D2HT: the best of both worlds, Integrating RPS and DHT

Marin Bertier, François Bonnet, Anne-Marie Kermarrec, Vincent Leroy, Sathya Peri, Michel Raynal

D2HT: the best of both worlds, Integrating RPS and DHT

Marin Bertier
IRISA/INSA Rennes
Rennes, France
Marin.Bertier@irisa.fr

François Bonnet
IRISA/Université de Rennes 1
Rennes, France
Francois.Bonnet@irisa.fr

Anne-Marie Kermarrec
INRIA Rennes - Bretagne Atlantique
Rennes, France
Anne-Marie.Kermarrec@inria.fr

Vincent Leroy
IRISA/INSA Rennes
Rennes, France
Vincent.Leroy@irisa.fr

Sathya Peri
Dep. of C.S., Memorial University
St. John's, Canada
Sathya@mun.ca

Michel Raynal
IRISA/Université de Rennes 1
Rennes, France
Michel.Raynal@irisa.fr

Abstract

Distributed Hash Tables (DHTs) and Random Peer Sampling (RPS) provide important and complementary services in the area of P2P overlay networks. DHTs achieve efficient lookup while RPS enables nodes to build and maintain connectivity in the presence of high churn. Clearly, many applications, *e.g.* in the area of search, would greatly benefit if both these services were available together at a reasonable cost.

This paper integrates a structured P2P overlay and a Random Peer Sampling service through gossip protocols. This system called *D2HT*, leverages the small-world nature of DHTs and relies on two cohabiting gossip protocols maintaining the close and long-range links respectively. The long links are chosen according to a harmonic distribution, following the Kleinberg small-world model. This approach exhibits several benefits: *(i)* The resulting DHT is highly dynamic and self-stabilizing, changes are tracked for free through the gossip nature of the protocol. This removes the need for complex, usually disjoint, and expensive join and repair procedures. Yet, it achieves reasonable routing performance with respect to standard DHTs; *(ii)* The resulting peer sampling service provides a biased sampling following a harmonic distribution: this improves the routing without jeopardizing the quality of the RPS. The set of long-range links which are a source of RPS can be used independently by others applications for free. They change continuously, achieving well-balanced routing across the nodes. We perform extensive simulations and compare the performances of *D2HT* with Cyclon, HRing, Symphony and Pastry to demonstrate the gains achieved by the approach proposed in this paper.

1 Introduction

Peer-to peer (P2P) overlay networks have proven efficient to support a large number of large-scale distributed applications for they combine scalability and high resilience to dynamics. Initially deployed in the context of file sharing systems, they have been since applied in many other contexts such as information dissemination, video streaming, video-on-demand, voice on IP, etc. Two main classes

emerged, structured and unstructured overlays, which differ on the structure of the resulting topology and the functionalities they provide.

Structured P2P overlays, also known as *distributed hash tables* (DHT) for they provide a natural support to support such a functionality, logically organize peers in a well-defined structured. We call such structured P2P overlays, DHT in the sequel. DHTs are of great importance in P2P systems as they offer an effective means of performing exhaustive and exact search in large-scale systems. Structured P2P overlay networks provide efficient lookups by building rigid architectures and precisely placing links between nodes across the network [1, 2, 3, 4].

Unstructured networks on the other hand achieve random graph-like topology, allowing more flexibility in the structure. In this context, gossip protocols have emerged as an efficient way to build such unstructured networks. More specifically the *Random Peer Sampling (RPS)* service is a basic building block in unstructured P2P systems to deploy and maintain connected overlays even in the presence of high churn. RPS can be employed in a wide variety of settings such as information dissemination [5, 6], load balancing [7] and overlay bootstrapping [8]. RPS is also a generic tool that acts as a source of randomness to guarantee both connectivity and convergence to weakly structured overlays like the ones built through peer clustering [9]. Such overlays provide an efficient support for keyword-based or range queries for example.

While P2P overlays are either of one kind or the other, they provide complementary functionalities. Therefore, many applications could greatly benefit if a DHT and an RPS were available concomitantly. This would enable an application to exploit the full potential of each without suffering from their limitations. For example, search applications could easily leverage both: the exact match interface of DHT is natural for exact search while keyword or range queries are easier to implement on top of an unstructured network. This was addressed first in [10] where an overlay providing both peers clustering and exact resources lookup through a DHT is designed. The clustering protocol, instead of relying on a dedicated RPS protocol to ensure convergence, uses the peers provided by the DHT. This approach relies on existing protocols and does not explore the possibility of designing protocols meant to be used in this context.

This paper proposes a simple protocol *Dynamic DHT (D2HT)* providing simultaneously both DHT and RPS functionalities by the means of two cohabiting gossip-based protocols. *D2HT* leverages the small-world nature of DHT on one hand and the fact that RPS can implement a small-world structure without sacrificing on its properties. A small world topology is characterized by the fact that each node is connected to its closest neighbors in the logical topology and maintains a set of additional long-range contact(s), also called shortcut(s). Greedy routing can achieve poly-logarithmic performance in such networks if the shortcuts are chosen according to a d-harmonic distribution in a d-dimensional mesh [11].

It turns out that on the one hand, each DHT node maintains its neighbors in a distribution close to the harmonic distribution. On the other hand, it has been shown that gossip protocols can achieve small-world topologies. Leveraging this commonality, *D2HT* maintains two sets of links with two gossip protocols.

The *Neighbors Peer Sampling (NPS)* maintains a set of closest short-range links and the *Kleinberg Peer Sampling (KPS)* selects long-range links. The set of long-range links is updated periodically and approximates the Kleinberg-like-small-world distribution [11]. The ever changing long-range links selected by *KPS* makes it an efficient RPS service. Both *NPS* and *KPS* can be deployed on the fly from any connected overlay.

