Winter 2002

Federating Heterogeneous Digital Libraries by Metadata Harvesting

Xiaoming Liu
Old Dominion University

FEDERATING HETEROGENEOUS DIGITAL LIBRARIES BY METADATA HARVESTING

by

Xiaoming Liu
B.S. July 1994, ShanDong University
M.S. March 1997, Shanghai Jiaotong University

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirement for the Degree of

DOCTOR OF PHILOSOPHY

COMPUTER SCIENCE

OLD DOMINION UNIVERSITY
December 2002

Approved by:

Kurt Maly (Co-Director)

Mohammad Zubair (Co-Director)

Frank C. Thames

Chris Wild

Steven J. Zeil

ABSTRACT

FEDERATING HETEROGENEOUS DIGITAL LIBRARIES BY METADATA HARVESTING

Xiaoming Liu
Old Dominion University, 2002
Co-Director: Dr. Kurt Maly
Co-Director: Dr. Mohammad Zubair

This dissertation studies the challenges and issues faced in federating heterogeneous digital libraries (DLs) by metadata harvesting. The objective of federation is to provide high-level services (e.g. transparent search across all DLs) on the collective metadata from different digital libraries. There are two main approaches to federate DLs: distributed searching approach and harvesting approach. As the distributed searching approach replies on executing queries to digital libraries in real time, it has problems with scalability. The difficulty of creating a distributed searching service for a large federation is the motivation behind Open Archives Initiatives Protocols for Metadata Harvesting (OAI-PMH). OAI-PMH supports both data providers (repositories, archives) and service providers. Service providers develop value-added services based on the information collected from data providers. Data providers are simply collections of harvestable metadata. This dissertation examines the application of the metadata harvesting approach in DL federations. It addresses the following problems:

- Whether or not metadata harvesting provides a realistic and scalable solution for DL federation.
- What is the status of and problems with current data provider implementations, and how to solve these problems.
- How to synchronize data providers and service providers.
- How to build different types of federation services over harvested metadata.
- How to create a scalable and reliable infrastructure to support federation services.

The work done in this dissertation is based on OAI-PMH, and the results have influenced the evolution of OAI-PMH. However, the results are not limited to the scope of OAI-PMH. Our approach is to design and build key services for metadata harvesting and to deploy them on the Web. Implementing a publicly available service allows us to demonstrate how these approaches are practical. The problems posed above are evaluated by performing experiments over these services.

To summarize the results of this thesis, we conclude that the metadata harvesting approach is a realistic and scalable approach to federate heterogeneous DLs. We present two models of building federation services: a centralized model and a replicated model. Our experiments also demonstrate that the repository synchronization problem can be addressed by push, pull, and hybrid push/pull models; each model has its strengths and weaknesses and fits a specific scenario. Finally, we present a scalable and reliable infrastructure to support the applications of metadata harvesting.

ACKNOWLEDGMENTS

This thesis would not be possible were it not for the love and support of a great many people. First, I would like to thank my co-advisors, Kurt Maly and Mohammad Zubair. Through their support, care, and patience, they have transformed a struggling graduate student into an experienced researcher. Their insight and ideas formed the foundation of this dissertation as much as mine did, and their guidance and care helped me get over various hurdles during my graduate years. I would also like to thank Michael Nelson, who gave me many valuable insights into the world of digital libraries. I would also like to thank Frank C. Thames, Chris Wild, and Steven Zeil for being the readers of my thesis. After reading a draft, they provided helpful comments on various aspects of my thesis. These comments made my thesis much better, and I learned a lot from their comments.

I would also like to thank other members of the digital library research group, including Rong Shi, Yang Zhao, Jianfeng Tang, Shanmuganand Naidu, Hesham Anan, Mohamed Kholief, Rong Tang, Dun Tan, Satish Kumar. They have made design or coding suggestions, developed supporting technologies, and helped tremendously in keeping our system running smoothly.

A number of people outside Old Dominion University played significant roles in supporting my work. Among these people are: Rick Luce and Herbert Van de Sompel (Los Alamos National Laboratory), Stevan Harnad and Tim Brody (Southampton University).

Finally, I would like to thank my wife, Susu Shi, and my parents for all their love and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

With the recent growth of the Internet and the World Wide Web, there has been an exponential growth of online resources [1]. Some examples of online resources are: pre-prints, including technical reports; tutorials, posters, and demonstrations from conferences; student projects; theses and dissertations; and working papers. However, a variety of obstacles (such as dispersion over the Web and lack of metadata) hamper discovery of such materials and hinder their widespread use. Digital Libraries provide an efficient means of managing, discovering, and distributing digital information [5, 63]. Consequently, a number of very good digital libraries, as described in NSDL (National Science, mathematics, engineering, and technology Digital Library) and DLI-2 (Digital Libraries Initiative phase 2)[62, 56, 43], have been built to address the need for information management and distribution. However, most of these libraries have been built in isolation utilizing different technologies, protocols, and metadata in terms of both syntax and semantics. These differences hinder the development of digital library services that will enable users to discover information from multiple libraries through a single unified interface. In the article of Paepcke, the objective of digital library interoperability is:

> Interoperability is a central concern when building digital libraries as collections of independently developed components that rely on each other to accomplish larger tasks. The ultimate goal for such systems is for the components to evolve independently yet be able to call on one another efficiently and conveniently. [106]

As identified by the NSDL community:

> Interoperability requires cooperation at three levels: technical, content, and organizational. Technical agreements cover formats, protocols, and security systems so that messages can be exchanged, etc. Content agreements cover the data and metadata, and include semantic agreements on the interpretation of the messages. Organizational agreements cover the

[1] This dissertation follows the style of *The Physical Review*

> ground rules for access, for changing collections and services, payment, authentication, etc. [98]

From the technical point of view, there are basically two ways to implement digital services across heterogeneous digital libraries: a distributed searching approach and a harvesting approach [57, 115]. The former would require archives to implement a joint protocol to handle queries and other services over distributed libraries in real time. Such an approach has important problems of scalability, in view of the possible emergence of thousands of institutional and/or subject-oriented archives worldwide [44, 45].

The difficulty of creating a large distributed searching service is the motivation behind recent efforts to build federation services based on the concept of metadata harvesting. The key idea here is to harvest metadata from different collections at one location, and provide high-level services on the collected metadata set. One result of these efforts has been the Universal Preprint Service (UPS) Prototype [126], which was developed as a proof-of-concept of a multi-discipline digital library of publicly available scholarly material. The Prototype harvested nearly 200,000 records from several different archives and created an attractive end-user environment. The UPS service is partially based on the SODA [91], NCSTRL+ [93] and Buckets concept [90, 81] introduced at Old Dominion University, and the RePEc model introduced by Krichel [23]. The motivation of this thesis came from the promises of the UPS service.

Experience developed during the creation of the UPS prototype was one of the foundations of the concepts brought forward in the Santa Fe Convention [128], and, lately, the OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting) [57]. The objective of the OAI-PMH is to develop a framework to facilitate the discovery of content stored in distributed archives. OAI-PMH is becoming widely accepted, and many archives are currently or soon-to-be OAI-PMH-compliant. OAI-PMH 1.0 was released in 2000 [127], and after worldwide experiments, OAI-PMH 2.0 was released in 2002 as a stable specification [58, 129].

While the OAI-PMH solves one very important set of problems, there are many open questions not well understood and which require considerable research and experimentation to allow the development of a body of design knowledge and community practice. For this thesis, we are especially interested in usability, scalability, and practical issues involved in implementing a large scale of metadata harvesting.

Our work focuses on more general views of metadata harvesting instead of protocol details of OAI-PMH. Our work has influenced the evolution of OAI-PMH, and is based on the large base of existing OAI-PMH compliant repositories.

1.2 OBJECTIVE

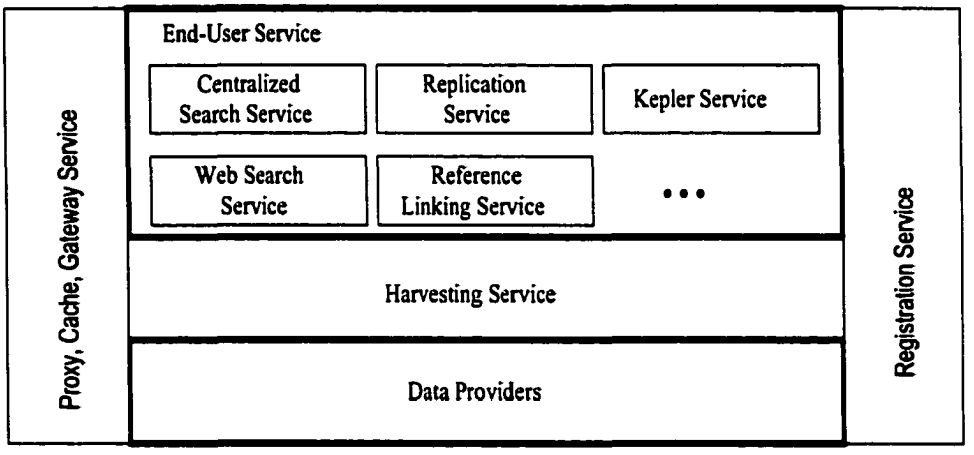

FIG. 1.1: Architecture of a metadata harvesting system

The objective of this thesis is to demonstrate the feasibility of the harvesting approach for federating digital libraries. To achieve this objective, we develop, deploy, and evaluate data providers and key services on collective metadata in real environment. The interactions between data providers and services are illustrated in Figure 1.1.

Data Provider The data provider maintains one repository for metadata harvesting. We study a series of performance criteria for data providers, including server availability, reliability, metadata variability, and update frequency. These criteria influences the implementation of harvesting and other services.

Harvesting Service The harvesting service is the key service, which maintains data coherency between data providers and service providers. The harvesting service should maintain data freshness (that is keep the harvested metadata in synchronization with data providers), and at the same time it needs to minimize the impact on data providers. In our work, we study methods of reaching better freshness with limited resources.

End-User Service Above the harvester, various services, such as centralized federation services, replication services, and citation linking services, can be built for end-users. The centralized federation service harvests metadata to a central database and provides a unified interface to search all the collections. The replication service can be viewed as mirrored OAI-PMH-compliant repositories, where every participant has its own user interface providing federation service over harvested metadata.

Registration Service The OAI-PMH raises the "awareness" question, namely how service providers can find out the existence of data providers, and how data providers can find out the appropriate service providers to register. In addition, different data/service providers can be aware of and linked to each other by using OAI-PMH unique identifiers. A distributed service model could be accomplished by sharing different services.

Proxy, Cache, and Gateway Service The proxy, cache, and gateway services optimize the functioning of the model underlying the OAI-PMH. They provide an infrastructure that can be used by all other components to achieve interoperability, scalability and reliability. Various applications and services can exploit the services included in this infrastructure. An OAI-PMH proxy dynamically forwards OAI requests to data providers. For example, it can dynamically fix common XML encoding errors and translate between different OAI-PMH versions. An OAI-PMH cache caches metadata and can filter and refine them before exposing them to service providers. It also serves as a simple cache that reduces the load on source data providers and improves server availability. An OAI-PMH gateway can convert the OAI-PMH to other protocols and applications. For example, the gateway could convert between different protocols and OAI-PMH.

In Figure 1.1 the bold boxes represent the original OAI framework that partitions the world into data providers and service providers. In our view, we layer the service providers by providing a fundamental service of harvesting that will provide a clean, validated set of metadata to all other services. The vertical boxes represent services that cut across the layers of service providers and data providers and are accessed by any one of them.

1.3 APPROACH AND ISSUES

Our approach is to design and build several services and to deploy them on the Web. Implementing and observing a publicly available service allows us to demonstrate that our approach is practical since we have no control over the user community. By performing various experiments, we are able to verify the concept of metadata harvesting, answer open questions, and demonstrate the feasibility of our approach.

The feasibility of metadata harvesting for DL federation This problem is demonstrated in the Arc federation search service, the first OAI-PMH compliant service provider [71, 69, 70]. Arc has been operational as a public search service at Old Dominion University since October, 2000. It has harvested more than 1M records from over 100 OAI-PMH-compliant repositories and has heavily influenced the model for building an OAI-PMH service provider. Lately, the technology of Arc has been used in local projects as well as outside researchers, for example, NCSTRL [2], Archon [82], metaArchive [86], and the OLAC project [101].

The architecture of building federation service We develop and analyze both a centralized approach and a replicated approach. The centralized approach is demonstrated by Arc; the replicated approach is demonstrated by the TRI (Technical Report Interchange) project among four national laboratories that allows them to share technical report collection through the native DL interfaces [73]. We compare the efficiency of the two approaches.

The quality of data providers and metrics Through the running of Arc over a large number of data providers and the experience of implementing OAI-PMH data providers, we are able to define several metrics of quality of data providers and measure data providers against these qualities.

Freshness and repository synchronization The OAI-PMH is based on a service provider "pulling" model. However, there are additional approaches such as "push" and "push/pull" hybrid model. The "pulling" model itself can also be improved by supplying parameters in the data provider side to notify the service provider of its update frequency. For this thesis, we implemented the Kepler service – an OAI compliant data provider for individual publishers [80, 68]. It is

a broker-based, peer-to-peer network that focuses on repository synchronization problems and a registration service.

Service building The usefulness of the metadata harvesting approach is based on the quality of metadata and how these metadata from heterogeneous sources can be built into a unified search interface. We demonstrate our approach of building rich cross archive search and linking service in Arc, TRI, and Archon [82].

Scalability and Reliability We propose and implement a series of software components to build a scalable and reliable infrastructure for metadata harvesting applications [67]. In cooperation with Southampton University, several services, such as OAI aggregator [15], DP9 [72], and OAI proxy, have been deployed [67].

1.4 ORGANIZATION OF DISSERTATION

In this dissertation, we present the challenges and issues that we encountered during the design, development, and implementation of these systems and then describe our experimental solutions that address the challenges. To that end, the rest of this dissertation is organized as follows:

Chapter 2: Background We start by discussing various DL federation techniques. There are basically two ways to implement these: a distributed searching approach and a harvesting approach. We present both approaches and typical systems. We also highlight the recent efforts of OAI-PMH and its relationship with our work.

Chapter 3: Metadata Harvesting System Architecture Chapter 3 presents an architecture of a metadata harvesting system. This chapter is a generalization that presents the three major components in a metadata harvesting system: the data provider, the harvester, and the service provider. We also present a survey of current data provider implementations and their problems, including server availability and metadata variability.

Chapter 4: Repository Synchronization In Chapter 4, we investigate the update frequency of typical OAI-PMH-compliant repositories. A series of metrics are proposed to measure the update frequency of data providers and the

freshness of harvesters. An algorithm is presented to optimize the repository synchronization.

Chapter 5: Centralized Federation Service Chapter 5 discusses the approach to implementing federation through a centralized approach. We then address how we built a unified search interface over heterogeneous repositories with a user-focused approach.

Chapter 6: Replicated Federation Service The metadata harvesting system described in Chapter 3 is flexible enough to build federation service based on a replicated approach as well. This approach can be viewed as mirrored OAI repositories, where every digital library has its own federation service. The consistency between these services is maintained using OAI-PMH. In addition, this approach supports several levels of redundancy, thereby improving the availability of the whole system.

Chapter 7: Kepler Service The Kepler service supports the concept of an archivelet, which is a self-contained, self-installing software package that easily allows a researcher to create and maintain a small, OAI-PMH-compliant archive. The Kepler service poses a series of new challenges to metadata harvesting. The OAI-PMH is insufficient in such a scenario, and we extend the harvest model to a "push" and hybrid "push/pull" model to support the dynamic application scenario of Kepler.

Chapter 8: A Scalable Architecture for Metadata Harvesting Chapter 8 discusses the requirements of current and emerging applications based on metadata harvesting and emphasizes the need for a common infrastructure to support them. Inspired by HTTP proxy, cache, gateway and web service concepts, a design for a scalable and reliable infrastructure that aims at satisfying these requirements is presented. Moreover, it is shown how various applications can exploit the services included in the proposed infrastructure.

Chapter 9: Conclusion and Future work Finally, Chapter 9 summarizes the results presented and provides suggested directions for future work.

CHAPTER 2

BACKGROUND

The DL federation problem is the problem of building a coherent set of digital library services that will enable users to discover information from multiple libraries through a single, unified interface. This chapter highlights previous work done to address the DL federation problem. This chapter is organized as follows:

- We begin this chapter with a definition of DL federation problem.
- In Section 2.2, we discuss the distributed search approach to addressing the DL federation problem and highlight a selection of the key works.
- Next, in Section 2.3, we describe the harvesting approach to addressing the DL federation problem.
- Finally, we introduce the OAI-PMH approach in Section 2.4.

2.1 THE DL FEDERATION PROBLEM

Most digital libraries have been built in isolation utilizing different technologies, protocols, and metadata in terms of both syntax and semantics. This situation hinders interoperability, which is essential for building a coherent set of digital library services that will enable users to discover information from multiple libraries through a single unified interface [106]. There are basically two ways to implement DL federation: a distributed searching approach and a harvesting approach. The distributed searching service, or metasearching service, is a service that provides unified query interfaces to multiple search engines. It requires each search engine to implement a joint distributed search protocol; moreover, as it needs post-process search results in real time, it has important problems of scalability [115, 57]. A distributed searching system may provide other services based on real-time processing of query results from participating search engines as well.

A harvesting approach collects data from heterogeneous sources in advance, therefore, it is more realistic in dealing with large number of digital libraries. Harvesting approaches have the additional attractive property that they allow data enhancing procedures to be run on the collected data. Enhancements such as normalization,

augmentation and restructuring are applied to data originating from different sources in order to create consistent end-user services. In a harvesting scenario, these activities can be dealt with in a batch manner. The harvesting approach requires a protocol to synchronize database and a mechanism to ensure interoperability of data.

2.2 DISTRIBUTED SEARCHING AND REAL-TIME PROCESSING

In the distributed searching approach of DL interoperability, each archive maintains its own search service; a metasearcher provides unified query interfaces to multiple search engines. Thus, users have the illusion of a single, combined document source. The distributed searching approach has three kinds of typical models: (1) distributed searching based on same software deployed, like Dienst [57]; (2) distributed searching based on protocol agreement between search engines, like the STARTS protocol [41, 83], SDLIP [105], and Z39.50 [135]; (3) distributed searching without individual search engine involvement, like general metacrawler [116] in the Web, the Lyceum project to federate selected digital libraries [76], and InterOp project [117]. The limitations and advantages of these models are described below.

