The p2pWeb model: a glue for the Web

Pedro García López, Marc Sánchez Artigas, Jordi Pujol Ahulló

Universitat Rovira i Virgili
Tarragona, Spain

{ pedro.garcia, marc.sanchez, jordi.pujol}@urv.cat

Abstract

We foresee promising cross-fertilizations of peer-to-peer and Web models in the next years. Although both models are already influencing each other, there is still a lack of seamless integration between them in order to achieve constructive synergies. In this paper, we present a p2pWeb infrastructure which interconnects a network of Web servers through a novel hierarchical Distributed Hash Table called H-Symphony. We show why the hierarchical approach outperforms the traditional flat designs in terms of content and path locality. Furthermore, our approach offers a seamless integration with Web technologies and creates a novel infrastructure for information distribution, data dissemination and query systems. Our proposed proof of concept application is framed in a known open source project called Moodle: A virtual campus web application used in thousands of Universities and Learning Institutions around the world. We have developed an extension of Moodle that aims to interconnect all the Moodle sites in the Internet using our p2pWeb infrastructure.

1. Introduction

In the last years, we noticed a slow but clear convergence between peer-to-peer (p2p) and Web environments. On the one hand, interoperable web standards and protocols are being massively used for data retrieval between edge peers. On the other hand, there are several approaches aimed to offer web services and applications on top of networks of edge peers.

Examples of such services are p2p web hosting (YouServ [1]) or p2p web caches (Coral [2]) among others. They have a key common denominator: They offer transparent access for Web clients to different web services and applications deployed in edge peer overlays.

Another possible approach can be to interconnect stable Web servers using peer-to-peer technologies in order to provide novel internet-wide services. We also believe that Distributed Hash Tables (DHTs) represent a suitable interconnection (glue) layer for several reasons:

- **Churn vs. Stability.** Web servers are stable peers which are not continuously entering and leaving the network. In fact, dynamicity is a strong problem for DHTs that are deployed in networks of volatile edge nodes. Therefore, our DHT abstraction can offer almost optimal efficiency without the burden of heavy self-adjusting stabilisation protocols.

- **NAT traversal and firewalls vs. Web accessibility.** Edge peers are usually located behind firewalls or NATs that preclude their full participation as routers in the network. Although there exist proxying techniques to overcome such a problem, they represent a burden for the whole overlay. We know that the utilization of HTTP permits to avoid such firewall problems: All nodes are full-fledged routers.

- **Connections vs. decoupling.** Most edge peer-to-peer networks use TCP or UDP as their communication protocol and HTTP only for
downloads. Every machine has a limit of open connections, since the bandwidth is consumed even in the lightest protocols. The utilization of HTTP for inter-webnode communication creates a decoupled infrastructure that saves traffic and open connections, thus creating a very lightweight infrastructure. In particular, a webpeer routing table is not a collection of open connections, but instead a list of URLs. This costless approach permits the creation of URL-caches to improve the performance of the overall infrastructure.

- **Peers vs. Super-peers.** The decentralized network of Web servers (glue DHT) is in fact an overlay of super-peers that can offer added services to the enormous mass of edge peers. Through the interconnection of these 2 levels, one can devise innovative distributed services derived from the interaction of both layers. It is a transition from a Web of isolated server islands to a true interconnected (glued) Web of peer servers.

Another key contribution of our approach is to base the p2pWeb infrastructure on a **Hierarchical DHT design.** We present a novel methodology that converts a flat overlay to a hierarchical one without adding noticeable overhead. In this work, we show why the hierarchical design outperforms the flat one in terms of path and content locality. These locality properties are suited for a considerable number of p2pWeb services such as query mechanisms, content distribution, aggregation and publish/subscribe.

Finally, we present a proof of concept application of our p2pWeb infrastructure in the context of an open source campus application called Moodle. We have developed an extension of Moodle that aims to interconnect all the Moodle sites in the Internet. We leverage the resources in this network made up of Web servers to offer efficient content distribution and search mechanisms in this community.

### 2. Related Work

In the last years, most peer-to-peer (p2p) systems and p2p applications have focused on leveraging the resources and bandwidth of edge desktop peers. In this context, several research projects try to exploit the combination of Web technologies and peer-to-peer to provide Web services and applications on top of edge peer networks.

