Peer-to-peer Data Integration with Distributed Bridges

N. Arthorne, B. Esfandiari
Department of Systems and Computer Engineering
Carleton University
1125 Colonel By Drive, Ottawa, Canada, K1S 5B6
babak@sce.carleton.ca

Abstract

This paper proposes an approach to collaborative and distributed data integration that relies on peer-to-peer data sharing and the creation of "bridges" between data sources. The bridges are themselves created and shared in a peer-to-peer manner, making this approach scalable and accessible to 3rd party applications. A case study of a distributed digital library is used to illustrate and validate the approach.

1 Introduction

The integration of autonomous, heterogeneous data sources can be a crucial advantage to both large and small-scale databases systems that operate in both private commercial systems and in public sector and academic research. There are many reasons to integrate data across multiple sources, the foremost being the desire of a user to search across multiple databases using a single query or query language and to return a result that contains data from multiple sources.

The traditional methods of data integration use a centrally administered global schema that is not well-suited to a large number of data sources that frequently change. Using the peer-to-peer approach, we propose the use of user-contributed mappings or "bridges" that can be published and exchanged over a network of peers. The Universal Peer-to-Peer (U-P2P) [17, 7] is a framework that allows peer-to-peer sharing of documents enclosed in a user-defined and searchable XML structure. A U-P2P "community" specifies the structure of the documents to be shared (using XML Schema) and the type of deployment (centralized, peer-to-peer, etc) to be used for publication and search. Web forms for searching and publishing documents can then be derived from the schema or customized by the creator of the community. In this paper we describe how to provide peer-to-peer data integration by using and extending the key concepts that were implemented in U-P2P. We first give a background on data integration approaches as well as recent work on peer-to-peer data integration. We then introduce the U-P2P framework and argue why its concepts are a good candidate for use in peer-to-peer data integration. Next, we describe our approach to data integration with U-P2P. We finally illustrate our approach with a case study integrating digital libraries using two different formats.

2 Background

2.1 Data Integration

The integration of autonomous, heterogeneous sources of data is a widely researched field that has changed over the years to reflect new data formats and new methods of providing access to data. The main challenges deal with the differences in distribution, autonomy and heterogeneity [19]. Indeed, the databases may need to be accessed by specific protocols or networks across a geographically wide area with implications for security, bandwidth requirements and network latency. Also, organizations that create and maintain data sources prefer to remain in control of all aspects of their databases. Any integration project must start by addressing who will gain or lose control of the data, and if any, what aspects of autonomy need to change [19]. This implies that each database is designed differently and with different needs, requirements and end users in mind. The differences in design philosophy often produce the low-level differences in data formats, databases schemas, access control and other functionality. This autonomy can then lead to heterogeneity issues, such as differences in Database Management Systems, syntax, database schemas and data models and perhaps most crucially differences in the meaning or intended use of data fields stored in each database.

A typical approach to data integration consists of a federated schema that acts as a front-end and has the means to access all the databases through their various interfaces and query languages [20]. This can work as long as the database schemas are stable and that databases aren't added or removed dynamically from the system. A mediator approach addresses this by first wrapping each database in a layer that translates database-specific queries into a common information model, and then by using the front-end (the mediator) to combine the query results to present to the user [12]. The mediator can then communicate through a single interface with the translators and other mediators, and this provides the added flexibility. One limitation of this approach is the burden it puts on each database to provide the appropriate translator.

Third Generation approaches to data integration focus on integrating semantic differences in addition to the differences in data syntax and structure mentioned earlier. The emphasis is on knowledge and the possibility of inference and reasoning. For this to be possible, it is necessary to define a *shared ontology*, i.e. a vocabulary for describing a specific domain of information [14]. This will provide a model at a semantic level of the shared data. With an ontology or vocabulary that describes concepts in a set of data, we can create mappings into other ontologies and describe the relationships embodied in the mappings using predefined terms. For example, an equivalent mapping can be used to map the term 'Author' in one ontology to the term 'Creator' in another ontology. Once this mapping is established, it is possible to use automated tools that explore the mappings and allow a query to be translated and answered by data that uses a different ontology. Mappings are typically established in one of two ways: global as view (GAV) or local as view (LAV).

In GAV, the elements of a global schema are defined as views over the local schemas. A view is simply a named query or virtual relation that is created from existing database relations. A disadvantage of GAV is that the local schemas must be stable and should not change frequently. If they do change, elements of the global schema will be affected and views may need to be redefined to re-establish a complete description of the global schema.

In LAV, each element of the local schema is defined as a query over the global schema. This leads to problems when the global schema does not have enough information to describe the relations in the local schema and the result is that local schemas may be incompletely described. An advantage of LAV mappings is that the source databases can change their schemas without disturbing the global schema. The global schema, on the other hand, must remain relatively stable and unchanged. If the global schema were to change, each database might have to redefine its views over the global schema. Thus LAV is suitable when there are many data sources that frequently change and one global schema that remains stable. Such would be the case when an enterprise uses a globally shared ontology to which all local databases relate.

2.2 Peer-to-peer Data Integration

A peer-to-peer networking approach to data integration tackles the problems of distribution and autonomy by making each node on the network equal and independent of its peers. The primary advantage of peer-to-peer is redundancy of data. Redundancy increases availability and the speed at which data or resources can be retrieved from the network. Instead of having a central server that gets flooded with requests from clients, data transfers take place between individual peers and the availability of a file will increase as more peers retrieve a copy of the data and share it on their system. Since every node is a client and server, the joining or leaving of a node will not adversely affect the availability of a file, provided that there are enough copies stored throughout the network.