Numerous DHT systems that route efficiently have been proposed so far. Yet many of these solutions [1, 2] use a segregated approach for building the DHT. They rely on distinct protocols for DHT initialization, regular churn and catastrophic failures recovery. Hence, they remain complex to implement and maintain. The solution presented in this paper adopts a more holistic approach. It uses a simple protocol based on gossip for handling at once all these situations. Starting from an arbitrary connected overlay, the DHT self-stabilizes to the expected links distribution. Protocols such as T-Chord [12] initialize the DHT using a gossip-based protocol. This approach allows building a DHT from an existing overlay instead of creating a new one: it is especially suitable when the DHT services are not always required, therefore allowing to build the DHT when needed only, thus removing the maintenance overhead the rest of the time. However, unlike our solution, once the DHT has been constructed, the traditional protocol (Chord in this example) handles fur-

ther maintenance. In addition, *KPS* incorporates built-in features that ensure an evenly balanced in-degree of the nodes without the need for an auxiliary load-balancing protocol. Combined with the dynamic of the links, it causes the routing load to be well balanced across all nodes even when the routing requests all originate from the same node. The simplicity of *D2HT* together with its ability to be deployed on the fly and the fact that the long-range links can be leveraged by other protocols make it a better candidate than other more complex DHTs to use as a generic P2P middleware.

Section 2 gives a brief description of structured overlay networks and gossip-based protocols. Section 3 describes the *D2HT* protocol. Section 4 presents the simulation results and compares *D2HT* against competitors namely DHTs and RPS protocols. Section 5 concludes this paper.

2 Background and related work

This section reviews the existing DHT protocols used in our simulations and provides information about gossip protocols.

2.1 DHT Systems

DHTs map keys to nodes that store the corresponding object in large-scale systems. In DHT systems, objects and nodes are uniformly assigned unique identifiers from a name space called keys and node IDs respectively. In one-dimensional systems, identifiers are ordered on an *identifier ring* with the identifiers increasing in the clockwise direction. Objects are assigned to nodes in a deterministic manner (for example in Pastry, an object is assigned to the node whose ID is the closest to its key). Usually, a DHT relies on each node having a set of pointers to other nodes which consists of both short and long-range links. The short-range links connect a node to its closest (in the identifier space) nodes while the long-range links point to nodes further away. This typically reflects a small-world topology [11].

A lookup operation consists in routing to the object key being searched for. The routing protocol ensures that the request reaches the node responsible for that key. A routing request is initially handled greedily by the long-range links while the short-range links terminate it, by delivering it to the destination. Many structured P2P systems route in $O(\log(n))$ steps where n is the size of the system. By connecting a node to its immediate neighbors on the ring, the short-range links ensure firstly that the overlay remains connected and secondly that lookup requests eventually reach their destination. The way the long-range links are constructed differs among DHT systems. We classify DHTs into two categories based on long-range link construction, namely *(i)* probabilistic systems *(ii)* deterministic systems.

Probabilistic Systems Symphony [13] and HRing [14] typically construct long-range links using probabilities based on harmonic distributions. In both systems, each node maintains two short-range links with its immediate neighbors. In Symphony, nodes get an identifier chosen uniformly at random from the interval $I = [0, 1)$. The k long distance links are chosen through picking random IDs based on Kleinberg's small-world network [11] probabilities. The in-degree of each node is bounded by $2k$. In HRing, the long-range links are distributed based on harmonic series similar to Symphony. The routing table of each node consists of the short links and $O(log(n))$ long-range links on average. HRing differs from other ring structured DHT systems by building long-range links based on node position space and not node ID space.

Deterministic Systems Pastry [2] is representative of deterministic systems. In Pastry, nodes are assigned a r digit number of base 2^b. The leaf set (short-range links) contains a fixed number of routing entries whose node IDs are the closest to the local node ID. The routing table consists of links to distant node IDs (*long links*) and therefore provides routing short-cuts. The Pastry routing tables have r rows and 2^b columns. The first $i-1$ digits of the node ID in the cell $[i, j]$ are the same as the local node ID and the i^{th} digit is j.

2.2 Gossip protocols

In gossip protocols, each node maintains a set of links to a small number of neighbors, constituting a partial view of the network. The resulting graph forms an unstructured overlay. The neighbor-set of a node is called its *view*. Nodes periodically gossip with other nodes to ex-

change data and/or membership information. Here we focus on membership gossip protocols. At each gossip cycle, each node selects a node from its view to gossip with (*selectPeer()*), exchanges a part of its view with it (*selectToSend()*) and comes up with a new view depending on the information it has received from the gossip exchange (*selectToKeep()*). Such protocols achieve different topologies depending on the implementation of these functions. They range from random graphs, extremely resilient to churn [15], to clustered overlays [16, 9].

Random Peer Sampling Cyclon [17] is a RPS protocol in which nodes swap random links (exchange a subset of their neighbors) in order to *shuffle* the overlay. This policy prevents the duplication of links and ensures that the in-degree of nodes remains balanced.

Clustering In T-Man [16] and Vicinity [9], nodes compute a distance between them based on a given property (geometrical position for instance). The gossip algorithm is used to communicate with nodes increasingly closer, share the list of neighbors, and keep the ones that are the closest in order to converge to a perfect view. Clustering protocols often lead to network partitioning, which is why they are used in parallel with a RPS protocol that ensures connectivity (with high probability) and provides the clustering protocol with extra information.

3 *Dynamic DHT (D2HT)*

This section gives the intuition of *D2HT*, before providing the details of the protocols that maintain *D2HT* namely *KPS* and *NPS*. Note that they both fit the gossip framework presented before and are simple.

