# RC-Chord: Resource Clustering in a Large-Scale Hierarchical Peer-to-Peer System

Daniel R. Karrels, Gilbert L. Peterson, Barry E. Mullins
Air Force Institute of Technology
{daniel.karrels,gilbert.peterson,barry.mullins}@afit.edu

*Index Terms*—**Distributed Multi-Agent System, Hierarchical Peer to Peer, Command and Control, Resource Clustering**

***Abstract*—Conducting data fusion and Command and Control (C2) in large-scale systems requires more than the presently available Peer-to-Peer (P2P) technologies provide. Resource Clustered Chord (RC-Chord) is an extension to the Chord protocol that incorporates elements of a hierarchical peer-to-peer architecture to facilitate coalition formation algorithms in large-scale systems. Each cluster in this hierarchy represents a particular resource available for allocation, and RC-Chord provides the capabilities to locate agents of a particular resource. This approach improves upon other strategies by including support for abundant resources, or those resources that most or all agents in the system possess. This scenario exists in large-scale coalition formation problems, and applies directly to the United States Air Force's CyberCraft project. Simulations demonstrate that RC-Chord scales to systems of one million or more agents, and can be adapted to serve as a deployment environment for CyberCraft.**

## I. Introduction

Security and systems management requirements continue to grow and evolve. The CyberCraft program proposes to use the latent processing power of United States Air Force (USAF) network attached devices to conduct redistributable computing for automated network defense, extracting Cyber Situational Awareness (SA), and other critical cyber missions. The CyberCraft project seeks to equip every business class computing system node in the USAF with an agent capable of receiving and executing payloads that directly correspond to USAF missions. The agents themselves are multi-taskable and the tasks may require multiple agents.

One component of this effort is the design of a large-scale C2 structure to operate the CyberCraft Distributed Multi-Agent System (DMAS). Building upon existing P2P technologies, RC-Chord provides a framework in which agents can be located by resource or capability. Resources are defined as assets or services that an agent may possess. They may include processor time, distributed mutex locks, decision authority, or other services such as data fusion for Cyber SA.

Given the size of the United States Department of Defense (DoD), the network of CyberCraft agents may exceed one million agents. A fundamental obstacle to the CyberCraft project is the organization and management of a network of autonomous agents of this scale. With one million or more agents in the system, even modern P2P designs become burdened with maintenance traffic. RC-Chord builds upon existing P2P technologies by constructing a hierarchy of clusters aimed to support searching by resource, and incorporates simplifying assumptions and distributed stores to reduce the total computation necessary for self-organization algorithms.

The contribution of the RC-Chord structured overlay is the development of a system that is explicitly built to perform distributed C2 in systems of one million agents. Associated with each agent is one or more resources that the system may need to locate, and RC-Chord thus supports agent location by either address or resource type. Moreover, RC-Chord is designed to support abundant resources, or those resources that many or all agents may possess, such as processing time.

This paper begins with a discussion of related work, a statement of RC-Chord's objectives, and its design principles. It introduces the system's design elements and measurement criteria for simulation experiments. Once the experiments are described and analyzed, this paper concludes with a discussion of future work and conclusions.

## II. Related Work

A P2P network is one in which nodes interconnect to each other, typically with out-degree greater than one, where a node may connect to multiple other nodes. Peers maintain no distinction between server and client nodes. Rather, nodes are considered equal, and any of them may provide services to the network, to include routing services.

These systems scale without maintaining global knowledge. Instead, each node remains connected to some number of other nodes, often logarithmic in the size of the system, and messages are passed between peers using a heuristic [5]. Due to the lack of global knowledge, the heuristic cannot deterministically know the target node's location, and instead forwards each message in the most likely direction of the target node. The design of this heuristic is a distinguishing component that separates the different structured P2P topologies.

The current generation of P2P systems is led by two similar approaches, Chord [10] and Pastry [9]. Each design provides node resolution through address lookup. In these systems, each node maintains a table of pointers to nodes in differing parts of the address space. Nodes maintain more pointers to other nodes that are closer to themselves than those far away. With this system, finding target nodes proceeds through divide and

The authors would like to acknowledge the funding and support of the Air Force Research Laboratory, Cyber Defense Branch.

The views expressed in this article are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

conquer, with the resolution improving with each successive step.

