Structured P2P Technologies for Distributed Command and Control

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

***Index Terms*—Multi-Agent System, Distributed System, Peer to Peer, Large-Scale, Command and Control**

***Abstract*—The utility of Peer-to-Peer (P2P) systems extends far beyond traditional file sharing. This paper provides an overview of how P2P systems are capable of providing robust command and control for Distributed Multi-Agent Systems (DMASs). Specifically, this article presents the evolution of P2P architectures to date by discussing supporting technologies and applicability of each generation of P2P systems. It provides a detailed survey of fundamental design approaches found in modern large-scale P2P systems highlighting design considerations for building and deploying scalable P2P applications. The survey includes unstructured P2P systems, content retrieval systems, communications structured P2P systems, flat structured P2P systems and finally Hierarchical Peer-to-Peer (HP2P) overlays. It concludes with a presentation of design tradeoffs and opportunities for future research into P2P overlay systems.**

I. Introduction

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 [1] and Freenet [2] 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, we envision 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 paper examines P2P technologies as the organizational component in large-scale DMASs applications. A primary trait of these Multi-Agent Systems (MASs) is the need for Command and Control (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 II 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

The views expressed in this article are those of the authors and do not reflect any official policy or position of the United States Air Force, Department of Defense, or the U.S. Government. This research is funded by the Air Force Research Laboratory (AFRL).

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 VII, including unstructured and content retrieval P2P systems. The focus area of this document is the discussion of communications structured P2P systems beginning in Section IX, continuing with flat and hierarchical structured P2P systems. This document concludes by listing design trade-offs and future work, followed by concluding remarks in Section XIV.

II. 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 [3]. 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.

This discussion is not intended to fully explore the problem and idea spaces of achieving large-scale C2. Rather, it is a first step toward building a solid foundation on which to construct a working distributed C2 system. P2P systems may provide the necessary scale and affordability for extending DMAS research, even though they lack in certain key areas (see Section XIII). We will discuss our motivations and analyze the suitability of various P2P overlay approaches as the discussion continues, and conclude with a more concise summary of our findings in Section XIV.

III. 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.

A. Electric Elves

The Electric Elves project [4] 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.

B. Virtual Environments

Virtual Environments (VEs) [5] 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 [6]. 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.

C. 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 [7]. 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[1]. 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.

IV. 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 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.

[1]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.

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.

A. 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 1

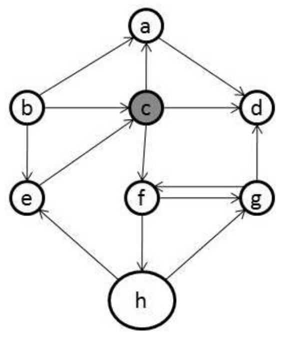


Fig. 1. An example P2P network with eight nodes. The arrows represent a set of transactions at a given time, but the links are considered to be bidirectional.

A HP2P architecture, as shown in Figure 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 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 [8], [9].

B. Small World

Stanley Milgram's small-world phenomenon [11] 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

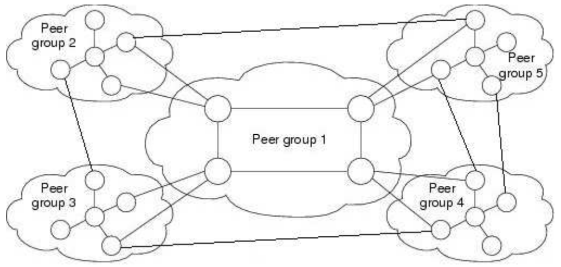


Fig. 2. An example two-layer HP2P network [10]. Each of the five virtual nodes in the network is a separate P2P network, connected by two super peers to other virtual nodes. As in a simple P2P system, each peer (cluster) is able to connect to other peers (clusters).

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

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

This taxonomy is adapted from the work of Cao [13] and Dudek [14] who analyzed the characteristics of multi-robot systems. In addition, this paper builds on the work of Lua, et. al. [15], 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 [16] and Tapestry [17] 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.

