Peer-to-Peer Networking and Applications, December 2010, Volume 3, Issue 4, pp 294-322. The final publication is available at at Springer via http://dx.doi.org/10.1007/s12083-009-0061-7.



# A Conceptual Model for Data Management and Distribution in Peer-to-Peer Systems

Ramazan S. Aygün[1*], Yi Ma[2], Kemal Akkaya[3], Glenn Cox[4], and Ali Bicak[5]

1 Computer Science Department, University of Alabama in Huntsville, Huntsville, AL 35899 USA (phone: 1-256-824-6455; fax: 1-256-824-6239; e-mail: raygun@cs.uah.edu).* Corresponding Author.

2 Dept. of Computer Science, Virginia Polytechnique Institute and State University Blacksburg, VA USA (e-mail: may05@vt.edu).

3 Dept. of Computer Science, Southern Illinois University, Carbondale, IL USA (e-mail: kemal@cs.siu.edu).

4 Computer Science Department, University of Alabama in Huntsville, Huntsville, AL 35899 USA (e-mail: gcox@cs.uah.edu).

5 Dept. of Information Technology & Management Science, Marymount University, Arlington, VA 22207 (e-mail: ali.bicak@marymount.edu)

***Abstract*— While Peer-to-Peer (P2P) model gains significant attention in distributed computing, it is also expected to be a powerful model for information sharing. P2P systems are expected to provide exhaustive reliable computational resources and scalable accessibility. The data management and distribution in such systems requires storage, replication, data modeling, indexing, querying, retrieval, streaming, and topology management. While a lot of data management strategies have been proposed through the last years, these strategies have not been investigated with respect to a common model for P2P systems. However, since the services provided by the P2P systems are so diverse, it is very challenging to come up with a common layer-based model for all P2P systems. In this paper, we firstly propose a conceptual model for P2P systems, and then provide a classification and summary of data management and distribution strategies by referring to this model. The horizontal layers of the model correspond to modules of a P2P system whereas the columns are related to the services provided. The modules include base P2P service, storage, indexing, logical, service, and application modules. The services include security, querying, publish, join/leave, collaboration, and streaming. The paper concludes by providing a comprehensive list of data management and distribution strategies used in the existing P2P systems.**

***Keywords*—Peer-to-peer system; Data management; Data distribution**

# 1 INTRODUCTION

Peer-to-Peer (P2P) systems have brought great attention with Napster (Napster), (Shirky, 2001) after several years booming of the Internet around 1995. Before the P2P model, the dominant Client/Server model (CSM) had formed an asymmetric distributed environment that accommodated various profitable applications. However, CSM provides end-users only limited capability in resource sharing. In addition, depending merely on servers greatly reduces scalability and reliability since servers become bottleneck and single point of failure in Internet applications. The better utilization of idle resources at the edge of Internet also goes far beyond the capability of CSM. Therefore, P2P systems are expected to be a major component in future Internet applications.

The power, simple, and free-use of P2P systems has attracted millions of internet users. The basic services of P2P systems can be identified as: join, publish, search, and fetch. Most of the P2P systems enhance these services to match their application requirements. A P2P system distributes the data to be published on the computers of users automatically. During this process, the user intervention is minimal as opposed to applications like web page development and registration. The available data on P2P systems with respect to the data on the World Wide Web are incomparably huge. In addition to vast amount of available data, anonymity while accessing and delivering data is charming many Internet users worldwide. Most of the shared data are in the form of files and in fact the initial P2P systems like Napster, Gnutella basically emerged as file sharing systems. However, the applications of P2P systems include more services than file sharing such as streaming, data management, and collaborative working environments.

Although P2P systems work on the same (or similar) resources, it is hard to talk about any agreement on types of data management and distribution strategies to be used among various types of P2P systems. However, certain layers are already identified in order to perform specific jobs in a P2P system. The lower layers are related with system specific issues like peer join and leave whereas upper layers are concerned with application specific issues like information retrieval and data streaming with quality of service (QoS). Nowadays, it is possible to see reusability at the lower layers (e.g., some P2P systems are based on CAN (Ratnasamy, 2001a), Chord (Stoica, 2001), JXTA (JXTA)) however; it is hard to talk about the same type of reusability at the upper layers. Since there is limited or no reusability at upper-layers, it is difficult to come up with a standard layering. The ultimate layer independence might be impossible. However, if the interaction between layers or modules can be identified, we claim that more efficient P2P systems can be built very fast in collaborative environments.

Layer-based models have the advantage of developing and focusing on individual components of a system independently while improving the reusability of the approaches developed at different layers. Layered architecture enables incorporation of new protocols and services to support growing number of users (Milojicic, 2002). For example, user datagram protocol (UDP) may be preferred at the transport layer with respect to Open Source Interconnection (OSI) model for real-time video streaming. Can we answer the question on which data management strategies to be used for multimedia information retrieval and streaming in P2P systems easily? Although P2P systems provide peer-to-peer communication at the lower layers, there are no peer defined interactions among modules at the upper layers for data management (e.g., for multimedia streaming). Therefore, identification of modules, responsibilities of modules, and how services use these modules is more than a necessity for P2P systems to increase the reusability of developed strategies.

However, considering the data management and distribution strategies implemented by current P2P systems, like Napster (Napster), Gnutella (Adar, 2000), KaZaA (Kazaa), Edutella (Nejdle, 2002), CAN (Ratnasamy, 2001a), Chord (Stoica, 2001), PeerDB (Ng, 2003), FreeNet (Clark, 2000), etc., we observe that the strategies tend to be scattered in many ways due to the lack of a well accepted consensus on what features is best preferred in certain P2P systems and what is really expected from P2P systems. Typically, each P2P application develops its own data management and distribution strategies to provide the best service to its users. Therefore, there is also a need to classify, analyze and survey data management strategies with respect to a common model.

The organization and survey of P2P systems is challenging due to sheer number of contributions. It is possible to find applications of P2P systems in almost all research areas. Our search on the Collection of Computer Science Bibliographies for publications having P2P or peer-to-peer only in their titles returned around 9500 publications in September 2008. Obviously, it is impossible to incorporate even the majority of these papers into a survey. Based on

this fact, Risson and Moors (Risson, 2006) emphasized two important points on generating surveys for future P2P research: a) Generating surveys on specific areas of P2P systems like security and storage and b) Generating surveys on the full body of related research that cuts across numerous research disciplines. In this paper, we target the second point. Note that, comparing all the possible approaches in numerous research disciplines for P2P systems is more challenging than targeting a specific area for P2P systems. While we focus on data management and distribution for P2P applications, we also cover most of the research areas such as indexing, storage, security, and streaming for P2P systems specified in (Risson, 2006).

The main goal of this paper is to provide a conceptual model for P2P systems based on the current research on P2P data management and distribution. To show the practicality and applicability of our conceptual P2P system model, we classify the data management strategies with respect to our model. However, we cannot provide in-depth analysis and discussion of these data management strategies since in-depth coverage of each strategy requires a specific survey. In addition, there are already surveys conducted on specific areas of P2P systems such as search methods (Risson, 2006), overlay network schemes (Park, 2008), (Lua, 2005), storage techniques (Hasan, 2005), databases (Bonifati, 2008), video streaming (Liu, 2008), computing (Milojicic, 2002), content distribution (Andtroutellis-Theotokis, 2004), and security (Wallach, 2002). In this paper, we aim at providing the big picture on a conceptual model in order to visualize the commonalities and discrepancies among different data management strategies developed for specific application functionalities. The horizontal layers of our conceptual model correspond to modules such as base P2P service, storage, indexing, logical, support, and application modules. The columns correspond to services such as security, querying, publish, join/leave, collaboration, and streaming. While we investigate data management issues, we also explain the relationships among the modules of our model. To the best of our knowledge, this is the first paper that provides a conceptual model for P2P systems and investigates data management and distribution strategies with respect to a model.

The contributions of this paper are as follows:

- Providing a survey on the full body of related research that cuts across numerous research disciplines for P2P systems,
- Developing a conceptual model for data management and distribution in P2P systems,
- Studying and classifying various research techniques according to the proposed model, and
- Promoting the reusability of developed techniques for not only lower levels of the P2P systems but also for upper levels including indexing, storage, and logical modules.

This paper is organized as follows. In the following section, we give an overview of current models for P2P systems. Section III describes our conceptual P2P system model. Section IV explains modules and services of P2P systems. The modules include base P2P service, storage, indexing, and logical modules. The services include querying, streaming, and security. Section V provides a discussion and summary of the proposed techniques. The last section concludes our paper.

## 2 A BRIEF REVIEW OF LAYERED P2P SYSTEMS

P2P model expects two significant features from peers (Milojicic, 2002): a) to share their resources, information, processing, presence, etc. and b) to be able to handle a limited connectivity (wireless, unreliable modem links, etc.), support possibly independent naming, and share the role of the server. A P2P network is based on a P2P model and is defined as an overlay network of peer-level computing resources that are self-organized, use decentralized control, share resources, and may join or leave the network at will. Peers are interconnected by a data network and their communication patterns are defined by a logical or overlay system architecture. A connection between two peers may be implemented by multi-hop physical connections in the underlying communication system. Current P2P systems are always organized around a specific service and there is no general purpose P2P system. A P2P system that provides system-dependent services such as storage, indexing, and querying, is built over a P2P network.

The current layer-based approaches for P2P systems are primitive, abstract, (usually) application-specific, and far from satisfying the requirements of P2P data management and distribution systems. In this subsection, we will briefly summarize the current layer-based architectures for P2P systems.

### 2.1 Three-Layer Architectures

In general, a peer is conceptually divided into three layers: application, service, and core (Osais, 2006). The core layer is responsible for basic P2P services like peer join and leave. The service layer bridges the gap between the core and application layer and provides basic services like indexing, searching, and sharing. The application layer provides the user interface for application specific services. The P2P software delivery system (Turcan, 2002), JXTA model (JXTA), P2P group communication system (Conti, 2006), P2P distributed directory (Cordasco, 2004), PROST (Portmann, 2004), Open P2P Network (Harwood, 2004), Multi-layer resource sharing system (Dan, 2005), and AXMEDIS P2P node (Bellini, 2007) are all composed of three layers. Although different names are used for each layer, these layers have the same purposes.

The layer-based architecture for software delivery with P2P systems (Turcan, 2002) has three layers as P2P communication layer, software distribution middleware layer, and an upper layer composed of application (GUI), databases, and knowledge base. The multi-layer resource management architecture (Dan, 2005) is composed of three layers: peer layer, portal layer, and user layer (Dan, 2005). Peer layer maintains the P2P network; user layer is responsible to receive queries from users and then returning back to them; and portal layer is between peer layer and user layer and has the capability of storing and returning the results in case a user leaves and connects to the system in the future.

The JXTA model has three major layers: core layer, service layer, and applications layer (JXTA). The core layer provides basic services like peer join and monitoring; the service layer provides services like file sharing, indexing, and searching whereas the application layer is the layer for specific applications like music sharing. The OPEN P2P Network architecture presented in (Harwood, 2004) has three layers: connectivity, core services, and application layer. The connectivity layer is responsible for object management and message routing where different protocols can be utilized. The core services layer is responsible for generic services like peer management. The application layer is responsible for providing standard interfaces for core services.

The programmable peer node of PROST has three layers (Portmann, 2004): key-based routing layer, programmable peer, and application/service layer. The key-based routing layer is responsible for mapping objects to live nodes and locating them in the P2P system. The programmable layer provides the Application Programming Interface for the application/service layer. In (Cordasco, 2004), a P2P distributed adaptive directory with three layers is provided. These layers are bottom (CHILD), middle (DAD), and top (MOM) layer. The bottom layer provides the basic functionalities of a P2P system. The middle layer is responsible for providing useful information from the bottom layer to provide the top layer. The top layer is actually the application layer to manage ontologies.

### 2.2 Multi-Layer Architectures

In addition to three-layer P2P architectures, there are a few P2P architectures that support more than three layers. For instance, in (Osais, 2006), a layer-based P2P collaborative system having four layers (application, workspace, session, and communication) is presented. Their system is built on top of JXTA (JXTA). This JXTA layer is serving for core services of P2P system. The communication layer is on the top of JXTA and it provides services like reliable message transportation among peers and connection to JXTA network. Session manager provides the join of peers to session for collaborative working. Workspace manager deals with the consistency of objects in a session. The application layer hosts shared Web browser application providing telepointers and chat. The layer-based model (Christensen, 2006) for light peers in mobile P2P applications has five layers: network layer, P2P transport layer, service layer, application layer, and actuator/sensor layer. The network layer is assumed to exist under core P2P layer in all P2P systems. P2P transport layer provides the core services for P2P systems. The only new layer in this system is actuator/sensor layer that has a GPS unit and electric motor.

In (Milojicic, 2002), an informal architecture for P2P systems having the following five layers is described: communication, group management, robustness, class-specific, and application-specific layers. Communication layer connects the peer to the underlying network and is responsible for peer join/leave. The group management layer is responsible for the discovery of peers as well as locating and routing of messages through optimizing paths of messages from peers to peers. The robustness layer provides security, reliability, and resource aggregation. Security is about giving access to peers that are authorized to access the data. Reliability is related with redundancy management. Resource aggregation ranges from file sharing to storage and CPU power sharing. Class-specific layer is responsible for scheduling, metadata,

messaging, and management. Scheduling is related with dividing a task into multi-tasks and finding the peers for each task. Metadata is related with the management of content. Messaging is usually used in collaborative environments. Management is related about the topology of the P2P network. Application-specific layer has tools, services, and user-interfaces like file-sharing, chatting, etc. Note that this proposed architecture is an informal architecture without imposing any strict ordering of these layers.

## 3 A CONCEPTUAL P2P MODEL

### 3.1 A Preliminary Modular Abstract P2P Model

P2P systems are usually treated as a network system rather than a data management system. In P2P data management systems, the functionalities of data management and networking is so mixed that it is very hard to build a layered model for P2P data management systems. Since previous approaches on P2P data management systems have not been built based on following a specific, common model, it is hard to extract these layers. In this sense, the functionality of some specific problems may range multiple layers or components.

Since P2P systems provide a variety of services built upon diverse systems, the classification of developed approaches for specific applications is not easy. Data management in P2P systems is critical and heavily depends on the topology of an overlay network unlike federal or distributed databases.

We have surveyed the data management and distribution techniques in P2P systems. Based on the outcome of this survey, we concluded that a P2P data management system should have 8 modules as follows: *base P2P service, storage, indexing, logical, querying, security, support,* and *application modules.* This model is based on the data management and networking system models with emphasis on data management. We also consider the applications and services provided by current P2P systems. In terms of data management, the three-level schema architecture is used as the basis. There are three levels: external level, logical (conceptual) level, and physical (indexing) level. External level is related to the requirements of the application and is not used explicitly in our model. The physical level is divided into indexing and storage modules. Since a P2P system is a distributed system, we believe it is a better idea to have a separate module for storage and thus have separated the storage module from the indexing module. Below the storage module, we need a module to satisfy base P2P Services like naming, join/leave etc.. Moreover, modules for querying and streaming were considered to meet the specific requirements of querying and streaming. A support module was needed to achieve the requirements of the P2P application such as streaming and collaboration. To be consistent with the layered models, we have a specific module for the application.

