Semantic-laden Peer-to-Peer Service Directory

Tim Hsin-ting Hu, Sebastien Ardon, Aruna Sereviratne
University of New South Wales, Sydney, Australia
timhu@mobqos.ee.unsw.edu.au, ardon@unsw.edu.au, a.sereviratne@unsw.edu.au

Abstract

The most intuitive way to build a service directory application that allows for service entities to register or search for services on top of a structured peer-to-peer network is to build reverse indices at appropriate nodes on the network. However, this implies trust on the reliability and integrity of other nodes on the network, which may be too risky an assumption for businesses. This paper proposes a service directory that groups service entities of the same category together; this is achieved by dedicating part of the node identifiers to correspond to their service category semantic. Using Chord as the peer-to-peer substrate, this scheme logically divides the Chord circle into equidistant arcs; each arc is called an island. This scheme will result in the formation of islands of varying population, and thus changing the uniformly spread topology of the original Chord. Simulations are used to investigate the path length and message load of the changed topology. An additional routing scheme is also proposed and simulated to exploit the new topology to gain better path length.

1. Introduction

Peer-to-peer systems have proliferated in recent years, beginning with the original centralized system Napster [11], the subsequent rise of unstructured systems such as Gnutella [4], and the ever so popular structured variants such as Chord [6], CAN [16], Pastry [3] favored in academia.

Underlying all the different types of peer-to-peer systems is the notions of co-operation amongst the peers and harnessing the collective resources of all participants. Many applications have thus been developed over the years, including file sharing [4], distributed file storage [8], content distribution [9], email delivery [2], indirection services [5], and many more.

The scope of this paper will be centered on the context of a service directory application which allows Service Producers to list its Service Offerings and facilitate Consumers in finding the desired service. Service directories are usually centralized, ranging from the very traditional (and original) paper-based Yellow Pages, to modern Internet directories the likes of Yahoo and DMOZ [12]. The current emerging Web Services technology also employs a centralized system called Universal Description, Discovery and Integration (UDDI), which is a registry (with optional replication) containing information about businesses and the web services that they offer. However, like any other centralized model, UDDI suffers from the usual problems of limited scalability and single point of failure. It also implies that someone needs to operate and maintain this infrastructure.

Our paper takes the position that this infrastructure need not be centrally provisioned and maintained; the responsibility and cost, whether monetary or otherwise, should be shouldered by all participants who benefit from this infrastructure. This distribution of responsibilities will also mean better robustness for the system on the whole. The maturity of the current peer-to-peer technology actually makes this possible, and Chord is the chosen peer-to-peer substrate for our project because of its elegance and simplicity.

The rest of this paper is organized as follows: Section 2 contains a brief scope of the related work. In Section 3 the two service directory schemes are contrasted, and Section 4 outlines the ramifications of the semantic scheme. Section 5 presents the results of the simulations and Section 6 concludes this paper and presents some possible future work.

2. Related Work

The problem of finding service listings generalizes to a searching problem. It is clear from literature that unstructured systems such as Gnutella [2] implement searches by flooding queries to a subset of the peers.

Structured systems do not support search directly, but allows inverted indices to be reached via the native distributed hash table scheme. Thus searches in structured systems are very targeted and deterministic, and this is no exception with our scheme.

Schemes that operate in similar fashion include peer-to-peer DNS systems [10], keyword search for document storage and retrieval [11], and email services [12]. In terms of motivation and concept, [13] is the closest to our own; it organizes participating nodes into a hypercube structure and builds ontological links between the nodes. The main differences to the approach of this paper is the underlying peer-to-peer substrate, and the approach to search – node identifiers in hypercup correspond to a combination of ontology semantics, whilst our work at the moment only assumes single correspondence to a specific service category.

