Symphony & Small Worlds as Overlay Networks

CS224w Final Report

Kamil Pawlowski (kamilp@stanford.edu)

Contents

Introduction

This paper presents an experiment in running small world rings (specifically Symphony) as overlay networks on top of various base graphs/networks. The aim of the experiment is to characterize the behavior of small world rings in terms of the graphs on which they are overlaid. The over arching goal of the experiment is to gain an understanding of the applicability of small world rings in real world scenarios.

In the model used by this paper nodes in the base network are said to host nodes that belong to the small world ring overlaid on top of it. Nodes in the small world ring maintain their own links (edges), but those links run over the underlying base graph. Thus navigating along the small world ring means visiting all of the base graph nodes that carry those particular ring edges. The specific focus of this paper is the number of base graph nodes that must be visited when traversing the small world ring. Shorter total paths are considered superior. This is sometimes referred to as having lower latency, or shorter distance.

This paper finds that if host nodes are chosen at random, assigning small world nodes to those host nodes so that base graph distance between adjacent small world nodes is minimized, results in shorter paths than when the host node to small world node assignment is random.

This paper finds that small world rings overlaid on top of data center like tree graphs have shorter total paths than those overlaid on top of random graphs, and Internet like graphs.

This also paper finds that when a routing operation is performed on the overlaid ring a reasonable heuristic exists for the number of base graph nodes that are traversed during this operation. This heuristic is:

$$AveTotalPathLength \le AvePathLength_{BaseGraph} \times AvePathLength_{SmallWorldOverlayGraph}$$

Finally the paper finds that time it takes to route a message around the ring is too large for main line data center applications, and that small world rings are more likely to be useful as command and control infrastructure in datacenters.

Motivation

Much of today's most widely used software runs on large clusters of computers hosted in enormous data centers. This includes everything from Gmail, to Facebook, to Amazon's Elastic Cloud (EC2 – where the datacenter is the application). Systems such as these are required to be both responsive and robust. That is, they must respond to user actions quickly, and they must be available all the time. In such systems a good deal of thought, time, and money is spent on performance. However, just as many resources, if not more, are spent on banal operational tasks required to provide availability. This includes everything from provisioning new systems, to software configuration, to replacing hardware when a fan fails. It has been the author's experience when working for Good Technology/Motorola, that operational costs of a system, especially due to salaries of datacenter employees, can eclipse hardware costs in data center applications. Thus software architectures that are fast, highly resilient to failure, and easy to maintain are highly attractive in large datacenters.

Small world networks built on one-dimensional lattices (rings) offer two principle features: as we saw in assignment 1, they are highly robust in the face of link failure; and they offer polylogarithmic routing performance [1]. Symphony [2] provides a small world that can be bootstrapped quickly and handles the arrival and departure of nodes from the ring without impacting routing performance. Thus small world networks offer fast routing and high failure resistance. Symphony is even more attractive as an architecture for use in large datacenters. Nodes can arrive and depart without impacting performance, so they can be taken out of service for maintenance or upgrade without impacting the user experience. Further, capacity can be added without user impact. This simplifies or obviates a large number of common, but often complex, scenarios that keep operations teams sweating.

Having shown how attractive a small world ring network can be, consider its construction. If one attempts to build a small world ring, in practice, it has to be implemented as an overlay network on top of other physical (or logical) infrastructures. It is not practical or cost effective to build a physical small world ring because of the probabilistic nature layout of shortcut (aka long) links. When nodes are added to or removed from the ring, the shortcut links must be re-balanced. In practice this requires laying out additional cable when the topology changes, or building a vastly over provisioned underlying infrastructure. (For the purposes of this discussion setting up short cut links as some kind of virtual circuits on an Token [3] or FDDI [4] ring counts as building an overlay network). Thus in practice small world rings must be constructed as overlays.