Based on our investigations on the existing data management techniques in P2P system, we have come up with an architecture for P2P data management and distribution systems as shown in Fig. 1. The brief description of modules and their interactions are as follows:

- The *base P2P service module* provides basic peer services like peer ID, membership, and message passing, and is responsible for maintaining the topology of the P2P network. Base P2P service module is closely related with the topology of the network and how peers join and leave the P2P network. However, the details of how peers leave and join the network as well as topology management are beyond the scope of the paper. Base P2P service module is closely related with indexing, storage, and security modules. The interactions are listed with the other modules.
- The *storage module* serves other modules by providing proper data storage. This module is mostly responsible for data naming, replica, and redundancy management. The storage module is closely linked with indexing, base P2P service, support, and security modules. The support module has to make sure that application specific services like Quality of Service (QoS) should be satisfied by at least proper distribution of the data in the network. The security module expects the data to be recovered when the portions or some of the replicas of the data are corrupted by the malicious peers. The storage module relies on base P2P service where to store data.
- The *indexing module* is responsible for management of indices for resources and data. The indexing module is closely linked with the logical, storage, base P2P service, and security modules. The logical module determines the granularity of data to be indexed. The storage module and the topology enforced by the base P2P service module enforces where the data needs to be stored. The security module comes

into picture when data is partitioned to increase availability of data.

- The *logical module* is responsible for the data modeling and organization. The methodology to be maintained for the logical module may be even critical in the implementation of the base P2P service module. The logical module identifies the smallest size of data that can be queried and indexed from the perspective of the querying and indexing modules.
- The *querying module* is related with data accessing functionalities such as querying. Querying module is related with support, logical, and storage modules. The support module may provide services like caching to the querying module. The logical module restricts the queries whether the data can be queried as a whole entity or content-based. The logical module also determines the type of querying languages that can be used for the retrieval. The storage module may replicate the data over the peers in a way that the query results can be retrieved to peers quickly.
- The *support module* is responsible for system specific requirements like Quality of Service (QoS) as in multimedia systems and session management in collaborative P2P systems. The support module is closely linked with the application, querying, storage, and security modules. The application module should have the graphical user interface to deliver the services of the P2P system to the system. The support module may require the storage module to replicate the data to meet QoS requirements. The support module may provide services for encryption for the security module. It may provide services to optimize queries or cache query results.
- The *security module* is responsible for the reliability of the data, peers, and the P2P system. It can be incorporated at different levels and various relationships with other modules. The support module can encrypt the data before inserting to the system. The indexing of the data at indexing module when data is partitioned to increase the accessibility when some of the partitions are corrupted and where to store the segments can protect the data from malicious peers are possible interactions with the indexing and storage modules. The security can also be maintained at the base P2P service module by encrypting through a chain of peers before publishing in the P2P system.

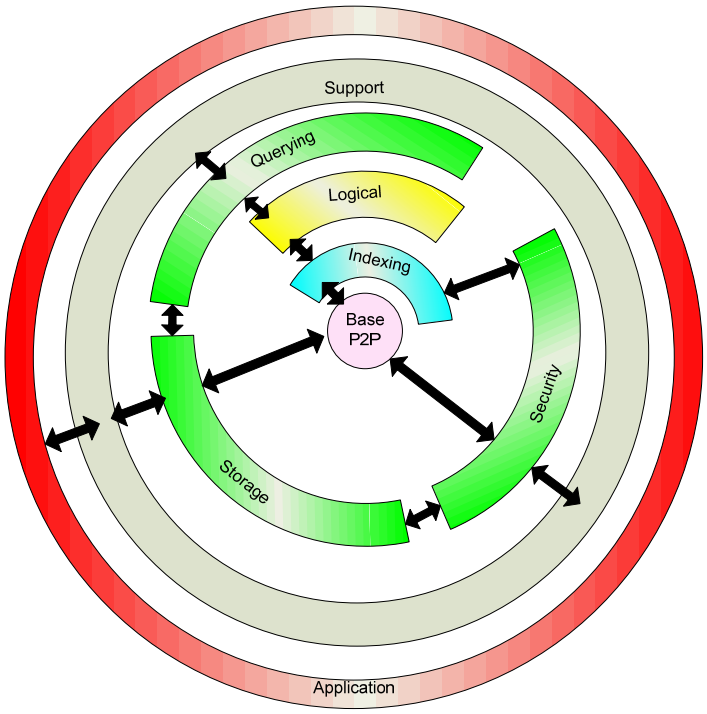


Fig. 1. A complex P2P Data Management System Model

- The *application module* provides application dependent services like user interface. It may have an application dependent user-interface where the user can submit queries to the system as well as visualize the results of the queries. This module connects the user to the P2P system.

However, this type of architecture is very complex and not beneficial to help future P2P designers find a standard framework for design decisions. On the other hand, the three-layer architecture is too simple for P2P data management systems since it does not have any layers that target data management. Therefore, we propose a sophisticated conceptual model for P2P data management systems as shown in Fig. 2. The reason we give such a model is that we consider it is beneficial for the future of P2P to reach a common reference model to help future P2P designers find a standard framework for design decisions.

### 3.2 Modules and Services in Conceptual P2P Model

In our conceptual model, the modules are represented as rows whereas the services provided by P2P systems are provided as columns. The services are related to services such as join, search, publish, fetch provided by P2P systems. The modules are as follows:

- The base P2P service module provides basic peer services like peer ID, naming, membership, and message passing.
- The storage module is mostly responsible for data naming, replica, and redundancy management.
- The indexing module is responsible for indexing of resources and data.
- The logical module is responsible for the data modeling and organization.
- The support module is responsible for system specific requirements like Quality of Service (QoS) as in multimedia systems and session management in collaborative P2P systems.
- The application module provides application dependent services like user interface. This module connects the user to the P2P system.

The services of our conceptual model are as follows:

- The security service is responsible for the reliability of the data, peers, and the P2P system.
- The querying service is related with data accessing functionalities such as querying.
- The streaming service provides streaming of data from a set of peers to another set of peers.
- The publishing service enables peers to publish and enable their data for access to other peers.
- The Join/Leave service enables a peer to join and leave a P2P system.
- The collaboration service enables a set of peers to communicate and work on a set of shared objects in a consistent manner.

The numbers in the grid shows a possible ordering of how modules can be used for specific services. Since this type of modular representation has not been an issue for previous systems, not all P2P systems are expected to conform to such representation. From Fig. 2, it is also seen that not all modules are used for all services.

### 3.3 Use of Modules by Services in Conceptual P2P Model

In this subsection, we give an overview of how modules interact with each other in our model for providing different services. We have to mention that not all modules are used in all P2P systems. This is just an approximation of services and possible interactions among modules. The ordering is not strict either but it may serve as a guide for further improvements on the architecture. For each service, we describe which modules are used and how they interact.

*Security.* 1) The application module gets input from the user whether the data need to be encrypted or not. 2) The support module provides encryption methods for the data (e.g. self-certifying data, erasure code). 3) The storage module identifies where to store the blocks of data. 4) The base P2P service module actually stores the data at the desired locations.

*Querying.* 1) The application module provides user interface for querying. 2) The logical module identifies the query language to be used for the query. 3) The support modules provide services like schema mediation. 4) The indexing module provides information about where to locate the data. 5) The base P2P service module connects the

host peer and retrieves the data. 6) The storage module may replicate data on the route from the host peer to the current peer.

*Streaming.* 1) The application module provides the user interface for the selection and (dis)play of streams. 2) The support module provides services such as buffering to ensure that the stream can be displayed smoothly. 3) The indexing module is responsible for the management of the structure of streaming hierarchy where applicable. 4) The base P2P service module transmits the data from peer to peer and manages how peers should join the P2P network.

| | Security | Querying | Streaming | Publish | Join/Leave | Collaboration |
|---|---|---|---|---|---|---|
| Application Module | 1 | 1 | 1 | 1 | 1 | 1 |
| Support Module | 2 | 3 | 2 | 2 | 2 | 2 |
| Logical Module | | 2 | | 3 | 3 | |
| Indexing Module | | 4 | 3 | 4 | 4 | |
| Storage Module | 3 | 6 | | 5 | | |
| Base P2P Service Module | 4 | 5 | 4 | 6 | 5 | 3 |

Fig. 2. Our proposed conceptual P2P System Model

*Publish.* 1) The application module provides the user interface to select the items to publish. 2) The support module provides services like keyword extraction for the data item. 3) The logical module makes sure that the data is compliant with the local schema and may use wrappers to map the schema to a desired schema in schema-based P2P systems. 4) The indexing module provides the information where the data need to be stored. 5) The storage module makes sure that the data are replicated at various locations to improve the performance. 6) The base P2P service module communicates with other peers and transfers and forwards the data to the locations identifies by the indexing and storage module.

*Join/Leave.* 1) The application module provides the user interface to join a P2P system. 2) The support module may provide services such as discovering available groups. 3) If semantic communities need to be established based on the content of a peer, the necessary information about the content needs to be extracted. 4) Based on the content and peer name, the indexing module assigns an identifier for the peer. 5) Base P2P service module eventually connects the peer to the P2P system.

*Collaboration.* 1) The application module provides functionalities to join a collaborative environment. 2) The support module is responsible for management of sessions as well as the consistency of the items. 3) The base P2P service module provides services like join, leave and messaging among peers.

Next, we will explain in detail the base P2P service, storage, indexing, and logical modules in different sections. We do not provide specific sections to application & support modules since they are usually application specific. We rather mention them in other sections whenever we feel it is necessary. In addition, we will also discuss some modules such as security, querying, and streaming in different sections. We have not dedicated a specific section for collaborative services since there is not significant number of papers in this area. We briefly mention join/leave in the next section and also state in other modules as join/leave is important for the corresponding module.

# 4 P2P MODULES AND SERVICES

## 4.1 BASE P2P SERVICE MODULE

The first step to join a network is to get an identifier to be distinguished in a P2P network. A P2P system has to provide identifiers (or names) to its resources. The resources for P2P networks are peers, data, groups etc. (JXTA). P2P systems can be categorized into centralized and decentralized (Lv, 2002) based on the structure of the P2P system. In centralized P2P systems peers issue their queries to a central server to determine the peers that have the data (e.g. Napster (Napster)). The decentralized systems can be categorized as unstructured and structured (Lv, 2002). In structured P2P systems, the peers may join the P2P system following the predefined structure of the P2P system and data are located at specified locations for efficient querying. In unstructured P2P systems, there is usually no constraint on how peers join and leave, and also how data are located in the system. In all of these P2P systems, peers are the fundamental resources.

In this section, we are going to look at briefly on peer naming. The data naming and storage are considered under the responsibility of the storage module. Usually naming strategy is similar for different types of resources. The naming strategy depends on whether a P2P network is structured or not. Joining, leaving a P2P system and messaging among peers are some of the responsibilities of the base P2P service module.

### *4.1.1 Peer Naming*

Peer naming can be classified as secured or unsecured naming (Fig. 3). This type of classification exists in Windows Peer Name Resolution Protocol (PNRP) (Windows P2P). The peer names (identifiers) are usually assigned independent of the physical address even though the physical (IP) address of a peer may change. Whenever a peer connects the system, it always has the same name.

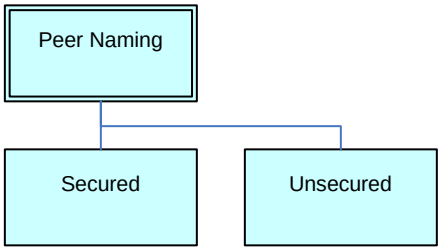


Fig. 3. Peer Naming strategies

**Unsecured Naming.** The peer (user) assigns its name as the user wills. The user can assign any name as long as it is at least 3 characters in PNRP (Windows P2P). Insecure names are the names provided by the hosts. More interestingly, a single peer may provide multiple names for different services through classifier attribute in peer names. However, there is no guarantee that same names will not be used by different users. This type of naming is risky since when connecting to a peer, it is not guaranteed that the peer to be connected is the actual peer to be connected. This type of naming is only recommended for *private* P2P systems that require certain membership, censorship, and consist of peers from recognizable organizations.

**Secured Naming.** Secure name is generated by mapping an input string to a key using a (hash) function. JXTA (JXTA) uses universally unique identifier (UUID) to name its resources. UUID is a random generated 128-bit string that is very unlikely but not guaranteed to have the same resource names. Microsoft Windows Vista provides Peer Resolution Name Protocol (PRNP) (PNRP) for P2P services. The PRNP requires the use of IPv6 and allows the publishing of peer names along with IPv6 addresses. The hosts can use secure or insecure names where there is no guarantee that insecure names will not be used for more than one peer. Secure names are generated by using secure hash function of a public key. In DHT-based P2P systems, each peer is assigned a key using a hashing function such as SHA-1.

### *4.1.2 Peer Join/Leave*

The most basic service a P2P system must provide is a mechanism to join the P2P network. Based on the structure of a P2P network, there are multiple strategies to join a P2P network. In unstructured P2P networks, a peer is allowed to connect any peer in the P2P network. In structured P2P networks, a peer can only connect the network through specific peers in the network. In DHT-based systems, peers are assigned a key. This key determines the successor and predecessor peers in the network (in other words, the neighbors of a peer). If semantic services are provided by the P2P system, the peers are allowed to have communities. In such a case, peers join communities if they provide similar information as peers in the same community.

**Discussion.** The advantage of unsecured naming is its simplicity and less computational complexity. Its disadvantage is the difficulty of the management of anonymity and indefinite size of the name. The advantage of secured naming is the enabling of regular and uniform names. In addition, secured naming is useful for the implementation of

anonymity. Unsecured peer naming should be preferred in private systems since the same peer name might be used by multiple peers. Hybrid naming gives the option of using implicit or explicit naming. In small groups, implicit naming (of hybrid) may be preferred since the name is provided by the user and possibly easy to remember. In large P2P systems, the collisions of (explicit) peer names are likely to happen since there is no verification whether the name is being used or not. On the other hand, secured naming provides unique names although they are difficult to remember given that secured names include the hash value part of the name.

## 4.2 STORAGE MODULE

Storage module is mainly concerned with the physical organization and distribution of data in P2P systems. Goals of P2P storage module include: load balancing through distribution, improved availability through off-site backup, anonymous storage service, scalability, and avoiding hot spot through decentralization. Free Haven (Free Haven), OceanStore (OceanStore), PAST (Rowstron, 2001a) (on top of Pastry (Rowstron, 2001b)), CFS (Dabek, 2001), Intermemory (Goldberg, 1998), and Farsite (Adya, 2002) are sample systems that provide storage services. This module is mostly responsible for data naming, replica, and redundancy management. The functionalities of the storage module are heavily dependent on the P2P system application.

The storage module has to make sure that application specific services like Quality of Service (QoS) with streaming should be satisfied by at least proper distribution of the data in the network. The security service expects the data to be recovered when the portions or some of the replicas of the data are corrupted by the malicious peers. The storage module is critical in the performance of query propagation process of the querying service to efficiently locate and retrieve data objects.

### *4.2.1 Data Naming*

Data naming is the first step of the data placement (to identify and locate) problem. The indexing module provides index for the data based on the granularity of the data specified in the logical module. This index keeps information about the data to be named. Base P2P service module provides the naming service for the indexing module. Although we explain data naming service under storage module, it is closely related to the indexing module and may be moved to the indexing module.