For example, Coral’s [2] goal is to provide an efficient web cache designed to use the bandwidth of volunteers to reduce the load on websites. Coral follows a hierarchical DHT approach with clusters of the participating nodes grouped by latencies, thus improving the overall performance.

YouServ [1] offers web hosting and file sharing of static content in an edge p2 network by using replication and proxying mechanisms. Although it does not balance the access to content, it represents a simple alternative to publish content in the Web.

Although all these projects try to integrate peer-to-peer and Web technologies, they share a classical approach focused on edge peers and not as glue for Web server populations. Probably our approach is closer to projects like OpenHash [4] and the Internet Indirection Infrastructure [5]. They aim to provide Internet-wide services on top of a p2 network of stable super-peers. We however are more focused in the Web server setting.

### 3. P2PWeb Architecture

The key idea of our architecture is to interconnect the participating Web servers through a hierarchical DHT in which the communication mechanism is the HTTP protocol.

On the one hand, the use of HTTP is appropriate thanks to Web servers’ stability, NAT and firewall traversal. Hence, routing consists of HTTP redirects: There is no need for a heavy stabilization protocol.

On the other hand, the hierarchical design offers two key benefits to the p2pWeb model architecture: content locality and path locality. **Content locality** permits applications to store information in specific domains and **path locality** ensures that a routing path stays entirely within an organization if needed. We will prove the importance of these two services for our architecture.

In this section, we present our hierarchical DHT model that transforms a flat overlay to a hierarchical one without cost. We also describe the H-Symphony overlay used in our p2pWeb implementation. Then, we evaluate the key contributions of our approach by simulation.
3.1 Hierarchical DHT Methodology

In this section, we describe our methodology for constructing hierarchical DHTs. For convenience, we will refer to the underlying structured overlay as a “ring”, because the keyspace of protocols like Chord and Pastry form a ring. The great strength of our technique is that it configures a hierarchy built on top of a DHT, which reflects administrative and organizational domains, but with the same costs of running a flat DHT. It affords content/path locality. Rather than having a new routing table per domain, our technique properly distributes the links of each node into the different administrative domains. By a proper distribution, we mean that routing within a domain is as efficient as it will be in the flat design with the same membership. In fact, we seek to share a common overlay infrastructure, while constructing a hierarchy to work with. To applications, the multi-ring hierarchy appears as a single DHT that spans multiple organizations and networks, thus achieving the application-specific requirements of the p2pWeb model.

The system forms a tree of rings. Each ring has a globally unique clusterId, which is known to all members of that ring, indicating the ring position in the tree. To configure a hierarchy, our methodology recursively decomposes the keyspace into clusters. In practice, this means that for any two clusters in the tree. To maximize load balancing, DHTs typically use the same hash function \( h() : [0, 2^b - 1] \rightarrow [0, 2^b - 1] \) to map data to keys and nodes to nodeId, obscuring two basic concerns in DHT design: on the one hand, the distribution of nodes over the keyspace and on the other hand, the distribution of keys over the keyspace. Our technique considers the two concerns separately. While all levels share the entire keyspace \([0, 2^b - 1]\) for keys, the keyspace for nodeList is hierarchically decomposed into interleaved clusters. At level 0, a node \(n\) is assigned a nodeId from the whole keyspace \([0, 2^b - 1]\), which determines its position within the organization. At level \(i\), node \(n\) is assigned to the cluster whose clusterId is congruent to \(n's\) nodeId \(\text{mod} 2^i\). For sake of clarity, let us denote nodeId by its representation in base two (i.e. high order bits are on the left). Consequently, we view a nodeId as \(b\)-tuple \((x_1, x_2, ..., x_b)\), where each \(x_i \in \{0,1\}\). Now, consider that node \(n\) has as nodeId
the tuple \( \langle x_1, x_2, \ldots, x_b \rangle \). Then, it is not difficult to see that the cluster for node \( n \) at level \( i \) is given by the \( n \)'s nodeId suffix \( \langle x_{b-i}, \ldots, x_b \rangle \), and the prefix \( \langle x_1, \ldots, x_{b-(i+1)} \rangle \) is the \( n \)'s virtual nodeId for this cluster. As a result, routing within a ring is done according to the DHT routing algorithm, but considering only the nodeId prefixes. Note that there is exactly one cluster at each level \( i \) that encloses the \( n \)'s nodeId. In particular, Cyclone can be seen as a collection of regular polygons with corners uniformly distributed along the keyspace. Technically speaking, given a cluster with a \( b \)-bit clusterId and a peer \( p \) within this cluster, \( p \)'s closest neighbour will be at least at a distance \( 2^b \) away of him in the keyspace.