DHT Structure As mentioned in the previous section, a DHT requires short-range links for routing correctness and long-range ones to achieve efficiency. Each of these two sets of nodes is maintained through a dedicated gossip protocol in *D2HT*. The long-range links are provided by *Kleinberg Peer Sampling (KPS)* protocol detailed in Section 3.1. It is derived from existing RPS protocols. *Neighbors Peer Sampling (NPS)* protocol, described in Section 3.2, is a clustering protocol that converges to the short-range links and maintains them. It uses *KPS* as a source of additional information to increase the convergence speed, similarly to many clustering protocols. The union of those two views constitutes the routing table of *D2HT*. Nodes use greedy routing for forwarding lookup requests and route using absolute distances. Each node is assigned a unique identifier at random and those identifiers are organized on a ring. The size of each view can be easily configured and can even be changed at runtime, without discarding any existing link. This opens the possibility of a self-configuring DHT that automatically reacts to the network conditions. The cost of the protocol is limited to the information transmitted during gossip and can also be tuned simply by changing the gossip period to match the network conditions and the expected level of dynamic.

Peer Descriptors Both *NPS* and *KPS* maintain a view which consists of descriptors of the node's neighbors. Each descriptor consists of an age and a neighbor's node ID. The age field is incremented every time a gossip takes place. It characterizes the freshness of the descriptor. Many gossip protocols such as Cyclon [17] and Vicinity [9] gossip with the peer corresponding to the descriptor with the highest age. If the chosen peer happens to be disconnected, then the descriptor is deleted and the peer with the second highest age is chosen and so on and so forth. The lower the age, the less likely is a peer to be disconnected as the peer's (descriptor) age is reset whenever the node gossips it. Biasing the peer selection towards the peer with the highest age nodes enables to rapidly get rid of disconnected nodes. Similarly to those protocols, *D2HT* picks the peer with the highest age to gossip with.

Node Join and Departures When a node joins, it contacts an existing node in the system. Starting from this node, the joining node starts filling the views of the two protocols (*NPS* and *KPS*) by gossiping. There is no specific procedure for a node departure. A node that has left simply stops gossiping and its descriptor is gracefully deleted from the system, as described in [15]. Therefore no specific mechanism is required for managing node joins and departures. In [4], the authors show that in most situations, this scheme is more efficient than the reactive recovery used in many DHTs. The ring topology of *D2HT*

provides many alternative routing paths, increasing the probability for a request to reach its destination in case of massive failure [18]. A particular feature of *D2HT* is that it can be deployed on demand from any existing connected overlay. In that case, the views are initialized to the known nodes and converge to the DHT after a few cycles. If the application already requires a RPS protocol, *KPS* can be used as an implementation, making the on the fly deployment of the DHT faster since the *KPS* view already has the appropriate node ID distribution.

3.1 Kleinberg Peer Sampling (*KPS*)

This section describes *KPS*, the RPS protocol used for selecting the long-range links of *D2HT*. It is based on [19] in which the neighbors in the local view are chosen in order to follow the Kleinberg distribution of links. This protocol is briefly summarized in Table 1: each node maintains a local view of ℓ neighbors and exchanges g of them upon gossip.

Hook	Action taken in *KPS*
selectPeer()	Increment the age of all descriptors in *KPS* view and select *Q*, the peer with the highest age.
selectToSend()	Store only $(\ell - g)$ elements in the *KPS* view according to harmonic probability function. Send the removed g elements.
selectToKeep()	Merge the remaining *KPS* view with the g entries received.

Table 1: Implementation details of *KPS* on a node P gossiping with Q

Hook	Action taken in *NPS*
selectPeer()	Increment the age of all descriptors in *NPS* view and select *Q*, the peer with the highest age.
selectToSend()	Select nodes closest to *Q*, s clockwise and s counter-clockwise, from *NPS* and *KPS* views.
selectToKeep()	Select nodes closest to *P*, s clockwise and s counter-clockwise, from the received nodes, *NPS* and *KPS* views.

Table 2: Implementation details of *NPS* on a node P gossiping with Q

Links selection In *KPS* each node chooses $\ell - g$ links to keep and the g remaining links are exchanged. In order to converge towards Kleinberg's distribution of links, the choice of the links to be kept in the local view is done using the probabilities of small-world networks (the harmonic distribution). More precisely, *D2HT* protocols uses a one-dimensional space for node ID and then in *KPS* a node A keeps a node B in its view according to a probability proportional to $\frac{1}{d(A,B)}$, d being the natural distance on the ID space.

Impact on clustering In Figure 10 of Section 4, the evaluation of *KPS* shows that the distribution of links is effectively closer to Kleinberg's distribution compared to the distribution of links obtained using Cyclon [17]. This results in an increased convergence speed of *NPS*. More generally, any clustering protocol built for a given parameter should benefit from a RPS protocol biased with this parameter: in our case the nodes in the *KPS* view of a node A are globally closer to A compared to a RPS view. Therefore these nodes provide more useful links to the clustering protocol.

3.2 Neighbors Peer Sampling (*NPS*)

The *NPS* protocol creates the short-range links of each node. Basically, it is a clustering protocol [16] in which each node gets connected to its $2s$ closest neighbors, s chosen clockwise and s counter-clockwise on the ring.

Convergence Process To discover the closest neighbors, the *NPS* protocol exploits the transitivity property. Each node looks into the neighbors set of its neighbors to discover the closest nodes. Since each node has neighbors on both the clockwise and counter-clockwise direction, a node will eventually converge to a perfect view of the closest neighbors. Yet, using short-range links only can make this process very slow. A node joins the DHT from a random node whose DHT ID can be far apart. To reach nodes with the closest ID faster, a node can also

look at random nodes away from its immediate neighborhood, typically using the *KPS* protocol, which is a source of randomness to provide far away nodes. Thus a node running the *NPS* protocol also uses nodes from its *KPS* view to select nodes for its *NPS* view.

Gossip Peer Selection In the description of the protocol (Table 2) a peer chooses the descriptor with the highest age. The way a node chooses peers from its view plays an important role in ensuring that its view converges. To speed up the convergence, nodes use additional information, the history. Each node remembers the last $2s$ nodes it gossiped with. When choosing a gossip target, peers that have not been gossiped with in the recent past have priority over the other peers. These peers are sorted using the DHT distance from the node and the peer with the smallest distance is chosen. This is a generic improvement that can be used in any clustering protocol. It relies on the observation that a peer that has not been contacted for gossip has a higher probability to have fresh relevant information than a peer which has already been communicated with in the recent past. Once the overlay is stabilized, the history contains all the elements of the *NPS* view and the timestamp is the only criterion for choosing the next gossip target. Section 4 shows the benefits of history.