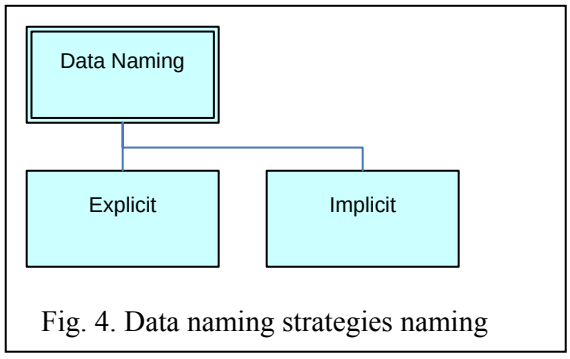

Fig. 4. Data naming strategies naming

Naming mechanism is the first issue to be solved in data management. In order to name a resource (e.g. files), a traditional operating system (OS), LAN, and Internet use file name (or path name), host name plus path name, and Uniform Resource Locator (URL), respectively. Some systems use Uniform Resource Identifier (URI) to name any object in P2P systems. In early development of P2P systems, two arguments about naming mechanism have been broached by researchers: explicit naming service and implicit naming service (Fig. 4).

***Explicit Naming Service.*** This naming service uses host name, peer ID, URI (only locally used without support of services like DNS), or IP address plus the file name. Napster employs this method. This is almost borrowed from LAN solution directly. We call this approach as *direct* or *explicit* naming.

***Implicit Naming Service.*** Many P2P systems choose to build a community around certain services, in which members coordinate in a P2P fashion. As the community is somewhat "local", their name service is also limited within the community. The implicit naming, maps the object (includes peer IP address, file name, or even file content) to a keyword using a hash function. For example, CAN (Ratnasamy, 2001a) use SHA-1 (secure hash algorithm) to hash both peer IP address and file content into keywords and keeps files on those nodes having the same or successor keywords. DHT-based methods all lie under implicit naming service.

**Discussion.** The advantage of explicit/implicit naming is analogous to unsecured/secured naming. The advantage of explicit naming is its simplicity and less computational complexity. Its disadvantage is the difficulty of the management of anonymity and indefinite size of the name. The advantage of implicit naming is the enabling of regular and uniform names. In addition, implicit naming is useful for the implementation of anonymity. Its disadvantage is the necessity for extra computation, and some distributed hash table (DHT) based systems aggravates

the data updating. The explicit data naming is usually preferred in centralized P2P systems whereas implicit naming is usually preferred in decentralized networks.

*4.2.2 Dynamic vs. Static Storage*

We classify P2P storage policies into two groups: dynamic and static storage (Fig. 5). In dynamic storage, when the content of file changes, the whole file changes its storage place whereas static storage policy does not require the change of location. The only necessary thing is to hash the content into a key. Since the key is combined with the file storage, the key can be used to find the file directly. PAST, CAN, and Chord employ dynamic storage. Although it arose from DHT based systems, some DHT based systems tried to avoid floating storage. Ivy (Muthitacharoen, 2002) and OceanStore try to avoid floating storage. Ivy uses logs of updates rather than updating the original data whereas OceanStore uses versioning of data. CFS only allows updates by the publisher. FreeNet overwrites the original data using the public key of the document. Eliot (Stein, 2002) requires the change of address in the underlying P2P network if update occurs.

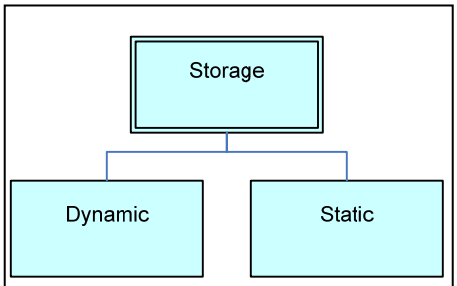

Fig. 5. Storage strategies in P2P systems.

***Structured vs. Unstructured P2P Systems.*** In structured P2P systems, a predefined topology imposes restrictions on peers on the assignment of their neighbor peers. Moreover, a data item, either itself or its abstraction (index, advertisement, etc.), can be stored only on predefined node based on a key obtained from its content. In many P2P systems, such as Napster, Edutella, Gnutella (Adar, 2000), and PeerDB, storage strategies resemble the strategies in file systems and database systems. Content-to-address combination is widely used in structured systems. Such a combination might cause limitations over write operations of data within the P2P system. The concepts of structured P2P systems arise from systems like CAN and Chord. In Chord, for instance, each peer, as well as data item, is assigned a hashed identifier through consistent hash function (SHA-1). Node forms a Chord-ring according to their identifier. Data items are assigned to nodes that have the same hashed identifiers or successor identifiers. In CAN, nodes and data items are mapped into some zones according to hash table. Each node of CAN possesses a set of hash keys. And files in CAN are hashed onto some specific key and stored on the node that possesses this key. Therefore, if a write operation on a file changes its hash key, it also changes the node on which the file resides in. Since it may cause data movement on peer join or leave, it makes data updating more expensive. However, its advantage in efficient data locating makes it competitive to unstructured systems. The advantage of unstructured P2P system is that it is very simple to build. There is no extra computation for peer join or leave. The disadvantage is its high cost on data location.

***Hash-based Systems.*** Normally hash-based systems are dynamic storage systems. If a hash-based P2P system does not provide update mechanism, the update problem is not considered as an issue to be resolved. However, such a hash-based P2P system is still a dynamic storage system if no additional caution is taken. The basic caution in no-update systems is to consider an updated object as a new object and then to insert it into the system. It is also possible that when a new object enters the system, it is given a signature and all the updates of the object are bound to its original signature (like versioning in OceanStore).

**Discussion.** The advantage of dynamic storage is quick location of a file. The disadvantage of dynamic storage is that it makes file updating cost much more expensive. The advantage (disadvantage) of dynamic storage is the disadvantage (advantage) of the static storage. The P2P application developers have to decide on how frequent updates are likely on the data and then decide the storage strategy. For example, multimedia data is usually accessed rather than updated.

*4.2.3 Redundancy Management*

Improved availability and reliability are two important focuses on which P2P systems are concerned to surpass Client/Server model. However, P2P systems' decentralized and volatile characteristics become major barriers to realize them. Moreover, the capacity of individual peers, stability of membership and (heuristic) information for searching are not powerful in P2P systems with respect to client/server systems. These make the above two highlights of P2P systems hard to realize. As a consequence, most P2P systems use redundancy in keeping low access latencies and good load balancing to improve availability and reliability. In some applications, such as real-time multimedia streaming, nearby replicas also help improve bandwidth load balancing.

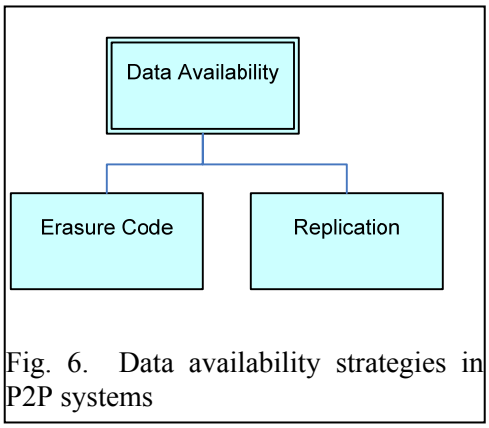


Fig. 6. Data availability strategies in P2P systems

In P2P systems, replication and erasure code are two popular mechanisms in providing data redundancy (Fig. 6). Redundant control protocols could be built either as a stand-alone service or as an auxiliary supporting mechanism for placement service. Erasure code (Lin, 2004) can be thought of dividing a data into *m* segments in a way that the data can be reconstructed from *n* segments. There are six questions that setup the framework of redundancy control in P2P systems: *when* to create, *which* data to replicate, *where* to store, quantity, duration, and maintenance of replicas (Fig. 7).

Good cost model and gain model of redundancy control are underlying stones for all the approaches to analyze system behavior, constitute justified strategies, and measure their performance. Typically, the cost model needs to consider following costs: storage space for replicas, bandwidth to transfer replicas, and cost to maintain data consistency. The gain model needs to consider the following factors: performance in different query patterns, reduction in query latency, and distribution rate of load.

*Replica creation.* There are mainly two strategies according to the creation time. The first is to duplicate when data first enter into P2P system (Maniatis, 2005). We name it as proactive replication. The second is to duplicate when data are requested (Lv, 2002). We call it reactive replication. DiCAS (Wang, 2006) provides an example of reactive replication where queries are forwarded to a predefined group.

*Data to replicate.* There are three approaches: replicate all data, replicate only data that are requested by others, and only replicate those frequently requested data (hotspot). Typically, proactive replication replicates all data items, while reactive one replicates either all requested data or hotspot data. Online Pointer-Replication (OPR) system (Zhou, 2008) replicates the pointers (locations of resources) rather than data

*Replica storage.* There exist many strategies: replicate data on *k* nearest neighbor nodes, replicate data on those nodes that are on request route (e.g., Oceanstore, Freenet), randomly select nodes to replicate data, and so on.

*Replica quantity.* There are mainly three approaches: uniform, proportional or query-sensitive, and probabilistic replication. The uniform replication replicates all data items equally. Proportional replication or query-sensitive replication chooses the number of replicas according to the frequency of the data requests. It is verified that both strategies give poor query latency gain in reacting to some query pattern (Cohen, 2002). If the queries to data items constitute some hotspot, the uniform replication improves query latency. On the contrary, when the queries are evenly distributed among data items, the proportional replication does not gain much in latency. The probabilistic replication lies between the above two (Tsoumakos, 2003), (Tsoumakos, 2006). Square-root replication is an example of the probabilistic replication.

*Replica durability.* The durability of replicas in the system could be either temporary or long-term. In some special purpose P2P systems that provide permanent storage service, replicas may even exist permanently. It is also possible that P2P system sets a duration span for each replica. After the duration expires, the replica is deleted directly (e.g., Free Haven).

*Replica consistency.* The major operation in maintaining data consistency is to update replicas after its original copy has been modified. There are mainly two approaches for updating: instant and on-demand. In instant update, whenever the original copy is modified, the peer is required to send update message to all replicas. This also implies that the original peer needs to keep track of replicas in some way. The original peer will always be advocator of updating. In (Fahrenholz, 2004), a group of peers has at least one coordinator and the coordinator is responsible for updates. However, they provide weak data consistency since the peers in a group as well as the coordinator may change. In on-demand update, the replica is updated only when it is requested by other peers. This requires the peer holding replica keep track of the original copy. This also generates extra cost for each data request that reached the replica.

**Discussion.** *Creation of Replica.* Proactive replication may use unnecessary system resources due to possible replication of rarely accessed data. On the other hand, reactive replication needs to keep track of data that are frequently accessed and then replicate the data as requested or needed. *Storage Places.* The storage places depend on the access patterns of data by peers. A specific access pattern for data may benefit from specific data storage policies (e.g., *k*-nearest neighbor). *Quantity.* The probabilistic replication strategy is expected to perform better than uniform and proportional replication strategies. *Durability.* If the replica is permanent, it will occupy unnecessary resources even though it may not be needed. If the replica is temporary, a peer or a set of peers has to make sure that the obsolete replica is removed from the system. *Consistency.* The consistency maintenance is perhaps the most critical issue in replica management where updates are possible. The consistency may be performed as soon as an update occurs or may be delayed to save from frequent updates. The consistency is usually maintained by coordinator peers.

### 4.3 INDEXING MODULE

The indexing module is closely linked with the logical and storage modules. The logical module determines the granularity of data to be indexed. The storage module decides where the data need to be stored. For example, the storage module may store the closely related data into specific clusters.

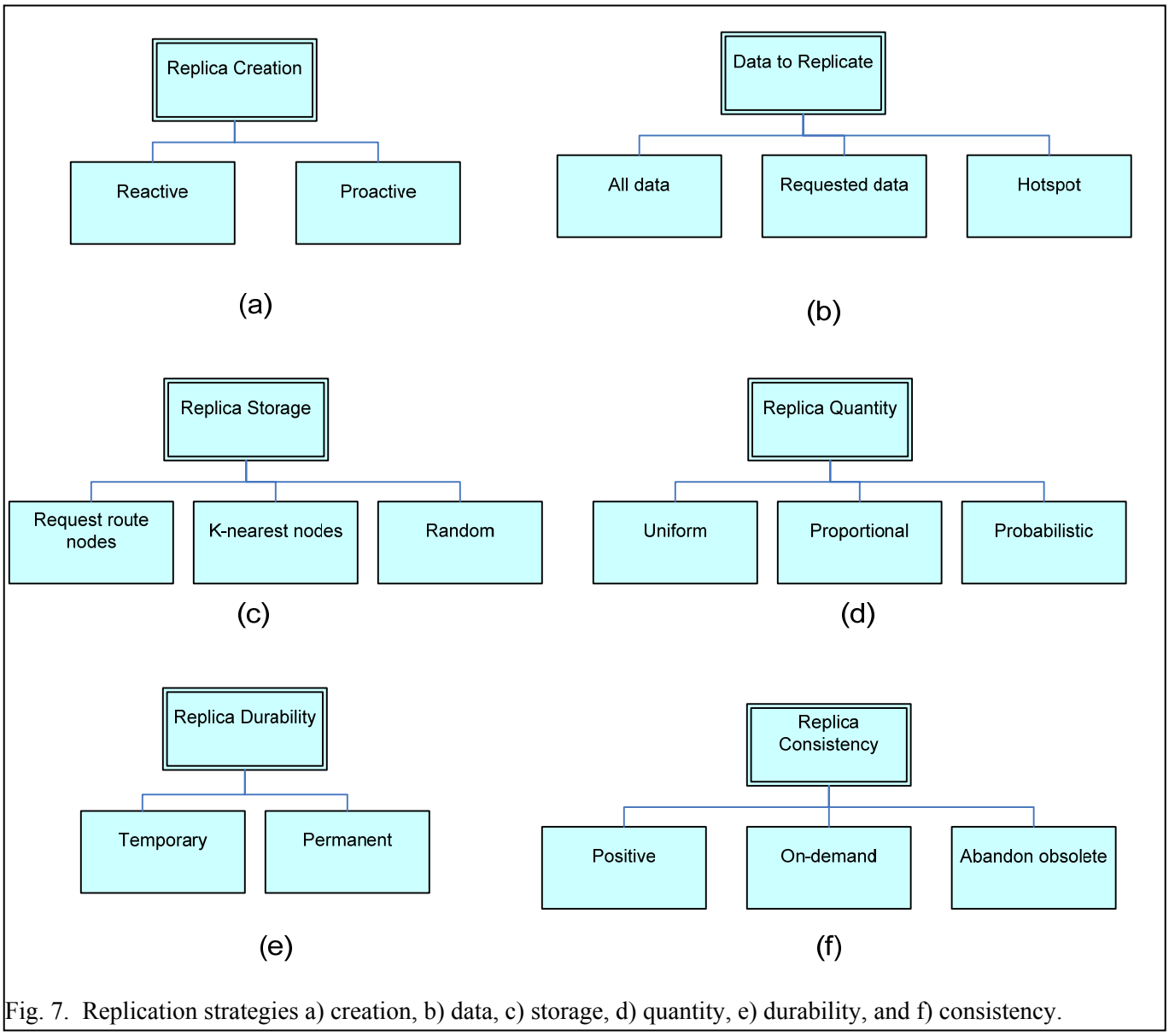


Fig. 7. Replication strategies a) creation, b) data, c) storage, d) quantity, e) durability, and f) consistency.

