Cyclone: a Novel Design Schema for Hierarchical DHTs

Marc Sánchez Artigas, Pedro García López,
Jordi Pujol Ahulló
Universitat Rovira i Virgili
Tarragona, Spain
{marc.sanchez, pedro.garcia}@urv.net

Antonio F. Gómez Skarmeta
Universidad de Murcia
Murcia, Spain
skarmeta@dif.um.es

Abstract

Recent research efforts have improved the existing flat Distributed Hash Tables to accommodate hierarchical structure. Nevertheless, many problems still remain to be solved regarding scalability issues, autonomous systems, connection degree, and network proximity. In this paper, we present a new Hierarchical DHT called Cyclone that aims to solve the aforementioned issues with a near-optimal architecture. Cyclone provides optimal logarithmic routing hops without establishing unnecessary connection links to other nodes. Our approach follows a horizontal and uniform leaf-based approach that considerably reduces the overall number of links per node. Furthermore, Cyclone also offers a disjoint multipath routing scheme that benefits from network proximity and thus creates a more robust overlay infrastructure.

1. Introduction

In the peer to peer (P2P) arena, Distributed Hash Tables have received considerable attention because of their scalability properties and their optimal routing infrastructure. In DHTs, nodes are arranged in structured overlays, so that queries for keys or resources can efficiently be routed to the appropriate node in the network. Thanks to these properties, DHTs have been proposed as a substrate for a plethora of services and distributed applications like replicated data stores, application level multicast, publish /subscribe systems or even to implement the Domain Name System (DNS).

Nevertheless, most of the existing DHT systems follow a pure flat design that precludes most of the interesting advantages that can offer the hierarchical architectures. This flat design provides a uniform distribution of (equal) peers and resources that assures scalability and load balancing in the overlay network. We however believe that the hierarchical approach used in many distributed systems can be successfully combined with the flat DHT systems to create innovative environments.

In fact, in the last years, several research works have introduced the benefits of hierarchical designs to the flat world of DHTs achieving very promising architectures. We classify the existing system in two groups regarding their inter-connection architecture: vertical approaches and horizontal or leaf-based approaches.

Most of the existing hierarchical DHT systems follow the vertical approach because of their simplicity and straightforward implementation. In a vertical approach, every layer or leaf in the hierarchical tree is a self-contained DHT overlay network.

Such architecture has many advantages: local traffic does not affect other layers, network proximity, efficient caching and many others.

The major problem of these vertical architectures is mainly related to scalability, network size and overall number of connections in the network. DHTs are mainly designed for very big networks, and the creation of several DHTs assigned to subdomains or layers can represent a burden for the problem that they aim to solve.

We however are mainly interested in the Hierarchical DHTs (HDHT) that follow a horizontal or leaf-based approach. In a horizontal overlay, all the leaf overlay networks are connected using a single DHT that contains the conceptual hierarchy and optimizes the routing in the whole network. As we will explain in this paper, this approach considerably reduces the number of connections that builds the hierarchical infrastructure. Besides, in an ideal leaf-based network, every autonomous network could choose their local or internal overlay arrangement. For example, one network could be completely connected due to its small size, other one could connect using a Gnutella unstructured overlay, and other one could use

Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing (P2P'05)
0-7695-2376-5/05 $20.00 © 2005 IEEE

Kademlia or Chord[1] overlays. It is the horizontal glue the responsible to bring order to the isolated autonomous overlays.

At the time present, only Canon[5] can be considered a horizontal HDHT. Canon merges the leaf-networks to form a single overlay that routes efficiently without adding many extra links. Besides, Canon does not offer independent multipaths and it is suboptimal in connection degree, scalability and efficiency to our approach.

In this paper we present Cyclone, a horizontal HDHT that provides optimal logarithmic routing hops without establishing unnecessary connection links to other nodes. Our approach follows a uniform leaf-based approach that considerably reduces the overall number of links per node. Cyclone is based in a single and circular ID space for all nodes in the different leaf-networks of the tree. By using a uniform distribution of peers in leaf-networks, and a structured ID partitioning scheme, we can route efficiently to other leaf clusters using ID distances.

A key concept to understand Cyclone is that we take a b-bit circular identifier space $[0, 2^b)$ and assign each member a unique *ID* that consists of two parts: a PREFIX of $p–b$ bits and a SUFFIX of p bits, where p is a positive integer $p \leq b$. As we explain later, this technique let peers reuse optimally the connections and optimize the routing performance of our protocol.