### A. Chord Based Hierarchical Peer-to-Peer (HP2P)

HP2P overlay structures combine flat P2P systems together to form a hierarchy of P2P systems [6]. Two-layer HP2P systems provide a top-level topology for indexing into second layer P2P networks. HP2P systems can also be organized into arbitrarily many layers to provide further scaling and organization. Each cluster, or group, in a HP2P network is a separate P2P network, and contains one or more super peers. A super peer is a node in a cluster that is given additional responsibilities, such as decision authority, message routing to other clusters, or maintaining replicated copies of distributed data structures. Super peers are normally chosen by their superior reliability or performance characteristics. The super peers from two or more clusters interconnect to form another P2P network, and this process may be repeated many times to form a hierarchy of P2P networks.

Garces-Erice et al. [3] demonstrate that even adding a single P2P layer to an existing P2P architecture can improve the lookup path of searches by a factor of $logN/logI$, where $N$ is the total number of peers in the system and $I$ is the number of clusters. Their system consists of two layers: a top-level Chord ring of super peers in a modified Chord overlay, and a second layer of multiple heterogeneous structured P2P overlays. The authors specifically cite the advantages of a HP2P approach as transparency, reduction in average hop length, lower bandwidth consumption, and improved heterogeneity [11], [12].

The Canon project [2] extends this work by providing methodology for merging structured P2P overlays (Chord, Symphony, CAN, and Kademlia) into hierarchical structures. The methods employed also support $O(\log N)$ bounds on the degree for nodes in Crescendo, Canon's adaptation of Chord.

ML-Chord [8] applies these technologies to create a system organized by available resources. ML-Chord is a two-layer HP2P overlay network, where the top (or bridge) layer joins super peers from each of the clusters at the second layer. These Category Layer clusters each represent a single resource, and nodes join one or more clusters based on their available resource(s). Chord is used as the base node location protocol, and is suitably modified to accommodate search by resource category. Two notable disadvantages of ML-Chord are its fixed size (two layers), and limited scalability for large-scale systems. RC-Chord extends ML-Chord to address these limitations.

For DMAS and C2 applications, HP2P systems provide the important opportunity for separation of function. This is a critical property in C2 systems, where security of missions must be strictly maintained and monitored. A separation of function supports this scenario by imposing quantifiable and observable boundaries for mission oriented systems, while at the same time permitting the necessary communications channels for non-mission related data transfer (management, coalition formation, etc). It also provides practical restrictions on the size of the coalition formation problem for large-scale P2P systems.

## III. RC-Chord

RC-Chord extends the Chord structured overlay technology by incorporating the HP2P organizational structure. It also adds the capability to search for agents by the resource(s) they own. The primary objectives of RC-Chord include:

- Leverage existing structured overlay techniques to develop a new strategy which supports the HP2P topology. This structured overlay strategy provides reliable and scalable communications in mission oriented systems.
- Scale to one million or more agents.
- Support agent location by address or resource identifier.
- Incorporate a search mechanism to support the construction of coalitions of agents to accomplish missions within large-scale HP2P systems.

Existing P2P solutions support systems of the required scale, however they do so with higher maintenance costs and with limited facilities for global algorithms. HP2P systems incorporate the ability to organize by criteria, which can have significant impact on the performance and maintenance characteristics of the system. In particular, the HP2P design construct incorporates separate clusters of self-sufficient P2P systems, thus greatly reducing the maintenance and search bandwidth requirements for cross-boundary links [3]. This localization also provides a logical grouping that commanders can more easily understand and supports localized ownership by system administrators.

### A. Top-Level Design

RC-Chord scales to many levels, with each level composed of one or more clusters. Each cluster is a stand-alone instance of Chord, and connects to a neighbor cluster in the next higher level of the hierarchy through a set of super peers. These super-peers are so named because they generally exhibit additional capabilities, particularly by supporting the increased burden of communications and processing associated with being a gateway node between two clusters. Each cluster may have zero or more sub-clusters attached to it, and up to one higher level cluster. The organization of clusters into a HP2P structure resembles a tree, with a single root and $c \in [0, b]$ children, with branching factor $b$ determined by the ratio of peers to super peers.

RC-Chord associates each node with one or more resources. A resource is defined as a capability or asset that an agent possesses. These include, but are not limited to, processor capabilities, hardware resources, data sets, payloads, and others. RC-Chord supports searching for agents by global identifier or resource. The hierarchy grows and shrinks dynamically to accommodate network churn and abundant resources. An abundant resource is a resource that many, if not all, agents possess, such as processor time.