3. Building a peer-to-peer service directory

3.1. Traditional Approach

The traditional approach of building a service directory on top of peer-to-peer networks usually treats the peer-to-peer layer as a massive storage composed of all the participating nodes. Reverse indices are built at each node in the network by having Service Producers register their Service Descriptions at known location. These known locations correspond to nodes with unique identifiers; they are also termed keys in peer-to-peer terminology. As a simple example, a music encoder service may register its Service Description using the peer-to-peer primitive put( ) in the following fashion:

```
put( key , value )

key = hash_m( “music.encoder” )
value=“<description>wavtomp3</description>”
m  = number of bits in the identifier space
```

Other music encoder Service Producers may also register in the same known location by hashing “music.encoder” to produce the key, and register with their own Service Descriptions. The node responsible for this key will store a list of all the registered Service Descriptions, and return this list when other nodes query for this key looking for all the music encoders services available. The query is performed using the generic peer-to-peer primitive lookup( ), and a list of Service Descriptions are returned from the node responsible for this key. Please see Figure 1 for detailed workings of this scheme.

```
{value1, value2 … valueN} = lookup( key )
```

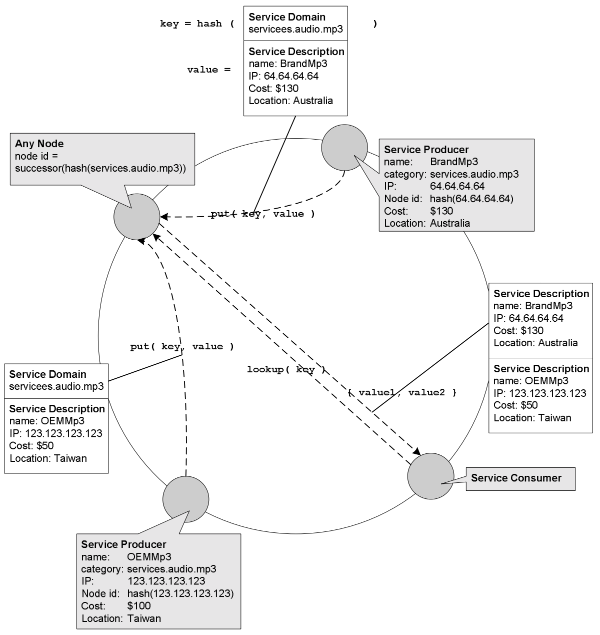


Figure 1. Traditional approach of building a peer-to-peer service directory

This approach is the most intuitive way of using a distributed hash table to store the data, analogous to the real life example of Yellow Pages. In real life, service entities submit their details to Yellow Pages (equivalent to the distributed storage) and then the details are “returned” when consumers look under a particular service category

However, the issue of trust severely limits this model when transferred to the peer-to-peer platform. Two issues at the forefront include firstly, the reliability of the node storing the Service Descriptions is unpredictable; a multinational company's Service Description could be hosted by a node run by a little shop in Siberia! Secondly, although possible, it is difficult to ensure that nodes hosting Service Descriptions will be resistant to bribes and collusions to favor one Service Producer over another.

3.2. Semantic Approach

To address the issue described above, we propose the scheme where the service semantic is inbuilt in the node identifiers. Instead of the key implicitly representing the semantic information, the node identifiers reflect the type of service that the Service Producer offers. So instead of Service Producers registering their Service Descriptions at another node, each Service Producer serves its own Service Descriptions, but “registers” by choosing an appropriate node identifier to participate in the network. Using the previous example, whereby the key represents the service entitie's category as “music.encoder”, now this information actually forms part of the node's identifier. This is achieved by allocating certain bits in the identifier for semantic information and using a hash that produces a result of the required number of bits. For example:

```
Id=[hashx(“music.encoder”)][hashy(“10.1.1.1”)]

x+y = number of bits in the identifier space
```

Diagrammatically this scheme will divide the Chord circle into equi-distant arcs, and the number of arcs will depend on the number of bits allocated to the semantic information. Each arc will contain nodes that belong to the same service category, collectively called an “island”. Logically the node identifier is split into two domains: the first part identifies the island that the node belongs to, and the second part identifies its position within the island. Conceptually this may be even more analogous to the real life example of the Yellow Pages: the placement of services in the same service category is listed together on the same pages, and the user finds the right pages to view the numerous offerings under the particular service category. The only difference is that Yellow Pages offers a list of all the offerings, whilst this scheme would require “scrolling” through the island for the full list. However, this difference can be easily addressed as an implementation issue.