4 *D2HT* evaluation

D2HT integrates a DHT and a RPS protocol, factorizing the maintenance cost at the price of a slight performance loss in routing. The objective of this evaluation is to assess the properties of *D2HT* through extensive simulations. The results are compared with system developing a DHT or a RPS alone.
Our results show that *D2HT (i)* provides a RPS with properties very close to those of dedicated ones; and *(ii)* achieves a DHT with properties close to those of a dedicated DHT with respect to routing and significantly improves the resilience to churn. This demonstrates the efficiency of *D2HT* to build an on-the-fly DHT without sacrificing the quality of the RPS while reducing the global maintenance cost.

Experimental Setup All the protocols were implemented using the *PeerSim* simulator [20], a JAVA based P2P simulator. In the setup, nodes initially contact a random entry point (among the nodes already in the system). Each protocol was executed in each cycle and the order in which the nodes were chosen by the simulator to execute was random.

4.1 *D2HT* versus RPS

The main difference between *KPS* and a classic RPS is that the provided sample is biased to achieve a harmonic distribution. Therefore, the expected difference is mainly in the clustering coefficient of the resulting graph. In order to assess the quality of *KPS* as a RPS service, we compare it to Cyclon [17], along the following metrics: graph properties, the ability to discover new nodes and the impact on dissemination. We created a 1,000-node graph using both Cyclon and *KPS* with a view size of 10, 5 neighbors are exchanged at each gossip cycle. We also measure various properties after 100 cycles during 1,000 cycles.

Graph properties It has been shown that the overlay constructed by a RPS protocol exhibit characteristics close to those of a random graph [15], providing a relevant support for fast dissemination of information and providing robust connectivity in the presence of churn. Simulations show that Cyclon has a clustering coefficient of 0.00923 and an average path length of 3.21, while *KPS* has values 0.0154 and 3.22 respectively for these properties. We observe that the graph created by *KPS* is slightly more clustered, but both values remain very low. Therefore *KPS* also provides a very good overlay for flooding and maintains the network well connected. This is illustrated in Figure 1. A 100,000-node network was flooded 100 times using Cyclon and *KPS*. We observe that the number of nodes reached at each cycle of the flooding is the same, which demonstrates how close the graph properties of the overlays are.

Node discovery A RPS protocol is traditionally used to discover new nodes and to provide a clustering protocol with them [9]. The discovery capabilities of a node are therefore crucial for the accuracy of the clustering pro-

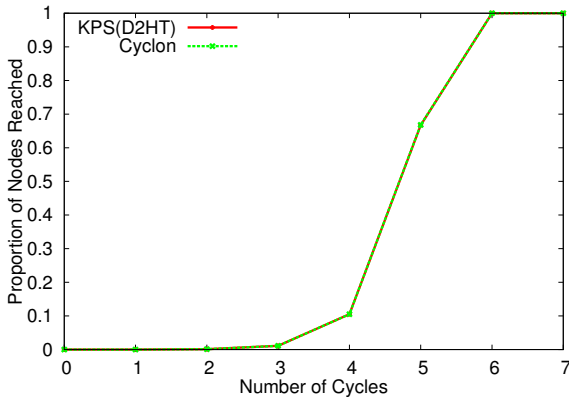

Figure 1: Flooding performance

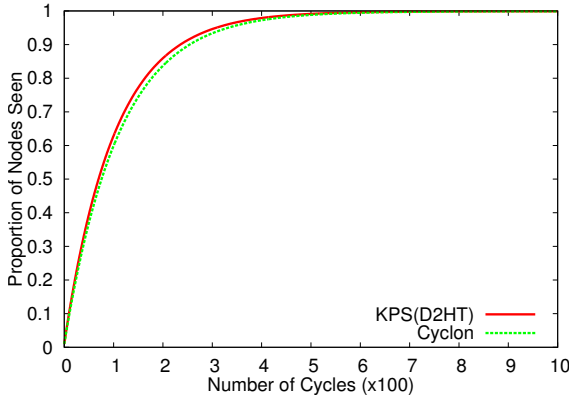

Figure 2: Rate of new nodes discovery

tocol. Figure 2 shows the number of different peers each node in the system can discover through the RPS protocol. We observe that the node discovery rate is very similar in Cyclon and *KPS*.

These results confirm that *KPS*, although biased to provide long links for *D2HT*, is an efficient RPS protocol. This means that *KPS* can be used instead of a standard RPS with no extra cost and without damaging the performance while providing possibility of deploying a DHT on demand at any time. Once the DHT is created, *KPS* can still be used as a RPS without any side-effect.

Application to clustering In Section 3.1, we mentioned the fact that any clustering protocol could benefit from a biased peer sampling service like *KPS*. To illustrate this situation, we study the case of a classic clustering problem introduced in [16]. Each node is assigned random coordinates on a torus and the clustering protocol should provide each node with its 10 closest neighbors, in terms of Euclidean distance. The clustering protocol uses gossip to converged to the desired view, and benefits from 10 random nodes provided by a dedicated peer sampling protocol. Figure 3 shows that *KPS* is much more efficient than a uniform RPS, namely Cyclon. In a 10,000-node network using *KPS*, 80% of the nodes converge to a perfect view in 29 cycles, while this takes 44 cycles using Cyclon.