Aside from distributing data across a set of peers, the peer-to-peer approach has several implications with respect to data integration. First, a globally shared schema is eschewed in favour of each peer maintaining a schema for its own data. Thus, there is no global schema that must be centrally maintained as in federated databases, or extra layers of translation software as in mediator-based systems. Instead, mappings are from the local schema or peer schema to schemas used by other peers. This is a great advantage in large data integration systems that use many different peers in a dynamic network environment. With no central schema to modify, peers can join the network, advertise their schemas and establish mappings to new schemas. A disadvantage is that mappings between peer schemas may quickly become outdated or invalid when a peer disconnects from the network or modified their schema. Also, the mappings have to be exchanged over the network or stored in a central location where they can be used in query translation. Each peer will use the mappings to reformulate their queries before sending them to the network, or alternatively peers could accept queries over different schemas and perform the query translation on behalf of the peer that sent the query.

Lenzerini [15] presents a framework for peer-to-peer data integration using logical descriptions. It describes each peer as holding a set of relations (given in Datalog or first order logic (FOL)) that describe the data they make available on the network. Each peer has a local or peer schema and holds mappings to other peer schemas to be used in query translation. In [13], the authors describe their mapping language used in mediating or integrating peer schemas in the Piazza Peer Data Management System (PDMS). However, in all such systems, complete knowledge of the network is assumed and all mappings are centrally stored for use in computations. This limits the use of these systems in a practical environment where no centralized store is used and networks are dynamic with nodes connecting and disconnecting continuously.

Our proposed approach, and its difference with the work mentioned above is that we don't consider our peer-to-peer network to consist of a federated database system under unique control. For instance, we are not preoccupied with the problem of global consistency or update. We are rather interested in searching, publishing and retrieving data, potentially conflicting, that is produced by independent sources. This is consistent with current peer-to-peer file-sharing systems in which one does not control the production of data by third party sources, and therefore cannot modify or delete it. What we propose in this paper is to allow third parties (i.e. any data source in the peer-to-peer network) to publish mappings (bridges) between existing data, and to share such bridges as pieces of data themselves. This can in turn lead to the possibility of contradictions or cycles between bridges. The issue of cycles is partially addressed by the use of a Time To Live (TTL) in the Gnutella protocol. To address the issue of contradictions as well as false bridges, one could restrict the search for bridges to only those that are authored by trusted nodes or certified by them.

Since in our proposal there is no global control over the bridges, there is no guarantee of completeness of search results either. We are merely proposing to leverage the "network effect" of many publishers who create bridges for their own benefit and share them with others.

Our approach uses and extends an existing peer-to-peer file-sharing framework called U-P2P, and turns it into a P2P data integration framework. The following section describes U-P2P and explains why it can be used as the starting point for our approach.

2.3 U-P2P

U-P2P [17, 7] is an open source framework for sharing data described in XML over any supported peer-to-peer network. At its core, the three shared components in U-P2P are resources, communities and attachments.

2.3.1 A U-P2P Resource

A *resource* is a data record that is written in XML and conforms to a specific XML Schema. It may contain metadata that describes a physical object, such as a book, or a network-accessible object such as a website. For example, in a digital repository a record could describe a scientific paper that is attached to the record in electronic form. A resource may also be a data record itself and may not reference an attachment or external entity. For example, a resource may be a name and address that conforms to a schema for address book entries.

2.3.2 A U-P2P Attachment

An *attachment* is a file that is bundled or associated with a resource. If a resource is downloaded from a peer, so are its associated attachments. Thus, as mentioned above, a resource can serve to describe an attached object or it can have no attachments and simply contain a data record. Because attachments are binary objects associated with a resource, they cannot be searched like resource metadata. Instead, users search through the metadata in a resource that describes the binary attachments.

2.3.3 A U-P2P Community

A *community* is a conceptual grouping of U-P2P resources that follow the same schema and are shared using the same P2P protocol. As such it is a combination of:

- an XML Schema to which all U-P2P resources shared in the community will conform,
- stylesheets to display, publish and search for the shared data,
- keywords to describe the community
- and a *Network Adapter* (described further) that determines the type of peer-to-peer deployment used by the community.

For example, a Stamp community created for sharing descriptions of stamp collections would provide a schema for a stamp (i.e. its country of origin, its nominal value, its quality, a picture, etc.), stylesheets to convert the schema into entry forms for publication and search, another stylesheet for the display of an instance of stamp, and the choice of deployment via the Network Adapter.

A community itself is a resource and is published in the Community community. This allows for the discovery of communities in the same way as for any other resource.

2.3.4 Network Protocols

The underlying peer-to-peer network layer for a given community is handled in U-P2P by a special attachment called a Network Adapter. A Network Adapter is responsible for providing a means to search the network, retrieve a file from the network and publish a file to the network. The way in which these operations are completed depends on the implementation of the adapter. For example, the default Network Adapter included with U-P2P is the Napster style of peer-to-peer networking, i.e. using a central registry for resource advertisements, while the actual resources are downloaded in a peer-to-peer manner.

2.3.5 Synthesis

Figure 1 summarizes the relations between the main concepts in U-P2P: a resource is essentially used to share attachments using descriptive metadata (not represented in the diagram). A community is used to specify the template for a resource, in the form of an XML Schema document. A community itself can be discovered and published like any other resource (this is represented using the inheritance relationship in the diagram). Among its attachments are the XML Schema of the resources it shares, as well as the Network Adapter used for the choice of peer-to-peer protocol. The bootstrap community that shares communities is called the *Community community.*

The advantage of sharing communities just like any other resource is that anyone can create their own peer-to-peer network. For example, if a user logs on and sees no suitable community for sharing their data, they can create a schema, write up the description and start a new community. As long as they publish the community to the Community community,

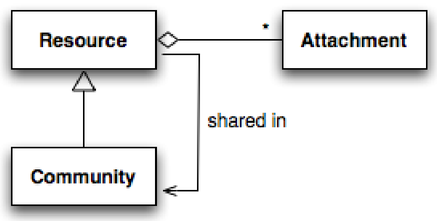

Figure 1: Conceptual View of U-P2P

other users will be able to join it and start sharing data.