Dienst is a protocol and software for distributed digital libraries developed as part of the Computer Science Technical Reports Project (CSTR), which is the foundation for historical NCSTRL [24, 27], the Networked Computer Science Technical Reference Library. Dienst specifies the operational characteristics of core digital library services and mandates an open extensible protocol for communicating with digital library services and accessing documents [55]. The interoperability of Dienst is implemented by using the same protocol or software suite. The historical NCSTRL encountered a number of technical and social problems, including metadata quality, connectivity, and server quality, as specified by Powell in a study of NCSTRL:

> Reliability of the distributed system is low. ... shows that many servers are highly available, specifically 23 of 38 (61%) are up 90% of the time. ... we see that the system had at least one server failure 100% of the time. ... Our measurements indicate that engineering reliable, distributed digital libraries will be a challenge. A federated system is vulnerable to its weakest component. Strong institutional commitment will be necessary for success. [109]

The historical NCSTRL was developed and maintained by Cornell University from

1994 to 2001. In 2001, NCSTRL moved to an OAI-PMH based architecture mainly due to its scalability problem [2].

Perhaps the most widely known search middleware is the Z39.50 standard. It defines a broad range of facilities, such as a standard machine representation of queries and an extensible collection of document attributes that may be used both in queries and for the retrieval of document fragments. The Z39.50 is a comprehensive, often complex approach and generally does not fit well with light-weight approaches typical in the design of web related protocols. The Simple Digital Library Interoperability Protocol (SDLIP) defines a layered, uniform interface to query and retrieve the results from each searchable collection through a common interface. SDLIP also supports an interface to access source metadata.

The Stanford protocol proposal for Internet retrieval and search (STARTS) is a protocol for Internet retrieval and search that facilitates the task of querying multiple document sources. STARTS is a group effort coordinated by Stanford's Digital Library project and involving over 11 companies and organizations. The goal of STARTS is to facilitate the main three tasks that a meta searcher performs: (1) Choosing the best sources to evaluate a query; (2) evaluating the query at these sources; (3) Merging the query results from these sources. STARTS tries to solve the interoperability problem by reaching a simple but expressive agreement between search engine vendors. However, the details about the workings of most search engines are proprietary, and it is becoming complicated to fully describe any useful search engine. STARTS also has the scalability problem with increased numbers of search engines. Lately SDARTS [42] has been designed to combine STARTS and SDLIP. It can be viewed as an instantiation of SDLIP with metasearch-specific elements from STARTS.

The MetaCrawler [116] is a parallel web search service at the University of Washington and is now part of Go2Net. It provides users with a single interface with which they can query popular general-purpose web search services, and has some sophisticated features that allow results of much higher quality than simply regurgitating the output from each search service. The MetaCrawler provides a single central interface for web document searching. Upon receiving a query, the MetaCrawler posts the query to multiple search services in parallel, collates the returned references, and loads those references to verify their existence and to ensure that they contain relevant information. MetaCrawler's major advantage is that it does not require

individual search engines' involvement. MetaCrawler, however, has the problem of performance. A naive implementation would wait until all search services return results and then wait again until each reference has been downloaded. In addition, MetaCrawler does not cover the source-metadata problem, so its query is sent in parallel to all search engines.

The InterOp project proposes an LFDL (Lightweight Federated Digital Library) system [117]. In LFDL, a universal search interface is defined as the basic interoperation middle layer. A DL definition language is defined to describe the rules of query mapping between universal interface and native interface. This approach has the advantage of supporting a number of heterogeneous digital libraries without prior coordination.

The distributed searching approach without prior coordination is also used by a number of automated brokerage services (e.g. shopping agents) in the Web [31]. The main advantage is that it does not require any changes in individual search engine, however, the format in any participants may change format overnight, and the automatic broker would be confused [10]. Moreover, the distributed searching approach has important problem of scalability because it needs to merge search results in real time.

2.3 HARVESTING

The harvesting approach for accommodating diversity is to collect data from a set of underlying repositories and combine it into a homogeneous whole. The harvesting approach collects data into a centralized collection, and it can pre-build various services (e.g. indexing, normalization), thus it has better scalability. However, it has the problems of repository synchronization and data duplication. The harvesting can be based on structured or non-structured data. In the digital library domain, we are especially interested in structured data such as metadata that exploits the semantics of existing digital resources and potentially provides richer service. The structured data requires agreement algorithms for correlating information. We now highlight several representative techniques that use the general harvesting approach.

A web robot is a program that automatically traverses the Web's hypertext structure by retrieving a document, and recursively retrieving all documents that are referenced. These programs are sometimes called "spiders," "web wanderers," or "web

worms" [53]. Robots can be used to perform a number of useful tasks such as: statistical analysis, maintenance, mirroring, and resource discovery. Although robots for resource discovery are widely used in search engines such as Lycos, Google, and Infoseek, robots have the important problem of updating overhead as identified by Koster:

> There is no efficient change control mechanism in the Web; there is no single request that can determine which of a set of URL's has been removed, moved, or modified. [54]

Again, keeping synchronization between the centralized server and the joined web site is difficult and resource consuming; several research projects have been done to increase the freshness of web search engines by speculating the update frequency of web pages [19, 20, 107].

Robots are also used to build focused digital libraries, such as ResearchIndex (formerly CiteSeer) [61]. ResearchIndex is a scientific literature digital library built by selectively harvesting the Web. ResearchIndex uses web search engines and heuristics to locate good starting points for crawling the Web. ResearchIndex downloads Postscript or PDF files, which are then converted into text. ResearchIndex checks to verify that the document is a research document by testing for the existence of a reference or bibliography section. In the NSDL project, Bergmark [9] uses crawlers to synthesize document collections on various topics in science, mathematics, engineering, and technology; it is based on matching document similarity between harvested data and a pre-defined dictionary for selected subject.

Some projects try to relieve the update overhead problem by introducing additional protocols or software modules. Harvest [12] is a research project at University of Colorado – Boulder, portions of it have found its way into various commercial products, including the Netscape Catalog Server and @Home Network. Harvest focuses on providing a framework for indexing and querying multiple document sources and includes a set of tools for gathering and accessing information on the Internet. The Harvest gatherers collect and extract indexing information from one or more sources. Then, the brokers retrieve this information from one or more gatherers, or from other brokers. The brokers provide a querying interface to the gathered information. The Harvest architecture can reduce both server load and network traffic. However, Harvest is a combination of different tools and is complex, focuses on unstructured web documents, which are different from structured document in DLs,

and has only basic increment update methodology based on file-access time.

The above approaches are useful applications of the harvesting approach; however, they deal with non-structured or semi-structured data. A more advanced service needs richer metadata and since the coordination between data providers and harvesters is minimal (at most times it is just the robots.txt file), it is difficult to implement efficient repository synchronization. These problems inspire the introduction of the emerging standard OAI-PMH in the digital library community.

2.4 OPEN ARCHIVES INITIATIVE PROTOCOL FOR METADATA HARVESTING

The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) presents a technical and organizational metadata harvesting framework designed to facilitate the discovery of content stored in distributed archives. The OAI-PMH is becoming widely accepted and many archives are currently or soon-to-be OAI-PMH-compliant. OAI-PMH 1.0 was released in 2000, after worldwide experiments, OAI-PMH 2.0 was released in 2002 as a stable specification.

The OAI framework is the most important short-range interoperability effort that we are aware of in the DL community. OAI-PMH is based on a model that puts a very clean divide between data-providers (entities which expose metadata) and service-providers (entities which harvest metadata, presumably with the intention of providing some service). OAI-PMH thus defines a protocol to synchronize data providers and service providers by selective harvesting and a mechanism for metadata interoperability and validation. As specified by Lagoze and Van de Sompel:

> The technical framework of the Open Archives Initiative is intended to provide a low-barrier approach to interoperability. Nevertheless, there are functional limitations to such a low-barrier framework and other interoperability standards. For example, Z39.50 addresses a number of issues in a more complete manner. However, as noted by Bill Arms, interoperability strategies generally increase in cost (difficulty of implementation) with an increase in functionality [5]. The OAI technical framework is not intended to replace other approaches but to provide an easy-to-implement and easy-to-deploy alternative for different constituencies or different purposes than those addressed by existing interoperability solutions. [57]

The OAI technical framework addresses two well-known metadata requirements: interoperability and extensibility. The metadata interoperability is addressed by requiring that all OAI data providers supply metadata in a common format called the Dublin Core Metadata Element Set [131]. Community-specific description, or metadata specificity, is addressed in the technical framework by support for parallel metadata sets. The technical framework places no limitations on the nature of such parallel sets, other than that the metadata records be structured as XML [14] documents, which have a corresponding XML schema for validation [123].

OAI-PMH supports the concept of selective harvesting, which makes it possible to specify a subset of records to be harvested. OAI-PMH opted for two relatively simple criteria for selective harvesting: datestamps and sets. Datestamp is defined as the date of creation, deletion, or latest modification of a record, and sets are mechanisms to group records in a repository for the purpose of selective harvesting.

OAI-PMH is based on a pull-only interaction via HTTP [33] using XML. Service providers make requests to data providers; there is no support for data-provider-driven interaction. All requests and replies occur using the HTTP protocol. Requests may be made using either the HTTP GET or POST methods. All successful replies are encoded in XML. OAI-PMH protocol requests are made using one of six verbs: Identify, GetRecord, ListIdentifiers, ListRecords, ListSets, and ListMetadataFormats. Some of these verbs accept or require additional parameters to completely specify the request (Table 2.1). The correctness of the protocol request and response are verified by XML schema.

2.5 DISSCUSION

Clearly, although we are focusing on federation service in digital libraries community, other communities face similar problems. One important initiative is Semantic Web – " a web of data that can be processed directly or indirectly by machines" [10, 11] and underlying RDF (Resource Description Framework) [60]. Intelligent agents can collect machine-readable web data and apply logic to conduct deduction, in which both distributed searching and harvesting will play significant roles.

OAI-PMH is originated from eprints service. There is a broad movement now well established within the scholarly publishing world such as BOAI [102] and Public Library of Science [108], championed by people like Stevan Harnad at the University of Southampton, to enhance public access to scholarly journal articles through the

TABLE 2.1: Six OAI-PMH verbs

Verb	Arguments	Summary
GetRecord	identifier, metadataPrefix	This verb is used to retrieve an individual metadata record from a repository.
Identify		This verb is used to retrieve information about a repository.
ListIdentifiers	from, until, metadataPrefix, set, resumptionToken	This verb is an abbreviated form of ListRecords, retrieving only headers rather than records. Optional arguments permit selective harvesting of headers based on set membership and/or datestamp.
ListMetadataFormats	identifier	This verb is used to retrieve the metadata formats available from a repository. An optional argument restricts the request to the formats available for a specific item.
ListRecords	from, until, metadataPrefix, set, resumptionToken	This verb is used to harvest records from a repository. Optional arguments permit selective harvesting of records based on set membership and/or datestamp.
ListSets	resumptionToken	This verb is used to retrieve the set structure of a repository, useful for selective harvesting.

use of eprint servers [46]. Clifford Lynch pointed out:

> OAI-PMH grew out of an effort to solve some of the problems that were emerging as eprints servers became more widely deployed. However, as work on the protocol advanced it became clear that it provided a very general-purpose mechanism that could address a surprisingly wide range of urgent needs. [74]

The most obvious applications that are enabled by OAI-PMH are repository synchronization and federated search. Besides that, it also focuses attention on a number of other issues, such as registration and metadata schema, that will have to be addressed as applications proliferate. This thesis contributes to the development of OAI-PMH protocol and applications.

CHAPTER 3

METADATA HARVESTING SYSTEM ARCHITECTURE

In the previous chapter, we highlighted the key techniques to solve the digital libraries federation problem. The metadata harvesting approach is one way to implement the digital libraries federation system. This chapter defines architecture of the metadata harvesting system. The remainder of this chapter is organized as follows:

- Section 3.1 presents a layered architecture of the metadata harvesting system and its major components.
- A data provider maintains one repository that supports the OAI-PMH as a means of exposing metadata. In Section 3.2, we summarize the features of OAI-PMH-compliant data providers.
- A harvesting service traverses the data providers automatically and extracts metadata. It exploits the incremental, selective harvesting defined by the OAI-PMH. The harvesting service is presented in Section 3.3.
- A registration service is essential for the metadata harvesting system. We introduce the issues of the registration service in Section 3.4.
- Value-added services such as cross archive searching can be built over harvested metadata. We discuss end-user service in Section 3.5.
- We briefly introduce the proxy, cache, and gateway service to achieve interoperability, scalability, and reliability of services in Section 3.6.
- In Section 3.7, we introduce Arc, the first OAI-PMH service provider. Arc implemented the model defined in this chapter.

3.1 INTRODUCTION

The basic structure of OAI-PMH supports two roles: the service provider and the data provider, which can be seen in Figure 3.1. Data providers administer systems that support the OAI-PMH as a means of exposing metadata; and service providers use metadata harvested via the OAI-PMH as a basis for building value-added services.

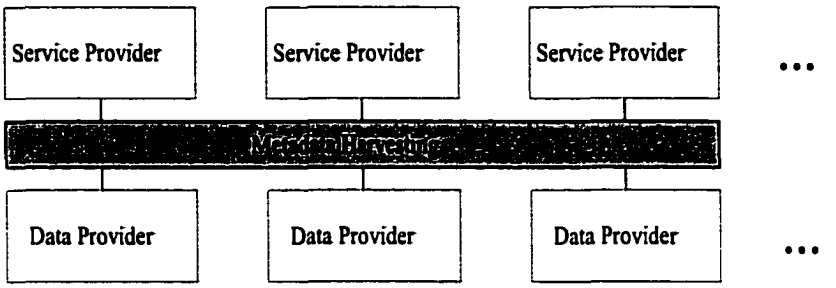

FIG. 3.1: Architecture of the basic OAI-PMH model

The OAI-PMH protocol focuses on the clear interface between data providers and services providers. Many other issues, such as the registration service, are essential to support a large scale of distributed and replicated services. In Figure 3.2, we define a model for metadata harvesting which addresses many of these issues. The data provider maintains one repository for metadata harvesting. The harvester is the key service which uses OAI-PMH to maintain the synchronization between data providers and various services. Above the harvester, various services, such as centralized federation services, replication services, and citation linking services, can be built for end-users. In addition, we introduce the OAI-PMH proxy, cache, and gateway services to optimize the functioning of the model underlying the OAI-PMH, they provide an infrastructure that can be used by all other components to achieve interoperability, scalability and reliability. A registration service is essential if the number of OAI-PMH compliant repositories keeps growing, it addresses the resource discovery and identifier resolution problem within the highly replicated environment of OAI-PMH.

3.2 DATA PROVIDERS

A data provider maintains one repository that supports the OAI-PMH as a means of exposing metadata. There are more than 100 registered OAI-PMH-compliant repositories. The design of a good data provider presents many challenges. After running a metadata harvesting system for nearly two years, we have discovered a number of problems. These include metadata quality, server availability, service quality, and implementation of resumption token. In OAI-PMH, community-specific description,

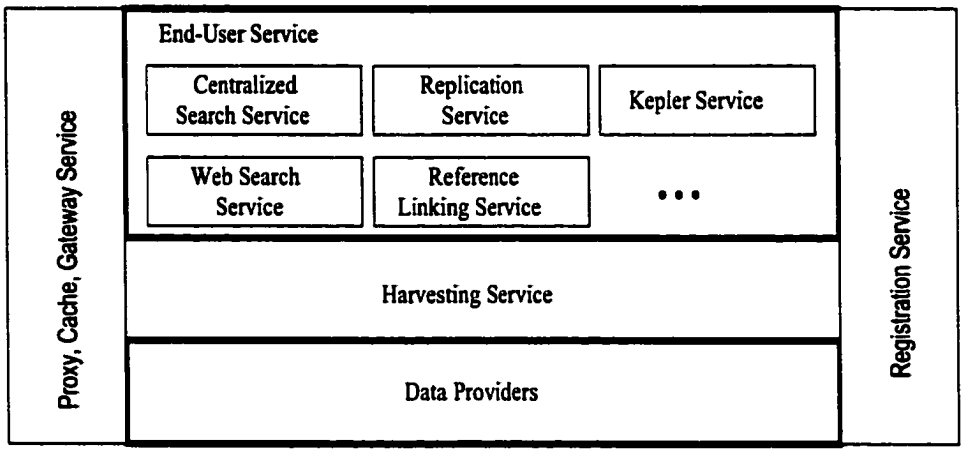

FIG. 3.2: Architecture of an optimized metadata harvesting model

or metadata specificity, is addressed in the technical framework by support for parallel metadata sets. We study current metadata format variability in OAI-PMH and present a solution to support multiple metadata formats in service providers.

3.2.1 Architecture of a Data Provider

Figure 3.3 shows the major components of a data provider. It includes three major modules. First, a request processor accepts OAI-PMH requests and validates the correctness of the request, then sends it to a record factory. Next, the record factory fetches metadata from data sources and converts them if necessary. The matched records are then encoded into XML format in the XML encoder and responded in an HTTP response. The harvester may issue further requests based on the replies it receives.

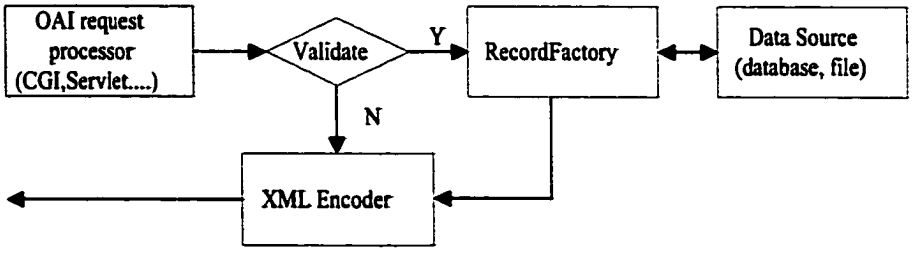

FIG. 3.3: Architecture of an OAI-PMH data provider

3.2.2 Quality of Data Providers

The quality of data providers has been a significant problem since the release of SFC in 2000. During the testing of data harvesting from OAI-PMH data providers, numerous problems were found. Alan Kent reported that 36 out of 76 data providers could not be harvested in March, 2002 [51]. In July, 2002, a review of the Celestial Service [15] showed that 31 out of 96 data providers have problems in harvesting. We discovered that not all archives strictly follow the OAI-PMH; many have XML syntax and encoding problems, and some data providers are periodically unavailable. Many responses were not well-formatted XML files. Sometimes foreign language and other special characters were not correctly encoded. XML syntax errors and character-encoding problems were surprisingly common and could invalidate entire large data sets. Incremental harvesting proved beneficial as a work-around. In Table 3.1, we summarize the error logs of harvester for nearly one year. In summary, we experienced errors in 77 of 103 repositories; 2539 of 20952 rounds of harvesting have errors. In our harvester, one round of harvesting usually includes a series of OAI-PMH requests: Identify, ListSets, and ListRecords. OAI-PMH is based on an incremental harvesting model, The first round of harvesting collects all available metadata records and is error-prone; after that, only new published data is harvested and the problem occurs less.