A. 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 [18]. 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 [19], Freenet [2], and BitTorrent[2] [20], 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:

- **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 [21]. 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 I introduces the overlay networks discussed in this paper, categorized by their topological strictness.

TABLE I
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

[2]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.

B. 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 [18], [22].

- **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 [2] 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.

C. 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 [38].

D. 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 II 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.

E. 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 [39], 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).

TABLE II
LARGE-SCALE P2P MULTI-AGENT SYSTEM STRUCTURED OVERLAY CHARACTERISTICS

Name	Routing Geometry	Query Path Length	Node Memory	Addressing	Scalability	Bandwidth
Accordion [23]	Ring	$O(1)$ - $O(\log N)$	$O(1)$ - $O(\log N)$	Consistent Hashing	High	Low
Bamboo [24]	Ring	$O(\log N)$	$O(\log N)$	Uniformly Distributed	High	Low
Butterfly [25]	Butterfly	$O(\log N)$	$O(\log^2 N)$	Uniformly Distributed	High	Low
CAN [26]	Hypercube	$O(d\sqrt[d]{N})$	$O(2d)$	Cartesian Zones	High	Moderate
Chord [27]	Ring	$O(\log N)$	$O(\log N)$	Consistent Hashing	High	Low
DKS [28]	Tree	$O(\log_k N)$	$O(\log N)$	Unique	Low	Low
Kademlia [29]	Tree	$O(\log N)$	$O((2^b - 1)\log_{2^b} N)$	Uniformly Distributed	High	Low
Kelips [30]	Linear	$O(1)$	$O(\sqrt{N})$	Consistent Hashing	Moderate	Moderate
Koorde [31]	de Bruijn	$O(\log N/\log\log N)$ - $O(\log N)$	$O(1)$ - $O(\log N)$	Consistent Hashing	Low	High
Mercury [32]	Ring	$O(\log^2 N/k)$	$O(\log N)$	Uniform Random Sampling	Moderate	Moderate
One-Hop [33]	Linear	$O(1)$	$O(N)$	Unique	Low	High
P-Grid [34]	Tree	$O(\log N)$	$O(\log D)$	Uniformly Distributed	Moderate	Moderate
Pastry [16]	Ring	$O(\log N)$	$O(\log N)$	Uniformly Distributed	High	Low
SkipNet [35]	Skip List	$O(\log N)$	$O(\log N)$	Skip Lists	Moderate	High
Symphony [36]	Ring	$O(\frac{1}{k}\ log^2 N)$	$O(2k + 2)$	Uniformly Distributed	High	Low
Tapestry [17]	Ring	$O(\log N)$	$O(\log N)$	Uniformly Distributed	High	Low
Two-Hop [33]	Linear	$O(1)$	$O(N/k)$	Unique	Moderate	Moderate
Viceroy [37]	Ring & Butterfly	$O(\log N)$	$O(\log N)$	Consistent Hashing	High	Low

- **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 [40].
- **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.

F. 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 [41] 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.

G. 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.

H. Reconfigurability

Deployed P2P networks tend to have a high churn rate [42], 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 II, and instead is discussed inline, where appropriate.

VI. 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.

VII. 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 [1] use a full (one-to-all) broadcast at each step of a message query, whereas nodes in a Freenet [2] 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 [43]. Still, some systems such as Gnutella [44], [45] have gained popularity even while using this approach.

An improvement to the basic flooding protocol is the Modified Breadth First Search (MBFS) [46], a gossip algorithm [47]. 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 [48]. 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 [49]. 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 [50] 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 [51] 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 [52].

SCAMP [53] 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 [54] 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 [2] 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 [20], [32] 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.

VIII. 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 [55]. 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 [56]. Lu and Callan [57], [58] 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. [59] 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 [60], [61].