2.3.6 Using U-P2P

The user interacts with U-P2P through 3 main activities: Create, Search and View. Each activity is carried out through a web browser (U-P2P is based on Java Servlets and JSPs). Each web page is generated by applying an XSLT stylesheet to the XML output of the respective activity (provided by the community creator as additional attachments to the community definition). Thus, the Create stylesheet transforms the retrieved community Schema into a web form where the user can provide metadata and the attachments.

The fields that can be used in the Search activity are again dependent on the author of the Search stylesheet, but generally they are the same fields that can be entered on the create page, minus the attachments. Search terms in U-P2P are specified using XPath [4] and consist of a set of XPath value pairs that must be matched in documents in the database or in documents shared by users on the network. For example, a search for a book might use the XPath value pairs /book/title=Great Expectations and /book/author=Dickens.

U-P2P also provides default stylesheets that can be used in the absence of user-defined ones.

2.3.7 U-P2P as a Candidate for Data Integration

There are mainly three reasons why we believe that U-P2P concepts make it a good candidate for Data Integration:

- *Communities as a First-class Concept:* U-P2P allows any user to create a new community of interest based on an XML Schema and network adapter. This is the equivalent of giving every user the power to create a network like Napster or Gnutella, but centered on their particular type of shared data. This is important for data integration because it allows networks of data to be formed around existing XML Schemas.
- *Rich Metadata Searches:* U-P2P goes beyond simple keyword searches used in file-sharing systems and allows structured metadata to be published and searched on the network. Complex searches are required in order to search across integrated data, as the structure of the data may hold semantic information that is relevant to mappings between schemas.
- *Flexible Peer Deployment:* Through the Network Adapter modules, U-P2P also allows flexibility in how a community is deployed. New protocols can be incorporated by writing a new adapter and plugging it into U-P2P. This flexibility means data sources can access the network with the latest protocols and not be tied to one topology or type of deployment. Certain network deployments may also be unsuitable for peers sharing large amounts of data in an integrated system. For example, a protocol such as the one used in Napster that requires uploading all metadata to a centralized index would not be suitable for peers with large databases.

However, a flexible P2P framework is not enough by itself to be used for peer-to-peer data integration. The following section defines our requirements and how they were addressed.

3 Peer-to-peer Data Integration with U-P2P

3.1 Requirements

- *Bridging Communities:* A user in one community should be able to query another community by translating their query to the different schema. This requires mappings between schemas and a mechanism to allow searches to be automatically translated to the target community. Our proposed way to deal with this is to create "bridges" between communities. A bridge can be modeled as a U-P2P resource. The resource would indicate the two endpoints of the relationship (i.e. the two communities that are being linked). A bridge in U-P2P also needs to describe the semantic relationship between communities to allow for higher level traversal of communities related within a domain. While going through the mappings, the query should be translated, dispatched to the target community and results should be returned in either the original schema or the target community's schema. The bridge format needs to be flexible enough to support all kinds of mappings and new types of relationships between schemas that may be required in the future. For example, attachments will be used to describe the ontology mappping (using an OWL document) and/or the query translation (using XQuery).

The most interesting consequence of this solution is that the creation of bridges is possible *for all users*, as long as they decide to become members of the *Bridge Community*. New bridges can be created and shared by anybody as new databases appear or as schemas change.

- *"Pull" Model for Search:* The choice of the network deployment is also crucial. One cannot expect each data source to publish its metadata to a central repository, as it would in a Napster-like model. A Network Adapter implementing a fully-distributed peer-to-peer protocol such as Gnutella is required to remove the dependency on a centralized server used for indexing metadata. As a result, databases only need to react to queries, answering them while also propagating them to their neighbors. Query cycles are broken through the implementation of a time-to-live (TTL). Furthermore, a fully-distributed protocol scales up to a larger number of users, which is essential for integrating large amounts of shared data. This is especially true when Gnutella routing is restricted to the specific community, which can then be viewed as an overlay network: neighbors are obtained from members of the same community, and only they will be recipients of the query.

- *Access to Legacy Data:* A final requirement, which required some extension to the initial U-P2P framework, is to make existing databases accessible on the peer-to-peer network using proxies. A peer acting as a proxy for a database translates incoming queries into the database query language and responds to data requests by translating data into the XML format used in a community. The peer also takes care of maintaining a Gnutella neighborhood and propagating search queries to those neighbors, in a way that is entirely transparent to the database. This effectively brings large amounts of data to the network without having to modify databases or harvest their metadata and translate it to XML.

The following sections will describe the realization of the above requirements in detail.

3.2 The Bridge Community

The following simple examples might help to further motivate the use of bridges in U-P2P:

1. A user in the Carleton University Library community wishes to expand their search to include related resources in the University of Calgary library. The user wants to search for books or journal articles using the schema of the Carleton library community and be presented with results in the same schema.

2. A research lab shares chemical molecules in a community in U-P2P. A student in the lab performs a search in the chemical community and results are returned with various chemical molecules and compounds. A link on the search results page offers to perform the same molecule search

on a related medical database. The user follows the link and a search is performed in the medical database with the results presented in the medical database schema. The user was previously unaware of the existence of the medical database or its connections to chemical molecules.