RC-Chord extends the ideas present in Multi-Level Chord (ML-Chord) to accommodate large-scale systems (greater than one million nodes). ML-Chord introduced the importance of

resources in large-scale networks, and used a two-layer HP2P system to organize the network to provide the property that agents could be located by resource. Unfortunately, approaches such as ML-Chord do not scale for systems of abundant resources, because the layer associated with processor time may consist of all agents in the system (one million or more), and this presents a significant bottleneck for system performance due to maintenance overhead. Operating on such a large cluster would also amplify the difficulty of building coalitions that include members of an abundant resource. Under such a system, the $\mathcal{NP}$-complete coalition formation problem becomes intractable. RC-Chord adds the ability to extend the hierarchy to an arbitrary number of layers, and to support abundant resources directly. The introduction of more than two layers in the hierarchy also reduces the scope of control of the higher layer of the hierarchy, wherein the super-cluster was originally directly responsible for all processing in the network.

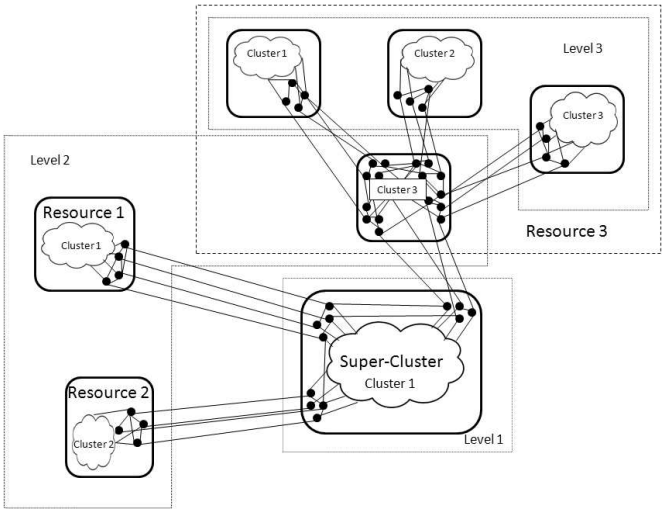


Fig. 1. An example RC-Chord instance with three levels. The super-cluster exists as the sole level one cluster. Its nodes serve as super peers for level two clusters, with a single level-two cluster for each resource. Clusters at level two begin the formal sub-graph for each resource type, and may extend into additional levels based on number of nodes for each resource.

Figure 1 shows an example RC-Chord instance with seven clusters. Three resources are present, with all three represented in the super-cluster. When a resource's agent population high-threshold limit is exceeded at the super-cluster, a new cluster for that resource is created at the second level. Once a cluster at the second level is filled, nodes joining the system with that resource are attached to a new cluster at the next level of the hierarchy. Figure 1 shows a single level two cluster for each of the system's three resources. Nodes possessing resource three have continued to join the system, and new clusters for that resource were created at level three. This process repeats, with sub-graphs of each resource growing outward from the super-cluster, to accommodate new agents joining the system.

The number of sub-clusters that each cluster can maintain depends on the ratio of super peers to peers. This ratio is determined through experimentation and analysis of application requirements, and directly affects the overlay's properties. With more super peers per cluster, nodes will see a reduced average latency and improved resilience through higher numbers of inter-cluster communications links. However, due to performance considerations, an agent may only be a super peer of a single cluster, and increasing the ratio of super peers to peers increases the number of clusters in systems of equivalent numbers of agents.

The top level cluster in the RC-Chord hierarchy is the super-cluster. The super-cluster is an entry point for locating a resource in the system, and resides in layer one of the hierarchy. The first layer is composed of the single super-cluster, however subsequent layers have a geometrically increasing number of clusters. The super-cluster is the only cluster in the system that contains agents of different resource sub-graphs. The super-cluster joins all resource networks together for the purposes of simplifying resource location algorithms.

Presently, the number of resources available to the network remains fixed during runtime, and the number of nodes dedicated to each resource in the super-cluster is evenly distributed [10]. Each node in the super-cluster is responsible for a single resource and serves as a super peer of the layer two cluster for that resource. The number of nodes in the super-cluster for each resource is configurable, thus providing performance tuning capabilities.

Beginning with the second layer, each cluster along the path from the super-cluster to a leaf node is responsible for the same resource. When a cluster becomes full or exceeds an upper population threshold due to the introduction of new agents into the system, a new cluster of that resource is formed. This new cluster is either placed at the same level, or a new level of clusters is created. The reverse scenario also applies for systems that experience a disproportionately large number of agents leaving the network: underpopulated clusters are combined to form larger clusters, and underpopulated levels are merged to form more populated levels.

Similar to nodes, each cluster receives an identifier that is locality-unique. The identifier minimizes the width of global node addresses and creates a fullness attribute that is used for agent addressing. Should an agent leave the network, an agent from a lower layer cluster is promoted to fill the previous agent's position and assumes its responsibilities. Much like a balanced tree, this promotion propagates down the tree to the lowest level. This strategy maintains a full address space in each cluster, reducing agent lookup failures, and enforcing the Chord requirements of uniform distribution of addresses.