TABLE 3.1: Harvester error logs (from 08/13/2001 to 07/29/2002)

Type	Total	Any Error	XML Error	Server Error
Number of Repositories	103	77	18	73
Round of Harvest	20952	2549	433	2116

The OAI website in Cornell university validates registered data providers for protocol compliance. It uses XML schemas to verify the standard conformance [119]. However, this verification is not complete; it does not cover the entire harvesting scenario and does not verify the entire data set. Additionally, such verification cannot detect semantic errors in the protocol implementation, such as misunderstanding of DC fields. For certain XML encoding errors, an XML parser can help avoid common syntax and encoding errors. If the data provider builds quality control and data cleaning into its local accession policy [121], the service provider will have significantly less work to do and will have to discard fewer dirty data records. These errors can also be partially addressed on the service provider side. The approach adopted by

our harvester is presented in Section 3.3.

3.2.3 Implementation of Resumption Token

When a harvester issues a request to an archive that will result in a large amount of data, the archive may provide part of the data and then offer a resumption token as placeholder for the rest. If more is desired, that token is sent, and another batch of results is returned. OAI-PMH leaves the resumption token format up to developers.

There are two ways to implement a resumption token: stateful and stateless. A stateful implementation follows the direction of SQL-style transactions and cursors [29]; it is complicated and usually not robust in error recovery. In a test conducted by our harvester in February, 2002, five repositories could not be harvested due to the error in the resumption token.

To solve the problem, we design a stateless resumption token for data providers. The request parameters and cursor are encoded in the resumption token; the cursor includes the information of datestamp and identifier. The state information is thus saved in the resumption token, and the data provider does not need to keep the state information. Whenever a request with a resumption token comes, the data provider can decode the resumption token and re-build the query ordered by datestamp. By this way, a repository will be guaranteed to be harvested completely even if it has frequently changed data. The algorithm is described in Figure 3.4.

3.2.4 Parallel Metadata Format

For a digital library to be OAI-PMH-compliant, it must expose its metadata in DC and use the OAI-PMH protocol. That is, a DL community may use its own richer metadata set intra-community, but must have a second set of DC metadata exposed in order to be OAI-PMH-compliant.

TABLE 3.2: Parallel metadata sets usage in OAI-PMH (08-02-2002)

Number of Archives	Total number of metadata formats used	Total number of unique metadata formats
92	**149**	**21**

Algorithm 3.1 *Stateless ResumptionTokenforDataProvider*
Input : $OAI-PMH$ *ListRecords, ListIdentifiers, ListSets request*
Procedure :

```
while(TRUE){
        wait(request);
        if(request has resumptionToken){
                parameters, cursor = decode(resumptionToken);
        }else{
                parameters = decode(request);
                cursor = null;
        }
        Select from database where (parameters is true) and (datestamp
        +identifier > cursor)order by (datestamp + identifier);
        Assemble replied records till lastrecord;
        newcursor = datestamp(lastrecord) + identifier(lastrecord);
        new_resumptionToken = encode(parameters + newcursor);
        response (resultset + new_resumptionToken);
}
```

FIG. 3.4: Algorithm for stateless resumption token

The use of unqualified DC as a common metadata format in OAI-PMH proves to be very helpful for building a quick prototype. However, richer metadata formats are essential for building a richer service. The current usage of parallel metadata set is listed in Table 3.2. In total, 21 different metadata formats are used in 92 repositories. Figure 3.5 shows how many metadata formats are used in each repository. The majority of them (53) use unqualified DC only, 28 repositories use two metadata formats, four repositories use three metadata formats, and seven use four metadata formats. Figure 3.6 shows the number of usage each metadata format except oai_dc (oai_dc is mandatory and every repository uses it). The OLAC metadata set is a standard defined by the OLAC community [101]; OAI_RFC1807 [59] and OAI_MARC [99] are standard metadata formats and recognized in the OAI-PMH. The other 16 kinds of metadata format are local formats; it is very difficult to implement richer service over these metadata without individually studying each format.

Since OAI-PMH is XML based, if a data provider supports a stylesheet file for locally defined XML metadata formats, the service provider can automatically present the harvested records without knowing the semantic meaning. Figure 3.7 shows how

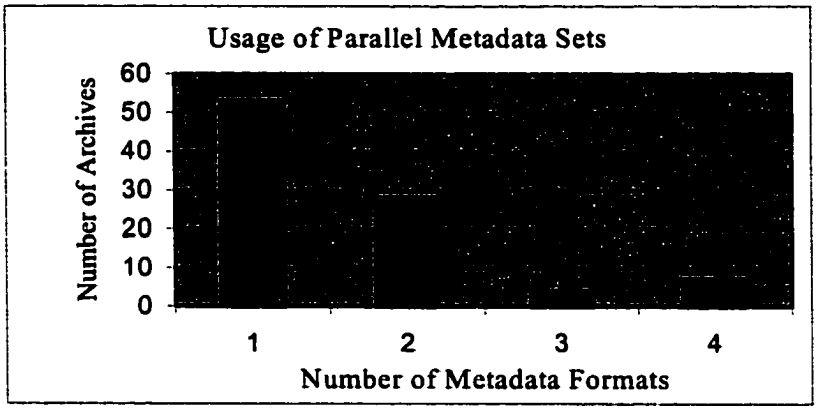

FIG. 3.5: Usage of parallel metadata sets

we support parallel metadata presentation by XSLT [21] processing. Data providers will export their metadata (by OAI-PMH) and presentation format (by XSLT). A service provider harvests the metadata and builds a search interface. The resource discovery is performed by the service provider, and the final presentation of the data is accomplished by the data provider's XSLT. With this mechanism, data providers may define an explicit method for presenting the metadata format, which is especially useful for rarely used or repository-specific metadata formats. OAI-PMH 2.0 introduces a "branding" mechanism for the service provider to render the metadata in the stylesheet specified by the data provider. The branding mechanism supports the model defined in Figure 3.7.

3.3 HARVESTER

Similar to a web crawler, the OAI-PMH harvester traverses the data providers automatically and extracts metadata. The significant differences between the OAI-PMH harvester and a web crawler are that the OAI-PMH harvester normalizes the metadata, thus producing more complete and accurate results, and exploits the incremental, selective harvesting defined by the OAI-PMH.

Data providers are different in data volume, partition definition, service implementation quality, and network connection quality. All these factors influence the harvesting procedure. Historical and newly published data harvesting have different

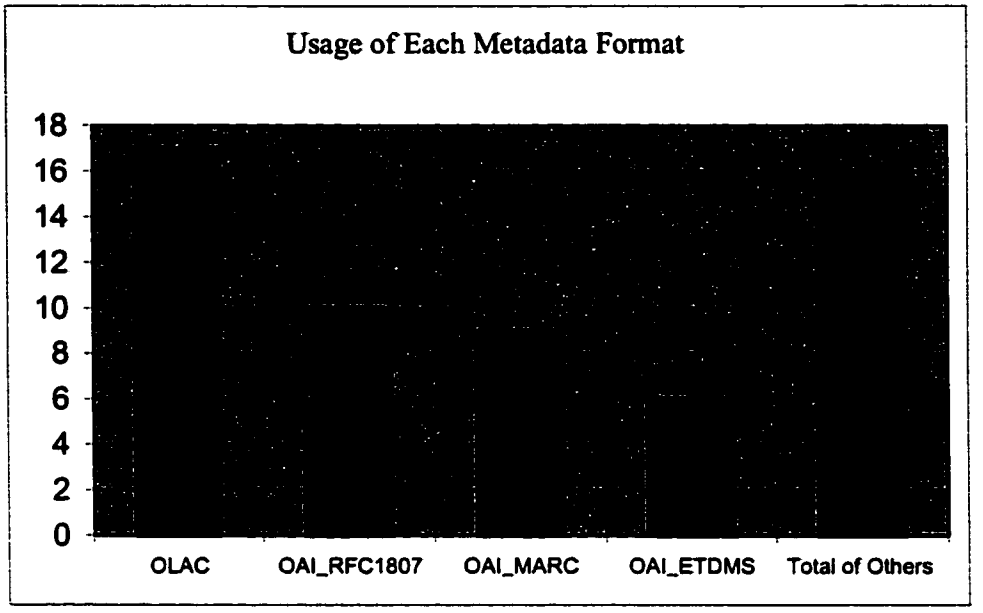

FIG. 3.6: The number of usage of each metadata format

requirements. When a service provider harvests a data provider for the first time, all past data (historical data) needs to be harvested, followed by periodic harvesting to keep the data current. Historical data harvests are high-volume and more stable. The harvesting process can run once, or as is usually preferred by large archives, as a sequence of chunk-based harvests to reduce data provider overhead. To harvest newly published data, data size is not a major problem, but the scheduler must be able to harvest new data as soon as possible and guarantee completeness – even if data providers provide incomplete data for the current date. The OAI-PMH provides flexibility in choosing the harvesting strategy; theoretically, one data provider can be harvested in one simple transaction, or one can be harvested as many times as the number of records in its collection. But in reality, only a subset of this range is possible; choosing an appropriate harvesting method has not yet been made into a formal process. We defined four harvesting types:

- bulk-harvest of historical data
- bulk-harvest of new data
- one-by-one-harvest of historical data

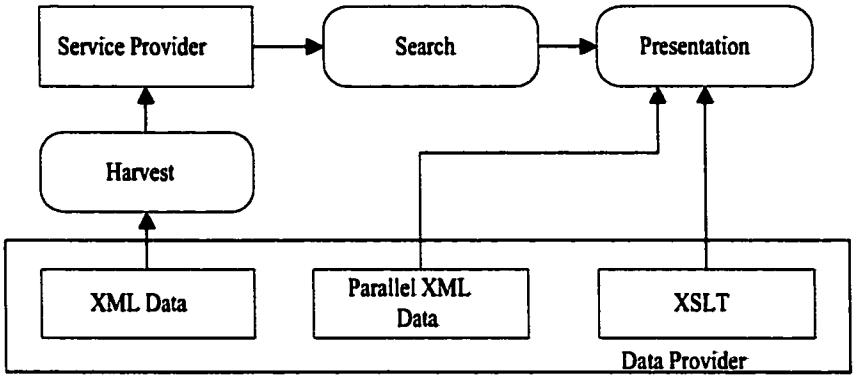

FIG. 3.7: XSLT processing to support parallel metadata format in OAI-PMH

- one-by-one-harvest of new data

Bulk harvesting is ideal because of its simplicity for both the service provider and data provider. It collects the entire data set through a single HTTP connection, thus avoiding a great deal of network traffic. However, bulk harvesting has two problems. First, the data provider may not implement the resumption token flow control mechanism of the OAI-PMH, and thus may not be able to correctly process large (but partial) data requests. Secondly, XML syntax errors and character-encoding problems – these were surprisingly common – can invalidate entire large data sets.

One-by-one harvesting is used when bulk harvesting is infeasible. However, this approach imposes significant network traffic overhead for both the service and data providers since every document requires a separate HTTP connection.

The default harvesting method for every data provider begins as bulk harvest. We keep track of all harvesting transactions, and if errors are reported, we determine the cause and manually tune the best harvesting approach for that data provider.

3.3.1 Robust Harvesting

There are two common problems that a robust harvester should deal with: unstable servers and XML encoding errors.

There are various problems that the network or data provider may experience. As we specified before, the cost of an interrupted resumption token can be expensive. In response, a robust harvester should try several times to improve the efficiency in case the interruption is short-term. The harvester can simply re-try after a random

period of time, or period calculated by the exponential backoff algorithm.

XML syntax errors and character-encoding problems are surprisingly common. They can invalidate entire large data sets. Besides the efforts on the data provider side to avoid this problem, a robust harvester can be designed to handle XML encoding errors. This mechanism is implemented by using specific features of SAX parser [114]. A SAX parser will report the precise position (line number and column number) of the first XML encoding error. Based on position information, the first bad record is detected and removed. The remaining data is validated again. The process runs iteratively until all bad records are removed; only good records are saved for further processing. This harvester manages to harvest most OAI-PMH-compliant repositories despite common XML encoding errors.

3.3.2 Hierarchical Harvesting

We have also implemented an OAI-PMH layer over the harvested metadata that allows our service provider to act as a data provider, disseminating metadata harvested from other data providers (Figure 3.8). This allows for the hierarchical harvesting of content, similar to the system of gatherers and brokers defined in Harvest. This structure has a great deal of flexibility in how information is filtered and interconnected between data providers and service providers. For example, one service provider might index papers in computer science, while another could build a general scientific service by harvesting the existing computer science harvester. Hierarchical harvesting also could provide the mechanism for caching and replication services.

A service provider normalizes harvested data. Thus, the re-exposed data might not be the same data harvested from the data providers. This situation can introduce both intellectual property and provenance issues. The document identifier is the one unique metadata item that should be kept in all locations to allow for tracking the source of the document.

3.4 REGISTRATION SERVICE

The OAI-PMH raises the "awareness" question, namely how service providers can find out the existence of data providers and vice versa. In addition, different data/service providers can be aware of and link to each other by using OAI-PMH unique identifiers. A distributed service model could be accomplished by sharing

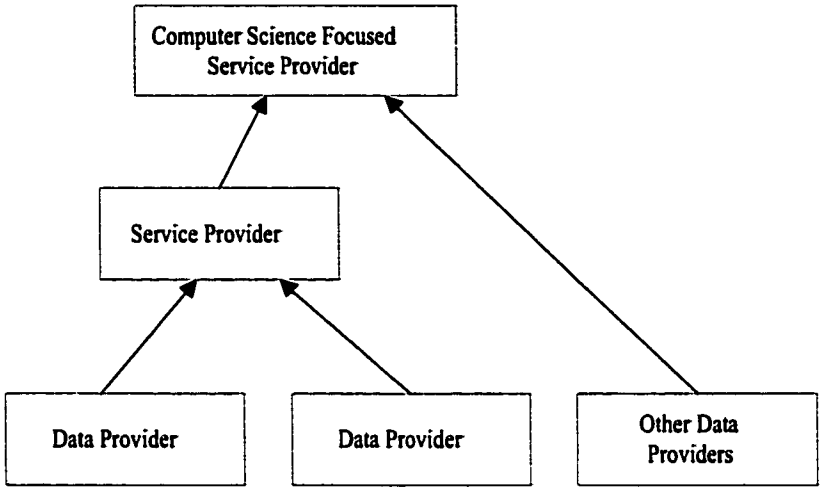

FIG. 3.8: Hierarchical harvesting

different services.

We believe there are two levels of registration process in the environment of OAI-PMH: repository-level and record level. The repository-level registration needs support three mechanisms:

1. The data provider registers and updates its status in a service provider. A basic registration service allows the registration of base URL of a data provider; an advanced registration service allows the data provider to notify service provider when its status (such as server availability and update schedule) changes.

2. The service provider automatically discovers appropriate data providers for harvesting. For example, a central OAI registration service for data providers [100] is provided by Cornell University. Service providers can periodically harvest base URLs of registered data providers.

3. The service provider exposes information about its harvested repositories. An OAI-PMH layer can be supported to describe the archives from which it harvests. That is, instead of the records corresponding to records from the data providers, the records returned from this interface describe the actual archives themselves. This interface was implemented to provide a dynamic and machine-readable mechanism for discovering the data providers from which a service provider harvests.

The record-level procedure needs to support two mechanisms:

1. Identifier Resolver: Given an oai-identifier, the resolver is able to return the corresponding XML records. The identifier resolver maintains the mapping relationship between the oai-identifier and its base URL. It provides a single source from which other services can fetch record instead of keeping a local list of correspondences between identifiers and base URLs.

2. Service Linking: In OAI-PMH, data providers may be harvested by many service providers, each providing different services for the same record. All these services potentially could link to a broker page, the broker page dynamically checks whether or not a service exists for a specific record. If so, it adds a link to the corresponding service provider. In order to know which records are available in advance, the broker issues an OAI-PMH GetRecord lookup to the target service (which has an OAI-PMH export). Based on the reply, the broker knows whether a record is harvested.

3.5 END-USER SERVICE

Many services become possible with the adoption of OAI-PMH, the federated search service and repository synchronization are most obvious applications. The OAI-PMH also provides an interface which exposes the "hidden" information to general web search engines. Other services such as cross-archive citation linking are also emerging.

Based on the OAI-PMH, there are two approaches to building a federated digital library that allow users to access contents in all the libraries through a single interface: centralized and replicated. In the centralized approach (Figure 3.9), a federation service harvests metadata from the OAI-PMH-enabled libraries and provides a unified interface to search all the collections. This approach has been adopted by Arc and other OAI-PMH service providers [84, 45, 120]. However, a centralized search service is not a suitable approach if the primary objective is to use native library interfaces. Besides this limitation, the centralized approach suffers from the organizational logistics of maintaining a centralized federation service and having a single point of failure. The replicated approach addresses these problems (Figure 3.10). This approach can be viewed as mirrored OAI-PMH-compliant repositories, where every participant has its own federation service. The consistency between

these services is maintained using OAI-PMH. As a federation service is locally available, it becomes easy to push other participants' metadata into the native library. In addition, this approach supports several levels of redundancy, thereby improving the availability of the whole system. For example, a failure of a system at one repository would not severely impact users at other repositories. In fact, users at the affected repositories would continue to search and discover reports from other repositories, though they may not be able to see reports that are added to the system at other repositories during the down time. The centralized approach is further discussed in Chapter 4, and replicated approach is discussed in Chapter 5.