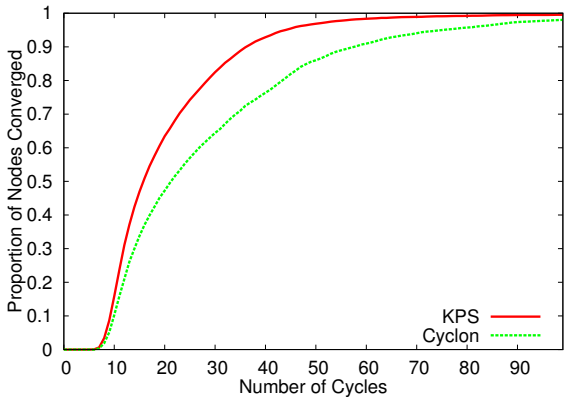

Figure 3: Clustering convergence speed

4.2 *D2HT* versus DHTs

We compare the routing performance of *D2HT* against HRing, Symphony and Pastry. In all the experiments, 16 short-range links are used, spread differently depending on the protocol: in Pastry and *D2HT*, which route using absolute distance, they divide in the 8 immediate closest on the ring on both sides of the node; in HRing and Symphony, 15 clockwise and 1 counter-clockwise links are maintained as they route in clockwise direction. As Pastry's routing table is more constrained, we use it as the baseline in several experiments to set the number of long links in the other protocols to make a fair comparison. This is due to the fact that the number of long links in Pastry is not precisely chosen and is only known during the simulation. Since HRing as presented in [14] is designed to have a fixed number of long-range links that depend only on the size of the network, we patched HRing to al-

low the protocol to configure extra links without changing the long link distribution. The number of long-range links varies from one experimentation to another. Most experiments consider only the case where the number of long-range links is exactly $log(n)$, but since it can be easily configured in our system, we also experimented different values and fixed that number to what seemed reasonable given the size of the network.

Convergence from arbitrary overlay *D2HT* relies on gossip protocols to create links so that a DHT functionality can be achieved. This protocol can be used to build a DHT on demand from any arbitrary configuration. To guarantee correct lookups a DHT must configure each node with the right set of short-range links.
To assess the self-recovery properties of *D2HT*, we consider two scenarios *(i)* where the DHT is created on the fly and *(ii)* where the DHT recovers from a catastrophic failure. In the first case, the initial overlay is random while in the second case, in an existing DHT, half the nodes in the network simultaneously crash. To stretch the failure scenario, it was broken by crashing groups of 8 consecutive nodes on the DHT ring, isolating as many groups of 8 nodes as possible. The nodes cannot rely on the short-range links to quickly repair the ring, they have to use the long-range links to converge back to a stable structure. Note hat this is one of the worst scenarios that can happen to a DHT. Most existing DHTs either do not support such extreme failure scenario, or would have to go through complicated mechanisms to recover from it. We ran an experiment on a 100,000-node network and with 16 short-range and 40 long-range links. Figure 4 shows the convergence patterns of the DHT starting from a random network with and without the history mechanism (described in Section 3.2) in both considered scenarios. We observe that the convergence is achieved very quickly, namely 32 cycles with history, considering that the DHT structure did not exist at all at cycle 0. Recovery from catastrophic failure is also very promising. Within 14 cycles the DHT was back to a fully operational mode. Note that we consider the recovery of the full set of short-range links to provide very high fault tolerance. In both cases, one can see the benefit of the history structure we added to the *NPS* protocol: it drastically speeds up the convergence.

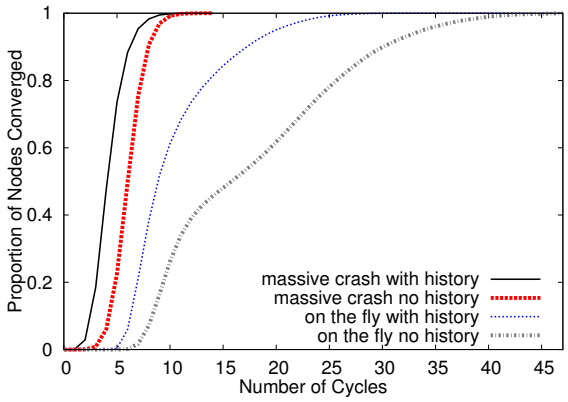

Figure 4: Short-range links convergence speed

These measurements show that the DHT of *D2HT* can be deployed on demand in a short amount of time. Moreover, it recovers almost instantly from extreme failures in which half of the network was crashed. This is achieved without the need for any specific protocol to handle failures, illustrating the self-recovering nature of *D2HT*.

Resilience to regular churn In the previous paragraph, we studied the convergence properties of *D2HT* in extreme cases. In most situations, a P2P overlay is only subject to regular churn caused by nodes joining or leaving the network. Like most ring-based DHTs, *D2HT* maintains the short-range links (*NPS*) very aggressively in order to protect the structure. However, contrary to other protocols, *D2HT* relies on gossip to maintain the long-range links. Therefore, disconnected nodes are actively detected upon gossip, even when there is no routing activity. We simulate a $10,000$-node network under different churn conditions. At each cycle, a fixed amount of nodes leaves the network and the same amount of new nodes joins. The network size remains constant but the nodes are regularly renewed at random. Figure 5 depicts the proportion of dead nodes in a *KPS* of size 40 under different churn rates. *KPS* maintains the proportion of dead links low, even when the churn is quite high (14% dead links for 1% churn). In a second experiment, we fix the churn rate to 0.5% and vary the size of *KPS* from 10 to 100 (Figure 6). As expected, the smaller *KPS*, the less likely are nodes to be disconnected. Indeed, they participate to gos-

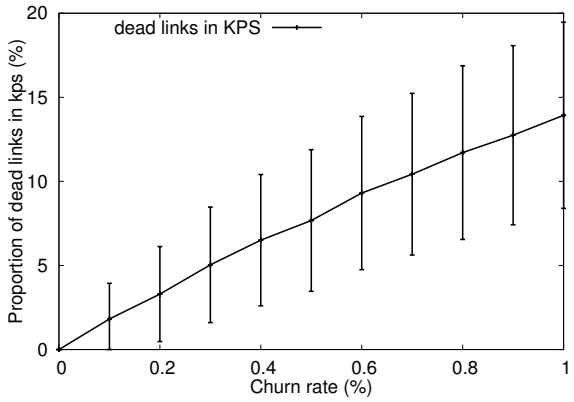

Figure 5: Dead nodes in KPS wrt churn rate (view size: 40)