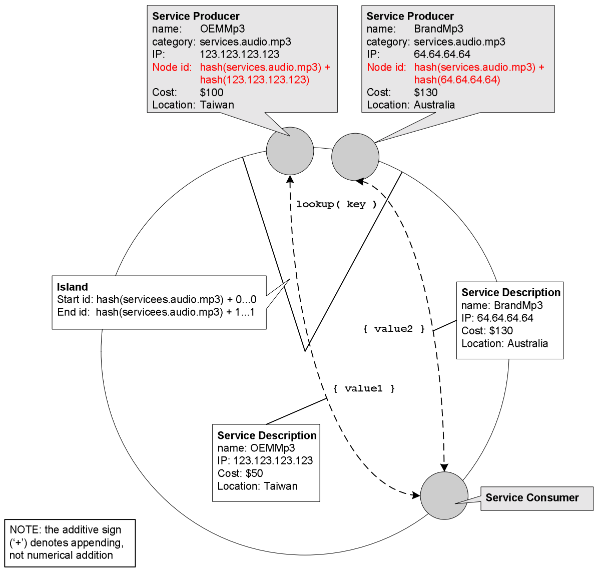


Figure 2. Semantic approach of building a peer-to-peer service directory

In terms of the functionality of a service directory, the aim is obviously to provide as much of a global view as possible for all the services under a particular category. This will provide users with maximal choice, thus been able to find the optimal Service Producer to satisfy the need. However, the reality of the situation is that looking for a service does not always mean optimization of all the possible parameters – in real life it is perhaps one or two parameters, such as locality and price of the service. The computation requirement to assess each and every service description for optimization would also be staggering expensive. Thus the more reasonable assumption to start with would be similar to the idea of anycast [15], where any Service Producers belonging to the same category can satisfy the need. Of course, Service Consumers may employ targeted anycast (e.g. choosing any service from a few known equivalent brands), which is more selective compared to pure anycast and more moderate than broadcast.

4. Ramifications

4.1. Topology

As previously mentioned, nodes offering the same type of service will sit on one continuous arc of the ring topology. The length of the arc in the identifier space will be dependent on the number of bits allocated to represent this information. From the overall network point of view, the topology will be very different to the normal Chord ring where consistent hashing guarantees the even spread of nodes across the whole ring. Now that semantic information is part of the node identifiers, there will exist a spectrum of island population, ranging from very dense to zero. However, assuming consistent hashing still applies, node distribution within an island will still be spread fairly evenly. The key to counter the imbalance is to fix the number of bits representing the semantic information at an appropriate level (within orders of magnitude) corresponding to the number of service categories.

4.2. Service Entities

The way that service entities operate on the network will also be slightly different to the traditional approach. Service Producers now serve their own Service Descriptions, giving them the full control of the robustness of their nodes on the network. It also allows the deployment of physical fail-safe measures such as clusters and shadow servers. Replication then becomes redundant (but still possible) at the peer-to-peer layer, because each node is only responsible for its own service description. In the event of an unexpected node failure, it only brings down its own presence instead of potentially many other services. Having no need for replication at the peer-to-peer layer will also mean less traffic for participating nodes. However, replication can still be useful as required by higher-layers, for example, negotiate with downstream nodes to co-host the service description as a failsafe measure or for load balancing.

Under the traditional approach of the service directory, Service Producers can indiscriminantly register their Service Description under every category, much like spamming the service directory. By having Service Producers operate their own physical nodes, the scheme dissuades irresponsible spam because each registration means more participation in the network, which is at an extra cost.

5. Experiments

The change in topology due to the assignment of semantic information in the node identifiers will result in differences to the properties at the peer-to-peer layer. Compared to the original Chord, the distribution of nodes can no longer be assumed to be uniform over the whole circumference of the ring; however, uniform distribution of nodes within islands can still be assumed. This will impact the routing and load balancing on the Chord ring.

It is well known from literature that Chord routing path length is O(logN), N being the number of nodes in the network, due to the exponential distance between each finger entry. To understand the difference between the semantic scheme and the original Chord's routing performance, simulations were performed with the same parameters for both schemes – the number of nodes (N), the number of finger entries each node maintained (m=24), and the number of messages routed in the network (50N query messages).

There was no modification done to the Chord simulator itself, but the topology generator had to be modified to generate the semantic scheme. It had to randomly produce island identifiers and then randomly assign the nodes to have identifiers that sit within these islands. For both semantic and original schemes, the simulations were setup so that each node hosted only one key that matched its own identifier. The simulations were then run as per normal of the original, with random nodes querying for random (existing) keys.

