Large-Scale Distributed Coalition Formation

DISSERTATION

Daniel Robert Karrels, Captain, USAF

AFIT/DCE/ENG/09-11

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis 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 United States Government.

AFIT/DCE/ENG/09-11

Large-Scale Distributed Coalition Formation

DISSERTATION

Presented to the Faculty
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
In Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy

Daniel Robert Karrels, B.S.C.E., M.S.C.E.
Captain, USAF

September 2009

AFIT/DCE/ENG/09-11

Large-Scale Distributed Coalition Formation

Daniel Robert Karrels, B.S.C.E., M.S.C.E.
Captain, USAF

Approved:

Dr. Gilbert Peterson (Chairman)	date
Dr. Mark Oxley (Member)	date
Dr. Barry Mullins (Member)	date

Accepted:

M. U. Thomas   Date
Dean, Graduate School of Engineering and Management

AFIT/DCE/ENG/09-11

Abstract

The CyberCraft project is an effort to construct a large scale Distributed Multi-Agent System (DMAS) to provide autonomous Cyberspace defense and mission assurance for the United States Department of Defense (DoD). It employs a small but flexible agent structure that is dynamically reconfigurable to accommodate new tasks and policies. This document describes research into developing protocols and algorithms to ensure continued mission execution in a system of one million or more agents, focusing on protocols for coalition formation and Command and Control (C2). It begins by building large-scale routing algorithms for a Hierarchical Peer-to-Peer (HP2P) structured overlay network, called Resource Clustered Chord (RC-Chord). RC-Chord introduces the ability to efficiently locate agents by resources that agents possess. Combined with a task model defined for CyberCraft, this technology feeds into an algorithm that constructs task coalitions in a large-scale DMAS. Experiments reveal the flexibility and effectiveness of these concepts for achieving maximum work throughput in a simulated CyberCraft environment.

Table of Contents

List of Figures

Figure Page

Figure Page

List of Tables

Table Page

List of Abbreviations

AFIT	Air Force Institute of Technology
ANOVA	Analysis of Variance
API	Application Programming Interface
CAN	Content Addressable Network
C2	Command and Control
C3	Command, Control, and Communication
CLOE	Centralized Likelihood of Execution
CNP	Contract Net Protocol
COM-MTDP	Communicative Multi-agent Team Decision Problem
CRP	Centralized Random Policy
DCSP	Distributed Constraint Satisfaction Problem
Dec-POMDP-Com	Decentralized Partially Observable Markov Decision Process with Communication
DHT	Distributed Hash Table
DIB	Defense Industrial Base
DLOE	Distributed Likelihood of Execution
DLP	Distributed Learned Policy
DMAS	Distributed Multi-Agent System
DoD	United States Department of Defense
DoS	Denial of Service
DPS	Distributed Problem Solving
GWoT	Global War on Terrorism
HES	Heterogeneous Search
HP2P	Hierarchical Peer-to-Peer
IPSec	IP Security
LAN	Local Area Network
LOE	Likelihood of Execution
MAS	Multi-Agent System

MBFS	Modified Breadth First Search
ML-Chord	Multi-Level Chord
MRTA	Multi-Robot Task Allocation
MTBF	Mean Time Between Failure
P2P	Peer-to-Peer
pdf	Probability Distribution Function
PKI	Public Key Infrastructure
RC-Chord	Resource Clustered Chord
RPS	Recursive Partitioning Search
SCADA	Supervisory Control and Data Acquisition
SCP	Set Covering Problem
SPP	Set Partitioning Problem
TCP	Transmission Control Protocol
TPP	Total Priority Points
TTL	Time To Live
UAV	Unmanned Aerial Vehicle
WAN	Wide Area Network
VE	Virtual Environment

LARGE-SCALE DISTRIBUTED COALITION FORMATION

I. Introduction

Recent threats to the electronic infrastructures of governments around the world have prompted the United States Department of Defense (DoD) to pursue new technologies to combat cyberterrorism in the Global War on Terrorism (GWoT). In particular, the DoD seeks to find new methods to defend its Defense Industrial Base (DIB), which includes all elements necessary or utilized by organizations within the DoD. This includes, but is not limited to, portions of the Internet (on top of which military medium and high encryption systems exist), DoD intranets and any external systems with whom they interact, Supervisory Control and Data Acquisition (SCADA) systems in power, water, waste water, wireless and radio systems used by military bases and military aircraft, and a host of other subsystems necessary to maintain the function of the United States, both at home and in deployed locations worldwide. These systems combine to form the electronics and communications infrastructure used by the DoD, are therefore subject to attack, and must be defended appropriately.

One of the focused research areas involved in the overall effort to defend the DoD's electronic infrastructure is the development of a Distributed Multi-Agent System (DMAS) composed of autonomous lightweight software and hardware agents used to secure and sustain military networks and attached (including wireless) electronic systems. The goals of the CyberCraft project include:

- monitor systems and respond to runtime variations in near real time;
- enforce current policies;
- provide feedback to human operators on mission relevant status;
- support varying levels of autonomy, depending on situation and commander's intent;

- be dynamically configurable during runtime.

The CyberCraft project is undergoing research in the military and defense contractor industries. It is centered around a lightweight agent, the CyberCraft platform, that receives and executes CyberCraft payloads. The payloads are modules that execute persistently in agent process space, such as system and network sensors and communications modules. The internal framework of the agent combines a three-layer architecture with a multi-staged information flow model, with each stage capable of storing multiple dynamic modules. The three-layer architecture provides a tiered framework for coarse and fine grained planning. The staged information flow paradigm enables a simple framework for communication of information along a path from sensors to global state, to decision and learning components, eventually leveraging hardware and software actuators to affect the environment. Combining these strategies allows payloads to access and develop sophisticated plans and actions within an object oriented framework, simplifying development of third-party payloads. Standard modules support communications capabilities for coordinating with other agents, and actuator modules that report relevant data to human operators.

To operate at the scale that the DoD requires, CyberCraft must function in networks of one million or more agents. One focus area of this project is to develop a communications architecture that can support such a large-scale deployment. Several topologies exist that scale to hundreds of thousands of clients, although little research exists discussing the protocols and algorithms that work well in large-scale environments. The algorithms supporting this architecture must be able to create groups of CyberCraft delineated by mission or geographic location. Therefore, the primary goals of this research are to develop the high level design of the communications architecture, protocol suite, and a strategy for high level command and control of payloads in a network of CyberCraft agents. This includes the development of a large-scale testbed, leveraging the Hierarchical Peer-to-Peer (HP2P) architecture, development of large-scale agent coalition formation techniques, and the integration of these protocols to support a large-scale coalition formation algorithm.

1.1 Definition of Command and Control

Command refers to the ability to assign a task or mission to a set of workers, to include the allocation and deconfliction of resources and schedules. This process is complicated by limited visibility and the perpetual deficit of available resources in a large system. To accomplish this task requires additional thought in scheduling tasks and resources based on priorities and availability, and ensure workers understand the timetable for their resources and task processing.

Control is defined as monitoring of the progress of the tasks and workers for the above command set. In a large-scale distributed system, the challenge is tied to scope, distance, and the costs of communications therein. Controllers that monitor status of a large number of workers or tasks incur a high cost of communications; greater network distance is likewise penalized. The end result is a requirement that controllers have scope of monitoring defined by a multi-variable optimization. This process must occur at the time of task allocation, so as to ensure that the entirety of the mission(s) can be successfully monitored.

1.2 Command, Control, and Coordination in Military Networks

The CyberCraft DMAS will be used to defend existing and future military Command, Control, and Communication (C3) systems. Beyond the Command and Control (C2) aspects of large-scale systems, this research must also provide a mechanism to support coordination of agents. This coordination revolves around building teams of agents capable of receiving and executing tasks at runtime. These teams are necessary to pursue individual mission objectives, and do so by harnessing the distributed processing power of individual agents sequestered into teams. Payloads are then distributed to the agent coalitions. These payloads contain the algorithms necessary to achieve the mission objectives.

In addition, military networks carry with them requirements that separate them from business networks in the private sector. Namely, they:

- operate under tighter security constraints;
- require fault tolerance and self healing;
- may affect human life.

The last attribute of military C3 systems distinguishes them from most other non-medical civilian systems. The capability to affect human life is critical to the mission of the military, and carries an extraordinary responsibility. Although CyberCraft will not have the ability to directly affect human life, they may do so inadvertently through inefficient communications, consuming too much network or processor resources, or failing to protect critical mission systems.

These requirements serve to create a separation between normal business class networks on which a typical DMAS might operate, and the military networks on which the CyberCraft will operate. The considerations found in much of the existing literature on Peer-to-Peer (P2P) (and variant) systems deal with distributed data storage [86] and federated search [63, 64] and do not entirely meet the requirements of the CyberCraft. Specifically, these approaches aim to enable information retrieval, whereas the CyberCraft project requires facilities to search for individual nodes. The distinction is small, but the methods for achieving these differing objectives are quite different. While many structured overlay systems provide communications mechanisms for large-scale systems, the majority presently lack security considerations. In addition, the task allocation and coordination techniques currently available do not scale well enough to support the intended CyberCraft deployment.

Each CyberCraft agent configuration must meet restrictions on the function it performs, and must also meet a code review to establish trustability. The nature and implementation of its processing are to be scrutinized, as well as the resources it utilizes on its host. Thus, creating a small, open source CyberCraft framework, for which trust can be verified via verification and validation by the community, is essential to ensure that CyberCraft will meet requirements to join military networks.

1.2.1 Agent Communications Security. The basis of CyberCraft agent communications security relies upon basic encryption methods. This approach forms a foundation for secure communications, but is extended in production environments by incorporating security modules and additional trust models. This system can be further enhanced for higher security networks by using IP Security (IPSec) to secure lower layers of the communication model, as well as any number of hardware solutions.

Agents in this system communicate directly with one another using Public Key Infrastructure (PKI) algorithms. The DoD is a registered certificate authority, and so the agents in this system inherit directly from the root DoD authority chain. The entirety of the application layer data communication is encrypted, and also uses digital signatures to add an extra redundancy in security and checksums.

Deliberate trust chain models are used to enhance security of the data propagated through the network. This is an active field of research at the Air Force Institute of Technology (AFIT), and many such models are being examined for this application. In addition, key chain management forms a basis for establishing and maintaining a secure network, as well as supporting fault tolerance by allowing agents to reconnect to different parts of the network under certain failure conditions. These topics are outside of the scope of this research effort and are discussed further in future research sections.

1.2.2 Maintenance. Given a constant Mean Time Between Failure (MTBF) [89], the frequency of failures increases with increasing number of nodes in a network. It is therefore important to design into the CyberCraft network architecture a means by which CyberCraft can smoothly transition into and out of the network. Incorporating this functionality into the fundamental design of the architecture will help to ensure that an often overlooked capability operates with impunity in the running system, leading to fewer delays and cascading failures or the introduction of erroneous data. This helps to achieve the end goal of a trusted system.

Much of the existing research into fault tolerance in large scale multi-agent systems is custom tailored to the particulars of the architecture, protocols, and purpose of the applications they support [64,100]. However, scenarios such as rapid joining and parting of agents into and out of the network and leader election have been addressed in previous work [14, 16, 20, 24, 59, 60, 76], and can be adapted to the CyberCraft project.

1.3 Research Focus

The intended initial application of the CyberCraft project is to provide supplementary and autonomous network defense capabilities to DoD networks. This document focuses on building capabilities to support an example network defense scenario. This process should then support the deployment of the package, the construction of groups, analysis of the data, choosing and initiating a response, and more importantly, the C2 capabilities necessary to accomplish these tasks.

This document contributes to the body of existing research of large-scale C2 by addressing the following shortfalls:

- Leveraging existing structured overlay techniques to develop a new strategy which supports the HP2P topology. This structured overlay strategy will be used to provide reliable and scalable communications in a mission oriented system.
- Establishing a task model for CyberCraft agent payloads that facilitates prioritized task execution.
- Constructing a mechanism for building coalitions of agents to accomplish missions within large-scale HP2P systems.

1.3.1 Structured HP2P Overlay. As Chapter II describes, technologies currently exist to support large-scale P2P communications overlays. An overlay is the superposition of a logical topology onto a physical topology. A structured overlay implies a level of organization to the formation of the nodes in that network, and in the case of P2P overlays, it refers to the addressability of nodes in the overlay. Structured P2P overlays support addressing and as little as $O(\log N)$ search times in the network [40, 43, 59, 67, 69, 76, 78, 86, 106], where N is the number of nodes in the network. Current P2P overlays support hundreds of thousands to millions of nodes.

The CyberCraft project and the coalition formation problem benefit from a layering of organizational structures. CyberCraft nodes will be physically separated, which implies an organization which reflects the physical location of the nodes. Relating network closeness to physical closeness is a simple method of performance optimization. This benefit is compounded by the sometimes less reliable or bandwidth constrained communications links such as satellite channels which the military uses to communicate with elements of its networks.

CyberCraft nodes may also be logically organized by mission area. There is no limit to the number and types of missions that CyberCraft may pursue, and the CyberCraft networks can benefit from an organizational structure that is defined by mission or mission area. This helps nodes to communicate relevant data with each other more easily, and concentrates nodes in the network that are focused on a particular area.

The field of multi-agent systems is just beginning to explore the subject of HP2P structured technologies in earnest. Current techniques vary in their applicability to an HP2P network, but elements such as ring geometric routing and identifier addressing can be used in an HP2P system. This document's novel contribution to this area of research centers around constructing self-organized clusters of mission oriented agents in a system which provides routing complexity guarantees in stable HP2P environments.

1.3.2 Constructing Mission Coalitions.

Once reliable large-scale communications are achieved, this document next addresses the challenge of finding groups of nodes that will together work toward a single objective. This assumes that the objective can be accomplished more efficiently by a group of nodes rather than individual nodes – that is, they operate in an environment in which the sum of the whole is greater than the sum of the parts. Described in more detail in Chapter II, the coalition formation problem is formalized with $O(2^N)$ asymptotic complexity, where N is the number of agents in the system. Using a relaxation of the definition of the coalition formation problem, tied with a set of suitable heuristics, the HP2P structure is directly used as a tool to aid in the formation of task coalitions of agents. The HP2P structure lends itself to an organizational grouping, which can be based on mission, goal, location, or other stratification. The creation of a structured HP2P overlay is shown to aid in building tractable online solutions to the coalition formation problem.

This research objective will fully describe the techniques for building a group of agents to accomplish a mission. A mission is defined by a measurable objective, includes a list of necessary resources (bandwidth, processing, power, etc), and may intersect both the cyber and physical domains. Necessary resources for these missions may include capabilities such as a node with high visibility and connectivity, a group of nodes to analyze possible attacks, and a web (or other) server for internal monitoring of the system by operator personnel.

The challenge for this goal is substantial: establishing a coalition of agents to satisfy task requirements in an environment of one million or more agents. The group construction takes place in a large-scale HP2P system. As subsequent discussions will support, the construction of groups in large-scale DMAS is $\mathcal{NP}$-hard [23, 34], and current solutions do not scale to the size of existing P2P structured overlay networks [35, 81, 82, 95]. This document presents methods to ease the constraints on the system based on runtime characteristics and practical optimizations to improve the algorithm execution time, and develops a new coalition formation algorithm that fills the research gap of large-scale agent coalition formation.

1.4 Assumptions

This research assumes an otherwise reliable means of communications, such as Transmission Control Protocol (TCP), in addition to the network assets necessary to facilitate machine to machine communications. Available bandwidth and processing power on machines will be considered as part of the research project, and are therefore not assumed to be infinite or available. However, this effort does assume the existence of at least enough processing power and bandwidth to receive, process, and transmit background and query messages. This is necessary to support the large-scale P2P communication overlays described in Chapter II, as well as to determine if a particular agent is capable of participating in a coalition.

The agents considered are assumed to be cooperative and trustworthy. They can therefore be expected to place value on optimizing global profit over individual profit, and will provide honest best cost estimates for coalition formation cost functions.

1.5 Document Layout

Chapter II describes the current technologies available for structured overlay systems, which are necessary for building addressable communications in a large-scale P2P system. It also introduces the HP2P architecture and the coalition formation and task allocation problems. Chapter III describes the design methodology of the solutions to the large-scale C2 and coalition formation problems presented here. The Resource Clustered Chord (RC-Chord) HP2P structured overlay experiments and results are described in Chapter IV. Testing and analysis of the large-scale coalition formation algorithm, called Distributed Likelihood of Execution (DLOE), is described in Chapter V. Future work and conclusions are presented in Chapter VI.

II. Related Work

This chapter presents a discussion on the current body of research for large-scale Peer-to-Peer (P2P) communications and agent task allocation and coalition formation algorithms. Each of these areas is a necessary and fundamental part of the CyberCraft project and of this research effort. In particular, the issues associated with the scale of the CyberCraft network are at the center of the challenges in this effort, and require a thorough examination before embarking upon building useful and scalable solutions.

This discussion begins by presenting several of the fundamental network topologies and their properties. P2P network structures are then introduced, and presented as solutions to the scaling restrictions on earlier networking technologies. Armed with this understanding, the P2P architectures are built upon to provide coherent and complete communications solutions for large-scale systems through a survey of existing techniques, including their properties and applicability to the CyberCraft project. The young body of research into Hierarchical Peer-to-Peer (HP2P) structured overlays is presented to demonstrate the properties of HP2P systems for large-scale coordination and communication.

Toward building an understanding of the large-scale Command and Control (C2) facilities used by the Distributed Likelihood of Execution (DLOE) algorithm, this chapter concludes with a discussion of the task allocation and coalition formation problems. This includes an introduction to the classifications of robot systems as pertains to solving the coalition formation problem.

2.1 Topologies

With a goal of one million or more CyberCraft agents residing on the same network, issues of scale become a dominating factor in the design of a suitable communications architecture and protocol suite to satisfy the CyberCraft project requirements. There are few networks in existence today that accommodate so many nodes. Three of them, the Internet, GNUTella, and KaZaA, are organized differently, resulting in different characteristics and performance[1]. The Internet follows a topology governed by several power-law relationships, GNUTella employs a P2P architecture, and KaZaA uses an HP2P architecture.

Many of the existing multi-agent architectures fall into one of five categories: hub-based, P2P, power-law, multi-layer, and hybrid. The original sources and motivations for each architecture vary greatly, as do their applications. What has developed in recent years is a technology evolution between the creators and proprietors of malicious networks of agents (often referred to as botnets [30]) that exist to perpetrate activities such as Denial of Service (DoS), spamming, identity theft, and a great many other legal and illegal activities, and the authorities who seek to halt the malicious and illegal activities that both harm the economic and social infrastructures of nations worldwide, as well as consume enormous amounts of network resources. This competition has sparked further advances in large scale communications systems.

Research into large scale Distributed Multi-Agent Systems (DMASs) is still in its infancy [92]. Here, scalability refers to the increase in the number of agent nodes in a network with relation to the overhead associated with maintaining and utilizing that network. Much of the previous work addresses agent networks of up to several hundred agents, with several systems reaching a thousand or more agents.

Most modern operating systems do not support more than a few thousand simultaneous connections. It is possible to create a spanning tree of such servers to support several hundred thousand clients, but the burden on those machines becomes large, and fault recovery is difficult. The current solution to this problem is to build networks based on a P2P topology. These systems come in several flavors, and attempt to distribute the computational and network loads. Moreover, they are resilient against faults, and minimize processing load at each node.

[1] It is believed that several malware botnets (discussed later in this document) have eclipsed 100,000 nodes, but verifying this is difficult.

The first P2P systems were unstructured, or lacking discrete organizational rules. They relied upon broadcast mechanisms to locate data items, and were characteristically bandwidth intensive. These systems suffer from scalability constraints, although persistent efforts to improve their technology has resulted in continued relevance. Systems such as Gnutella [7] and Freenet [21] continue to evolve, incorporating lower network diameter, caching, and other techniques to improve scalability and flexibility.

Successive generations of P2P networks introduced more structure to the system, initially focusing on content storage, and subsequently on node organization. Content structured systems are concerned with the efficient representation and retrieval of information, and provide structured protocols to identify and locate data items. These systems are often called libraries, and index data by unique key, subject area, or range queries. This technology serves as the reinforcing means for unstructured systems to continue their life cycles, as well as leading into a more general class of P2P networks called P2P structured overlays.

Structured overlay designs improve structured content storage systems through a less drastic but important mutation: the nodes themselves are now uniquely identified, instead of simply the data they store. The underlying structure is based on a Distributed Hash Table (DHT), which is a dictionary based approach to storing and retrieving information. The nodes in communications structured P2P systems can be located through this process, thus making the network itself a DHT, regardless of the data stored in the network. We will therefore use the terms communications structured P2P system and DHT interchangeably, as they differ only in application. Early structured overlays were flat, using a single layer of peers, however recent examination of these technologies has extended to include hierarchical formations of structured overlays to increase scalability and flexibility. This generation of P2P systems, both flat and hierarchical, is the focus area of this document.

Whether hierarchical or flat, the research team envisions the continued pursuit of P2P systems as foundations for large-scale distributed application development. Generalized P2P application frameworks provide the necessary network foundations upon which to pursue this goal. Much in the way that early routing and network communications were decoupled to form large networks of heterogeneous applications, such as the Internet, modern P2P systems are moving toward providing large-scale networking functions upon which to build generic systems. This (mostly) transparent layer provides a network Application Programming Interface (API) for application software to use, and the scope, method, and purpose of those applications are unbounded. This design decouples the implementation of communications from the application that uses it, allowing more flexibility and sustainability. Such P2P communications overlays can thus be interchanged with minimal effort to provide differing capabilities to the application layer.

This chapter examines P2P technologies as the organizational component in large-scale DMAS applications. A primary trait of these Multi-Agent System (MAS) is the need for C2. C2, in an electronic system, includes the methods used to organize and communicate with nodes in a distributed system. The ability to conduct useful C2 is highly dependent on the structures and protocols used to organize and maintain these systems. Inefficient routing protocols, for example, result in lower performance for application level C2. As a first step toward developing a large-scale C2 capable P2P overlay, this paper examines current methods for organizing networks to establish reliable large-scale communications using P2P systems.

Section 2.2 presents a brief definition of C2, followed by three brief case studies to highlight several of the key features necessary for large-scale C2. The discussion then introduces P2P systems, followed by the survey taxonomy and analysis criteria. The analysis begins with a discussion of the early generations of P2P networks in Section 2.7, including unstructured and content retrieval P2P systems. The focus area of this chapter is the discussion of communications structured P2P systems beginning in Section 2.9, continuing with flat and hierarchical structured P2P systems. The discussion of current large-scale peer-based large-scale structured overlays concludes with a presentation of the design trade-offs between the varying P2P approaches. To further motivate the need for reliable large-scale communications, this chapter concludes with an introduction to large-scale coordination in Section 2.14, including a discussion of the coalition formation and task allocation problems.

2.2 Command and Control

Command refers to the ability to issue runtime orders to a subset of all nodes in a network. Specifically, it:

- Provides the capability to assign tasks or missions to one or more agents.
- Schedules tasks to run in a manner that avoids contention and deadlock, and optimizes performance.
- Permits the allocation of resources.
- Autonomously deconflicts resources at runtime, which may involve distributed agreement.

Command, in general, involves many of the same functions as are found in the coalition formation problem [34]. The coalition formation problem is considered to be $\mathcal{NP}$-complete, and so building a large-scale solution requires new thinking about an old problem.

The ability to control a set of agents involves capturing their runtime state, and proactively and reactively responding to changing conditions, to include the introduction of new requirements or constraints. At its essence, control refers to:

- Monitoring the progress of tasks at runtime.
- Identifying and resolving runtime conflicts.
- Feeding updated system state into the task scheduler.

Control solutions are built upon reliable and efficient communications mechanisms, and are intended to maximize efficiency of the runtime system by discovering and correcting runtime difficulties. The combination of efficient and reliable network structures and communications mechanisms creates an environment in which this can succeed.

2.3 DMAS Case Studies

Of the many applications of DMASs, we have chosen three that serve to motivate the need for a comprehensive C2 policy framework. These applications rely on scalable, flexible, and reliable communications mechanisms. We believe that P2P architectures will meet these requirements. These projects are not meant to enumerate all scenarios in which C2 is necessary, but only to provide through example a basic understanding of why a scalable C2 strategy is necessary.

2.3.1 Electric Elves. The Electric Elves project [19] is an initiative to create digital advocates for the intentions of human members of an organization. The Elves coordinate amongst each other to schedule meetings and presentations, order lunch, make and cancel appointments, monitor project status, and provide information about the person they represent (such as location or preference). The project is built upon a heterogeneous DMAS, where each agent is capable of representing the interests and goals of an organizational member (whether human or otherwise). It is a novel team-based system, combining adversarial and cooperative strategies, that must adapt to changing scenarios, and whose members must constantly interact with other Elves to coordinate an optimal self-interested schedule.

The project spans heterogeneous MASs, distributed cooperative and adversarial coordination, multi-objective optimization, adjustable autonomy, and human interaction. The authors of Electric Elves intend their application to scale up to large-scale organizations, where the agents run continuously for weeks or months to optimize the daily schedules of its members. Such an initiative accentuates the need for distributed

C2. In particular, this project requires the scheduling of tasks, runtime allocation and deconfliction of resources (physical in this case), and monitoring the progress of tasks to identify and resolve conflicts.

2.3.2 Virtual Environments. Virtual Environments (VEs) [85] refer to systems in which actors interact with one another and the environment in the pursuit of some set of goals. This evolving field is gaining momentum in many application areas, including large-scale online gaming, education, design in the engineering industries, interactive communications, and many others [12]. In many scenarios, humans enter into a VE to consume or provide services to other actors and the environment. The other actors may themselves be representations of humans or of software or hardware agents. The agents serve many purposes, to include providing fundamental services for the users of the environment. The agents must coordinate with each other to achieve varying goals, and the environment information for each agent may be incomplete. In particular, online games may involve many thousands of agents who dynamically form teams to interact with players in different parts of the VE.

These environments provide a challenging domain in which agents must operate, and highlight the need for a comprehensive C2 strategy. Agents in a VE must coordinate in real-time to form adversarial or cooperative teams and coordinate and schedule services. The introduction of human actors creates a myriad of unknown scenarios to which the agents must respond. In terms of C2, VEs provide a compelling opportunity: these systems may be centralized, decentralized, or dynamically choose the better alternative for a given scenario. Such heterogeneity requires a robust and well designed C2 architecture to respond to the evolving landscapes found in VEs.

2.3.3 Network Routing. One of the earlier applications for DMASs is modifying network routes to reduce bottlenecks tied to increased traffic or hardware or software failures [87]. These systems use networks of agents to monitor network traffic conditions at key points, and modify the routes in realtime to correct problems or provide differing levels of service. The agents must coordinate these actions so as to avoid livelock[2]. The agents must coordinate their efforts to maximize overall system performance, which can be challenging in high traffic situations, where the communications between agents may be delayed.