*4.3.1 Index Types*

Management of index structures is the major component of the indexing module. Index is an effective mechanism widely used in facilitating information retrieval and resource discovery in P2P systems. Unlike indices in databases, any shared resources in P2P systems could be indexed to facilitate resource discovery. Most P2P systems provide certain kind of index services in order to support acceptable discovery performance and improve system scalability. Different types of P2P index is used in P2P systems (Fig. 8). The main purpose of index is to accelerate searching or querying and thus to improve system scalability. We can classify P2P index into four groups: routing index, file index, content index, and resource index.

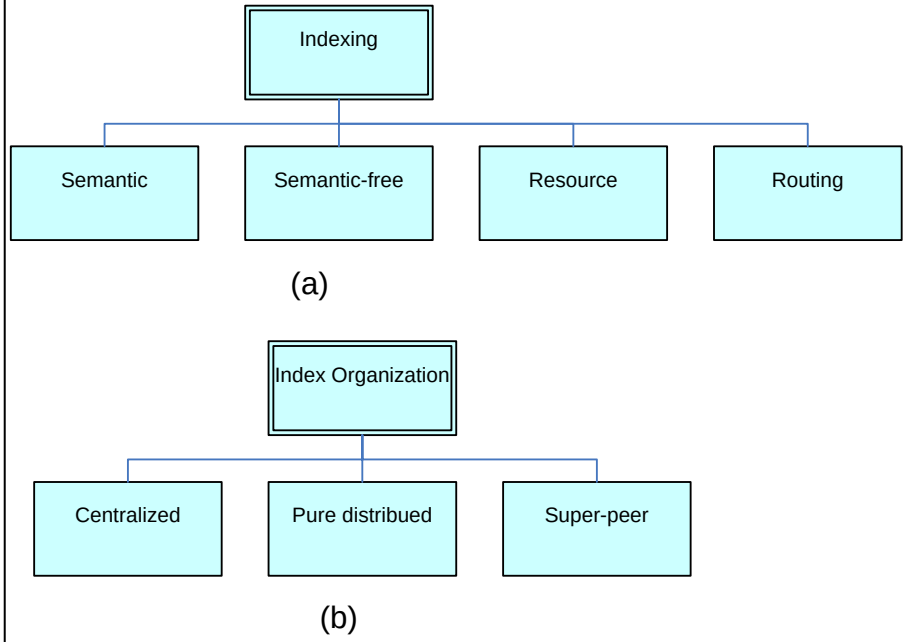


Fig. 8. Indexing strategies in P2P systems a) data and b) organization.

***Routing Index.*** A clear trend of indexing in P2P system is to combine both the contents and structure of index with overlay topology (Bawa, 2003). The first two factors in designing an index are its contents and storage structure (location and structure). The purpose of index is to accelerate peer location in overlay system in supporting message passing. Typically, an entry of index maintains the name or ID of a neighbor peer as well as either the direct address or the next hop in the path to it. The actual function of these indices is just like the routing tables in IP systems. That is why they are also called routing index. An application of routing index where indexing of elements is performed both at the instance and schema level has been studied in (Karnstedt, 2004).

***Semantic vs. Semantic-free Index.*** Semantic index is a human readable index (Risson, 2006). On the other hand, semantic free index is usually composed of computer generated key values and not easily readable by humans. In this sense, DHTs based on file name and content provide semantic-free index. In P2P file sharing systems that provide gross data granularity, the shared atomic item is a file. In these systems, only file name, sometimes file ID and even with file content descriptions, is maintained in their indices. For example, in Napster, a central index maintains names of music files and links to peers that host the files. In (Risson, 2006), semantic index supports keyword lookup and database indexing. Only keyword matching queries can benefit from file index. Content index allows access to the data contents of files.

The main function, as well as purpose, of content index is to facilitate complex queries in P2P systems. To enable fine granularity, the content index is inevitable. It is hard to put a strict border between P2P systems whether they use a semantic or semantic-free index. For example, AmbientDB uses DHT on the primary key of tables to enable semantic index. Another approach is to build another layer on hash-based systems to provide semantic index.

**Resource index.** In a P2P system, peers share not only data but also resources like disk spaces, system bandwidth, and CPU time. However, just as in data sharing, finding desired resources in P2P system is costly if there is no incentive information to direct such resource search. Some P2P systems that concern on sharing resources other than data can provide certain index services to record resource distribution in the system. We call this kind of index as resource index.

*4.3.2 Organization*

A good survey of indexing strategies is provided in (Risson, 2006). The index depending on where it is stored is classified into three: local, centralized, and distributed. The storage structure or location of index in P2P system greatly affects its performance in accelerating searching and querying. Based on this, we provide three types of

index: centralized index, pure distributed index, and super-peer index. In the centralized index organization, when a peer joins the network, it sends the information of data to be shared with other peers to a central server. All the searches are performed trough the central server where the central index is maintained. Napster uses central index. This type of P2P systems has a single point of failure. In pure distributed index, the index structure is distributed among nodes. The most widespread distributed index is Distributed Hash Index (DHT). In this type of index, each peer behaves as if it is a bucket of a big hash table. Each peer is assigned an identifier and keeps data close to its identifier. This type of P2P systems has to minimize the data transfer when a peer leaves and joins as well as when the data are updated. In super-peer index, the index is maintained at super-peers. A set of peers is organized around a super-peer. The super-peer communicates with other super-peers for data location.

***Discussion.*** The centralized index is actually against the notion of P2P system that avoids dependence on a single system. The centralized index also means single-point of failure. On the other hand, the index management is easier than distributed index management. The distributed index avoids single point of failure; however, it needs to minimize the data transfer for index management. To get the best performance out of indexing, the base P2P service module should provide the necessary organization structures for indexing.

## 4.4 LOGICAL MODULE

The logical module is one of the most critical components in a P2P system. The logical module determines the granularity of data to be indexed at the indexing module. The logical module identifies the smallest size of data that can be queried from the perspective of the querying service.

*Challenges for Data Modeling.* Bonifati et al (Bonifati, 2008) provide a rough classification of the P2P databases whether the overlay network is structured, unstructured, or hybrid. Such classification may not be satisfactory to understand the challenges for data modeling. In the history of computerized data processing, the schema-based data management provides more flexibility on accessing the data contents. It uses well-defined models not only to manage the data model but also to express internal semantics of the data. Moreover, proper query processing methods are provided to ensure the usability of these models. On the other hand, the lack of schema and semantics in early P2P systems stimulates utilizing database aspects in P2P data management. However, neither federal database nor distributed database techniques can be directly borrowed in P2P paradigm due to the following two basic assumptions made in P2P systems. Firstly, the global and consistent information for data content and distribution over the whole system is not available. The second is that peers have autonomy to join and leave at any time on their will, and no limitation of standard data model can be forced over them either.

These assumptions raise challenges for applying schema-based data management techniques in P2P systems. There are mainly two approaches to integrate heterogeneous data. First, we can assume a simple and common model to represent data used by members. For example, Edutella assumes users use XMLBase (XML Base) to store their data. On the other hand, we can use intermediate mechanisms to adapt differences. For example, in PeerDB, each peer has a local wrapper to extract global schema from local data and use agent-based mediator to integrate data from other peers. In (Ouksel, 2003), the context is referred as inter-schema mapping from the schema of local database to that of remote database.

### *4.4.1 Data Granularity*

The simple but fundamental factor in P2P system design is what kind of information sharing is desired in a P2P system. There are mainly two types of P2P data sharing: entire file sharing and content-based (schema-based) sharing. Accordingly, in current P2P systems two levels of granularities have been provided in data sharing: gross (non-value-based) granularity and fine (value-based) granularity (Fig. 9).

***Gross Granularity.*** Early P2P data management provides only gross granularity data access. At this level, data object is the whole file and can only be accessed through file name (key) or keyword (such as a hashed ID) as a non-divisible entity. In this approach, only full name matching query is supported. This approach does not provide complex information retrieval like range queries over detailed file contents. P2P systems like Napster, Gnutella, CAN, Chord, and BitTorrent (BitTorrent) fall into this category. Such coarse data management cannot alone fully satisfy the requirements of P2P information sharing.

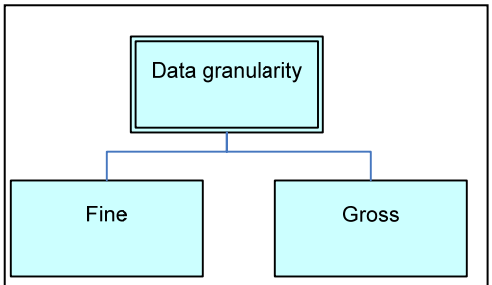

Fig. 9. Data granularity in P2P systems.

***Fine Granularity.*** Some P2P systems try to provide fine granularity in data placement and manipulations where detailed contents of files are accessible. Since data items of a file should be considered individually, a serious problem caused by providing fine granularity data is the management of extremely huge data. This requires semantic model to ensure acceptable efficiency otherwise the tremendous amount of data items are intractable for keyword-based queries. However, fine granularity has become the mainstream nowadays in P2P data management due to its power in information sharing. Since fine granularity P2P systems model the contents of data (file) using a (strict or non-strict) schema, they are assumed to provide semantics by providing the relationships among data. In this sense, fine-granularity systems can also be called as semantic based P2P systems. DBGlobe (Pitoura, 2003), PeerDB, Edutella, and Piazza (Halevy, 2004) are some good examples in this area.

**Discussion.** The data granularity is the elementary factor that determines the level of information sharing and hence the efficiency and satisfactions of information retrieval. Structured (DHT based) P2P systems typically could provide only gross granularity. To enable independence between storage, indexing, and logical modules, it is possible to build an index at fine granularity on top of an architecture that provides only gross granularity.

### *4.4.2 Data Modeling*

Based on data granularity, we can categorize P2P data management systems into two categories: schema-based P2P data management (for fine granularity systems) and P2P file system (for gross granularity systems). The main challenge in fine granularity P2P sharing is to control huge volumes of data items and share them with others. Not surprisingly, researchers choose schema to model data and represent content semantics. Most research efforts aggregate on mainly two directions: relational model and XML based approaches. It is not appropriate to say which one is better, because in different P2P systems, different schemas may be preferred.

***Relational Model.*** The relational model is useful in those contexts that data need to be stored and accessed with respect to a strict schema. Such context often exists in private P2P systems. Some P2P systems, such as PeerDB (Ng, 2003) on top of BestPeer (Ooi, 2003) uses relational schema to model data objects. For example, in (Serafini, 2001) a local relational model is proposed in order to share medical data of hospital and family doctors whose computers are connected through a private P2P system. In such context, data accuracy and consistency is required for reliable medical decisions.

***Semi-structured & Unstructured Data Models.*** Most data existing in *open* P2P systems that accept a peer without censoring are not supposed to be suitable for a relational model since they are semi-structured or unstructured. In other words, we do not need a strict schema like a relational model to describe data in such contexts. Instead, the more flexible and rather less restrict XML based schema is widely used in open P2P systems. For example, semantic web applications and Piazza choose Resource Description Framework (RDF) (RDF) to manage data schema in P2P systems. RDF assigns uniform semantic meaning to certain reserved objects and properties. It also explicitly names the relationships of objects. RDF is more suitable for modeling these unstructured data. In (Kokkinidis, 2004), a semantic overlay system is created for peers sharing the same schema information and the queries that are generated on this system are considered as semantic queries. They provide an RDF-based schema named as RDF/S.

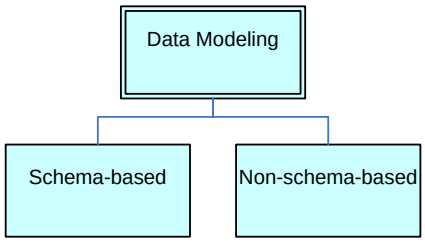

Fig. 10. Data modeling.

***Advancements.*** Although in systems providing only gross granularity there is no need to model shared data, since data are processed as an indivisible entity accessed only by file name or hashed ID, many recent research make these P2P systems provide rich and complex queries. Recent research introduces partial key matching (Zhang, 2003) and range query possible in some of these systems (Sahin, 2004). However, files remain the indivisible element to be shared. Another interesting approach is introduced in AmbientDB (Boncz, 2003). It uses DHT as global index to route query and to setup a relational global schema over Chord to model data. However, it assumes the presence of

global schema, which is not reasonable for most P2P environments. Arenas *et al.* (Arenas, 2003) considers a P2P database without any global schema where P2P acts as an interoperability layer.

***Discussion.*** It is proper to name P2P systems that share and model detailed data (file) contents as schema-based and P2P file system as non-schema-based (Fig. 10). Here, although the term schema-based arose from semantic web, we consider applications using relational model (as in PeerDB), and XML schema (as in DBGlobe) as also schema-based since they also use certain schema to model data. Note that, in semantic web, schema-based also implies to combine the topology management with schema mediation in P2P systems, while in our paper, such combination is not required. It is hard to enforce a strict data model for all peers and all kinds of data in P2P systems. In cases where strict model cannot be achieved, wrappers to map from a model to another model can be achieved to provide more usability. Strict models are usually preferred in private P2P systems.

#### *4.4.3 Semantics and Data Integration*

A proper way to model data is not enough for efficient P2P information sharing. When there are overlaps and interrelations of data in P2P systems, semantic services like schema mapping are needed for exploring information since not all peers are expected to use the same schema to model data. The purpose is to represent and record the above overlaps and interrelations of data and to facilitate information query. The basic approach of semantic service is to create certain mapping among peers and their data (Kementsietsidis, 2003).

Two obvious and common approaches that can be used are creating mapping of data models—schema mediation (Nejdl, 2003) or mapping data (Kementsietsidis, 2003) directly. Practically, schema mediation is preferred and discussed in existing P2P systems. Based on the range/number of peers involved, we can classify current systems into two groups: individual mediation and aggregated mediation (Fig. 11).

***Individual Mediation.*** In individual mediation, each pair of peers, when necessary, creates a private mapping that is only used for their own information exchange. For example, in (Serafini, 2001), each "acquaintance" in a pair creates a schema mapping of their own Local Relational Models, based on the mutual information exchange to be followed.

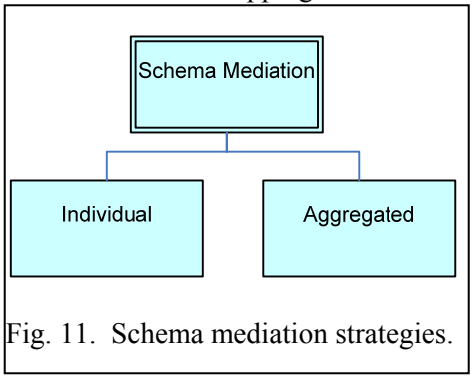

Fig. 11. Schema mediation strategies.

***Aggregated Mediation.*** In aggregated mediation, certain semantics community is created and local schema is mediated to a common schema. Typically, such communities are organized around certain ontology concepts to accelerate the performance of data integration so that peers that are concerned on the same information set will get aggregated. For example, in DBGlobe, peers are aggregated in building up semantic communities around semantic concepts, controlled by CoASs (Community Administration Servers). Schema mediation is performed on CoASs, which keep track of involved peers and extract metadata from them. The connections of CoASs are arranged in certain topology to improve semantic aggregation and mediation. A more interesting application is implemented in a Semantic Web application (Edutella (Nejdle, 2002)), which sets up a super-peer backbone to handle both schema meditation and query processing. Its super-peer back-bone is arranged as HyperCuP so that the system topology is combined with semantic mediation in a more scalable approach.