The rest of the paper is organized as follows. In section 2, we discuss some related work in this setting. In section 3, we present a brief overview of the overall Cyclone ID assignment method. In section 4, we present Whirl, a cyclone version of Chord that supports hierarchical overlays. In section 5, we present an optimized version of Cyclone that outperforms Whirl in many aspects. In section 6, we present the major benefits accomplished by Cyclone, and in section 7 we justify our approach through simulation. Finally, we draw conclusions from our work and present future research using this approach.

2. Related Work

Many research works have outlined the advantages of applying hierarchical design to P2P overlay networks. For example, in [3] the authors present the advantages of combining DHT algorithms with a hierarchical design. They outline the advantages derived from using superpeers, network proximity, caching or replication in this kind of P2P networks.

The more straightforward solution to create hierarchical DHTs is what we call the vertical approach. In this model, every layer or edge is an independent DHT containing its own routing tables, and different layers are connected using gateway nodes. In [7], the authors present a hierarchical DHT structure with independent rings connected in a hierarchy by gateway nodes. These nodes use Scribe application multicast trees to receive queries from other networks. Using some ringId these gateways are capable of routing (through Scribe trees) in the overall network.

The main drawback that arises in the pure vertical approach is the disproportionate overhead that nodes must afford as a result of the maintenance of the routing tables at any tier.

Other approaches create a unique ID scheme in all the rings that considerably improves the performance in the hierarchy. Nevertheless, and as stated in HIERAS[6], “besides its highest layer finger table, each node in HIERAS create m-1 (m I s the hierarchy depth) other finger tables in lower layer P2P rings it belongs to”. Another interesting approach is CORAL [4], in which a performance improvement is achieved by using a unique global node ID and a XOR distance metric to route in any layer. Furthermore, CORAL includes a dynamic system of cluster overlays constructed taking benefit from network proximity information.

As stated before, all the aforementioned approaches, need multiple routing tables for each of the rings they are connected to. We believe that horizontal approaches can considerably reduce the number of connections while maintaining the overall performance.

Probably Canon is the more related research to our work. Canon aims to create unique DHT that merges the different networks that participate in the hierarchy. Canon also creates a conceptual hierarchy in a single overlay but it is restricted to merging equal systems (Chord-Chord, CAN-CAN, Symphony-Symphony). Nevertheless, Canon is not optimal in number of connections due to its lack of load balancing of clusters and nodes. Besides, this lack of load balance affects the performance in routing hops that can be obtained in the overlay. Finally, the maintenance protocol and scalability in Canon is more complicated and irregular that our approach in Cyclone.

To conclude, while other approaches have tried to create a unique ID with cluster or hierarchical prefix bits, our ID scheme employing suffixes for cluster identification provides good scaling properties like load balancing and routing performance.

3. Cyclone Overview

Cyclone blends the flat DHT design with the traditional concept of hierarchy. In fact, the *Cyclone* main goal is

to let the participants construct a *conceptual hierarchy* of autonomous clusters that route as efficiently as in flat DHT designs. That is, a conceptual tree in where leaves are autonomous clusters and internal nodes stand for reflecting the real-world organization. In particular, Cyclone attempts to minimize the transfer latency. Because the number of clusters will be typically orders of magnitude smaller than the total number of peers, queries will travel over fewer hops.

3.1 Cyclone ID assignment method

To effectively exploit likelihood of nodes, *Cyclone* paradigm takes a *b*-bit circular identifier space $\tilde{I}= [0,2^b)$ and assigns each member a unique *ID* that consists of two parts: a PREFIX of $p–b$ bits and a SUFFIX of p bits, where p is a positive integer $p \leq b$. The PREFIX, drawn uniformly at random, let participants have a unique intra-cluster identity whereas the SUFFIX let them know their residence cluster, that is, the *clusterID* of the cluster in which they reside. Figure 1 depicts a fragment of the tree of departments at URV University. The hexagons stand for the participant clusters in the *Cyclone* DHT. The black boxes refer to the internal nodes of the hierarchy and represent the subspaces. In fact, the design of *Cyclone* ensures that the participant peers are drawn uniformly from the circular *ID* space. In other words, the *p*-bit SUFFIXs are simply the translation of the full namespaces into binary strings.