This application requires a great deal of cooperative teaming to maximize system efficiency. Due to traffic conditions, the agents must communicate with a robust language, and use a reliable communications framework. The agents perform realtime monitoring of the system, and must cooperatively schedule the use of resources. In addition, this application requires a suitable security strategy, and poor decisions may have a noticeable impact to many users.

Independently, these applications define the need for specific and challenging capabilities in distributed systems. However, collectively, they define a subset of the principles of C2. Key among the current and emerging requirements for such applications is scalability. P2P overlays provide the scalable communications needed. P2P overlays do not currently provide the explicit control semantics to employ coalition formation, task allocation and scheduling, or resource distribution algorithms in support of these applications as they increase in scale and complexity. While this genericity provides flexibility, it is necessary to consider the set of C2 principles that build upon large-scale P2P overlays to construct more sophisticated and feature rich applications.

2.4 Large Scale Peer-to-Peer Communications Overlays

A communications overlay is the set of protocols and algorithms necessary to build and maintain a topology of nodes in such a way as to guarantee a set of performance parameters. This model can then be used as a basis for communications by applications. In the context of existing P2P technologies, these overlay structures

[2] Livelock in a dynamic routing management system can occur when one agent modifies a route to redirect traffic to another section of the network. If the agent responsible for the targeted portion of the network is unaware of the change, it may reverse the effect by redirecting traffic back through the previous route. This process can occur very quickly with software agents, thus leading to a deadlock of live nodes.

describe the formation of nodes into a system of peers capable of identifying and locating remote nodes without foreknowledge of their exact location or even their existence. Such a consideration is necessary in many systems where the scale of the system is large enough to preclude the possibility of global knowledge. First generation systems solved this problem by query broadcast, but this solution failed to scale. Newer systems have developed more advanced techniques for locating remote nodes, and the utility of such systems has brought about the emergence of P2P networking to the domain of mainstream applications.

Large-scale systems refer to those that support several hundred thousand or more simultaneous nodes. Such systems are in use today by corporations, the military, criminal elements, research institutions, and others. Each field of applications has differing requirements, and short of creating new technologies, application designers must choose an existing approach (or combination of approaches) to fulfill their requirements. This serves as the motivation for our discussion of the current body of research into P2P overlay technologies.

2.4.1 Peer-to-Peer Networks. This analysis examines the use of P2P networks to accomplish distributed C2. Systems based on hierarchical or client/server paradigms fail immediately due to lack of scalability, and are omitted from this presentation. Although much of the related work presented here refers to P2P systems, the protocols themselves are functional on HP2P systems.

A P2P network is one in which nodes interconnect to each other, typically with out-degree greater than one. This means that a node may connect to multiple other nodes. There is no distinction between service and client nodes. Rather, nodes are considered equal, and any of them may provide services to the network, to include routing services. An example P2P network is shown in Figure 2.1

A HP2P architecture, as shown in Figure 2.2, places additional organizational constraints on a P2P network – it segments nodes into clusters (groups). Each cluster is a smaller P2P network itself, connected to the rest of the network through one or

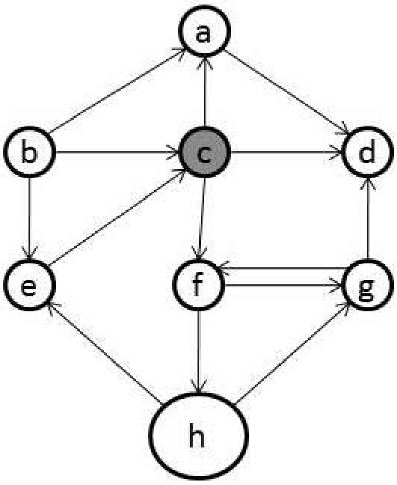

Figure 2.1: An example P2P network with eight nodes. The arrows represent a set of transactions at a given time, but the links are considered to be bidirectional.

more super peers. Super peers act as routing hubs, and provide convenient points for additional application specific processing. Groups of super peers can be connected to create clusters of (super) peers, to which a subset are promoted to higher level super peers. This layered (hierarchical) structure can be applied repeatedly to achieve design goals [23, 100].

2.4.2 Small World. Stanley Milgram's small-world phenomenon [70] is based on the sociological observation that most people can be creatively linked by a short chain of acquaintances. Early experiments demonstrated that letters could be routed to arbitrary destinations by traveling through the hands of kind volunteers, with the restriction that each person along the chain was already on a first-name basis with the next individual (thus preventing ambitious individuals from traveling cross country to reach the destination). This result and the idea of the small-world phenomenon was applied to computer networking by John Kleinberg [54]. It has continued to serve as inspiration to many P2P networking protocols, by authors' recognition that digital messages could be likewise transmitted between members of a computer network without requiring extensive network planning or global view of

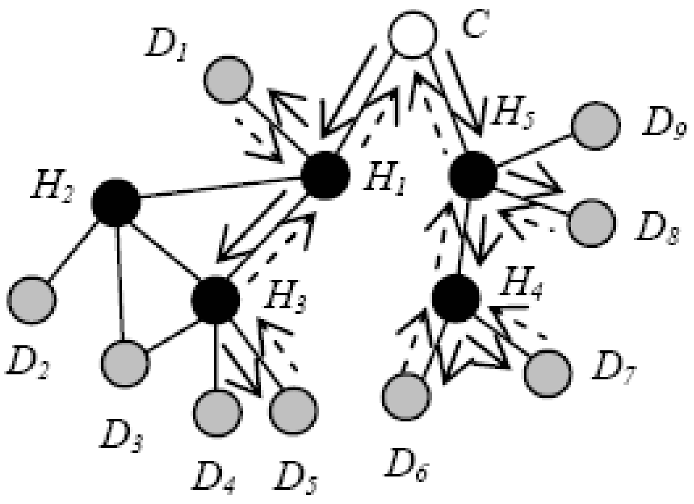

Figure 2.2: An example two-layer HP2P network [1]. Each of the five virtual nodes in the network is a separate P2P network, connected by two super peers to other virtual nodes. As in a simple P2P system, each peer (cluster) is able to connect to other peers (clusters).

the network. This is a founding principle of P2P design, and many of the approaches here employ this idea in their designs and discussions.

2.5 A Taxonomy of Command and Control Principles in Large-Scale Multi-Agent Systems

This taxonomy is adapted from the work of Cao [17] and Dudek [27] who analyzed the characteristics of multi-robot systems. In addition, this paper builds on the work of Lua, et. al. [65], who provide an early survey of P2P structured and unstructured systems. We extend their work to include more recent developments in the field of P2P overlays with a focus on application to large-scale C2.

Note that the characteristic of differentiation is not considered here (homogeneous versus heterogeneous systems). This is because the systems described are used to generate and maintain overlay networks, on top of which a given application may reside. In particular, systems such as Pastry [78] and Tapestry [106] have a built-in API to provide explicit support for applications. These applications use the P2P system as a middleware to provide replication, networking, data storage, etc. The

P2P system is independent of the application residing on it, and therefore the system differentiation is instead a property of the application rather than the P2P system.

2.5.1 Topology. It is necessary to distinguish between two classes of P2P networks: structured versus unstructured. Structured P2P networks enforce a rigid set of rules on the topology and location of nodes and perhaps data in the system. The advantage of this design is that it provides more efficient routing. Unfortunately, structured networks may lack resilience in networks where nodes are transient [40]. It is possible to loosen the location constraints – rather than enforcing absolute structure, the system can provide "hints" about where nodes and data are placed.

An unstructured P2P topology is one in which node and data locations are not enforced. Nodes are free to join the network based on an unrestrictive set of rules. This design is useful for situations where the frequency of nodes joining and leaving the network is high. However, searching generally consists of flooding the network, which does not scale well. In general, pure flooding approaches do not assign unique identifiers to nodes. This greatly complicates attempts to provide routing protocols in unstructured networks. These networks tend to be more focused on content storage and retrieval rather than providing a communications substrate for large scale systems. Unstructured systems such as Gnutella [29], Freenet [21], and BitTorrent[3] [22], are still in use and provide a unique set of properties useful for content distribution. However, this study focuses on structured systems to provide solutions to large scale multi-agent application requirements, and only discusses the properties of unstructured systems as a means of comparing or unique ideas.

The qualitative measures of a model's topology are given by one of the following three definitions:

[3] Arguments persist about the true nature of BitTorrent's topology. For the purposes of this document, we consider BitTorrent to be a combination of structured and unstructured ideas.

Table 2.1: Structured Overlay Topologies By Organization Rigidity

Structure	Overlay Approaches
Strongly Structured	Koorde
Structured	Accordion, Bamboo, Butterfly, CAN, DKS, Mercury, Pastry, P-Grid, SkipNet, Tapestry, Viceroy
Loosely Structured	Chord, Freenet, Kademlia, Kelips, One-Hop, Symphony, Two-Hop

- **Strongly Structured** - The topology of the network is rigidly enforced, and is inherently inflexible. This is an undesirable trait for a high churn network, as maintenance actions dominate processing and bandwidth consumption.
- **Structured** - An approach which requires nodes in a network to be identified and ordered in a manner that is consistent with query protocols. This topology has weaker requirements than a strongly structured overlay, and typically requires only that keys for nodes be uniformly distributed across the network.
- **Loosely Structured** - Requires that nodes be identified uniquely in the network, but few, if any, other constraints are imposed. This increases flexibility for nodes joining the network, at the expense of higher maintenance costs.

Large-scale systems are generally expected to have a high churn rate [16]. Strongly structured systems offer the benefit of more strictly assigned node locations, but at the expense of flexibility. More loosely structured solutions exploit their polymorphic nature to adapt to changing network conditions, but at the expense of higher maintenance costs. Table 2.1 introduces the overlay networks discussed in this paper, categorized by their topological strictness.

2.5.2 Routing Geometry. Many of the innovative ideas that separate the approaches to P2P communications lie in the method of building routes between nodes. The routing methods are delineated by the geometry formed by combining the node addressing with the routing scheme for each overlay. Many of these approaches can then be visualized as a fundamental computer science data structure. This is a convenient metric for describing a system's function, and for analyzing its performance [40, 61].

- **Hypercube** - A geometric structure, derived from a square, with dimension (typically) greater than two. These structures are used to represent planes in which segments of a P2P network reside. For example, routing may rely upon a Cartesian coordinate system, where moving from source to destination can be visualized as moving between quadrants along planes in the n-dimensional node identifier space.
- **Ring** - A circular structure used to store references to nodes in neighbor or routing tables. Finding the next hop in the route usually involves finding the nearest identifier in the ring using a similarity measure (such as modulo arithmetic). References around the ring generally divide the identifier space evenly to provide best "guesses" about which direction to choose.
- **Skip List** - Arrays of linked lists that point to nodes in the network node identifier space based on level. The nearest level nodes are close in the identifier space, whereas higher (or semantically lower) level nodes point further into the identifier space. Choosing a level for the next jump generally depends on the distance to the destination identifier.
- **Butterfly** - A network of log N stages, where N is the number of nodes in the network, and nodes at stage i interpret the i^{th} bit of the routing address to choose the routing node in the next stage. This differs from a standard search tree in that nodes each have two inbound and two outbound links, and a butterfly network generally does not have a single root node.
- **Linear** - Neighbors and next hop routing nodes are stored in a linear structure, such as an array.
- **Tree** - Refers to a standard tree data structure. The branching factor or node outdegree (logarithmic base) is specified where appropriate.

- **Adaptive Linear** - An improvement on flat linear routing introduced by Freenet [21] in which key nearness is used as an initial attempt to locate content. Upon successful queries, the involved nodes record the key and destination node information about the query for later transactions.
- **de Bruijn Graph** - A directed graph whose nodes are addressed by ordered proper prefix. The graph has b^m vertices, where b is the address base, and m is the width of the address.

2.5.3 Query Path Length. A critical feature of all overlay protocols is the expected number of hops to route a message from source to destination. Basic analysis shows that most overlay structures achieve $O(\log N)$ optimal path length, where N is the total number of nodes in the system. When considering different overlay strategies, it is important to examine this measure in the context of the other features an overlay provides. For example, Kademlia supports $O(\log N)$ path length, but does so even in environments with high rates of hardware and software failures, or those with high churn rate. The One-Hop protocol supports $O(1)$ path lengths, but at the expense of maintaining $O(N)$ neighbors. A low path length is desirable for performance reasons, but achieving high performance typically incurs a compromise in one or more other features [96].

2.5.4 Node Memory. A primary tradeoff space in large scale P2P systems is the compromise between query path length and the amount of memory used at each node. These two attributes are generally inversely proportional, and the methods of compromise, and their applications, form a primary differentiating factor for the techniques discussed in this paper.

Note that Table 2.2 includes ranges for the query path length and node memory for several overlays. This indicates that the approach either has several different techniques to choose from based on design considerations, or autonomously varies its strategy based on runtime parameters.

Table 2.2: Large-Scale P2P Multi-Agent System Structured Overlay Characteristics

Name	Routing Geometry	Query Path Length	Node Memory	Addressing	Scalability	Bandwidth
Accordion [59]	Ring	$O(1)$ - $O(\log N)$	$O(1)$ - $O(\log N)$	Consistent Hashing	High	Low
Bamboo [76]	Ring	$O(\log N)$	$O(\log N)$	Uniformly Distributed	High	Low
Butterfly [28]	Butterfly	$O(\log N)$	$O(\log^2 N)$	Uniformly Distributed	High	Low
CAN [75]	Hypercube	$O(d \sqrt[d]{N})$	$O(2d)$	Cartesian Zones	High	Moderate
Chord [86]	Ring	$O(\log N)$	$O(\log N)$	Consistent Hashing	High	Low
DKS [8]	Tree	$O(\log_k N)$	$O(\log N)$	Unique	Low	Low
Kademlia [69]	Tree	$O(\log N)$	$O((2^b - 1)\log_{2^b} N)$	Uniformly Distributed	High	Low
Kelips [42]	Linear	$O(1)$	$O(\sqrt{N})$	Consistent Hashing	Moderate	Moderate
Koorde [47]	de Bruijn	$O(\log N/\log\log N)$ - $O(\log N)$	$O(1)$ - $O(\log N)$	Consistent Hashing	Low	High
Mercury [13]	Ring	$O(\log^2 N/k)$	$O(\log N)$	Uniform Random Sampling	Moderate	Moderate
One-Hop [41]	Linear	$O(1)$	$O(N)$	Unique	Low	High
P-Grid [5]	Tree	$O(\log N)$	$O(\log D)$	Uniformly Distributed	Moderate	Moderate
Pastry [78]	Ring	$O(\log N)$	$O(\log N)$	Uniformly Distributed	High	Low
SkipNet [43]	Skip List	$O(\log N)$	$O(\log N)$	Skip Lists	Moderate	High
Symphony [68]	Ring	$O(\frac{1}{k}\ log^2 N)$	$O(2k + 2)$	Uniformly Distributed	High	Low
Tapestry [106]	Ring	$O(\log N)$	$O(\log N)$	Uniformly Distributed	High	Low
Two-Hop [41]	Linear	$O(1)$	$O(N/k)$	Unique	Moderate	Moderate
Viceroy [67]	Ring & Butterfly	$O(\log N)$	$O(\log N)$	Consistent Hashing	High	Low

2.5.5 Node Addressing. Addressing refers to the assignment of an identifier to each node in a system. In most systems, the address of each node is unique. However, some systems assign nodes randomly, whereas others use a distributed algorithm to ensure uniqueness and other properties. The addressing directly supports the algorithms used to locate nodes in the network. Note that most systems use a form of hashed addressing for identifying content [49], and this subject it outside of the scope of our discussion. For that reason, we consider only requirements for node addressing here.

- **Consistent Hashing** - Creates unique addresses based on the hash value of one or more properties of the node. These properties generally include host name or IP address, as well as a salt. This is a distributed algorithm that all nodes in the system follow to generate addresses that ensure some global property, such as uniform distribution of addresses.
- **Uniformly Distributed** - The property of node identifiers being distributed across the network according to a uniform random distribution. Routing algorithms use this property to ensure that each node's routing table has a diverse sampling of the identifier subspaces that exist in the overall identifier space, which is useful for choosing the next hop in a route without full knowledge of the network. Uniform distribution is a property, whereas consistent hashing is a mechanism to ensure that property. Systems that identify consistent hashing as part of the protocol are considered to have stronger semantics, although requiring only uniform distribution of identifiers is more flexible (it permits the use of different algorithms).
- **Pseudo-Random/Signature** - Node identifiers are pseudo-random, each generated using an independent random number generator. This salt is combined with other elements of the node's properties to generate a (hopefully) unique identifier for that node.

- **Cartesian Zones** - Nodes are identified by Cartesian coordinates, and separated into zones in a multi-dimensional Cartesian coordinate space. Routing generally takes place by identifying the source and destination coordinates, and routing from zone to zone along a line connecting the two end points.
- **Ordered Proper Prefix** - A method of addressing used to support deterministic routing of messages. Each node is assigned a unique identifier within a fixed width space. The nodes are connected to each other in such as way that moving from node to node follows an ordered prefix of the final destination node's identifier. This is similar to how search is conducted in a trie [45].
- **Skip List** - An approach that divides nodes into routing zones based on the levels used in skip lists. Each node is uniquely identified, and stored in one or more skip lists. Level jumps in the skip lists differentiate identifier subspaces, and are used to expedite routing queries.
- **Unique** - A loose constraint, requiring only that nodes in a system are identified uniquely.

2.5.6 Scalability. Large-scale systems are those that may reach or exceed several hundred thousand to a million nodes. This size requirement inflicts a toll on both the topology design as well as C2 techniques used to interface with the agents. It may be reasonable to employ a supervised cooperation design at the level of a single cluster in a HP2P network, where the super peer is responsible for coordinating its cluster's agents. However, a more resilient and scalable solution would be necessary to support the C2 of the cluster super peers themselves. Therefore, we generally desire a decentralized command approach over a supervised strategy.

Isoefficiency [56] is defined as the rate at which the problem size must increase with respect to the number of processing elements to keep the efficiency fixed. Here, we will use the term scalability to refer to an adaptation of isoefficiency: the rate at which efficiency decreases as the size of the network increases. For the purposes of C2, the efficiency of node discovery and single and group messaging is considered.

In addition, the amount of memory storage per node is included in this analysis. Most systems discussed use an amount of memory per node that is logarithmic in the number of nodes in the system. We explicitly describe cases where memory usage is unique or otherwise differs notably from comparable norms.

Scalability is qualitatively described here as High, Medium, and Low, where High scalability refers to systems that are most resilient to increases in size. As a goal of this paper, we desire highly scalable architectures upon which application level solutions can be developed.

- **Highly Scalable** - The amount of work necessary to manage and use the system increases linearly (or better) with the size of the system.
- **Moderately Scalable** - Work required to use and manage the system increases in a small but non-linear fashion (with a positive acceleration) with respect to the size of the system. These approaches support systems up to a point, but fail on medium sized networks (tens of thousands of nodes, for example).
- **Poorly Scalable** - The system fails to function beyond tens or hundreds of nodes. This failure can manifest itself as lost packets, misunderstood communications as a result of high latency, or failure of one or more nodes due to large amounts of bandwidth or number of connections imposed upon them.

2.5.7 Bandwidth Consumption. The amount of bandwidth available to a multi-agent system is generally far greater than for a system of robots, but the use of that bandwidth can still incur a cost to the functioning of the system and its missions. C2 strategies must therefore be careful to ensure no unnecessary bandwidth is consumed, as it can have a significant impact in a large-scale system. Many of the P2P strategies considered here sacrifice bandwidth consumption to achieve better message routing constraints (and some vice versa). Bandwidth will be described as High, Medium, and Low, with High bandwidth systems requiring the most bandwidth for search or maintenance activities. Low bandwidth systems are most desirable, but must be considered alongside search efficiency.

This metric is, in general, related to scalability. However, while it is true that scalable solutions tend to have low bandwidth consumption for maintenance functions, it is not necessarily true that poorly scalable solutions use large amounts of bandwidth. The distinction lies primarily in the topology organization. Therefore, we present this metric to help distinguish the reasons for scalability, and as an additional means for evaluating scalable solutions. We prefer low bandwidth consumption, even though it may need to be compromised to gain stronger guarantees on routing complexity. It may therefore be necessary to accept a medium bandwidth system, although very rarely will a high bandwidth system be justified for a large scale system.

- **High** - The system uses large amounts of bandwidth to maintain and organize its structure.
- **Medium** - A qualitatively modest amount of bandwidth is necessary to operate the network routing and support structures.
- **Low** - Given the properties of the system, little bandwidth is used to maintain its structure.

2.5.8 Reconfigurability. Deployed P2P networks tend to have a high churn rate [10], where agents may join and part unexpectedly, and with high frequency. A high churn rate is a result of events such as users turning off their machines unexpectedly, unreliable communications links, etc. Large-scale C2 systems must therefore be resilient to the loss of productive agents and must be able to restructure task allocations to ensure mission progress.

As relates to distributed C2, the group architecture, whatever its form, must support efficient search and broadcast. This is most commonly observed in agent systems organized into structured or loosely structured topologies. Reconfigurability directly affects, and is affected by, the properties necessary to maintain a system's routing and search complexities. This in turn affects the level of bandwidth consumption for maintenance activities. Reconfigurability is also, in general, directly related to bandwidth consumption and strictness of topology. As it impacts and is affected by many other properties, and is a derived metric, reconfigurability is omitted from Table 2.2, and instead is discussed inline, where appropriate.

2.6 Peer-to-Peer Communications Characteristics

The following sections discuss the available structured P2P and HP2P overlay protocols currently available. The emphasis is on distinguishing the unique approaches of the above design taxonomy, and how they impact the properties of a large-scale P2P system. The discussion is structured in parallel to the stages of P2P evolution presented above, beginning with an introduction to message broadcast in unstructured large-scale P2P systems. Although the focus of this discussion is the technology behind structured P2P systems, it is instructive to first briefly introduce the founding ideas and techniques used to build the first popular unstructured P2P systems. In particular, many of the problems found in early systems are cleverly solved in the design phase of subsequent systems, and many of the early solutions are retained and applied to similar problems in later systems.

2.7 Unstructured Peer-to-Peer Systems

A defining feature of the first generation of large-scale P2P networks is the method of broadcast. Systems such as Gnutella [7] use a full (one-to-all) broadcast at each step of a message query, whereas nodes in a Freenet [21] network select only a subset of neighbors to which to send message queries. One result of the research efforts for the first generation of P2P systems is the need for efficient and duplicate free broadcast. A primary difficulty is that P2P networks can become large enough that maintaining a global view of the network becomes impossible due to limited system resources. Nodes in these networks may have little or no information about the roles or locations of other nodes in the network, yet they may still need to communicate. It is therefore necessary to develop routing protocols for large-scale P2P networks that permit one-to-one and one-to-many communications.

Perhaps the most obvious approach to message broadcast for P2P networks is the flood-fill algorithm. Message flooding involves a node sending a query to all of its neighbors, who in turn send to all of their neighbors, repeating until all nodes have (hopefully) been queried. This system is inefficient, creating many duplicate messages, requiring a duplication ratio of about 80% to achieve a 90% success rate [90]. Still, some systems such as Gnutella [77, 102] have gained popularity even while using this approach.

An improvement to the basic flooding protocol is the Modified Breadth First Search (MBFS) [48], a gossip algorithm [24]. This is a slight improvement over blind flooding, and uses a probabilistic approach. When a node receives a query, rather than forwarding it to all of its neighbors (assuming it does not have the information locally), it sends the message only to a randomly chosen portion of its neighbors. While this reduces total network traffic for a given search, it is still probabilistic and provides no guarantees that all nodes will be visited, or of duplication constraints. A slight improvement to the MBFS is the Random Walk [66]. This approach chooses at most K neighbors at each hop, but incorporates a Time To Live (TTL) parameter. This introduces the constraint that at most $K * TTL$ messages will be sent to the system, but does not guarantee that a message will reach all nodes.

A further refinement of the flooding approach is called Efa [91]. In this approach, each node maintains information about its neighbors up to several (two) hops away. The algorithm then employs several set operations to determine possible overlapping lines of communication that might occur in a broadcast to any of its neighbors. While this approach achieves improvement over the standard message broadcast, the core of its design relies upon a heuristic decision engine that ”anticipates” the actions of other nodes, without actually establishing a coordination agreement ahead of time. In this respect, the algorithm must guess if another node will send a message forward, and that other node is likewise anticipating about the first node. There exists no determinism or guarantee of delivery to all nodes.

CAP [55] takes a step toward solving this broadcast problem by incorporating several structural rules, and extends the Gnutella protocol to incorporate locality sensitive clustering. The idea is that nodes that share lower common latency are to be matched and organized into clusters, so as to minimize overall search times. The organization is performed by a centralized cluster server that is responsible for matching nodes to clusters, extending the idea of early Napster [80] implementations. In creating clusters, the authors assign additional responsibilities to certain nodes in each cluster, calling them delegate nodes. These delegate nodes are later named super peers or super nodes, and appear in the popular file sharing system KaZaa [57].

SCAMP [32] incorporates a membership service into unstructured P2P systems. It provides nodes with a partial view of the network using a probabilistic subscription protocol. Broadcast is handled by each node forwarding a message to $\log N + c$ of its neighbors, where N is the number of nodes in the network, and c is a small constant. This establishes a high probability ($-e^{-c}$) of all nodes in the system receiving the message.

BAR [58] extends the gossip based broadcast mechanism to include a deterministic routing function. Instead of using probabilistic neighbor selection in message forwarding, BAR uses a pseudo-random number generator combined with unique signatures to choose neighbors. This scheme also provides rudimentary security through Public Key Infrastructure (PKI) encryption.

Freenet [21] acts as an anonymous distributed file system by sharing the persistent storage mechanisms of its users' machines. It is a hybrid approach in that it uses flooding to locate data items, however it does optimize searches by replicating popular data along frequently searched paths. Thus, Freenet could be considered either unstructured or content structured.

In the Freenet protocol, each file is given multiple hashed keys, used to locate and asymmetrically encrypt the file. This encryption is the source of Freenet's anonymity: due to the difficulty of examining the raw data, each user in the system can maintain plausible deniability about the documents being stored locally. Freenet makes extensive use of caching. When a file is retrieved from a remote node, a copy of that file is then stored on the local machine, as well as at the nodes along the route between the source and destination nodes. This replication then improves the performance of queries for commonly sought files. It will also remove all files that the network considers uninteresting due to attrition: storage areas become filled, and least recently used data is purged.

Search is conducted based on the hash values of content and description strings, and follows semantics similar to Transmission Control Protocol (TCP). Each search request is sent to the neighbor node with nearest key, specifying TTL and a (pseudo) unique identifer for the message. If the message is found before the TTL expires, then the file is returned, and the source node's routing table is updated with the destination node's information. This is another method in which Freenet adapts itself to the changing landscape of the identifier space: high quality neighbors are remembered for later transactions. In addition, nodes do not have a specific addressing scheme. Instead, nodes are known for the content they provide.