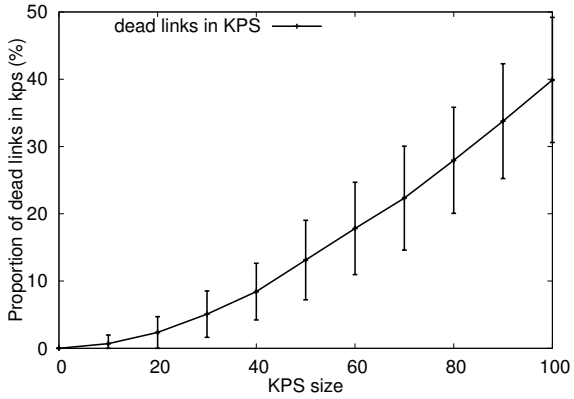

Figure 6: Dead nodes in KPS wrt view size (churn: 0.5%)

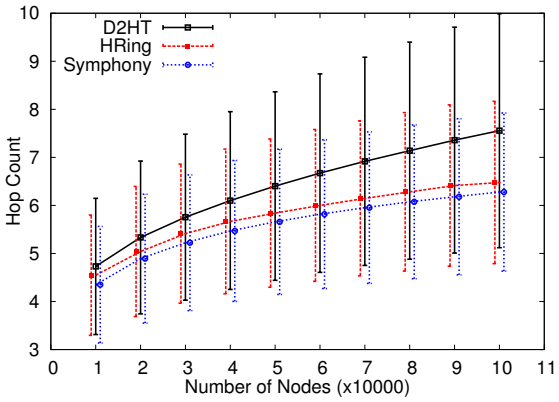

Figure 7: Routing perf / network size

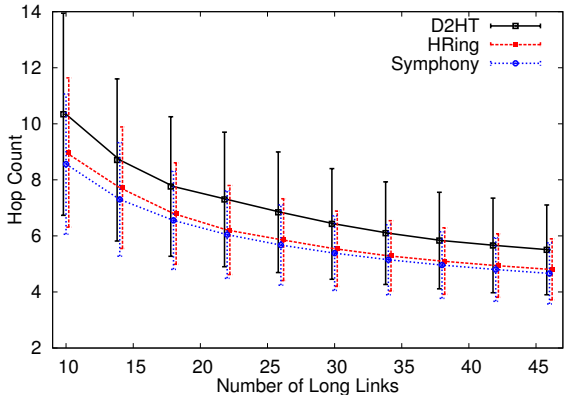

Figure 8: Routing perf / long-range links

sip more often and hence are more actively monitored. *D2HT* maintains the long-range links through gossip and benefits from an active detection of disconnected nodes. Unlike reactive DHT protocols, the average liveness of long-range links remains stable even when the routing activity is irregular, making the quality of service more predictable.

Routing performance We compare the routing performance of *D2HT*, HRing and Symphony. In all the experiments, for each measurement, 1,000,000 routing requests were sent from random nodes to random destinations. On each figure, we show the average performance as well as the standard deviation. Figure 7 shows how well their hop-count scales with the number of nodes. The number of long links was fixed to 20. Figure 8 shows the change in the hop-count as the number of long-range links increases from 10 to 46. Obviously, the more long links, the better the routing performance. Finally, we run an experiment adding Pastry to the performance comparison shown in Figure 9. For this experiment the number of long links is based on the one observed in Pastry and we change the number of nodes accordingly.

When compared to a structured DHT, the probabilistic DHTs (i.e, *D2HT*, HRing and Symphony) clearly show their limitations. With the same number of long-range links as Pastry they are not able to organize them effi-

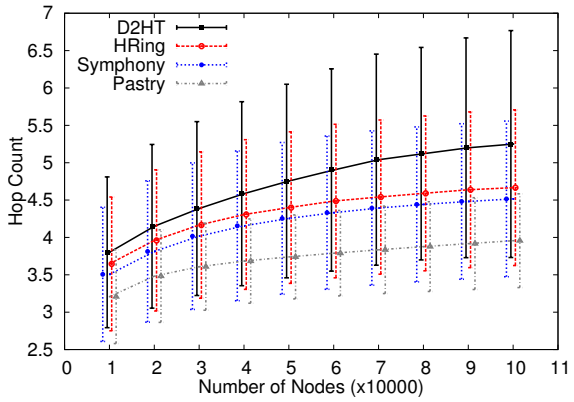

Figure 9: Routing perf / Pastry

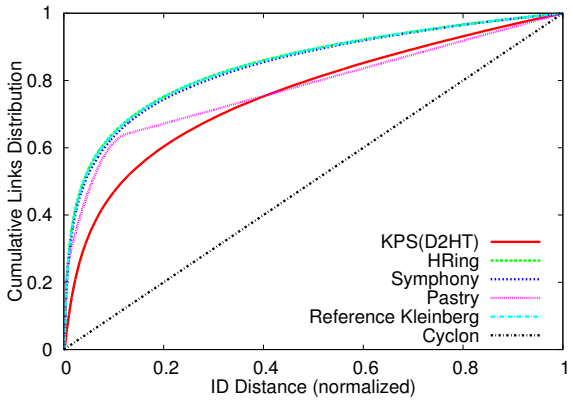

Figure 10: Long links distribution

ciently enough to match its performance. To give a better understanding of this, we plotted the average long-range links distribution of each DHT as depicted in Figure 10. We observe that Symphony and HRing reach a distribution very close to the Kleinberg distribution. *KPS* provides a small-world like distribution, but not as close as the others, which explains the slightly slower routing. Pastry does not have a fully small-world distribution. This is due to the routing table structure. Links are taken uniformly on a large part of the DHT ring, which explains the straight line from 0.1. Following this line of reasoning, one may think that Pastry routes slower than the other DHTs, but this is not the case. Pastry is a very structured overlay, meaning that all the nodes will have a long-range links distribution very close to the average. Since the other DHTs we consider are probabilistic, the standard deviation of this distribution is much higher, causing performance losses.