This having been said, overlay ring networks such as Chord [5] (an inspiration for Symphony) suffer from well-known issues with respect to the heterogeneity of nodes and links as well as with distance between nodes. Specifically, the performance of operations on a ring can be inconsistent because: some nodes are more powerful than others; some nodes are on higher bandwidth/lower latency links; some nodes that are adjacent from the point of view of the ring are very far apart in the underlying network. In each case, the performance of the ring is adversely affected because either, the node or the network inhibits the rate of flow of messages to be forwarded (routed).

Small world rings have considerable advantages. When implemented as overlay networks they potentially have serious drawbacks as well. How serious are these drawbacks? This paper seeks understand the impact of the underlying network topology on the polylogarithmic routing guarantee provided by a small world ring. The goal is to form an intuition regarding the suitability of small world rings in large-scale data center environments. Chord and Symphony originated as ideas for Internet scale Distributed Hash Tables. They were intended to provide search infrastructure for large peer-to-peer networks. A secondary goal is to determine how a small world ring performs when overlaid over the Internet.

Prior Work

Klienberg (2006)

Klienberg [1] provides theoretical background for a small world graphs. The paper provides a loose but consistent definition of a small world network: "when all (or most) pairs of nodes are connected by paths that are polylogarithmic in n". It also provides the theorems that form the backbone of the small world construction used in symphony. For example: if, for $k \geq 3$, all nodes in a [n node] graph have a degree of k there is a high probability there will be a path of length $O(\log n)$. Similarly Klienberg discusses decentralized search, as the model used in Milgram's original experiment, and the theoretical limits on that search. Finally, Klienberg lists the evolution of graphs into small world networks as an open problem.

Clauset and Moore (2003) & Sandberg and Clarke(2008)

Both of these papers [7] [8] attempt to provide mechanisms by which a small world network can arise from a graph, or by which it can be maintained. In each case, the time it takes for the algorithms to converge to a small world ring is significant enough to preclude it from being used in practice. Further, in both cases, the worst-case search performance is $O(n)$. This does not meet the standards for small world graphs.

Chord (Stoica) 2001

Chord [5] is the original distributed hash table paper. It describes a ring-based system where requests are forwarded from one node to the next, using long links to obviate an $O(n)$ search. However Chord is very fragile in relation the arrival and departure of nodes. Chords has other issues as well, such as the authors recommending that multiple nodes be run on one system, so that there will be enough nodes in the ring to ensure a good distribution of keys. While brilliant, it is not practical.

Symphony (Manku) 2003

Symphony [2] is a distributed hash table protocol built on top of a Klienberg style small world ring. The authors have taken great care to provide a ring that is able to handle the arrival and departure of nodes. Further, they provide experimental evidence showing their system in action. Of particular interest in the context of this paper is the fact that the ring can, and does, grow or shrink. The behavior is shown in their results. In contrast with the Clauset and Sandberg papers the construction in the Symphony paper is practical. One can imagine it being deployed in the real world. This is why it forms the basis for the experiment in this paper.

Experiment

A small world network (ring or graph) is implemented as an undirected overlay on top of some underlying (undirected) network (graph). Nodes in the underlying graph are said to host nodes that make up the overlaid small world ring. In this experiment, the goal is to determine the impact of the structure of the underlying network on the routing performance of the overlaid small world network. The routing performance (or path length, latency) is defined to be the path length in the base graph when routing is done between nodes in the overlaid graph.

To this end a symphony ring is overlaid on a variety of different underlying topologies. Symphony is used because it is both a small world ring (under the Klienberg definition), and has the ability to handle the arrival and departure of nodes. This means it has the routing speed and resilience of a small world ring and also the operational properties that make it worthwhile to use en vivo. It is an ideal candidate to evaluate for deployment. The properties of the overlaid ring are kept constant throughout the experiment.