The hybrid approach of BitTorrent [13, 22] incorporates security and fairness into its file sharing protocol. BitTorrent adopts some elements of Napster's sharing model: it builds a separate unstructured P2P network for each data item being shared, but stores information about which nodes are participating in that torrent network in one or more repositories. This reduces the total size of its networks, resulting in improved performance. Unlike Napster, BitTorrent tracker files, which record information about a particular sharing network, can be posted anywhere (such as on websites), and do not require a single centralized server. In addition, BitTorrent provides remarkable robustness. Earlier file sharing protocols suffered from data integrity issues: a malicious user could announce its possession of a particular file, and instead provide poorly formed blocks of data in its place. Through the replication of data across a P2P file sharing network, this would eventually contaminate a high percentage of downloads of that file. BitTorrent addresses this problem by assigning checksums to each block and the file as a whole, allowing peers to identify poorly formed blocks.

2.8 Content Structured Information Storage Systems

The second generation of P2P systems is built around the ability to store a library of information across a network. This data set consists of elements that are identifiable by a unique key. Search in such systems is concerned with locating a particular data element given a query which consists of the key itself, or in resolving one or more ”clues” (range queries, subject area, etc) to a small set of data. Unlike previous P2P systems, which relied upon searches for files by filename or perhaps a short subject description, content retrieval systems rely on more sophisticated data representation models. These models are most often adapted from database theory, and are outside the scope of this paper.

These approaches use both flat and hierarchical P2P systems as a means to store information. A key, although subtle, distinction between these systems and those in the next section is that content structured systems modify network structure to organize the content stored by nodes, whereas communications structured systems do not impose constraints on the data stored by nodes, but rather emphasize the efficiency of locating nodes in the network. More generally, content overlays search for content, whereas communications overlays search for nodes.

An early use of P2P systems is to represent digital libraries, and to build search mechanisms for locating content in these systems. An inherent problem in such systems is the differing semantics and organizational structures used to represent and index the heterogeneous data sets stored in large digital libraries [2]. To this end, many techniques borrowed from database theory are applied to distributed systems. In particular, content similarity, data representation, resource ranking and selection, and query semantics are discussed thoroughly in this body of research [9]. Lu and Callan [63,64] explore this topic very well, and use their acquired knowledge to develop robust and sophisticated content search networks for large-scale federated search in P2P systems. Their efforts are formally based on language models and data representation, and provide insight into the different methods for storing data of various classifications.

Zhang et al. [103] observe that the uninformative search strategies of unstructured P2P systems perform poorly for even simple queries. To this end, they introduce a topology reorganization that attempts to group context-similar data elements. They later extend this idea to include a multi-level hierarchical structure that groups agents into levels, and again by groups, by content similarity. Their search algorithms improve greatly on basic flooding approaches by assuming a cooperative system. In such an environment, agents cooperate to forward search queries intelligently based on content organization rules. Much in the way a HP2P system operates, they introduce the ideas of super nodes and peer nodes (group mediators and group processors, respectively), with super peers absorbing much of the decision making and management functions, while peer nodes are primarily concerned with responding to query requests [104, 105].

P-Grid [5, 6] (Peer-Grid) is a hybrid content storage and retrieval overlay that uses a virtual balanced tree (trie) to maintain a searchable structure of data items identified by unique keys. The tree structure itself is logical rather than physical, where nodes record locations of data items stored on other nodes, but do not form a physical topology in the shape of a tree. Each node is responsible for a subset of the data stored in the network, indexed by a specific data key prefix. The key space is segmented and reordered according to a self-organizing algorithm with the objective of achieving runtime search load balancing. The path to a key follows a trie search algorithm, where a jump from node to node proceeds along the bits of the key being searched, moving downward in the tree shown in Figure 2.3. Nodes at each level in the virtual tree store the location of a node corresponding to data keys that are in

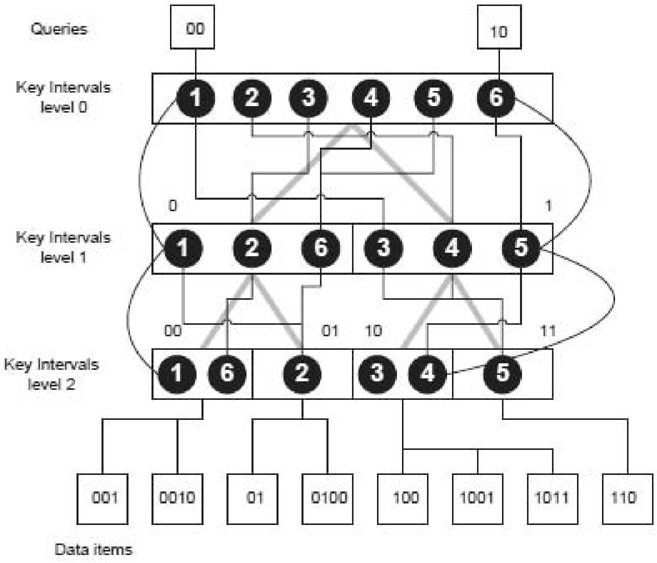

Figure 2.3: An example P-Grid tree structure [5]. This tree holds eight data items, shown in the bottom level. Searches begin from the top level, and progress downward, with one bit resolved at each level. Backward and cross-level links are maintained to reduce search time for locating distant (key-wise) data items.

different segments of the key space for that level. Storage per node is $O(\log D)$, where D is the number of data items in the tree, and expected query length is $O(\log N)$.

Rather than developing a single solution to the representation, organization, and search for a heterogeneous data set, Bao et el. [11] capitalize on the diversity of data sets and available federated search techniques to build the Heterogeneous Search (HES) system. The HES technique is built upon the idea that the data stored in large-scale P2P systems is semantically random[4]. HES incorporates multiple content storage and retrieval algorithms into a single agent structure, and uses a probabilistic, rather than deterministic, algorithm to choose a search technique to satisfy inbound search requests. The exact probabilistic module selector algorithm is itself interchangeable, and hinges upon either a learning algorithm or database language analysis of incoming queries to optimize runtime algorithm selection.

[4] Note: It may not be the case that the dataset itself is semantically random. Instead, the probability of existence of a data item in a network, the probability of a search query from a source reaching the node that stores the desired data item, and the differing semantics used to store and query items may cause searches to appear as probabilistic to an outside observer.

2.9 Communications Structured Peer-to-Peer Systems

Evolving out of the content retrieval systems is the more generic ability to locate nodes by key, rather than the data they store. This initial leap was a small one, although modern systems have proven to be quite sophisticated in regards to network organization and searching. In addition, the organizational constraints necessary to ensure reliable performance results also provide the opportunity to introduce improved feature sets. Many such systems incorporate one-to-one discovery and message routing, and some also provide direct support for maintenance, fault tolerance, and security.

This section introduces the structured overlay protocols, intended to be instructive and representative of the concepts available in the current body of research. This discussion includes both pure P2P systems, as well as HP2P systems that are developing more recently, offering many of the same advantages, in addition to an improved feature set. The results of this analysis are summarized in Table 2.2. This table organizes the below P2P strategies based on our taxonomy of large-scale multi-agent systems. The HP2P strategies are not included in this table because many of them are in early development, and thus lacking in formal rigor. Performance evaluations for these systems are included inline, where available.

2.10 Flat Peer-to-Peer Communications Systems

This section describes those communications structured P2P systems that provide a unique advantage or design technique over other approaches. Most commonly these differences in overlay strategies are a result of fundamental design differences, such as routing geometry, but also include performance variations, such as expected hop-length for node location queries under certain conditions. Two of the most cited and extended approaches are Chord and Pastry, and we provide a separate discussion of each of these seminal approaches and their offspring.

2.10.1 Chord. The Chord protocol [15, 86] uses consistent hashing [49] to locate nodes in a loosely structured P2P network. Consistent hashing in Chord uses a standard hashing technique (such as SHA-1 [52]) to create two hash values for a node. The node's identifier is the hash image of the node's location (IP address, port number, etc). The key identifier is produced by hashing a key that describes the node, such as the subject of the documents it stores or the node's task information.

The identifiers are ordered in a circle of size modulo 2^m, where m is the length of the hashed identifiers, in bits. A key k is inserted into this ring by finding the first node that matches the key, or the node that directly follows it in the idenfitifier space. (This process is essentially a hash table with collision detection [23].) This node is called the successor of k, and denoted $successor(k)$. To insert a key into a ring of nodes, the key k is assigned to the node at position k *mod* 2^m. If there is no node at that position, the key is inserted at the first successor node of k.

As an example, consider the Chord ring shown in Figure 2.4. This figure shows three nodes in a ring with $m = 3$. This yields a ring of size $2^3 = 8$, with nodes numbered on the set [0,2^m - 1] = [0,7]. Nodes are currently present at positions 0, 1, 3. An element with key $k = 1$ is stored at node 1 *mod* $2^3 = 1$. However, when inserting key $k = 2$, a node is not found at position 2 *mod* $2^3 = 2$. For $k = 2$, the first successor node of position 2, which is node 3, is assigned as the successor of this new node at 2. Inserting key $k = 6$ again hits a location with no node, and the first successor node in the ring is at position 0.

Using the Chord consistent hashing protocol, it can be proved that no node will store more than $O(\log N)$ keys in a steady state system [86].

Chord nodes also store a routing table describing the nodes they know about in terms of successors of keys they have seen. The finger table at a node n contains at most m entries, and the ith entry contains the identity, s, of the first node that succeeds n by at least 2^{i-1}. The node s is called the i^{th} $finger$ of node n. The finger tables are used to lookup keys in the network by querying nodes known to store keys

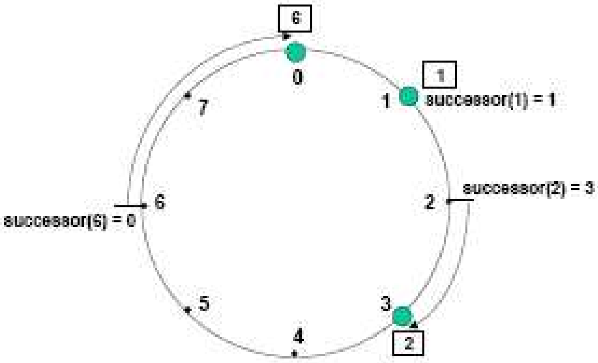

Figure 2.4: An example Chord ring with three nodes: 0, 1, 3 [86]. The locations with a filled circle indicate the presence of an agent.

close to the desired key. The number of nodes in a steady state system to be examined is, with high probability, constrained by $O(\log N)$ [86].

The restriction that these relationships hold under steady state is a product of the population and stability of the finger and identifier tables at nodes in a network. In a steady state network, the nodes have suitable knowledge about their neighbors within a given range (a few hops typically) to successfully route messages according to the logarithmic time predictions. However, in a network with a high frequency of transient nodes, the message queries may take longer, although they are still predicted to succeed. In addition, the structure of the finger tables can be exploited to provide duplicate-free broadcast.

An extension to the Chord protocol is called Recursive Partitioning Search (RPS) [90]. The purpose of this protocol is to extend Chord by improving the performance of lookup delays in duplicate-free broadcasts. That is, a Chord network performs $O(N)$ calculations in finding successor nodes for broadcast routing. However, RPS improves this processing time to $O(\log N)$. It operates by partitioning nodes into overlapping regions. Nodes performing search, to include broadcasts, are permitted to query only nodes within their own regions. In addition, the size of the permissible search region is reduced at each hop. When a message is sent to a following region, a tag is included which specifies a seed that is used to describe the allowable search regions. A simple algorithm is applied to the tag which guarantees the uniqueness of the next region to visit. In this way, under the Chord steady state constraints, a duplicate-free broadcast is achieved. However, the key space must be uniformly distributed across the network for the algorithm to function correctly.

The RPS and HP2P both organize the network of nodes into different segments. These zones may be organized based on application or locality. HP2P networks also incorporate the idea of a super peer, which is responsible for gateway activities for each cluster (zone). In this way, RPS implicitly addresses some of the necessary search and routing considerations found in a HP2P.

Another extension to Chord is Accordion [59]. Accordion incorporates a variable routing table size. Based on a tunable bandwidth limitation parameter, the protocol maintains routing information about a set of nodes whose cardinality is inversely proportional to the distance from the reference node. That is, it will store information about more nodes that are closer than farther away, with distance determined by the bandwidth tuning parameter. This allows system designers and maintainers, or the nodes themselves, to modify the tuning parameter to store more or less information about the network in each node as the system progresses. In addition to varying the amount of memory used to store routing tables, this approach also adapts to changing network traffic conditions. Based on bandwidth utilization, nodes may self-tune themselves to reduce the size of routing table, which also reduces the bandwidth used to perform table maintenance. Accordion will also perform parallel routing lookups [60] to reduce average lookup times, while still staying under the bandwidth limitation.

A variation of Chord, called Koorde [47], uses a de Bruijn graph to represent a DHT. A de Bruijn graph of base n values and m bits of resolution will have a node identified by each possible combination of the n^m bits. For example, Figure 2.5 shows the de Bruijn graph when $n = 2$ (base 2) and $m = 3$. The graph has 2^3 total nodes, each with two incoming and two outgoing edges. The outbound edges of this graph

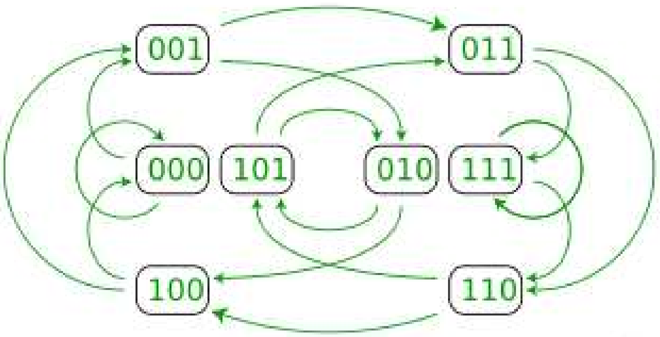

Figure 2.5: An example de Bruijn graph for $n = 2$ bits, $m = 3$, with 2^3 possible symbols [93]. Each node has exactly n incoming and n outgoing edges.

point at the two nodes whose identifiers are obtained by performing two left shifts of the bits identifying the source node: once shifting in a one, and once shifting in a zero. Koorde exploits this ordered connectivity to reduce the state of each node in a network. It is possible in such a graph to, given a key k, deterministically find the proper route to the destination by following the sequence afforded by the de Bruijn properties. Aside from the smaller routing tables, the rest of the protocol follows Chord. However, this approach requires a highly ordered and rigorously maintained organization, which has high maintenance cost. By maintenance alone this approach does not scale. In addition, the maximum size of the network must be pre-determined so the key space can be configured.

The One-Hop routing scheme [41] uses the Chord protocol as a foundation to support messages from source to destination to travel only one hop, in a steady state system. Each node in the system maintains location knowledge of each other node in the system. The routing tables are stored as in Chord. The challenge of this approach is network churn: nodes joining and leaving the network require updates. The structure itself is resilient as per Chord, however both cases require a broadcast to occur. This broadcast uses the assumption that all keys are uniformly distributed across the nodes of the network. The overlay structure is then divided into a set of k zones. Since there is a uniform distribution of keys, these zones will be probabilistically equal in size. The node at the mid-point of each key zone is forwarded a message about the new (or past) node, and that message is distributed as in a balanced tree, splitting the distance in each sub-zone at each hop. This approach yields efficient broadcast, however is not scalable due to bandwidth consumption.

The Two-Hop scheme builds upon the One-Hop scheme, but relies more on zone leaders. For each of the k zones, a (slice) leader is chosen. Each zone is then partitioned into units, again evenly partitioned across the sub-key space. Every slice leader submits the information for a unit of its nodes to another slice leader. That other slice leader then disseminates the unit's information to its entire slice. In this way, each node has routing information about a unit from each other slice. In order to send a message, the source node need only locate the node with closest key in the remote unit, and forward that message. In the worst case, the message will make a second hop once in the remote unit. This amount of cross-information may be undesirable for secure applications, and the assumption of randomly distributed keys may be unrealistic for the CyberCraft some networks. However, the authors did recognize the need to have "super peers" to facilitate the bandwidth requirements for this scheme, which is one step toward a HP2P configuration.

2.10.2 Pastry. Pastry [78] is a self-organized overlay network, intended to support applications. Machines with one of these applications also hosts a Pastry node, which is part of the Pastry network. Pastry provides a large-scale communications API for applications to use. Each node in a Pastry network has a *nodeId*. Requests are routed to the node that is numerically closest to the desired key, with $O(\log N)$ expected hop counts, where N is the number of nodes in the Pastry network. The *nodeId* space is distributed randomly, as each new node is given a random *nodeId*. As a result, with high probability, Pastry nodes with similar *nodeId*'s are distributed uniformly throughout the network. Each node maintains a list of the k nearest nodes, by *nodeId*. Using this, applications can replicate information or processing across these k nodes, which provides fault tolerance to failures because the nodes are distributed. Pastry also incorporates a small number of long-haul links for each node. These links are built according to a proximity heuristic which attempts to minimize network diameter [18]. Message routing in Pastry is very similar to the Chord protocol. Both systems maintain a ring of adjacent nodes, with logarithmic (base 2) addressing.

Bamboo [76] extends Pastry to address three key performance issues: reactive versus periodic recovery from failures, calculation of message timeouts during lookups, and choice of nearby over distant neighbors. Each of these attributes is given tunable features in Bamboo. Reactive recovery refers to the reaction of a node when it determines that one of its neighbors has failed. In this case, the node broadcasts its updated routing and neighbor sets to all of its k - 1 neighbors. The problem occurs when either (a) all nodes detect the failure at the same time and forward their full tables to each other (an $O(k^2)$ event), or (b) the keep alive messages were delayed due congestion, and the node didn't fail at all. Case (b) can further congest and even overload the network, thus causing additional nodes to appear to have failed. The authors call this a positive feedback cycle. The alternative to reactive recovery is periodic recovery. This approach is more patient, and relies upon periodic updates of differences to a node's table to be sent to its neighbors. Loss or acquisition of a neighbor doesn't change the operation of this approach, and it is thus less prone to congestion and is more resilient. However, it is also slower to notify the system of the change, which can delay updates of routing tables, thus resulting in a higher message query failure rate.

Bamboo supports two types of timeout calculations: TCP-style and virtual coordinates. In the TCP-style timeout calculation scheme, each nodes maintains an exponentially weighted mean and variance of response time for each neighbor. This allows nodes to have a rough idea of expected base timeouts for issuing searches to different portions of the network. The alternative scheme relies upon virtual coordinates. Virtual coordinate timeouts use machine learning to assign to each node a coordinate in a virtual metric space such that the latency between two nodes is represented as a line between them in the virtual coordinate space. Bamboo uses the virtual coordinate system found in Chord, called Vivaldi [25]. Vivaldi maintains an exponentially weighted average of past round trip times between nodes, and uses that to create reasonable timeout values.

Bamboo's final improvement over Pastry is to incorporate a smarter table population scheme. In global sampling, a node fills a slot with prefix p in its neighbor table by using the search capabilities of the DHT to its advantage. It performs a search for a random key with prefix p, and recording the first result. In a steady state system, repeated sampling will result in high quality neighbors. For local tuning [76], a source node contacts another node in its routing table at level l, and asks it for its level l neighbors. The idea is that some of these nodes may have lower latency than some of the source node's existing neighbors, and will have a similar search key prefix. The results are compared, and the source node's tables are updated if any closer nodes are found. The neighbors' inverse neighbors protocol samples those nodes who have the same neighbors as the source node. For example, two nodes may reside on the same network segment and be initially isolated from the rest of the network and unaware of each other. However, they may have the same neighbor in common. Querying that neighbor for its neighbors will help the two near nodes to discover each other. The final technique introduced into Bamboo is similar to Tapestry's nearest neighbor algorithm, and roughly combines the previous approaches. It begins with sampling the neighbors of nodes at level l. Then only the k nearest (lowest latency) nodes are kept from that set. The level l is decremented by one, and another sample is performed on the remaining k nodes. This process continues until $l < 0$, with consideration paid at each step to possible new neighbors.

Incorporating attributes from both Pastry and Chord, Kademlia [69,94] seeks to improve routing efficiency and knowledge sharing. It uses the symmetric properties of bitwise XOR operations to determine the distance to a target node. Kademlia stores information about other nodes in k-buckets, where k is the number of bits of address resolution. Each bucket may store multiple pointers to nodes, and all of the entries in a given bucket are examined when choosing a query's next hop. With a separate bucket for each bit of address resolution, the XOR distance between source and destination nodes is compared bit by bit with the bucket list indices. Progressing in this linear fashion effectively reduces the number of node segments under consideration by half at each step. Each bucket contains multiple entries, increasing the breadth of nodes under consideration for each network segment and increasing search accuracy and fault tolerance. Messages also store additional meta information, and intermediate nodes along a route peek at that information to maintain more consistent routing tables.

Tapestry [106] provides an API similar to Pastry, but stores its routing tables differently. A node's neighbors are stored by prefix, and a prefix search is conducted similar to how a trie operates [45]. Like Pastry and Chord, the message is forwarded to the node with the closest identifier after conducting a local search. The underlying assumption that makes it unsuitable for a HP2P is that nodes are free to connect anywhere in the network they choose. Doing so permits them to maintain routing tables that index prefixes that may be part of distant clusters. Although this provides reasonable search complexity, the information separation aspect of a HP2P is lost.

2.10.3 Other Design Paradigms. Viceroy [67] addresses two specific challenges found in large scale P2P distributed storage and search systems: the distribution of data to provide predictable performance bounds, and the maintenance of routing table in a high churn network. Viceroy's routing tables operate similarly to the Chord protocol, except that the outdegree for any node is constant. This constant outdegree is meant to aid in the maintenance of routing tables, which the authors believe is a more common (and higher priority) activity than searches in a high churn network. It is built on a butterfly topology, where nodes maintain links to other nodes at varying distance and level, so as to provide expected performance bounds. Forward and backward links facilitate routing table maintenance for nodes that join or leave the network.

Mercury [13] is a multi-attribute search DHT. Its routing protocols are derived from Chord. It introduces the idea of attribute hubs, which are solely responsible for a single attribute (although one physical hub may support multiple logical hubs). Mercury supports multi-attribute searches by dividing the key space into zones organized by primary attribute. Hubs are arranged in a ring according to contiguous values of attributes. This reduces the difficulty in search to simply finding a hub which stores the attribute. From there, the hub forwards the query to all of its leaf nodes which might match the remaining portions of the multi-attribute search. In this way, Mercury is a hybrid system, closer to a distributed P2P relational database than a DHT. This approach could be well suited for running in a HP2P architecture, since the idea of hubs in a P2P system lends itself to the thoughts of super peers HP2P architecture. Mercury is able to achieve $O(\log^2 N/k)$ hops for lookups, where k is the number of neighbors per node.

Content Addressable Network (CAN) [75] is a design for peer-to-peer indexing based on the idea of a DHT. The hashtable space is divided amongst the N CAN nodes using a deterministic hashing function, forming N zones based on a d dimensional Cartesian coordinate system. A query for a key K is hashed, and the location P determined by the value of that hash function refers to the zone in which K resides, if it exists. To facilitate nearness between adjacent zones in the search space, nodes dynamically reconfigure themselves to be connected to their zone neighbors. Routing is performed by moving messages to their destination zones. In terms of a Cartesian coordinate system, a line between source and destination is formed, and the message travels along that line by moving from zone to zone. Inserting a new node into a CAN network requires splitting an existing zone, but not modifying the original size of the space. Increasing the size of the space, and maintaining routing tables, is moderately expensive in this configuration.

Borrowing from the idea of skip lists [45], SkipNet [43] nodes store information about predecessor and successor nodes in a skip list. Nodes maintain points to neighboring nodes in the same subject area (i.e., similar hashed key identifiers), as well as pointers that skip over a number of levels of records. Nodes at level h from a source node are 2^h nodes to the left or right of the source node. The nodes are organized into a hierarchy of rings. The root ring contains pointers to sub-rings (with overlap), with each successive level splitting the ring into two roughly equal parts. Search is as efficient as Chord ($O(\log N)$), as the hierarchical ring structure essentially works like a search tree. Also, because of the highly organized and ordered nature of the skip lists, SkipNet also supports range queries. SkipNet nodes tend to store significantly more routing information than Chord nodes. As a result, search performance is comparable, but maintenance actions are also more costly in SkipNet.

The Distributed K-ary System (DKS) [8, 36, 37] builds structured peer-to-peer overlays using k-ary trees. The identifier space is recursively partitioned into intervals, and modeled as successive levels of a tree. This tree is then used to navigate the identifier space when searching for identifiers. Solutions exist to provide replication free broadcast and multicast, and updates are handled with a combination of change on use and correct on change semantics. This system is elegant and simple in its representation, however it does not scale due to higher level nodes holding more knowledge of the identifier space, with level zero maintaining a copy of the entire identifier space. Additionally, the system must be initialized with a maximum node value, and we have found no discussion of rebuilding the network with larger maximum values at runtime.

Kelips [42] segments the network into $O(\sqrt{N})$ affinity groups. Nodes in each group maintain a small constant number of links to other nodes in the same and foreign affinity groups. The number of affinity groups ($\sqrt{N}$) helps to ensure that each group maintains at least one link to each other group. Groups are divided by a uniform partitioning of the key space (using consistent hashing), and with $O(\sqrt{N})$ memory space per node, $O(1)$ lookups are achievable in a steady state system. Nodes use epidemic/gossip protocols to perform maintenance actions, with a fixed bandwidth limitation to prevent flooding. The system has been shown to be resilient in the face of failed nodes in networks of moderate size (100,000 nodes).

The Symphony protocol extends the small world principle by recognizing that increasing the number of long distance links, k, can lead to improved performance. The authors show that by choosing the k long distance links along a Probability Distribution Function (pdf) of $p_n(x) = 1/(x\ln N)$, with $x \in [\frac{1}{N}, 1]$, and 0 otherwise, that the average query length of searches scales with $O(\frac{1}{k}\ log^2 N)$ hops. This pdf is a harmonic function, by which the name Symphony is inspired. Symphony distributes nodes uniformly around a ring structure, and provides subtle optimizations such as look ahead (piggy-backing control information on pings), fault tolerance algorithms, runtime parameter tuning, and load balancing.

2.11 Hierarchical Peer-to-Peer Overlays

HP2P overlay structures combine flat P2P systems together to form a hierarchy of P2P systems. 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. [33] demonstrate that even adding a single P2P layer to an existing P2P architecture can improve the lookup path of searches by a factor of log N / log I, 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 four advantages of this approach:

- Provides transparency: data may move around and nodes may join or leave the network in each cluster, but the overall system is unaffected because each cluster is independently managed. This leads to improved reliability and search consistency.
- Significantly improves the average path length due to the hierarchical organization.
- Consumes less bandwidth than traditional P2P structured overlays under stable conditions. This occurs in HP2P networks whose clusters are formed based on network locality (such as TSO [99] and Brocade [107]). In these situations, clusters spend more time performing intra-cluster communications, leading to fewer long haul messages.
- Better supports heterogeneity. Each cluster in a HP2P network is a separate and fully functioning P2P overlay, such as Chord or Pastry: only super peers need speak the same language.