P-Grid [34], [62] (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 3. Nodes at each level in the virtual tree store the location of a node corresponding to data keys that are in 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. [63] 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

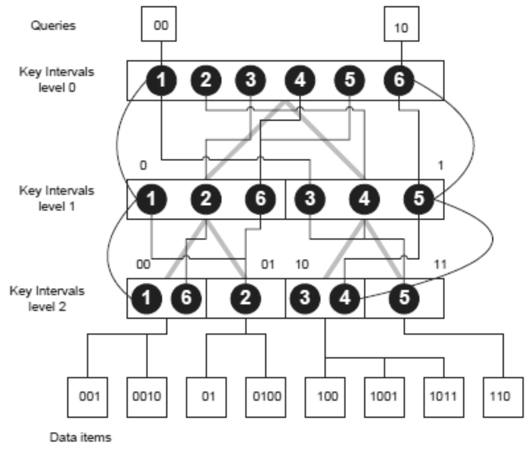


Fig. 3. An example P-Grid tree structure [34]. This tree holds eight data items, shown in the bottom level. Searches begin from the top level, and progress downward, with one bit resolved at each level. Backward and cross-level links are maintained to reduce search time for locating distant (key-wise) data items.

is semantically random[3]. 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.

IX. 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 II. 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.

[3]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.

X. 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.

A. Chord

The Chord protocol [27], [64] uses consistent hashing [39] to locate nodes in a loosely structured P2P network. Consistent hashing in Chord uses a standard hashing technique (such as SHA-1 [65]) 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 [8].) 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 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.

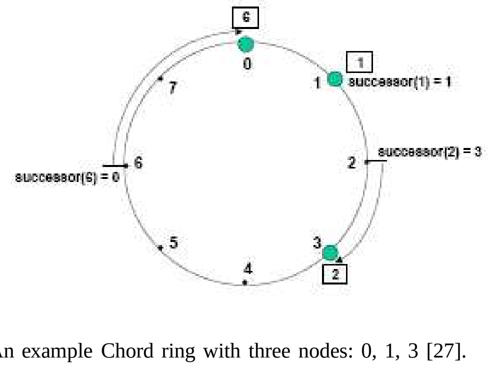


Fig. 4. An example Chord ring with three nodes: 0, 1, 3 [27].

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 [27].

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 close to the desired key. The number of nodes in a steady state system to be examined is, with high probability, constrained by $O(\log N)$ [27].

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) [43]. 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 [23]. 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 [66] to reduce average lookup times, while still staying under the bandwidth limitation.

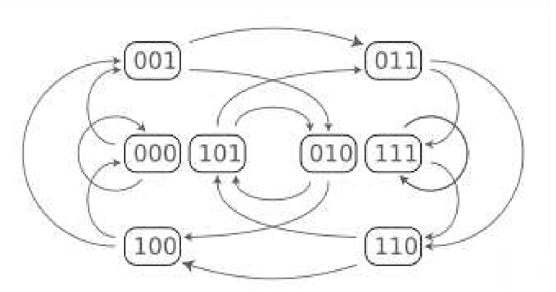


Fig. 5. An example de Bruijn graph for n = 2 bits, m = 3, with 2^3 possible symbols [67]. Each node has exactly n incoming and n outgoing edges.

A variation of Chord, called Koorde [31], 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 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 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 [33] 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.

B. Pastry

Pastry [16] 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 [68]. 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 [24] 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 [69]. 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 [24], 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 [29], [70] 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 [17] 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 [40]. 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.

C. *Other Design Paradigms*

Viceroy [37] 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 [32] 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) [26] 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 [40], SkipNet [35] 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) [28], [71], [72] 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 [30] 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.

XI. 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. [73] 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 [74] and Brocade [75]). 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 [76] 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 [77], [78], 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 [25] have built a HP2P structured overlay based on a butterfly network [33], shown in Figure 6. They apply the butterfly topology to build a 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 [79] can reduce this to $O(\log N)$.

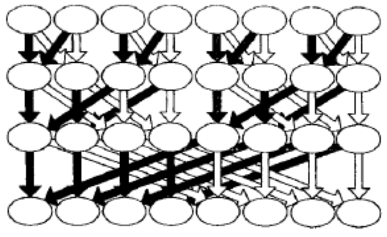


Fig. 6. An example butterfly network [25]. There exist $O(\log N)$ levels, where each level is considered a super peer level. Leaf nodes connect to a random sampling of nodes at each super peer level to provide redundancy and fault tolerance.