Not all of the nodes in the underlying network are used in the overlaid ring. Which ones are used depends on the topology under test. Once the underlying nodes that will host the ring are selected, they are joined to form the lattice. Two different mechanisms are used to select which nodes are adjacent: proximity and randomness. These represent two real world cases: preplanning and organic growth. The preplanned case represents applications as they might come out of the box, with an engineer optimizing for efficiency. The organic growth case, simulates both expansion of an existing system, as well as what happens as systems are maintained – it is very difficult to keep them organized because failures are random.

In the first case, proximity, after the host nodes are chosen, overlay nodes are assigned to them so that overlay nodes that are adjacent in the lattice have the smallest possible base path between them. Practically: of the underlying graph nodes chosen to host the ring, one is selected to host the first overlay (lattice) node. The next node selected is, of the nodes chosen to host the ring, the one that is closest (as determined by BFS) to the first. The two hosted nodes are then linked by an edge that makes up part of the lattice. (No edges are added to the underlying graph). Subsequent nodes are chosen the same way. No underlying node can host more than one overlay nodes, so no repetition in selection is permitted. This does mean that the last edge added to the lattice can be longer than desired, but alternate construction mechanisms would be prohibitively long for the scope of this experiment.

In the second case (randomness), after host nodes are chosen, the assignment of overlay nodes to them is done at random, without repetition, until they have all been used. The lattice is built as before with overlay nodes being linked together as they are added.

This construction results in two different tests for each topology: one dubbed “ordered” where the ring lies on shortest paths, and one dubbed “random” where the underlying path length is unknown. When the lattice is in place, the Symphony long link algorithm is run on each node in the lattice to generate the long links required for a small world ring.

When the ring is in place, a number of measurements are made for each topology. Whenever a reference is made to an overlay graph, below, it is referring to a Symphony ring that has been built on top of some underlying graph. The measurements include:

- The clustering co-efficient of the nodes in the base (underlay) graph.
- The average distance between nodes in the base (underlay) graph.
- The average distance between nodes in the overlay graph (without long links)
- The average distance between nodes in the overlay graph (with long links, as computed by Breadth First Search - BFS)
- The average small world distance between nodes in the overlay graph (with long links, as computed by the algorithm described in the Symphony paper)
- The average underlying distance between overlaid nodes
- The average distance in terms of underlaid nodes in an overlaid path computed with BFS.
- The average distance in terms of underlaid nodes in an overlaid path computed with the algorithm described in the Symphony paper.

In these last two cases, this means that a path along the ring is computed using either BFS or Symphony as the search algorithm. Then for each hop in this path the BFS distance is computed in the underlay graph. For example if overlay node A is hosted by underlay node 1 and overlay node B is hosted by underlay node 10, then assuming that A and B are adjacent in the overlay graph, a computation of the path cost from A to B would consist or running a BFS between node 1 and 10.

In general trials that require the computation of an average, pick 1% of the nodes in the graph at random, and compute the average distance between those nodes and all other nodes in the graph. This is done to speedup the experimental process.

Trials for each graph are repeated five times and an average is taken, as there is a large amount of randomness in choosing the inputs.

The experiment is implemented in Java, using:

```
bash-3.2$ java -version
java version "1.6.0_26"
Java(TM) SE Runtime Environment (build 1.6.0_26-b03-384-10M3425)
Java HotSpot(TM) 64-Bit Server VM (build 20.1-b02-384, mixed mode)
```

In a number of cases, to improve the performance of the system, batch BFS computations are used. This means that a single BFS is given multiple destination nodes for a single source, and, for example, distances are computed and aggregated for all of them in one call.