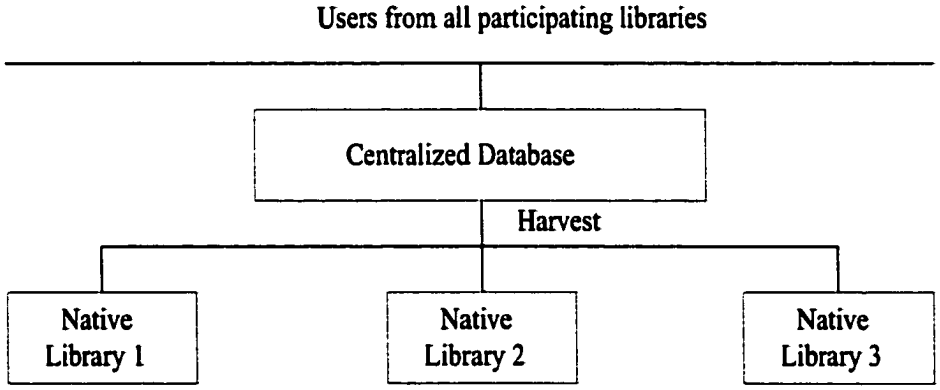

FIG. 3.9: Centralized federation service

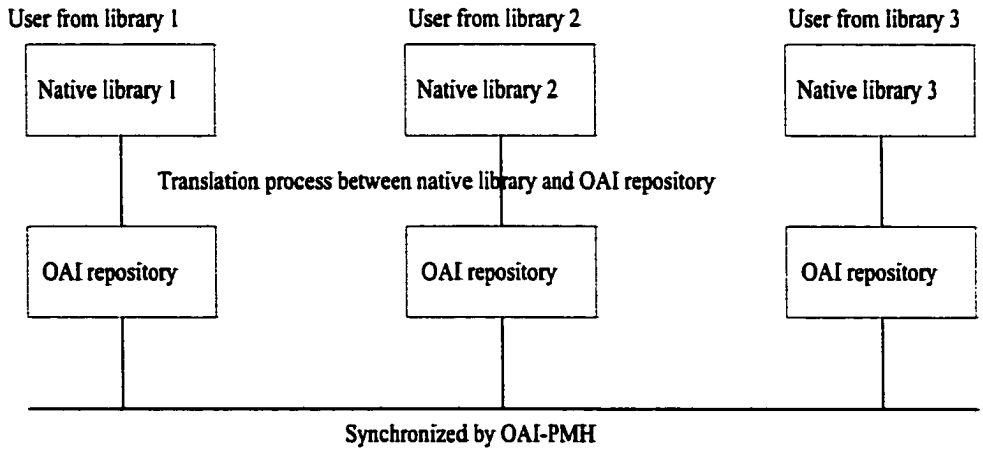

FIG. 3.10: Replicated federation service

3.6 PROXY, CACHE, AND GATEWAY SERVICES

The current and emerging applications based on metadata harvesting require a scalable and reliable infrastructure to support them. We introduce the concepts of OAI-PMH proxies, OAI-PMH caches, and OAI-PMH gateways as tools for the optimization of the functioning of the data provider/service provider model underlying the OAI-PMH. The goal is to achieve interoperability, scalability, and reliability of OAI-PMH services. Various applications and services can exploit the services included in this infrastructure. An OAI-PMH proxy dynamically forwards OAI requests to data providers. For example, it can dynamically fix common XML encoding errors and translate between different OAI-PMH versions. An OAI-PMH cache caches metadata and can filter and refine them before exposing them to service providers. It also serves as a simple cache that reduces the load on source data providers and improves server availability. An OAI-PMH gateway can convert the OAI-PMH to other protocols and applications. For example, the gateway could convert between different protocols (e.g. SOAP [13]) and OAI-PMH.

3.7 TESTBED

Arc is designed as a testing system to study the challenges in metadata harvesting. It implemented the model presented in this Chapter. Arc harvests all open OAI-PMH data providers regardless of their contents. Arc includes a harvester and a search engine built over harvested metadata. As of August 2002, there were more than 100 data providers with over 1M metadata records. The number of records keeps growing with more OAI-PMH-compliant data providers (Figure 3.11), the spike in Figure 3.11 is usually caused by newly added collections.

In Arc, we also implement an experimental OAI-PMH layer over harvested data. Thus, one service provider can collect information from both data providers and service providers. By retrieving information from other service providers, service providers can also cascade indexed views from one another – using the service provider's query interface to filter or refine the information from one service provider to the next.

We encountered a number of problems in developing Arc. Different archives have

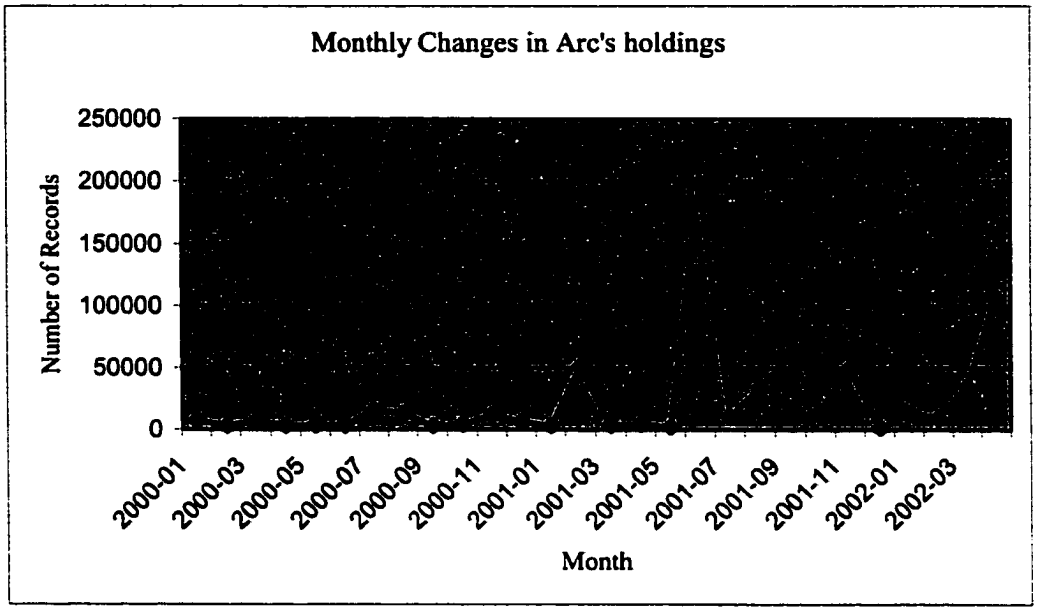

FIG. 3.11: Monthly changes in Arc's holdings

different format/naming conventions for specific metadata contents, thus necessitating data normalization. Arbitrary harvesting can overload the data provider making it unusable for normal purposes. The data providers' security protection can block the crawler and make harvesting difficult to implement. Initial harvesting when a data provider joins a service provider requires a different technical approach than periodical harvesting that keeps the data current.

The Arc architecture is based on the Java servlets-based [87] search service that was developed for the Joint Training, Analysis and Simulation Center (JTASC) [79]. This architecture is platform-independent and can work with any web server. Moreover, the changes required to work with different databases are minimal. Our current implementation supports two relational databases, one is commercial (Oracle [103]), and the other is open source (MySQL [88]). The architecture improves performance by employing a three-level caching scheme. Figure 3.12 outlines the major components: Search Engine, Harvester, and an OAI layer over Arc for hierarchical harvesting.

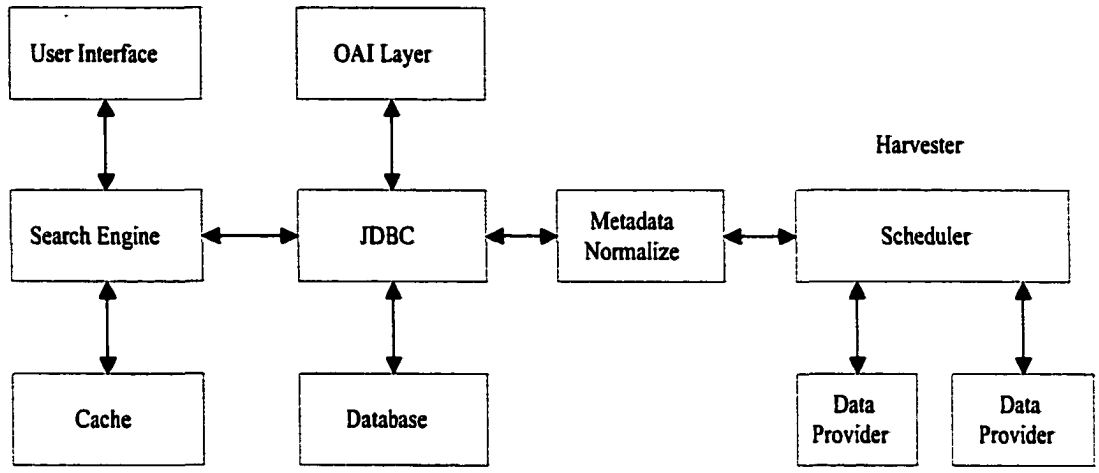

FIG. 3.12: Architecture of Arc

The Arc harvester is implemented as a daemon written in Java. At the initialization stage, it reads the system configuration file, which includes properties such as user-agent name, interval between harvests, data provider URL, and harvesting method. The harvester then starts a scheduler, which periodically checks and starts the appropriate task.

3.8 DISCUSSION

Little is known about the long-term implications of a harvest-based DL. The presented architecture motivates studies on naming service, repository synchronization problem, metadata quality, and scalability. We summarize this chapter and further studies are introduced in later chapters in this dissertation.

Data Provider and Service Quality During the testing of harvesting from data providers, numerous problems were found. We have presented how to implement a robust harvester in this chapter. An OAI-PMH proxy or cache service can dynamically resolve these problems, these are further discussed in Chapter 8.

Update Frequency, Push Model, and Security The OAI harvesting model is built on service providers "pulling" metadata from a set of data providers. It is interesting to study the update frequency of data providers and to design

the optimal harvester to reach better freshness with minimal cost. We discuss more about the repository synchronization problem in Chapter 7.

Controlled Vocabulary Some normalization was necessary to achieve a minimum presentation of query results. A controlled vocabulary will be of great help for a cross-archive search service to define such metadata fields as "subject." We discuss how to build a federation service with existing controlled vocabulary in Chapter 5.

CHAPTER 4

REPOSITORY SYNCHRONIZATION

Because data providers constantly change, a harvester should periodically contact data providers for newly added and changed data, so that the contents are maintained up to date. The OAI-PMH harvesting model is built on service providers "pulling" metadata from a set of data providers. It is designed with the intention of stable traditional publishing mediums (journal, e-prints, etc.). However, the synchronization problem is quite different in an author self-archiving environment such as the Kepler service. The synchronization problem of OAI-PMH is studied in this chapter. The synchronization problem in Kepler is studied in Chapter 7.

The remainder of this chapter is organized as follows:

- Section 4.1 introduces the repository synchronization problem in OAI-PMH and outlines our approach.
- We formally study the repository synchronization problem in Section 4.2. Several metrics such as update frequency and freshness are defined.
- The frequency of new or modified records available through the data provider plays a major role in determining the harvesting frequency. We study the update frequency of OAI-PMH-compliant data providers in Section 4.3.
- Section 4.4 presents two algorithms to implement repository synchronization in the OAI-PMH framework.
- Section 4.5 presents a syndication container by which the data provider can identify its update rate.
- Section 4.6 summarizes the related work in synchronization and freshness problem.

4.1 INTRODUCTION

The synchronization problem – how to keep the metadata records of data providers and service providers consistent – is a problem that can distort the results a user obtains from a search. The user must trust that the service provider has an accurate

assessment of the contents of the data providers that it harvests. The OAI-PMH supports selective, incremental harvests, and the synchronization is maintained by periodic re-harvesting. Service providers are expected to exploit these properties in order to limit the load imposed on the data providers while still maintaining fresh data for their services.

To study this problem, it is imperative to understand the requirement of the application. For example, maintaining freshness in seconds for the market value of a stock could be critical; a news aggregator need to maintain hourly freshness for a satisfactory service; a web service engine may re-harvest its indexed page in several months. We define l as the acceptable latency at which the harvester should be synchronized with data providers. The OAI-PMH 1.x only supports granularity of day, the OAI-PMH 2.0 starts to support granularity of second. However, the granularity of the protocol is not likely change the nature of update frequency of a repository.

The OAI-PMH harvesting model is built on service providers "pulling" metadata from a set of data providers. After studying data and the harvest log of Arc, we conclude that most data providers have a steady change rate, but different data providers present significantly different rates. For the traditional publishing medium, the OAI-PMH harvesting model works well. However, freshness can be further improved if the harvester can dynamically adjust the "pulling" rate based on the change rate of data providers. Motivated by the work in RSS (RDF Site Syndication Format) [6] and other applications such as news syndication, we define an optional container in which a data provider can describe its update rate.

4.2 METRICS FOR UPDATE FREQUENCY AND FRESHNESS

The OAI-PMH is based on a coordinated model that a harvester can issue a request to get all updated records in a repository after a specific date. This model is superior to the model of web crawlers, which have to discover the update time of each record individually.

We formally define several metrics to measure the update frequency of data providers. Let $\{r_1, ..., r_M\}$ be the M data providers we are going to monitor. We assume that n observations are made of each repository, the observations being made at regular intervals. The choice of interval, Δt, will be made appropriate to the latency of the repositories involved and is lagely irrelevant to the metrics that follow.

We will therefore denote the observation times as $\{t_1, ..., t_n\}$, where $t_{j+1} = t_j + \Delta t$.

Repository Update Status: The update status of a repository r_i at time t_j:

$$S(r_i; t_j) = \begin{cases} 1 & if\ r_i\ is\ updated\ at\ time\ t_j \\ 0 & otherwise \end{cases} \quad (1)$$

Record Update Rate: Let $R(r_i; t_j)$ denote the number of updated records of a repository r_i observed at time t_j.

Repository Update Interval: We define the update interval at time t_j:

$$I(r_i; t_j) = \begin{cases} 0 & if\ S(r_i; t_j) == 0 \\ j - k & t_k\ is\ last\ updatetime\ before\ t_j \end{cases} \quad (2)$$

Average Repository Update Interval: In the period of observance $\{t_1, ..., t_n\}$, the average update interval of a repository r_i is

$$U(r_i) = \frac{\sum_{j=1}^{n} I(r_i; t_j)}{\sum_{j=1}^{n} S(r_i; t_j)} \quad (3)$$

Average Repository Update Frequency: The average update frequency of a repository r_i is:

$$FRQ(r_i) = \frac{1}{U(r_i)} \quad (4)$$

Average Record Update Rate: The average records update rate of repository r_i is

$$AVG(r_i) = \frac{\sum_{j=1}^{n} R(r_i; t_j)}{\sum_{j=1}^{n} S(r_i; t_j)} \quad (5)$$

Freshness of a Data Provider: The freshness of a data provider r_i in harvester side at time t_j is

$$F(r_i; t_j) = \begin{cases} 1 & if\ r_i\ is\ up - to - date\ at\ time\ t_j \\ 0 & otherwise \end{cases} \quad (6)$$

Freshness of Harvester: The freshness of the harvester H at time t_j is

$$F(H; t_j) = 1/M \sum_{i=1}^{M} F(r_i; t_j) \quad (7)$$

Using these metrics we are able to measure the freshness of service providers.

Update Cost for Data Provider: Let C_d denote the update cost for a data provider.

Update Cost for Harvester: Let C_s denote the update cost for harvester.

In the observance period of T, the harvester issues requests to each data provider in an interval of l, and l is the acceptable latency at which the harvester should be synchronized with data providers. This latency could range from several seconds to several months, depending on different applications.

In a basic model, the harvester issues requests every l interval to each data provider:

$$C_d(r_i) = \frac{T}{l} \tag{8}$$

$$C_s = \frac{TM}{l} \tag{9}$$

In an optimal model, we assume that the harvester knows when the data provider is updated in advance, and it issues requests right after the data provider is updated.

$$C_d(r_i) = \frac{T}{U(r_i)} = T * FRQ(r_i) \tag{10}$$

$$C_s = \sum_{i=1}^{M} \frac{T}{U(r_i)} = \sum_{i=1}^{M} T * FRQ(r_i) \tag{11}$$

From formula (9) and (11), we can derive that the acceptable latency, number of data providers, and repository update frequency play important roles in caculating the update cost. We illustrate these metrics by two examples.

Example 4.1 In a typical digital library application, such as researchers discover the existence of a paper or technical report, a daily latency should satisfy most requirements, if the observance period of T equals one day, from formula (9) we can derive $C_s = M$.

Example 4.2 In a news aggregator, the acceptable interval is at the minute level; each participating news agency updates its site every hour. If the observation period is one day, in the basic model, $C_s = 1440 * M$, the basic model will not scale. In contrast, in the optimal model, $C_s = 24 * M$, it promises better efficiency.

In the optimal model, the harvester essentially allocates more resources to active data providers; the prerequisite is that harvester must know the update interval of the data providers. There are four approaches:

Best Estimation The harvester estimates the record update frequency by learning the harvest history. However, this requires the data provider must present a constant update frequency. After studying the harvest log of Arc, we conclude in Section 4.3 that many current OAI-PMH compliant data providers present a constant update rate.

Syndication A data provider may describe its update frequency explicitly; this is described in Section 4.5.

Subscribe/Notify Best estimation and syndication are compliant with the OAI-PMH framework, but it relies on a constant update rate of data providers, which may not be true in some applications. A data provider may notify a service provider whenever its content is changed. This model is an extension of OAI-PMH and is described in Chapter 7, the Kepler framework.

Push Model Data providers may directly push updates to service provider side, this is also demonstrated in Chapter 7.

4.3 UPDATE FREQUENCY OF DATA PROVIDERS

The frequency of new or modified records available through the data provider plays a major role in determining the balance between harvesting too often and not enough. The nature of the data provider can influence how often records are modified or updated. E-print type data providers are likely to have a small but steady stream of ongoing daily or weekly updates. Museum or historically oriented archives will have an initial burst period of accession (perhaps all at once), but then are likely to trickle down to just infrequent error corrections or edits. Although not currently implemented by any data providers, if a data provider allowed the metadata to change based on usage, annotations, or reviews as specified in the NSDL project [56], the required harvesting would likely become significant.

In this Section, we present our experimental results that show how OAI-PMH-compliant data providers change. We try to answer the following questions: (1) does the data provider change at a constant rate? (2) How often does a data provider change? We run the Arc harvester once a day to harvest approximately 100 OAI-PMH-compliant data providers. The datestamp of harvested record is kept in a database. The change rate covers new data, modified data, and deleted data.

Table 4.1 lists the monthly average update rate of records in selected e-print archives. They are randomly selected from stable OAI-PMH compliant e-print services (We consider stable services demonstrate a more reliable trend), the complete table is available at Appendix C. This table shows that in long term many e-print services have steady records update rate. This can be explained that e-print services have a relatively stable user base.