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.

XII. Design Tradeoffs

When examining Table II, 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 [36]. 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 [80], [81], 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 [73]. 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.

XIII. Future Work

While many DMAS applications are currently developed using flat P2P systems, we believe that certain applications may require a more segmented approach. For example, military systems require additional security, which is an area that many of the current overlay structures have yet to address. 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 through clustered isolation. The clustering of peer groups can be organized in many ways, to include security access levels or restrictions, which permits more flexibility in system design.

C2 systems require a stable and reliable communications infrastructure on top of which to build sophisticated and resource intensive mission-critical systems. Purely as a communications mechanism, structured P2P systems provide this necessary functionality. However, elements such as security have been neglected. We believe that a comprehensive and effective security solution must be designed into the core of a P2P communications technology before it will be widely accepted into enterprise use.

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 [82]. 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. That is, a trust chain is only as strong as its weakest link. This problem exists in epidemic proportions in a large-scale system which relies on communications with long hop lengths and few authoritative sources.

Many P2P systems propagate key identifiers whenever possible to improve search efficiency. While this is a desirable trait from a purely performance standpoint, it creates difficulties in systems which rely on compartmentalized security. The replication of keys in such systems must be restricted to certain areas of the network, which will require further design effort for the current class of communications structured P2P systems. Likewise, malicious nodes may advertise faulty keys, which can then be difficult to revoke given the widespread replication of those keys across the network.

The challenges of task allocation and coordination with partial visibility is still an active area of research for large-scale DMASs and P2P systems [83], [84], and will be our immediate focus. 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. The formal model of this problem is $\mathcal{NP}$-hard [3], [8], and current solutions do not scale to the size of existing P2P structured overlay networks [83], [85]–[87]. 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.

Tangent to the task allocation problem are the resource distribution and task scheduling problems. Much thought has been put into these challenges, but few currently examine the scale reached by large P2P systems. These problems, like the task allocation problem, are made more difficult through partial visibility in the system, including participants as well as tasks and resources. These problems are often framed as distributed constraint optimization problems whose solutions can, in many cases, be directly mutated to use P2P overlays. However, specific attention must be given to account for the scaling of modern peer based networks.

HP2P overlays may be able to facilitate a leap forward in realizing large-scale task allocation algorithms. With the clustered separation of groups by location, interest, specialty, or other criteria, HP2P systems provide an explicit and ordered framework of participants in large-scale teams. While this will not break the theoretical bounds of the coalition formation and task allocation problems, it may provide a framework for generating useful large-scale solutions that are tractable.

Viceroy has capitalized on the intersection of multiple routing geometries, Chordella has combined multiple instances of Chord to form a HP2P system, and others are following suit. What other properties can be gained, and feature sets improved, by combining technologies? 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. We believe that this trend will continue with HP2P systems, allowing developers to design and field flexible large-scale C2 applications.

XIV. Concluding Remarks

The early generations of P2P networks satisfy the requirement to share data without formal administrators or management of a complex set of hardware and software systems. Rather, they provide a usable and novel technique for sharing data with affordable and available systems. However, with increased popularity came problems with scalability. Early broadcast-based approaches persist today in the form of successful systems like Freenet, while less robust approaches such as KaZaa and Gnutella have waned in popularity. Despite this trend, they may still be useful to individuals or groups seeking small-scale and easy to use systems with little maintenance overhead.

Subsequent generations of P2P systems have grown in scalability as well as functionality. With improved feature sets has come increased maintenance, both by the user and the agents of the system. However, reduced bandwidth consumption and increased convenience and reliability have stimulated a rapid growth in the use of these systems. This trend continues today, and we envision it to continue, and even strengthen, in coming years.

The current generation of P2P systems is built upon an infrastructure of reliable and scalable communications systems. This frees the designer to continue the creative cycle and natural evolution of P2P based systems into architectures for heterogeneous application development and deployment in the hobbyist and low-security environments. However, we believe that comprehensive security solutions must be developed before the enterprise world will begin to seriously consider and adopt P2P systems for their business platforms.