Symphony Configuration

The symphony small world system is configured with k=1 (this gives polylogarithmic performance, and is equivalent to other small world configurations seen in research papers). Note that in symphony k is the number of long distance links, where as in a small world graph k is the degree of each node in the ring. In the lattice being used for this experiment k=3 (two links to adjacent nodes, and 1 long distance link). To disambiguate these two configurations this paper refers to k in a small world system as k_{sw} and k in symphony as k_{symph}. It should also be noted here that there is a specific fundamental difference between symphony and the Newman-Watts model [9]. The Newman-Watts model has random links added to random nodes with a probability p, while the symphony model has k_{symph} links added to each node. These links chose their destinations from a specific probability distribution. The symphony definition of a small world graph is inline with the Klienberg definition.

Symphony proposes that a look ahead be implemented to speed up the search process. This is not used because it violates the local routing principle that is fundamental to small world networks.

As small world networks are described in the literature as undirected, the overlay networks in the experiment are also un-directed. The underlying network is undirected as well.

Symphony rings are created via a very specific construction protocol that is run as nodes are added. This includes both a mechanism to estimate ring size, and a re-linking protocol to re-distribute links as the ring grows. The aim is to maintain a certain distribution of links as the ring grows or shrinks in size. Given that the ring sizes are known ahead of time, this experiment does not use these [Symphony] protocols. Rather it assumes that the ring has been relinked perfectly when the last node is added. This significantly simplifies the code used to construct the ring. Further, this does not compromise the efficacy of the experiment, as the results in the symphony paper make use of a less perfectly relinked ring.

It is assumed that because Klienberg's definition of small world rings is quite loose, and because the Symphony complies with this definition, wholly, that the results of this experiment can be generalized to small world graphs in general. However this is not a tight bound.

Topologies Under Test

The following topologies are used in the underlying graph:

- Large (48 port) Switch Data Center
- Small (24 port) Switch Data Center
- Small (2353 nodes) Random Graph
- Large (4706 nodes) Random Graph
- [6] CAIDA's internet router topology graph

In each case 2304 nodes are selected to form the Symphony Ring.

Large Switch Data Center

Datacenter tree with 48 nodes per switch, and three layers. This simulates a data center with high fan out switches. Nodes eligible to take part in the small world are only in the leaves. The leaves represent the servers in the datacenter. The total number of nodes is: 2353. The total number of leaves is 2304. The total number of edges is 2352. Configurations of this nature are common in industry.

Small Switch Data Center

Datacenter tree with 24 nodes per switch, and four layers. This simulates a datacenter with lower fanout switches. This requires 3 layers to match the previous case, though the top layer has only 4 nodes. The total number of nodes is: 2405. The total number of leaves 2304. The total number edges is 2404. Again, only the leaves are eligible to be part of the ring.

Small Random Graph

The small random graph is used as a control against both tree based data center models of the same size. Trees have a very low clustering coefficient (0) and this test is intended probe the impact of clustering. The number of nodes in the underlying graph is 2353, and the initial number of edges is chosen to be 2404. This matches the large switch datacenter. Edges are chosen at random. However, because the number of nodes and edges is very similar, the base setup results in a large number of small Strongly Connected Components (SCCs) that are disjoint from each other. Links are added between SCCs until the graph has only one SCC. Nodes that participate in the ring are chosen at random.

Large Random Graph

The large random graph doubles the number of nodes in the underlying graph. Otherwise it is constructed identically to the small random graph. It provides a view that is not as sparse as the trees, but more sparse than the small random graph.

CAIDA Router Topology Graph

This is an undirected graph of 192K routers in the Internet circa 2003. (More recent data sets are available on request, but for the purposes of this experiment this provides a sufficiently realistic set of data.). It is used to evaluate the performance of symphony in the Internet. 2304 routers are picked at random from the largest SCC in the graph (190K nodes) to be endpoints behind which a symphony node lives. Nodes are picked without repetition. While it may be the case that multiple Symphony nodes could live behind the same router, for the purposes of this experiment, they are assumed to be treated as just one node.

Expected Results

This experiment was conceived to see if Symphony in particular, and small world graphs in general had potential as real world systems. This having been said it also aimed to verify several thesis:

- Carefully constructed rings perform better than organically grown (evolved) rings.
- Random Graphs perform worse than trees.
- The average path length in the underlying network is an accurate heuristic for how much small world path cost increases when the underlying path length is added.