The Canon project [31] extends this work by providing methodology for merging structured P2P overlays (Chord, Symphony, CAN, and Kademlia) into hierarchical structures. The methods employed also support deterministic bounds on the degree for nodes in Crescendo, Canon's adaptation of Chord, on the order of $O(\log N)$.

Zöls et al. continue the trend of migration from existing structured P2P systems (Chord in this case) to hierarchical systems by analyzing the cost metrics for systems with limited bandwidth [44,108], such as mobile devices, and constructing hierarchical networks to conform to dynamic constraints. Their system, Chordella, dynamically adjusts the number of super peers based on available resources so as to create a cost-optimal value. Chordella also improves upon Freenet's caching algorithm by dynamically choosing which nodes along a path at which to store cached copies of content.

Fiat and Saia [28] have built a HP2P structured overlay based on a butterfly network [41], shown in Figure 2.6. They apply the butterfly topology to build a

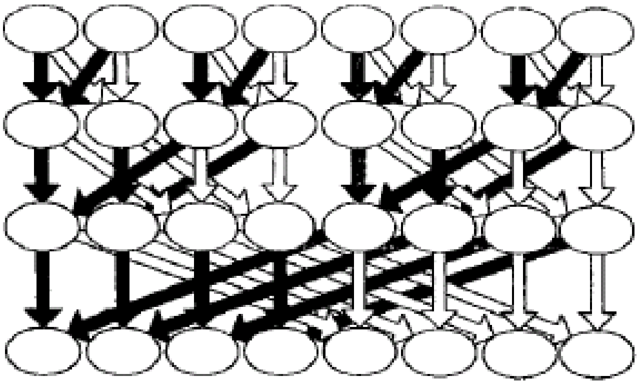

Figure 2.6: An example butterfly network [28]. There exist $O(\log N)$ levels, where each level is considered a super peer level. Leaf nodes connect to a random sampling of nodes at each super peer level to provide redundancy and fault tolerance.

censorship resistant P2P overlay structure. They refer to the nodes at each level above the leaf node level as super peers. Data items are stored at the leaf nodes, and the geometry of the system yields $O(\log N)$ provable search times. The butterfly loses some reconfigurability and fault tolerance, compared to other P2P approaches presented here, because the interconnectivity between nodes at higher levels reduces logarithmically to a small number (two). Therefore, loss of a node at the top level reduces 50% of the routing redundancy for that segment of the network. The pure butterfly network approach requires $O(\log^2 N)$ messages for a query, but the multi-butterfly aproach [26] can reduce this to $O(\log N)$.

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.

Current HP2P overlays combine existing technologies to enable distributed communications. However, the body of work into HP2P overlays has yet to establish its own uniqueness, wherein approaches are tailored to the specific advantages of HP2P architectures.

2.12 Design Tradeoffs

When examining Table 2.2, perhaps the most obvious distinction is the tradeoff between route hop length and node memory usage. In general, increasing the amount of node memory increases performance because nodes have more knowledge about the global state, and can make better decisions for routing queries. Unfortunately, these systems tend to have lesser scalability as large systems will use more memory, which may be a limiting design factor. In addition, maintaining each node's memory state requires higher bandwidth consumption as the amount of global state stored per node is increased. Systems such as Accordion offer a nice compromise by permitting the designer and maintainer to specify runtime limits for memory and bandwidth, and allowing the network to tune itself based on bandwidth utilization.

The overlay routing geometry does not appear to directly affect the scalability of the system. The performance of systems with similar routing geometries varies based on an overlay's goals and implementation. For example, DKS and Kademlia both use trees for routing, but in different manners, and their resulting performance differs as a result.

However, the strictness of the overlay rules does seem to affect the scalability. Systems such as Koorde impose a rigid set of rules for the locations of nodes and suffer a penalty for maintenance actions and reorganizing in high churn or expanding systems. Armed with knowledge of previous systems, approaches such as Viceroy harness the strengths of several approaches to create a more resilient and better performing system than their predecessors. In addition, more loosely structured solutions exploit their polymorphic nature to adapt to changing network conditions, but at the expense of higher maintenance costs.

Although scalability is qualitatively defined, there appears to be a relationship between query path length and scalability. Systems that offer much lower query path length, such as Kelips and One-Hop, also suffer in scalability. This is tied to the means necessary to acquire a significant advantage in query path length: increased memory usage and associated maintenance bandwidth. While these systems are designed for performance in smaller systems, large-scale systems will suffer an indirect negative impact at the expense of route length. Although the route length is not the cause of the lack of scalability, there exists a correlation between the two.

In communications structured P2P overlay networks, all nodes are identified through some unique identifier. Some of the approaches discussed here use a distributed algorithm to ensure uniqueness of the node identifiers, while others rely on the uniqueness of a node's properties combined with a hashing algorithm to generate the identifier. Whichever method is used, and so long as the addresses are generated dynamically, does not appear to directly affect the performance attributes of the system. In addition, several systems offer the advantage of allowing hashing mechanisms to be substituted, which increases their flexibility.

The (network) physical distances between nodes must be considered before choosing an approach. Many of the approaches described here rely upon a number of local neighbor links and long haul links to establish smaller network diameters for improved search efficiency. While this works well in a local system with high bandwidth links, reliability can suffer when connecting large groups of peers across long distances [68]. This happens when many long haul links attempt to span the network by using lower bandwidth links. This self-organization property is addressed in more detail in many HP2P systems [109, 110], but less so in flat P2P systems.

HP2P overlays provide additional opportunities for heterogeneity and autonomy by allowing subordinate organizations to independently manage and organize their networks according to their own missions [33]. This is especially important for enterprise systems that consist of many separate units, missions, and available computing architectures – one solution will not suffice for all scenarios. Network churn (and associated maintenance) is localized to clusters, and in general does not affect the large-scale system functionality.

2.13 Applicability to HP2P and CyberCraft

While most overlay structures are currently developed as pure P2P systems, certain applications may require a more segmented approach [50]. For example, military systems require additional security, which is an area that many of the current overlay structures have yet to examine. One possibility to facilitating this objective is to adapt existing approaches to HP2P structures. This topology explicitly provides an environment which is more suited to security constraints than existing P2P technologies.

A primary focus of many of the P2P overlays is to provide guarantees on the length of routes and node memory. Both of these elements are important for the CyberCraft application, but some methods of achieving these goals do not directly support the HP2P structure. For example, Pastry achieves its properties by evenly distributing communications links across the network. That is, a single node will attempt to connect to neighbors that are evenly distributed throughout the network. That node's neighbors attempt to do the same, with the end result that a maximum coverage is achieved, thus minimizing expected route length between nodes. This approach works wonderfully for a pure P2P network, however it will not work as well for a HP2P.

One of the purposes for using a HP2P architecture for the CyberCraft application is to minimize the amount of traffic crossing long haul links, which may be unreliable and slow, such as satellite communications or other wireless technologies. Systems such as Pastry rely on a uniform distribution of links, where each node is expected to maintain communications links with nodes evenly spread across the network. This reduces the average hop length by providing a nearness property for remote nodes, but does not consider the quality of communications links available. These long haul links may be costly and unreliable.

The HP2P architecture addresses this challenge by partitioning nodes into clusters that are expected to conduct most of their communications at the cluster level, and communicating with remote links sparingly, and predominantly through cluster leaders (super peers) which are chosen based on routing availability (bandwidth and processing). Link quality can be determined at runtime, however the creation of clusters is a function of the mission objectives and self-organization properties.

Once a clustering algorithm is developed, the techniques for choosing routing paths described above can be applied. The ring (or other) technologies seek to obtain optimal routing neighbors based on identity similarities. This technique can continue to be used in a HP2P, so long as additional restrictions can be applied about which neighbors can be chosen. These restrictions will be based on architecture and runtime clustering, and must reflect the characteristics of the system as designed and deployed for the intended application(s).

In addition, the issue of security in a large-scale system of peers is complicated by key storage and knowledge, rooted in the difficulty in establishing a foundation of trust. Trust chaining in a system of peers suffers from the idea that any node could be weaker, in terms of trust, than other nodes, and therefore the entire chain of communication after that node has a lower trust level. This problem exists in epidemic proportions in a system which relies on communications with long hop lengths and few authoritative sources.

The problem of task allocation and coordination in large-scale systems with partial visibility is still an active area of research. The difficulty of such a problem is compounded by constrained visibility of nodes in the network, greatly complicating allocation of resources and coordination of nodes in the system. This problem can be modeled as a non-stochastic Decentralized Partially Observable Markov Decision Process with Communication (Dec-POMDP-Com) [38, 98], which is PSPACE complete [23, 34]. This model prevents solutions that scale to the size of existing P2P structured overlay networks [35,81,82,95], let alone the potential scale of CyberCraft. Beyond simply sharing of data between nodes, new methods for distributing tasks must be developed to maintain the momentum of the usefulness of large-scale P2P systems.

Based on the observation that Viceroy has capitalized on the intersection of multiple routing geometries, what other gains can be made by combining techniques? Innovative ideas and adaptations of previous work contribute to the body of structured overlay research, with considerable success in improving runtime performance and providing additional design choices for system designers and application developers. I believe that a continued effort along this path will continue to yield improvements to the usefulness of structured P2P overlay networks in fielded systems.

The HP2P topology provides several advantages to such a system. Given a geographic organization of the network, the clusters of agents will roughly represent different physical areas, and should therefore have greater capacity to communicate with each other. Communications between clusters occur through super peers. While the super peers may change at runtime, we can assume that the agents of a cluster or group of clusters will autonomously choose the best candidate for that job. We can further assume that those chosen super peers will prosecute their responsibilities in earnest, without greedy semantics; that is, the agents are trustable. Building these properties into a multi-agent system may require an agent trust model, and is a subject of ongoing research. This situation aids in the realization of a fielded CyberCraft network: its size is also its greatest strength. Despite the complexity of the analytical solutions to the task allocation and coalition formation problems, it may be possible to build a distributed algorithm that can decompose the tasks and solve these problems in parallel, using the great number of CyberCraft as advantage.

However, it may not be the case that all CyberCraft will be geographically oriented. Under certain conditions, mission oriented coalitions must be formed to accomplish higher priority tasks (this is similar to how the U.S. military uses Centralized and Specified Commands). Should a high priority task be created, such as analysis of a possible realtime attack, it may be necessary to sequester additional computing resources via other agents. A large enough task may span far beyond the originating node's cluster, and even span continents, involving slow or unstable connections (satellites). The structure of a network in this case, except as it is used to route communications efficiently, is neither an advantage nor disadvantage. In addition, the question of task allocation becomes more difficult, as multiple high priority tasks may be competing for resources. Such contention can lead to failure of multiple tasks that must have realtime processing capabilities. For example, standard timesharing or loose realtime constraints may be insufficient: it is unacceptable that a security monitoring task be suspended to perform one of many other tasks, regardless of their priorities. The question becomes one of describing tasks in a way that assures fairness and progress, while also maintaining realtime requirements.

2.14 Large-Scale Coordination

Once reliable large-scale communications are established for a mission oriented system, the missions themselves must be deployed for agents to accomplish. Missions may be single-agent or multi-agent, and each agent may be single- or multi-task. A single agent mission is one that can be accomplished by a single agent, and a multi-agent mission requires multiple agents. A single-task robot can only perform one task (mission) maximum, whereas a multi-task robot can perform multiple tasks simultaneously. The CyberCraft is expected to be a multi-robot, multi-task system.

Early techniques for distributed problem solving revolve around centralized algorithms where a single agent is responsible for distributing equal shares of the workload to slave agents. However, like the progression of network architectures, this centralized approach does not scale and requires new ideas to accommodate systems with large numbers of players. Decentralized algorithms to solve problems are under active research as more and more large-scale systems become available to solve more complex problems. An initial challenge of these algorithms is the effective recruitment of processing agents to a team of agents to solve a problem. These teams are referred to as coalitions, and form a sub-field of distributed problem solving necessary to employ more useful problem solving algorithms.

Coalitions may be formed for a variety of reasons, and methods for their formation are a subset of the Multi-Robot Task Allocation (MRTA) problem known as multi-robot, multi-task assignment [34]. It involves the grouping of agents to produce benefits to the system, and may involve the use of game theory reward-based incentives. Distributed Problem Solving (DPS) need not require incentives for agents, but does seek to increase total utility of a system. The end goal of both is to create coalitions that increase the value of some function over that which can be achieved with individual agents.

Many of the existing algorithms for forming coalitions are based on economic incentives. Given a system of rational agents, whether cooperative or competitive, they only merge to form coalitions when the effect of joining is more productive (or profitable) than working individually. There currently exists a void in research efforts for building coalitions in large-scale systems, and so the existing techniques will not directly apply in the CyberCraft environment. However, it may be possible to adapt and combine existing ideas to form a basis for a new approach which can be useful in a large-scale system.

2.14.1 Coalition Formation. Shehory and Kraus [82] describe two methods for coalition formation using reward incentives. In a negotiation based formation, all single agent coalitions begin by interacting with other agents to determine if forming a joint coalition can yield a higher payout than remaining alone. In the case that two agents both determine profit can be increased by forming a coalition with each other, they negotiate a sharing of the additional payout yielded by forming the coalition. This payout can be different for the two agents, and they need not share it equally. Rather, the agents will negotiate a fair split of the profits, based on greedy or other semantics. In the negotiation algorithm, this process occurs between all pairs of agents, and each agent attempts to form a coalition with its most profitable partner.

Once a multi-agent coalition is formed, one of its agents is chosen as a representative. The representative's duties include negotiating for further coalition forming on behalf of the coalition it represents, and assuming risks (rewards) for the accuracy of utility calculations [53]. This formation process is repeated for coalitions of ever increasing size, and in a super-additive environment, ends with the formation of a single grand coalition. The overall purpose of this algorithm is to maximize the profit achieved at each step, yielding a likewise highest profit grand coalition. Each step of the process, repeated at most $n-1$ times, requires $O(n^2)$ communications operations and $O(n^2 \mathrm{log} n)$ computations, where n is the total number of agents in the system. The complexity of the general algorithm is then $O(n^3)$ communication operations, and $O(n^3 \mathrm{log} n)$ computations.

The second algorithm builds upon the Shapley formula [95]. This is a centralized algorithm in which a single agent collects information about the resources and other relevant information from all other agents in the system. The agent will then calculate the Shapley value, which involves finding the payout values of all 2^n pairs of agents. These payouts are organized into a prioritized data structure, and all agents are then informed of the new coalition schedules. This centralized algorithm requires $O(n)$ communications (it contacts each agent twice), and $O(2^n)$ computations.

The Contract Net Protocol (CNP) [84] is a popular contract system to allocate tasks, or portions of tasks, to one or more agents. Given a system of agents, any agents with a surplus of work to perform may start an inverted blind auction (contract proposal) for which other agents with a surplus of resources can bid. The bidder with the most attractive offer (lowest payout) is awarded the contract. Agents form networks of auctions, and may join and part them at will. This concept can be applied in a HP2P structure, where agents are naturally organized into clusters. This method is extended to build upon more modern communications facilities, such as ordered delivery of TCP and higher assumed bandwidth, easing constraints in the original protocol [79]. The CNP is useful in heterogeneous and homogeneous systems in which agents do not have full information about other agents. Rather, the agents submit themselves as candidates for processing a certain task, based on availability and capabilities, without revealing their full state information.

The previous accounts lead to the observation that the formation of coalitions in a system of agents can be viewed as a Set Covering Problem (SCP) or a Set Partitioning Problem (SPP) [34], both of which are $\mathcal{NP}$-hard [23,39,72]. As such, an algorithm to find optimal solutions requires at least exponential time.

Shehory and Kraus [81] address this challenge by constructing a heuristic that constrains the maximum size of any coalition. The objective is to reduce the total communications cost for a multi-agent system when forming coalitions. In addition, the calculation of the coalition value functions used to decide which agents should merge to form coalitions is distributed. For a system with maximum coalition size k, the computational complexity is of order $O(tn^{k-1})$, where t is the number of tasks being evaluated. The communications complexity is $O(n)$. Because it employs a heuristic, this is an anytime algorithm, meaning that if halted before termination it should still form coalitions.

Xu, et al. [97] use a token-based scheme to support more scalable DMAS coordination. They use tokens to represent anything that needs to be shared, to include tasks, resources, and information. This approach simplifies the challenge of locking shared resources, especially in a system with partial visibility. However, tokens can migrate to remote parts of the network, and so fairness can be difficult to guarantee in large-scale systems. Should a majority of these tokens move to one section of the system, then other parts of the system become starved for resources. The authors considered systems of up to 400 Unmanned Aerial Vehicle (UAV)'s. However, for large-scale systems of up to one million agents, passing tokens that represent resource locks across the network may incur performance penalties due to excessive communications in accessing those resources. Introducing a heuristic, or capitalizing on the properties of a structure such as the HP2P overlay, reduces these effects by localizing the tokens.

Abdallah and Lesser [3, 4] capitalize on the organization of agents to construct task coalitions that maximize global utility. Their solution builds a hierarchy of manager and leaf agents that delegates coalition formation authority to lower levels of the hierarchy to satisfy task requirements. The algorithm uses distributed learning to modify the organization at runtime, based on task allocation patterns, to improve global utility over time. Their Distributed Learned Policy (DLP) algorithm was successfully verified on systems of 103 agents.

2.14.2 Communication and Coordination in a DMAS Agents coordinating toward a global utility function must communicate to share information. However, perfect information sharing is unreasonable for even smaller systems because communications incur a cost. Agents must therefore attempt to evaluate the effectiveness of a communicating before initiating a transaction with one or more other agents. This is a decentralized decision process because agents do not have global state knowledge. The decision to communicate can be modeled as a Dec-POMDP-Com, possibly aided by heuristics, where agents attempt to evaluate the state of other agents. This interpretation of another agent's state can be used to build an informed decision about whether or not to communicate to retrieve or push updated state information. Without explicit communication between the agents, they can attempt to interpret state changes in the environment to decode another agent's state and action pairs. This is accomplished with a Markov decision process, and is used to develop an agent's plans and decide when to communicate, if necessary. ItX is therefore informative to understand the classical framework for Markov decision processes for multi-agent communications.

Using the definition from Xuan et al. [98], for a Markov decision process in a multi-agent system we define $\alpha = \{X, Y\}$ to be the set of agents, and $M^x =$

$(S^x, A^x, p^x(s_j^x|s_i^x, a^x))$ is the Markov decision state for agent x, where S is the local state space, A is the local action space, and p is the probability of moving from state s_i^x to state s_j^x by taking action a^x. For a decentralized Markov decision process involving a pair of agents x and y, we define the global reward function $r_{xy} = (s_i^x, s_j^y, a_k^x, a_l^y)$ to be the reward given to the system for the joint action (a_k^x, a_l^y) taken when the agents x and y are in states s_i^x and s_j^y, respectfully. The cost of communication is defined as $c_t^x = (s^x, m^x)$, where m is the content of the communication, and t represents a particular time. For a decision not to communicate, m is null (or zero), otherwise m represents the type of message, where each message type may incur a different cost. The cost function is similarly defined for agent y, since we are considering joint actions in a decentralized system.

Xuan et al. [98] compare two heuristic approaches to policy definitions for the Dec-POMDP-Com coordination framework. The so called "No news is good news" (NN) approach adopts a policy of optimistic progress: communications are only performed when current plan execution fails or progress is not as expected. This policy is based on a heuristic function $f(s_t^x, s_t^y)$ that builds short term goals, and a progress function $g_l^x(s_t^x, \hat{s}^x, t)$ that determines if the current plan has made sufficient progress toward the goal state $\hat{s}^x$ at time t. attempts to determine if sufficient progress has been made toward the goal to consider the current plan on track (successful). Their second policy is the "silent commitment" (SC) that performs an initial division of labor, and performs no further communications before a pre-scheduled rendezvous point. For the duration of the processing, no communications occur and so the cost $c = 0$.

The goal of each of these heuristics is to create a model that is tractable and can be computed locally at each agent. The SC reduces communications cost at the expense of increased uncertainty, whereas NN pays communications costs to decrease uncertainty. The authors incorporate qualities of both approaches into a hybrid heuristic that attempts to measure the uncertainty of progress toward the goal, and communicates when that cost exceeds the cost of communications. It is based on building short term local goals and observing communications from other agents: if no communications are occurring, the agent assumes that remote agents are progressing toward their goals, which reduces local uncertainty as well. Agents continually evaluate new short term goals, and will change plans and communicate only when uncertainty in the current plan may incur a cost greater than the cost of communicating. The authors found that NN performs better when uncertainty is higher, since it will transition to a communicative stage. Under such circumstances, the hybrid approach adopts the NN's policy of periodic communications and will perform similarly. In the case of low uncertainty, all three approaches perform closely and with understandable success. Due to the continued evaluation of new plans, the hybrid heuristic does consume more processing power.

Goldman and Zilberstein [38] extend this formal model to establish separate action and communication policies, granting more dexterity to heuristics in choosing when and how to communicate. They perform additional testing to demonstrate further optimizations to the existing solution paths, demonstrating the effectiveness of policy separation and goal decomposition.

The Communicative Multi-agent Team Decision Problem (COM-MTDP) [73] combines and extends many existing theories about Dec-POMDP-Com's and economic incentive algorithms for multi-agent teamwork. The key benefits of this model are an analytic representation of both the complexity and optimality of team performance for various classes of problem domains. This approach borrows and extends the theory of economic team theory. It incorporates more rigorously defined notions of communications, reward functions, and observability to generalize and incorporate existing theories into a single larger and more descriptive model. This model provides a framework for evaluating the complexity and optimality of multi-agent coordination strategies.

The coalition formation problem can also be considered as a variant of the task allocation problem. It is therefore instructive to examine the properties and approaches to solving the task allocation problem in this context, as doing so may reveal additional ideas useful in creating large-scale solutions to the coalition formation problem.

2.14.3 Task Allocation. The MRTA problem can be stated as follows [35]. Given are m robots, each capable of executing one or more tasks, and n possibly weighted tasks, each requiring one or more robots. Also given for each robot is a nonnegative efficiency rating estimating its performance for each task (if a robot is incapable of executing a task, then the robot is assigned a rating of zero for that task). The goal is to assign robots to tasks to maximize overall expected performance, taking into account the priorities of the tasks and the efficiency ratings of the robots.

Given that the MRTA problem is also $\mathcal{NP}$-hard [34], it is possible to view it as many different types of problems. As with the transformation of the coalition formation problem to a set covering or set partitioning problem, the MRTA problem can also be transformed to other $\mathcal{NP}$-hard problems. MRTA can be further divided into categories [34], depending on the form of task and agent system:

- Single-task robots, single-robot tasks, instantaneous assignment: Each robot is capable of performing a single task at once, the tasks assigned are intended for a single robot to complete, and the task requires no future planning (all variables are solidified upon task assignment).
- Single-task robots, single-robot tasks, time-extended: Same as previous, except the tasks require a planning step to build task schedules to minimize total cost (execution time).
- Single-task robots, multi-robot tasks, instantaneous assignment: Each robot is capable of a single simultaneous task, tasks require multiple robots to complete, and task scheduling is complete at time of task assignment.

- Single-task robots, multi-robot tasks, time-extended assignment: A more difficult system where tasks require multiple robots to complete, where task assignment includes scheduling.
- Multi-task robots, multi-robot tasks, instantaneous assignment: The addition of multi-task robots eases the burdens of assigning tasks.
- Multi-task robots, multi-robot tasks, time extended assignment: The most flexible option, and the intended application domain of the CyberCraft.

Theocharopoulou, et al. [88] examine task allocation and distributed task scheduling in a large-scale system to solve the Distributed Constraint Satisfaction Problem (DCSP) [101]. They determined that the only feasible means of mitigating the analytic complexity of the problem in large-scale systems is by means of heuristics or relaxation of constraints. They create a "gateway" protocol which is used to recruit members of a coalition. Each gateway node is then used for further communications with other nodes outside of its group. In this respect, they are moving toward a HP2P structure for coalition formation. Although their approach is promising, they examine only systems with less than 150 agents and less than 10 tasks. This work is leveraged here and incorporated into an approach that explicitly supports and exploits the properties of a HP2P system for large-scale coalition formation.

The CyberCraft project is an instantiation of the multi-robot multi-task environment, in which agents are capable of performing tasks requiring either one or more agents, and with each agent capable of performing one or more simultaneous tasks. CyberCraft tasks will be introduced at runtime (online assignment), and the form and goals of those tasks may not be known ahead of time. Application of this paradigm to multi-agent systems is not yet fully understood, and to large-scale multi-agent systems remains an open problem. Indeed, finding an optimal allocation of tasks in this system is challenging due to the complexity of the fundamental problem.

III. Methodology

This chapter describes the design of the Resource Clustered Chord (RC-Chord) structured Hierarchical Peer-to-Peer (HP2P) overlay and the Distributed Likelihood of Execution (DLOE) coalition formation algorithm. These two systems are used in conjunction to develop a solution to the large-scale distributed cooperative task allocation and coalition formation problem.

The primary contributions of this work to the field of Peer-to-Peer (P2P) networking and distributed systems are the creation of the first scalable HP2P structured overlay system that organizes agents by the resources they posses, the definition of a CyberCraft task model, and a large-scale cooperative coalition formation strategy. This discussion includes the goals of each approach, and focuses on the design methods used to achieve those goals.

The chapter begins by introducing the RC-Chord HP2P structured overlay in Section 3.1. It continues by describing the RC-Chord design paradigm in Section 3.1.2, and concludes the RC-Chord discussion in Section 3.1.7 by describing RC-Chord's wealth of tuning parameters and their effects. Section 3.2 introduces the methodology of the the DLOE coalition formation algorithm. This includes a definition of the CyberCraft task model in Section **??**, and of the task scheduling model in Section **??**. The chapter concludes with a summary of the techniques described here, as well as how they combine to form a comprehensive solution to the cooperative coalition formation problem.

3.1 RC-Chord

The aptly named 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 at higher cost 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 [33]. This localization also provides a logical grouping that commanders can more easily understand and supports localized ownership by system administrators.

As described in Chapter II, research into large-scale HP2P structured overlays is currently a developing field. RC-Chord extends this body of research by incorporating with each agent one or more resources or capabilities. These agent capabilities serve as a criteria for the creation of HP2P clusters, which are described here.

3.1.1 HP2P Structured Overlay. Existing techniques for P2P structured overlays are adapted and extended to support a new HP2P structured overlay, called RC-Chord. This technique uses many of the design tools associated with Chord [86], as described in Chapter II. Chord provides a protocol that is mature and well accepted and adopted by the community, and achieves commonly accepted runtime performance constraints for large-scale structured overlay systems ($O(\log N)$ path length). However, Chord and many other strategies rely on a uniform distribution of node keys throughout the network. This is generally achieved through assignment of random identifiers to nodes, salted with location dependent information, and those nodes are then connected in some optimal manner so as to minimize the average distance between any two agents in the system. In a system where communication links vary greatly in performance and reliability (such as satellite channels), these long distance links may not be optimal or even feasible. Instead, the communication links across these channels should be allocated carefully and sparingly so as to minimize the burden on them, and also to maximize locality of information and mission objectives.