The framework in which to both command and control systems of agents in large-scale P2P systems is present, although building scalable algorithms for distributed coordination remains an active area of study. Our research continues into developing the principles and techniques that will govern the prosecution of large-scale C2 in P2P systems.

References

[1] E. Adar and B. A. Huberman, "Free riding on gnutella," *First Monday*, vol. 5, 2000.

[2] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A distributed anonymous information storage and retrieval system," *Lecture Notes in Computer Science*, vol. 2009, pp. 46–63, 2001.

[3] B. P. Gerkey and M. J. Mataric, "A formal analysis and taxonomy of task allocation in multi-robot systems," *International Journal of Robotics Research*, vol. 23, pp. 939–954, September 2004.

[4] H. Chalupsky, Y. Gil, C. A. Knoblock, K. Lerman, J. Oh, D. V. Pynadath, T. A. Russ, and M. Tambe, "Electric elves: Applying agent technology to support human organizations," in *Proceedings of the Thirteenth Conference on Innovative Applications of Artificial Intelligence Conference*. AAAI Press, 2001, pp. 51–58.

[5] K. M. Stanney, R. R. Mourant, and R. S. Kennedy, "Human factors issues in virtual environments: A review of the literature," *Presence: Teleoper. Virtual Environ.*, vol. 7, no. 4, pp. 327–351, 1998.

[6] A. Barella, C. Carrascosa, V. Botti, and M. Martí, "Multi-agent systems applied to virtual environments: a case study," in *VRST '07: Proceedings of the 2007 ACM symposium on Virtual reality software and technology*. New York, NY, USA: ACM, 2007, pp. 237–238.

[7] N. Tao, J. Baxter, and L. Weaver, "A multi-agent policy-gradient approach to network routing," in *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 553–560.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. The MIT Press, 2001.

[9] B. Yang and H. Garcia-Molina, "Designing a super-peer network," *icde*, vol. 00, p. 49, 2003.

[10] [Online]. Available: http://www.cisco.com

[11] S. Milgram, "The small world problem," *Psychology Today*, no. 2, pp. 60–67, 1967.

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

[13] Y. U. Cao, A. S. Fukunaga, and A. B. Kahng, "Cooperative mobile robotics: Antecedents and directions," *Autonomous Robots*, vol. 4, no. 1, pp. 7–23, March 1997.

[14] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "A taxonomy for multi-agent robotics," 1996.

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

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

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

[18] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica, "The impact of dht routing geometry on resilience and proximity," in *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2003, pp. 381–394.

[19] "Gnutella," Internet. [Online]. Available: http://gnutella.wego.com

[20] B. Cohen, "Bittorrent protocol specification v1.0," WWW, June.

[21] F. E. Bustamante and Y. Qiao, "Designing less-structured p2p systems for the expected high churn," *Networking, IEEE/ACM Transactions on*, vol. 16, no. 3, pp. 617–627, June 2008.

[22] D. Loguinov, A. Kumar, V. Rai, and S. Ganesh, "Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilience," in *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2003, pp. 395–406.

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

[24] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling churn in a dht," in *ATEC '04: Proceedings of the annual conference on USENIX Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 2004, pp. 10–10.

[25] A. Fiat and J. Saia, "Censorship resistant peer-to-peer content addressable networks," in *SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2002, pp. 94–103.

[26] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A scalable content-addressable network," in *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2001, pp. 161–172.

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

[28] L. O. Alima, A. Ghodsi, and S. Haridi, "A framework for structured peer-to-peer overlay networks," *Lecture Notes in Computer Science*, vol. 3267, pp. 223–250, 2004.

[29] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the xor metric," in *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*. London, UK: Springer-Verlag, 2002, pp. 53–65.

[30] I. Gupta, K. Birman, P. Linga, A. Demers, and R. van Renesse, "Kelips: Building an efficient and stable P2P DHT through increased memory and background overhead," in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, 2003.