TABLE 4.1: Monthly records update rate ($R(r_i; t_j), \Delta t = 1\ month$) of E-Prints archives (from 2002-01 to 2002-09), the complete table is available at Appendix C.

archive	J-02	F-02	M-02	A-02	M-02	J-02	J-02	A-02	S-02
CPS	28	10	9	2	11	15	2	1	6
VTETD	16	25	10	84	115	52	51	78	45
arXiv	7744	3198	3874	3089	3605	3672	4462	4181	4505
bmc	50	20	5	11	68	3	0	0	5
cogprints	13	19	10	11	8	40	15	41	11
in2p3	180	140	276	57	90	110	108	52	141
ltrs.larc.nasa	12	40	31	22	42	35	71	31	24
mathpreprints	5	6	3	3	20	40	12	7	12
mit.etheses	46	86	142	119	189	63	75	124	82

Based on the data we collected, we can analyze how long it takes for a data provider to change. For example, if a data provider changes 5 times in 5 months, we may estimate that the average update interval of the data provider is 5 months/5 =1 month. Note that the granularity of the estimated change interval is one day, because OAI-PMH 1.x uses day as the unit of datestamp. In Table 4.2, we list the daily average update interval, average update rate, standard deviation of update interval, and Cofficient of Variation (C.O.V.). The complete table is in Appendix D. It shows that most of them have a relatively small C.O.V. In Figure 4.1, we summarize the result of this analysis. In the figure, the horizontal axis represents the average update interval of data providers, and the vertical axis shows the fraction of data providers changed at the given average interval. We can observe that less than 10% of the data providers change daily, while about 70% of the data providers change monthly or longer.

TABLE 4.2: Repository update interval (09/30/3001-09/30/2002)

archive	$AVG(r_i)$	$U(r_i)$	$stdv(I(r_i))$	$C.O.V.(I(r_i))$
arxiv.org	145.32	1	0	0
bmc	3.19	3.25	11.93	3.67
cogprints	20.07	3.8	4.13	1.09
CPS	1.58	3.7	7.15	1.93
in2p3	7.89	1.71	2.92	1.71
LTRS	2.36	2.81	7.12	2.53
mit.etheses	7.33	2.03	3.3	1.62
VTETD	3.5	2.11	56.5	26.77

$\Delta t = 1\ day$
$AVG(r_i)$: Average Update Rate
$U(r_i)$: Average Update Interval
$stdv(I(r_i))$: Standard Deviation of Update Interval
$C.O.V.(I(r_i))$: Coefficient of Variation of Update Interval

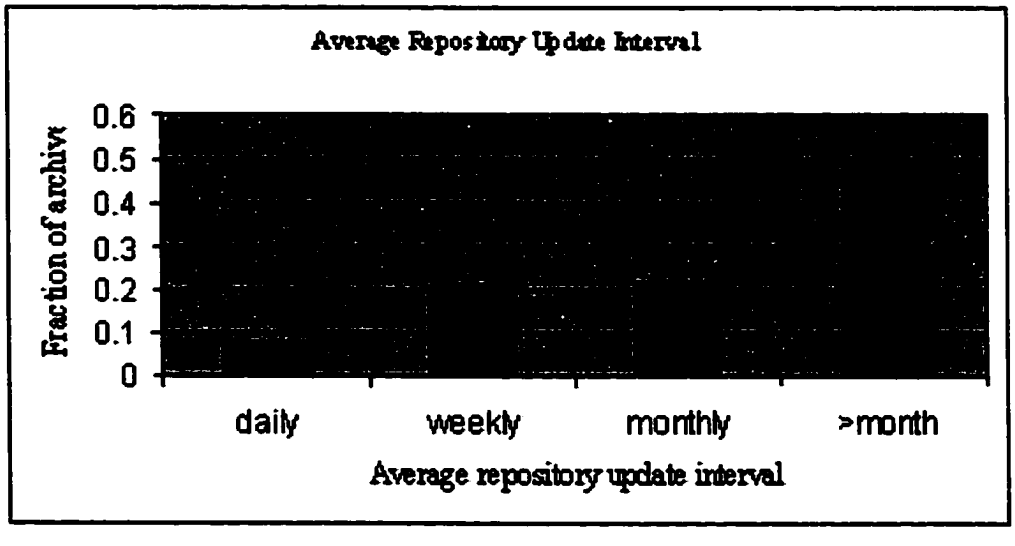

FIG. 4.1: Average repository update interval of OAI-PMH repositories

In summary, OAI-PMH data providers (especially E-prints archives) change at a steady rate overall, and the rates vary dramatically from site to site. Although it is difficult to precisely predict update time of one specific repository. However, a harvester may not necessarily provide 100% freshness at any time, for example, a harvester may harvest repositories with higher average update frequency more frequently, and harvest all other repositories once a week, it will still save a significant percentage of update cost. Typically, the requirement of freshness is decided by the application.

Because OAI-PMH supports the features of incremental harvesting, the implementation of a harvester with good freshness is not very difficult. For example, in the current configuration, Arc, a single thread-based harvester, takes less than one day to complete a harvesting cycle over all participating data providers.

4.4 SYNCHRONIZATION ALGORITHM FOR HARVESTER

In Section 4.3, we conclude that the OAI-PMH model of synchronization works well for current OAI-PMH data providers (out of about 100 data providers, most of them are e-prints archives or other digital library applications). However, there are some scenarios that require better synchronization.

- If the OAI-PMH becomes more popular and there are a large number of repositories available.
- If the annotation or review services are widely used, such as the NSDL project [56].
- If the OAI-PMH is used in some applications which require rapid dissemination in the unit of minute or hour, such as news or mailing lists.

We define two synchronization policies: fixed-list policy and adaptive-list policy. The fixed-list policy is implemented in the Arc harvester. The adaptive policy is based on the features that most data providers change at a constant but different rate. The change rate can be observed by the harvester, or it can be defined by the data provider, so we define an optional container to specify the change rate in the spirit of RSS.

4.4.1 Fixed-list Policy

Under the fixed-list policy, we synchronize the repositories in the same order repeatedly. We describe the fixed-order policy more formally in Figure 4.2. Here, each archive is historically harvested first, and a fresh harvest is repeated forever. Note that the last harvested time is fetched from the response of data providers in order to avoid clock skew. To contain any updates that happen during the harvesting period, the last harvested time is recorded before each harvest.

Algorithm 4.4.1 $Fixed-list\ synchronization$
$Input: ArchiveList = \{a_1, a_2, ..., a_n\}$
$LastHarvestTime = \{t_1, t_2, ..., t_n\} = null$
Procedure
$for(i = 1; i \leq n; i++)\{$
$\quad t_i = getresponsetime(a_i);$
$\quad historical\_harvest(a_i);$
$\}$
$while(true)\{$
$\quad for(i = 1; i \leq n; i++)\{$
$\quad\quad responsetime = getresponsetime(a_i);$
$\quad\quad fresh\_harvest(a_i, t_i);$
$\quad\quad t_i = responsetime;$
$\quad\}$
$\quad sleep(pre\_defined\_interval);$
$\}$

FIG. 4.2: Algorithm of fixed-list synchronization policy

4.4.2 Adaptive-List Policy

In this policy, the harvester changes its synchronization rate based on the average repository update interval. The average repository update interval can be learned from the previous harvest, or be defined by an optional container in the data provider as we describe in the next section (Figure 4.3).

Algorithm 4.4.2 *Adaptive − list Harvesting*
Input : $ArchiveList = \{a_1, a_2, ..., a_n\}$
$AverageUpdateInterval = \{u_1, u_2, ..., u_n\}(such\ as\ 1\ day,\ 90\ days, etc, .)$
$LastHarvestTime = \{t_1, t_2, ..., t_n\} = null$
Procedure
$for(i = 1; i \leq n; i++)\{$
    $t_i = getresponsetime(a_i);$
    $historical\_harvest(a_i);$
}
$while(true)\{$
    $for(i = 1; i \leq n; i++)\{$
        $if(currenttime - t_i \geq u_i)\{$
            $responsetime = getresponsetime(a_i);$
            $fresh\_harvest(a_i, t_i);$
            $t_i = responsetime;$
        }
    }
    $sleep(pre\_defined\_interval);$
}

FIG. 4.3: Algorithm of adaptive-list synchronization policy

4.5 SYNDICATION CONTAINER FOR UPDATE FREQUENCY

In OAI-PMH, the response to an *Identify* request may contain locally defined description containers that can be used to express properties of the repository. We define an optional container that identifies the update frequency of a data provider. The information provides an alternate way to build the algorithm in Figure 4.3.

The RSS (Rich Site Summary) syndication module provides syndication hints to aggregators and others picking up RSS feed regarding how often it is updated. For example, if a file was updated twice an hour, the update Period would be "hourly" and the updateFrequency would be "2."

UpdatePeriod Describes the period over which the data provider is updated. Acceptable values are: hourly, daily, weekly, monthly, yearly. If omitted, daily is assumed.

UpdateFrequency Used to describe the frequency of updates in relation to the

update period. A positive integer indicates how many times in that period the data provider is updated. For example, an updatePeriod of daily, and an updateFrequency of 2 indicates the data provider is updated twice daily. It omitted a value of 1 is assumed.

UpdateBase Defines a base date to be used in concert with updatePeriod and updateFrequency to calculate the publishing schedule. The date format takes the form: yyyy-mm-ddThh:mm.

The XML schema is defined in Figure 4.4, and an example is shown in Figure 4.5.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://purl.org/rss/1.0/modules/syndication/"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:syndication="http://purl.org/rss/1.0/modules/syndication/"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
 <element name="syndication">
    <complexType>
      <sequence>
        <element name="updatePeriod" minOccurs="0" maxOccurs="1"
                 type="syndication:updatePeriodType"/>
        <element name="updateFrequency" minOccurs="0" maxOccurs="1"
                 type="integer"/>
        <element name="updateBase" minOccurs="0" maxOccurs="1"
                 type="dateTime"/>
      </sequence>
    </complexType>
  </element>
<simpleType name="updatePeriodType">
    <restriction base="string">
      <enumeration value="hourly"/>
      <enumeration value="daily"/>
      <enumeration value="weekly"/>
      <enumeration value="monthly"/>
      <enumeration value="yearly"/>
    </restriction>
  </simpleType>
</schema>
```

FIG. 4.4: XML schema for syndication

4.6 RELATED WORK

Cho and Garcia-Molina [19, 20] gathered data from 270 web sites over a four months period and analyzed it by defining age and freshness metrics and by modeling the