Beyond the initial adaptation of Chord to support resource location in RC-Chord, the next advantages of RC-Chord are the construction and organization of Chord instances into a HP2P architecture. Each cluster within a HP2P network acts as a pure P2P network. These clusters are then connected together to form a hierarchy of smaller networks, where each sub-network is viewed as an individual node in a larger network. The hierarchy is created by organizing clusters together into a tiered or level based approach, so as to construct an n-ary tree of P2P networks. This process repeats indefinitely to achieve the desired organization structure, stratified by performance and topological goals.

3.1.2 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 standard computer science 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.

Figure 3.1 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 3.1 shows a single level two cluster for each of the system's three resources. Nodes possessing resource three have continued to join the

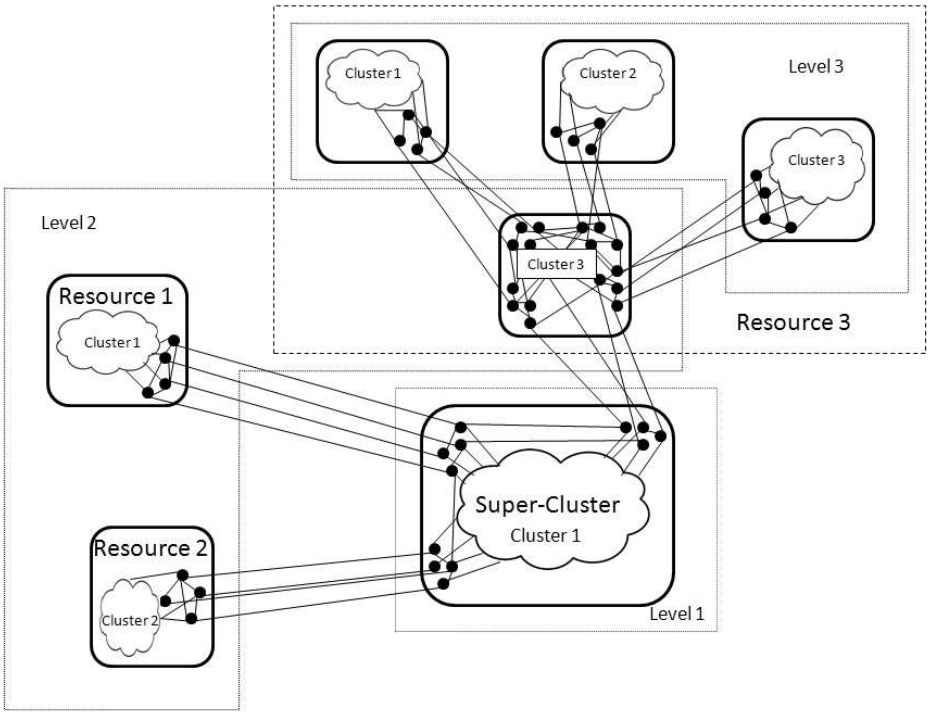

Figure 3.1: An example RC-Chord instance with three levels. The super-cluster exists as the sole level one cluster. Its nodes serve as super peers for level two clusters, with a single level-two cluster for each resource. Clusters at level two begin the formal sub-graph for each resource type, and may extend into additional levels based on number of nodes for each resource.

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 [86]. 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.

3.1.3 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, the number of clusters per level, C_l, is:

$$C_l = C_{i-1} * 2^m / P_{sp} \tag{3.1}$$

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

$$N = 2^m * \sum_{i=1}^{l} C_i \tag{3.2}$$

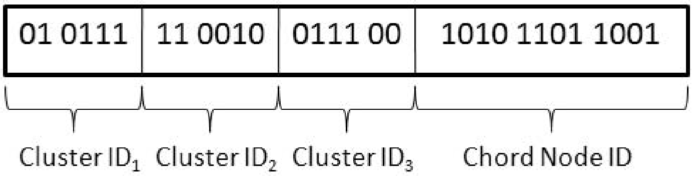

Figure 3.2: Global node address for a node at level four. Beginning from the left at the super-cluster, each stage in the address represents the ID of the cluster at the next level. The final segment of the address is the cluster-local identifier for the target agent. The combination of these cluster ID's and agent ID together form a global agent address.

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.

3.1.4 Addressing in RC-Chord Addressing in RC-Chord operates similarly 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 3.2 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.

Based on the peer to super peer ratio, the total number of sub-clusters for each Chord cluster is below 2^m. For example, a cluster with $m = 12$ and $P_{sp} = 50$ has roughly 81 super peers connecting it to its next higher cluster. That means the higher level cluster has 81 agents dedicated to serving that sub-cluster. Since agents in this system may serve as super peer for at most one cluster, this means the set of 81 agents in the higher cluster may not serve as super peers for any other cluster. Each cluster therefore has a branching factor of 50 sub-clusters. Addressing 50 sub-clusters requires six bits of binary address space. Therefore, each cluster ID in this system needs only six bits of address space, and so the first three segments of the address shown in Figure 3.2 consume 18 bits. The agent's final address segment uniquely identifies it within its own cluster, and so requires 12 bits. This leads to a total address width for this address of 30 bits. Compared to full addressing used in typical Chord networks, this is a savings of 36 - 30 = 6 bits, or enough address width for another level of the hierarchy.

3.1.5 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 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

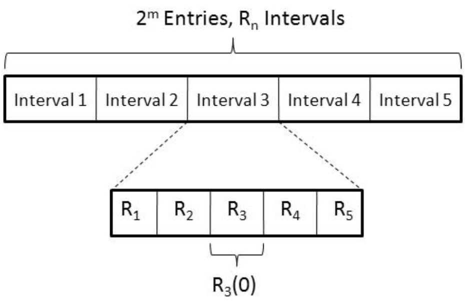

Figure 3.3: Example agent mapping for a cluster with m bits of address width, and n resources. The 2^m address entries are divided into R_n intervals. Each interval has one entry for each of the n resources. The initial interval for resource three is shown as $R_3(0)$, being the first entry for resource three in interval three.

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.

3.1.6 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 3.3 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. The index variable n chooses which of the intervals to examine for each resource. To help eliminate clustering of nodes into each interval, the initial value $n_0 = R_n$ specifies the starting interval for each resource. Since the resources are uniquely identified, and assuming a stochastic entry pattern for nodes by resource type, this constrains the first node of each resource type to start in a different interval. The k^{th} subsequent node for a resource lands into address interval $(n_0 + k) mod R_N$. The impact of this scheme is to enforce the Chord protocol's uniform distribution requirement, from which the Chord performance guarantees are built.

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 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.

3.1.7 RC-Chord Parameters. This section describes the key RC-Chord parameters. These variables are meant to tune the performance of an RC-Chord instance. The following section introduces those variables within the simulator that directly affect the performance of RC-Chord during experiments, but that are not directly tied to real RC-Chord instances.

The identifier length, m, is the number of bits used to store agent identifiers. This variable is an integral part of the Chord protocol. It specifies the address width of agents within each cluster, and therefore also dictates the maximum number of agents that can exist in any Chord instance (2^m). When the Chord clusters in the RC-Chord are full as a result of meeting the 2^m nodes in a cluster, additional clusters are constructed. For systems of many nodes, setting a low value of m can result in a great many clusters being formed, which also uses a significant amount of simulator memory. For RC-Chord testing, this variable is moved between 10 and 20 ([8,20]) to examine its effect, and for coalition formation testing of systems of one million nodes, it is left at 12 (4096 agents per cluster).

The variable *num_resources* is the number of resources recognized by the system. Each agent in the system may have any number of resources on [1,*num_resources*]. As part of testing a baseline Chord instance, *num_resources* is set to one, and m is set large enough that all nodes in the system fit into a single cluster. This ensures that only the super-cluster is created, and that all nodes within that cluster are equal and free to communicate without restriction (i.e., no semantics associated with crossing resource, cluster, or super-peer boundaries). However, in order to exercise the full body of software developed for this system, multiple resources are made available during additional testing (typically five).

The *peer_to_superpeer_ratio* establishes the number of peers for every super peer in normal clusters, which serves as the mean for a distribution that decides when to promote a new node to a super peer. That is, for each *peer_to_superpeer_ratio* nodes, make one super peer. This ratio is observed universally within RC-Chord, and across all time boundaries (startup versus churn, etc). For simulation purposes, agents that act as super peers also act as a normal peer in the next higher cluster. This is done for simplicity reasons, and for maintaining the cluster fullness property more easily. As such, specifying large values of the *peer_to_superpeer_ratio* can increase the number of clusters in the system because those super peers are each a member of two different clusters.

It is important to choose values for m that make sense with the simulation or real network being constructed. The number of sub-clusters from the super-cluster is calculated as 2^m / *num_resources*. Thus if *num_resources* $>=$ 2^m, then the configuration becomes invalid. For clusters other than the super-cluster, the number of sub-clusters is calculated as $2^m/peer\_to\_super\_peer\_ratio$. As with the super-cluster, choosing improper values for these variables can yield an improper configuration.

max_super_peers is the maximum number of super peers per cluster. This is a safety variable that helps to reduce any chance of creating the improper configurations described above, by establishing a hard limit on the number of super peers per cluster.

The goal of this variable is to reduce the chance of errors in the system configuration, and to ensure reasonable performance under abnormal situations such as high churn and network failures.

maxresource_per_node specifies the maximum amount of resource allocated per node. Likewise, *minresource_per_node* specifies the minimum amount of each resource that a node may possess. Both parameters are tunable to the coalition formation workload and task requirements. These variables provide a simple and effective method to tune the simulations to different environments.

k is the number of pointers in the Chord finger table. This variable is tuned to the address width of the network, and is useful in achieving address location flexibility. Larger values of k increase the number of entries in the Chord finger table, thus providing greater resolution in agent searches, and also storing more node references. Both of these characteristics can improve the constant factor lookup performance of the Chord protocol. However, with higher k comes increased memory consumption. In practice, k is generally kept at or near m.

r is the maximum size of the successor list. This list maintains references to the r most interesting or most used nodes that an agent may know. In practice, the successor list is actually used more than the finger table, as it includes the same nodes as in the finger table, plus additional nodes up to the limit r. The successor list is sorted, so searches can sometimes be more efficient using the successor list, however the finger table must always be populated as best as possible to maintain the Chord protocol guarantees of $O(logN)$ lookup time. In this respect, the finger table acts as the absolute authority of what nodes are considered important, whereas the successor list is consulted more frequently for normal search duties.

The size of the successor list, r, is generally a small multiple of m. This is done to maintain the logarithmic memory expectation for Chord agents, and to reduce the runtime space used during simulations. During churn situations, the successor list can become full very quickly, and for larger values of r, storing and using the successor list dominates both memory and processor time.

3.1.8 Simulation Variables. This section briefly describes those variables that are of primary importance to the simulator, and not intrinsic values that can be easily controlled in RC-Chord.

size is the target number of agents for the network. This number of agents will be created initially within the simulations, but this number may not necessarily be maintained in churn situations. The *size* variable is used to allocate memory internally to speed processing in certain instances and to identify file naming conventions within the logging and data instrumentation subsystems.

num_messages is the maximum number of simultaneous messages in the system. This is included to reduce the total memory used during simulations. The traffic generator is tuned to stay at or near this maximum, if possible, to ensure that as much work is being performed within the agent lookup system as possible, while staying below process memory limits.

churn is the static churn rate measured in percent. For example, a churn rate of 0.1 is 0.1%, or a 0.001 multiplier of the current network size. Obtaining real statistics about just how much churn is reasonable can be difficult. However, for simulations, setting this variable much above 5.0 generates large workloads on the system. Also, for a system of one million agents, a churn amount of 50,000 agents per step may be impractical for some applications.

churnstep is the second tuning parameter for the static churn subsystem. This is the simulation time step duration used to incur the above static churn amount. For each *churnstep* time units, *churn* percent of the agents in the system will be removed, replaced with an equal amount of new agents. The new agents will not have the same properties as those that just left, and RC-Chord applies its self-organization protocols to place the new agents into proper locations within the hierarchy.

businesschurn represents the ability of the simulation to perform network churn at times, and with appropriate magnitude, to model a standard business system where users log on in the morning, and log off in the afternoon. The business churn subsystem follows a mixture of several distributions that model a typical day in the churn of a business class network: large in-flux of nodes to the network around 0800, and departures around 1700.

When using business churn, the *churnstep* value should be set in accordance with the established realtime step ratio. This ratio is calculated at runtime, and establishes a mapping between the scheduled simulation length and a 24-hour business churn time model. The business churn subsystem dominates scaling of the simulation, wherein one time step usually represents one second. Care must be taken when configuring the business and static churn models together, to avoid unwanted bursts of agents joining and parting the system, and to create the desired effect for the application design being investigated.

maxhopcount is the number of hops a message takes before the system decides the message will not reach its destination. This simulation variable is used to prevent infinite loops of messages spinning back and forth between agents. It does happen that, during times of high churn, the system is not stable enough to resolve all nodes [86]. During these times, messages may become trapped in infinite loops. To properly capture this phenomenon for data analysis, the *maxhopcount* variable is used to specify how many hops should pass before stopping the message's transactions. This variable is usually set to a small multiple of m because the expected lookup time for a node in a Chord system is logarithmic in the address width, m.

threads specifies the number of threads to run during the simulation. Only certain parts of the simulation are multi-threaded, and so the threads only execute those portions of the simulation. These areas include cluster maintenance, task allocation by coalition formation algorithms, and task execution steps.

The *transport* delays specify the number of steps between message passing. For each message that is submitted to the simulator framework, a step duration is chosen from a uniform distribution on [*mindelay*,*maxdelay*] milliseconds. This duration must pass before the message is delivered to the target. Depending on the resolution of the simulation time system, these delays can be on any order of magnitude, but are generally in the low tens of milliseconds for Local Area Network (LAN) simulations, or hundreds of milliseconds for Wide Area Network (WAN) simulations.

3.2 Coalition Formation Methodology

The following discussion introduces the Distributed Likelihood of Execution (DLOE) algorithm, which is used to solve the cooperative coalition formation problem. The algorithm is built upon the RC-Chord structured HP2P overlay. It leverages RC-Chord's properties to construct coalitions based on speed of coalition formation and projected efficiency of task execution. These objectives require an understanding of how RC-Chord is organized and what properties it provides, as well as understanding the task and scheduling models used to define and execute tasks.

3.2.1 Coalition Formation Problem Definition.

Coalition formation focuses on the construction of teams of agents to execute tasks, with the goal of employing the capabilities and assets of under-utilized agents to achieve larger and more sophisticated tasks. A task is defined as a function, with a desired end state, that requires one or more agents and resources to complete.

Forming optimal coalitions requires input from each agent in the system, and is an $\mathcal{NP}$-complete problem [83]. As defined by Abdallah and Lesser [3], consider the set of tasks $T = \langle T_1, T_2, ..., T_q \rangle$. Each task T_i is defined as $T_i = \langle u_i, rr_{i1}, ..., rr_{im} \rangle$, where u_i is the utility gained for accomplishing task T_i and rr_{ik} is the amount of resource k required by task T_i. The set of agents is defined as $I = \{I_1, I_2, ..., I_n\}$, where each agent $I_i = \langle cr_{i1}, cr_{i2}, ..., cr_{im} \rangle$, and cr_{ik} is the amount of resource k possessed by agent i.

The coalition formation problem is then defined as the allocation of the subset of tasks $S \subseteq T$ to agents that maximizes the global utility, U,

$$U = \sum_{i|T_i \in S} u_i. \tag{3.3}$$

The task allocation algorithms are charged to build a set of coalitions $C = \{C_1, ..., C_{|S|}\}$, where $C_i \in I$ is the coalition assigned to task T_i, such that each task coalition provides enough resources of each type to satisfy that task's requirements.

$$\forall T_i \in S, \forall k : \sum_{I_j \in C_i} cr_{j,k} \geq rr_{i,k}. \tag{3.4}$$

A constraint on the problem is that each agent is capable of only executing a single task:

$$\forall i \neq j : C_i \bigcap C_j = \emptyset. \tag{3.5}$$

This classical form of the coalition formation problem assumes single-task robots [34], "all or none" resource allocation, and exponential time coalition formation due to task group enumeration [74]. These properties will be modified and extended to define the cooperative coalition formation problem examined here.

3.2.2 Cooperative Coalition Formation. The objective of the coalition formation problem is to maximize the global utility across all task coalitions. The cooperative coalition formation problem extends this definition by assigning a different meaning to global utility.

Achieving global knowledge, and thus global coordination, in a large-scale system is thought to be intractable [51]. As such, the scale of the target systems motivates a new global objective: ensure that each agent is doing something, rather than ensuring each agent is doing something useful. This objective is based upon the observation that global coordination is difficult or impossible to achieve, and so a measure such as global utility may be impossible to accommodate. Therefore, each agent should minimize its time spent negotiating coalitions, and instead focus on contributing as much useful work to the system as possible. Agents that are part of multiple task coalitions are then capable of making local task scheduling decisions so as to maximize global utility. Clearly, agents in cooperative systems must be considered trustworthy enough to choose decision paths that benefit the system before themselves.

Agents seek to accomplish the greatest amount of work over time. Work is defined as the number of units of processing performed per unit time over the entire system, and each agent is capable of executing at most one unit of work per unit time. This concept is based on the observation that maximizing work throughput reduces mean task duration, and leads to a definition of the task model.

3.2.3 Task Model. Tasks used in experimentation are designed to accommodate the diversity of real time tasks expected to be executed within a large-scale agent network. In particular, each task has a quantity of work to complete, a list of required resource types, and a required total quantity of each resource.

Tasks are redefined as $T_i = \langle w_i, p_i, s_i, rr_{i1}, ..., rr_{im} \rangle$, where w_i is the number of units of work necessary to complete task T_i, p_i is the task priority, and s_i is the task synchrony.

Each task, T_i, specifies an amount of work, w_i, that must be performed to complete the task. Experiments performed in this body of work seek to validate the model and task execution process. Each task is assigned an amount of work that follows a uniform distribution, according to the test matrix in Section 5.1.2. The work executed per unit time is compared to the workload generated, and serves as the primary forcing function for the system. Each agent of a task coalition is capable of contributing one unit of work per unit time, and the work for a task may be distributed across multiple agents.

Task priority, p_i, provides a mechanism for runtime tuning, as well as scheduling fidelity. Task priorities for this model fit within a range of [1,10], with 10 being the highest priority. These simulations place most task priorities at the median priority in the middle of the priority range. A second, lower probability, task priority distribution generates high priority tasks. This is meant to reflect the applied nature of this research, namely the application to corporate Command and Control (C2) and defense systems. In such systems, a standard workload of median priority tasks is executed during standard operations. However, occasionally high priority tasks will surge into the system, caused by threats, attacks, or other stimulus. These higher priority tasks are required to execute more quickly than standard tasks to achieve real time mission objectives.

Not all tasks are completely parallel [56], and such distributed processing tasks require periodic barrier synchronization points. These synchronization points halt processing on all agents that have reached the barrier while any other agent has not yet arrived. These synchronization points are sometimes created artificially to verify accuracy, integrity, security, or logging. However, most often barriers are caused by bottlenecks in the distributed algorithm or data set. Coalitions will likely execute some algorithms which experience these bottlenecks, and therefore the task model must accommodate them. As such, each task is assigned a task synchrony value, s_i, that specifies the number of steps each agent can perform before reaching a barrier. Simulations exercise this property, varying the synchrony amongst several low values. The low values represent more frequent synchronization points, and will yield lower overall distributed performance, and thus serve as more interesting data points. Purely parallel algorithms, or those without synchronization points, are also examined to construct baseline statistics for comparisons. In these experiments, task synchrony is said to be disabled.

An example of a task that requires no synchronization is scanning a hard disk. Multiple agents can be assigned to this task, and each can function independently by scanning a portion of the disk. However, any processing that requires the disk to be fully scanned requires a synchronization point, as the tasks performing the scans may not all complete at the same time.

To accommodate the allocation of an agent's resources to multiple coalitions, the following extension is made to Abdallah and Lesser's model:

$$\forall I_j \in I, \forall k : \sum_{T_i \in S} cr^i_{jk} \leq |cr_{jk}|. \tag{3.6}$$

In Equation 3.6, cr^i_{jk} is the portion of agent j's supply of resource k that is allocated to coalition i, which is not to exceed the agent's total supply of resource k, defined by $|cr_{jk}|$. This limits the amount of total resources that an agent may obligate. That is, each agent may only obligate up to the maximum amount of each resource k that it possesses, but may split the allocation between multiple coalitions. This allows each agent flexibility to participate in multiple coalitions simultaneously. It also increases the satisfiability of agent coalition formation by allowing partial amounts of resources, where applicable, to participate in multiple coalitions. This alleviates the "all or none" approach to resource allocation, by allowing tasks to receive only the amount of each resource they need.

To maximize performance, each agent is permitted to contribute only one resource (or partial resource) to a task. This eliminates the possibility of contention in task scheduling, ensures minimum time between barriers, and is a reasonable assumption for a large-scale system. During simulations, tasks choose which of the available resource types to include as requirements. The number of resources, and the quantity of each, follow a uniform distribution. This allocation design seeks to create diversity in task allocation decision making, and to exercise resource location algorithms within the overlay.

The cooperative coalition formation problem is defined as the allocation of the subset of tasks $S \subseteq T$ to agents that maximizes:

$$W(t) = \sum_{i \in I_i} w_i(t). \tag{3.7}$$

Note that the global work throughput, W, is not resolved with the formation of task coalitions. Rather, W is time-dependent, and increases with the number of agents that are able to execute a unit of work per time. Coalition formation algorithms must therefore focus on the allocation of tasks to agents such that the number of agents per task is globally balanced. This balance reduces the workload on some agents, instead placing it on under-loaded agents, and increases the total system work throughput.

To maximize work throughput, and to accommodate tasks with multiple priorities, agents must be capable of processing multiple tasks simultaneously (via internal scheduling). The work throughput is thus dependent on how tasks are executed on each agent.

3.2.4 Scheduling. Each agent is capable of receiving and processing multiple tasks. Each task has a quantity of work that must be executed to complete the task's processing. For simplicity, each agent may only execute a single unit of work per unit time. Agents may choose which task to execute from those tasks they possess. In the event that an agent must choose which among a list of active tasks to execute, it uses a simple priority based scheduling algorithm. Scheduling follows a simple lottery scheme designed to facilitate execution of arbitrary priority distributions. Under this approach, each task is given a number of tickets proportional to its priority. Thus higher priority tasks receive more tickets, and have a higher likelihood of being selected to receive processor time. The sum of these tickets per agent is called the agent's Total Priority Points (TPP). Once all tickets are created, a ticket is chosen at random, and the task to which that ticket belongs is executed for one time step. This simple scheduling algorithm ensures fairness and progress, while implicitly deconflicting between two optimization parameters: number of tasks on the agent, and their priorities. Incorporating more sophisticated scheduling is an area for future work.

Figure 3.4 shows the task priority distribution for simulated tasks. Tasks priorities are assigned based on this bi-modal distribution. The dominant distribution has a mean of 3.5, and represents the creation of tasks during normal operations. Because this system is meant to follow the roles of corporate operations, to include network defense, a second distribution models the low probability occurrence of high priority events. Such high priority events represent real world situations that may require a sudden burst of additional computing power. Tasks assigned into this category are centered around priority eight, and represent a small percentage of all tasks.

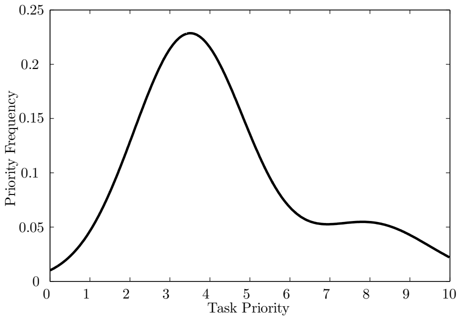

Figure 3.4: Simulation task priority distribution. This is a bi-modal mixture model of two Gaussian distributions, one for standard operating missions, and a second for low probability, high priority missions.

When allocating agents to a new task, the DLOE algorithm examines the task priority of the new task, p_n, and compares it to task priorities of the agents it investigates. With knowledge of the task scheduling algorithm on each agent, the coalition formation algorithm attempts to maximize the likelihood that the new task will receive processor time on each agent. The algorithm assigns points to each task, T_i, resident on a target agent, I_j, based on the priorities of those tasks:

$$priority\_points = \sum_{T_i \in I_j} priority(T_i) \tag{3.8}$$

The value of each agent's *priority_points* is compared, and the agent with the lowest value is assigned the new task. This ensures minimum competition for processing time for the new task on the target agent, thus maximizing its likelihood to execute, and reducing the task's total execution duration.

3.2.5 Coalition Formation in a Large-Scale Environment. The goal of building a solution to the large-scale cooperative coalition formation problem is to facilitate the usefulness of distributed processing in the CyberCraft project. The CyberCraft network is expected to have one million, or more, agents cooperating together to achieve a set of mission objectives. A second goal is to develop a suitable process to create coalitions of cooperative agents in a large-scale Distributed Multi-Agent System (DMAS). The approach centers around two elements: relaxation of the formal coalition formation problem, and development of a suitable heuristic to construct coalitions of CyberCraft.

Much of the existing research into coalition formation does not scale beyond at most a few hundred agents. The theoretical complexity of $O(2^N)$ is an artifact of the definition of the problem: the number of steps it takes to find the optimal configuration for N agents into c coalitions, where $c > 1$. This problem is $\mathcal{NP}$-hard, and assumes an all-to-all coordination mechanism to form coalitions. An underlying assumption in this problem formulation is that agents are self-interested, and therefore the only method to guarantee that an agent receives its best coalition partner(s) is to evaluate all possible coalition formations. However, the CyberCraft project introduces an important restriction which will be exploited in the creation of a large-scale coalition formation algorithm: the agents are cooperative. This cooperation specifies that all nodes are more interested in finding a suitable large-scale solution that benefits the group as a whole. Because such large-scale systems implicitly prohibit the possibility of global knowledge, the agents must seek to satisfy the greatest common utility gain with what little information they have. Since tasks arrive at runtime, an agent cannot possibly predict what tasks are to come, and therefore must also restrict its decision making to the little information it has at any point in time. These restrictions aid in the creation of a large-scale coalition formation algorithm.

With this cooperation among agents, it becomes possible to introduce a heuristic which forms the basis of an agent's decision making. Like search in a large tree, each decision point is based on limited data, but can still yield near optimal solutions. The size of the CyberCraft target network precludes the possibility of globally optimal solutions, but agents can instead optimize the solution to the cluster level. Given appropriate tuning parameters, clusters can be formed such that internal (intra-cluster) broadcast, or maintaining a shared database, can yield near optimal local coalitions. As each cluster can be viewed as a node to a higher level cluster, this pattern can be reproduced at increasingly higher levels, yielding near optimal global coalition formations.