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

[32] A. R. Bharambe, M. Agrawal, and S. Seshan, "Mercury: supporting scalable multi-attribute range queries," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 353–366, 2004.

[33] A. Gupta, B. Liskov, and R. Rodrigues, "Efficient routing for peer-to-peer overlays," in *NSDI'04: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2004, pp. 9–9.

[34] K. Aberer, "P-grid: A self-organizing access structure for p2p information systems," in *In CoopIS*, 2001, pp. 179–194.

[35] N. J. A. Harvey, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman, "Skipnet: a scalable overlay network with practical locality properties,"

in *USITS'03: Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems*. Berkeley, CA, USA: USENIX Association, 2003, pp. 9–9.

[36] G. S. Manku, M. Bawa, and P. Raghavan, "Symphony: distributed hashing in a small world," in *USITS'03: Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems*. Berkeley, CA, USA: USENIX Association, 2003, pp. 10–10.

[37] D. Malkhi, M. Naor, and D. Ratajczak, "Viceroy: a scalable and dynamic emulation of the butterfly," in *PODC '02: Proceedings of the twenty-first annual symposium on Principles of distributed computing*. New York, NY, USA: ACM, 2002, pp. 183–192.

[38] D. H. Wolpert and W. G. Macready, "No free lunch theorems for search," Santa Fe Institute, Working Papers 95-02-010, Feb. 1995.

[39] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin, "Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the world wide web," in *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 1997, pp. 654–663.

[40] E. Horowitz, S. Sahni, and S. Rajasckaran, *Computer Algorithms: C++*. New York, NY, USA: W. H. Freeman & Co., 1996.

[41] V. Kumar, A. Grama, A. Gupta, and G. Karypis, *Introduction to parallel computing: design and analysis of algorithms*. Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc., 1994.

[42] S. Ateconi, D. Hales, and O. Babaoglu, "Broadcasting at the critical threshold in peer-to-peer networks," University of Bologna, Department of Computer Science University of Bologna Mura Anteo Zamboni 7 40127 Bologna (Italy), Tech. Rep., March 2007.

[43] V. Vishnevsky, A. Safonov, M. Yakimov, E. Shim, and A. D. Gelman, "Scalable blind search and broadcasting in peer-to-peer networks," in *P2P '06: Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 259–266.

[44] M. Ripeanu, A. Iamnitchi, and I. Foster, "Mapping the gnutella network," *IEEE Internet Computing*, vol. 6, no. 1, pp. 50–57, 2002.

[45] D. Zeinalipour-Yatzi and T. Folias, "A quantitative analysis of the gnutella network traffic," 2002.

[46] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yazti, "A local search mechanism for peer-to-peer networks," in *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*. New York, NY, USA: ACM, 2002, pp. 300–307.

[47] G. F. Coulouris and J. Dollimore, *Distributed systems: concepts and design*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1988.

[48] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," in *ICS '02: Proceedings of the 16th international conference on Supercomputing*. New York, NY, USA: ACM, 2002, pp. 84–95.

[49] S. Vuong and J. Li, "Efa: An efficient content routing algorithm in large peer-to-peer overlay networks," in *P2P '03: Proceedings of the 3rd International Conference on Peer-to-Peer Computing*. Washington, DC, USA: IEEE Computer Society, 2003, p. 216.

[50] B. Krishnamurthy, J. Wang, and Y. Xie, "Early measurements of a cluster-based architecture for p2p systems," in *IMW '01: Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*. New York, NY, USA: ACM, 2001, pp. 105–109.

[51] S. Saroiu, K. P. Gummadi, and S. D. Gribble, "Measuring and analyzing the characteristics of napster and gnutella hosts," *Multimedia Syst.*, vol. 9, no. 2, pp. 170–184, 2003.

[52] N. Leibowitz, M. Ripeanu, and A. Wierzbicki, "Deconstructing the kazaa network," in *WIAPP '03: Proceedings of the The Third IEEE Workshop on Internet Applications*. Washington, DC, USA: IEEE Computer Society, 2003, p. 112.