***Discussion.*** The advantage of individual mediation is that it could create detailed, accurate, and specific mapping that can best facilitate the information sharing and query processing. However, from the system view, there are too many mappings created and many of them overlap. This increases the complexity and reduces the performance of queries involving more than two peers. Aggregated mediation organizes the peers into communities that have similar interest. This increases the performance of the queries.

### 4.5 QUERYING SERVICE

Querying and information retrieval service is the most important functionality of this service. The efficiency of the querying service can be significantly increased with an efficient distribution of the data by the storage module. The storage module may replicate the data over the peers in a way that the query results can be retrieved to peers quickly. The logical module determines the queries whether the data can be queried as a whole entity (semantic-free) or content-based (semantic). The logical module also determines the type of querying languages that can be used for the retrieval. The application and services modules have to provide the user interface to the user to submit queries

and visualize results.

Querying and retrieval in P2P systems is quite different from those in traditional distributed databases. Three features that affect the query and retrieval in P2P systems are decentralization, dynamic membership, and heterogeneity, which determine topology, query routing, and schema/semantics mediation (mapping) mechanisms, respectively. Unlike in traditional distributed information sharing that is based on Client/Server model, the overlay system topology plays a critical role in P2P query processing and retrieval.

#### *4.5.1 The Effect of System Topology*

In traditional distributed databases, the system topology is not that critical since global information and static location of nodes help distributed databases optimize query processing. However, in P2P world, global information is not available. This makes system topology critical for both on how to distribute data and propagate query correctly and efficiently. When some kind of structure is setup over system topology, it benefits the efficiency of query processing in P2P systems.

There are two main approaches in arranging system topology for data placement and retrieval. The first is DHT based approach. These P2P systems use DHT as the global index table to locate data and direct query routing. HIERAS (Xu, 2003) provides a hierarchical DHT-based routing algorithm by creating P2P rings in a way that the routing tasks are handled at lower layers before they go up to higher levels. However, due to the nature of DHT, DHT is incapable to handle fine granularity data directly. The other approach is to cluster peers into communities. For example, in DBGlobe, semantically related data are aggregated in the same community and corresponding query is processed in the community. In Edutella (Nejdle, 2002), super-peer based system topology is another form of clustering. In Edutella, schema and content indices are clustered in super-peer. Queries are first handled by super-peer backbone. After finding an appropriate super-peer, queries are broadcast to all peers connected with the Super-peer.

***Discussion.*** At this point, there is no clear separation of querying functionalities that need to be handled at the base P2P service module and querying module. The traditional (keyword-based or file-based) retrieval can be handled properly at the base P2P service module. We believe that the organization of peers (e.g., the creation of communities) should be pushed to base P2P service module. Moreover, the dependence between querying service and base P2P service module should be minimized. The P2P application should be allowed to choose any type of topology for the organization of peers to improve querying performance.

#### *4.5.2 Query Processing*

Efficient query processing is the most important function provided by P2P data management systems to support information sharing. In P2P paradigm, query processing is related not only to the data model but also to the system topology.

Three problems are crucial to P2P query processing: query representation, query propagation, and query result mediation. The first one is query representation. An expressive query language needs to address the data request. It is determined by the chosen schema for modeling the data. The second one is query propagation. We need an efficient method to send our query to those resources contain requested data. It is addressed by the combination of schema or semantic concept with topology arrangement. The third one is query result mediation. Query result should be tailored to local data model and ontology. To address this problem, some translation methods or languages are required to guarantee both the query and resulting data could be understood by all peers. Query optimization is an area that waits for contribution of researchers (Hellerstein, 2003). Query caching is used to improve the querying performance (Patro, 2003), (Sahin, 2004).

*Query Representation.* In P2P systems that provide gross granularity (P2P file sharing systems) data sharing, they only need to include file name or object ID in data requests. In these systems, it is unnecessary to provide a special query language for expressing data request. On the contrary, in fine granularity P2P systems (semantic based systems), an appropriate query language is critical for correct and efficient information retrieval. Typically, the query language is determined directly by the schema of chosen data model. For example, those systems that use relational model usually use SQL as query language. The systems that use XML based schema could use XML query languages such as XPath and XQuery (Ou, 2003). RQL is also used in various semantic P2P systems including (Kokkinidis, 2004).

*Query Propagation.* In the case of query propagation, problems arise from the lack of global information and

dynamic topology. In topology-independent systems, there are only two approaches available for query routing. The first one is to use a central server to target query globally. The other way is to use flooding or broadcasting with Time-to-Live (TTL). In topology-aware systems, the dynamic of topology is well handled through creating certain kind of community. For example, CAS in DBGlobe is such a community. Certain data request related to a semantic concept is processed in the community with the same semantics. Typically, a certain query will be decomposed into sub-queries and sent to corresponding community. The query routing service should be provided by the indexing and base P2P service modules. The querying service should rather be concerned with decomposition and aggregation of query results.

*Query Result Mediation.* Wrapper and mediation approach is widely used in existing systems. For example, in Edutella and (Ding, 2004), wrapper is used to extract commonly agreed schema from local data repository. Mapping service is provided to interpret XML-based schema. A language family, RDF-QEL, is provided to standardize query exchange mechanism.

Although non-schema-based P2P systems provide user only gross granularity, they could use some kind of partition in either data placement or query processing to provide semantics and range queries. There are mainly two approaches. First, the keywords can be partitioned. Therefore data request is processed as partial key matching instead of exact keyword matching. Systems like PinS (Villamil, 2004) provide upper layers on top of traditional P2P systems (like DHT-based) and provides indexing of higher level data. Then the multicast features to support range queries and similar queries can be utilized. Second, in some systems, such as BYPASS (Kwon, 2004) and the one in (Sahin, 2004), peers are clustered into groups around some semantics concepts. These groups are called community or zone. Then data relative to the same semantics concepts are stored in corresponding groups. In other words, data are placed together around same semantics concept. When complex queries about some semantics concept occur, they are dispatched to corresponding groups.

***Discussion.*** The query representation is directly related with the schema of the data model. For example, if relational model is chosen, the queries are represented with SQL. The query propagation can be considered at the low and high levels. At the low level, the query is propagated with respect to the P2P topology. At the high level, the query is propagated with respect to previously defined semantic communities. The low level query propagation is handled by base P2P service module. We believe that high level query propagation should be handled by indexing and base P2P service modules. For query result mediation, the wrapper approach is more flexible than mediation. The mediation requires schema exchange within a community of peers or between a pair of peers. The query representation and result mediation depends on how the data is modeled at the logical module. At this point, the querying service heavily depends on the logical module. In the future, the dependence on the logical module should be minimized, and conversion from one query representation to another should be supported wherever possible.

## 4.6 STREAMING SERVICE

The most typical example of streaming service that needs to be satisfied is the maintenance of Quality of Service (QoS) as in multimedia streaming applications. In this section, we focus on QoS for multimedia applications since it requires strict QoS requirements. QoS is one of the important services that needs to be satisfied if the P2P system provides media streaming. The interest in multicasting using P2P systems has increased in recent years (Zhuang, 2001), (Helder, 2002), (Ratnasamy, 2001b), (Jannotti, 2000), (Kostic, 2003), (Castro, 2003), (Eugster, 2001), (Castro, 2002), (Li, 2003), (Zhang, 2005). The maintenance of QoS under fluctuating system conditions is challenging. In (Mielke, 2002), a client/server system where the client informs its status during transmission of packets is studied. Even with continuous feedback messages with a powerful server to maintain QoS is challenging. Currently, most P2P multimedia applications focus on P2P multimedia streaming and QoS (Bulterman, 2003), (Xiang, 2004), (Kalapriya, 2004) , (Habib, 2004), (Jiang, 2003), (Kalogeraki, 2004). There are also some other applications focusing on multimedia information retrieval, such as AmbientDB (Boncz, 2003) and applications as in (Maxim, 2003) and (Yang, 2003). However, we believe that P2P model could bring multimedia more novel applications in content based information retrieval, hypermedia, multimedia databases, and other directions.

### *4.6.1 Issues in P2P Multimedia*

Many characteristics of P2P systems are attractive for multimedia applications. The first one is the huge storage space that is much larger than traditional databases could provide. The second one is the computing power provided by peers in P2P systems since applications like content-based retrieval is computational intensive. The third one is bandwidth that has not been considered in other P2P systems but is mostly relative to multimedia streaming. Fault

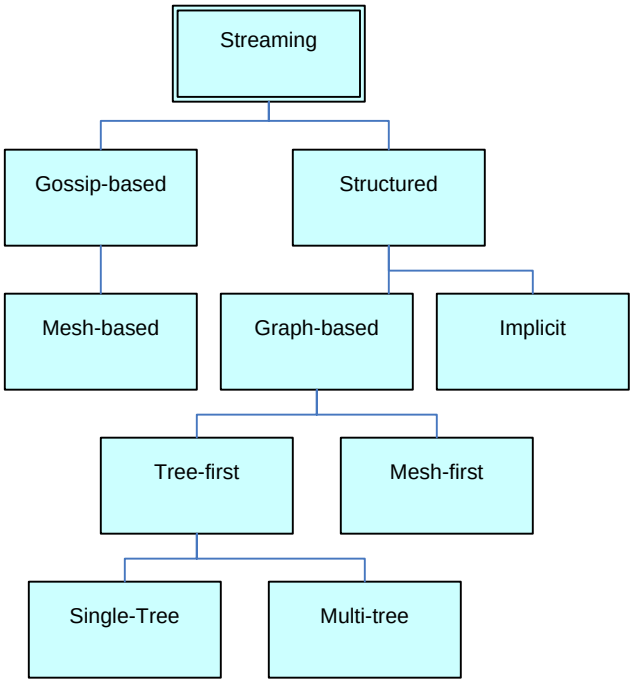


Fig. 12. Streaming strategies.

tolerance, as in serverless communication and naming mechanism out of DNS, is another desired attractive feature for P2P multimedia. Abad et al. (Abad, 2004) and Liu et al. (Liu, 2006) provide surveys of broadcasting streams on top of P2P networks.

***Problems.*** Problems arise from three characteristics of P2P systems. The first one is the lack of global information in P2P systems. Global information needed by P2P multimedia applications is quite different from above discussion, which is more concerned on global data schema. Here, computing power, storage space, and out-bound/in-bound bandwidth of each peer as well as their distribution are both critical for P2P multimedia applications. Secondly, P2P multimedia applications are also prone to the dynamic membership and autonomy of peers. The reason is that, unlike traditional data processing, P2P multimedia applications, like streaming and content-based retrieval, are cooperative, real-time, and computational sensitive. The dynamic and unpredictable feature of P2P system may fluctuate the applications' performance. The third element as addressed in (Hefeeda, 2003) and (Tran, 2004) is the system topology. Unlike the topology problems discussed in above discussion, here we mean the physical topology of P2P system. This is mostly relative to multimedia streaming. The bandwidth distribution, overlap and current status of certain communication path should all be considered in P2P multimedia streaming since these features are critical to streaming performance.

### *4.6.2 P2P Multimedia Streaming*

The most popular P2P multimedia application is P2P multimedia streaming. Since multimedia streaming is real-time, it is more prone to system status (including bandwidth, availability of data source, overlap of connection path, etc.). The first problem to handle is how to arrange system topology.

The approaches for multimedia streaming can be classified into two as gossip-based and structured streaming techniques. The gossip-based streaming is similar to flooding and whenever a peer receives blocks of a stream, it forwards the blocks randomly to its neighbors. CoolStream (Zhang, 2005) and LPBCast (Eugster, 2001) employ gossip-based multicasting. The mesh-based streaming (Liu, 2008) can also be considered as gossip-based streaming. On the other-hand, in structured streaming, there is a source that distributes the data. The other peers form a tree to support reliable and fast broadcasting. Structured streaming guarantees delivery of blocks with log(N) connections from the source where $N$ is the number of peers in the tree. Agarwal proves that the same efficiency can also be achieved using gossip-based streaming (Agarwal, 2006). Structured streaming can be decomposed as graph-based and implicit. Graph-based systems can further be classified into two as in (Abad, 2004): mesh-first and tree-first. In (Liu, 2008), the tree-based approaches are classified as single-tree and multi-tree approaches. Multi-tree streaming tries to minimize the effect of peer churn by maintaining and streaming through multiple trees. In tree-first strategy, the peers create the tree and the tree is optimized in the later stages. YOID (Francis, 2006), Banana Tree Protocol (Helder, 2002), Overcast (Jannotti, 2000) and Hostcast (Li, 2003) are examples of tree-first strategies. In mesh-first strategy, the tree (like a spanning tree) is extracted of a mesh graph to provide the most efficient streaming. End System Multicast (Chu, 2003), ALMI (Pendarakis, 2001), and Prime (Magharei, 2007) are examples of mesh-first strategy. The implicit strategy uses the underlying P2P network topology (indexing) like CAN, Tapestry, and Pastry for streaming. CAN-multicast (Ratnasamy, 2001b) is built on top of CAN and uses intelligent flooding on top of Cartesian-space. Bayeux (Zhuang, 2001) is built on top of Tapestry; and Scribe (Castro, 2002) is built on top of Pastry. Figure 12 displays various methods of streaming.

The streaming services are classified as live streaming and video-on-demand (VoD) (Liu, 2008). The tree and mesh streaming techniques can be applied to both live streaming and VoD. However, the peers are synchronized for

live streaming whereas the peers may request different portions of a stream in VoD.

*Live Streaming.* For live streaming, the approaches can be studied under tree-based streaming or mesh-based streaming (Liu, 2008). The tree-based streaming can be classified into single-tree or multiple-tree streaming. In a single-tree streaming, if peers are first to join the live streaming, they directly connect to the server if there are no other peers that have the streaming data. Otherwise, the peers connect to another peer and get the stream from that peer. In a single-tree streaming, the depth of the tree is a critical factor as it may cause delay for peers that appear at the leaf nodes. The fan-out of internal nodes could be increased to reduce the depth of the tree. However, the fan-out of a node is limited by the uploading bandwidth of a peer. In single-tree streaming, some peers (the majority) only act as receivers and they do not contribute to streaming to other peers. Instead, in multi-tree streaming, subtrees serve different substreams of the original stream. While a peer may appear as a leaf node in one subtree, it may appear as an internal node in the other subtree. Managing the significant number of leaves of peers is a major problem for smooth presentations. In mesh-based streaming, there is no parent-child relationship and the peers may use tracker as in Bittorent. The peers may get a list of active peers from a tracker or exchange the peer lists with its neighbors. In mesh-based streaming, there are two strategies for streaming data: push and pull. In push strategy, a peer uploads a chunk of video to its neighbors. However, it is likely that the downloading peer might already have that chunk or it is even possible that two peers may try to upload the same chunk to the same peer. This problem can be minimized by using the pull strategy. In the pull strategy, peers exchange their buffer maps with other peers. Therefore, a peer may only request missing chunks from peers that have the data by avoiding overloading the network.