### *B. Application Design*

RC-Chord makes extensive use of the facilities provided by the Chord protocol. It can be creatively tuned to yield specific performance metrics and organizational hierarchies. Among the more important parameters, RC-Chord includes the following coarse-grain variables:

- $N_n$ The number of expected nodes
- $N_r$ The number of resources
- $m$ The width of a node address, in bits

- $P_{sp}$ The number of peers to super peers (peer to super peer ratio).

Strictly speaking, the number of nodes, $N_n$, is not a design parameter as the system is built to scale to undetermined numbers. Among its strengthes, RC-Chord networks maintain the relative expansion of clusters per level no matter how large the system becomes. However, the expected size of the system is an important parameter in choosing the particular hierarchical structure that RC-Chord adopts, as experimental results demonstrate.

The maximum number of agents in each cluster is $2^m$ nodes. A single cluster exists at level one ($C_1 = 1$), and the number of clusters at level two is $C_2 = N_r$. For each successive cluster, $C_i$, the number of clusters per level, $C_l$, is:

$$C_l = C_{i-1} * 2^m / P_{sp} \tag{1}$$

Given a uniform distribution of resources to nodes, the total size of the network up to level $l$ is given by:

$$N = 2^m * \sum_{i=1}^{l} C_i \tag{2}$$

As shown, the clustering hierarchy is tightly tied to the address width, $m$, and the peer to super peer ratio, $P_{sp}$. Decreasing $m$ yields smaller and better performing clusters, but increasing the number of clusters and layers in the hierarchy, for a fixed number of nodes in the system. Increasing the super peer ratio increases the number of super peers to leaf peers, increasing the available bandwidth between clusters, but also increasing the number of clusters in the system. Both of these parameters provide useful tuning opportunities for runtime performance and bandwidth consumption.

### C. Addressing in RC-Chord

Addressing in RC-Chord operates similar to Chord: each agent has a unique address inside of an $m$-bit identifier space. The distinction in RC-Chord is that each Chord cluster maintains its own unique identifier space, and the global identifier for each node is determined by its path from the super-cluster. Using the fullness property, the width of a global identifier can be reduced by representing the path from the super-cluster to the target node as a sequence of cluster identifiers, followed finally by the agent's cluster-specific identifier.

Figure 2 shows the global address for an agent located in level four. Starting with the super-cluster, the addressing proceeds left to right. Each segment of the address before the final segment represents the unique cluster identifier for the next cluster in the path to the target agent. The final segment of the address is the agent's locality-unique address for its own cluster.

### D. Searching for a resource

RC-Chord provides the facilities to locate the sub-graph associated with a resource, but does not explicitly maintain any form of storage mechanisms for tracking quantities of the available resource. Searching for a resource begins by

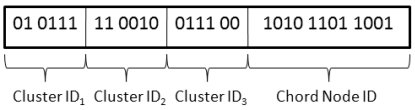


Fig. 2. Global node address for a node at level four. Beginning from the left at the super-cluster, each stage in the address represents the ID of the cluster at the next level. The final segment of the address is the cluster-local identifier for the target agent. The combination of these cluster ID's and agent ID together form a global agent address.

forwarding a request from the source agent to the super-cluster. This process is expected to consume $(l-1) * O(log(2^m))$ or $(l-1) * O(m)$ hops, where $l$ is the level number of the source node, and $m$ is the node address width. Since the super-cluster has membership for each possible resource, only $O(log(2^m)) = O(m)$ more steps are required to locate a super-peer of the necessary sub-tree. This yields an expected minimum resource search time of $l * O(m)$ hops. This process depends on a reliable mechanism for mapping resource identifier to the set of nodes responsible for that sub-tree.

Once the proper resource sub-graph is located, the search algorithm may choose to descend as far into that tree as it desires. Increasing the depth of the search examines larger portions of the network, and may improve searches for agents with the desired characteristics. The exact algorithm used to decide which level/cluster/agent to choose relies on the application. RC-Chord provides the entry point into the proper resource sub-graph from which to initiate these searches, and the communications mechanisms with which to perform that search efficiently and reliably.

### E. Resource to ID Mapping

Given a resource $r$, the protocol maps the set of nodes responsible for $r$ within the super-cluster. RC-Chord's node address mapping for each cluster attempts to evenly distribute the addresses of new nodes into the address space. This is done using an algorithm that aids in later searches by assigning node addresses in a known order. Miss mitigation occurs along a known path that is most likely to result in the earliest possible address hit, while at the same time uniformly distributing node addresses.