Beginning at root node, each internal node has a *power of two* number of branches, that is, 2^{L_i} branches for some positive integer $L_i \leq p$. Such L_i-bit strings enable to label the branches and represent subsequently subspaces at each level. For example, hanging off root node there are a couple of subspaces, *DEIM* and *DEEA* labeled as 0 and 1, respectively. Loosely speaking, this means that the *clusterIDs* for subspace *DEIM* will always terminate in 0 whereas the clusters on *DEEA* will have 1 as the rightmost *clusterID* bit. Thus, by picking more bits leftwards at each level, *Cyclone* attains to construct the hierarchy and organize the participant peers on clusters. For example, the *clusterID* for namespace *EI.DEIM.URV* is 010 and thereby all members on that cluster will end 010. From a technical viewpoint, this means that two consecutive peer on that cluster will be at least at a distance of 8 on the circular b-bit *ID* space. Hence and in stark contrast to all hierarchical DHTs we know, this let the peers gather on clusters without renouncing to the properties of flat DHT design, such as load balancing, that rely on the uniform distribution of peers.

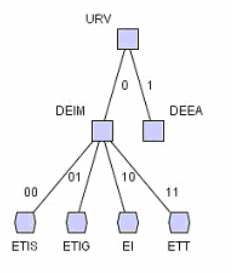

Figure 1. Fragment of the hierarchy at URV.

4. Whirl: Cyclone version of Chord.

In this section, we discuss a hierarchical version of *Chord.* The aim of that section is to demonstrate that *Cyclone* idea enables to build hierarchical DHT systems that route as efficiently as flat designs preserving the *total number* of connections per node. For the remaining of this paper we refer to the Cyclone version of Chord as *Whirl*.

4.1 Whirl Construction

Thanks to the fact that Cyclone form disjoint clusters, to construct hierarchical *Chord* consists in recursively *interconnecting* sibling clusters in each subspace to form larger Chord clusters from the bottom–most level to the root. In fact, *Cyclone* replaces the internal nodes of the hierarchy by new *Chord* DHTs that incorporate children clusters. Therefore, the challenge we face is how interconnect these children *Chord* DHTs to form larger rings. As we will further discuss, the standard *Chord* mechanisms will be enough to do it.

We first use an example to show how Cyclone combines two clusters to form a larger one. Figure 2 shows the two children clusters *E* and *O* for a two-tier hierarchical *Chord* of a *3*-bit identifier space [0, 8) that reserve p=1 SUFFIX bits for *clusterID*s. Consequently, the cluster *E* has the even participants and the cluster *O* the odd ones. For sake of simplicity, we will focus on the edges created by node 0 in ring *E*. Since *Cyclone* nodes on both *E* and *O* DHTs have *2*-bit PREFIXs, the identifier space for both clusters is [0, 4). Hence, the node 0 creates links to the nodes whose PREFIX equals or follows its PREFIX by at least a distance of 2^0=1 and 2^1=2 away. Recall that each *Chord* node maintains a link to the closest node that is at least 2^k away, for each $0 \leq k < b$, where b are number of bits for the b-bit Chord identifier space $[0, 2^b)$. Consequently, node 0 creates a pair of *intra*-cluster links to nodes 2 and 4 in ring *E.* Note that if we remove the SUFFIXs (*clusterID*s) from the peers global ID, we obtain a *Chord* network in which peers respond to 0, 1,

2 and 3 instead of 0, 2, 4 and 6 for ring *E* or 1, 3, 5 and 7 for ring *O*.

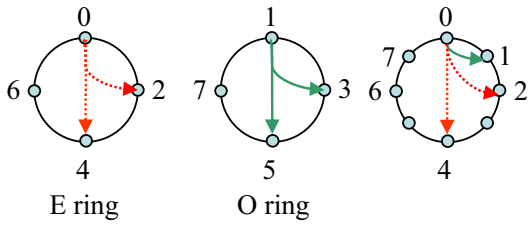

Figure 2.Whirl's resultant ring from the union of both E and O Chord DHTs. The dashed lines depict the Chord links for node 0 in E ring. The continuous lines plot the Chord links for node 1 in O ring. Note that node 0 solely must fix a new link to node 1 to properly route in the resultant ring.