*VoD.* The methods for VoD are based on tree-based and mesh-based streaming (Liu, 2008). In tree-based VoD, peers that are close in time form a session. This leads to two types of streaming: base and patch serving. In base stream, peers get the data synchronously. In patch serving, the peers maintain the initial part of the video for latecoming peers. If a peer is late, it connects to the peers that have the closest in time and gets the data from these peers filling the patches. A better approach compared to tree-based streaming is the Cache & Relay VoD. In this version, a peer caches a sliding window of video content with respect to the current video segment to be played. Whenever a peer joins for downloading, it checks whether another peer has their data in its buffer. If that is the case, the joining peer starts getting the data from those peers. A peer is allowed to leave after it completes streaming of its window. Another alternate option is the use of mesh-based VoD. In this case, the system performance can be improved if peers have diverse set of chunks of the same video. In BiTos (Vlavianos, 2006), the buffer is divided into 3 parts: received, high priority, and low priority. Received parts maintain the chunks that are received. The high priority part has missing chunks that needs to be retrieved urgently for display in time. The low priority part maintains missing chunks that will be needed in the future. The chunks in high priority have a higher probability of $p(p \geq 0.5)$ of retrieval than the probability of chunks that are missing in the low priority area $(1-p)$.

In Zigzag (Tran, 2004), a dynamic tree like structure is utilized to arrange system topology. The root of either the tree or a sub tree is considered as a server that could provide out -stream data. At first, a peer downloads a media stream from a unique source. Then these peers become server for some new-joined peers. Thus the overhead of streaming is distributed and system performance is improved. When some server peer leaves, the tree is restructured to gain an optimal performance. However, simultaneous peer connections are omitted by the above application. Multiple peers may make concurrent application to the server. These concurrent peers may be different in out-bound bandwidth. Thus the order of admission to peers affects the overall performance.

In (Xu, 2002), the author addresses this problem by peer selection through an optimal knapsack algorithm called $OTS_{p2p}$. The author further resolves the segment overlapping of communication path in PROMISE (Hefeeda, 2003). In this project, they use topology-aware selection to avoid bandwidth bottleneck caused by path segments overlapping. However, both projects need too much global information about both peers and system status.

**Discussion.** The multimedia streaming is affected directly by system topology. The major issue for P2P streaming is the topology to be employed for streaming. As of now, structured and gossip-based streaming strategies compete with each other. The advantage of gossip-based streaming is the scalability. In (Agarwal, 2006), it is proved that gossip-based protocols can also achieve the efficiency of structured streaming. CoolStream proves that P2P streaming on large scale is possible. In the future, it is possible to have hybrid systems that benefit gossips as well as structure of the topology.

## 4.7 SECURITY SERVICE

There are various interpretations of security and in the most general case it has some close relationships with the storage module. The security can be incorporated at different levels and may have various relationships with other modules. The application and support modules can encrypt the data before inserting into the system. The security can also be maintained at the base P2P service module by encrypting through a chain of peers before publishing in the P2P system. The security can be considered in terms of data and peer integrity (Fig. 13). Although peer integrity is not directly related with data integrity, the surveys (Andtroutellis-Theotokis, 2004) including security explain them under the same title. Even if a peer corrupts or does not transmit some portions (blocks) of data; the data should be able to be regenerated from the rest of the data blocks. In this section, we also briefly explain how reputation-based trust management can be used to avoid security attacks.

Windows P2P achieves security in five ways (Windows P2P): peer names, group membership certificates, roles, secure publishing, security policies, and secure connections. Secured peer name is generated by the owner and maintained using public key cryptography. A secured peer name belongs to the peer having the corresponding private key. Secured peer names are *statistically* unique. Group security uses peer names to identify the members of a group. Each group also acts like a peer and has a public/private key pair. A peer is called as group owner, if its private key is used for group's private key. Windows P2P system uses group membership certificate to check the membership of the peers. The peers in the group may have roles like administrator to issue group membership certificates to peers having intent to join the group. Data integrity is achieved by checking the security data derived from the contents of a record. This security data is part of the record. Not all members are authorized to publish records or update the records that are published by others. The security policies include issues about peers like whether they are able to issue administrator right to other peers. The secure connection between peers is established using group security service providers (SSPs). For groups, P2P group connect protocol SSP is used.

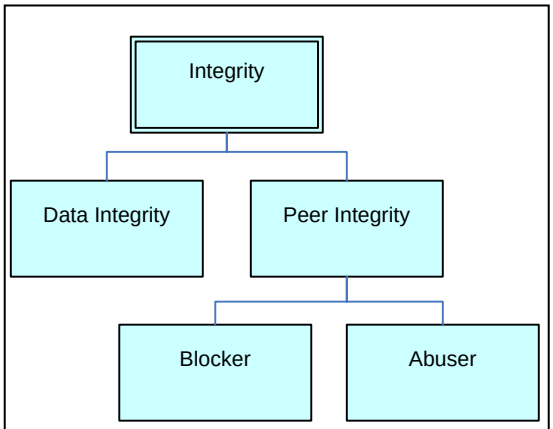

Fig. 13. The security of the resources.

*Data Integrity.* In serverless peer environments, a peer must handle authentication by itself (Windows P2P). Self-certifying (SC) data (Wallach, 2002) is one way of determining the integrity of the data. In this method, each node assigns a key to the file while inserting to the system. The retrieving node is able to check the integrity of the data by using the hash function used in generating the key.

Information dispersal (ID) strategy (Andtroutellis-Theotokis, 2004) partitions the (encoded) data in a way that the data can be retrieved from a subset of partitions even some partitions are lost or faulty. This approach is closely related to indexing and storage modules. The indexing module may limit the peers where data can be stored. The storage module is responsible for the reliability of the data. Erasure coding (EC) (Andtroutellis-Theotokis, 2004) and forming of rings through agreements such as Byzantine (Castro, 1999) agreement protocol are examples of information dispersal security strategy. For establishment of agreements, base P2P service module must support such protocols.

*Peer Integrity.* The peer integrity should be investigated under two classes: blocker and abuser. The blocker peer is a malicious peer that tries to destroy the functionality of the system by not forwarding messages or by blocking answers from other peers. The abuser peer is a malicious peer that tries to abuse the resources of the system. In Windows P2P security, each peer generates a public/private key pair where private key is used to generate the self-signed certificate used for authentication and to provide information about the peer (Windows P2P). The Sybil attack (Douceur, 2002) in which a peer can join the system with various IDs is a consequence of a blocker peer. The blocker peer usually works on the base P2P service module and precautions can be taken on this module. Another approach that is related with the storage module is to replicate the data in order to avoid the destruction of blocker peers.

*Reputation-based Trust Management.* Reputation-based trust management is an important way of dealing security in P2P networks (Hamlen, 2007). This can be achieved by assigning a global trust label $t_p$ to each peer *p*. The opinions of interacting peers with peer *p* contribute to the trust of peer *p*. In this fashion, the reputation of a peer *p*

may increase or decrease. The goal is to set this reputation system in a way that malicious peers get a low reputation whereas non-malicious peers receive high reputation. The reputation of the peers may be maintained as centralized as well as distributed. In a distributed trust management, a set of *k* peers maintain the reputation of a peer. After an interaction with a peer *p*, the peers report their feedback to all *k* peers. Changing the reputation of a malicious peer is difficult since at least k/2 of these peers need to be compromised.

In (Tsybulnik, 2006), the objects also get a global integrity label, $t_o$, for an object *o*. After a peer downloads object *o* from peer *p*, it communicates with the peers that maintain the reputation of object *o* and peer *p*. This trust-based system will generate honeypots that have high reputations and whose opinions are respected by other peers in the network. This avoids the transmission of malware intentionally by malicious peers and unintentionally by non-malicious peers.

If malicious peers are known in the system based on their trust tables, secure routing can also be established by avoiding these malicious peers during routing (Hamlen, 2007). The malicious peers may drop messages, forward to another destination, lie about its location in the networks, and appear as multiple peers in the network (sybil attack). The secure routing can be achieved by having multiple non-deterministic routes from the source to the destination so that the message will be eventually delivered by avoiding the malicious peer in one of the routes. The peers that misroute the messages should be penalized by reducing their reputation. This can be achieved by using self-certifying identifiers and constrained routing tables. A receiver only forwards a message if the receiver peer appears in the routing table. If a message is forwarded to a destination that is not part of the route, it can be penalized by reducing its reputation.

The Sybil Attacks can be minimized by using identity-based cryptography (Ryu, 2007). Three protocols are developed: trusted third party, trusted bootstrap, and trusted assignor. In trusted third party protocol, a node first contacts a trusted third party with its IP address, and gets a random ID and its key from the trusted third party. Then it contacts a bootstrap node with timestamp and its ID both signed with its key. The bootstrap node returns a signed copy of the ID and the timestamp. This method prevents Sybil attack since the joining node cannot get the message from the bootstrap node if it provides incorrect IP address. Moreover, another peer may check the authenticity of a message by contacting the bootstrap node. Trusted bootstrap mode is a centralized version of the previous protocol since the bootstrap node acts as the arbiter of the network and the trusted information. Trusted assignor protocol is a hybrid of the first two where a bootstrap node selects a set of nodes that can be trusted. It then transfers the role of assigning node IDs to one of these trusted assignor nodes. The Sybil attacks can also be tracked as in (Danezis, 2005). A malicious peer needs to convince a non-malicious peer to join the network and the rest of the malicious peers can join without interacting the non-malicious peer. The collection of low-reputation peers that are malicious can be determined and the source of malicious peers can be banned from the network.

***Discussion.*** Depending on the security to be satisfied, the security can be achieved at various modules at different levels. Data integrity is usually achieved with the help of application, service, and storage modules. Peer integrity is usually achieved at base P2P service module. Reputation-based trust management can avoid various attacks that include peer integrity as well as data integrity.

## 5 DISCUSSION

A summary of the modules & services and strategies is provided in Table I. Our conceptual model provides a good picture of possible interactions among modules and services. Base P2P service module covers naming of entities in the P2P system and is responsible for peers' joining/leaving and naming. Storage module is directly related to the Base P2P service module since the type of the P2P network may affect where data need to be stored. Storage module considers the data naming, availability, replica creation strategy, when to create the replica, which data to replicate, where to store data, the quantity of replica, the durability of replica, and consistency of the replica. Indexing module manages the organization of peers for indexing as well as method of indexing based on whether the indexing would be semantic-free or not. The logical module should determine the granularity of data to be indexed, the conceptual model to be used, and how data should be aggregated in case of discrepancies among schemas. Application & support modules should provide the necessary services for the application such as QoS and querying interface. Table I also lists three services: querying, streaming, and security. Querying service can be subdivided into languages that are used and organization of peers for improving querying performance. Streaming service is usually related to how peers should be organized for smooth live streaming or VoD. Security service provides the security of

peers and data while maintaining the robustness of the P2P system.

The traditional reusable components of P2P systems are close to the bottom modules (or layers) of the proposed model. Different P2P networks are proposed such as CAN, Chord, Tapestry, and these can be used by various applications. Whenever a new application needs to be developed, one of these P2P networks are chosen and the rest of the components are implemented independently. For upper modules (or layers) of P2P systems, there is almost no reusability. Each system has its own way of dealing with the problems although majority of these problems are shared problems. Our conceptual model indicates what modules can provide what types of services for applications. As a result, we suggest that whenever a new P2P application needs to be developed, the application developer should be eligible to select from the techniques in Table I. This is similar to selecting different transport protocols (e.g., TCP, UDP) for transport layer in networking and different indexing methods (e.g., B+ tree) in database systems. Similar case should be applicable for P2P systems.

Table II provides the strategies used in current P2P systems at different modules and services. It is important to note that not all P2P systems have modules that are specified here and support all services in this paper. When preparing Table II, one problem that we faced is missing layer or limited application. For example, if a technology does not provide any method on how to deal with updates of data, we decided the storage scheme based on the underlying P2P network. Some systems use additional layers to manage updates to avoid relocating data (e.g., logging in Ivy, versioning in OceanStore). We also provided whether the system is read-only or allows updates in Table II. We believe that Table II provides a good summary of what is accomplished in some P2P systems.

## 6 CONCLUSION

While a lot of data management strategies have been proposed for P2P systems through the last years, these strategies have not been investigated with respect to a common model for P2P systems. However, since the services provided by the P2P systems are so diverse, it is very challenging to come up with a common layer-based model for all P2P systems. The tradition in current P2P systems is to maximize the efficiency of their current applications. There has not been a clear strategy on how to provide new services without making major changes to a P2P system. In this paper, we have provided a conceptual model for P2P systems and the classification of data management and distribution strategies in P2P systems. Our conceptual model has modules as horizontal layers and services as columns. We have explained the storage, indexing, logical, base P2P service, support, and application modules as horizontal layers of our model. We have also discussed security, querying, and streaming services.

While working on this survey, we have realized that there are many specialized P2P systems. The bottom layers of P2P systems already use some common modules for establishing and managing the P2P systems. However, the rest of the system components turn out to be system specific with little-to-none reusability among higher levels of the P2P system. The applications are not developed in a modular way. Each application aims to optimize its application and P2P system is developed accordingly. We believe that the reusability for higher levels of P2P systems should be increased. There are many challenges which can be pursued as future work for P2P systems as listed below:

- *Functionalities of Modules.* The current methodologies target how some specific problems (e.g., data integration) are solved for a specific P2P application. After identifying the relative components for a module, the specific functionalities should be identified for the module. Moreover, the methods to support each functionality should also be provided for P2P system developers.
- *Compatibility of Methods at Different Modules.* There is limited work on the compatibility of various techniques that are used at different modules. For example, aggregated schema mediation may require P2P system to be structured. As future work, the compatibility of methods that are provided at different modules should be analyzed. A hierarchy of compatible methods needs to be provided. In cases where the methods are not compatible with each other wrapper-based techniques may be introduced to increase the independence between layers. The responsibilities of the modules should be clearly delineated to minimize the strong dependence between layers. The independence between modules is desired. For example, the mapping of query representations should be supported.