RC-Chord uses a mapping of the form $\delta = R_n * n$, where $R_n$ is the resource number, and $n$ is the $n^{th}$ node in the cluster of resource $R_n$, and begins with $n_0 = R_n$. Figure 3 illustrates this process. The mapping divides the $2^m$ entries in each cluster's address space into $R_n$ evenly sized intervals. Each interval has an address identifier for each of the $R_n$ different resources, with the resource in the same relative location for each interval.

This resource to identifier mapping also provides the property of reversibility. Given a node address in the super-cluster, it is possible to determine the resource that the node represents at the super-cluster with modulo arithmetic. This is useful for an alternate global addressing scheme which maintains node addresses at each cluster, versus cluster identifiers. Should a node depart, a miss occurs, and a reverse mapping is performed

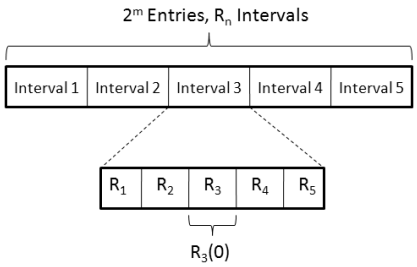


Fig. 3. Example agent mapping for a cluster with $m$ bits of address width, and $n$ resources. The $2^m$ address entries are divided into $R_n$ intervals. Each interval has one entry for each of the $n$ resources. The initial interval for resource three is shown as $R_3(0)$, being the first entry for resource three in interval three.

to identify the resource with which that node was associated. The forward mapping is then applied, and a new node for that resource is identified to complete the message transfer.

## IV. Experimental Setup

These experiments profile and verify the expected results of RC-Chord. An RC-Chord implementation has been created to operate within the Peersim [4] P2P simulator. The baseline system for validation is an RC-Chord instance with one resource, and an identifier width, $m$, large enough to allow all nodes in the system to reside in a single cluster. This scenario replicates the baseline Chord protocol, and is used to validate experimental results. The testing assumes a reliable communications mechanism.

### A. Response Variables

The response variables in these experiments are the mean message hop length of messages between any two nodes and the agent lookup success rate. The mean hop length is determined based on messaging from each agent to each other agent in the network. This one-to-all broadcast includes destinations that are both inside and outside of the source node's cluster. This is a primary measure of the efficiency of the protocol. Because RC-Chord relies upon the Chord protocol, and because it incorporates a layered approach, the mean hop length for RC-Chord should not differ from that of Chord by more than a constant factor proportional to the number of layers in the HP2P structure.

During network churn, agents that are new to the system will not be reachable immediately. Likewise, agents will not know about recently departed agents until the notification of the departure is propagated outward. These scenarios create possibilities for agent lookup or message delivery failures, and the rate of these failures is proportional to the churn rate. This response variable presents a useful indication of the quality and delays of the protocol during periods of high churn or failures.

### B. Process Variables

The RC-Chord experiments exercise several process variables. These variables, shown in Table I, are chosen to configure the system according to the baseline Chord protocol.

TABLE I
Simulation Process Variables

| Variable | Range |
|---|---|
| Address Width (bits) | 10, 11, 12 13, 14, 15, 16, 17, 18, 19, 20 |
| Network Size (agents) | 1000, 2000, 4000, 8000, 16000, 32000, 64000, 128000, 256000, 500000, 1000000 |
| Static Churn Rate (percent) | 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 1.0 |
| Business Churn | True, False |

The primary process variables are the Chord identifier address width, the number of agents in the system, and the churn rate. The Chord address width, $m$, is initially set to the minimum size required to place all agents of a 1000 node system into a single cluster ($m$=10). This is one of the baseline Chord experiments, used to validate the approach against a known system. The value of $m$ increases to 20, which is the minimum size required to place all agents in a system of one million agents into a single cluster. In cases where $m$ is too small to allow all agents to reside in a single cluster, the RC-Chord protocol distributes the agents into multiple clusters at multiple levels. The number of agents in the system varies between 1000 and one million agents. The static churn rate varies from 0% to 1.0%, with churn performed every 60 simulated seconds. For a system of one million agents, this represents a churn of up to 10,000 agents leaving, and 10,000 agents joining the system every minute.

Experiments employ a business class churn model in addition to the static churn rate. An agent's session time, or length of time an agent is connected to the network before leaving, is patterned as a mixture model of two Laplace/Pareto distributions [1], [7]. This model represents the standard business practice of logging into a computer at the beginning of the day, and logging off at the end of the work day. For the purposes of these experiments, the entry time of 0800 is used, and the departure time is 1700. This model follows a bimodal Laplace distribution centered around the above times, with a mean variance of 30 minutes to accommodate those who arrive or depart either before or after the median times. The magnitude of the maximum business churn is also 1.0%, and occurs at the median of each business churn distribution.