```
<description>
<syndication xmlns="http://purl.org/rss/1.0/modules/syndication/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://purl.org/rss/1.0/modules/syndication/
http://dlib.cs.odu.edu/OAI/2.0/syndication.xsd">
        <updatePeriod>hourly</updatePeriod>
        <updateFrequency>2</updateFrequency>
        <updateBase>1999-02-01T00:00</updateBase>
</syndication>
</description>
```

FIG. 4.5: Example of syndication container

individual elements of a database as well as the database in its entirety. They then looked at synchronization frequency and compared synchronization order and resource allocation policies. However, they were dealing with un-coordinated changes of web pages, which is different from the incremental harvesting concept of OAI-PMH. They tried to maximize the harvested data in a very large collection (the entire Web) with limited resources, while we focus on improving freshness in a selected number of repositories.

The RSS syndication module provides hints to aggregators and others picking up this RSS feed regarding how often it is updated [6]. The RSS is widely used in news aggregation services.

The proposed "HTTP Distribution and Replication Protocol" (DRP) [130] creates an index page based on content digests to avoid unnecessary data transmission in deliberate replication over HTTP. After the initial download, a client can keep the data up-to-date using the DRP protocol. Using DRP the client can download only the data that has changed since the last time it checked. DRP is based on the Message Digest algorithm, such as MD5 [112], to identify the changes of content.

4.7 DISCUSSION

The harvesting service and repository synchronization are the key problems that OAI-PMH tries to solve. OAI-PMH optimizes the repository synchronization by supporting the incremental and selective harvesting. The model of OAI-PMH is sufficient for most typical digital libraries applications. However, it can be further enhanced to support a wide range of applications with the support of syndication

CHAPTER 5

CENTRALIZED FEDERATION SERVICE

One major objective of digital library interoperability is to provide a unified search interface across heterogeneous collections. This chapter introduces the effort of building a search interface based on structured metadata in heterogeneous OAI-PMH repositories. To solve this problem, we first study the structured metadata usage in OAI-PMH repositories. The analysis indicates that controlled vocabularies and values are widely used in most repositories. Usage is extremely variable however. We then implement an advanced searching interface that allows users to search and select in specific fields with data we construct from the harvested metadata, and also by an interactive search for the subject field.

The remainder of this chapter is organized as follows:

- Section 5.1 introduces the problem of building a unified interface over heterogeneous structured metadata.
- Section 5.2 analyzes the metadata variability in OAI-PMH repositories.
- In Section 5.3, we discuss the advanced search and interactive search approach to build the federated interface over heterogeneous metadata. In metadata harvesting, the metadata records are incrementally harvested, and the search interface adaptively adjusts with frequently-added new collections and harvested data.
- Section 5.4 discusses related work.
- Section 5.5 analyzes the initial experiences and discusses future work.

5.1 INTRODUCTION

One major objective of digital library interoperability is to provide a unified search interface. For the purpose of this thesis, a unified search interface is defined as an interface that can seamlessly search across multiple repositories. Many repositories have significant investment in controlled metadata fields. This includes controlled vocabularies (thesauri, subject heading lists, etc.), controlled values (a type of encoded

schema, usually a string formatted in accordance with a formal notation or parsing rules), and other locally controlled value or text. This chapter discusses several metadata fields used in DC while emphasizing controlled vocabularies and values. Controlled metadata is crucial for effective search and retrieval of Internet resources [25]; French *et al.* [34] point out that controlled metadata is of little use if it is not used effectively in query formulations. Our focus is how to build a rich, unified search interface that can exploit the controlled metadata across heterogeneous collections. The problem is solved by an advanced searching interface that allows users to search and select in specific fields with data we construct from the harvested metadata, and also by an interactive search for the subject field.

In the harvesting approach, the data are usually harvested on the service provider side, so we have the luxury of pre-building advanced services without relying on real-time interactive access to the remote archives. However, building a rich unified search interface over harvested metadata brings about new challenges. Many information-rich repositories have major investments in detailed metadata, which frequently includes some forms of controlled vocabularies and/or controlled values. To build better services, we need to understand how metadata control is used in these repositories, and to determine if we can exploit them in a unified interface. Furthermore, we need to know how easily new collections and freshly harvested metadata can be built into the unified interface.

One straightforward approach is to build a keyword search similar to typical web search engines. Web search engines represent a well-proven, successful technology based on harvesting and keyword searching. Keyword searching is a useful way to assume little about the semantics of a document, which works well for the heterogeneous, unstructured data sources that make up the Web. Nevertheless, when structured metadata is available, it fails to exploit the additional semantics.

Another approach to address the lack of a unified controlled metadata is to create a standard and map each repository's controlled metadata to the standard [52]. For controlled vocabularies, this approach can be improved by a meta-thesaurus based solution like UMLS (Unified Medical Language System), which

> preserves the meanings, hierarchical connections, and other relationships between terms present in its source vocabularies, while adding certain basic information about each of its concepts and establishing new relationships between concepts and terms from different source vocabularies.

[66]

Both approaches introduce significant human effort to maintain the relationships. Adding new collections to the federation leads to the complexity of updating relationships. Therefore, neither is feasible in our scenario that the federation service is maintained with limited resources.

In order to verify our approaches, we collect data from several different communities, ranging from museum to eprint collections. As participating archives add new records to their collections, their metadata records are also incrementally harvested. Analysis of these heterogeneous collections indicates that metadata control is widely used in most repositories, especially in certain metadata fields. Usage is extremely variable, however. From our study, it is clear that no single approach would allow effective use of the manifold metadata control we encountered. We solve the problem by implementing an advanced searching interface that allows users to search and select in specific fields with data we construct from the harvested metadata, and also by an interactive search for the subject field. As the metadata records are incrementally harvested, we address how to build these services over frequently-added new collections and harvested data.

We must point out that we are building a demonstration service to study the issues of metadata harvesting. It harvests all OAI-PMH-compliant repositories regardless of their subject or contents. In a specific community such as OLAC, a standard about how to use controlled vocabulary may be designed. If such a standard is successfully used across the community, it can reduce the integration works done on the service provider side.

5.2 METADATA VARIABILITY

A metadata field can be based on either controlled or free text. We consider three types of metadata control: controlled vocabularies, controlled values, and other locally defined metadata. Controlled vocabularies are typically used for subject access and can control synonyms, variant spellings, as well as providing broad term, narrow term, and other subject relationships. Controlled vocabularies include thesauri and classification schema. Controlled values are usually a string formatted in accordance with a formal notation or parsing rules (e.g. "2000-01-01" as the standard expression of a date). These controlled values include values of a "fixed or set length" (e.g.

"eng" or "en" as the standard expression of ISO 639-2 three-character vs ISO 639-1 two-character language code for "English"). Locally defined metadata may contain a mix of locally controlled text strings or values for a given metadata field/element used in records from a given repository, and an "encoded schema" may be implied but is not clearly identified and available for public use.

OAI-PMH uses unqualified DC as the default metadata set to enable minimal interoperability. Although OAI-PMH supports other metadata formats, our discussions are based on DC because it is the common metadata set supported by all OAI-PMH compliant repositories. Over the past several years, DC has developed as a *de facto* standard for simple cross-discipline metadata. It defines 15 metadata elements: creator, title, subject, description, publisher, contributor, date, type, format, identifier, source, language, relation, coverage, and rights. DC does not specify anything about syntax in any of these fields. From our observance, among the 15 DC fields, some, such as description, are most likely free-text based. The subject field tends to be based on controlled vocabularies, and other fields, such as type, date, format, and language may be based on either controlled values or locally defined values. In contrast to qualified DC, unqualified DC does not include an encoding scheme to aid in the interpretation of an element, so there is no definite way to decide whether a metadata field is controlled without consulting the original data providers. However, by studying the harvested metadata, in most cases the difference between controlled and free text input is obvious, so we consider the tendency be correct. We manually examined the subject, language, format, date, and type fields for further study of metadata variability. The meaning of "subject," "date," and "language" are fairly straightforward; the definition of "type" is "The nature or genre of the content of the resource," and of "format" is "The physical or digital manifestation of the resource." [131]

To understand how metadata fields are used, consider Table 5.1, which has been constructed based on the collections in Arc. Table 5.1 contains an excerpt from our analysis of how metadata is used in four archives. The complete data are available at Arc's website [3]. Table 5.2 lists the number of records harvested and the number of distinct subject, type, format, and language fields used in each archive. Table 5.2 also lists whether consistent formatting is used in the date field. In columns 4-7, a "zero" value means this metadata field is not used, either because the metadata is not available in the repository, or because the repository simply ignores it because

TABLE 5.1: Sample data of subject, format, language and date fields in four archives (excerpt)

	CogPrints	MIT etheses	NCSTRL
Subject	Complexity-Theory Computational-Linguistics Computational-Neuroscience	Computer-animation Computer- architecture Computer-composition	N/A
Format	N/A	application/pdf image/gif	N/A
Language	N/A	N/A	English French German
Date	1950-01-01 1953-01-01 1954-01-01	1900-01-01 1903-01-01 1921-01-01	1958-12-01 1959-03-01 1959-12-01
Type	Book Chapter Conference Paper Conference Poster	Thesis	Proceeding Proceedings

it is constant across all records (e.g. English language archives may simply leave the DC language field empty). In column 3, dealing with date field formatting, the value "Y" means consistent formatting is used; "N" means free input is used; "N/A" means this field is never used. In Table 5.2, if the number of records is significantly larger than the number of distinct values in one metadata field, it suggests that a controlled metadata is used. This is not always true, however, so we manually verified these results.

Figure 5.1 shows the percentage of archives that use controlled values in each metadata field. It indicates that metadata control is widely used among archives, especially in the type, format, language, and date fields. As we can see, about half use controlled vocabularies in the subject field. In the archives labeled "without metadata in specific field," many use a constant value (e.g. "English" in the LANGUAGE field based on other factors such as an assumption that records in "English-based" repositories represent only publications in the English language.)

In many circumstances, even if controlled metadata are used, each archive may employ its own semantics for these fields. Archives may have different semantics for the same field, and they frequently use different standards, such as subject classification methods. However, data providers have invested significant human and machine

TABLE 5.2: Metadata variability in Arc (to April 3, 2002). The full table is available in Appendix A.

Archive	No of Records	date	subject	type	format	language
8657690236	798	Y	53	1	0	0
AIM25	3962	N	2424	1	1	0
anlc	5	Y	2	1	2	0
anu	114	Y	22	6	0	0
aps	422	N/A	5	0	180	0
arXiv	182996	Y	121	1	0	12
bmc	220	Y	0	13	0	1
caltechCSTR	504	Y	8	2	0	0
caltecheerl	140	N	1	1	0	0
caltechETD	30	Y	10	1	4	1
cav2001	111	Y	103	1	0	0
CBOLD	89	Y	136	1	20	3
CCSDthesis	99	Y	16	1	0	0
CDLCIAS	36	Y	9	4	0	0

resources to use controlled metadata, and service providers should try to re-use these rich metadata.

One straightforward approach is to use a standard and map the controlled metadata from an individual archive to the standard. We could then reflect this approach in the search interface by showing the standard as a selectable option. This approach has three major limitations:

1. The standard may differ in terms of levels and semantics from that of an individual archive, leading to low-precision searches.

2. Adding a new collection to the federation leads to complexity in updating mapping tables.

3. Significant manual effort may be required to define the standard and the mapping tables. Moreover, any new archive may differ significantly from the standard, necessitating an update to the unified scheme.

Table 5.2 demonstrates that while the number of subject fields is large, the number of different language, type and format fields is limited in most archives. This leads directly to our design decision to create a browse interface for language, type, and

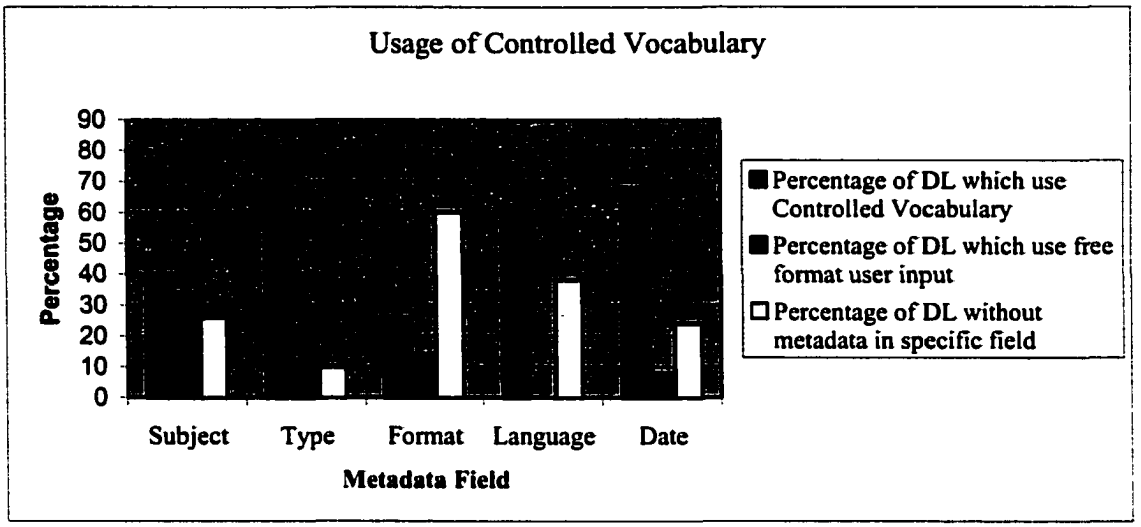

FIG. 5.1: Controlled metadata

format fields. An interactive search interface is designed for subject search. Because most date fields follow strict controlled values, we implement the date field as a free input with restricted format.

5.3 APPROACHES

Our solution to solve these problems is based on the user-centric approach where users engage in a series of interactions with the federation service to communicate their queries. There are two phases of interactions. In the first stage, a user searches the controlled value, and in the second stage, the user continues resource discovery based on the results from the first stage. We built browse capacity and interactive search interfaces based on this user-centric approach.

In the metadata harvesting approach, there are no pre-defined authority files available and the unified interface has to be built over harvested data that are added on a regular basis, so the search interface has to be adaptive to the frequently changing metadata. Figure 5.2 shows the components of the system and how it works. The harvester keeps harvesting data from sources. Another process periodically collects key metadata fields from harvested metadata, builds an index, and refreshes the search interface. Users interact with the search interface to identify their preferred controlled metadata and then execute a search.

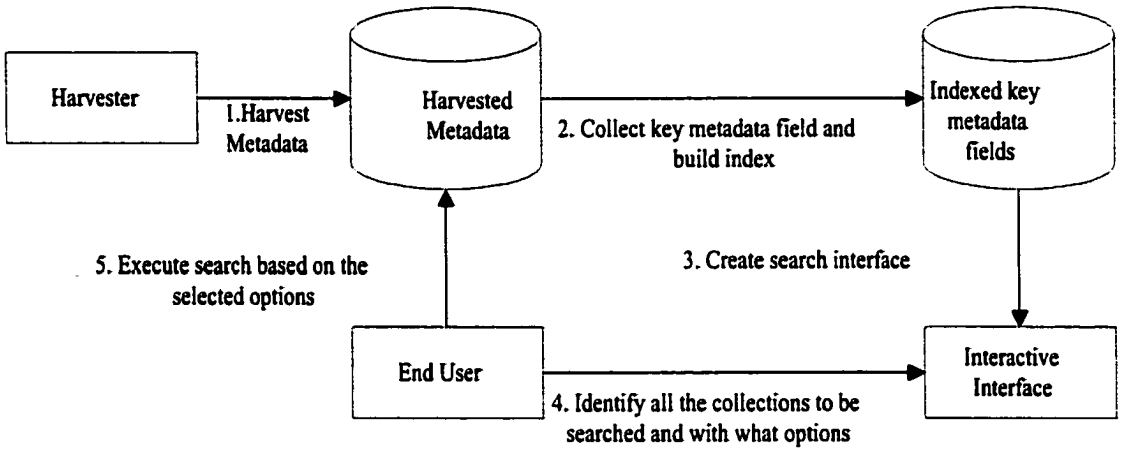

FIG. 5.2: Building a search interface based on harvested metadata

In the metadata harvesting approach, synchronization between data provider and service provider is very important. In our case, since the search interface is built over harvested data, it must be adaptive to the frequently changed data. We implemented an interface builder for this objective. The interface builder is responsible for creating a new interface when new archives and records are added. The interface builder is resource-expensive and cannot run just-in-time. Instead, it periodically builds a cached interface on the server side, and the user always sees this cached version. This interface builder does not change the layout of the interface nor the type of fields the user can choose to interact with; rather, it creates values the user can choose from when selecting a field for queries.

5.3.1 Keyword Search

Keyword search allows users to search all metadata fields across archives. It is implemented by accumulating and indexing all metadata fields together. Keyword search provides a simple and familiar way to conduct search across all archives, and the input can include Boolean operators. It is probably the only way to search across extremely variable sources without major work, but it cannot exploit the rich metadata set defined by source archives.

5.3.2 Advanced Search

Advanced search (Figure 5.3) provides a way for a user interactively to pick up the controlled values defined by specific archives via the search interface. The searcher picks an interesting archive, then the system creates a series of selectable options for each metadata field, and the user selects the exact controlled value and executes the precise search. Author, title, and abstract searches are based on user input, and the input can include Boolean operators. Archive, set, type, language and subject fields use controlled vocabularies. For search results sorting, there is a pull down menu for either type of searching that allows specifying the sorting of search results. Search results can be sorted by rank, datestamp, or archive. For the search result group, there is a pull down menu for choosing the grouping of results. Search results may be grouped according to archive, year of datestamp and subject.

In the implementation, the metadata fields are accumulated from the archives' source data. One background process periodically checks the harvested data and recreates the browse list. The advanced search capacity fits the user who is familiar with specific archives, but it does not scale well for a large number of archives because the browse list becomes too large to use.

5.3.3 Interactive Approach