This solution can be viewed as building upon the task allocation problem: given a task T which requires p processing units and r resources, build a coalition of available and willing agents that will solve T in the shortest possible time. This contribution provides a mechanism for large-scale systems to organize themselves into working structures that can achieve one or more simultaneous tasks, although perhaps not optimally.

3.2.6 Distributed Likelihood of Execution (LOE) This solution to the distributed task allocation or coalition formation problem is called DLOE. The DLOE algorithm attempts to achieve maximum work throughput by forming coalitions for work tasks. Since the algorithm is distributed and designed to operate in large-scale systems, it does not use global knowledge.

Each node in the system periodically passes its resource amounts and TPP (Section **??**) to one of its cluster's super peers. This information is collected and the TPP is tracked according to resource amount. For example, in a system that permits resource amounts [1,1000], the resource interval is split n times. For each of these n equally divided intervals, the count of TPP for the cluster is tracked[1]. Along with

[1] It is possible to divide these intervals based on runtime usage patterns. This idea is discussed in Chapter VI.

the number of nodes in the cluster, again divided by resource interval, this vector of TPP per resource interval is all that is passed upward between clusters. This information continues an upward ascent through the levels of the HP2P network until it reaches the super-cluster. The DLOE algorithm uses this information to build an approximate picture of the TPP for each sub-graph. When searching for agents to satisfy resource requirements for coalition formation, the algorithm employs a simple heuristic to determine whether or not to continue searching for more optimal solutions.

The data collection and dissemination accounts for a small amount of overhead. Given 10 intervals on a 64-bit architecture, the information passed from a cluster to its next higher level cluster consumes approximately 160 bytes. A system of one million agents, with 1000 agents per cluster, will have roughly 1000 clusters. This entire periodic maintenance process therefore consumes approximately 160kB per update period for a large-scale system. Cluster super peers store the TPP data for members of their cluster, updating as new information becomes available. This results in a maximum of $num\_intervals * 2^m$ entries stored per super peer, or 82KB of memory on a 64-bit machine with 10 resource intervals and a maximum of 1024 agents per cluster.

3.2.7 Distributed LOE The DLOE algorithm attempts to achieve maximum work throughput by forming coalitions for work tasks. Since the algorithm is distributed and designed to operate in large-scale systems, it does not use global knowledge.

3.2.8 DLOE Algorithm Design. Tasks may be introduced to the system by any agent, at any time. When an agent introduces a task to the system, the primary objective for building a task coalition is to locate agents that can satisfy the task's requirements. For resources inside of the source agent's cluster, Algorithm 1 is applied directly. However, to satisfy requests for other resource types, the task information is forwarded to the super-cluster, where the distributed agent location algorithm is then executed.

Algorithm 1 receives a task to be allocated, along with the resource types and quantities to allocate, and the cluster [identifier] of the agent that produced the task. The algorithm is shown as recursive, although the actual distributed implementation can be either iterative or recursive. The entry point into the search is the supercluster, which can be reached by any agents in the system in O($log_m(N)$), where m is the Chord address width per cluster. The objective of the search is to locate agents capable of satisfying resource requirements for the new task.

Algorithm 1 build_coalition(T,$R = \{r_1, r_2, ..., r_n\}$)

```
for $i$=1 to $length(R)$ do
    $C \leftarrow C \bigcup chooseNodeRecursive(this\_cluster, r_i)$
end for
```

Algorithm 2 is the core process for choosing from which cluster to allocate a node. Note that all nodes in the system that satisfy the resource requirements are capable of receiving another task. However, the goal of this algorithm is to allocate tasks to agents such that those tasks have the best opportunity to execute. Determining the task's Likelihood of Execution (LOE) is a function of the task's priority and the priorities and quantities of tasks located on the agents being surveyed.

LOE is calculated as follows. The algorithm begins by calculating the cluster's TPP. This process is executed on a super peer, and therefore the agent has access to such information. The total priority points feed into the agent's task scheduling system described in Section 3.2.4. Minimizing the total number of competitors (each of whom has one or more lottery tickets) will minimize the competition's likelihood of winning a processor time unit. The algorithm examines the LOE for the task against all agents in the current cluster. This value is computed at the super peer without further inter-agent communications, and is compared to the LOE for the sum of all sub-clusters from the current cluster. A lower LOE value is more desirable, as lower values indicate less competition in the task scheduling algorithm. If the LOE of the current RC-Chord cluster is higher than that of one or all of its sub-clusters, then the algorithm moves to the best sub-cluster from the current point. At that next cluster,

Algorithm 2 chooseNodeRecursive($cluster, r_i$)

```
$clusterTPP = cluster.localTPP + cluster.numNodes$
$subNodes = cluster.subNodes(r_i)$
$subTPP = max(int)$
**if** $subNumNodes > 0.0$ **then**
    $subTPP = cluster.subTPP(r_i)$
**end if**
**if** $clusterTPP \leq subTPP$ **then**
    $bestAgent = choodeAgentLocal(cluster, r_i)$
**end if**
$cluster[]subClusters = getSubClusters()$
$subBestTPPSoFar = max(int)$
$clustersubBestSoFar = nil$
**for** $i = 0 : subClusters.length$ **do**
    $subTest = subClusters[i]$
    $subTPP = subTest.totalTPP(r_i)$
    $subNodes = subTest.nodes(r_i)$
    $subLOE = subTPP/subNodes$
    **if** $subLOE < subBestTPPSoFar$ **then**
        $subBestTPPSoFar = subLOE$
        $subBestSoFar = subTest$
    **end if**
**end for**
**return** chooseNodeRecursive($subBestSoFar, r_i$)
```

the algorithm repeats, continuing until the LOE of the current cluster is better than any of its sub-clusters. If no sub-clusters exist, then the algorithm has reached the lowest level of the RC-Chord hierarchy, and a node from the current cluster is chosen.

Once the task coalition is formed, agents contribute units of work to the task based on their internal scheduling system. Upon reaching a task synchronization barrier, an agent halts processing on that task until all other agents assigned to the task reach that barrier. During that time, the agent removes the task from the ready queue, and instead executes work for other task coalitions of which it is a member. Once the synchronization barrier has been met by all other agents, the task becomes ready to execute by all agents agents in the task coalition.

Implicit in the decision making of the DLOE algorithm is that all holders of a resource are considered equal in quality, although perhaps not quantity. Likewise, agents contribute equal work units to each task they host. These assumptions establish a balance among agents, and reduce the search space of the DLOE algorithm.

Contrary to most coalition formation algorithms, the agents under this problem definition have no decision authority about which task coalitions they join. Rather, the distributed algorithm decides the task allocation strategy that best benefits the system work throughput. The DLOE algorithm borrows from the military command paradigm: centralized authority, decentralized execution. The algorithm itself is distributed, however the authority it carries is centralized in the sense that agents are less autonomous than other approaches. The objective of this paradigm is to minimize negotiations resulting in coalition formation, thus reducing the overhead of the algorithm, and yielding higher system work throughput.

Each node in the system periodically passes its resource amounts and TPP to one of its cluster's super peers. This information is collected and the TPP is tracked according to resource amount. For example, in a system that permits resource amounts [0,1000], the resource interval is split n times. For each of these n equally divided intervals, the count of TPP for the cluster is tracked. Along with the number of nodes in the cluster, again divided by resource interval, this vector of TPP per resource interval is all that is passed upward between clusters. This information continues an upward ascent through the levels of the HP2P network until it reaches the super-cluster. The DLOE algorithm uses this information to build an approximate picture of the TPP for each sub-graph. When searching for agents to satisfy resource requirements for coalition formation, the DLOE heuristic chooses the route that minimizes the expected scheduling contention by examining the TPP at each step. Finding the node with minimum TPP will yield the highest likelihood of execution for the new task and minimize its expected task duration.

The data collection and dissemination accounts for a small amount of overhead. Given 10 intervals on a 64-bit architecture, the information passed from a cluster to its next higher level cluster consumes approximately 160 bytes. A system of one million agents, with 1000 agents per cluster, will have roughly 1000 clusters. This entire periodic maintenance process therefore consumes approximately 160kB per update period for a large-scale system. Cluster super peers store the TPP data for members of their cluster, updating as new information becomes available. This results in a maximum of $num\_intervals * 2^m$ entries stored per super peer, or 82KB of memory on a 64-bit machine with 10 resource intervals and a maximum of 1024 agents per cluster.

3.2.9 DLOE Configuration Variables. The DLOE algorithm relies heavily upon the function of RC-Chord. As such, the tuning parameters used to configure RC-Chord directly impact the effectiveness of the DLOE algorithm. In addition, the DLOE algorithm also includes two other tuning parameters: $number\_of\_intervals$ and $level\_update\_interval$.

The $number\_of\_intervals$ variable specifies how many intervals to generate for the range of resources available to the system. RC-Chord defines how many resources the system possesses, and the DLOE divides this range into smaller intervals to improve task formation quality. An agent's TPP characteristics are placed into a vector of TPP for its cluster. The range of resource amounts is divided by *number_of_intervals* to determine how many elements the vector tracks. The amount of a resource that an agent possesses is used as an index into the TPP vector, and that value is incremented by the amount of the resource for that agent. Increasing *number_of_intervals* increases the fidelity of the LOE subsystem, at the expense of slightly increased memory and bandwidth consumption.

The *level_update_interval* variable specifies how often, in simulation time steps, that the above TPP and number of nodes vectors are collected and transmitted up the RC-Chord hierarchy. Specifying lower values for *level_update_interval* causes the system to maintain LOE data that is closer to realtime, at the expense of increased bandwidth consumption. This value is used to tune the system based on expected workload.

3.2.10 Summary. The RC-Chord HP2P structured overlay system provides reliable large-scale communications for use in C2 applications. The ability to locate agents by resource is a critical component in the development and execution of the DLOE cooperative coalition formation algorithm. This algorithm seeks to locate the best agents in the system to join new task coalitions. The objective of the combined system is to maximize work throughput across the system, paying special regard to the task model, which identifies the priority of each task to be executed.

IV. RC-Chord Experiments and Analysis

This chapter describes the experimental objectives, configuration, and results of testing Resource Clustered Chord (RC-Chord). The analysis begins with a validation against the baseline Chord system, and moves forward to examine the effects of scale on RC-Chord's runtime properties. The chapter ends with a summary of the design methodology and experimental results.

4.1 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 [46] Peer-to-Peer (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.

4.1.1 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 Hierarchical Peer-to-Peer (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 possibil-

Table 4.1: 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

ities 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.

4.1.2 Process Variables. The RC-Chord experiments exercise several process variables. These variables, shown in Table 4.1, are chosen to configure the system according to the baseline Chord protocol.

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 will distribute the agents into multiple clusters at multiple levels.

The number of agents in the system varies between 1000 and one million agents. CyberCraft requires the capability to support one million or more agents simultaneously, and so the larger systems are examined in more detail than the smaller systems. However, trend data across this spectrum of network sizes serves to establish an understanding of the scalability of the system, and how RC-Chord's properties vary with scale.

The static churn rate will vary 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. This churn rate far exceeds the expected instability to be found in a business class system such as CyberCraft, but serves to further validate RC-Chord's resilience under high flux or failure conditions.

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 [16, 59]. 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 bi-modal 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. Figure 4.1 shows the business churn model distribution. The system is initialized with one million nodes at time step zero, or noon simulated time. The departure of agents begins at approximately 1630, lasting until 1730. The time lapse scale shows the arrival of agents the next morning.

4.1.3 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.

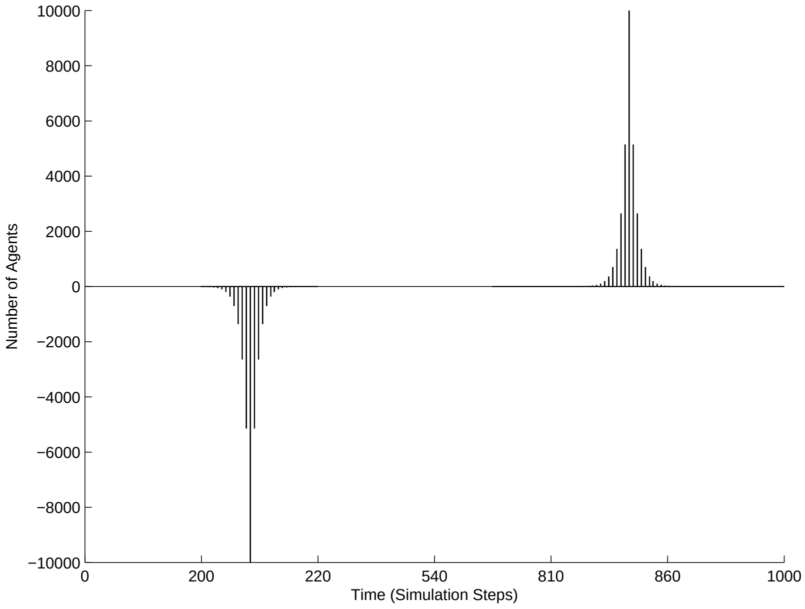

Figure 4.1: Time lapse illustration of the business model churn distribution. This stimulus consists of two Laplace distributions. The first distribution describes for the departure of agents from the network at the end of the business day, and the second models the arrival of new agents to the system at the beginning of the work day.

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.

4.2 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 4.1 is followed to exercise the RC-Chord protocol.

4.2.1 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 4.2 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.

The baseline agent lookup failure rates for RC-Chord systems, with business churn disabled, are shown in Figure 4.3. These baseline systems set $m = 32$, so all

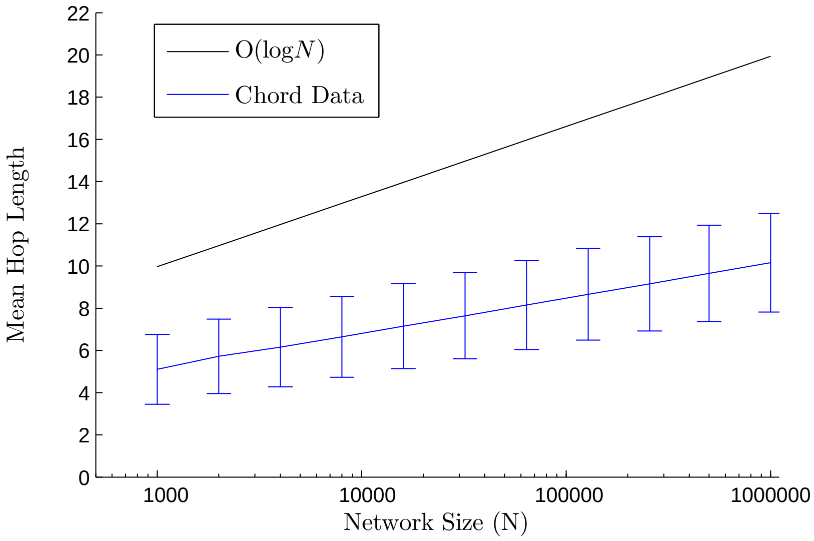

Figure 4.2: Chord baseline validation for mean hop length. The plot of $O(\log N)$ is the expected upper bound for the mean hop length in a pure Chord system. The results obtained via experiments sit below the upper bound, partially validating the RC-Chord protocol.

agents are in a single cluster, and vary the level of static churn. A static churn rate of 0% yields 100% agent lookup success rate. The data demonstrate that higher churn rates negatively affect the ability of the system to resolve agents. The magnitudes of these results are consistent with other approaches [62], and help to validate the Chord agent lookup failure model.

These experiments with RC-Chord in baseline configuration demonstrate the accuracy and effectiveness of the model, as measured against the standard Chord protocol. The results successfully validate the basic RC-Chord protocol and implementation, building a foundation upon which to conduct further testing.

4.2.2 RC-Chord Experimental Results. Figure 4.4 shows agent lookup costs for networks of size varying from 1k to one million agents, in a system with 5 resources, business churn enabled, and 1.0% static churn. Overall the trends are similar, with an increase in hop length in lower address widths. Once the address width for each test reaches logarithmic size in the number of agents, the hop length stabilizes to the results closer to those obtained by a pure Chord system. This decrease in hop length

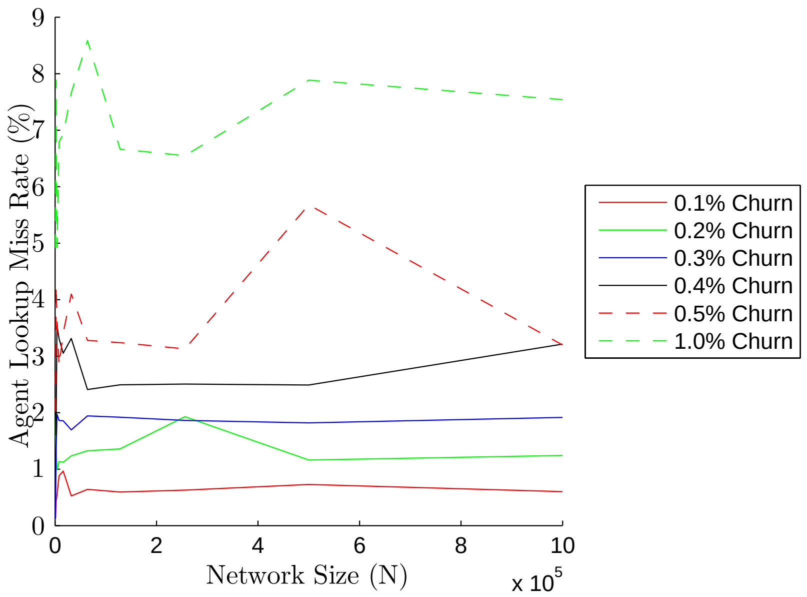

Figure 4.3: Agent lookup failure rates for systems with varying levels of static churn. Business churn is disabled for these experiments. The system size varies from 1000 to one million agents, and $m = 32$.

occurs because the address widths below logarithmic size in N force the creation of new clusters at lower levels. Even though an address width of $O(log(N))$ forces a small percentage of nodes into a lower level cluster, it is not until a significant portion of nodes are forced to lower levels that the overall hop length for the experimental sample will begin to increase.

The effects of the business churn model alone are shown in Table 4.2. This table describes the mean hop length results for systems with 0% static churn, varying $m \in [10, 20]$, and varying the network size from 1000 to one million agents.

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

Table 4.2: Mean hop length versus network size and address width for system with static churn disabled and business churn enabled.

	Network Size										
m	1000	2000	4000	8000	16000	32000	64000	128000	250000	500000	1000000
10	4.99752	5.08002	5.119	5.35603	6.85287	8.11699	9.48132	11.6409	13.3801	13.9596	13.5162
σ	1.57701	1.74935	1.83542	2.25947	3.1978	3.16885	3.00451	3.84667	3.64318	3.2353	3.14477
11	4.9325	5.35428	5.58119	5.5656	5.65853	7.44273	8.89424	10.1036	11.5039	14.0019	14.9662
σ	1.60348	1.67805	1.73475	1.83013	2.16687	3.38951	3.49251	3.02065	3.8646	3.85187	3.36746
12	4.98608	5.54371	5.92334	6.05036	5.98538	6.17674	8.14088	10.0426	10.2973	13.1574	14.6662
σ	1.69202	1.7328	1.73626	1.83089	1.84284	2.3515	3.66325	3.59815	3.36063	4.38302	3.83626
13	5.26734	5.5513	5.94909	6.39149	6.55303	6.57246	6.6722	8.33912	10.467	10.7928	13.8274
σ	2.97704	1.79504	1.75775	1.80403	1.94011	2.02557	2.4169	3.40795	3.7753	3.31788	4.1587
14	5.12468	5.604	5.93069	6.47455	6.91345	7.10654	7.1633	7.18759	9.03296	11.0607	11.9766
σ	1.67459	1.75445	1.79535	1.841	1.88339	2.02198	2.08111	2.47541	3.56996	3.77676	3.36768
15	4.99749	5.58401	6.09704	6.54138	7.01605	7.42134	7.5437	7.59703	7.90838	9.265	11.0408
σ	1.67871	1.77076	1.82909	1.89157	1.91906	1.94097	2.06191	2.15657	2.56179	3.38788	3.61402
16	5.11959	5.49659	6.02791	6.51348	7.03852	7.50774	7.94657	8.05592	8.07357	8.41531	10.2297
σ	1.69779	1.71366	1.84087	1.87434	1.93936	2.03342	2.01839	2.18809	2.1652	2.61824	3.90636
17	5.13469	5.5296	6.0746	6.61817	7.08397	7.55152	8.02049	8.43871	8.59464	8.58673	8.93788
σ	1.80038	1.71065	1.79456	1.89112	1.94932	2.01138	2.05355	2.07142	2.24754	2.2413	2.75932
18	5.05897	5.54472	6.10343	6.60046	7.12274	7.61299	8.07075	8.51336	8.94976	9.07028	9.20737
σ	1.70066	1.7799	1.85286	1.90209	1.97447	2.04361	2.15449	2.12769	2.13683	2.2667	2.35396
19	5.17358	5.46293	6.09957	6.66615	7.10322	7.58879	8.11336	8.57762	9.03152	9.44076	9.5973
σ	1.66912	1.78016	1.82419	2.38002	2.20207	2.03608	2.14261	2.15652	2.17956	2.19145	2.36679
20	4.94354	5.51943	6.11935	6.64158	7.13932	7.62813	8.10538	8.59567	9.07131	9.52215	9.93736
σ	1.6252	1.79585	1.86743	1.90542	2.0182	2.06411	2.09848	2.14887	2.23903	2.26818	2.24882

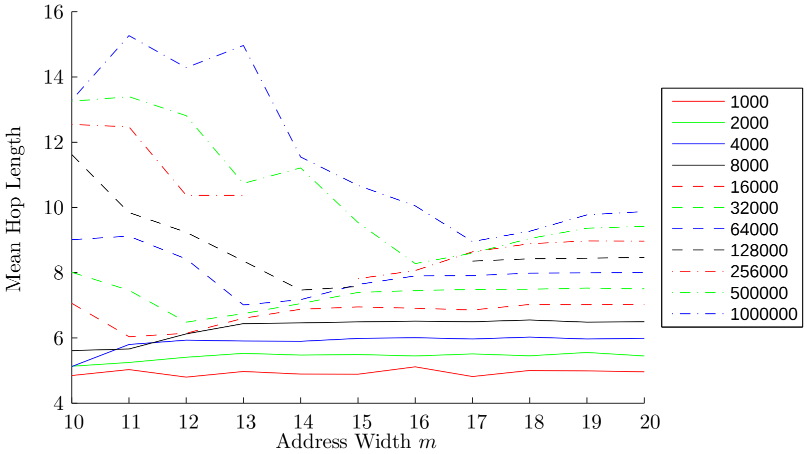

Figure 4.4: Global agent lookups for a system of agents varying in size from 1000 to one million agents. In these experiments, the business class churn model is enabled, and static churn is set to 1.0% per 60 simulation seconds. The X-Axis shows values of the address width, m, and the Y-Axis shows mean number of hops per lookup.

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, and from super-cluster to target.

The full effects of churn rate and address width on mean hop length can be seen in Table 4.3, 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 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

Table 4.3: 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)

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.

Table 4.4 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, as shown in Figure 4.5, 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.

4.3 Summary

The RC-Chord HP2P overlay separates the agents in a network into multiple clusters. Clusters are placed in a downward forming tree, whose branching factor is proportional to the peer to super peer ratio. Each sub-graph from the top level

Table 4.4: 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	**0.00**	4.33	6.20
18	0.00	0.54	1.16	1.74	**0.00**	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

cluster, or super-cluster, represents an individual resource present in the system – all agents within each sub-graph contain the same resource type. Agents may join one cluster for each resource they possess, and are permitted to join a second cluster per resource when accepting duties of cluster super peer.

Baseline experiments validate the expected performance of RC-Chord against the Chord protocol. By varying the size of the network, the address identifier width, and the churn rate, experiments demonstrate the scalability and stability of RC-Chord for large systems. Even in high churn systems, the mean hop length remains stable, and agent lookup miss rates are comparable to those in previous research. Overall, experiments reveal that RC-Chord protocol achieves its objectives to provide reliable and scalable communications in a system of one million agents.

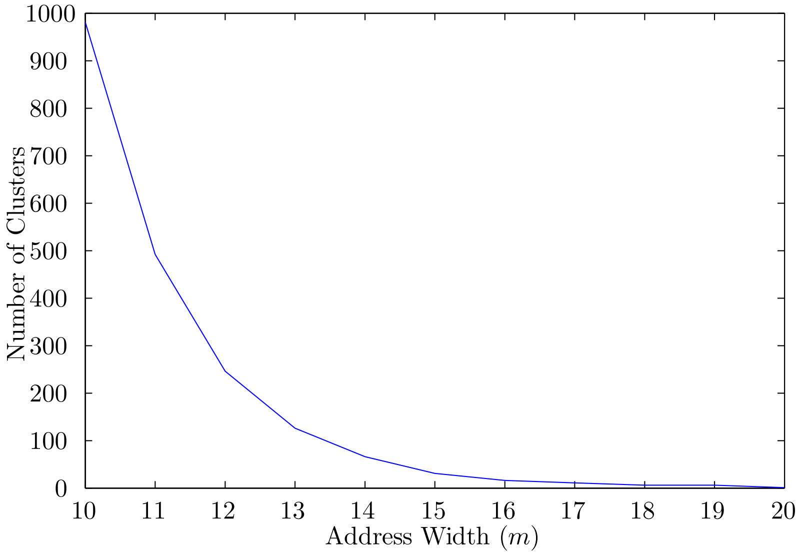

Figure 4.5: Number of clusters for systems of one millions agents of differing address width. The reduction in the number of clusters with increasing m contributes to higher agent lookup success rates in high churn systems.

V. Coalition Formation Experiments and Analysis

This chapter describes the experiments used to validate the correctness and benefit of the Distributed Likelihood of Execution (DLOE) coalition formation algorithm. All experiments are designed to replicate the expected CyberCraft environment, to include the capabilities of its agents. Due to the size of the CyberCraft network, simulations are performed to examine the runtime characteristics of the DLOE algorithm, compared to baseline and optimal algorithms.

5.1 Experimental Setup

Experiments demonstrate the effectiveness of the DLOE algorithm for large-scale Distributed Multi-Agent System (DMAS) coalition formation. The Peersim simulator [46], a Java based simulator engine designed for network and agent simulations, provides the environment for experiments. A complete Chord protocol has been implemented, on top of which Resource Clustered Chord (RC-Chord) algorithms and state have been included. The coalition formation algorithms used in the simulations are built upon the RC-Chord overlay, and experiments evaluate systems of one million simulated nodes.

Experiments last 10,000 time units each, which is suitable time for data trends to become stable. At each time step, tasks are allocated according to the desired load for the simulation. Tasks complete when their required number of work units have been executed by the task's agents. Because each agent completes at most one unit of work per unit of time, if it has a task present to execute, the maximum theoretical workload that a fully loaded system of one millions agents can accomplish is one million units of work per time step.

Agents are cooperative, and seek to maximize global system throughput. Agents will also answer queries honestly, and execute tasks according to the semantics described above (such as task priorities). Agents may only join a task once, since joining more than once introduces contention for processing power.

