2005.
- [18] NIST, "Secure Hash Standard (SHS)," *FIPS Publication 180-1*, Apr 1995.
- [19] J. Panetta, P. Souza, C. Cunha, F. Roxo, S. Sinedino, I. Pedrosa, A. Romanelli, L. Monnerat, L. Carneiro, and C. Albrecht, "Computational characteristics of production seismic migration and its performance on novel processor architectures," in *19th International Symposium on Computer Architecture and High Performance Computing*, Oct 2007.
- [20] D. Patterson, "Latency lags bandwidth," *Commun. ACM*, vol. 47, no. 10, pp. 71–75, 2004.
- [21] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips, "The Bittorrent P2P File-sharing System: Measurements and Analysis," in *Proc. of IPTPS*, Feb 2005.
- [22] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content Addressable Network," in *Proc. of SIGCOMM*, 2001.
- [23] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz, "Pond: The Oceanstore Prototype," in *Proc. of the USENIX Conference on File and Storage Technologies (FAST)*, Mar 2003.
- [24] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling churn in a DHT," in *Proc. of USENIX Conference*, Jun 2004.
- [25] J. Risson, A. Harwood, and T. Moors, "Stable high-capacity one-hop distributed hash tables," in *Proc. of ISCC*, Jun 2006.
- [26] A. Rowstron and P. Druschel, "Pastry: scalable, decentralized object location and routing for large-scale peer-to-peer systems," in *Proc. of Middleware*, Nov 2001.
- [27] —, "Storage Management and Caching in PAST, A Large-scale, Persistent Peer-to-peer Storage Utility," in *Proc. of SOSP*, Oct 2001.
- [28] S. Saroiu, P. Gummadi, and S. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," in *Proc. of Multimedia Computing and Networking (MMCN)*, Jan 2002.
- [29] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for Internet applications," *IEEE/ACM Transactions on Networking*, Feb 2003.
- [30] C. Tang, M. J. Buco, R. Chang, S. Dwarkadas, L. Luan, E. So, and C. Ward, "Low traffic overlay networks with large routing tables," in *Proc. of SIGMETRICS*, Jun 2005.
- [31] TOP500, "www.top500.org," Jan 2009.
- [32] B. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiatowicz, "Tapestry: A global-scale overlay for rapid service deployment," *JSAC*, Jan 2004.