It was not clear when the experiment was conceived if the Internet graph would perform more akin to a tree structure or a random graph.

Results

	Clustering Coefficient	Ave Path Btwn Base Nodes	Ave Path Btwn Lattice Nodes	Ave BFS Path on Ring	Ave Symp Path on Ring	Ave Path Btwn Ring Nodes in Base Graph	Ave Total Path Length (BFS)	Ave Total Path Length (Symp)
SmallDC (O)	0.000000	4.4	576.0	6.0	17.8	5.0	22.0	50.6
SmallDC (R)	0.000000	4.4	576.0	6.0	17.8	5.0	37.0	100.6
LargeDC (O)	0.000000	2.6	576.0	6.0	17.6	3.0	18.8	44.8
LargeDC (R)	0.000000	3.0	576.0	6.0	17.6	3.0	26.8	71.8
Rnd Grph (O)	0.000326	9.8	576.0	6.0	18.0	10.0	42.8	87.6
Rnd Grph (R)	0.000663	10.2	576.0	6.0	17.6	9.8	74.2	201.0
Rnd Lg Grph (O)	0.000028	43.0	576.0	6.0	17.6	42.8	117.2	189.4
Rnd Lg Grph (R)	0.000035	39.6	576.0	6.0	17.8	38.4	278.0	721.8
Internet (O)	0.157413	6.0	576.0	6.0	17.8	6.0	39.4	99.2
Internet (R)	0.157413	6.0	576.0	6.0	17.6	6.0	48.0	126.0

Table 1: Summary of Results

Table 1 presents the average of the data gathered across five runs of the system. In order to facilitate the discussion below a more detailed explanation is provided for each column.

Average Path Between Base Nodes

This is the average path length in the base graph computed by BFS. This is computed by picking a fraction of the nodes at random and calculating the average path lengths to their peers. In DC (data center) and Random Graphs, 1% of nodes are chosen as probe points, and distance to all the other nodes in the SCC (graph) is computed. In the Internet graph 0.1% of nodes are chosen and distance to all other nodes in the SCC is computed.

Average Path Between Lattice Nodes

This is the average path length computed by BFS between nodes in the overlaid lattice. This is computed between all nodes in the lattice. It is computed before long links are added to the overlay.

Average BFS Path On Ring

This is the average path length, computed by BFS, between nodes in the lattice once long links are added. This is computed by using probe points in the same way that the Average Path Between Base Nodes is computed.

Average Symphony Path On Ring

This is the average path length computed by using the Symphony routing protocol between nodes in the lattice once long links are added. This is computed by using probe points in the same way that the Average Path Between Base Nodes is computed.

Average Path Between Ring Nodes in Base Graph

Given pairs of nodes in the overlaid small world ring, this computes the BFS distance between the nodes that host them. This is computed by using probe points.

Average Total Path Length (BFS)

This measure of the total path length between two nodes in the small world (Symphony) ring is gathered by first computing the BFS path between the nodes in the overlay, then computing the sum of the BFS paths between the underlay graph nodes that host them, hop by hop. This is computed by probe points.

Average Total Path Length (Symphony)

This measure of the total path length between two nodes in the small world (Symphony) ring is computed by first computing the Symphony path between the nodes in the overlay, then computing the sum of the BFS paths between the underlay graph nodes that host them, hop by hop. This is computed using probe points.

Analysis

Verification of Ring Construction

Before a further study can be undertaken, one must verify that the Symphony ring is correctly constructed.

First consider the average path between lattice nodes. Given the structure of the lattice, the node farthest from a given node is half way around the ring. So the maximum path length must be $\frac{n}{2}$. However if one computes the average path to all the other nodes in the ring, then the average path length should be half of this. I.E.: $\frac{n}{4}$. Consider that the average path between lattice nodes is 576.0 and $576 * 4 = 2304$. This means that the lattice has been properly constructed.