TABLE I

THE MODULES, SERVICES, and STRATEGIES.

| **Modules & Services** | **Issue** | **Method & Tools** |
|---|---|---|
| Base P2P Service | | Chord, Agent-based, DHT-Based, d-torus, Ring, Cell-Based, JXTA, HyperCup, Charles, unstructured, NFS-like, supernode, Tapetry, Pastry, BestPeer, Plaxton, Tree<br>(Note that these are also related with the storage layer) |
| Storage | Policy | Static, dynamic |
| | Data Naming | Implicit, explicit |
| | Data availability | Erasure Code, Replication |
| | Replica creation | Reactive, proactive |
| | Data to replicate | All data, requested data, hotspot |
| | Replica storage | Request route nodes, k-nearest nodes, random |
| | Replica quantity | Uniform, possibilistic, probabilistic |
| | Replica durability | Temporary, permanent |
| | Replica consistency | Instant, on-demand |
| Indexing | Indexing level | Semantic vs. Semantic-free, routing, resource |
| | Indexing organization | Centralized, pure distributed, super-peer |
| Logical | Granularity | Fine, gross |
| | Data modeling | Structured (Relational Schema, Global Relational System), Semi-structured & Unstructured (RDF/S, XML-Schema, (attribute, value) pairs) |
| | Schema Mediation | Individual, Aggregated |
| Application & Support | | Multimedia information retrieval, multimedia streaming QoS, file distribution, object location and routing, storage, database system, education, semantic web |
| Querying | Peer organization | Peer clustering, Semantic communities, Semantic clustering, Peer rings |
| | Query language | SQL-based, SQL, XQuery, XPath, RQL, RDF-QEL |
| Streaming | | Gossip-based, graph-based, implicit, tree-based, mesh-based |
| Security | | Self-certifying, information dispersal, fix tool, erasure code, trusted public key, reputation-based trust management |

TABLE II
THE SUMMARY OF SYSTEMS AND THEIR TECHNIQUES

| | Base P2P Service | Storage Naming vs. Dynamic | | Indexing | Logical | Querying | Application | Security | Streaming |
|---|---|---|---|---|---|---|---|---|---|
| **AmbientDB** | Over Chord | | | | GRS | SQLbased | MIR | | |
| **AXMEDIS P2P** | Over bittorrent | | | Semantic | MPEG-21, XML, Relational | Web-based | Digital content sharing | | |
| **BestPeer** | Agent-based | | static | Semantic | | | Data sharing | | |
| **BitTorrent** | | implicit | static (read-only) | Sem-free | | | File distribution | | |
| **ByPass** | Over any DHT-based system | | | Sem-free | | Peer clustering | File locating service | | |
| **CAN** | d-torus | implicit | dynamic | Sem-free | | | OLR | | |
| **CAN-Multicast** | CAN | | | | | | multicasting | | Intelligent flooding |
| **Chord** | Ring | implicit | dynamic | Sem-free | | | OLR | | |
| **CFS** | Over Chord | implicit | dynamic (limited update) | Sem-free | | | Storage | SC | |
| **CoolStream** | Gossiping | | | | | | multicasting | | Gossiping |
| **DBGlobe** | Cell-based | | static | Semantic | XML Schema | Semantic communities | Database system | | |
| **Edutella** | JXTA, HyperCup | | static | Semantic | XML Schema | Semantic clustering | Education | | |
| **Eliot** | charles | implicit | dynamic (read/write) | Sem-free | | | Storage | | |
| **Farsite** | Byzantine ring | Tree-based | static (read/write) | Sem-free | | | Storage | SC, ID | |
| **Free Haven** | unstructured | implicit | static | Sem-free | | | Anonymous Storage | ID | |
| **FreeNet** | structured (loosely) | implicit | static(limited update) | Sem-free | | | File sharing | | |
| **Gnutella** | Broadcast | | static | Sem-free | | | File sharing | | |
| **HIERAS** | Over Chord | | | | | Peer rings | Routing algorithm | | |
| **InterMemory** | | implicit | static (read only) | Sem-free | | | archival | EC | |
| **Ivy** | NFS-like | explicit | static (read/write) | Sem-free | | | File sharing | Fix tool | |
| **Kazaa** | supernode | | static | Sem-free | | | File sharing | | |
| **LpbCast** | gossiping | | | | | | multicasting | | Gossiping |
| **Napster** | unstructured | explicit | static | Sem-free | | | File sharing | | |
| **OceanStore** | Over Tapestry | | static (read/write) | Sem-free | | | Storage | EC | |
| **OPR** | Over structured | | | Sem-free | | | Data sharing | | |
| **PAST** | Over Pastry (read-only) | | | | | | Storage utility | SC | |
| **Pastry** | Ring | implicit | dynamic | Sem-free | | | OLR | | |
| **PeerDB** | Over bestpeer | | | Semantic | Relational schema | SQL | Database system | | |
| **Piazza** | unstructured | explicit | static | Semantic | XML schema | XQuery | Semantic web | | |
| **PinS** | DHT-based | implicit | dynamic (insert/delete) | Semantic | (attribute, value) pairs | B-tree | indexing | | |
| **PROMISE** | Over Pastry (read-only) | | | Sem-free | | | QoS, Multimedia | | Multiple-source selection |
| **Tapestry** | Plaxton | implicit | | Sem-free | | | OLR | Trusted publickey | |
| **ZigZag** | tree | | static (read-only) | Sem-free | | | QoS Multimedia | | tree-based |

[a]SC=Self-certifying, EC=Erasure coding, ID=Information dispersal, OLR= Object location and routing, MIR= Multimedia information retrieval, GRS=Global relational schema, Sem-free: Semantic-free

- *Services vs. Modules.* Each service should be investigated individually with respect to the modules of a common model. Possible ways of providing the service should be discussed and studied. For example, if security service is considered, possible ways of maintaining data and peer integrity with respect to the available modules should be studied.
- *Surveys on Modules.* The services of P2P systems are separated from the modules. We believe that specific surveys should be performed for each module in the future. The functionalities that can be provided to other modules should be investigated and presented in these surveys.
- *Structure of P2P Systems.* Some P2P applications (e.g., P2P streaming) also consider the structure of the P2P system as its main responsibility. However, the structure of the P2P systems should be the total responsibility of the base P2P service module. The base P2P service module should provide a wide range of structures for efficient retrieval as well as streaming. For example, hybrid systems can be studied in the future as they can use both gossiping and structured topology which may be more efficient than using only one of these alone.
- *Logical Data Management.* Modeling data and managing the schemas is one of the key problems for the logical module. The data management for P2P systems is not similar to federal or distributed databases since the global and consistent information distribution over the P2P system is not possible. Absence of a standard model that can be imposed on peers is a limitation for efficient querying. The problems of mapping schemas with respect to the same logical model as well as different logical models are one of the challenging problems to be studied as future work.
- *Data Updates.* One of the key issues with storage and indexing of data is the updates. Update operation is very critical as they may cause the data to be relocated. If the data is relocated, the indexing structure needs to be updated to locate the data. If the storage module supports replicas, it should be made sure that the replicas are consistent. The future surveys should study the burden of update operations of the P2P system and provide performance analysis whether static or dynamic storage needs to be followed.
- *Replica Management.* The management of replicas is one of the key factors that affect the performance of specific services such as querying, streaming, and security. However, there has not been any comparative research on replica management services when to create, where to create, how many to create, how long to maintain a replica, and how to manage replicas if updates are allowed. A comparative and performance analysis of replica management is required as a future work
- *Performance Studies.* The performance of applications with respect to chosen functionalities should be studied to observe the advantages and disadvantages of using those functionalities. The performance analysis can be provided in terms of retrieval accuracy, retrieval time, utilized storage area, robustness to churns, and streaming and presentation quality.

We believe that our model also serves as a guide for researchers and developers when they build novel P2P applications or provide additional services for their P2P systems. A common conceptual model like ours will help increase the reusability of methods.

**REFERENCES**

(Abad, 2004) Cristina Abad, William Yurcik, and Roy H. Campbell, "A Survey and Comparison of End-System Overlay Multicast Solutions Suitable for Network-Centric Warfare," SPIE Defense and Security Symposium / BattleSpace Digitization and Network-Centric Systems IV, 2004

(Adar, 2000) E. Adar and B. A. Huberman. “Free riding on gnutella,” Technical report, Xerox PARC, August, 2000.

(Adya, 2002) A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. P. Wattenhofer, “FARSITE: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment,” 5th Symposium on Operating Systems Design and Implementation, 2002.

(Agarwal, 2006) S. Agarwal and Shruti Dube. Gossip based streaming with incentives for peer collaboration. IEEE 8th International Symposium on Multimedia, San Diego, CA, December 2006.

(Andtroutellis-Theotokis, 2004) S. Andtroutellis-Theotokis, and D. Spinellis, “A Survey of Peer-to-Peer Content Distribution Technologies”, ACM Computing Surveys, Vol. 34, No. 4, pp. 335-371, December 2004.

(Arenas, 2003) M. Arenas, V. Kantere, A. Kementsietsidis, I. Kiringa, R. J. Miller, and J. Mylopoulos, "The Hyperion Project: From Data Integration to Data Coordination," In SIGMOD Record, Special Issue on Peer-to-Peer Data Management, 32(3), pp. 53-58, 2003

(Banerjee, 2002) S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in Proc. of ACM SIGCOMM 2002, Aug. 2002

(Bawa, 2003) M. Bawa, B. F. Cooper, A. Crespo, N. Daswani, P. Ganesan, H. Garcia-Molina, S. Kamvar, S. Marti, M. Schlosser, Q. Sun, P. Vinograd, and B. Yang, "Peer-to-peer research at Stanford," SIGMOD Record, September 2003

(Bellini, 2007) P. Bellini, I. Bruno, D. Cenni, P. Nesi, and D. Rogai. P2P architecture for automated B2B cross media content distribution. In *3rd International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution, AXMEDIS 2007*, pages 105–112, 2007.

(BitTorrent) BitTorrent. http://www.bittorrent.com.

(Boncz, 2003) P. A. Boncz, and C. Treijtel, "AmbientDB: relational query processing in a P2P network," International Workshop on Databases, Information Systems, and P2P Computing (DBISP2P) (co-located with VLDB 2003), Volume 2788 of Lecture Notes in Computer Science/Lecture Notes in Artificial Intelligence (LNCS/LNAI), © Springer-Verlag, Berlin, Germany

(Bonifati, 2008) Bonifati, A., Chrysanthis, P. K., Ouksel, A. M., and Sattler, K. 2008. Distributed databases and peer-to-peer databases: past and present. *SIGMOD Rec.* 37, 1 (Mar. 2008), 5-11.

(Bulterman, 2003) D. C. A. Bulterman, "Using SMIL to Encode Interactive, Peer-Level Multimedia Annotations," Proceedings of the 2003 ACM symposium on Document Engineering, pp. 32 – 41, Grenoble, France

(Castro, 1999) Castro, M., Liskov, B.: Practical byzantine fault tolerance. In: Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI'99), New Orleans, Louisiana, 1999

(Castro, 2002) M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "SCRIBE: A large-scale and decentralized application-level multicast infrastructure," IEEE Jrnl. on Selected Areas in Communications (J-SAC), Sp. Issue on Network Support for Group Communication 20, Oct. 2002

(Castro, 2003) M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: Highbandwidth multicast in cooperative environments," in Proc. of the 20th ACM Symp. on Operating Sys. Principles (SOSP 2003), Oct. 2003

(Christensen, 2006) Bent G. Christensen. "Experiences Developing Mobile P2P Applications with LightPeers," Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing (P2P'06).

(Chu, 2003) Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," in IEEE Jrnl. on Selected Areas in Communications (J-SAC), Sp. Issue on Network Support for Group Communication, 2003.

(Clark, 2000) I. Clark, O. Sandberg, B. Wiley, and T. Hong, "Freenet: A distributed anonymous information storage and retrieval system," Proc. of the Workshop on Design Issues in Anonymity and Unobservability, pp. 311-320, Berkeley, CA, July 2000.

(Cohen, 2002) E. Cohen and S. Shenker, "Replication Strategies in Unstructured P2P networks," SIGCOMM 2002, August 2002.

(Collection of CS Bib) Collection of Computer Science Bibliographies. http://liinwww.ira.uka.de/csbib. Accessed in December 2006

(Conti, 2006) Marco Conti, Jon Crowcroft, Franca Delmastro, and Andrea Passarella. "P2P support for Group-Communication Applications: a Cross-Layer approach for MANET Environments," IEEE INFOCOM 2006, Barcelona, Spain, 2006

(Cordasco, 2004) Gennaro Cordasco, Vittorio Scarano, and Cristiano Vitolo. "A P2P Distributed Adaptive Directory", Proc. of the 13th Int. Conf. on World Wide Web, WWW 2004, New York, NY, 2004.

(Dabek, 2001) F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-area cooperative storage with CFS", ACM SOSP 2001, Banff, October 2001

(Dan, 2005) Wang Dan and Zhao Rongjuan. "A layered resource management model in P2P system," Proc. of the 6th International Conference on Parallel and Distributed Computing, Applications, and Technologies (PDCAT'05), 2005.

(Danezis, 2005) G. Danezis, C. Lesniewski-Laas, M. F. Kaashoek, and R. Anderson, "Sybil-resistant DHT routing," in Proc. 10th European Symposium on Research in Comp. Sec., Milan, Italy, September 2005, pp. 305–318.

(Douceur, 2002) Douceur, J.R.: The Sybil attack. In: Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, Massachusetts (2002)

(Ding, 2004) H. Ding, I. Sølvberg, and Y. Lin, "A Vision on Semantic Retrieval in P2P Network," International Conference on Advanced Information Networking and Applications (AINA 2004), March 2004, Fukuoka, Japan.

(Eugster, 2001) P. Eugster, S. Handurukande, R. Guerraoui, A.-M. Kermarrec, and P. Kouznetsov, “Lightweight probabilistic broadcast,” in Proc. of the Intl. Conf. on Dependable Sys. and Networks (DSN 2001), July 2001

(Fahrenholz, 2004) D. Fahrenholtz and V. Turau, “A Tree-based DHT Approach to Scalable Weakly Consistent Peer-to-Peer Data Management,” PDMST '04 1st International Workshop on Peer2Peer Data Management, Security and Trust, 2004.

(Francis, 2006) P. Francis, Y. Pryadkin, P. Radoslavov, R. Govindan, and B. Lindell, “YOID: Your Own Internet Distribution.” http://www.isi.edu/div7/yoid/, Accessed in December 2006

(Free Haven) Free Haven. http://freehaven.net.

(Goldberg, 1998) A. V. Goldberg and P. N. Yianilos, “Towards an Archival Intermemory”. Proceedings of IEEE Advances in Digital Libraries (ADL 98), 1998.

(Habib, 2004) A. Habib and J. Chuang, "Incentive Mechanism for Peer-to-Peer Media Streaming," International Workshop on Quality of Service (IWQoS) pp. 171-180, June, 2004.

(Harwood, 2004) Aaron Harwood, Sarana Nutanong, Egemen Tanin, and Minh Tri Truong. “Complex Applications over Peer-to-Peer Networks,” ACM/IFIP/USENIX 5th International Middleware Conference, Toronto, Canada, 2004.

(Halevy, 2004) A. Y. Halevy, Z. G. Ives, J. Madhavan, P. Mork, D. Suciu, and I. Tatarinov. "The Piazza Peer Data Management System," IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 7, pp. 787-798, July 2004

(Hamlen, 2007) Hamlen, K.W.; Thuraisingham, B., "Secure peer-to-peer networks for trusted collaboration," *Collaborative Computing: Networking, Applications and Worksharing, 2007. CollaborateCom 2007. International Conference on* , vol., no., pp.58-63, 12-15 Nov. 2007

(Hasan, 2005) R. Hasan, Z. Anwar, W. Yurcik, L. Brumbaugh, and R. Campbell. A survey of peer-to-peer storage techniques for distributed file systems. In *International Conference on Information Technology: Coding and Computing, ITCC*, volume 2, pages 205–213, 2005.

(Hefeeda, 2003) M. Hefeeda, A. Habib, B. Botev, D. Xu and B. Bhargava, “PROMISE: Peer-to-Peer Media Streaming Using CollectCast,” ACM Multimedia, 2003, Berkeley, California, USA.

(Helder, 2002) D. A. Helder and S. Jamin, “End-host multicast communication using switch-tree protocols,” in Proc. of the 2nd Workshop on Global and Peer-to-Peer Computing on Large Scale Distributed Sys. (GP2PC 2002), May 2002

(Hellerstein, 2003) J. M. Hellerstein, “Toward Network Data Independence,” ACM SIGMOD Record, SPECIAL ISSUE: Special topic section on peer to peer data management, Volume 32 , Issue 3, pp. 34 – 40, September, 2003

(Hyojin, 2008) P. Hyojin, Y. Jinhong, P. Juyoung, G.K. Shin, and K.C. Jun. A survey on peer-to-peer overlay network schemes. In *International Conference on Advanced Communication Technology, ICACT*, volume 2, pages 986–988, 2008.