### 3.2 Hierarchical Symphony

In this section, we introduce H-Symphony: the Cyclone's hierarchical version of Symphony.

Briefly, Symphony is an adaptation of Kleinberg construction to model small world phenomenon over a dynamic set of nodes lying in a unitary circle \([0, 1)\). Each node in Symphony maintains: a) two short links with its immediate neighbours along the ring and b) \( k \geq 1 \) long distance links with the nodes lying clockwise distance \([x]\) away from itself, where \( x \) is drawn from probability distribution \( p_N(x) = 1/(x \ln N) \), for some \( x \in [1/N, 1] \). Routing is greedy and absolute: a node forwards a request for a key \( x \) along that out-going link (short or long) that minimizes the absolute distance to \( x \).

Our hierarchical design relies on our technique to configure the nodes into a hierarchy of clusters that reflect the p2pWeb administrative domains. To this end, all we are required to do is to transform the unit interval \([0, 1)\) into a circular \( b \)-bit keyspace \([0, 2^b - 1]\) and scale linearly function \( p_N(x) \) to adapt it to the new keyspace.

A node in H-Symphony maintains the following sets of links: a) successor and predecessor links for each ring and b) \( k \) long distance links for efficient routing within any ring. Each node distributes its \( k \) long links into its sequence of telescoping rings, so that routing within any ring is as efficient as it will be in the flat design with the same membership. To do so, we have applied the standard Symphony rule to create links in each ring. Note that our technique distributes the nodeIds uniformly within each ring. For further details, see Cyclone[8].

Routing in H-Symphony is identical to routing in Symphony. If a node wishes to route a message for a key \( x \), it forwards the message to its neighbour that is closest to \( x \), without overshooting the destination.

When there are multiple levels, routing proceeds as follows. When a node requests a key \( x \), it first routes on its leaf cluster in order to take advantage of path locality/caching. If the message reaches the destination \( x \), the query finishes. If not, the query reaches the closest node to \( x \), say \( p \), within this ring. Node \( p \) is then responsible for switching to the next upper cluster and for continuing routing in that ring. The preceding procedure is repeated until the node responsible for \( x \) is found.

Even if the request eventually reaches the global ring, the total number of routing hops incurred will be approximately the same as in Symphony.

### 3.3 Content and Path Locality

The main goal of our technique is to provide path and content locality as a basis for effective caching and hierarchical storage for the p2pWeb model. We now describe how these two properties are achieved in H-Symphony.
Path locality. The path from a peer to another never leaves the cluster in the hierarchy which contains both nodes. Clearly, this is true since nodes route in progressively larger rings, and the requests finish in the cluster that contains both the source and destination nodes. Strictly coupled to path locality, we have caching. For effective caching, one can easily exploit path locality provided by hierarchical routing. In particular, we cache the query answers in the exit points within each ring. In H-Symphony, when different nodes within a cluster $C$ route for the same key $x$ outside its cluster, all the routes to key $x$ exit $C$ through a common node $y$ which is the closest neighbour of $x$ within $C$. Therefore, we propose to cache the answers at each exit node encountered on the path to the query’s destination.

Content locality. Hierarchical design offers more alternatives for content storage than a flat DHT. For instance, if a node wishes to insert content, it can specify the ring in which the content will be stored. To this end, it can specify the clusterId for the content’s storage ring by replacing the last bits of key’s content suffix. Nodes can also specify the content’s access ring, a superset of the storage domain, to all of whose nodes the information is made accessible. In other words, nodes are allowed to store and retrieve content bundled to a real organization in the p2pWeb model.