When considering if the long link construction has been carried out properly, the most accessible verification (though imprecise) can be done by comparing the average symphony path with Figure 3 on page 7 of [2]. If one considers the static bi-directional latency graph (for 1 long range link), it passes 2^{11} in the range of just under 18. (2^{11} = 2048 ~2304).

Considering the problem mathematically, the symphony paper indicates that the upper bound on bidirectional search should be:
$O\left(\frac{1}{k}\log^2 n\right)$ where k is the number of long links. In view of the construction of theorem 3.1 from [2] the upper bound for the scenario in this experiment is:
$O\left(\frac{1}{k}\log^2 n\right) \Rightarrow \log_2^2 2304 = 124.8$.

Clearly on all parameters the experimental data is in range. Thus the small world ring has been properly constructed, and both the search and construction comply with the specification provided in the Symphony paper.

Verification of Thesis

Ordered vs. Random

It was hypothesized that selecting nodes at random from the base graph to form the over overlaid ring was going to result in worse total path length than selecting nodes in an ordered fashion. This has been borne out.

Setup	Performance Penalty
Small Data Center	1.99
Large Data Center	1.60
Random Graph	2.29
Large Random Graph	3.81
Internet	1.27

Table 2: Performance Penalty for randomly choosing nodes

Table 2 illustrates the factor by which using random node selection is worse than choosing the nodes in an ordered fashion. Interestingly the penalty for doing so in the

Internet graph is, relatively speaking, not very large. The Internet graph also has the largest clustering co-efficient of any of the configurations. This implies that as long as the underlying graph sufficiently connected, it can overcome the choice of nodes. This bodes well for running small world based systems in the Internet, but poorly for data center configurations, where higher clustering coefficients correspond to more wiring.

This having been said, it must be noted that the datacenter graphs, which exhibited no clustering, performed better than the random graphs, which exhibit some clustering. Likely there is a combination of average path length and clustering that determine this relationship; as the average path length in the data center graphs is lower than that of the random graphs (and much lower than the large random graph).

Random Graphs vs. Trees

It was hypothesized that trees would perform better than random graphs.

Graph Type	Average Total Symphony Path Length
Tree	66.95
Graph	299.95
Internet	122.60

Table 3: Average Symphony Path Length by Underlying Graph Type

Table 3 averages the total symphony path lengths across all the graphs of a given type. This indicates that the gross hypothesis is correct. However looking under the covers one can see that there is some overlap between the worst case performance of trees and the best case performance of the graphs and Internet. It is not clear why this is the case.

Path Length as a Heuristic for Total Symphony Path Length

It was hypothesize that the path length in the underlying graph would provide a heuristic that would indicate how much path cost increases when the underlying path cost is added to the cost of a symphony path.

	Clustering Coefficient	Ave Path Btwn Base Nodes	Ave Path Btwn Ring Nodes in Base Graph	Ave Total Path Length (Symp)	Ave Total Path Length (Symp) /Ave Path Btwn Base Nodes	Ave Total Path Length (Symp)/ Ave Path Btwn Ring Node in Base Graph
SmallDC (O)	0.000000	4.4	5.0	50.6	11.50	10.12
SmallDC (R)	0.000000	4.4	5.0	100.6	22.86	20.12
LargeDC (O)	0.000000	2.6	3.0	44.8	17.23	14.93
LargeDC (R)	0.000000	3.0	3.0	71.8	23.93	23.93
Rnd Grph (O)	0.000326	9.8	10.0	87.6	8.94	8.76
Rnd Grph (R)	0.000663	10.2	9.8	201.0	19.71	20.51
Rnd Lg Grph (O)	0.000028	43.0	42.8	189.4	4.40	4.43
Rnd Lg Grph (R)	0.000035	39.6	38.4	721.8	18.23	18.80
Internet (O)	0.157413	6.0	6.0	99.2	16.53	16.53
Internet (R)	0.157413	6.0	6.0	126.0	21.00	21.00