(Jannotti, 2000) J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O’Toole, Jr., “Overcast: Reliable multicasting with an overlay network,” in Proc. of the 4th Usenix Symp. on Operating Sys. Design and Implementation (OSDI 2000), Oct. 2000

(Jiang, 2003) X. Jiang, Y. Dong, D. Xu, and B. Bhargava. “GNUSTREAM: A P2P Media Streaming System Prototype,” IEEE International Conference on Multimedia & Expo (ICME), 2003

(JXTA) JXTA. http://www.jxta.org. Accessed in December 2006.

(Kalapriya, 2004) K. Kalapriya, S. K. Nandy, and V. K. Babu, “Can Streaming Of Stored Playback Video Be Supported On Peer to Peer Infrastructure?” In Proceedings of 18th International Conference on Advanced Information Networking and Application (AINA’04), pages Vol:2, pp. 200-203, Fukuoka City, Japan.

(Kalogeraki, 2004) V. Kalogeraki, F. Chen. "Managing Distributed Objects in Peer-to-Peer Networks," IEEE Networks Magazine, special issue on Middleware Technologies for future Communication Networks, Vol:18(1), pp. 22-29, 2004.

(Karnstedt, 2004) M. Karnstedt, K. Hose and K. Sattler, “Query Routing and Processing in Schema-Based P2P Systems,” Proceedings of the 15th International Workshop on Database and Expert Systems Applications (DEXA’04), 2004

(Kazaa) Kazaa, http://www.kazaa.com.

(Kementsietsidis, 2003) A. Kementsietsidis, M. Arenas, and R. J. Miller, “Mapping Data in Peer-to-Peer Systems Semantics and Algorithmic Issues”, SIGMOD 2003, pp. 325-336, June, 2003, San Diego, CA.

(Kokkinidis, 2004) G. Kokkinidis and V. Christophides, “Semantic Query Routing and Processing in P2P Database Systems: The ICS-FORTH SQPeer Middleware,” HDMS '04, Athens, Greece, 2004

(Kostic, 2003) D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, “Bullet: High bandwidth data dissemination using an overlay mesh,” in Proc. of the 20th ACM Symp. on Operating Sys. Principles (SOSP 2003), Oct. 2003

(Kwon, 2004) G. Kwon and K. D. Ryu, “BYPASS: Topology-Aware Lookup Overlay for DHT-based P2P File Locating,” In the 10th IEEE International Conference on Parallel and Distributed Systems (ICPADS), Newport Beach CA , July 2004.

(Li, 2003) Z. Li and P. Mohapatra, "Hostcast: A new overlay multicasting protocol," in Proc. IEEE 2003 Intl. Conf. on Communications (ICC 2003), May 2003

(Lin, 2004) W. K. Lin, D. M. Chiu, Y. B. Lee, "Erasure Code Replication Revisited," Proceedings of the 4th IEEE International Conference on Peer-to-Peer Computing (P2P'04), August, 2004.

(Liu, 2006) J. Liu, S. G. Rao, B. Li, and H. Zhang, Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast, invited by Proceedings of the IEEE, 2006

(Liu, 2008) Yong Liu, Yang Guo and Chao Liang, "A survey on peer-to-peer video streaming systems" vol. 1, no. 1 pp. 18-28, March, 2008,

(Lua, 2005) Eng Keong Lua; Crowcroft, J.; Pias, M.; Sharma, R.; Lim, S., "A survey and comparison of peer-to-peer overlay network schemes," *Communications Surveys & Tutorials, IEEE* , vol.7, no.2, pp. 72-93, Second Quarter 2005

(Lv, 2002) Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and Replication in Unstructured Peer-to-Peer Networks," In Proceedings of 16th ACM International Conference on Supercomputing, June, 2002

(Magharei, 2007) Magharei N, Rejaie R (2007) Prime: peer-to-peer receiver driven mesh-based streaming. In: Proceedings of IEEE INFOCOM

(Maniatis, 2005) Petros Maniatis, Mema Roussopoulos, T. J. Giuli, David S. H. Rosenthal, and Mary Baker. "The LOCKSS peer-to-peer digital preservation system". ACM Transactions on Computing Systems, 23(1):2--50, 2005

(Maxim, 2003) R. Maxim, S. C. Hui, "Intelligent Content-Based Retrieval for P2P Network," International Conference on Cyberworlds, pp. 318, December 2003

(Mielke, 2002) M. Mielke, R. S. Aygun, Y. Song, and A. Zhang. "PLUS: Probe-Loss Utilization Streaming Mechanism for Distributed Multimedia Presentation Systems", IEEE Transactions on Multimedia, Vol:4(4), pp. 561-577, December 2002.

(Milojicic, 2002) D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. "Peer-to-peer computing," Technical Report HPL-2002-57, HP Lab, 2002.

(Muthitacharoen, 2002) A. Muthitacharoen, R. Morris, T. M. Gil, and B. Chen. "Ivy: A read/write peer-to-peer file system," In the Proceedings of the 5th USENIX Symposium on Operating Systems Design and Implementation (OSDI '02), Boston, Massachusetts, December 2002.

(Napster) Napster, http://www.napster.com.

(Nejdl, 2003) W. Nejdl, W. Siberski, and M. Sintek, "Design issues and challenges for RDF- and schema-based peer-to-peer systems," ACM SIGMOD Record, SPECIAL ISSUE: Special topic section on peer to peer data management table of contents, Volume 32 , Issue 3, pp. 41-46, 2003.

(Nejdl, 2002) W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmr and T. Risch, "EDUTELLA: a P2P Networking Infrastructure based on RDF," 11th International World Wide Web Conference Hawaii, USA, May 2002

(Ng, 2003) W. S. Ng, B. C. Ooi, K. L. Tan, and A. Y. Zhou, "PeerDB: A P2P-based system for distributed data sharing," In Proceedings of the 19th International Conference on Data Engineering, Bangalore, India, March 2003

(OceanStore) OceanStore. http://oceanstore.cs.berkeley.edu

(Ooi, 2003) B.C. Ooi, Y. Shu, and K.L. Tan, "Relational Data Sharing in Peer-based Data Management Systems," ACM Sigmod Record, Special issue on P2P data management, Vol: 32(3), pp. 59-64, 2003

(Osais, 2006) Y. Osais, S. Abdala, and A. Matrawy. "Multilayer Peer-to-Peer Framework for Distributed Synchronous Collaboration," IEEE Internet Computing, Vol:10, No: 6, pp. 33-41, Nov-Dec 2006

(Ouksel, 2003) A. M. Ouksel, "In-context peer-to-peer information filtering on the Web," ACM SIGMOD Record, SPECIAL ISSUE: Special topic section on peer to peer data management, Vol:32(3), pp. 65 – 70, 2003

(Ou, 2003) C. Qu and W. Nejdl, "Searching SCORM Metadata in a RDF-based E-Learning P2P Network Using XQuery and Query by Example," In Proc. of the 3rd IEEE International Conference on Advanced Learning Technologies (IEEE ICALT 2003), July 2003, Athens, Greece

(Patro, 2003) S. Patro and Y. C. Hu, "Transparent Query Caching in Peer-to-Peer Overlay Networks," In Proceedings of the 17th International Parallel and Distributed Processing Symposium (IPDPS), Nice, France, April 2003.

(Pendarakis, 2001) D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, "ALMI: An application level multicast infrastructure," in Proc. of the 3rd Usenix Symp. on Internet Technologies and Sys. (USITS 2001), Mar. 2001

(Pitoura, 2003) E. Pitoura, S. Abiteboul, D. Pfoser, G. Samaras, and M. Vazirgiannis, "DBGlobe: a service-oriented P2P system for global computing," ACM SIGMOD Record, Vol:32(3), pp. 77-82, 2003

(PNRP) Peer Name Resolution Protocol. http://www.microsoft.com/technet/network/p2p/pnrp.mspx. published in September 2006.

(Portmann, 2004) Marius Portmann, Sébastien Ardon, Patrick Senac, and Aruna Seneviratne. "PROST: A Programmable Structured Peer-to-peer Overlay Network," Proc. of the 4th Int. Conf. Peer-to-Peer computing, Zurich, Switzerland, 2004.

(Ratnasamy, 2001a) S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," In proceedings of the 2001 ACM SIGCOMM, pp. 161-172, 2001.

(Ratnasamy, 2001b) S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Application-level multicast using content-addressable networks," in Proc. of 3rd Intl. Workshop on Networked Group Communication (NGC 2002), Nov. 2001

(RDF) Resource Description Framework (RDF). http://www.w3.org/RDF/

(Risson, 2006) J. Risson and T. Moors: "Survey of Research towards Robust Peer-to-Peer Networks: Search Methods", Computer Networks, 50(17):3485-521, Dec. 2006

(Rowstron, 2001) A. Rowstron and P. Druschel. "Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility," 18th ACM SOSP'01, Lake Louise, Alberta, Canada, 2001

(Rowstron, 2001b) A. Rowstron and P. Druschel. "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," In Proceedingsof IFIP/ACMMiddleware. Heidelberg, Germany, 2001

(Ryu, 2007) Ryu, S.; Butler, K.; Traynor, P.; McDaniel, P., "Leveraging Identity-Based Cryptography for Node ID Assignment in Structured P2P Systems," *Advanced Information Networking and Applications Workshops, 2007, AINAW '07. 21st International Conference on* , vol.1, no., pp.519-524, 21-23 May 2007

(Sahin, 2004) O. D. Sahin, A. Gupta, D. Agrawal, and A. El Abbadi, "A Peer-to-peer Framework for Caching Range Queries". In 20th International Conference on Data Engineering, pp. 165-176, Boston, Massachusetts, 2004

(Serafini, 2001) L. Serafini, F. Giunchiglia, J. Mylopoulos, and P. Bernstein. The Local Relational Model: Model and Proof Theory. IRST Technical Report 0112-23, Instituto Trentino di Cultura, December 2001

(Shirky, 2001) C. Shirky, "Listening to Napster," Book chapter in Peer-to-Peer: Harnessing the Power of Disruptive Technologies, edited by Oram, Andy, 2001, O'Reilly.

(Stein, 2002) C. A. Stein, M. J. Tucker, and M. I. Seltzer, "Building a Reliable Mutable File System on Peer-to-Peer Storage," 21st IEEE Symposium on Reliable Distributed Systems (SRDS'02), p. 324, 2002

(Stoica, 2001) I. Stoica, R. Morris, D. R. Karger, M. F. Kaashock, and H. Balakrishman, "Chord: A scalable peer-to-peer lookup protocol for internet applications," In Proceedings of the ACM SIGCOMM, pp. 149--160, San Diego, California, August 2001.

(Valduriez, 2004) P. Valduriez and E. Pacitti, "Data Management in Large Scale P2P Systems," VECPAR, pp. 104-118, 2004.

(Tran, 2004) D. A. Tran, K. Hua, and T. Do, "A peer-to-peer architecture for media streaming," IEEE Journal on Selected Areas in Communications, Special Issue on Advances in Service Overlay Networks, vol. 22, pp. 121-133, Jan 2004.

(Tsoumakos, 2003) D. Tsoumakos and N. Roussopoulos. "Adaptive Probabilistic Search for Peer-to-Peer Networks," in 3rd IEEE Intl Conference on P2P Computing, 2003

(Tsoumakos, 2006) Dimitrios Tsoumakos and Nick Dimitrios Roussopoulos. "APRE: A Replication Method for Unstructured P2P Networks," Technical Reports of the Computer Science Department of University of Maryland, February, 2006

(Tsybulnik, 2007) Tsybulnik, N.; Hamlen, K.W.; Thuraisingham, B., "Centralized Security Labels in Decentralized P2P Networks," *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual* , vol., no., pp.315-324, 10-14 Dec. 2007

(Turcan, 2002) Eduard Turcan, Nahid Shahmehri, and Ross Lee Graham. "Intelligent Software Delivery Using P2P," Proceedings of the Second International Conference on Peer-to-Peer Computing (P2P'02).

(Villamil, 2004) M.-D.-P. Villamil, C. Roncancio, C. Labbe. "Pins: Peer to Peer Interrogation and Indexing System," Database Engineering and Applications Symposium, 2004. IDEAS '04. pp. 236- 245, 2004

(Vlavianos, 2006) Vlavianos A, Iliofotou M, Faloutsos M (2006) Bitos: enhancing bittorrent for supporting streaming applicati ons. In 9th IEEE global internet symposium 2006, April 2006

(Wallach, 2002) D. S. Wallach, "A survey of peer-to-peer security issues," International Symposium on Software Security, Tokyo, Japan, 2002

(Wang, 2006) Chen Wang, Li Xiao, Yunhao Liu, Pei Zheng, "DiCAS: An Efficient Distributed Caching Mechanism for P2P Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 10, pp. 1097-1109, October, 2006.

(Windows P2P) Windows Peer-to-Peer Networking. http://www.microsoft.com/technet/prodtechnol/winxppro/deploy/p2pintro.mspx. Accessed in December 2006.

(Xiang, 2004) Z. Xiang, Q. Zhang, W. Zhu, Z. Zhang,and Y-Q. Zhang. "Peer-to-Peer Based Multimedia Distribution Service," IEEE Trans. On Multimedia, Vol:6(2), pp. 343-356, April 2004

(XML Base) XML Base. http://www.w3.org/TR/xmlbase/

(Xu, 2002) D. Xu, M. Hefeeda, S. Hambrusch, and Bharat Bhargava. “On Peer-to-Peer Media Streaming,” Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02) pp. 363-371, 2002

(Xu, 2003)Z. Xu, R. Min and Y. Hu, “HIERAS: A DHT-Based Hierarchical Peer-to-Peer Routing Algorithm,” Proceedings of the 2003 International Conference on Parallel Processing (ICPP'03), pp. 187-194, Kaohsiung, Taiwan, October, 2003

(Yang, 2003) C. Yang. “Peer to Peer Architecture for Content Based Music Retrieval on Acoustic Data,” WWW 2003, pp. 376-383, May 20–24, 2003, Budapest, Hungary.

(Zhang, 2005) X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, CoolStreaming/DONet: A Data-driven Overlay Network for Live Media Streaming, IEEE INFOCOM'05, Miami, FL, USA, March 2005

(Zhang, 2003) M. Zhang, and K-L. Tan, “Supporting Rich Queries in DHT-Based Peer-to-Peer Systems,” Twelfth International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, pp. 95, 2003 Linz, Austria, Y. Zhu, H. Wang, and Y. Hu. “Integrating Semantics-Based Access Mechanisms with P2P File Systems,” Third International Conference on Peer-to-Peer Computing (P2P'03), pp. 118, 2003

(Zhao, 2004) B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz. “Tapestry: A Resilient Global-Scale Overlay for Service Deployment,” IEEE Journal on Selected Areas in Communications, Vol:22, No:1, pp. 41-53, January 2004.

(Zhou, 2008) Jian Zhou; Bhuyan, L.N.; Banerjee, A., "An effective pointer replication algorithm in P2P networks," *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on* , vol., no., pp.1-11, 14-18 April 2008

(Zhuang, 2001) S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. Katz, and J. Kubiatowicz, “Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination,” in Proc. of the 11th Intl. Workshop on Network and Operating Sys. Support for Digital Audio and Video (NOSSDAV 2001), June 2001