5.1. Topology Characteristics

The first set of simulations compared the routing performance of the original and semantic schemes. The number of populated islands is varied to see the impact of higher spreading of nodes across the ring. Note that "zero island" denotes the original Chord scheme. It can be seen from the Figure 3 that lower spreading (i.e. lower number of islands) meant a lower hop count. However, this decrease in hop count is negligible when the number of island is at roughly 10% of the total number of nodes (shown in Figure 4).

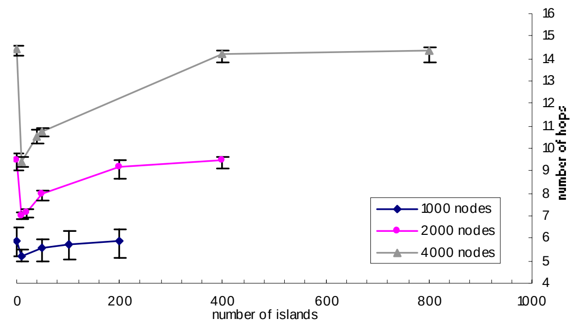


Figure 3. Average number of hops with varying amount of islands

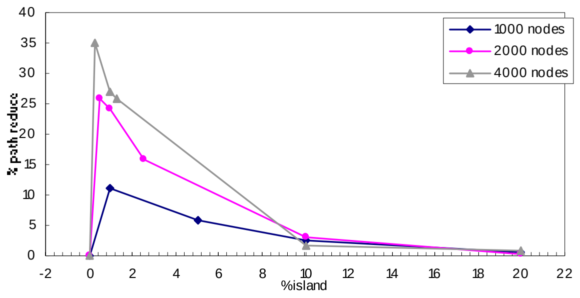


Figure 4. Percentage reduction of average hops vs. number of islands as percentage of node population

Note that for the remainder of this section, only the results for 2000 nodes will be presented due to space limitations; however, they are indicative of the trend that is observed for all node populations.

The total number of islands is determined by the number of bits allocated to represent the semantic information in the identifier. By changing the amount of bits allocated whilst keeping the number of populated islands constant, the effects of manipulating the ratio between populated and vacant islands can be seen. The default number of bits allocated for semantic information for all other experiments is set at 10 out of 24 bits. More bits allocated to semantic information means populated islands are further apart; therefore it is expected that the routing performance improves, as confirmed in the results for the second set of simulations shown in Figure 5.

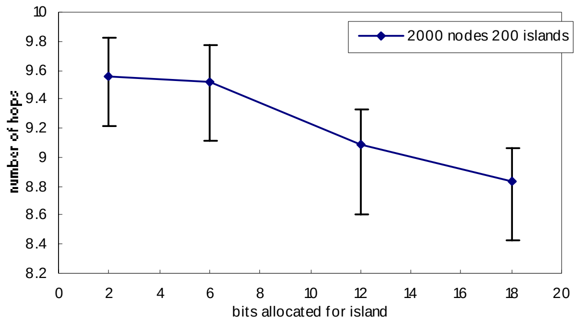


Figure 5. Average number of hops with varying amount of bits allocated for semantic information

Part of how nodes participate in the network is to forward transit packets for other nodes. The co-operative nature of peer-to-peer networks means that each node would be reciprocated when the node itself needs to send a message or a message is destined to it. We term all the messages that a node sees as the message load.

At the basic use of a service directory, each node will only host one key as each peer only serve its own service description. As each node is mapped to one key only, this scenario serves as a good indication to the base level of message load for the semantic topology. The number of nodes that experience a certain message load is presented as a probability, plotted against the number of messages per node in Figure 6. The percentile information is also given as an indication to the majority of message load.