5.1.1 Coalition Formation Algorithms. These experiments use three coalition formation algorithms:

- Centralized Random Policy (CRP): An agent is chosen at random. If that agent meets the resource task's resource requirements, then it is added to the task coalition [4].
- Centralized Likelihood of Execution (CLOE): Tasks are allocated to agents so as to maximize the probability of the task receiving processor time slots. This centralized algorithm uses global knowledge.
- DLOE: Similar to the CLOE algorithm, except that global knowledge is removed and the algorithm is distributed.

The CRP [4] and CLOE serve as baseline algorithms, with CRP providing a uniform distribution of tasks to nodes, and CLOE attempting to optimize a task's Likelihood of Execution (LOE) with global knowledge. The CLOE algorithm is an adaptation of the Contract Net Protocol (CNP), modified to operate in an environment where agents are required to volunteer. The advantage of global knowledge for CLOE is that no ratio averaging is used in the decision process. Instead, the algorithm can locate the agent in the system with lowest Total Priority Points (TPP) by examining every node at each coalition formation iteration. With this global knowledge, the CLOE algorithm serves as an optimal baseline. The CRP algorithm randomly chooses nodes to include in a task coalition. However, this process can fail as the targeted agent may not have the proper resource type, or may have an insufficient quantity of the resource. As such, CRP consumes additional bandwidth, and is the only algorithm that can miss.

5.1.2 Process Variables. Table 5.1 describes the process variables in these simulations. The objective of testing under these conditions is to exercise the critical parts of the coalition formation algorithms, and examine the resulting effectiveness metrics.

Table 5.1: Simulation Process Variables

Variable	Range
Algorithm	CRP, CLOE, DLOE
Task Synchrony	-1, 1, 2, 3, 4, 5, 10, 15
Load (tps)	500, 1000, 1500
CLOE Update Interval	0, 5, 10, 25, 50

Tasks are allocated uniformly across each simulation time line, with task priorities following the probabilities shown in Figure 3.4. The loading parameters are recorded as tasks created per time step (tps), and experiments demonstrate that they serve as the highest fidelity process variable for evaluating the algorithms. The incoming task work load, or work generated, is measured in units of work per unit time, with each task requiring between 750 and 1000 units of total work to complete. Given the optimal work throughput of the system at one million units of work per unit time, the values of 500, 1000, and 1500 represent, respectfully, under-loaded, critically-loaded, and over-loaded systems. The objective of these values is to measure the effectiveness of each algorithm under consideration in these critical scenarios.

The DLOE level heuristic algorithm has one additional critical process variable: the update interval. The update interval is the duration, in simulation time steps, between updates of DLOE shared resource data (Section 3.2.7). For this algorithm, the update interval is varied among 0, 1, 2, 3, 4, 5, 10, 15. Unless otherwise noted, the level heuristic update interval is kept at five.

During churn conditions, agents may join and part without warning. Since work completed by agents is attributed to the task, the loss of an agent does not cause a task to lose its work completed. Coalitions that lose an agent due to churn reallocate a new agent to fill the departed agent's role. Because the impact of churn on coalition formation is small, and because the introduction of churn would extend testing to nearly a year, the RC-Chord churn rates are set to zero.

Table 5.2: Simulation Control Variables

Variable	Range
Size	1000000 agents
m	12
Task Resource Required	Uniform [750,1000]
Agent Resource Available	Uniform [400,1000]
Number of Resources	5

5.1.3 Control Variables. All experiments simulate a system of one million nodes. Each Chord cluster is given 12 bits of address space, allowing for 4096 nodes per cluster, and a minimum of 244 clusters for a system of one million agents. Each task requires between 750 and 1000 units of a resource. This creates the scenario where some agents will be unable to satisfy the resource requirements of some tasks, and thus those tasks will need to split the resource requirement among several agents.

Each agent is assigned a single resource from a range of five different resources, with the resource quantities evenly distributed between 400 and 1000 units. This introduces sufficient diversity in the system to validate the model by forcing decision making in coalition formation algorithms, generating multiple resource sub-graphs in the topology construction phase, and exercising task generation by varying the number of required resources required per task.

5.1.4 Response Variables. The objective of these experiments is to evaluate the effectiveness of the DLOE heuristic algorithm against the baseline uniform (CRP) and optimal (CLOE) algorithms. Primary measures of this effectiveness are the agent solicitation miss rate, the sustained workload performance of the system as a whole, and the global balanced utilization of resources.

The CRP algorithm chooses nodes to solicit randomly, with global knowledge. Because some agents will not be able to satisfy the resource requirement, either because of insufficient quantity of that resource or incorrect type of resource, this algorithm will tend to miss, or query agents that cannot satisfy its request. These misses become important in a real system because of network bandwidth consumption and time delays. The CLOE algorithm uses global knowledge to carefully choose which agents to recruit into its coalitions, and therefore does not miss. The DLOE algorithm tracks agent availability by resource quantity intervals, and therefore also does not miss.

Workload performance is used to evaluate overall effectiveness because, as experiments show, poor task allocation can result in lower work executed per unit time. The CRP algorithm suffers the most from this, as scheduling restrictions are not considered as part of its decision process. The DLOE algorithm attempts to provide as much of the quality of the CLOE algorithm with respect to workload, but with partial information.

The lack of global information is a result of the scale of the system in which these algorithms operate. With the CLOE algorithm as a model, the DLOE algorithm maintains a shared knowledge of the likelihood of execution at each cluster and level. This information is shared periodically, creating a small amount of bandwidth overhead, and a reduced accuracy of LOE compared to the CLOE algorithm due to delays in updates. However, as experiments demonstrate, this delay is small compared to the scale of the system, and performance measures are favorable.

5.1.5 Noise Variables. The primary noise variable for the coalition formation experiment is tied to resource allocation. Each task is assigned a number of resources and an amount of each, both chosen from a uniform distribution. Agents are also assigned a resource type and amount from a uniform distribution. This variance is mitigated by using the same distribution between experiments.

5.1.6 Experiment Methodology. An RC-Chord instance of one million agents is created, following the attributes listed above. Each experiment lasts 15,000 time units. At each time unit, one or more tasks are created, depending on the desired workload, and the coalition formation algorithm constructs a team to satisfy the task.

Table 5.3: Impact of Task Synchrony on Work Throughput. Shown are work throughput mean and standard deviation for critically-loaded systems of varying task synchrony.

Synchrony Level	DLOE	CLOE	CRP
None	8728.37 (+/- 338.089)	8731.47 (+/- 338.109)	8571.92 (+/- 470.946)
1	8677.34 (+/- 378.728)	8729.38 (+/- 338.036)	6463.49 (+/- 486.299)
2	8678.93 (+/- 370.951)	8731.89 (+/- 338.389)	6503.07 (+/- 489.375)
3	8680.1 (+/- 371.482)	8730.37 (+/- 338.122)	6475.25 (+/- 481.184)
4	8681.89 (+/- 369.162)	8730.69 (+/- 338.194)	6449.54 (+/- 487.638)
5	8683.82 (+/- 366.644)	8731.82 (+/- 338.198)	6405.1 (+/- 475.077)
10	8686.63 (+/- 366.649)	8733.16 (+/- 338.77)	6507.06 (+/- 478.693)
15	8687.2 (+/- 363.776)	8731.08 (+/- 338.208)	6477.08 (+/- 464.304)

In addition, each agent performs one unit of work on a task chosen according to the priority scheduling algorithm described in Chapter III.

Each of the process variables is then modified, according to Table **??**, resulting in approximately 240 individual tests. From these experiments, the data which fully describe the response variables are extracted. These experiments are somewhat exhaustive, and focused to determine the effectiveness of coalition formation in many of the situations that can be expected to occur in a real CyberCraft system.

5.2 Results and Analysis

This analysis seeks to evaluate the effectiveness of the DLOE algorithm by comparing its results to the CRP and CLOE algorithms. The CRP algorithm serves as a baseline, and the CLOE algorithm forms a work optimal algorithm. When appropriate, Analysis of Variance (ANOVA) [71] is shown to characterize statistical similarities between experimental data.

5.2.1 CRP Algorithm. The CRP algorithm proves the worst in coalition formation. This is seen through the poor work throughput reported in Table 5.3, showing the results of running all three algorithms in a critically-loaded system, and varying the task synchrony between one and 15 (synchrony enabled). In all cases,

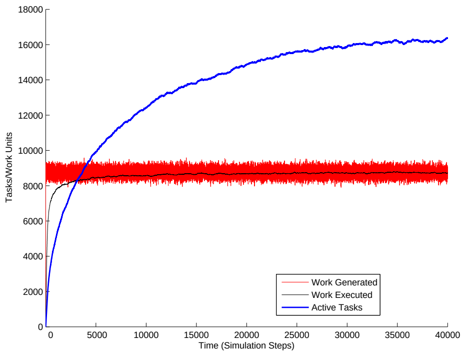

Figure 5.1: A critically-loaded system with task synchrony disabled using the CRP algorithm to allocate task coalitions. The work executed is slow to reach the work generated due to poor task allocation. Once the system becomes saturated enough with tasks, the number of agents executing one or more tasks becomes high, the work executed per unit time improves, and the rate of growth of number of tasks plateaus.

the CRP performs noticeably worse than the other algorithms (ANOVA $p = 0$) and yields higher standard deviation. The coalitions formed by this algorithm experience significant delays due to poorly allocated tasks, and the higher standard deviation shows less stability in the algorithm.

Figure 5.1 reflects the CRP algorithm operating in a critically-loaded system with task synchrony disabled. The incoming workload remains Even without barrier synchronization, the work executed over time for the CRP algorithm's task coalitions is slow to reach a stable limit. Only after a high percentage of agents receive a task to execute does the work executed over time reach the work generated per time. If the task synchrony is enabled for this algorithm in a critically-loaded system, the number of active tasks increases linearly without bound, caused by the work throughput remaining below the work generated. The only experiments in which the work executed meets the work generated for the CRP algorithm are the under-loaded systemS with synchrony enabled, or this critically-loaded system with synchrony disabled.

The hit rate data in Figure 5.2 is roughly the same for all experiments using the CRP algorithm. The hit rate measures the percent of agent query requests by the CRP algorithm that result in finding an agent with sufficient quantity of the correct resource type for the task request. Since the algorithm randomly chooses nodes to

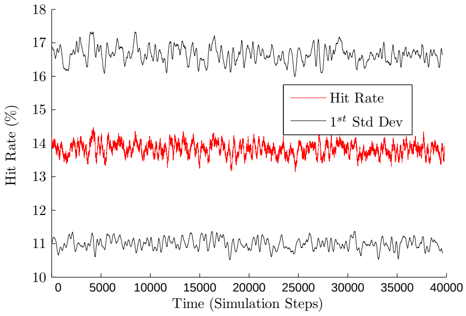

Figure 5.2: The agent query hit rate for the CRP algorithm. The data are equivalent for all experiments performed. The low hit rate is caused by the number of resources present in the system (five), and by the algorithm querying nodes that have insufficient quantity of the required resource to satisfy task requirements.

interrogate about joining coalitions, the hit rate is low, resulting in wasted search time. All experiments used five possible resources, with one resource allocated per node, and this accounts for the majority of the misses. In addition, not all agents possess sufficient quantity of the necessary resource to satisfy the request, which accounts for the remainder of the misses. The CLOE algorithm uses global knowledge, and does not miss. The DLOE algorithm uses a heuristic based on the small tables of resource levels, and also does not miss.

5.2.2 CLOE Algorithm. The CLOE algorithm performs well because it has global system knowledge. Using this knowledge allows it to achieve a zero miss rate on node queries, as well as choosing the best nodes to allocate to coalitions. This is generally accomplished by choosing the nodes that most minimally meet the resource requirements, combined with the least amount of competition for task scheduling. The algorithm achieves a stable number of tasks almost immediately for all under-loaded and critically-loaded systems.

The critically-loaded data in Table 5.3 are identical for the CLOE algorithm under any synchrony, and the DLOE algorithm with synchrony disabled. None of the algorithms achieve a balance between work load generated and executed for over-loaded systems, and the number of tasks increases linearly over time in those experiments.

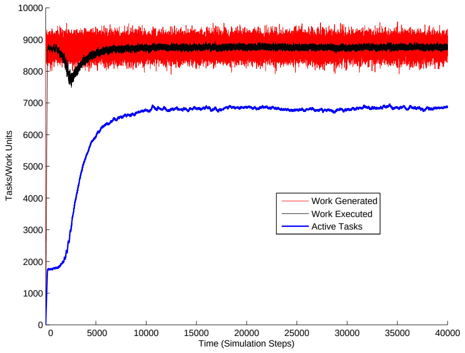

Figure 5.3: Critically-loaded DLOE system with task synchrony enabled. The work executed is roughly equal to the work generated, and this results in a stable number of tasks over time.

5.2.3 DLOE Algorithm. A critically-loaded system is shown in Figure 5.3 in which the DLOE algorithm allocates tasks to agents with barrier synchronization points enabled. The results are characterized by a stable number of tasks over time, where the amount of work executed is the same as the amount of work generated. The data show a slight dip in work executed at the time when the first tasks are being completed. With a mean of 875 units of work per task, this initial pause in the growth of the number of tasks occurs at slightly before time step 1000. Once this time threshold is reached, the number of tasks is held steady and the work executed per time drops slightly. The steady state for this system reflects a very close correlation between the work execution and work generation rates because the number of active tasks continues to climb. Once the work execution rate reaches its maximum, the number of tasks levels off, and the system establishes its steady state.

5.2.4 Comparison Analysis. Figure 5.4 shows the work throughput of the three algorithms on the over-loaded scenario with task synchrony disabled. Both the CLOE and DLOE approach the practical maximum of 10,000 units of work per unit time, which matches the test scenario work creation rate. The CRP algorithm is unable to reach this milestone, as it will always allocate more tasks to agents with higher resource amounts than those with lower amounts. This is one of the benefits of both the CLOE and DLOE algorithms: they try to allocate tasks that most minimally

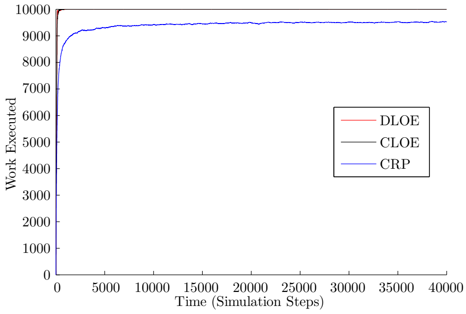

Figure 5.4: Comparison of the three algorithms in an over-loaded system with synchrony disabled. The work throughput of the CLOE and DLOE is comparable, whereas the CRP algorithm performs noticeably worse.

meet task resource requirements. Without any heuristic, the CRP algorithm simply accepts the first agent that can satisfy a task's requirements, regardless of the excess resource amounts possessed by the agent.

Figure 5.5 shows the number of tasks allocated per agent for the three algorithms in a critically-loaded system with synchrony disabled. Agents in the systems with the CLOE and DLOE algorithms have a lower number of tasks per agent as a result of the higher overall work throughput. Since tasks complete more quickly, the agents are able to satisfy the incoming workload more easily, and thus have fewer tasks. The CRP algorithm has a lower work execution rate, and therefore maintains more tasks per agent. This continues until a saturation point is reached wherein the randomness of the CRP coalition formation algorithm eventually provides enough tasks to agents with lower amounts of resources to meet the incoming workload.

Table 5.4 shows the impact of the update interval on the mean LOE per level. The results show that the LOE between levels for each experiment is similar, yielding a balanced distribution of tasks to agents in the network, even without global knowledge. Also, the update intervals chosen appear to have little impact on the overall effectiveness of the algorithm, as the mean LOE for each experiment is also statistically insignificant. **The remaining rows of the table will be filled in when the one million node experiments complete.**

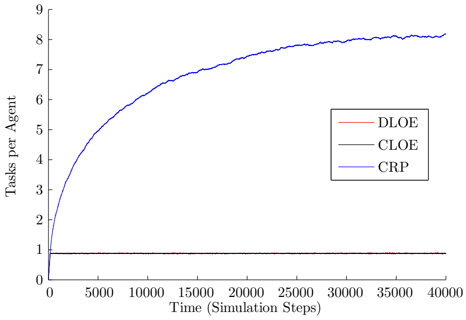

Figure 5.5: Comparison of the three algorithms in a critically-loaded system with synchrony disabled. The number of tasks per agent for the CRP algorithm eventually stabilizes at a significantly higher value than the CLOE or DLOE algorithms.

Table 5.4: Effects of Update Interval on LOE

	Update Interval (time steps)							
Level Number	0	1	2	3	4	5	10	1
Level 0	3.879326	3.878423	3.887911	3.875039	3.880180	3.873442	3.882252	3.875
Level 1	3.846253	3.849077	3.849792	3.847944	3.848183	3.840900	3.846466	3.844
Level 2	3.849562	3.848065	3.852423	3.841447	3.850868	3.839840	3.842462	3.842
Level 3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
Level 4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000

Under significant loading, the task duration for the DLOE and CLOE algorithms resembles Figure 5.6 for all levels of task synchrony. For under-loaded systems, task durations remain relatively uniform. The task duration for under-loaded and critically-loaded systems is roughly flat. This occurs because agents in those systems are able to meet the incoming workload, and so the scheduler is rarely forced to choose which task to execute based solely on priority.

A notable exception to this trend is the CRP algorithm, which poorly places its tasks in all cases, leading to a high standard deviation for the number of tasks per agent. This trend is shown in Figure 5.7. Under the CRP algorithm, many agents are overloaded, while others have zero load. This leads to bottlenecks at those higher loaded agents, creating a net reduction in the work executed per unit time, and therefore forcing critically-loaded systems to become unable to meet the incoming task workload.

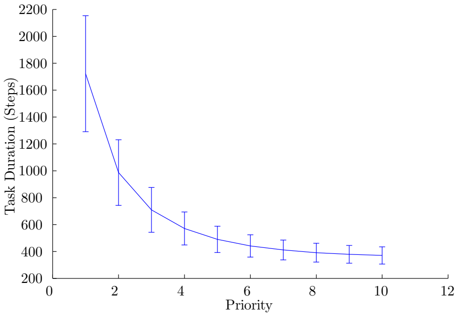

Figure 5.6: Task duration versus priority for critically-loaded and over-loaded systems using the CLOE and DLOE formation algorithms. The duration of tasks reduces with increased priority when the task scheduler is forced to decide based on priority.

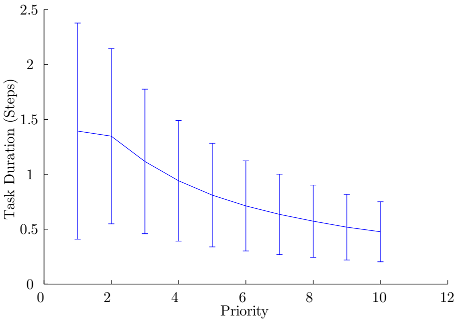

Figure 5.7: Task duration versus priority for a critically loaded system using the CRP coalition formation algorithm. The poor assignment of coalitions to tasks yields bottlenecks in systems with task synchrony, causing delays in processing and eliminating the effectiveness of the task scheduling algorithm.

For systems in which the work execution rate cannot meet the work generation rate, the task scheduler tends to grant more processor time to higher priority tasks. As a result, those higher priority tasks complete with lower mean duration than the lower priority tasks. For under-loaded and critically-loaded CLOE and DLOE systems, sufficient processing power exists to satisfy the incoming work load, and so existing tasks tend to complete before scheduling contention forces a difference in duration between tasks with different priorities.

Figure 5.8 shows the statistical comparison of the work throughput versus task synchrony for critically-loaded experiments where synchrony is disabled, and Figure 5.9 shows the same scenario with task synchrony enabled. The difference in performance for both cases between the CRP algorithm and the other two is sta-

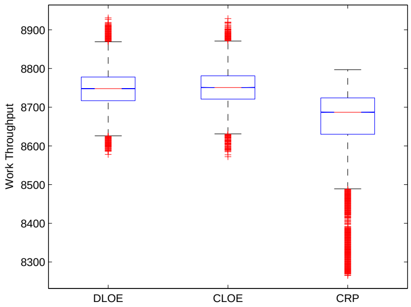

Figure 5.8: ANOVA for work throughput versus task synchrony in critically-loaded experiments where synchrony is disabled. The median performance range is similar, however CRP generates far more outliers.

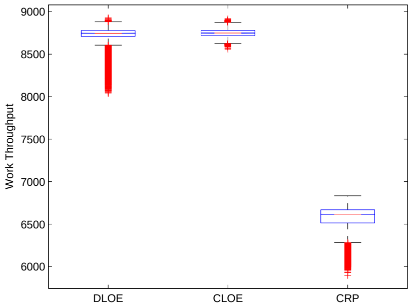

Figure 5.9: ANOVA for work throughput versus task synchrony in critically-loaded experiments where synchrony is enabled. The CLOE and DLOE algorithms produce coalitions that perform similarly, whereas the CRP algorithm produces coalitions with significantly lower throughput.

tistically significant, as the CRP performs far worse than the other two algorithms. Both the CLOE and DLOE algorithms have similar median work throughput, but the CLOE algorithm maintains a slightly lower standard deviation with fewer outliers.

The result of these experiments demonstrates that task synchrony plays an important part in the overall performance of coalitions generated using these different algorithms. The CRP algorithm, in particular, suffers from poor performance as a result of its coalition formation process. In addition, the CRP algorithm suffers from the highest miss rate of the algorithms considered, reaching over 80% in some experiments.

The CLOE demonstrates the best overall sustained task execution rate, as well as the lowest standard deviation. The DLOE algorithm is statistically similar in mean work throughput, with slightly higher variance. Both systems successfully tackle the challenge of task synchrony, and allow the task scheduler to make good decisions that reduce the durations of tasks with higher priorities during high workloads.

5.3 Summary

The result of these experiments demonstrates that task synchrony plays an important part in the overall performance of coalitions generated using these different algorithms. The CRP algorithm, in particular, suffers from poor performance as a result of its coalition formation process. In addition, the CRP algorithm suffers from the highest miss rate of the algorithms considered, reaching over 80% in some experiments.

The CLOE demonstrates the best overall sustained task execution rate, as well as the lowest standard deviation. The DLOE algorithm is statistically similar in mean work throughput, with slightly higher variance. Both systems successfully tackle the challenge of task synchrony, and allow the task scheduler to make good decisions that reduce the durations of tasks with higher priorities during high workloads.

VI. Conclusions and Future Work

The sum of this research effort has yielded a great deal of knowledge and usable technologies. However, this newly acquired knowledge represents perhaps a small fraction of the total value gained. For each challenge conquered, multiple hurdles were overcome, each bearing its own lessons. These lessons include observations of areas for future work. These items for future work identify areas whose scope or direction did not fit into the allotted time and tools for this research effort. Improving each of these areas will help to generate new ideas and knowledge that are useful in the larger community of research.

6.1 RC-Chord

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. The impact of this change is a more configurable runtime system, which is especially important in large-scale systems, which tend to be online for long periods of time.

Super peers exist in two simultaneous clusters, one in which they are super peers, and the other in which they are normal leaf peers. This scenario is present to simplify simulator development, but real systems should permit peers to exist in only a single cluster at a time. This will reduce the total number of clusters needed to satisfy the system's size, at the expense of slightly more complicated cluster formation and destruction algorithms.

Security has yet to achieve a sufficient enough presence in large-scale systems to permit their use on military systems. The CyberCraft agents themselves support Public Key Infrastructure (PKI) inter-agent encrypted communications. However, the system design is still lacking a comprehensive security solution that addresses locality, military classification levels, and failure and recovery scenarios.

The current churn models use a uniform distribution to choose which agents to part from the system. This model can be improved to use separate distributions to distinguish the patterns of normal peers versus super peers. The effect of this change is to improve flexibility to adapt the simulations to a wider variety of research areas.

The message transport delays vary on the range [0,10] milliseconds. This uniform distribution is suitable for local area systems, but does not necessarily represent the different network configurations present in real world systems. The distribution from which the transport delays are drawn should correspond to a specific instance of a target application environment, and studying this behavior before system deployment will provide a better understanding of the network's characteristics in those situations.

6.2 Coalition Formation

Several areas of suggested improvement have become apparent during the design and development process for RC-Chord and DLOE. First, the number and spacing of the resource intervals for the DLOE algorithm are currently static, and only configurable before runtime. These numbers can be tuned over time through observation, however even under those circumstances workload or priority flux cannot necessarily be accommodated, and performance will suffer. A better solution is to provide dynamic reallocation of these variables at runtime, based on observed runtime characteristics. Many learning algorithms could serve this role and significantly improve performance for systems whose runtime performance either varies or follows unexpected patterns.

The number of agents allocated per task can currently only follow a uniform distribution. However, this does not necessarily represent real world systems. Given specific instances of active systems, this parameter could be tuned to optimize performance for the particular task characteristics and task creation rate.

As this document was not meant to dive into the depths of task scheduling theory, a simplistic approach is used that supports the DLOE algorithm's measurement criteria. However, more sophisticated and capable scheduling algorithms should be considered to supplement improvements in the DLOE algorithm. Given the many permutations of scheduling algorithms and measurement criteria, this represents a fertile research area.

Work contributed by different agents should ideally contribute to different pools of work within the task. This is of particular importance when evaluating synchronicity in the task scheduling subsystem. In the event that a single resource requirement within a task is satisfied by multiple agents, the system should also consider how to allocate the work executed by those agents on behalf of the task. This scenario makes sense for resources that can be split evenly, and where work performed using that resource can be likewise divided in some reasonable manner. For example, in the event that the resource is processor time, splitting the resource between multiple slower machines whose sum processing capability satisfies the resource requirement may be reasonable, if the data set supports it. In such an ideal case, the amount of work performed toward this task by agents sharing this resource may indeed be the sum of time spent by each agent during simple processing. For simplicity this consideration was not explored in this model, but future systems may benefit from such a functionality.

Due to scoping, one of the most flexible features of the Hierarchical Peer-to-Peer (HP2P) structure is not under test: the organization of clusters by separate criteria. Constructing clusters by location, as an example, is a valuable property afforded by this topology, and should be explored more in the future, especially as real systems of the scale described here become more common. Moreover, forming coalitions that incorporate these additional metrics will further improve the quality of coalitions. Unfortunately, this promotes the coalition formation algorithm heuristic to a multi-objective optimization process, thus eliminating one of the primary attributes of this approach: speed of coalition formation.

To that end, further analysis is necessary to improve the allocation and destruction of clusters at runtime. Under high churn conditions, a cluster may be formed or destroyed very rapidly in a short period of time. A better approach is to install a threshold algorithm which restricts the formation and destruction of clusters based on time or cluster capacity. This will reduce the amount of cluster thrashing sometimes seen in these systems, reducing maintenance actions, and improving overall performance.

Coalition formation algorithm experiments were conducted in stable networks without churn. As with other stimulus, adding another variable under test would increase the duration of testing beyond the available time by as much as six months to a year. RC-Chord and the coalition formation algorithms do reallocate tasks to agents in the event of agent departures, but adequate testing and analysis of the effects of churn on the coalition formation algorithms remains an open area.