Given the two simple scenarios above, the requirements for a bridge can be summarized as follows:

1. Provide a means to define a semantic relationship between two uniquely identified resources in U-P2P. The relationships may be a known property in OWL/RDFS such as “same-as”, or a more complex user-defined relationship.

2. Provide a means to translate a query destined for one community into a query compatible with a different community. This is needed for both examples.

3. Provide a means to transform the instance data (i.e. the XML resources themselves) of one community to another community. This is useful for the first example.

A Bridge in U-P2P is defined and published as a resource in the Bridge Community, where other peers can find and retrieve it for use in query translation and ontology mappings. One key point of building a Bridge Community in U-P2P is to decouple the making of bridges from the publishing of communities. Anyone can publish a community in the Community community and anyone can create a bridge in the Bridge community. This means that a user who creates a bridge may not be the author of either of the communities that the bridge connects, but instead, may be a third party who is simply making the connection and publishing it for others to use.

Figure 2 shows how the Bridge concept relates to the other aspects of U-P2P. A bridge is a resource which, as part of its description, specifies two resources as its endpoints.

Let us now detail the various components of a bridge. To do so we examine the elements of the schema for the Bridge Community:

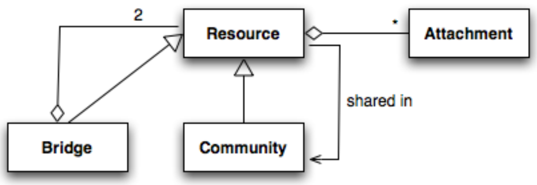

Figure 2: A Bridge in U-P2P

- Metadata: Title and description of the bridge. What the bridge does and what it is used for in general terms.

- Relation: OWL relationship between the two resources. This can be replaced by user-defined relationships using an XML namespace other than OWL.

- Bridge Source: ID of the resource and community of the source of the bridge.

- Bridge Target: ID of the resource and community of the endpoint of the bridge to which the source relates.

- Transform List: List of binary attachments with an accompanying type and description that are used for translating queries and data.

Notably, a U-P2P bridge contains a Web Ontology Language (OWL) statement (the Relation as defined in the schema) that relates one resource to another resource. This provides a simple high-level statement about how the two communities, or any two resources for that matter, are related. Each resource is identified by a Uniform Resource Identifier (URI) and the relation statement in the bridge uses the URIs to designate the endpoints of the relationship, i.e. the object and the subject of the OWL triple. The U-P2P URI format is as follows:

```
up2p:community=<community_id>/
    resource=<resource_id>
```

Where `community_id` and `resource_id` are unique hash values assigned to the resources when they are uploaded to U-P2P.

If the endpoint resources are in the Community community, then the bridge defines a relationship between two community schemas.

The requirement of translating instance data will be satisfied by providing an attachment in a U-P2P bridge to affix a document or file that is used to transform data. The third requirement for a bridge of providing query translation is specific to the relation defined in the bridge. The same mappings used to translate instance data may also be used to translate queries. The format of such documents in U-P2P is an XSLT stylesheet or alternatively an XPath mapping. The processing of these mappings is then performed respectively by an XSLT or XQuery engine. The following is an example of an XPath mapping in U-P2P:

```
<XPathMappings>
  <mapping source="/item/metadata/dc:title"
    target="/record/name" />
</XPathMappings>
```

In this example the XPath `/item/metadata/dc:title` is mapped to `/record/name`. It is important to note that the target XPath can be more complex but that overall, the mappings are a simple translation mechanism that can be supplanted by more complex data translation schemes.

Finally, the bridge-related information must be exploited in a user-friendly way. This requires a mechanism to retrieve a list of bridges related to a community without having the user execute a manual search. This requires that U-P2P lookup the mappings in all bridges related to the current community and offer links to perform the new searches.

3.3 Gnutella Search

The main benefit of using the Gnutella protocol for search was already explained in the previous section: it is not reasonable to expect each data source to advertize all of its content to a central registry; instead, by using Gnutella, such sources will only be accessed on an as-needed basis. In [10], in order to ensure complete and sound query answering, queries are propagated to all neighboring nodes while avoiding cycles by keeping track of query ids. But in existing peer-to-peer file-sharing networks with millions of nodes, with peers that appear and disappear very frequently, the requirement for a complete result for a search query is not as reasonable. The Gnutella protocol uses a Time-To-Live mechanism to avoid flooding the network with searches.

The community aspect of U-P2P was exploited in our Gnutella Adapter by restricting the neighboring peers only to other community members. This means that each community becomes a small Gnutella network and grows as more peers join the community and share data. In small communities, one can still hope to reach all available nodes while avoiding infinite search loops. Also, since search queries are not routed through non-participating nodes, one can expect a drastic reduction in number of messages and therefore better scalability of the network as a whole. The grouping of peers into communities does not partition the network (as intentionally done in some P2P systems), because peers can be members of multiple communities each having their own overlay on the network of U-P2P peers. Figure 3 shows an example of the community effect on two communities using the Gnutella adapter. The peer in the overlap, Peer A is a member of both communities but does not bridge the networks or relay any messages across community boundaries. Finally, one could mix and match communities with different Network Adapters. For example, the Community community could use a Napster Adapter to help with bootstrapping.