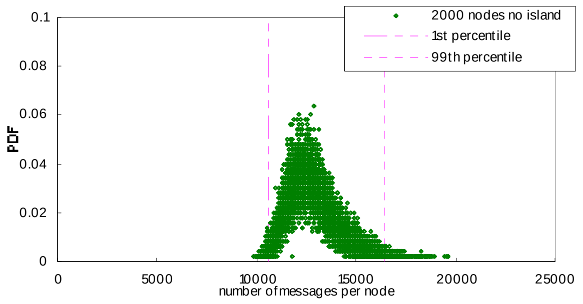


Figure 6. PDF of message load per node for original Chord topology

As one can see from the succession of spreading from Figure 6 to Figure 8, the most uniformly-spread case (i.e. no island) has the distribution that is closest to a normal distribution Figure 6. As the number of islands decreases (i.e. spreading reduces also), the distributions shows worsening skew, as depicted in Figure 7 and 8. Denser clustering produces more imbalances in the message load, meaning that a few nodes are receiving and transmitting a lot more transit messages than other nodes.

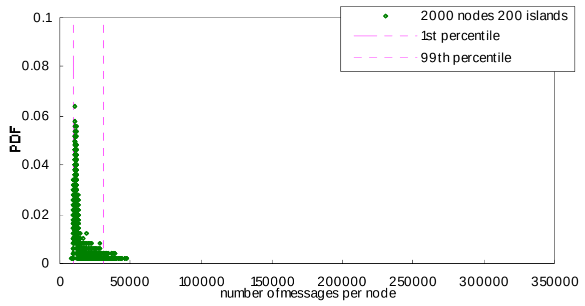


Figure 7. PDF of message load per node for 200 islands

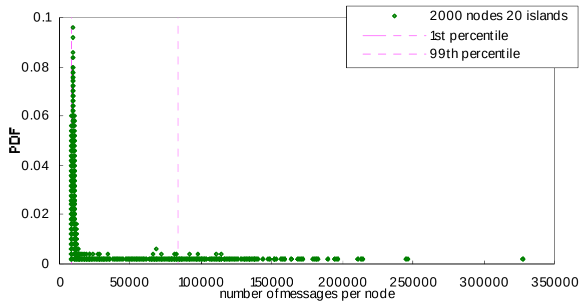


Figure 8. PDF of message load per node for 20 islands

Despite the fact that this network is used for service directory purposes, there is no reason why other types of data cannot co-exist on the network. One such example that happens to be complementary to the function of the service directory is the storage of the service tree. The service tree outlines the relationships between the service categories, much like the index of Yellow Pages which points the user to the relevant classifications. Again, unlike the centralized Yellow Pages, the service tree can be stored and accessed distributedly on the peer-to-peer network [15].

It is reasonable to assume that the storage of other types of data will be more uniform across the network, much like the spread of the keys produced by the hashing scheme of the original Chord. Thus, load balancing in terms of data storage on each node is also an important measure of the impact of the semantic scheme. In order to observe the effects of the changed topology, insertion of the same amount of random keys is performed on both original and semantic networks. The probability density function of the number of keys per node is contrasted in Figure 9 and 10, and it can be seen that the imbalance of key placement is again a lot worse in the semantic network.