4. Experimental Results

In this section, we evaluate our p2pWeb model, via simulation using Symphony and H-Symphony. All evaluations simulate networks with up to 16K nodes that are uniformly distributed at random over the keyspace. Each node has 2 short links and 7 long links. All values shown in the Fig. 2 are the result of the queries from all nodes to a fixed set of 400 keys uniformly distributed at random over the keyspace.

Fig 2.a) shows latency costs measured in number of hops. This figure shows that H-Symphony (up to 4 levels) has almost the same routing performance as Symphony.

We show a caching scenario in Fig 2.b) in order to show H-Symphony’s path locality skills. We have spread a total of 8 copies of an item throughout the network. In Symphony, the cached copies have been distributed uniformly at random. In H-Symphony, $2^l$ ($l$ is the hierarchy depth) copies of the eight possible has been cached at cluster exit points, and the rest uniformly at random. The figure clearly shows that H-Symphony outperforms Symphony in this test. As $l$ increases, the expected path length decreases.

Finally, H-Symphony’s caching algorithm storage cost is evaluated in Fig 2.c). More specifically, we wish to answer how many copies are required to reach a key within at most 10 hops from all nodes. In Symphony, copies are cached in route to the key. In H-Symphony, copies are cached at the cluster exit points as before. Again, this scenario points out that H-Symphony outperforms Symphony in number of copies.

5. Proof of Concept: MOODLE

Thanks to the fact that our design is hierarchical (i.e., path/content locality), our p2pWeb architecture offers tree and cluster-level services very efficiently. In particular, we have already developed on top of our PHP implementation of H-Symphony, available...
at http://planet.urv.es/p2pweb, a DHT interface with the additional cluster-level services:

```
put_cluster(cluster,k,v); get_cluster(cluster,k,v)
```

and the tree-level services (i.e. aggregation):

```
put_agg(cluster,k,v); get_agg(cluster,k,v)
```

As a proof of concept, we have recently decided to embrace Moodle which is a popular open source Virtual Campus Learning Tool deployed in around 17,599 Web servers around the world. Moodle.org actually provides two centralized services: Moodle Sites and Moodle Statistics. While Moodle Statistics stores information about available resources such as courses, users and quizzes in the Moodle network, Moodle Sites keeps track of the registered Moodle servers by geographical area.

Our MoodleShare standard extension to Moodle already offers both services on top of the p2pWeb infrastructure. Our novel contributions here are:

- **Updated Information.** If every node installs MoodleShare, our Sites and Statistics services will provide up-to-date information. Clearly, this is valuable information for monitoring the state of the community.

- **Geographical Information.** We group nodes into geographical clusters (continent, country, city, organization), thus enabling cluster-level access to information. A GoogleMap interface is provided to facilitate the access to clusters’ aggregated information (e.g., 300 servers with 10000 resources in Spain).

- **Moodle Search.** We accommodate a resource search mechanism for the whole community. Specifically, we provide cluster-level requests in order to constraint the scope of the search to a specific subset of geographical areas (e.g. Spain). With this service, a teacher can locate resources like courses in the whole network or in the specified clusters.

This project is in progress and it has received the attention of some leaders of the Moodle community like Martin Langhoff. They appreciate MoodleShare as a possible community hub that might complement the works about Moodle interoperability. We really believe that our infrastructure helps to improve the world-wide Moodle community offering information awareness and communication services.

## 6. Conclusions and Future Work

We have presented a novel p2pWeb infrastructure based on a hierarchical DHT which gives us key benefits like clustering, content and path locality or improved routing. These benefits seem to be quite adequate to define novel web reliable services such as Uniform Resource Name (URN) location, content distribution (i.e. web caching and replication), query services for clusters and publish/subscribe systems. In addition, our decoupled and lightweight approach will permit the creation of webrings of geographical, semantic or administrative clusters in a flexible way. We foresee a network of browser peers interacting in new ways, with the network of web superpeers as glue.

### Acknowledgements

This work is partly supported by European Commission under the FP6-2006-IST-034241 POPEYE project.

### References