We observe from these simulations that the performance of *D2HT* is reasonable and scales well with the size of the system. The routing curves for *D2HT* have a logarithmic shape. The number of hops increases only by 1.6 when the number of nodes is multiplied by 10. Increasing the number of long links gives better routing performances. Like *D2HT* both Symphony and HRing produce a probabilistic small-world links distribution. But our protocol keeps a part of randomness and dynamism in swapping the nodes. As a consequence, its distribution is not as good as these protocols. One solution to overcome this drawback is to add extra long-range links. Figure 8 shows that with just 1.5 more links as the others, our protocol can route as fast as the others. *D2HT* embeds a high level of dynamic. While this makes it more difficult to obtain a perfect long links distribution, it enables to quickly recover from failures and spreads more evenly the routing load as we will observe further.

Load balancing DHT nodes are in charge of forwarding routing requests from other users, this represents the routing overhead on each node. In *D2HT*, routing takes place in two steps. First the long-range links are used to get closer to the destination, then the short-range links are used to terminate the routing. Since the short-range links are imposed by the DHT ring and do not have a significant effect on the routing performance, we focus on the long-range links. If we consider a system where each node requests for data lookup at the same rate, the load on each node directly depends on its in-degree (i.e. the number of nodes holding a pointer to a given node). HRing and Symphony provide a load balance mechanism which requires keeping track of the incoming long links. But this information does not come for free, each node must keep track of the other nodes that selected it as a long-range link. It must check that they are still alive in order to maintain an up-to-date in-degree counter. Furthermore, before adding a node as a long-range link, a node must ask its permission. All these operations make the protocol more complex and are costly. They are generic and could be adapted to any P2P overlay.

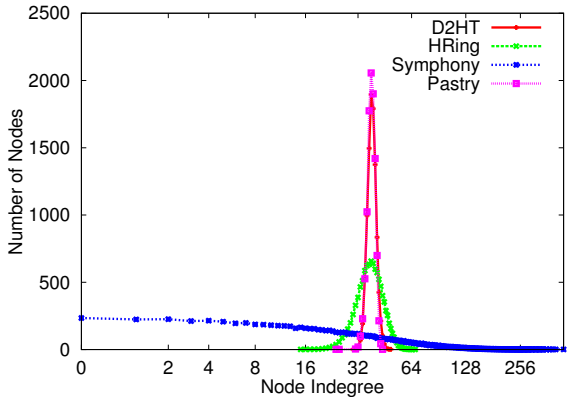

Figure 11: In-degree under uniform ID dist.

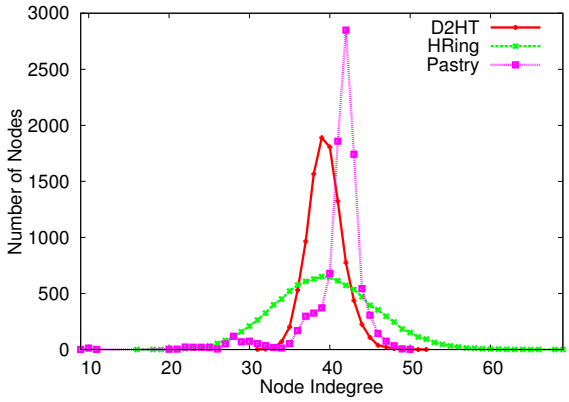

Figure 12: In-degree under gaussian ID dist.

Therefore we deactivated them in order to make a fair comparison with Pastry and *D2HT*. Our objective here is to evaluate the intrinsic load distribution pattern of the core protocol, without the help of any corrective measures, as such measures could be applied to any protocol.

Figure 11 shows the distribution of the in-degrees when the DHT IDs are chosen uniformly at random. We observe that the load in Symphony is fairly unbalanced. Many nodes have an in-degree of 0 while a few of them have more than 400 incoming long links. In HRing, it follows a Gaussian distribution, between 16 and 68 and centered on 40. Pastry and *D2HT* achieve much better performance where the distribution interval is smaller. Next, we modify the ID distribution to Gaussian and plot it Figure 12. Since Symphony in-degree distribution is not well balanced, we disregard it.

We observe that the in-degree distributions of HRing and *D2HT* are similar to the in-degree distribution with a uniform ID distribution. This shows that these protocols automatically produce a fair distribution, regardless of the ID distribution. In Pastry, the distribution remains acceptable but is largely affected by the way the IDs are chosen.

We also studied the effect of an unbalanced activity on the load of the nodes. This paper does not address the *hotspot* problem, where all nodes want to route to the same (popular) destination. A common solution to this problem is achieved by replication which have the same performance results for the DHTs that we consider. Other solutions imply different routing policies (like in [21]), but are out of the scope of this paper. We consider a different case that we call the *hotsource* problem where a very active node routes to random destinations. In Figure 13, we routed 1000 messages over 100 cycles from a unique random node to random destinations. The load distribution is really uneven in Symphony, a node even had to forward 683 messages. HRing is slightly better, with a maximum of 214, followed by Pastry with 75. Yet, *D2HT* clearly outperforms them all in this scenario, the maximum load being 11. This reflects the impact of the high dynamics of the data structures in *D2HT*, where each node renews part of its long links in each cycle so that the network load distribution is almost not affected by unfair node activity.