### C. Control Variables

The control variables for the RC-Chord simulations establish consistent parameters between experiments. With the exception of the baseline experiments, the number of resources in the system is held at five. The presence of more than one resource stimulates the creation of multiple branches of the P2P hierarchy, thus exercising the RC-Chord cluster construction and destruction protocols. Baseline Chord systems use a single resource, so the agents are allocated in a single cluster, and are otherwise equal peers.

A peer to super peer ratio of 512 is used to exercise the super peer promotion and inter-cluster communications portions of the RC-Chord protocol. The length of the Chord successor list is restricted to $2m$ to maintain the Chord logarithmic

node memory usage. Message transport delays are kept to the interval [0,10] milliseconds. This represents a Local Area Network (LAN) deployment, which will not necessarily meet all target environments. Evaluating systems which use different values of this control variable is an area of future work.

Message delivery attempts are halted after a message has reached $5m$ hops. This number is high enough to give the system ample opportunity to perform maintenance on newly departed or arrived agents, and eliminates any messages from entering an infinite cycle. A message delivery or agent lookup failure occurs when an agent is authoritatively determined to not be present in the system, or the hop limit is reached.

Minor maintenance occurs at most every 10 seconds. During a maintenance period, if an agent has detected a change in the system, it sends each of its neighbors a message with those updates. This information is used to update each agent's internal tables. Each agent is connected to approximately $\log N$ neighbors. Since each agent sends update messages to each of its neighbors, this maintenance traffic requires at most $O(N\log N)$ messages every maintenance period, under the unusual circumstance that each agent detects a change in the system in a single period.

## V. Results and Analysis

Testing against a baseline Chord instance is used to validate the RC-Chord simulations. These experiments configure RC-Chord such that all agents in the system reside in a single cluster. Once validated, the test matrix in Table I is followed to exercise the RC-Chord protocol.

### A. Chord Baseline Validation

The validation of RC-Chord against a baseline Chord system demonstrates the correctness of the underlying protocol and implementation of RC-Chord. To validate against Chord, RC-Chord is configured to use a single resource, large $m$ value (32), and no churn. This forces all agents in the system to form a single cluster in which all agents are peers. With this configuration, a set of standard experiments generates data used to validate the model.

Figure 4 shows the mean hop length with first standard deviation for the baseline test case. The logarithmic upper bound is the expected upper bound of the baseline Chord protocol. The mean hop lengths for these systems reside far below the upper bound, with no outliers.

In addition, all agent lookup queries are successful. Under a stable Chord system, the Chord finger tables should all point to suitable neighbors to resolve all agents in the system. Since every agent in the baseline RC-Chord experiments is resolved with 100% message delivery over the course of the experiments, this portion of the baseline validation also passes.

### B. RC-Chord Experimental Results

The full effects of churn rate and address width on mean hop length can be seen in Table II, which shows results of experiments of one million agents. The lower-left entry of $m = 20$ and 0% churn rate is close to the baseline Chord

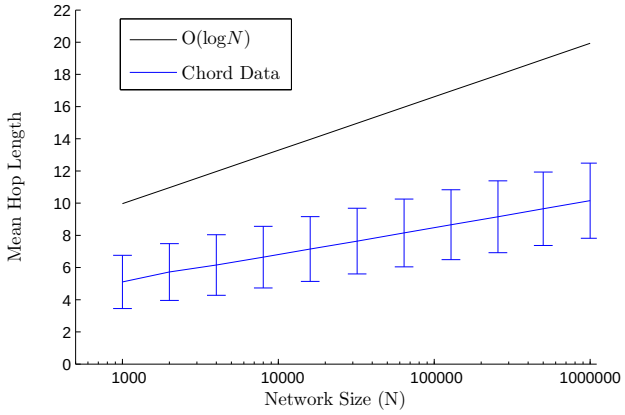


Fig. 4. Chord baseline validation for mean hop length. The plot of $O(\log N)$ is the expected upper bound for the mean hop length in a pure Chord system. The results obtained via experiments fall below the upper bound, partially validating the RC-Chord protocol.

protocol, with a slightly higher mean hop length caused by the business churn model. The higher standard deviation at lower values of $m$ is tied to the HP2P organization at those levels. Lesser address width results in more clusters of smaller size, and more levels in the hierarchy. Intra-cluster communications still maintain the $O(\log N)$ mean hop length, however more inter-cluster communications occur as a result of the increased number of clusters. These inter-cluster communications require messages to jump cluster boundaries, entering into a new cluster, where the intra-cluster $O(\log N)$ mean hop length expectation exists. With $l$ levels in the hierarchy, the longest path in the system is $O(2l\log N)$. However, since only a small percentage of communications follow this path, with most ending at target agents much closer, the hop length standard deviation increases in these situations.