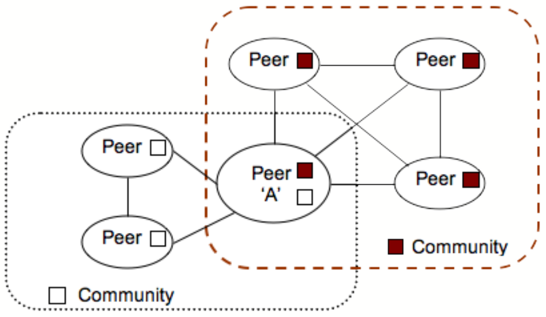

Figure 3: Two overlapping communities in U-P2P using the Gnutella Network Adapter

Since the Gnutella adapter is associated with a community and only connects to peers in the same community, the mechanism for bootstrapping the adapter is different from Gnutella peers on a public network. A community that uses the adapter will have its own cache file containing an initial list of hosts to which they should attempt a connection. Upon starting U-P2P this community-specific cache file is read and connections are opened up to other peers in the community. To keep the list of cached hosts fresh, the host cache file is periodically refreshed with the current known hosts by the adapter along with the addresses of all its currently connected neighbours. This host cache file is an attachment to the community and therefore will be downloaded by any peer that joins the community. If a peer cannot connect to any hosts in their community host cache file then leaving the community and searching for it again in the Root Community may successfully find new community members. The community definition along with the host cache could then be downloaded again and the new host cache file used to connect to known members of the community. This system of host caching is a side effect of using a fully-distributed peer-to-peer protocol with no centralized list of users or a permanent known host.

3.4 Proxy Peer

U-P2P is designed with a “Repository” interface to the backend database it uses for storing and indexing shared XML resources. This interface contains methods for searching the database, creating and deleting communities, and storing XML resources when they are published to the network. Using U-P2P as a proxy required replacing the module implementing the Repository interface with a module that connects to the live external database. The implementation of the Repository is required to open and maintain connections to the live database, process U-P2P search queries (that are given as XPath value pairs) and translate responses from the database into the XML format used by the current community. The implementation may disregard requests to certain methods of the Repository interface, if for example it does not have the right to remove entries from the external database. This is important in order to allow a proxy to operate on behalf of a database that remains unmodified or even unaware of its participation in a data integration system. The proxy is free to implement any or all of the Repository interface methods and may use any protocols or libraries needed to contact the external database.

4 Case Study: Digital Library

A candidate for peer-to-peer data integration is in digital libraries or repositories. With the lower cost of online storage and the availability of metadata protocols, institutions in both the public and private sectors are taking a more active role in the preservation of digital work products [16]. This has led to the development of digital repositories that store and index electronic documents in a permanent archive for preservation and dissemination with generally open but user-based access. By their nature, archives gather data into one place to facilitate access control and maintenance. This leads to each institution having its own digital archive running repository software that may be custom-built, commercially purchased or an open-source package. With many different repositories all running their own software or even different versions of the same software, searches spanning across archives are not possible without custom applications that glue the archives together. The effectiveness of the U-P2P framework and its extensions for integrating distributed data are hereby illustrated by our deployment of a simple yet realistic heterogeneous digital library scenario.

The Open Archives Initiative (OAI) is an organization that promotes standards for the efficient dissemination of content [3]. The OAI-PMH is a protocol that allows software scripts to harvest metadata from OAI-compatible archives. Any types of XML metadata can be harvested, but repositories typically only support terms from the Dublin Core Metadata Initiative [5] such as title, creator and date. Typically the harvested metadata is assembled into a larger collection and indexed by a search engine that is made accessible to the public (e.g. OAIster [2]). Such aggregators of OAI metadata provide gateways to several collections but suffer from constantly having to harvest their metadata records to keep them up to date. They also cannot take advantage of any metadata not supported by the harvester, such as non-DCMI metadata and they usually only target the largest OAI-compatible archives for harvesting. While the use of OAI-PMH in harvesting metadata from large repositories is proving useful in aggregators, it only scratches the surface of data integration and is not suited to a dynamic environment of numerous changing data sources with rich semantic metadata.

To demonstrate the flexibility of U-P2P in its capacity for data integration, a deployment is presented below to show how two digital repositories are used and integrated in U-P2P. The two digital repositories used in this case study, DSpace [1] and Fedora [21] each have their own community in U-P2P. The DSpace and Fedora communities are representative of how information from these digital repositories can be shared on a peer-to-peer network and integrated using bridges.

For this case study, four peers are deployed that span two types of networks and four communities. In Figure 4 Peer A and Peer C are members of the DSpace and Fedora communities, as well as the Bridge and Community community (the Community community is not represented in the figure for simplicity sake. All peers are members of that community by default). Peer B is a member of the Fedora Community and is a proxy for a Fedora database. Peer D is simply a member of the Bridge community. The Fedora and the Bridge communities use the Gnutella peer-to-peer protocol implemented in the GnutellaAdapter; the DSpace community and the Community community use the generic centralized (Napster-style) adapter. Peer C is the host of the central registry for the DSpace community, while peer D plays that role for the Community community.

If Peer A or Peer C performs a search in the Fedora Community, it will dispatch a Gnutella query message to its neighbors. Peer B is a proxy peer for a Fedora database and will translate the query into the Fedora format and query the database directly. It will then form a query response message and return it to the requesting peer.

A search in the DSpace community is sent from Peer A or Peer C to the centralized server that holds an index of all shared resources, in this case, Peer C. The server will respond to the query and direct the peer to the appropriate download location. Usually the topology of the centralized peer-to-peer model does not require Peer A to have any knowledge of Peer C, it is only because Peer C happens to be hosting the registry that A knows C.