On the other hand, the establishment of the *inter*-cluster links does not require any special construction mechanism either. The participants create the *inter*-cluster links by applying just the standard Chord rule for link creation upon the union of the nodes in children rings. This implies that nodes in both *E* and *O* rings must use the 3-bit global IDs as new identities to properly merge the clusters into root namespace. In other words, E and O members must extend their PREFIXs rightwards to include the bit on SUFFIX responsible of the namespace partition at second tier. Note that going up to next level l_i in *Cyclone* means to enlarge the PREFIX and reduce the SUFFIX by the number of bits used to represent the l_{i-1} level subspace. In fact, such property is what really enables to recursively form larger clusters without exceeding the degree of flat DHT design. Observe that the nodes at distances 2^0=1, 2^1=2 and 2^2=4 away from node 0 are the nodes 1 (in ring *O*), 2 and 4 (in ring *E)* respectively. Hence, node 0 requires solely a new link to route as efficiently as if it was in a 3-bit *Chord* flat DHT. The reason is that when we enlarge the PREFIXs *p* bits to the left, we are actually multiplying the current distances by a factor of 2^p. Therefore, the nodes at distances 2^0=1, 2^1=2 away in both *E* and *O* rings are likely to be also the nodes for the distances $2^0 2$=2 and $2^1 2$=4 in top-level overlay. Consequently, Cyclone let participants reuse the prior connections to route in the top-level DHTs without increasing the *total number* of connections per node.

From cluster's viewpoint, Cyclone is a collection of regular *N-gons* with corners uniformly distributed along the circular ID space. In particular, given a cluster *C* with a *p*-bit *clusterID* and a peer *m* within *C*, then *m*'s immediate successor is at least 2^p away in ID space. Thus, Cyclone increases the chances of reusing prior connections to nearby nodes in the network.

The above mechanism for interconnecting a pair of Chord rings can generalize to any number of rings. By applying bottom-up this algorithm, Cyclone can construct larger rings in such a manner that the degree and the expected number of hops between any pair of peers be the same as in the flat design. Therefore, we can state the next theorem:

Theorem 1. *The degree of any node in Whirl, with the IDs distributed uniformly into clusters is* $O(\log_2 N)$.

4.2 Lookup algorithm

Routing in the Whirl is similar to routing in the standard Chord. Messages are forwarded along those edges that diminish the distance by some power of two. Routing is clockwise and greedy, never overshooting the destination.

Our communication technique operates as follows. When a member requests a key *d,* it first initiates the query on its bottom-most level cluster to try to take advantage of the network locality. If RPC calls reach the destination *d*, the query halts without ever searching the higher-level clusters. If not, the query switches to the next higher-level cluster and routes to the closest predecessor of *d* at that level. This procedure repeats until the responsible for *d* is found.

Even if the query eventually switches to the root cluster, the total number of steps required is about the same as in the leaf clusters, as the queries continues from the point at which it left off in the previous cluster. In Figure 2, imagine that node 0 in ring *E* wants to contact node 7 in ring *O*. To do it, node 0 initially routes along *E* ring to the closest predecessor of 7, that is, node 6. Finally, node 6 switches to root cluster at the top-most level and completes the operation forwarding the query to node 7.

In general, our DHT protocol queries the nodes in bottom-most level, fast clusters before those in higher-levels, slower clusters. This both reduces the latency of lookups and increases the chances of returning values stored by nearby nodes. Because our technique maintains a uniform degree distribution and routes bottom-up through the hierarchy, we can state the following theorem:

Theorem 2. *In a Whirl network of N nodes, the expected number of hops is at most* $\frac{1}{2}\log_2 N$ *with high probability.*

Although Whirl architecture has several relevant advantages in comparison to the flat Chord DHT such

as fault isolation, in essence, it inherits the same dynamic maintenance problems of Chord. Sequences of consecutive node arrivals and departures could incur to reconnect peers at each level. On the other hand, Whirl characterizes because all the intra-cluster *paths* for a query *q* converge always to a common node *p* when *q* requires going outside. Node that *p* is, in fact, the closest predecessor of *q* within this cluster. Consequently, the Whirl can efficiently cache responses at the exiting nodes at each level of the hierarchy.

5. Cyclone Protocol

Regardless of the significant benefits of the last architecture, we appeal for a multipath infrastructure instead. Thus, each peer has multiple choices to route a packet or to adapt to the underlying network topology. As a result, Cyclone becomes a hybrid approach between a rigid partitioning and a topology-aware protocol. Because Cyclone forms disjoint clusters at each level, a natural methodology to afford *multipath routing* is to interconnect them in such a fashion that the participants can route through any neighbor cluster. Thus, Cyclone brings both the locality-aware and multipathing benefits together in order to produce a high-reliable system. Recall that the increase in the number of independent routes sharply decreases the probability of lookup failure.