Note that the churn rate provides only a small impact on the mean hop length. This is because the system provides substantial enough routing redundancy to allow dynamic re-routing in failure conditions. This basic stability in the underlying and derivative protocols is a primary objective of this research, and results in reliable performance even in churn conditions.

For any experiments in which more than one cluster must be formed, i.e., for all experiments in which $2^m \geq logN$, the mean hop length and standard deviation is higher than the baseline Chord systems. This occurs because the RC-Chord protocol forces the creation of one or more clusters at different levels. Once established, each cluster of at most $2^m$ agents communicates internally as a stand-alone Chord instance. To perform inter-cluster communications, messages are routed first to a super peer, then across the cluster boundary into the next cluster along the target route. This process repeats until the target is located. As such, the total hop length can reach as high as $O(l\log N)$, where $l$ is the number of levels in the system. The maximum hop length occurs when an agent from a cluster in the lowest level must communicate with another agent in the lowest level of a different resource sub-graph. Those messages follow the route from source to super-cluster,

TABLE II
MEAN HOP LENGTH (STANDARD DEVIATION) BASED ON ADDRESS WIDTH, $m$, AND CHURN RATE FOR A NETWORK OF ONE MILLION AGENTS. BUSINESS AND STATIC CHURNS ARE ENABLED.

| | Static Churn Rate (%) | | | | | | |
|---|---|---|---|---|---|---|---|
| $m$ | 0.0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 1.0 |
| 10 | 13.5 (3.14) | 13.7 (3.24) | 14.1 (3.02) | 14.2 (3.36) | 13.5 (3.02) | 13.6 (2.94) | 13.3 (3.40) |
| 11 | 15.0 (3.37) | 15.8 (3.45) | 14.6 (3.53) | 14.8 (3.48) | 14.4 (3.96) | 14.4 (3.50) | 15.3 (3.93) |
| 12 | 14.7 (3.84) | 14.6 (4.10) | 15.2 (3.94) | 14.6 (4.05) | 14.7 (4.01) | 13.8 (4.00) | 14.3 (4.23) |
| 13 | 13.8 (4.16) | 14.5 (4.49) | 14.8 (4.47) | 13.9 (4.26) | 13.8 (4.47) | 13.9 (4.37) | 15.0 (4.50) |
| 14 | 12.0 (3.37) | 11.9 (3.21) | 12.0 (3.47) | 12.4 (3.49) | 11.1 (3.55) | 11.5 (3.34) | 11.5 (3.41) |
| 15 | 11.0 (3.61) | 11.4 (3.64) | 11.0 (3.57) | 10.2 (3.41) | 11.3 (3.63) | 11.0 (3.56) | 10.7 (4.01) |
| 16 | 10.2 (3.91) | 10.0 (3.70) | 10.1 (3.76) | 10.3 (4.00) | 9.2 (2.80) | 10.0 (3.71) | 10.0 (3.82) |
| 17 | 8.9 (2.76) | 8.7 (2.72) | 8.7 (2.79) | 9.4 (2.81) | 8.5 (2.04) | 8.8 (2.91) | 9.0 (3.01) |
| 18 | 9.2 (2.35) | 9.1 (2.43) | 9.1 (2.35) | 9.1 (2.39) | 8.6 (2.09) | 9.2 (2.39) | 9.3 (2.48) |
| 19 | 9.6 (2.37) | 9.7 (2.38) | 9.6 (2.39) | 9.7 (2.48) | 9.6 (2.53) | 9.7 (2.45) | 9.8 (2.49) |
| 20 | 9.9 (2.25) | 9.9 (2.25) | 9.9 (2.26) | 9.9 (2.24) | 9.9 (2.25) | 9.9 (2.26) | 9.9 (2.28) |

and from super-cluster to target.

Table III shows the agent lookup failure rates for systems of one million agents. Unlike the mean hop length above, the lookup failure rates are significantly impacted by the presence of network churn. Larger address identifier widths generally see lower lookup failure rates due to larger clusters, and the reduced number of inter-cluster links. With more agents in a single cluster, more paths around failed nodes exist, resulting in higher overall agent lookup success rates.