Now suppose Peer A wishes to take a search performed in the DSpace Community and execute it in the Fedora Community. To perform an integrated search across the communities, a bridge is required to provide a mapping to translate the query. If we assume such a bridge exists and is shared in the Bridge Community, then Peer A needs to search for and download the bridge before a translated search can take place. Peer A finds and downloads the bridge (which was published by Peer D) and performs a search in the DSpace Community. On the results page for the query, a link is displayed allowing the user to run the search in the Fedora Community. By following this link, query translation is triggered and the translated query is run in the Fedora Community. By running the query in the Fedora Community, Peer A dispatches a search request to Peers B and C using the Gnutella protocol. The proxy Peer B would in turn query its Fedora database and return any matching results. The responses from Peers B and C are displayed to the user in the Fedora Community and the user has effectively changed communities without entering a new query that uses the Fedora Community schema. Thus the query translation saves the user from manually translating searches in communities that are related through bridges.

We outlined how the flexibility of U-P2P in connecting to multiple peer-to-peer networks in different configurations allows bridges to be built across networks of peers that may include proxies to live databases. The following subsections describe the details of the case study.

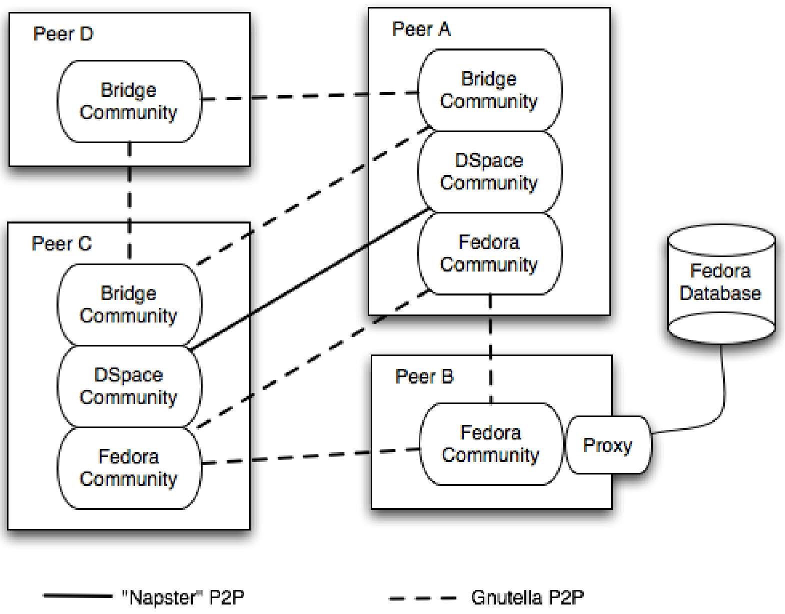

Figure 4: Case Study Peer Deployment

4.1 Emulating DSpace

DSpace is a system developed by the MIT Libraries and Hewlett-Packard Labs, and its main focus is storing digital output created by faculty and staff of a research institution [1]. DSpace was designed as a standalone archive of an institutions digital work product, and as such it does not provide remote access to the search facilities of DSpace other than through the website of the repository host. Other than OAI harvesting, DSpace has no externally visible APIs for accessing its search facilities or specifically for integration. While exporting records to XML could be useful in integration, it can only be performed by the administrators of the database and is intended for backing up data. Hence to integrate DSpace with another repository we decided to emulate DSpace with U-P2P. This will allow DSpace records (and their bitstreams) to be imported into U-P2P as XML metadata and attachments.

The following steps need to be performed to create a DSpace U-P2P community that emulates the storage, retrieval and searching capabilities of DSpace:

1. Create a DSpace schema that will store the metadata and Bitstream information used in DSpace. The XML encodings for Dublin Core elements suggested in [18] were used when possible and additional elements for DSpace data were added when needed. DSpace uses some non-standard qualifiers for DC terms and these had to be added to the schema. A section of the schema defines a place for the Bitstreams to be listed as attachments in U-P2P with their corresponding descriptions, sizes and dates.

2. Create the DSpace XSLT display stylesheet to display resources in full and simple record modes (as seen in the DSpace interface).

3. Create the DSpace community HTML Create page for U-P2P.

4. Create the DSpace community HTML Search page for U-P2P.

5. Copy the DSpace CSS stylesheets and adjust them to suit the View, Search and

Create pages.

6. Create the DSpace community schema file by describing the community and listing the above files.

7. Start up U-P2P and publish the newly created DSpace community.

4.2 Proxying Fedora

Fedora is a digital repository framework developed jointly by the University of Virginia and Cornell University [21]. It stores digital objects and defines mechanisms to disseminate the digital objects using various services for rendering the content. The Fedora Access API (API-A) provides search and retrieve methods for objects and disseminators available in the repository. A user can search for an object using Dublin Core metadata that is associated with each object.

To share Fedora objects in U-P2P, a schema was created that includes the system and descriptive metadata associated with a Fedora data object and links to the datastreams associated with an object. Although Fedora uses METS, an XML formatted language, the access APIs for Fedora do not return METS records so a new schema was written that includes most of the METS record contents. Create, Search and View pages were also created for a Fedora community in U-P2P. Since Fedora does not have a web-based interface other than through Web Services and the HTTP access methods, the stylesheets for displaying resources did not have to follow any particular guideline; this fact made the task much simpler than creating the stylesheets to emulate DSpace.