6.3 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 several hundred thousand agents, and is stable under churn.

The large-scale cooperative coalition formation problem is becoming a more important challenge as networked systems seek to leverage the power and scale of Peer-to-Peer (P2P) systems. The difficulty of efficiently sharing the capabilities and assets of attached systems is explored here, and a solution to the cooperative coalition formation problem is proposed and examined. The solution relies upon the facilities provided by the RC-Chord structured overlay, specifically the ability to allocate agents into clusters organized by resource or capability. With this structure, a distributed coalition formation algorithm is evaluated for allocating tasks to teams of agents that provide required capabilities. The DLOE algorithm stores task information for agents with each cluster, and passes this information up the network hierarchy to construct a general view of the task loading on each sub-graph. The DLOE algorithm uses this information to decide how far, and in which direction, to descend in search of agents to satisfy task allocation requests. Simulation results compare the DLOE algorithm against the CRP algorithm, which allocates tasks to agents using a uniform distribution, and the CLOE algorithm which uses global knowledge to allocate tasks such that the mean workload on each agent is minimized. Results indicate that our distributed algorithm performs nearly as well as the centralized optimal algorithm, and significantly better than the random baseline algorithm. With these results, we are confident in stating that the DLOE algorithm sufficiently satisfies its objectives to build efficient task coalitions in a cooperative, large-scale DMAS.

Bibliography

1. URL `http://www.cisco.com`.

2. Abdallah, Maha and Lynda Temal. *GriDB: A Scalable Distributed Database Sharing System for Grid Environments.* Technical report, Paris University, 8, rue du Capitaine Scott 75015 Paris, France, 2003.

3. Abdallah, Sherief and Victor Lesser. "Organization-Based Coalition Formation". *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, 1296–1297. IEEE Computer Society, Washington, DC, USA, 2004. ISBN 1-58113-864-4.

4. Abdallah, Sherief and Victor Lesser. "Organization-Based Cooperative Coalition Formation". *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Techonology, IAT*, 162–168. China, September 2004. URL `http://mas.cs.umass.edu/paper/356`.

5. Aberer, Karl. "P-grid: A self-organizing access structure for p2p information systems". *In CoopIS*, 179–194. 2001.

6. Aberer, Karl, Philippe Cudré-Mauroux, Anwitaman Datta, Zoran Despotovic, Manfred Hauswirth, Magdalena Punceva, and Roman Schmidt. "P-Grid: a self-organizing structured P2P system". *SIGMOD Rec.*, 32(3):29–33, 2003. ISSN 0163-5808.

7. Adar, Eytan and Bernardo A. Huberman. "Free Riding on Gnutella". *First Monday*, 5, 2000.

8. Alima, Luc Onana, Ali Ghodsi, and Seif Haridi. "A Framework for Structured Peer-to-Peer Overlay Networks". *Lecture Notes in Computer Science*, 3267:223–250, 2004.

9. Androutsellis-Theotokis, Stephanos and Diomidis Spinellis. "A survey of peer-to-peer content distribution technologies". *ACM Comput. Surv.*, 36(4):335–371, 2004. ISSN 0360-0300.

10. Ateconi, Stefano, David Hales, and Ozalp Babaoglu. *Broadcasting at the Critical Threshold in Peer-to-Peer Networks.* Technical report, University of Bologna, Department of Computer Science University of Bologna Mura Anteo Zamboni 7 40127 Bologna (Italy), March 2007.

11. Bao, Xiuguo, Binxing Fang, Mingzhen Hu, and Binbin Xu. "Heterogeneous Search in Unstructured Peer-to-Peer Networks". *IEEE Distributed Systems Online*, 6(2):1, 2005. ISSN 1541-4922.

12. Barella, A., C. Carrascosa, V. Botti, and M. Martí. "Multi-agent systems applied to virtual environments: a case study". *VRST '07: Proceedings of the 2007 ACM*

symposium on Virtual reality software and technology, 237–238. ACM, New York, NY, USA, 2007. ISBN 978-1-59593-863-3.

13. Bharambe, Ashwin R., Mukesh Agrawal, and Srinivasan Seshan. "Mercury: supporting scalable multi-attribute range queries". *SIGCOMM Comput. Commun. Rev.*, 34(4):353–366, 2004. ISSN 0146-4833.

14. Birman, Kenneth P. *Reliable Distributed Systems: Technologies, Web Services, and Applications.* Springer, first edition, March 2005.

15. Brunskill, Emma. "Building Peer-to-Peer Systems with Chord, a Distributed Lookup Service". *HOTOS '01: Proceedings of the Eighth Workshop on Hot Topics in Operating Systems*, 81. IEEE Computer Society, Washington, DC, USA, 2001.

16. Bustamante, F. E. and Y. Qiao. "Designing Less-Structured P2P Systems for the Expected High Churn". *Networking, IEEE/ACM Transactions on*, 16(3):617–627, June 2008. ISSN 1063-6692.

17. Cao, Y. Uny, Alex S. Fukunaga, and Andrew B. Kahng. "Cooperative Mobile Robotics: Antecedents and Directions". *Autonomous Robots*, 4(1):7–23, March 1997.

18. Castro, Miguel, Peter Druschel, Y. Charlie, and Hu Antony Rowstron. *Exploiting network proximity in peer-to-peer overlay networks.* Technical report, 2002.

19. Chalupsky, Hans, Yolanda Gil, Craig A. Knoblock, Kristina Lerman, Jean Oh, David V. Pynadath, Thomas A. Russ, and Milind Tambe. "Electric Elves: Applying Agent Technology to Support Human Organizations". *Proceedings of the Thirteenth Conference on Innovative Applications of Artificial Intelligence Conference*, 51–58. AAAI Press, 2001. ISBN 1-57735-134-7.

20. Chow, Randy and Theodore Johnson. *Distributed Operating Systems and Algorithm Analysis.* Addison-Wesley, 1997. ISBN 0201498383.

21. Clarke, Ian, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. "Freenet: A Distributed Anonymous Information Storage and Retrieval System". *Lecture Notes in Computer Science*, 2009:46–63, 2001.

22. Cohen, Bram. "Bittorrent Protocol Specification v1.0". WWW, June.

23. Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms.* The MIT Press, 2nd edition, 2001.

24. Coulouris, George F. and Jean Dollimore. *Distributed systems: concepts and design.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1988. ISBN 0-201-18059-6.

25. Dabek, Frank, Jinyang Li, Emil Sit, James Robertson, M. Frans Kaashoek, and Robert Morris. "Designing a DHT for low latency and high throughput". *NSDI'04: Proceedings of the 1st conference on Symposium on Networked Systems*

Design and Implementation, 7–7. USENIX Association, Berkeley, CA, USA, 2004.

26. Datar, Mayur. "Butterflies and Peer-to-Peer Networks". *ESA '02: Proceedings of the 10th Annual European Symposium on Algorithms*, 310–322. Springer-Verlag, London, UK, 2002. ISBN 3-540-44180-8.

27. Dudek, Gregory, Michael R. M. Jenkin, and David Wilkes. "A taxonomy for multi-agent robotics". *Autonomous Robots*, 3:375–397, 1996.

28. Fiat, Amos and Jared Saia. "Censorship resistant peer-to-peer content addressable networks". *SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, 94–103. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002. ISBN 0-89871-513-X.

29. Frankel, Justin and Tom Pepper. "Gnutella". Internet, 2000. URL `http://gnutella.wego.com`.

30. Freiling, Felix C., Thorsten Holz, and Georg Wicherski. *Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks.* Technical report, The Honeynet Project, 2005.

31. Ganesan, Prasanna, Krishna Gummadi, and Hector Garcia-Molina. "Canon in G Major: Designing DHTs with Hierarchical Structure". *ICDCS '04: Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, 263–272. IEEE Computer Society, Washington, DC, USA, 2004. ISBN 0-7695-2086-3.

32. Ganesh, Ayalvadi J., Anne-Marie Kermarrec, and Laurent Massoulié. "SCAMP: Peer-to-Peer Lightweight Membership Service for Large-Scale Group Communication". *NGC '01: Proceedings of the Third International COST264 Workshop on Networked Group Communication*, 44–55. Springer-Verlag, London, UK, 2001. ISBN 3-540-42824-0.

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

34. Gerkey, Brian P. and Maja J. Mataric. "A formal analysis and taxonomy of task allocation in multi-robot systems". *International Journal of Robotics Research*, 23:939–954, September 2004.

35. Gerkey, Brian Paul. *On multi-robot task allocation.* Ph.D. thesis, University of Southern California, August 2003.

36. Ghodsi, A., L. Alima, S. El-Ansary, P. Brand, and S. Haridi. "Self-Correcting Broadcast in Distributed Hash Tables", 2003.

37. Ghodsi, Ali, Luc Onana Alima, and Seif Haridi. "Low-Bandwidth Topology Maintenance for Robustness in Structured Overlay Networks". *HICSS '05: Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 9*, 302.1. IEEE Computer Society, Washington, DC, USA, 2005. ISBN 0-7695-2268-8-9.

38. Goldman, C. and S. Zilberstein. "Optimizing information exchange in cooperative multi-agent systems", 2003.

39. Goldman, Claudia V. and Shlomo Zilberstein. "Decentralized control of cooperative systems: Categorization and complexity analysis". *Journal of Artificial Intelligence Research*, 22:2004, 2004.

40. Gummadi, K., R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica. "The impact of DHT routing geometry on resilience and proximity". *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, 381–394. ACM, New York, NY, USA, 2003. ISBN 1-58113-735-4.

41. Gupta, Anjali, Barbara Liskov, and Rodrigo Rodrigues. "Efficient routing for peer-to-peer overlays". *NSDI'04: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation*, 9–9. USENIX Association, Berkeley, CA, USA, 2004.

42. Gupta, Indranil, Ken Birman, Prakash Linga, Al Demers, and Robbert van Renesse. "Kelips: Building an Efficient and Stable P2P DHT Through Increased Memory and Background Overhead". *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*. 2003.

43. Harvey, Nicholas J. A., Michael B. Jones, Stefan Saroiu, Marvin Theimer, and Alec Wolman. "SkipNet: a scalable overlay network with practical locality properties". *USITS'03: Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems*, 9–9. USENIX Association, Berkeley, CA, USA, 2003.

44. Hofstätter, Quirin, Stefan Zöls, Maximilian Michel, Zoran Despotovic, and Wolfgang Kellerer. "Chordella - A Hierarchical Peer-to-Peer Overlay Implementation for Heterogeneous, Mobile Environments". *P2P '08: Proceedings of the 2008 Eighth International Conference on Peer-to-Peer Computing*, 75–76. IEEE Computer Society, Washington, DC, USA, 2008. ISBN 978-0-7695-3318-6.

45. Horowitz, Ellis, Sartaj Sahni, and Sanguthevar Rajasckaran. *Computer Algorithms: C++*. W. H. Freeman & Co., New York, NY, USA, 1996. ISBN 0716783150.

46. Jelasity, Márk, Alberto Montresor, Gian Paolo Jesi, and Spyros Voulgaris. "The Peersim Simulator". `http://peersim.sf.net`.

47. Kaashoek, M. Frans and David R. Karger. "Koorde: A simple degree-optimal distributed hash table". *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*. 2003.

48. Kalogeraki, Vana, Dimitrios Gunopulos, and D. Zeinalipour-Yazti. "A local search mechanism for peer-to-peer networks". *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, 300–307. ACM, New York, NY, USA, 2002. ISBN 1-58113-492-4.

49. Karger, David, Eric Lehman, Tom Leighton, Rina Panigrahy, Matthew Levine, and Daniel Lewin. "Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web". *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, 654–663. ACM, New York, NY, USA, 1997. ISBN 0-89791-888-6.

50. Karrels, Daniel. *Specialty Exam: Response to Dr. Mullins*. Technical report, Air Force Institute of Technology, 2950 Hobson Way Wright Patterson AFB, OH, May 2008.

51. Karrels, Daniel, Gilbert Peterson, and Barry Mullins. "Structured P2P technologies for distributed command and control". *Peer-to-Peer Networking and Applications*, Online, March 2009.

52. Kaufman, Charlie, Radia Perlman, and Mike Speciner. *Network security: PRIVATE communication in a PUBLIC world*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2002. ISBN 0-13-046019-2.

53. Ketchpel, Steven P. "Forming Coalitions in the Face of Uncertain Rewards". *National Conference on Artificial Intelligence*, 414–419. 1994.

54. Kleinberg, Jon. "The Small-World Phenomenon: An Algorithmic Perspective". *in Proceedings of the 32nd ACM Symposium on Theory of Computing*, 163–170. 2000.

55. Krishnamurthy, Balachander, Jia Wang, and Yinglian Xie. "Early measurements of a cluster-based architecture for P2P systems". *IMW '01: Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, 105–109. ACM, New York, NY, USA, 2001. ISBN 1-58113-435-5.

56. Kumar, Vipin, Ananth Grama, Anshul Gupta, and George Karypis. *Introduction to parallel computing: design and analysis of algorithms*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1994. ISBN 0-8053-3170-0.

57. Leibowitz, Nathaniel, Matei Ripeanu, and Adam Wierzbicki. "Deconstructing the Kazaa Network". *WIAPP '03: Proceedings of the The Third IEEE Workshop on Internet Applications*, 112. IEEE Computer Society, Washington, DC, USA, 2003. ISBN 0-7695-1972-5.

58. Li, Harry C., Allen Clement, Edmund L. Wong, Jeff Napper, Indrajit Roy, Lorenzo Alvisi, and Michael Dahlin. "BAR Gossip". *USENIX'06: Proceed-*

ings of the 7th conference on USENIX Symposium on Operating Systems Design and Implementation, 14–14. USENIX Association, Berkeley, CA, USA, 2006.

59. Li, Jinyang, Jeremy Stribling, Robert Morris, and M. Frans Kaashoek. "Bandwidth-efficient management of DHT routing tables". *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, 99–114. USENIX Association, Berkeley, CA, USA, 2005.

60. Li, Jinyang, Jeremy Stribling, Robert Morris, M. Frans Kaashoek, and Thomer M. Gil. "A performance vs. cost framework for evaluating DHT design tradeoffs under churn." *INFOCOM*, 225–236. IEEE, 2005.

61. Loguinov, Dmitri, Anuj Kumar, Vivek Rai, and Sai Ganesh. "Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilience". *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, 395–406. ACM, New York, NY, USA, 2003. ISBN 1-58113-735-4.

62. Lu, Eric Jui-Lin, Yung-Fa Huang, and Shu-Chiu Lu. "ML-Chord: A multi-layered P2P resource sharing model". *Journal of Network and Computer Applications*, In Press, Corrected Proof:–, 2008. ISSN 1084-8045. URL `http://www.sciencedirect.com/science/article/B6WKB-4T6CTTG-1/2/19e838626733e01`

63. Lu, J. and J. Callan. "Federated search of text-based digital libraries in hierarchical peer-to-peer networks". *roceedings of the 27th European Conference on Information Retrieval.* 2004.

64. Lu, J. and J. Callan. "Content-based peer-to-peer network overlay for full-text federated search". *Proceedings of the Eighth Recherche d'Information Assistee par Ordinateur (RIAO) Conference.* 2006.

65. Lua, Keong, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. "A survey and comparison of peer-to-peer overlay network schemes". *Communications Surveys & Tutorials, IEEE*, 72–93, 2005.

66. Lv, Qin, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. "Search and replication in unstructured peer-to-peer networks". *ICS '02: Proceedings of the 16th international conference on Supercomputing*, 84–95. ACM, New York, NY, USA, 2002. ISBN 1-58113-483-5.

67. Malkhi, Dahlia, Moni Naor, and David Ratajczak. "Viceroy: a scalable and dynamic emulation of the butterfly". *PODC '02: Proceedings of the twenty-first annual symposium on Principles of distributed computing*, 183–192. ACM, New York, NY, USA, 2002. ISBN 1-58113-485-1.

68. Manku, Gurmeet Singh, Mayank Bawa, and Prabhakar Raghavan. "Symphony: distributed hashing in a small world". *USITS'03: Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems*, 10–10. USENIX Association, Berkeley, CA, USA, 2003.

69. Maymounkov, Petar and David Mazières. "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric". *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, 53–65. Springer-Verlag, London, UK, 2002. ISBN 3-540-44179-4.

70. Milgram, S. "The small world problem". *Psychology Today*, (2):60–67, 1967.

71. Montgomery, Douglas C. *Design and Analysis of Experiments*. Wiley, December 2004. ISBN 047148735X. URL `http://www.worldcat.org/isbn/047148735X`.

72. Pynadath, D. and M. Tambe. "Multiagent teamwork: Analyzing the optimality and complexity of key theories and models", 2002.

73. Pynadath, David V. and Milind Tambe. "The communicative multiagent team decision problem: Analyzing teamwork theories and models". *Journal of Artificial Intelligence Research*, 16:2002, 2002.

74. Rahwan, Talal. *Algorithms for Coalition Formation in Multi-Agent Systems*. Ph.D. thesis, University of Southampton, August 2007.

75. Ratnasamy, Sylvia, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. "A scalable content-addressable network". *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, 161–172. ACM, New York, NY, USA, 2001. ISBN 1-58113-411-8.

76. Rhea, Sean, Dennis Geels, Timothy Roscoe, and John Kubiatowicz. "Handling churn in a DHT". *ATEC '04: Proceedings of the annual conference on USENIX Annual Technical Conference*, 10–10. USENIX Association, Berkeley, CA, USA, 2004.

77. Ripeanu, Matei, Adriana Iamnitchi, and Ian Foster. "Mapping the Gnutella Network". *IEEE Internet Computing*, 6(1):50–57, 2002.

78. Rowstron, Antony I. T. and Peter Druschel. "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems". *Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, 329–350. Springer-Verlag, London, UK, 2001. ISBN 3-540-42800-3.

79. Sandholm, T. W. "An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations". *Proceedings of the 12th International Workshop on Distributed Artificial Intelligence*, 295–308. Hidden Valley, Pennsylvania, 1993.

80. Saroiu, Stefan, Krishna P. Gummadi, and Steven D. Gribble. "Measuring and analyzing the characteristics of Napster and Gnutella hosts". *Multimedia Syst.*, 9(2):170–184, 2003. ISSN 0942-4962.

81. Shehory, O. and S. Kraus. "Task Allocation via Coalition Formation among Autonomous Agents". *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, 655–661. Montréal, Québec, Canada, 1995.

82. Shehory, Onn and Sarit Kraus. "Coalition formation among autonomous agents: Strategies and complexity (preliminary report)". C. Castelfranchi and J.-P. Müller (editors), *From Reaction to Cognition — Fifth European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW-93 (LNAI Volume 957)*, 56–72. Springer-Verlag: Heidelberg, Germany, 1995.

83. Shehory, Onn and Sarit Kraus. "Methods for task allocation via agent coalition formation". *Artif. Intell.*, 101(1-2):165–200, 1998. ISSN 0004-3702.

84. Smith, R. G. "The contract net protocol: high-level communication and control in a distributed problem solver". 357–366, 1988.

85. Stanney, Kay M., Ronald R. Mourant, and Robert S. Kennedy. "Human Factors Issues in Virtual Environments: A Review of the Literature". *Presence: Teleoper. Virtual Environ.*, 7(4):327–351, 1998. ISSN 1054-7460.

86. Stoica, Ion, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. "Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications". *Proceedings of the 2001 ACM SIGCOMM Conference*, 149–160. 2001.

87. Tao, Nigel, Jonathan Baxter, and Lex Weaver. "A Multi-Agent Policy-Gradient Approach to Network Routing". *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, 553–560. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001. ISBN 1-55860-778-1.

88. Theocharopoulou, Christina, Ioannis Partsakoulakis, George A. Vouros, and Kostas Stergiou. "Overlay networks for task allocation and coordination in dynamic large-scale networks of cooperative agents". *AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, 1–8. ACM, New York, NY, USA, 2007. ISBN 978-81-904262-7-5.

89. Torell, Wendy and Victor Avelar. *Mean Time Between Failure: Explanation and Standards.* White paper, American Power Conversion Corporation, 2004.

90. Vishnevsky, Vladimir, Alexander Safonov, Mikhail Yakimov, Eunsoo Shim, and Alexander D. Gelman. "Scalable Blind Search and Broadcasting in Peer-to-Peer Networks". *P2P '06: Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing*, 259–266. IEEE Computer Society, Washington, DC, USA, 2006. ISBN 0-7695-2679-9.

91. Vuong, Son and Juan Li. "Efa: An Efficient Content Routing Algorithm in Large Peer-to-Peer Overlay Networks". *P2P '03: Proceedings of the 3rd International Conference on Peer-to-Peer Computing*, 216. IEEE Computer Society, Washington, DC, USA, 2003. ISBN 0-7695-2023-5.

92. Wijngaards, Niek J. E., B. J. Overeinder, Maarten van Steen, and Frances M. T. Brazier. "Supporting Internet-scale multi-agent systems". *Data Knowledge Engineering*, 41(2-3):229–245, 2002.

93. Wikipedia. “De Bruijn Graph”. Internet. URL http://en.wikipedia.org/wiki/DeBruijngraph.

94. WikiPedia. “Kademlia”. WWW, June 2008. URL http://en.wikipedia.org/wiki/Kademlia.

95. Winter, E. *The Handbook of Game Theory*, chapter The Shapley Value, 2026–2052. North-Holldand, 2002.

96. Wolpert, David H. and William G. Macready. *No Free Lunch Theorems for Search*. Working Papers 95-02-010, Santa Fe Institute, February 1995.

97. Xu, Yang, Paul Scerri, Bin Yu, Steven Okamoto, Michael Lewis, and Katia Sycara. “An integrated token-based algorithm for scalable coordination”. *AAMAS ’05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, 407–414. ACM, New York, NY, USA, 2005. ISBN 1-59593-093-0.

98. Xuan, Ping, Victor Lesser, and Shlomo Zilberstein. “Communication decisions in multi-agent cooperation: model and experiments”. *AGENTS ’01: Proceedings of the fifth international conference on Autonomous agents*, 616–623. ACM, New York, NY, USA, 2001. ISBN 1-58113-326-X.

99. Xue, Guangtao, Yi Jiang, Yinyuan You, and Minglu Li. “A topology-aware hierarchical structured overlay network based on locality sensitive hashing scheme”. *UPGRADE ’07: Proceedings of the second workshop on Use of P2P, GRID and agents for the development of content networks*, 3–8. ACM, New York, NY, USA, 2007. ISBN 978-1-59593-718-6.

100. Yang, Beverly and Hector Garcia-Molina. “Designing a Super-Peer Network”. *icde*, 00:49, 2003. ISSN 1063-6382.

101. Yokoo, Makoto, Edmund H. Durfee, Toru Ishida, and Kazuhiro Kuwabara. “The Distributed Constraint Satisfaction Problem: Formalization and Algorithms”. *Knowledge and Data Engineering*, 10(5):673–685, 1998.

102. Zeinalipour-Yatzi, D. and T. Folias. “A quantitative analysis of the gnutella network traffic”, 2002.

103. Zhang, Haizheng, W. Bruce Croft, Brian Levine, and Victor Lesser. “A Multi-Agent Approach for Peer-to-Peer Based Information Retrieval System”. *aamas*, 01:456–463, 2004.

104. Zhang, Haizheng and Victor Lesser. “Multi-agent based peer-to-peer information retrieval systems with concurrent search sessions”. *AAMAS ’06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 305–312. ACM Press, New York, NY, USA, 2006. ISBN 1-59593-303-4.

105. Zhang, Haizhong and Victor Lesser. “A Dynamically Formed Hierarchical Agent Organization for a Distributed Content Sharing System”. *IAT ’04: Proceedings*

of the Intelligent Agent Technology, IEEE/WIC/ACM International Conference on (IAT'04), 169–175. IEEE Computer Society, Washington, DC, USA, 2004. ISBN 0-7695-2101-0.

106. Zhao, B., L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiatowicz. "Tapestry: A resilient global-scale overlay for service deployment", 2003.

107. Zhao, Ben Y., Yitao Duan, Ling Huang, Anthony D. Joseph, and John Kubiatowicz. "Brocade: Landmark Routing on Overlay Networks". *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, 34–44. Springer-Verlag, London, UK, 2002. ISBN 3-540-44179-4.

108. Zoels, Stefan, Zoran Despotovic, and Wolfgang Kellerer. "On hierarchical DHT systems - An analytical approach for optimal designs". *Comput. Commun.*, 31(3):576–590, 2008. ISSN 0140-3664.

109. Zoels, Stefan, Simon Schubert, Wolfgang Kellerer, and Zoran Despotovic. "Hybrid DHT Design for Mobile Environments". 19–30, 2008.

110. Zols, Stefan, Rüdiger Schollmeier, Wolfgang Kellerer, and Anthony Tarlano. "The Hybrid Chord Protocol: A Peer-to-Peer Lookup Service for Context-Aware Mobile Applications". Pascal Lorenz and Petre Dini (editors), *Networking – ICN 2005. 4th International Conference on Networking, Proceedings, Part II*, volume 3421 of *Lecture Notes in Computer Science*, 781–792. Springer-Verlag, Berlin Heidelberg, April 2005.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 074-0188*

1. REPORT DATE *(DD-MM-YYYY)* 9-Sep-2009	**2. REPORT TYPE** **Doctoral Dissertation**	**3. DATES COVERED** *(From – To)* Aug 2006 – Sep 2009

4. TITLE AND SUBTITLE Large-Scale Distributed Coalition Formation	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Daniel R. Karrels, Capt, USAF	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765 DSN: 785-3636	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/DCE/ENG/09-11
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFRL Kirtland AFB, NM	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The CyberCraft project is an effort to construct a large scale Distributed Multi-Agent System (DMAS) to provide autonomous Cyberspace defense and mission assurance for the DoD. It employs a small but flexible agent structure that is dynamically reconfigurable to accommodate new tasks and policies. This document describes research into developing protocols and algorithms to ensure continued mission execution in a system of one million or more agents, focusing on protocols for coalition formation and Command and Control. It begins by building large-scale routing algorithms for a Hierarchical Peer to Peer structured overlay network, called Resource-Clustered Chord (RC-Chord). RC-Chord introduces the ability to efficiently locate agents by resources that agents possess. Combined with a task model defined for CyberCraft, this technology feeds into an algorithm that constructs task coalitions in a large-scale DMAS. Experiments reveal the flexibility and effectiveness of these concepts for achieving maximum work throughput in a simulated CyberCraft environment.

15. SUBJECT TERMS

Distributed Multi-Agent System, Hierarchical Peer to Peer, Large-Scale, Command and Control

16. SECURITY CLASSIFICATION OF:	**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF**	**19a. NAME OF RESPONSIBLE PERSON** **Dr. Gilbert Peterson, ENG**

REPORT **U**	ABSTRACT **U**	c. THIS PAGE **U**	**UU**	PAGES 150	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-6565; email: gilbert.peterson@afit.edu