Table 4: Base Path Length vs. Total Path Length

The data in Table 4 expands on the previous result with respect to ordered vs. random ring node selection. While the ordered setups vary widely with respect to the ratio of total path length to average path length between base nodes, the random graphs are fairly consistent. The ratio, when considering just the base nodes that host the small world ring, averages to 20.87; and 21.15 when the average path length between all base nodes are considered. This value [20.87] appears suspiciously close to the average path length between nodes in the symphony ring: 17.72. While this is not conclusive, and it is not clear why there exists a discrepancy, it does indicate that there is a possible reasonable heuristic for the total number of nodes that a message routed along the small world ring must travel. Recall that when messages are passed along the overlay small world ring, they must actually pass hop by hop between nodes in the base graph. Thus the path length constraint provided by the small world is not the only one that applies. That being said, it appears that, for a random arrangement of base graph nodes in the ring the average total path length appears to be approximately:

$$AveTotalPathLength = AvePathLength_{BaseGraph} \times AvePathLength_{SmallWorldOverlayGraph}$$

Further when the nodes in the base graph are ordered to shorten path lengths this becomes:

$$AveTotalPathLength \leq AvePathLength_{BaseGraph} \times AvePathLength_{SmallWorldOverlayGraph}$$

These results make intuitive sense. They are however incomplete, as it is not clear why the ratio is not exact. It is also not clear how involved the math would be to prove this bound absolutely.

Performance on the Internet Graph

When the experiment was conceived there was no expectation regarding the performance of the small world ring on top of the Internet graph. There was too little known about this graph. At this point it appears that a ring overlaid on the Internet graph behaves more like the data center graphs than the random graphs. This is likely due to the fact that the (relatively) high clustering coefficient in the Internet graph keeps the average base graph path length between hosted Symphony nodes low.

Additional Observations

Originally systems such as Symphony were intended to be implementations of dynamic hash tables. In the course of this experiment, Symphony has been (partially) characterized with the aim of determining its deployability. In the context of web applications hosted in large datacenters latency (that is path length) is key. [10] asserts that round trip time in a modern data center can be on the order of 250 microseconds, and that the amount of time a request has to complete (the "all up SLA") can be on the order of 300 milliseconds. This request completion time includes breaking up the request into subrequests, sending these off to other servers to compute, and then aggregating the result. Symphony and other small world rings are competing in this context with systems such as memcached [11] which use $O(1)$ (static) routing. Even if symphony were implemented natively (that is with no underlying graph) the average time to route a message would be: $\frac{250\mu s}{2}\times 17.72 = 2215\mu s = 2.215ms$. Not only is this a significant part of the all up SLA, it is an order of magnitude larger than the $125\mu s$ an $O(1)$ router requires to perform the same operation. Even if the most aggressive result from [2] is used (k=4, bidirectional routing, 1 look ahead) the time to route a request is still $7.5\times 125\mu s = 937.5\mu\text{s}$, which is very large relative to the $O(1)$ system. Where systems such as memcached fall down however, is that their routing tables are static, and thus a great deal of work must be done in order to keep these synchronized in the face of failure. Thus symphony and small world graphs do not have the performance required to be in the direct request-handling path.

This having been said, the operational properties (resilience to failure, ability to handle nodes being added and removed while maintaining search speed) of symphony are such that when combined with its relative speed (it is not slow, just not fast enough) it makes an attractive candidate for a command and control system. Possibly it can serve $O(1)$ routing information.