In this approach, the users provide some simple initial descriptions of their queries by means of a series of keywords. The system will then present the user with contextual metadata information from those archives that have relevant records. Users can then opt to add to the search query with richer metadata elements chosen from those presented by the system. The key to this approach lies in the interaction between the user and the system. This interaction provides increasing detail to the user by obtaining detailed values from the harvested metadata. Consider the example of the subject classification maintained by arXiv for physics and a subject classification maintained by the Human Development collection. Assume that the user types "accelerator", the system would find this term in the arXiv classification under "high-energy-physics: proton accelerator," and in the Human development collection under: "university education: science: accelerated learning." Users can then choose which more closely fits their view of the subject.

Based on the user-centric approach, we have implemented an interactive interface

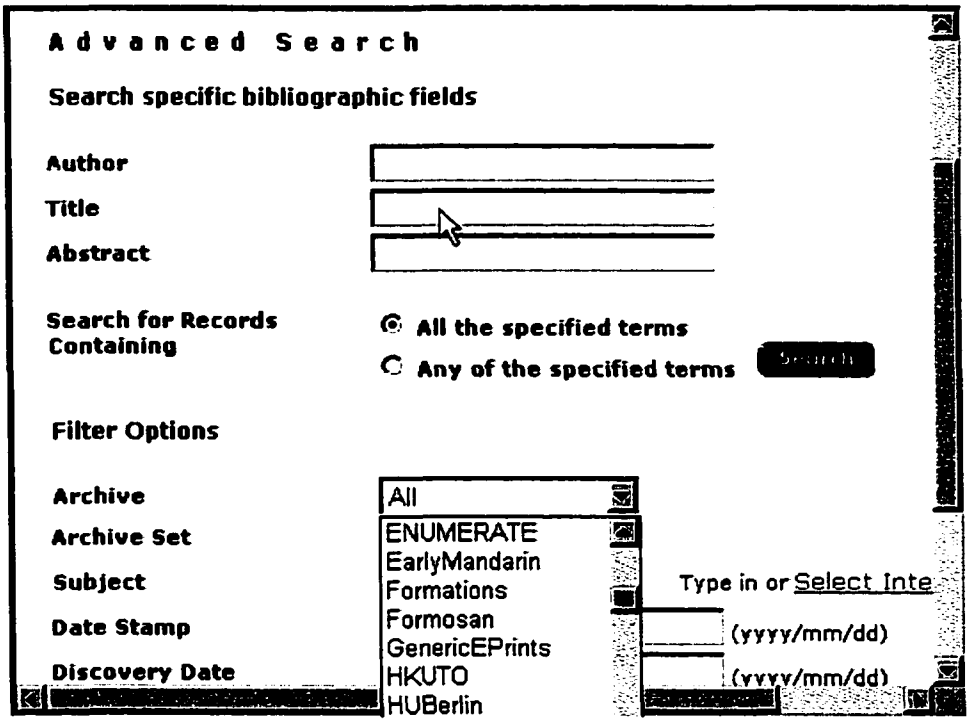

FIG. 5.3: Advanced search interface

to help users select the subject category. The interface is illustrated in Figure 5.4 (note that Figure 5.4 only shows the subject selection interface; the rest of the interface is similar to Figure 5.3). To view subject categories available in different archives, a user enters a subject keyword that closely matches the desired subject category. The input can include Boolean operators (AND, OR, NOT). Next, the user is shown matched subject categories from different archives. The user either selects one or more of the matched categories or further refines the matching list by typing more words in the field. This way, a user is able to select the desired subject categories, which are then used to construct the search query for the Arc database.

To support the interactive subject selection interface, we created a subject table in the database. The subject table is constructed by extracting the subject field(s) from each archive. The table consists of two fields: archive and subject. Once the user enters a keyword and hits enter in the subject selection interface, the keyword is sent to a servlet at the back-end. The servlet then connects to the database using JDBC [110] and searches the subject table for the keyword using an Oracle full-text search. The matched records are returned to the user and displayed in a

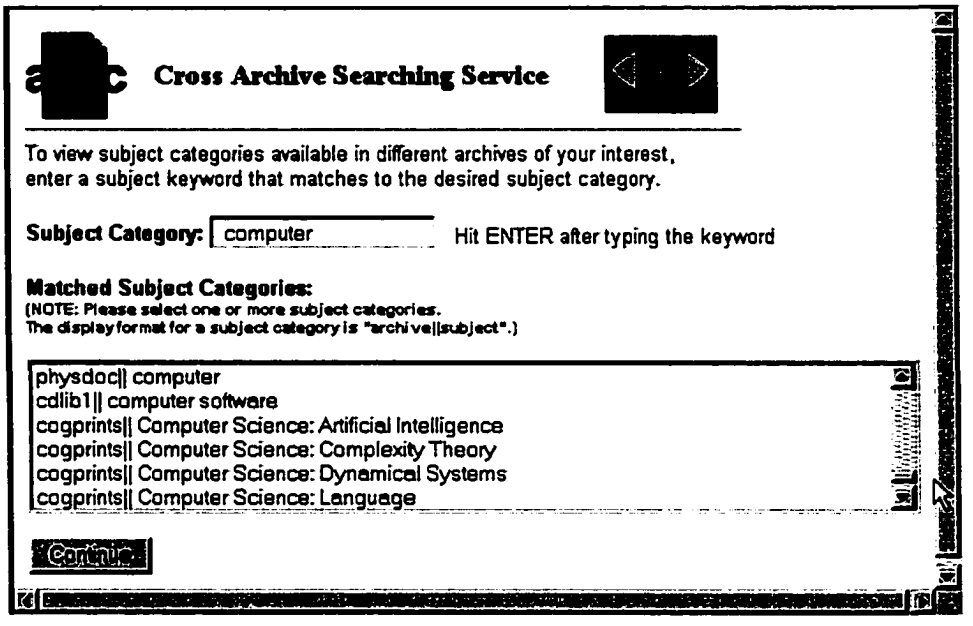

FIG. 5.4: Interactive subject selection interface

multi-selection list. The interactive subject selection interface improves the search precision by giving users the flexibility of selecting the archive and subject category of their choice.

We show the effectiveness of our approach by considering a few test cases, which demonstrate that the interactive subject selection interface improves the search precision by giving the user the flexibility to select the archive and subject category of choice. In Table 5.3, once the user searches for the subject keyword "science," the interactive subject selection interface returns 613 matched subject categories in 39 archives whose subjects include the word "science." If the user refines the query to "computer science," the search interface matches 60 subject categories in 19 archives. Next, the user selects the archives and subjects of choice.

5.3.4 Displaying the Search Result

Our experience proves that rich metadata sets not only provide a way to build a powerful search interface, but also help users to review the search results. Users have the flexibility of sorting and grouping by rank, date stamp, subject, or archive. In Figure 5.5, we see that in the result display page, the left frame shows all groups and hit numbers, and the right frame shows summary information about each document

TABLE 5.3: Number of matched archives and subjects using interactive search

Keyword typed by user	# of matched archives	# of matched subjects
science	39	613
computer science	19	60
computer science or computer engineering	20	68
computer network	4	19
physics	27	215
nuclear physics	6	32

in the selected group. Users can also traverse different pages if multiple search pages exist. When users are interested in a document, they can view the detail page and follow the link to the full-text document that resides in the data provider's repository.

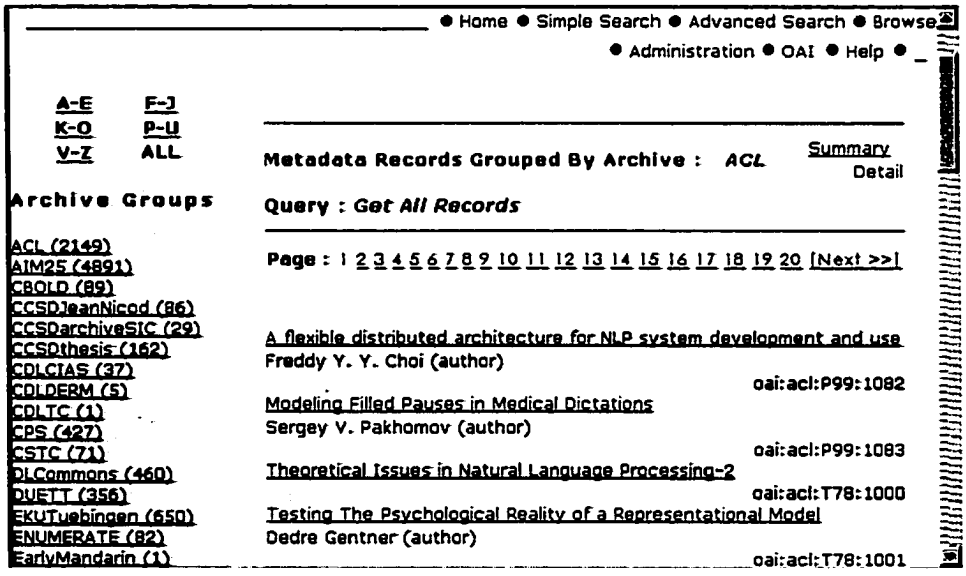

FIG. 5.5: Arc search result page

5.4 IMPLEMENTATION

5.4.1 Database Schema

OAI-PMH uses unqualified DC as the default metadata set, and all Arc services are implemented on the data provided in the DC fields. All DC attributes are saved in the database as separate fields. The archive name and sets information are also treated

as separate fields in the database for supporting search and browse functionality. In order to improve system efficiency, most fields are indexed using full-text properties of the database, such as the Oracle InterMedia Server [103] and MySQL full-text search [88]. The search engine communicates with the database using JDBC and Connection Pool [110].

5.4.2 Search Server Implementation

The search server is implemented in Java using Servlets. The components of the search server are shown in Figure 5.6.

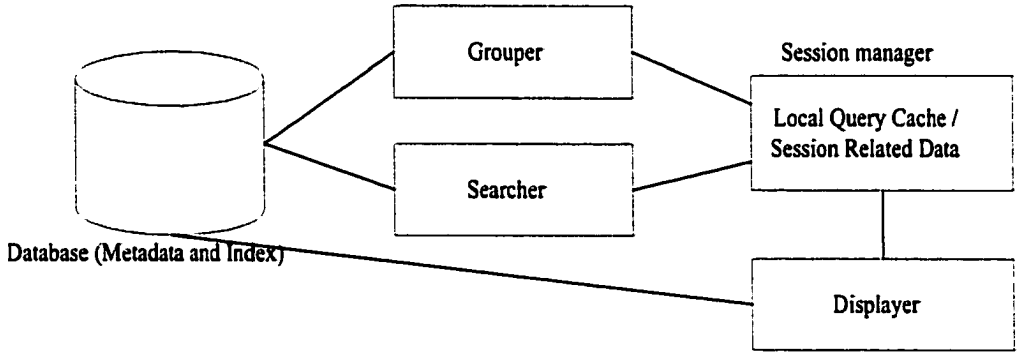

FIG. 5.6: Search engine implementation in Arc

The session manager maintains one session per user per query. It is responsible for creating new sessions for new queries (or for queries for which a session has expired). Sessions are used because queries can return a large number of results that cannot be displayed on one page. Thus, sessions are used to cache results in order to make browsing through the hits faster. The session manager receives two types of requests from the client: either a request to process a new query; or a request to retrieve another page of results for a previously submitted query. For a search request, the session manager calls the index searcher that formulates a query based on the search parameter, and submits it to the database server using JDBC, then retrieves the search results. The session manager then calls the result displayer to display the first page. For a browsing request, the session manager checks the existence of a previous session (sessions expire after a specific time of inactivity). If an expired session is referenced, a new session is created, the search re-executed, and the required page displayed. In the case where the previous session still exists, the required page is displayed based on the cached data (which may require additional access to the

database).

5.5 RELATED WORK

Although many projects, including Information Manifold [64], STARTS [41], InterOp [136], Lyceum [76], FlashPoint [77], and NCSTRL [24], have tried to provide uniform access to heterogeneous collections, almost all such systems use a distributed searching approach. These systems differ from the metadata harvesting approach. A method introduced in [40] is an interactive system for semi-structured data that helps the inexperienced user by focusing on a semi-structured graph-based database for web data. Entry Vocabulary [36] is another technology that enhances searching by mapping from the user's ordinary language to the metadata of the digital sources. French et al. [34] demonstrates a technique for mapping user queries into a controlled indexing vocabulary with the potential to radically improve document retrieval performance. Both of these methods, however, assume the existence of one unique classification scheme, which does not exist in our scenario.

5.6 DISCUSSION

We built the Arc search interface based on the approaches described above and the initial results are promising. Working with over 1M records in Arc, the advanced search interface html page is automatically daily refreshed. This interface can be accessed quickly with the speed of a conventional home Internet connection. For the interactive search, the user has the flexibility of continually refining queries so the system will scale to a larger number of data providers. After removing test queries from our own site, we found that 8053 queries were conducted in five months: among them, 6137, or 76%, were keyword searches; another 1916, or 24%, were advanced searches, indicating that users still prefer to use keyword search. In the NCSTRL project, which is based on Arc, a usability evaluation from Virginia Tech indicated:

> The interface is easy to understand and not difficult to use. The system functionality seems appropriate and the user interface is aesthetically pleasing. [118]

This study also addressed some potential usability problems that could aid future redesign and development. In another study, a focus group at Los Alamos National

Laboratory indicated that the interactive interface holds promise. The benefits of immediate feedback to the user hold great promise in enhancing the search experience as well as increasing the precision of the user's search. Making this interface more intuitive will be part of our future study. The code of Arc is released open source through SourceForge [4], and is used by metaArchive [86], NCSTRL [2], OLAC [101], and Archon [82] projects to build community-based digital libraries.

It is clear from our experiment that most archives tend to use controlled metadata, but the metadata are extremely variable from archive to archive. We have implemented two user-centric search interfaces, advanced searching and interactive searching, providing a unified search interface across heterogeneous collections and exploiting the rich controlled metadata.

CHAPTER 6

REPLICATED FEDERATION SERVICE

In the previous chapter, we demonstrated a centralized search service over harvested metadata. However, many institutions already have specific native library systems, and they would like to take advantage of metadata harvesting. Doing so would help them to integrate other data sources into their native library systems, and to share their institutional collections with third-parties.

In this chapter, we discuss the replicated approach to building federation service. This approach can be viewed as mirrored OAI-PMH repositories, where every digital library integrates harvested metadata into its native library. The consistency between these services is maintained using OAI-PMH. The replicated approach is demonstrated in the TRI (Technical Report Interchange) project taking place among several national laboratories.

The remainder of this chapter is organized as follows: We begin with a comparison of the centralized model and the replicated model. Section 6.2 presents the architecture of a system based on the replication model. Section 6.3 discusses the issues of integrating the metadata harvesting system with a native library. In Section 6.4, we present the TRI project, which is based on the replication approach.

6.1 INTRODUCTION

A replication model enables the sharing of documents housed in disparate digital libraries that have unique interfaces and search capabilities designed for their user communities. This allows a native digital library to export and ingest information from other digital libraries in a manner transparent to its user community. That is, the users access information from other digital libraries through the same native library interface the users are accustomed to using. The importance of this approach is that:

- A library may have significant investment and built-in service for its user community; it may not be realistic to change the native library system;
- It not only allows for one-time historical sharing of a corpus amongst participating libraries, it also provides for continuous updating of a native library's

collection with new documents when other OAI-PMH-compliant repositories add to their collections.

- Additionally, all libraries will always (with some tunable time delay) be consistent in having the totality of all holdings available within their own library.

There are many challenges in the replicated approach:

- Since each repository has its own DL management system and native search interface, the metadata harvesting system must be seamlessly integrated into the native DL system;
- Because each DL uses different native metadata format, we need to use a standard metadata format and there must be translation between the native and standard metadata formats in order to enable interoperability;
- The system must support new participants with limited effort, and any new participant should not adversely impact the existing installations;
- Metadata is duplicated in each DL, so when add/update/delete operations occur in one native library, the changes must be propagated to other libraries.

6.2 SYSTEM ARCHITECTURE

In the replication approach, each participant has its own user community and a local search interface allowing users to retrieve data from other library systems. A translation process in each DL is responsible for translating native metadata format to a standard metadata format and vice versa, i.e., MARC [95] tags are converted into DC and DC into MARC. The standard metadata format is saved in an OAI-PMH compliant repository, which can selectively serve metadata when an external OAI harvesting request arrives.

Since each library has its own data format and management system that is maintained by local librarians/information specialists, a file-system-based solution is a simple and flexible way for each library to import/export native metadata. The last modification time of records provides a basic mechanism to detect newly added or changed metadata. The exported native metadata is translated into unqualified DC format, which is the default used by OAI-PMH to support minimal interoperability.

The richer metadata formats such as MARC or Qualified DC provide richer semantics and support greater "precision" in search results; however, the unqualified DC is appropriate for a rapid prototype implementation.

The software that has been developed is highly modularized and can easily support new participants with minimal effort. Such software modules include:

Scheduler A tool manages and schedules various tasks in the replication system;

OAI repository A database-based system makes each library OAI-PMH-compliant;

Harvester An application issues OAI-PMH requests and collects metadata;

Translator A tool translates native metadata format in each library to a standard metadata and vice versa.

These modules are the same for all repositories. The translator requires some customization for particular libraries because its local metadata format needs to be mapped into a standard format. This can be accomplished by creating a mapping table between the metadata and the standard.

6.3 LOCAL REPOSITORY

While each site shares similar repository and harvester modules, they also have specific DL management systems and native metadata formats. We follow several guidelines in designing the local repository management in replication model:

- Each library should maintain its own management system; an identical one is not feasible or possible;
- Considering the different software/hardware environments in each library, the interface between the native library and metadata harvesting system should be portable across platforms and should be simple;
- The effort to add a new participant should be minimal.

Based on these requirements, we defined a file-system-based interface between native library and metadata harvesting system. Each library exports its native format to a configurable directory ("native" directory), and the changed/added document is

automatically marked by last modified time. A local reader periodically polls this directory and any file whose modified date is newer than the last harvesting time is translated into unqualified DC format and inserted into the OAI repository. Additionally, there is a "harvested" directory in each library; a local writer periodically checks whether any new/changed metadata is harvested from a remote repository, translates it into local format, and writes it to the "harvested" directory. Each site may have its own program that exports metadata from the local library system and a loader that reads the "harvested" directory. Such a mechanism is highly integrated with a given local repository, so its implementation is out of the control of the common modules.

The participating repositories may use different metadata formats. While it is possible to implement a one-to-one mapping for each metadata pair, the mapping complexity dramatically increases with the number of participants (n libraries would require $n(n-1)$ mappings). With a common intermediate metadata format, only $2n$ mappings are necessary. For this reason, we chose unqualified DC as the common intermediate metadata format and mapped each native metadata format to unqualified DC. With a common metadata format, however, the rich metadata element in each library may be lost, as the common metadata format is the minimal subset of all libraries. This problem can be alleviated if we adopt a richer common metadata format in the future, such as MARCXML [94] or qualified DC.

Finally, there are several approaches to address the lack of unified subject access. One way is to use a standard terminology and map each library's controlled metadata to the standard [52]. However, the granularity of subjects/keywords is significantly different among participating libraries; a unified standard is difficult to define, and two-step mapping may cause more inconsistencies. Another way is to perform an individual mapping for each subject category pair. This alternative approach is more accurate because only one-step mapping is used. Nevertheless, both approaches may introduce significant human effort to maintain the relationships. A third approach is to use an automatic classification algorithm, but the precision of this mapping is low, as we are dealing with limited metadata. The easiest approach, which is also used in our implementation, is to map all numeric subject codes into text strings using the mapping provided by the contributing organization; using this approach, the subject mapping is done only once in the source library, thus adding a new library will not influence the existing installations.

6.4 CASE STUDY

The Technical Report Interchange (TRI) project allows integration of technical report digital libraries at NASA Langley Research Center (LaRC), Los Alamos National Laboratory (LANL), Air Force Research Laboratory (AFRL), and Sandia National Laboratory (Sandia). TRI is based on the replication model as presented above.

6.4.1 Requirement

LaRC, LANL, AFRL, and Sandia all have thousands of "unclassified, unlimited" technical reports that have been scanned from paper documents or "born digital." Although these reports frequently cover complementary or collaborative research areas, it has not always been easy for one laboratory to have full access to another laboratory's reports. The laboratories would like to share access to metadata with links to full text documents initially, and eventually replicate the document collections. Each laboratory has its own report publication tracking, management, and search/retrieval systems, with varying levels of interoperability with each other. Since the libraries at these laboratories have evolved independently, they differ in the syntax and semantics of the metadata they use. In addition, the database management systems used to implement these libraries are different (Table 6.1).

TABLE 6.1: Native metadata formats and library systems

Laboratory	Native Metadata Format	Native Library System - Source	Native Library System - Destination
LaRC	MARC	BASIS+	TBD