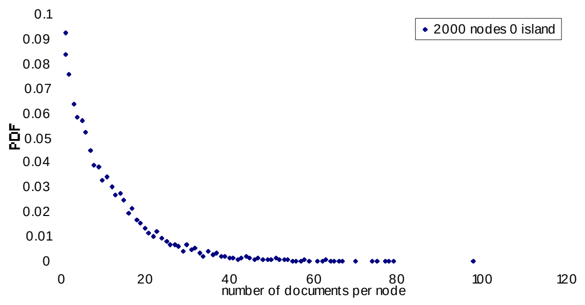


Figure 9. PDF of number of keys per node for original Chord topology

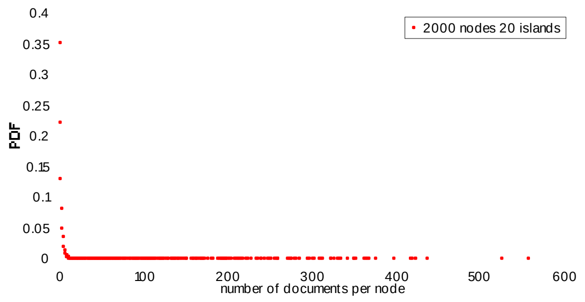


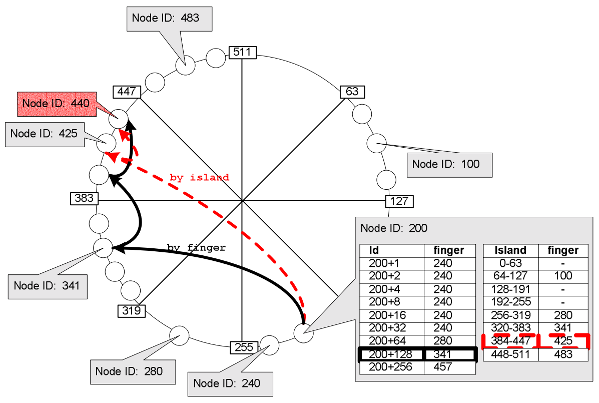
Figure 10. PDF of number of keys per node for 20 islands

5.2. Island Routing

Judging by the previous experiments, it can be seen that grouping of nodes into islands on the circular identifier space reduces the average number of hops. This phenomenon is best explained by the fact that nodes now congregate together more closely on the identifier space, within each island. Thus, messages need lesser hops to reach the destination because of the way that finger tables are constructed. The finger table of each node contains entries that are 2x units away in the identifier space, and thus there are more finger entries pointing to closer nodes than finger entries pointing further away.

Caching is a well known strategy to boost network performance, may it be reducing bandwidth or improving latency. However, caching is at a cost of storage space, and may not make any positive impact on the goals of the system if used inappropriately. It is found that broadcasting for cached information in an unstructured peer-to-peer system improves latency, but is counter productive to the goal of reducing bandwidth [17]. Structured peer-to-peer networks are designed to support up to millions of nodes (Internet size), and thus effective caching is difficult due to such large population and the inherent flat hierarchy. However, the advantage of forming islands in the Chord network means that caching can be more targeted due to the extra layer of organization.

These observations allow us to exploit a simple caching scheme which will improve the average number of hops. Basically, nodes within a close range on the identifier space would be already well-covered by the normal finger tables; but to reach nodes at a further distance would be inaccurate due to the fact that there are lesser fingers dedicated to far away nodes. Now that the semantic topology produces the property of discrete and dense islands, another routing table that caches nodes in specific islands is deemed appropriate. This is termed the island table. Nodes will use the island tables to obtain better hop counts for longer distances, yet continue to use the finger tables for closer distances.



Id	finger
200+1	240
200+2	240
200+4	240
200+8	240
200+16	240
200+32	240
200+64	280
200+128	341
200+256	457

Island	finger
0-63	-
64-127	100
128-191	-
192-255	-
256-319	280
320-383	341
384-447	425
448-511	483

Figure 11. Island routing

For this set of experiments, the simulator was modified so that each node would have the additional caching provided by the island tables. A node would use an island table entry when it can route further distance on the identifier space than the appropriate finger. The number of nodes cached per island is varied, and the figures shown below have the amount set at five nodes per island. For Figure 12, the number of nodes is increased but maintaining the same amount of islands at 10. It is clear that reduction in average hops exists only for smaller node population. However, if caching resource is increased proportionally to the increase in node population (i.e. maintain the amount of islands as a percentage of node population), then the average hop remains fairly constant and present a significant reduction at higher node populations (Figure 13).

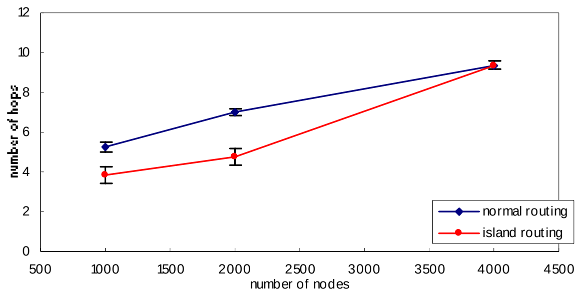


Figure 12. Average number of hops of island routing with 10 islands

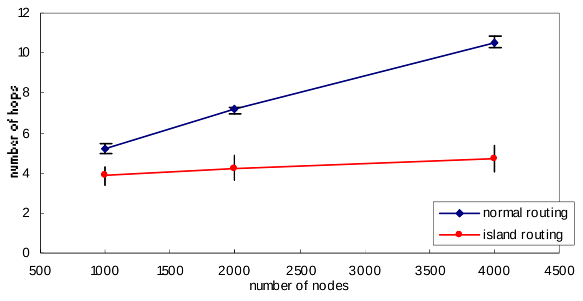


Figure 13. Average number of hops of island routing with number of islands at 10% of the node population