Conclusions and Future Work

This paper has evaluated small world rings, and symphony specifically, as overlay networks on other graphs. It has found that it is effective to choose nodes closer together in the underlay graph for adjacent nodes in the overlay graph. It has found that when such graphs are built on tree graphs, the aggregate performance of the system as a whole is better than when such graphs are built on random graphs. The experiment covered in this paper has also found that a reasonable heuristic for the total number of nodes that a message must travel in the underlay graph when being routed by the overlay is: $AveTotalPathLength \leq AvePathLength_{BaseGraph} \times AvePathLength_{SmallWorldOverlayGraph}$.

There are several investigations that should logically follow from this work. On the theoretical side, a more detailed investigation is required into the reason that the heuristic is approximate and not exact. Further this heuristic should be verified mathematically. If possible one should see if it could be turned into a tight upper bound. More practically, this experiment should be expanded to provide an analysis of which edges of the base graph get the most use. This would allow engineers working with such overlay networks to understand where the bottlenecks in their systems are. More importantly, the same kind of experiment should be carried out on other small world ring constructions to verify if the generalization from symphony to small worlds more generally is valid.

References

1. Kleinberg, Complex networks and decentralized search algorithms, Proceedings of the International Congress of Mathematicians, Madrid Spain 2006
2. Manku & al., Symphony: Distributed Hashing in a Small World, Technical Report. Stanford InfoLab, 2003
3. http://en.wikipedia.org/wiki/Token_ring
4. http://en.wikipedia.org/wiki/Fiber_Distributed_Data_Interface
5. Stoica & al., Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, Proceedings of ACM SIGCOMM 2001, San Deigo, CA, August 2001
6. http://www.caida.org/tools/measurement/skitter/router_topology/
7. Clauset and Moore, How Do Networks Become Navigable?, 2003
8. Sandberg and Clarke, The Evolution of Navigable Small-World Networks, 2008
9. Newman, Models of the Small World: A Review, J. Stat. Phys. 101, 819-841 (2000)
10. Alizadeh & all, Data Center TCP, SIGCOMM 2010
11. http://code.google.com/p/memcached/wiki/FAQ#Cluster_Architecture_Questions

Appendix 1: Running the Software

The machine used for this experiment was a 2.4 Ghz Core 2 Duo with 4GB of ram running Mac OS X version 10.6.8

The version of Java used for this experiment was 1.6.0_26.

The software is built with ant. A build.xml is provided.

Two targets are available: ant, ant clean. The former builds the system. The latter wipes out all binaries.

A shell script is provided to simplify execution.

To build and run a single test run, do the following:

```
ant clean
ant
./exec.sh Main.java itdk0304_rlinks_undirected
```

A single test run takes on the order of 75 minutes to run on the development machine.

The final output of a run looks like this:

```
+++++++++                 name  scc #scc       cc        base dist          r-dist
r-dist-ll  r-dist-ll-symp     underlay-rd underlay-rd by hop underlay-rd symp by hop
.........    SmallDC (ordered) 2405    1 0.000000              5             576
6              18              5               22             51
.........     SmallDC (random) 2405    1 0.000000              5             576
6              18              5               37             99
.........    LargeDC (ordered) 2353    1 0.000000              2             576
6              17              3               19             46
.........     LargeDC (random) 2353    1 0.000000              3             576
6              18              3               27             72
.........  Rnd Graph (ordered) 2353    1 0.000312             10             576
6              19              10              42             86
.........   Rnd Graph (random) 2353    1 0.000511             10             576
6              17              10              75            197
......... Rnd Lg Grph (ordered) 4706    1 0.000142             39             576
6              18              38             110            184
......... Rnd Lg Grph (random) 4706    1 0.000000             35             576
6              17              31             233            596
.........          I (Random) 190914  308 0.157413               6             576
6              17              6               49            122
.........         I (Ordered) 190914  308 0.157413               6             576
6              17              6               39            102
```

For this paper five runs were done in serial.

Appendix 2: Raw Experimental Results

Raw experimental results can be found in results.txt in the associated tarball.