TABLE III
AGENT LOOKUP FAILURE RATES (%) BASED ON ADDRESS WIDTH, $m$, AND CHURN RATE FOR A NETWORK OF ONE MILLION AGENTS. BUSINESS AND STATIC CHURNS ARE ENABLED.

| | Static Churn Rate (%) | | | | | | |
|---|---|---|---|---|---|---|---|
| $m$ | 0.0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 1.0 |
| 10 | 0.07 | 0.78 | 1.58 | 2.57 | 3.54 | 4.75 | 7.14 |
| 11 | 0.07 | 0.96 | 1.54 | 2.50 | 3.13 | 3.94 | 7.16 |
| 12 | 0.17 | 0.75 | 1.49 | 2.26 | 3.39 | 3.87 | 9.95 |
| 13 | 0.09 | 0.78 | 1.32 | 1.72 | 2.20 | 2.89 | 6.65 |
| 14 | 0.08 | 0.42 | 0.88 | 1.34 | 2.34 | 3.96 | 4.99 |
| 15 | 0.02 | 0.77 | 1.47 | 2.11 | 2.73 | 3.40 | 6.46 |
| 16 | 0.27 | 0.70 | 1.37 | 3.19 | 1.49 | 2.98 | 5.69 |
| 17 | 0.00 | 1.71 | 1.64 | 1.82 | 3.01 | 4.33 | 6.20 |
| 18 | 0.00 | 0.54 | 1.16 | 1.74 | 3.80 | 3.31 | 6.36 |
| 19 | 0.00 | 0.69 | 1.04 | 1.88 | 2.77 | 2.67 | 5.45 |
| 20 | 0.00 | 0.97 | 0.94 | 1.49 | 2.01 | 2.68 | 5.01 |

## VI. CONCLUSIONS

The objective of RC-Chord is to establish a flexible and scalable foundation upon which to build large-scale coalition formation algorithms. It builds upon the well tested Chord protocol, incorporating hierarchical structure to organize the system according to resource availability. Testing with multiple churn distributions introduces a small lookup failure rate, but the reliability of the system is maintained. Simulations demonstrate that RC-Chord performs well for systems up to one million agents, and is stable under churn.

The number of resources is currently fixed at configuration time. This significant hurdle can be overcome through the introduction of a suitable mapping strategy, combined with modification of the cluster creation, modification, and destruction algorithms, to accommodate unlimited numbers of resources at runtime.

## REFERENCES


[1] F. E. Bustamante and Y. Qiao. Designing less-structured p2p systems for the expected high churn. *Networking, IEEE/ACM Transactions on*, 16(3):617–627, June 2008.

[2] P. Ganesan, K. Gummadi, and H. Garcia-Molina. Canon in g major: Designing dhts with hierarchical structure. In *ICDCS '04: Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, pages 263–272, Washington, DC, USA, 2004. IEEE Computer Society.

[3] L. Garces-Erice, E. W. Biersack, K. W. Ross, P. A. Felber, and G. Urvoy-Keller. Hierarchical p2p systems. In *Proceedings of ACM/IFIP International Conference on Parallel and Distributed Computing (Euro-Par)*, Klagenfurt, Austria, 2003.

[4] M. Jelasity, A. Montresor, G. P. Jesi, and S. Voulgaris. The peersim simulator. `http://peersim.sf.net`.

[5] J. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *in Proceedings of the 32nd ACM Symposium on Theory of Computing*, pages 163–170, 2000.

[6] H. T. Kung and C.-H. Wu. Hierarchical peer-to-peer networks. Technical Report IIS-TR-02-015, Institute of Information Science, Academia Sinica, Taiwan, April 2001.

[7] J. Li, J. Stribling, R. Morris, and M. F. Kaashoek. Bandwidth-efficient management of dht routing tables. In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, pages 99–114, Berkeley, CA, USA, 2005. USENIX Association.

[8] E. J.-L. Lu, Y.-F. Huang, and S.-C. Lu. Ml-chord: A multi-layered p2p resource sharing model. *Journal of Network and Computer Applications*, In Press, Corrected Proof:–, 2008.

[9] A. I. T. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, pages 329–350, London, UK, 2001. Springer-Verlag.

[10] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.

[11] G. Xue, Y. Jiang, Y. You, and M. Li. A topology-aware hierarchical structured overlay network based on locality sensitive hashing scheme. In *UPGRADE '07: Proceedings of the second workshop on Use of P2P, GRID and agents for the development of content networks*, pages 3–8, New York, NY, USA, 2007. ACM.

[12] B. Y. Zhao, Y. Duan, L. Huang, A. D. Joseph, and J. Kubiatowicz. Brocade: Landmark routing on overlay networks. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 34–44, London, UK, 2002. Springer-Verlag.