In general, the aim of Cyclone is to construct a multipath hierarchical system on top of heterogeneous clusters (DHTs). To accomplish that goal, we must build a self-organized network that let nodes efficiently join or leave the system, maintains short network diameter and achieves fault tolerance. Hence, we've designed Cyclone network as follows.

5.1 Architecture

Let *L* denote the number of levels in the tree and let *m* be a node participant. Intuitively, *m* located at the bottom-most overlay is contained in an ascendant sequence of larger clusters until the root cluster. We denote these clusters by $C_{L-1}(m) \subset C_{L-2}(m) \subset ... \subset C_0(m) = \tilde{I} = [0, 2^b)$, where $C_{L-1}(m)$ denotes the *m*'s cluster. Recall that *Cyclone* infrastructure does not impose the overlay algorithm for C_{L-1} clusters. Let $S_k(m)$ be the siblings of the $C_k(m)$ at level *k*.

For each cluster $C_k(m)$ on *m*'s branch until root $C_0(m)$, node *m* maintains connections to the siblings $S_k(m)$ at each level except for the $s \in S_k(m) = C_{k-1}(m)$. Further, let $B_k(m)$ be the number of SUFFIX bits required for representing the namespace subdivision at level *k*. Then, the number of connections for node *m* at level *k* is $E_k(m) = (2^{B_k(m)} - 1)$. Therefore, the total number of connections for node *m* bounds to $|C_{L-1}(m)| + E_l(m)\ E_{l-1}(m)..E_0(m)$. For large values of $E_k(m)$, the *number* of connections can grow up to size H_k(m), where $H_k(m)$ is the maximum number of siblings connections for node *m* at level *k*. Then, the total number of links for node m bounds to equation $|C_{L-1}(m)| + \sum_{k=L-1}^{0} H_k(m)$.

Yet, it only remains to define the *Cyclone* link creation rule. The idea is to apply the XOR metric on *clusterIDs* to efficiently route between clusters. This implies to create at least *p*=|SUFFIX(m)| *inter*-cluster links in order to get any cluster in at most L hops, where L denotes the length of the *clusterID* for the bottom-most cluster of the tree. In fact, the H_k(m) at each level *k* must fulfill next inequality: $H_k(m) \geq B_k(m)$. Further, note that |SUFFIX(m)| denotes the level *L* in where *m* should reside if the hierarchy was binary tree.

The XOR metric distance between two IDs *x*, *y* is $d(x, y) = \sum_{i=0}^{b} |x_i - y_i| \cdot 2^i$. That is, the longest prefix matching techniques in base 2. Intuitively, the longer is the common prefix of *x* and *y,* the smaller is the distance between them. In fact, the XOR halves the distance between *x* and *y* by finding prefixes that share one bit more to *y* at each step. In our context, going up the hierarchy one level reduces the distance by a factor of $1/2^n$, where $n=B_k(m)$.

In order to support XOR metric, each peer *m* must contact at least one cluster at a distance $d \in [2^i, 2^{i+1})$ away, for $0 \leq i <$ |SUFFIX(m)|. In the levels that $B_k(m) \geq 2$, *m* maintains exactly $B_k(m)$ connections to $B_k(m)$ distinct clusters. In fact, Cyclone characterizes because nodes require no global information about the structure of the hierarchy. It suffices for each node *m* to know its own path in the hierarchy in order to compute the *B(m)* and *clusterID*s for all *m's* containers, these are, $C_{L-2}(m) \subset ... \subset C_0(m)$.

In addition, Cyclone promises to increase system security and reliability. One technique for doing so is to construct *d* edge-disjoint Hamilton cycles (HCs) – which connect all nodes and visit each node only once – at each level. To do it, Cyclone must interconnect *d* siblings at any tier. The parameter *d* is tight to the number of available connections for peers at each level, that is, H_k. Recall that all participants in the same subtree have identical degree because of Cyclone symmetry. There are no nodes with more network duties than the others. Let $SUCC_k(m)$ be the set of immediate successors for node *m* at the *m*'s sibling

clusters at level *k*. A straightforward method to get the *d* disjoint HC is to connect each peer *m* with the nodes in $SUCC_k(m)$ at any tier *k*. Graphically, this technique results in a rising sequence of bigger connected rings until root. Figure 3 sketches proof of Cyclone network topology.