6. Conclusion

The placement of nodes on the identifier space grouped by their semantic information has several advantages, one of which is lower routing path lengths as evident from the experimental results. This intentional deviation from the usual random topology with uniform distribution of nodes also allows caching strategy to be more targeted and thus effective. From the standpoint of a service directory, this scheme also discourages nodes from mass registering (spamming) their service descriptions, because now each registration requires an instance of the node on the network, which is an extra cost for participation.

However, there are also immediate disadvantages to this scheme, and most notable from the experimental results is the imbalance of message transfer. It can be seen that the distribution of messages is severely skewed compared to normal Chord. The amount of data each node is responsible is also skewed, as all the keys belonging to empty islands will be placed at the first node of the first preceding non-empty island. This scenario does not occur when the scheme is used for the service directory, for each node serves only its own service descriptions; but for any other type of random data that is not hosted by the source node itself will run into this problem. It is obvious that this imbalance of message transfer and key distribution need to be addressed in the future. As a first step to address this, an incentive based solution have already been sought to reimburse nodes for forwarding traffic.

Of particular interest as future work to the goal of building a service directory include methods of incorporating locality information which facilitates efficient usage of the underlying hierarchical Internet. Locality will generally produce better latencies for services, as well as advantages of localized services, whether for reasons of better network performance (e.g. real-time transcoding) or other higher layer service requirements (e.g. only use services provided in the same country or finding a plumber in the vicinity of your house).

More general problems that also warrant further investigation include methods of dynamically increasing or decreasing the number of islands on the network, and further division of islands into sub-islands independently of a global control and policy.

7. Acknowledgement

This work has been done in collaboration with the CeNTIE project, which is supported by the Australian Commonwealth through the Building on IT Strengths (BITS) Advanced Network Program of the Department of Communications Information Technology and the Arts (DCITA).

8. References

[1] “Host Anycasting Service”, RFC 1546.

[2] A. Mislove, A. Post, et al., “POST: A secure, resilient, cooperative messaging system”, In HotOS lX, May 2003.

[3] A. Rowstron, and P. Druschel “Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems,” IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), 2001.

[4] Gnutella Protocol Development. http://rfc-gnutella.sourceforge.net/

[5] I. Stoica, D. Adkins, S. Zhuang, S. Shenker et S. Surana, “Internet indirection infrastructure”, ACM SIGCOMM'02, August 2002.

[6] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek and H. Balakrishnan, “Chord: A scalable peer-to-peer lookup service for Internet applications,” ACM SIGCOMM, 2001.

[7] J. Kangasharju, K. Ross, D. Turner, “Secure and Resilient Peer-to-Peer E-Mail: Design and Implementation”, IEEE International Conference on Peer-to-Peer Computing, 2002.

[8] J. Kubiatowicz et al., “Oceanstore: An architecture for global-scale persistent storage”, International Conference on Architectural Support for Programming Languages and Operating Systems, 2000.

[9] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, “Splitstream: High-bandwidth content distribution in a cooperative environment”, IPTPS 2003.

[10] M. Schlosser, M. Sintek, S. Decker, W. Nejdl, “A Scalable and Ontology-Based P2P Infrastructure for Semantic Web Services”, IEEE International Conference on Peer-to-Peer Computing, 2002.

[11] Napster.

[12] Open Directory Project, http://dmoz.org

[13] P. Reynolds, A. Vahdat, “Efficient Peer-to-Peer Keyword Searching”, ACM/IFIP/USENIX International Middleware Conference, 2003.

[14] R. Cox, A. Muthitacharoen, R. Morris, “Serving DNS using a Peer-to-Peer Lookup Service”, IPTPS 2002.

[15] S. Ardon, “OPENDIR: An Open Distributed Service Directory”, unpublished.

[16] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Schenker, “A scalable content addressable network,” ACM SIGCOMM, 2001.

[17] T. Hu, A. Sereviratne, “General Clusters in Peer-to-Peer Networks”, IEEE International Conference on Networks, 2003.