A proxy peer for Fedora was created to allow a peer to provide access to legacy Fedora data from U-P2P. The Repository interface was implemented to provide a read-only search and retrieval of objects in a Fedora repository. By only implementing the search methods and not the store or remove methods, the Fedora repository remains completely unchanged and is integrated with U-P2P. Searching in Fedora is performed by submitting the search terms to an URL on the Fedora server (`http://host:port/fedora/search`). The query terms are added to the URL and may query individual supported fields (i.e. Dublin Core attributes) or they can match any field. After submitting the search to the Fedora server, the response is received in XML format. The following is an example of the response to a query:

```
<result>
  <resultList>
    <objectFields>
        <pid>demo:5</pid>
        <title>My Title</title>
    </objectFields>
    <objectFields>
        <pid>demo:3</pid>
        <title>Another Item</title>
    </objectFields>
    ...
  </resultList>
</result>
```

The resulting list of objectFields are parsed and converted into U-P2P search responses and made available on the results page to which the user is directed. The same procedure is followed if a search is either submitted by another peer or if a user is searching locally on the proxy peer. The next step, should a user choose to retrieve one of the search results, is to download the resource from Fedora via the proxy peer.

When downloading a resource from Fedora, three separate URLs are used: one for the Object Profile (metadata about the document), another for the Item Index and another for retrieving attachments. After retrieving all three parts, the record is scanned for attachment URLs (particularly in the Item Index) which are replaced with URLs that point to the proxy peer. This allows the Fedora database to remain anonymous behind the proxy and not receive any requests directly from the peer-to-peer network. A complete XML resource is returned to the user and the user may subsequently request attachments associated with the item. Attachments are routed using a simple pipe that takes binary bytes from the attachment on the Fedora server and returns them to the user.

With the U-P2P interfaces implemented as such, a Fedora proxy can be set up to bring any Fedora repository into the U-P2P network. No further effort is required to change the data on the Fedora server and only the proxy peers needs read access to the database HTTP interface.

4.3 Bridging the Dspace and Fedora Communities

The remaining step was to create a bridge between the two communities and provide mappings between the DSpace and Fedora communities. The following is the bridge between the DSpace and Fedora communities:

```
<bridge
  xmlns:owl="www.w3.org/2002/owl#"
  xmlns:rdf="www.w3.org/1999/22-rdf-syntax-ns#"
  xmlns:rdfs="www.w3.org/2000/rdf-schema#">
  <title>DSpace to Fedora bridge</title>
  <description>Relates items in DSpace
     to items in Fedora</description>
  <comments>The mapping uses a simple
    translation of XPaths.</comments>
  <bridgeMapping>
    <source>
      <community>d2a9d6...</community>
      <resource>134d8f...</resource>
    </source>
    <relation>owl:sameAs</relation>
    <target>
      <community>d2a9d6...</community>
      <resource>da1058...</resource>
    </target>
  </bridgeMapping>
  <transformList>
    <transform>
      <file> file://community/BridgeCommunity/
      DSpaceFedoraXPathMappings.xml</file>
      <type>XPathMap</type>
      <description>XPath mapping for
       schema elements.</description>
    </transform>
    <transform>
    <file>file://community/BridgeCommunity/
      DSpaceFedoraMapping.owl</file>
      <type>OWL</type>
      <description>OWL mapping from DSpace
        to Fedora</description>
    </transform>
  </transformList>
</bridge>
```

The XPath mappings used to translate DSpace queries to Fedora queries are simple equivalence mappings show below:

```
<XPathMappings>
  <mapping source="dspace/item/metadata/dc:title"
    target="/fedoraItem/dublinCoreRecord/dc:title" />
  <mapping source="dspace/item/metadata/dc:creator"
    target="/fedoraItem/dublinCoreRecord/dc:creator" />
  <mapping source="dspace/item/metadata/dc:subject"
    target="/fedoraItem/dublinCoreRecord/dc:subject" />
  <mapping source="dspace/item/metadata/dc:description"
    target="/fedoraItem/dublinCoreRecord/dc:description" />
   <mapping source="dspace/item/metadata/dc:identifier"
    target="/fedoraItem/dublinCoreRecord/dc:identifier" />
  ...
</XPathMappings>
```

These mappings are used as described in the previous section to translate a query from DSpace to Fedora. When a query is translated, a mapping is required for each XPath that is to be mapped to the target schema. Since Fedora only supports fifteen Dublin Core attributes (the core set with no qualifiers), any additional attributes used in DSpace are not translated and are omitted from the resulting query. This form of query translation is a simplified view of potentially very complex mappings between schema elements. For example, a more complex mapping might want to perform queries over the input data or manipulate the input in order to clean and sort the data. Additional mappings for use in translating instance data were not implemented for the case study. Therefore a user can execute their query in the community that is a target of a bridge (e.g. a DSpace query executed in a Fedora community), but they can only view and download the results in the target community. This effectively points the user toward new sources of data without involving any complex automated searches for bridges or exploration of bridge relationships. This kind of automation and semantic-level interpretation are left for future work.

5 Conclusion and Future Work

This work showed how an existing peer-to-peer file-sharing framework could be used and extended to allow data integration. This is done through:

- the creation and sharing of bridges, which describe the means to transform a piece of data from one source into another;

- the use of a fully distributed peer-to-peer protocol such as Gnutella to avoid having the data sources to advertise all of their content to a registry. Also, the restriction of the propagation of routing messages to each community allows for better scalability of Gnutella;

- the creation of proxy peers that query legacy data sources and propagate search queries to other peers without the source being aware.

A case study involving two Digital Library database formats was implemented to validate and demonstrate our ideas. One of them, DSpace, was directly emulated by U-P2P, while the other one, Fedora, was proxied and accessed remotely. The libraries used the same metadata but in a different structure and with differing levels of detail. This required a one-to-one mapping between compatible metadata to be created and allowed simple XPath translation to be used between two communities in U-P2P.

Future work will investigate further automation of search using bridges. For example, the querying peer could be set to automatically become a member of related communities (via the bridges), and recursively look for related communities and relevant documents.