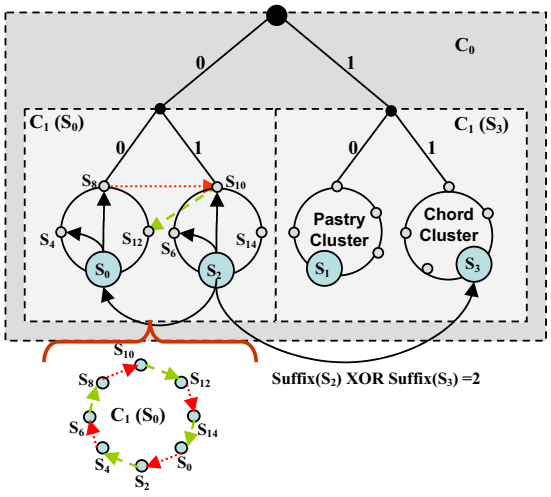

Figure 3.Cyclone topology. The above hierarchy has four clusters: *00*, *10*, *01* and *11*. Therefore, node S_2 in cluster *10* has SUFFIX=10 and PREFIX=0. The successor edges, plotted with dashed and pointed lines, form a new ring at the next upper layer C_1. The continuous lines depict the XOR connections between clusters. Note that peer S_2 is connected to clusters *00* (at XOR distance 1) and *11* (at XOR distance 2).

We must highlight the polyvalence of Cyclone links. In our approach, each link is simultaneously a valid edge to a remote cluster and a successor pointer to a neighbor node. The bivalence of Cyclone links let nodes reuse the pool of connections to route efficiently in terms of XOR metric, to route exclusively through the sibling edge-disjoint Hamilton Cycles or a combination of them.

5.2 Lookup algorithm

The lookup algorithm works as follows. Suppose a node *m* wants to look up a key *k* First, *m* routes within its inner cluster to take advantage of the network locality. If *m* finds *k* the query halts. Otherwise the query halts at the closest predecessor *q* for *k.* Then, node *q,* in order to get the cluster for *k*, takes the |SUFFIX(*q*)| rightmost bits of *k.* Let us refer to these bits as SUFFIX(*k*) . Next, *q* computes the distance in terms of *XOR* metric of SUFFIX(*k*) and all the *clusterIDs* in its routing table. Finally, *q* forwards the message to an arbitrary node y in candidate cluster. In general, *y* would be the immediate successor of *q* in that cluster. The process repeats until the query reaches a node *t* at destination cluster C_t that is, the closest one for *k* Finally, *t* uses C_t inner lookup service and contacts the manager for *k*. Observe that to let XOR metric work correctly, we must reverse the *clusterIDs* and the SUFFIX(*k*) for keys. The reason why we must reverse the *IDs* arises from the fact that Cyclone constructs them to the left, that is, taking the bits to identify the leaf clusters as the leftmost bits in *clusterID*s. Therefore, the most significant bits are just the bits on the right whereas the less significant ones are the bits on the left. Note that if we do not reverse the *clusterIDs*, the XOR metric will get us much further away from key *k* at each step.

5.3 Overlay Maintenance

When a node j joins the system, first asks an arbitrary node to get the closest predecessor of j, say j'. Thereafter, j' routes d special messages to all the nodes in SUCC(m). This d messages travel recursively by HCs through the successor pointers until they reach the immediate successor of j', say j''. These d special messages notify to immediate predecessors of j about the new arrival and record the immediate successors for j until j''. Then, j'' forwards these messages to j. Finally, j uses these messages to update the successor pointers and the XOR references since j knows the distribution of its container clusters until root. Thanks to the uniform distribution of Ĩ into leaf clusters, any node is surrounded by peers of nearly all clusters. Then, it suffices to recursively visit all nodes between a node m and its immediate successor succ(m) to discover the neighbor nodes at any tier.

6. Cyclone Benefits

Cyclone is perhaps the earliest hierarchical system that brings the following benefits together:

1. *Load Balancing.* It ensures that in a network with *N* nodes and *K* keys the average number of keys per nodes is tight around *K/N*. Unfortunately, the *non-uniform population* of clusters in most hierarchical systems unbalances the number of keys per node. Note that the peer with the longest arc is chosen $\Theta(n \log n)$ times more frequently than the peer with the shortest one. Therefore, some peers could be responsible of a disproportionate number of keys. Thanks to Cyclone distributes the nodes uniformly into the clusters, the number of keys is

approximately proportional to the number of nodes within the clusters.