[53] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulié, "Scamp: Peer-to-peer lightweight membership service for large-scale group communication," in *NGC '01: Proceedings of the Third International COST264 Workshop on Networked Group Communication*. London, UK: Springer-Verlag, 2001, pp. 44–55.

[54] H. C. Li, A. Clement, E. L. Wong, J. Napper, I. Roy, L. Alvisi, and M. Dahlin, "Bar gossip," in *USENIX'06: Proceedings of the 7th conference on USENIX Symposium on Operating Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2006, pp. 14–14.

[55] M. Abdallah and L. Temal, "Gridb: A scalable distributed database sharing system for grid environments," Paris University, 8, rue du Capitaine Scott 75015 Paris, France, Tech. Rep., 2003.

[56] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," *ACM Comput. Surv.*, vol. 36, no. 4, pp. 335–371, 2004.

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

[58] ——, "Content-based peer-to-peer network overlay for full-text federated search," in *Proceedings of the Eighth Recherche d'Information Assistee par Ordinateur (RIAO) Conference*, 2006.

[59] H. Zhang, W. B. Croft, B. Levine, and V. Lesser, "A multi-agent approach for peer-to-peer based information retrieval system," *aamas*, vol. 01, pp. 456–463, 2004.

[60] H. Zhang and V. Lesser, "A dynamically formed hierarchical agent organization for a distributed content sharing system," in *IAT '04: Proceedings of the Intelligent Agent Technology, IEEE/WIC/ACM International Conference on (IAT'04)*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 169–175.

[61] ——, "Multi-agent based peer-to-peer information retrieval systems with concurrent search sessions," in *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. New York, NY, USA: ACM Press, 2006, pp. 305–312.

[62] K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, and R. Schmidt, "P-grid: a self-organizing structured p2p system," *SIGMOD Rec.*, vol. 32, no. 3, pp. 29–33, 2003.

[63] X. Bao, B. Fang, M. Hu, and B. Xu, "Heterogeneous search in unstructured peer-to-peer networks," *IEEE Distributed Systems Online*, vol. 6, no. 2, p. 1, 2005.

[64] E. Brunskill, "Building peer-to-peer systems with chord, a distributed lookup service," in *HOTOS '01: Proceedings of the Eighth Workshop on Hot Topics in Operating Systems*. Washington, DC, USA: IEEE Computer Society, 2001, p. 81.

[65] C. Kaufman, R. Perlman, and M. Speciner, *Network security: PRIVATE communication in a PUBLIC world*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2002.

[66] J. Li, J. Stribling, R. Morris, M. F. Kaashoek, and T. M. Gil, "A performance vs. cost framework for evaluating dht design tradeoffs under churn." in *INFOCOM*. IEEE, 2005, pp. 225–236.

[67] Wikipedia, "De bruijn graph," Internet. [Online]. Available: http://en.wikipedia.org/wiki/DeBruijngraph

[68] M. Castro, P. Druschel, Y. Charlie, and H. A. Rowstron, "Exploiting network proximity in peer-to-peer overlay networks," Tech. Rep., 2002.

[69] F. Dabek, J. Li, E. Sit, J. Robertson, M. F. Kaashoek, and R. Morris, "Designing a dht for low latency and high throughput," in *NSDI'04: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2004, pp. 7–7.

[70] WikiPedia, "Kademlia," WWW, June 2008. [Online]. Available: http://en.wikipedia.org/wiki/Kademlia

[71] A. Ghodsi, L. Alima, S. El-Ansary, P. Brand, and S. Haridi, "Self-correcting broadcast in distributed hash tables," 2003.

[72] A. Ghodsi, L. O. Alima, and S. Haridi, "Low-bandwidth topology maintenance for robustness in structured overlay networks," in *HICSS '05: Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 9*. Washington, DC, USA: IEEE Computer Society, 2005, p. 302.1.