We are also studying bridge types other than "same-as". Communities and resources could indeed be connected through other meaningful relationships which could be exploited if U-P2P can be made aware of their semantics. For example, it would be interesting to evaluate the use of a "deprecates" relationship between documents, to help with tracking down the most up to date version.

References

[1] DSpace digital repository, MIT and Hewlett-Packard (HP). Retrieved on November 20, 2005 from `http://dspace.org/introduction/index.html`.

[2] OAIster, University of Michigan Digital Library Production Service. . Retrieved on November 20, 2005 from `http://oaister.umdl.umich.edu/o/oaister/`.

[3] Open Archives Initiative (OAI), Mission Statement. Retrieved on November 20, 2005 from `http://www.openarchives.org/organization/index.html`.

[4] XML Path Language. In DeRose S. Clark, J., editor, *Version 1.0, W3C Recommendation.* , Retrieved on November 20, 2005 from `http://www.w3.org/TR/1999/REC-xpath-19991116`, November 1999.

[5] DCMI Usage Board, DCMI Metadata Terms. In *Dublin Core Metadata Initiative*, Retrieved on November 20, 2005 from `http://dublincore.org/documents/2005/01/10/dcmi-terms/`, January 2005.

[6] Karl Aberer, Vana Kalogeraki, and Manolis Koubarakis, editors. *Databases, Information Systems, and Peer-to-Peer Computing, First International Workshop, DBISP2P, Berlin Germany, September 7-8, 2003, Revised Papers*, volume 2944 of *Lecture Notes in Computer Science*. Springer, 2004.

[7] Neal Arthorne, Babak Esfandiari, and Aloke Mukherjee. U-P2P: A Peer-to-Peer Framework for Universal Resource Sharing and Discovery. In *USENIX Annual Technical Conference, FREENIX Track*, pages 29–38, 2003.

[8] P. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. Data management for peer-to-peer computing: A vision. In *Workshop on the Web and Databases, WebDB 2002.*, 2002.

[9] Diego Calvanese, Elio Damaggio, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Semantic data integration in p2p systems. In Aberer et al. [6], pages 77–90.

[10] Enrico Franconi, Gabriel Kuper, Andrei Lopatenko, and Ilya Zaihrayeu. The codb robust peer to peer database system. In Karl Aberer, Stefan Decker, David De Roure, Carole Goble, and Hongsuda Tangmunarunkit, editors, *Proc. of The Second Workshop on Semantics in Peer-to-Peer and Grid Computing, collocated with the*

Thirteenth International World Wide Web Conference, May 2004.

[11] Enrico Franconi, Gabriel M. Kuper, Andrei Lopatenko, and Luciano Serafini. A robust logical and computational characterisation of peer-to-peer database systems. In Aberer et al. [6], pages 64–76.

[12] Hector Garcia-Molina, Yannis Papakonstantinou, Dallan Quass, Anand Rajaraman, Yehoshua Sagiv, Jeffrey D. Ullman, Vasilis Vassalos, and Jennifer Widom. The TSIMMIS Approach to Mediation: Data Models and Languages. *J. Intell. Inf. Syst.*, 8(2):117–132, 1997.

[13] Alon Y. Halevy, Zachary G. Ives, Dan Suciu, and Igor Tatarinov. Schema mediation in peer data management systems. In *Proceedings of the 19th International Conference on Data Engineering (ICDE 2003)*, pages 505–516, 2003.

[14] J. Heflin. OWL Web Ontology Language Use Cases and Requirements. In *W3C Recommendation, 14 February 2004. Retrieved on November 20, 2005*, `http://www.w3.org/TR/2004/REC-webont-req-20040210/`.

[15] Maurizio Lenzerini. Data Integration: A Theoretical Perspective. In *Proceedings of the 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2002)*, pages 233–246, Madison, Wisconsin, June 2002. ACM.

[16] C. A. Lynch. Institutional Repositories: Essential Infrastructure for Scholarship in the Digital Age. In *Association of Research Libraries (ARL) Bimonthly Report 226.* , Retrieved on November 20, 2005 from `http://www.arl.org/newsltr/226/ir.html`, February 2003.

[17] Aloke Mukherjee, Babak Esfandiari, and Neal Arthorne. U-P2P: A Peer-to-Peer System for Description and Discovery of Resource-Sharing Communities. In *RESH - Workshop on Resource Sharing in Massively Distributed Systems, in ICDCSW '02: Proceedings of the 22nd International Conference on Distributed Computing Systems*, pages 701–705, Washington, DC, USA, 2002. IEEE Computer Society.

[18] Johnston P. Powell, A. Guidelines for Implementing Dublin Core in XML. In *Recommendation, Dublin Core Metadata Initiative*, Retrieved on November 20, 2005 from `http://www.dublincore.org/documents/2003/04/02/dc-xml-guidelines/`, April 2003.

[19] A. Sheth. *Interoperating Geographic Information Systems*, chapter Changing Focus on Interoperability in Information Systems: from System, Syntax, Structure to Semantics, pages 5–29. Goodchild M.F., Egenhofer, M.J., Fegeas, R., Koffman, C.A. (eds.)., Kluwer Academic Publishers, Boston, 1999.

[20] Amit P. Sheth and James A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Comput. Surv.*, 22(3):183–236, 1990.

[21] Wayland R. Payette S. Staples, T. The Fedora Project: An Open-source Digital Object Repository System. In *D-Lib Magazine*, Retrieved on November 20, 2005 from `http://www.dlib.org/dlib/april03/staples/04staples.html`, April 2003.