2. *Fault Tolerance*. Cyclone strengthens the greedy routing schema by the construction of *d* edge-disjoint Hamilton cycles (HC) at each level. Since HC approach let messages progress through *d* independent routes, the probability of *lookup failure* reduces drastically. Now, a requester has more alternatives to bypass both overloaded and malevolent nodes. In fact, the number of lookup failures experiments an exponential reduction, proportional to *d*. Actually, we can stress that Cyclone offers a proper countermeasure to defend against routing attacks.

3. *Replication.* Typically, storage systems built on top of DHT employ replication as a mechanism to reinforce data availability. In our case, Cyclone stores the replicas for a key *k* at the immediate successors of its manager, say *m.* Since nodes in set *SUCC(m)* conduct to distinct clusters, the replicas fall off in distinct overlays, too. Clearly, this technique has a pair of advantages. First, it enables to contact replicas by exploiting *multipath-based routing*. Second, it let nearby clusters store a copy and avoid going outside for future requests. Thus, neither node failures nor hotspots prevent the system from recovering content.

4. *Heterogeneity*. Cyclone affords *heterogeneity* at both *peer* and *cluste*r levels. At *cluster* level, because nodes at bottom-most level are free to decide the DHT architecture for their inner cluster. At *peer* level, because designating the more powerful peers as *superpeers* to act as gateways between the clusters does not affect Cyclone behavior at all. Note that there is no explicit requirement in Cyclone architecture that forces peers to have the same network duties. In fact, the *superpeers* interconnect as in symmetric design. Unfortunately, the asymmetric design may diminish the ability of routing in the system. Now and so forth, requests will need to reach a *superpeer* before going outside the cluster. Consequently, we can claim that Cyclone makes possible to harness resources in a flexible cost-effective manner.

5. *Topology awareness*. The flexibility of XOR metric enables any node *m* to have multiple *candidates* at each level. In particular, the range of candidates gets wider as node *m* scales upwards to root $C_0(m)$. Recall that the i^{th} entry in routing table let a node choose a *candidate* away in $[2^i, 2^{i+1})$. Imagine that a node *m* resides in cluster 0000 i.e. it has |SUFFIX(*m*)| = 4 bits. Then, *m* must establish links to clusters at least a distance [1,2), [2,4), [4,8) and [8,16] away in the hierarchy. Except for the first interval –forces *m* to connect with its closest sibling (*cluster* 0001)–, *m* could select either cluster 0011 or 0010 for interval [1,2), clusters 0100, 0101, 0,110 or 0111 for interval [4,8] and so on. Thus, any node can choose the best candidate at each interval. In our case, the best candidate is the cluster decreases the latency by the greatest factor. Note that these links are not successors.

6. *Content caching*. If certain data is requested very often, a proper practice to reduce traffic load is to save it in a place much closer to the requester. Cyclone enables to efficiently exploit caching by storing popular data for a key *k* just in the closest predecessor of *k* within each cluster. This reduces the latency of lookups and increases the chances of returning values stored by nearby nodes.

7. Experimental results

In this section, we evaluate *Cyclone* using Chord as a self-organized substrate for leaf clusters. Further, we used PlanetSim[2] simulation framework to measure the *Cyclone*'s viability.

The experiments that we conducted were aimed at measuring empirically the *Cyclone*'s expected path length for different network sizes *N*=1024 to 8192 and hierarchies. In particular, we ran several simulations upon a pair of hierarchies. The first one was a complete tree; the second one was an unbalanced tree to let us analyze the Cyclone behavior in a more realistic scenario.

In general, the tests that we ran consisted in selecting a node uniformly at random in each cluster and then, used them to search the entire $\tilde{I} = [0, 2^b)$ space. Thus, we avoided to bias our tests to the distribution of clusters. Note that nodes in higher-level clusters have larger portions of $\tilde{I}$ space than the lower-level ones. Hence, nodes in such clusters have a greater probability to go outside their cluster and thereby, to increase the number of hops.

We evaluated Cyclone performance in number of hops. Because we know that latency is a more realistic metric, we believe that a more accurate analysis of Cyclone performance in terms of latency will be soon available in our website. Thereby, our tests simply recorded the total number of hops *T* for covering the $\tilde{I}$ space and divided this *T* by 2^b to report an estimation of the average path length.