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

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

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

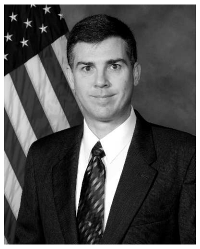
[76] P. Ganesan, K. Gummadi, and H. Garcia-Molina, "Canon in g major: Designing dhts with hierarchical structure," in *ICDCS '04: Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 263–272.

[77] S. Zoels, Z. Despotovic, and W. Kellerer, "On hierarchical dht systems - an analytical approach for optimal designs," *Comput. Commun.*, vol. 31, no. 3, pp. 576–590, 2008.

[78] Q. Hofstätter, S. Zöls, M. Michel, Z. Despotovic, and W. Kellerer, “Chordella - a hierarchical peer-to-peer overlay implementation for heterogeneous, mobile environments,” in *P2P ’08: Proceedings of the 2008 Eighth International Conference on Peer-to-Peer Computing*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 75–76.
[79] M. Datar, “Butterflies and peer-to-peer networks,” in *ESA ’02: Proceedings of the 10th Annual European Symposium on Algorithms*. London, UK: Springer-Verlag, 2002, pp. 310–322.
[80] S. Zöls, R. Schollmeier, W. Kellerer, and A. Tarlano, “The hybrid chord protocol: A peer-to-peer lookup service for context-aware mobile applications,” in *Networking – ICN 2005. 4th International Conference on Networking, Proceedings, Part II*, ser. Lecture Notes in Computer Science, P. Lorenz and P. Dini, Eds., vol. 3421. Berlin Heidelberg: Springer-Verlag, April 2005, pp. 781–792.
[81] S. Zoels, S. Schubert, W. Kellerer, and Z. Despotovic, “Hybrid dht design for mobile environments,” pp. 19–30, 2008.
[82] D. S. Wallach, “A survey of peer-to-peer security issues,” in *In International Symposium on Software Security*, 2002, pp. 42–57.
[83] O. Shehory and S. Kraus, “Coalition formation among autonomous agents: Strategies and complexity (preliminary report),” in *From Reaction to Cognition — Fifth European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW-93 (LNAI Volume 957)*, C. Castelfranchi and J.-P. Müller, Eds. Springer-Verlag: Heidelberg, Germany, 1995, pp. 56–72.
[84] S. P. Ketchpel, “Forming coalitions in the face of uncertain rewards,” in *National Conference on Artificial Intelligence*, 1994, pp. 414–419.
[85] E. Winter, *The Handbook of Game Theory*. North-Holldand, 2002, ch. The Shapley Value, pp. 2026–2052.
[86] O. Shehory and S. Kraus, “Task allocation via coalition formation among autonomous agents,” in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montréal, Québec, Canada, 1995, pp. 655–661.
[87] B. P. Gerkey, “On multi-robot task allocation,” Ph.D. dissertation, University of Southern California, August 2003.

Daniel R. Karrels is a Captain in the United States Air Force. He earned his B.S. and M.S. degrees in Computer Engineering at the University of Florida, and is a PhD student at the Air Force Institute of Technology. He is a member of the Tau Beta Pi and Eta Kappa Nu national honor societies. He is studying autonomous command and control in large scale distributed systems.

Gilbert ’Bert’ Peterson is an Assistant Professor of Computer Engineering at the Air Force Institute of Technology. Dr. Peterson received a BS degree in Architecture, and an M.S and Ph.D in Computer Science at the University of Texas at Arlington. He teaches and conducts research in digital forensics, and artificial intelligence.



Barry E. Mullins is an Assistant Professor of Computer Engineering in the Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson AFB OH. He received a B.S. in Computer Engineering (cum laude) from the University of Evansville in 1983, an M.S. in Computer Engineering from the Air Force Institute of Technology in 1987, and a Ph.D. in Electrical Engineering from Virginia Polytechnic Institute and State University in 1997. He served 21 years in the Air Force teaching at the U.S. Air Force Academy for seven of those years. He is a registered Professional Engineer in Colorado and a member of Eta Kappa Nu, Tau Beta Pi, IEEE (senior member), and ASEE. His research interests include cyber operations, computer communication networks, embedded (sensor) and wireless networking, and reconfigurable computing systems.