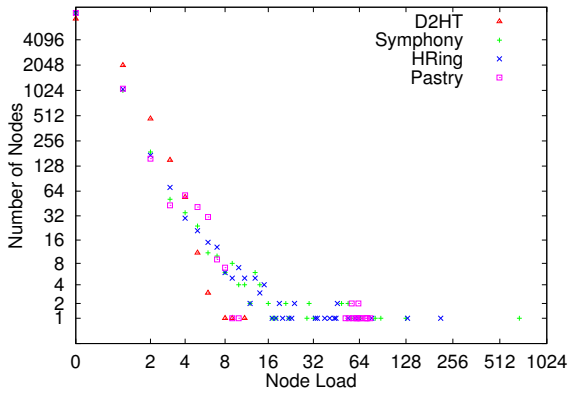

Figure 13: Routing load

5 Conclusion

This paper has presented the design of *D2HT*, that builds and maintains concomitantly a DHT and a RPS using gossip protocols. *D2HT* relies on two gossip protocols creating respectively a set of short-range links (the links to the closest peers in the ID space) and a set of long-range links, serving both the purpose of a DHT routing table and a peer sampling service. The long links are chosen according to a harmonic distribution, à la Kleinberg, to achieve efficient routing.

D2HT provides the following benefits. Firstly, it provides a RPS service together with a DHT, factorizing the maintenance cost. Secondly, the resulting DHT is dynamic in nature and offers better routing load balance and uniform in-degrees for the nodes in the systems. This also provides a proactive recovery in the presence of failures, avoiding the need for complex and expensive join and maintenance procedures, making *D2HT* a simple self-recovering protocol. Evaluations show that *D2HT* outperforms the three DHT candidates. It was compared against namely Pastry, HRing and Symphony with respect to failures and load balance, at the price of a slightly less efficient routing. Finally, peers can use the *KPS* links as a RPS service for clustering protocols and information dissemination. It scales well with the number of nodes in the system as well as adapt to high degrees of churn such as catastrophic failures.

An interesting future area of research is the configuration on the fly of the DHT (e.g. dynamically changing view size of *NPS* and *KPS*).

References

[1] 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 Service for Internet Applications," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 17–32, 2003.

[2] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems," in *Proc. of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, 2001, pp. 329–350.

[3] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," in *Proc. of the 6th International Workshop on Peer-to-Peer Systems (IPTPS'07)*, 2002, pp. 53–65.

[4] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling churn in a dht," in *Proc. of the USENIX Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 2004, pp. 10–23.

[5] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database maintenance," in *Proc. of the 6th ACM Symposium on Principles of Distributing Computing (PODC'87)*, 1987, pp. 1–12.

[6] P. T. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulié, "Epidemic information dissemination in distributed systems," *IEEE Computer*, vol. 37, no. 5, pp. 60–67, 2004.

[7] M. Jelasity, A. Montresor, and O. Babaoglu, "A modular paradigm for building self-organizing peer-to-peer applications," in *Proc. of Engineering Self-Organising Systems*. Springer, 2004, pp. 265–282.

[8] ——, "The bootstrapping service," in *Proc. of the 26th IEEE International ConferenceWorkshops*

on Distributed Computing Systems (IDCSW'06). IEEE Computer Society, 2006, pp. 11–16.

[9] S. Voulgaris and M. van Steen, "Epidemic-style management of semantic overlays for content-based searching," in *Proc. of Euro-Par 2005 Parallel Processing*, 2005, pp. 1143–1152.

[10] B. Maniymaran, M. Bertier, and A.-M. Kermarrec, "Build one, get one free: Leveraging the coexistence of multiple p2p overlay networks," in *Proc. of the 27th International Conference on Distributed Computing Systems (ICDCS'07)*, 2007, pp. 33–40.

[11] J. Kleinberg, "The small-world phenomenon: An algorithmic perspective," in *Proc. of the 32nd ACM Symposium on Theory of Computing (STOC'00)*, 2000, pp. 163–170.

[12] A. Montresor, M. Jelasity, and O. Babaoglu, "Chord on demand," in *Proc. of the 5th IEEE International Conference on Peer-to-Peer Computing (P2P'05)*, 2005, pp. 87–94.

[13] G. S. Manku, M. Bawa, and P. Raghavan, "Symphony: Distributed hashing in a small world," in *Proc. of the 4th USENIX Symposium on Internet Technologies and Systems*, 2003, pp. 127–140.

[14] X. Chen, X. Sun, and E. Yao, "Hring: A structured p2p overlay based on harmonic series," *IEEE Transactions on Parallel Distributed Systems*, vol. 19, no. 2, pp. 145–158, 2008.

[15] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen, "Gossip-based peer sampling," *ACM Transactions on Computer Systems*, vol. 25, no. 3, pp. 8–43, 2007.

[16] M. Jelasity, A. Montresor, and O. Babaoglu, "T-man: Gossip-based fast overlay topology construction," *Comput. Netw.*, vol. 53, no. 13, pp. 2321–2339, 2009.

[17] S. Voulgaris, D. Gavidia, and M. van Steen, "Cyclon: Inexpensive membership management for unstructured p2p overlays," *Journal of Network and Systems Management*, vol. 13, no. 2, pp. 197–217, June 2005.

[18] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica, "The impact of DHT routing geometry on resilience and proximity," in *Proc. of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM New York, NY, USA, 2003, pp. 381–394.

[19] F. Bonnet, A.-M. Kermarrec, and M. Raynal, "Small-world networks: From theoretical bounds to practical systems," in *Proc. of the 11st International Conference on Principles of Distributed Systems (OPODIS'07)*, 2007, pp. 372–385.

[20] PeerSim, "http://peersim.sourceforge.net/."

[21] S. Serbu, P. Kropf, and P. Felber, "Improving the dependability of prefix-based routing in dhts," in *Proc. of OTM Confederated International Conferences (OTM'07)*. Springer, 2007, pp. 206–225.

Glossary

D2HT	Dynamic Distributed Hash Table
DHT	Distributed Hash Table
KPS	Kleinberg Peer Sampling
NPS	Neighbors Peer Sampling
RPS	Random Peer Sampling