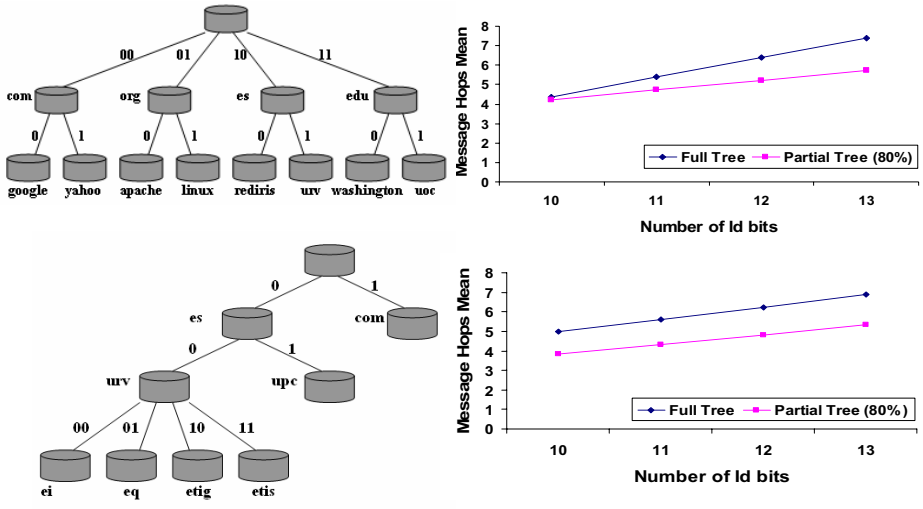

Figure 4.*Average path length* for Cyclone. (a) Results for a complete tree. (b) Results for an unbalanced hierarchy.

In order to provide a good estimation of Cyclone's maximum performance, we forced any peer *m* to have a pool of logical connections to exactly one node in each cluster. In fact, these nodes were the immediate successors of *m* in each cluster. Thus, the number of hops to get any key *k* was 1 + the number of hops to get the manager for *k* at the target cluster.

Figure 4 shows the average path length for the aforementioned hierarchies. Both (a) and (b) plot the average path length for a network of $N = |\tilde{I}|$ and $N = 0.8|\tilde{I}|$ nodes, where $|\tilde{I}|=2^b$ and *b* is the number of bits for IDs.

8. Conclusions

In this paper, we present a new Hierarchical DHT that achieves scalability and routing performance without the need for multiples routing tables.

Cyclone is, to our knowledge, the earliest hierarchical system that brings together load balancing, security (d edge-independent *multipaths*), replication, heterogeneity and topology-aware design.

Cyclone follows a horizontal design that connects leaf networks (irrespective of their underlying protocol) into a single self-organized overlay that contains the conceptual hierarchy with an optimal number of links.

Cyclone uses a unique ID in which the PREFIX identifies the node in a cluster and the SUFFIX the cluster in the hierarchy. This approach provides a load balanced architecture in which Cyclone represents the horizontal glue between the existing leaf-networks.

We believe that Cyclone is a step further in hierarchical DHT designs, and that many other research works can benefit from our generic architecture. In this line, we plan to work on Cyclone to support advanced caching and replication systems, to better adapt to network locality, and to construct optimal application multicast trees.

9. Acknowledgements

This work has been partially funded by the Spanish Ministry of Science and Technology through project TIC-2003-09288-C02-00.

10. References

[1] I..Stoica et al., "*Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications"*, Proc. ACM SIGComm 2001, pp. 149-160.

[2] P.García et. al., "PlanetSim: A New Overlay Network Simulation Framework". Proc.19th IEEE ASE 2004.

[3] Luis Garces-Erice, Ernst W. Biersack, Keith W. Ross, Pascal A. Felber and Guillaume Urvoy-Keller."Hierarchical P2P Systems". Proceedings of ACM/IFIP International Conference on Parallel and Distributed Computing (Euro-Par), 2003.

[4] Michael J. Freedman and David Mazières. "Sloppy Hashing and Self-Organizing Clusters". In Proc. 2nd Intl. Workshop on Peer-to-Peer Systems (IPTPS '03) Berkeley, CA, February 2003.

[5] Ganesan, Prasanna; Gummadi, Krishna; Garcia-Molina, Hector.Ganesan, Prasanna; Gummadi, Krishna; Garcia-Molina, Hector. "Canon in G Major: Designing DHTs with Hierarchical Structure", Proc. International Conference on Distributed Computing Systems (ICDCS) 2004

[6] Zhiyong Xu, Rui Min, Yiming Hu. "HIERAS: A DHT Based Hierarchical P2P Routing Algorithm". ICPP 2003

[7] Alan Mislove and Peter Druschel. "Providing Administrative Control and Autonomy in Structured Peer-to-Peer Overlays". Proc. IPTPS04, San Diego, CA, February 2004.