LANL	USMARC+ Local Fields	Geac ADVANCE	Science Server
AFRL	COSATI	Sirsi STILAS	Sirsi STILAS
Sandia	MARC	Horizon	Verity

6.4.2 Typical Workflow

Figure 6.1 illustrates typical workflow in the TRI system. The MARC records in the LaRC library are exported in flat file format, translated into DC format, and deposited into the database server. The data are now OAI-PMH-compliant and are harvested by the LANL harvester. Then a local write module reads the newly

harvested data and converts them into LANL native file format. Finally, these files are loaded into the LANL native digital library.

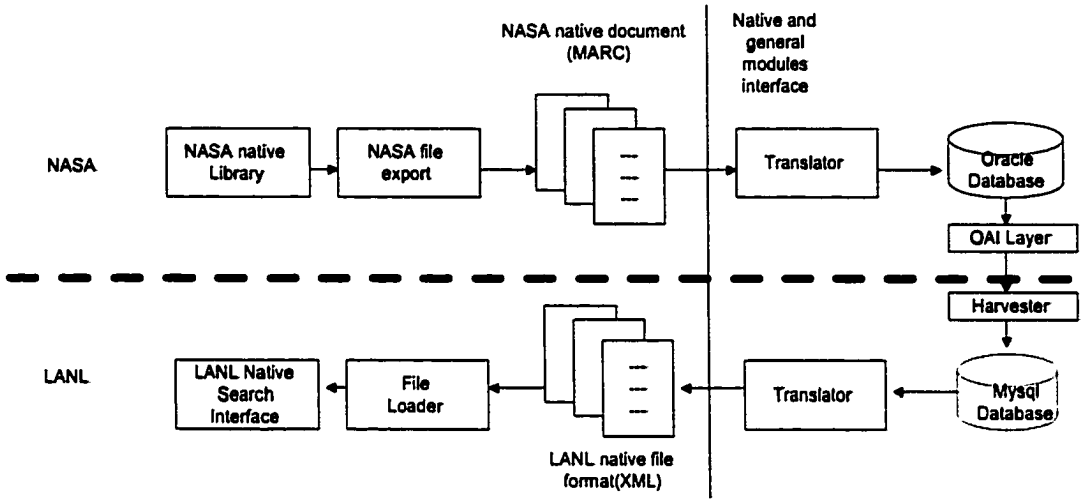

FIG. 6.1: A typical workflow – LANL shares documents from LaRC

6.4.3 Mapping Metadata Formats

Of the current four TRI participants, three (LANL, LaRC, and Sandia) use MARC in their local libraries, each with its own extensions or profiles. AFRL, on the other hand, supports COSATI. Each library exports its metadata in its own most convenient way and also defines a bi-directional mapping table (See samples in Table 6.2 and Table 6.3).

TABLE 6.2: LaRC MARC to DC mapping(Excerpt)

LaRC MARC Metadata Set	Dublin Core
D245a, D245d, D245e, D245n, D245p, D245s	Title
D513a, D513b	coverage
D520b	description
D072a,D072b(001), D650a,D659a	subject
D090a(000), D013a, D020a, D088a, D856q, 856w	identifier

TABLE 6.3: DC to Sandia mapping

Dublin Core element	Sandia Metadata Field
identifier	report numbers
identifier – URI	URL
subject	subject category codes
title	title
subject	keywords
creator	personal names
creator	corporate names
date	date
format – extent	extent
description	notes
rights	classification & dissemination

In Table 6.2, the mapping table follows the structure of the Library of Congress's MARC to DC crosswalk [96] with additional features from LaRC. In the MARC to DC mapping, the MARC file is parsed and corresponding fields are mapped to DC. In this process, some information may be lost; for example, the identifier field may be an ISSN number, technical report number, or URL. Information like ISSN and URL is clearly defined in MARC, but it will map to the undistinguished "identifier" field in unqualified DC, losing the distinctions between metadata fields.

6.4.4 Subject Mapping

In the TRI project, each library may use a different subject thesaurus and/or classification scheme. For example, LANL uses a combination of Library of Congress Subject Headings (LCSH) and subject terms from other relevant thesauri (including *International Energy: Subject Thesaurus (ETDE/PUB—2)* and its revisions). The metadata for a given LANL technical report may also include numerical subject categories or alpha-numerical report distribution codes representing a broad subject concept. Subject category code sources used by LANL include: *Energy Data Base: Subject Categories and Scope (DOE/TIC-4584-R#)* and its succeeding publication and revisions, *International Energy: Subject Categories and Scope* (ETDE/PUB—1). Report distribution category code sources include various revisions of *Program Distribution for Unclassified Scientific and Technical Reports: Instructions and Category Scope Notes (DOE/OSTI-4500).*

LaRC uses its own subject thesaurus and the NASA-SCAN system. The local library may organize the information by subject classification, making it necessary to do a subject classification mapping, such as, mapping the NASA subject code "77 Physics of Elementary Particles" to the Los Alamos report distribution code "UC-414" (Table 6.4). Subject metadata is an area where generically grouping the various subject-related metadata into a single unqualified DC data element results in loss of the source information for a given thesaurus or classification scheme, thereby complicating the subject metadata mapping.

TABLE 6.4: Subject mapping: LANL UC-414 maps to NASA SCAN 77

Digital Library	Subject Schema	Sample Subject Format
LANL	UC Report Distro Category	UC-414 sddoeur
	ETDE Subject Category	430100 edbsc
	INIS Subject Category (old)	E1610 inissc
	INIS Subject Category (new)	S43 inissc
	Text (LCSH)	Controlled formatted text
	Text (other thesauri)	Controlled formatted text
	Text (local subject heading)	Locally controlled text
NASA	SCAN	77
	Text	PHYSICS ELEMENTARY PARTICLES AND FIELDS

6.4.5 Integration with Native Library

The procedure of integrating the TRI system with a local library is highly dependent on the library's existing system. Here we describe the experience in LANL. LANL discussed various options for making TRI metadata available to local library users. One of the first suggestions, importing TRI metadata records from other institutions into the library's online catalog (the original source of exported LANL technical reports metadata), was ultimately rejected due to concerns about data mapping from the "lowest common denominator" DC format of TRI records to the MARC format required for the online catalog. It was decided to make TRI metadata records available through the library's Science Server software as a proof-of-concept test.

Science Server, a locally modified version of software provided by Science Server LLC, enables simple content management while delivering electronic journals and

IEEE Conference and Standards records directly to the desktop. At LANL, Science Server was ultimately selected for integration of and access to TRI records for the following reasons:

- It provides a unified, familiar search interface to library users;
- It offers robust indexing and searching capabilities with support for full text links (hyperlinks to technical reports);
- It permits the definition of "collections" for each harvested site, with appropriate access restrictions for the collections as needed.

Since the Science Server product was originally designed for access to journal literature, the "journal paradigm" was adapted for technical reports – with the TRI database becoming one collection within Science Server, each TRI archive institution treated as a "title," individual report years handled as volumes/issues, and the individual reports handled as "articles."

With the above paradigm in mind, it was a simple matter to design a loader for Science Server that mapped the TRI DC fields into Science Server fields. TRI's configuration tables were updated to perform "local writes," exporting the records from each archive to DC XML flat-file format. These records were then copied to test version of the Science Server system, converted from DC, and indexed. At this point, approximately 72,000 TRI metadata records are locally searchable through the test Science Server system.

6.4.6 Initial Results

In the first stage of the TRI project, LaRC and LANL installed TRI systems and each site had shared approximately 30K technical reports with each other. Both were able to automatically harvest newly published metadata from each other on a daily basis. LANL also loaded the harvested records into its native library, the Science Server, a system external to the TRI project repositories.

6.5 RELATED WORK

Web caches have been widely used to distribute load and reduce network traffic. Mirror software is designed to duplicate a directory hierarchy between two machines.

It avoids copying files unnecessarily by comparing the file timestamps and file sizes before transferring [85]. Recently, the Content Delivery Network (CDN)[75] has been widely used. However, a typical library system usually customizes data from a large scale of heterogeneous sources with authentication, thus the general Web cache may not fit well with the specific requirements of library community.

In the digital library domain, the replicated approach to build federation service has been validated by the experience within the library community in building and operating very large-scale (centralized) union catalog databases. However, this process has traditionally been done with human intervention, a typical process would involve periodic FTP downloads and/or CD-ROM delivery.

6.6 DISCUSSION

The replicated model enables the sharing of documents housed in disparate digital libraries that have unique interfaces and search capabilities designed for their user communities. It is fault-tolerant with the cost of data duplication in each repository.

During the implementation, one of the most significant problems was that unqualified DC does not match well with the sophisticated metadata formats used by the participants. The mappings, especially the subject mapping, is also difficult, and in many circumstances the semantics of original data is lost. This could be partially solved by defining a richer standard, such as qualified DC profile.

CHAPTER 7

KEPLER SERVICE

Kepler [1] service supports the concept of archivelet, which is a self-contained, self-installing software package that easily allows a researcher to create and maintain a small, OAI-PMH-compliant archive. The Kepler service poses a series of new challenges to metadata harvesting. The OAI-PMH is insufficient in such a scenario and we extend the harvest model to a "push" and hybrid "push/pull" model to support the dynamic application scenario of Kepler.

This chapter is organized as follows: From Section 7.1 to Section 7.4, we introduce the concept, architecture of the Kepler service. In Section 7.5, we discuss the repository synchronization problem using the metrics introduced in Chapter 4. Section 7.6 presents the "push" and hybrid "push/pull" models that improve the freshness and reduce update overhead in the Kepler service. In Section 7.7 we discuss the implementation.

7.1 INTRODUCTION

The Kepler service is based on OAI-PMH to support what we call "personal data providers" or "archivelets." The objectives of the Kepler service are to:

- Satisfy the need for researchers to publish results and disseminate them to a wide audience quickly, conveniently, and under the researchers' control, and possibly have the research results annotated and reviewed by peers outside the traditional and lengthy journal review process;
- Let the general public have seamless access to the totality of all such published material.

An archivelet is a self-contained, self-installing software package that easily allows a researcher to create and maintain a small, OAI-PMH-compliant archive. An OAI-PMH-compliant service provider harvests metadata from all existing archivelets and makes them available to the general public. In this vision, we see tens of thousands

[1]The Kepler service is named after the great theoretician, Johannes Kepler. According to Carl Sagan in his book Cosmos, Kepler struggled to get data from his sponsoring colleague, Tycho Brahe, the great observationalist. Only when Brahe was on his deathbed did he finally give Kepler access to all his data.

of researchers creating their own personal archives housed on a variety of machines in different network environments ranging from the sophisticated direct Internet access at the university to a home computer connected only by a modem during certain times. One or more service providers will make all these archivelets available seamlessly to any user as if they were all one large digital library.

7.2 CONCEPTUAL MODEL OF KEPLER SERVICE

We believe that two factors are critical to the success of any digital library effort: simplicity of use and control. Hence, we strongly feel that the publication tools to create an archivelet should be a downloadable, platform-independent software package that can be installed on individual workstations and PCs, rather than software that is installed by an organizational system staff. For example, the eprints.org software package exists, but its heavy footprint reflects its intended institutional-level use [30]. The archivelet needs to have an extremely easy-to-use GUI for publishing and needs to be an OAI-PMH-compliant data provider. Since we want to be as independent as possible of other software and we expect the archivelet to store relatively few objects, we shall use the native file system to store the objects rather than, for example, a database system. In supporting archivelets, the registration service takes on a bigger role than the registration server plays in regular data providers. The number of archivelets is expected to be on the order of tens of thousands, and their state in terms of availability will show great variation. Currently, the OAI registration service keeps track of OAI-PMH-compliant archives and the current registration process is mostly manual. In contrast to data providers at an organizational level, archivelets will switch more frequently between active and non-active states (e.g. a user connects with dial-up network). It will be necessary for the registration service to keep track of the state of the registered archivelets in support of higher-level services. For this, we borrow from Napster and the instant-messenger model the concept where the central server keeps track of active clients.

The current OAI-PMH framework is targeted for large data providers (at the organizational level). We propose the Kepler service based on the OAI-PMH to support archivelets that are meant for many personal publishers. The Kepler service promotes fast dissemination of technical articles by individual publishers. Moreover, it is based on interoperability standards that make it flexible so as to build higher-level services for communities sharing specific interests.

Figure 7.1 shows the four components of the Kepler service: OAI-PMH-compliant repository, publishing tool, registration service, and service provider. The OAI-PMH-compliant repository, along with the publishing tool, is targeted for individual publishers. The registration service keeps track of registered archivelets, including their state of availability. The service provider provides high-level services such as a discovery service that allows users to search for a published document among all registered archivelets.

The Kepler service supports two types of users: individual publishers using the archivelet publishing tool, and general users interested in retrieving published documents. The individual publishers interact with the publishing tool and the general users interact with a service provider and an OAI-PMH-compliant repository using a browser. In a way, the Kepler service looks very similar to a broker based Peer-to-Peer (P2P) network model (Figure 7.2). Typically, a user is both a data provider and a discovery user that accesses a service provider. Thus, the primary mode of operation might be construed as one of exchanging documents.

One key issue we needed to address in the Kepler service was the issue of scale. The intention of OAI has been to support a contributing audience consisting of few data providers, each representing a digital library with a large holding (on the order of a hundred thousand to a million objects). In the Kepler service, the opposite is true: each data provider has only a few objects (e.g., an order of a hundred) but there may be, if the Kepler service is successful, tens of thousands (or if extended to all interested persons, maybe millions) of such archivelets. The second issue we faced, normally not present in the regular OAI environment, was the issue of unreliable up-time of the machine that houses the archive(let).

7.3 ARCHITECTURE

In Figure 7.3, we show how we are addressing these issues at an architectural level. A registration server allows new archivelets to register, and the server is also used to keep track of the archivelets' active/inactive time. That is, each archivelet lets the

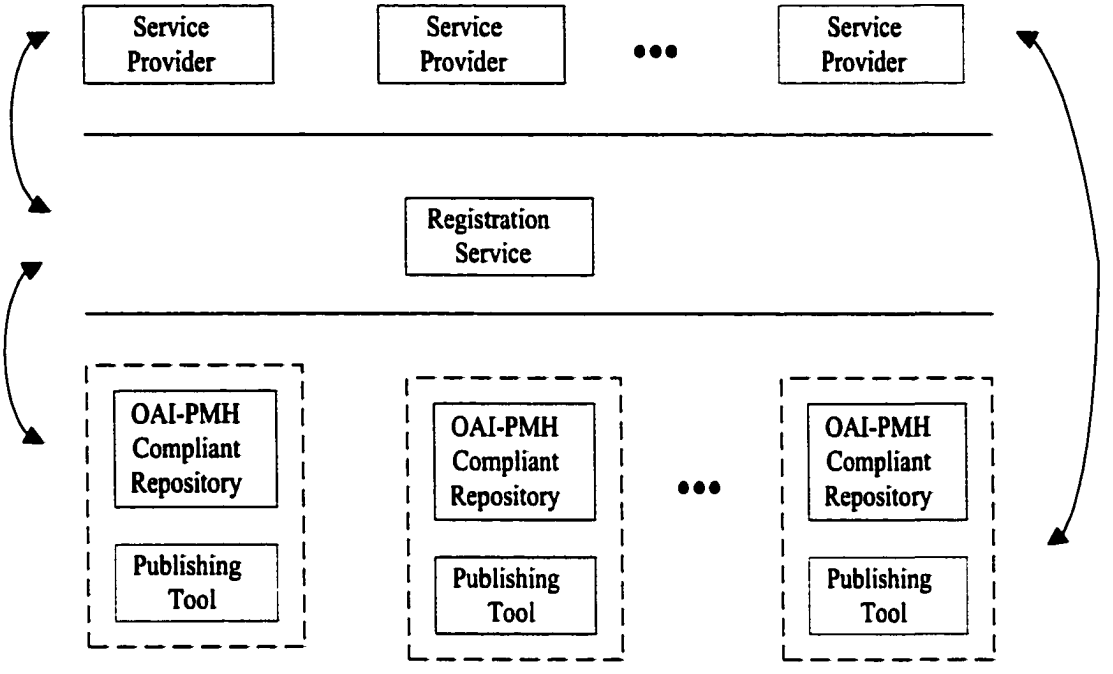

FIG. 7.1: Framework of Kepler service

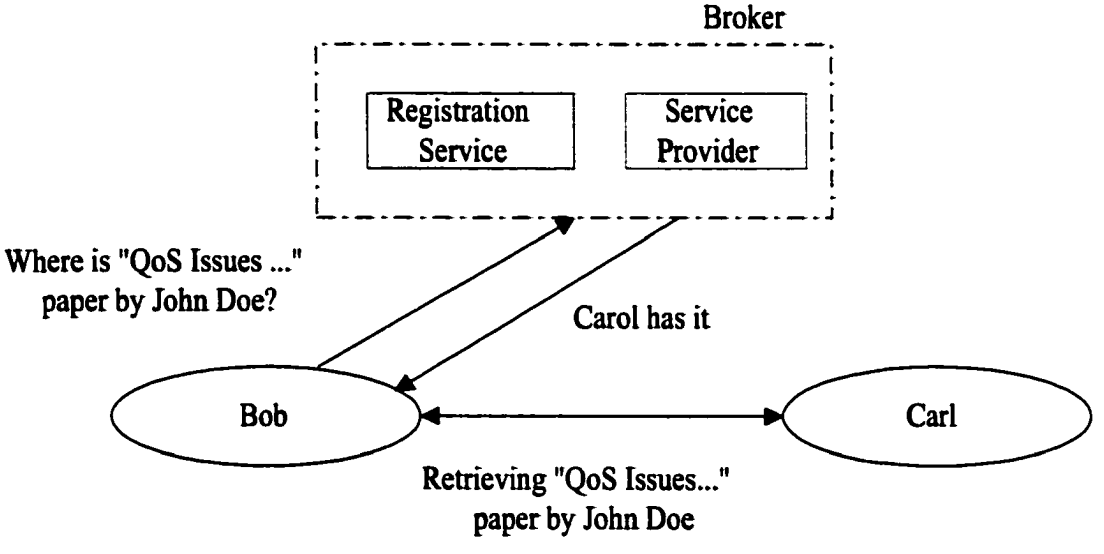

FIG. 7.2: Kepler service and Peer-to-Peer network model

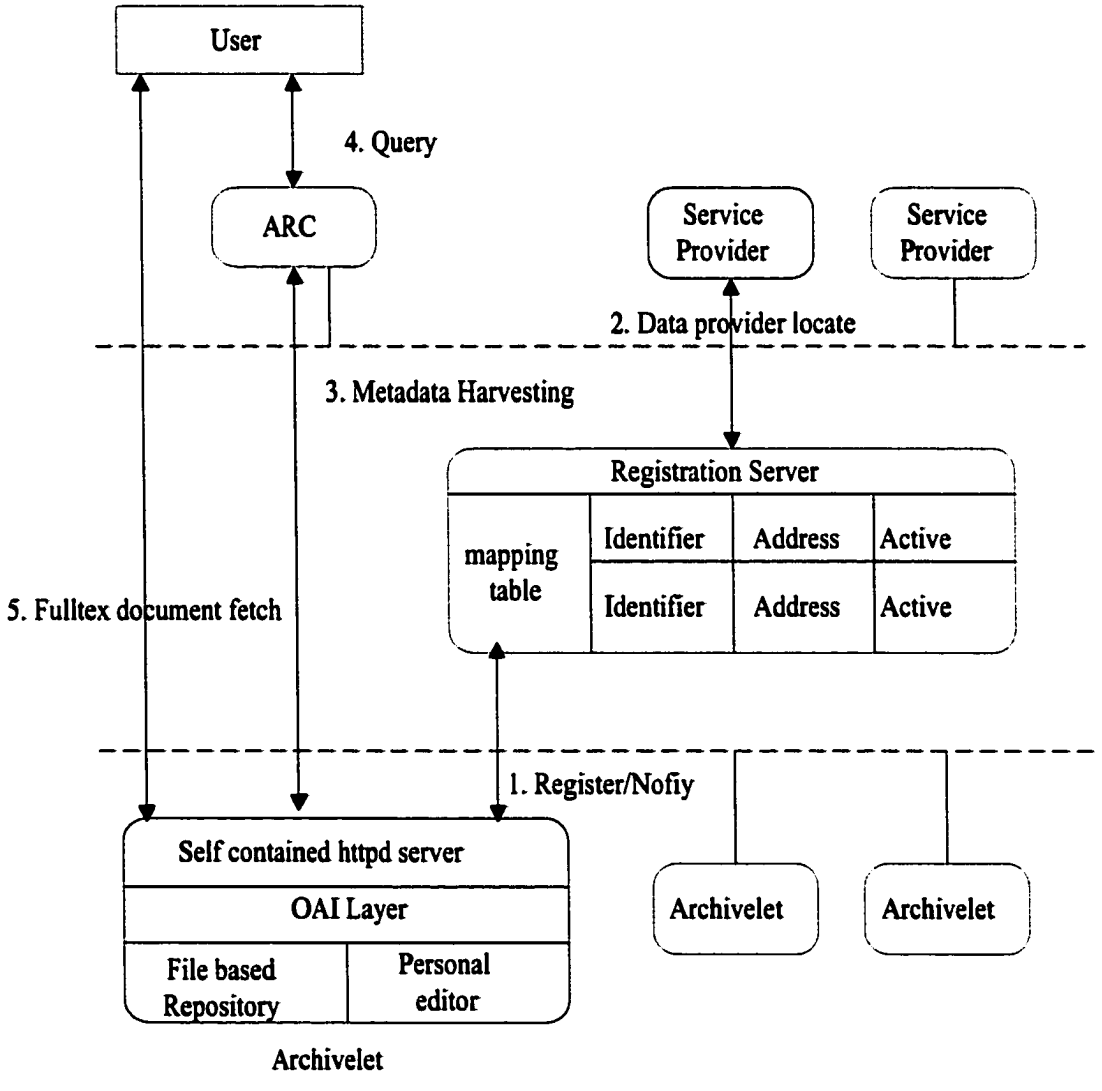

FIG. 7.3: Kepler architecture

registration server know when it goes off-line. The registration service needs to be able to handle tens of thousands of entries. A service provider uses the registration server to locate all Kepler archivelets for whatever service it wants to provide. For example, the one labeled "Arc" is a discovery service that harvests metadata from all known archivelets on a daily basis for updates and changes. Some of these services may also need to know when an archivelet is active. The information we need to keep in the mapping table of the registration for each unique archivelet identifier is its current IP address and its state.

The archivelet combines the OAI-PMH-compliant repository and the publication tool in a downloadable and self-installable component. We provide http transport as specified in the protocol, but only OAI-PMH requests are supported, not any other http actions. The basic service part of Kepler is the discovery service. Here, we want to address the issue of unreliability specific to Kepler. When a discovery user poses a query to service provider, we need to return not just the metadata of the hits matching the query, but we also need to get the state of the archivelets that contain the hits. Right now, we cache archivelets before they go off-line.

7.4 OPERATIONAL USAGE

In Figure 7.4, we show the process an archivelet must go through to register and then notify the server of its state of availability (e.g., being on- or off-line). In Figure 7.5, we have shown the flow of activities as they occur in the model, where the service provider caches the documents of the archivelets when needed so it can provide full-text fetch when a query comes for the document even though the archivelet is off-line. Notice step 6 where the service provider, and not the discovery user, fetches the document and caches it based on some historical information of the archivelet's behavior. In step 10, the service provider still goes to the archivelet to fetch the document, when a hit has been made, to make sure it has the latest version; only when the archivelet is off-line will it use the cached copy.

If we did not use caching, the process would consist of steps 1, 2, 3, 5, 7, 8, 9 and 10 from Figure 7.5. A third model can be realized by making the last step (get full-text document) consist of the following: if the document is in the cache at the service provider, return it to the discovery user; in either case record the usage pattern and, if indicated, cache it at the service provider.

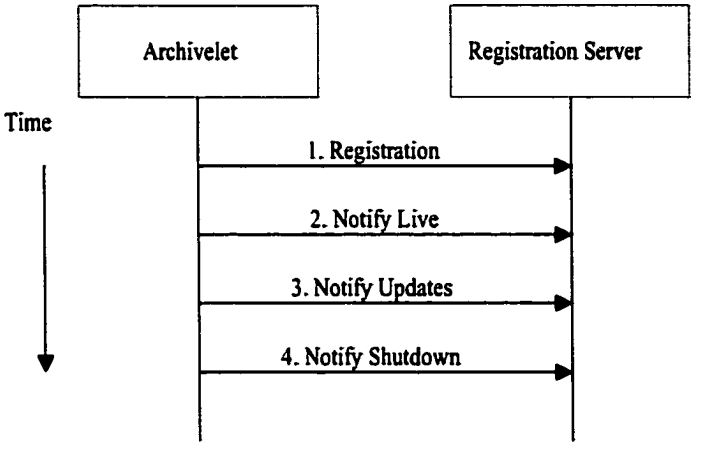

FIG. 7.4: Archivelet registration process

7.5 SYNCHRONIZATION PROBLEM IN THE KEPLER SERVICE

In Chapter 4 we discussed the update frequency in organizational level repositories, which are the designed objective of OAI-PMH. However, the synchronization model of OAI-PMH is "pull" model based; it does not fit well with a large number of unreliable repositories like archivelets in the Kepler service.

One key issue we needed to address in the Kepler service was the issue of scale. In the Kepler service, each data provider has only a few objects (e.g., an order of a hundred) but there may be, if the Kepler service is successful, tens of thousands (or if extended to all interested persons, maybe millions) of such archivelets. The second issue we faced, normally not present in the regular OAI environment, is the issue of unreliable up-time of the machine that houses the archivelet. The sparse updates must be reflected into the service provider immediately because the archivelet can be available for very short periods of time.

In Section 4.2, we define the basic model for update cost. If a harvester periodically checks all the data providers in the same frequency, the cost is:

$$C_s = \frac{TM}{l} \tag{12}$$

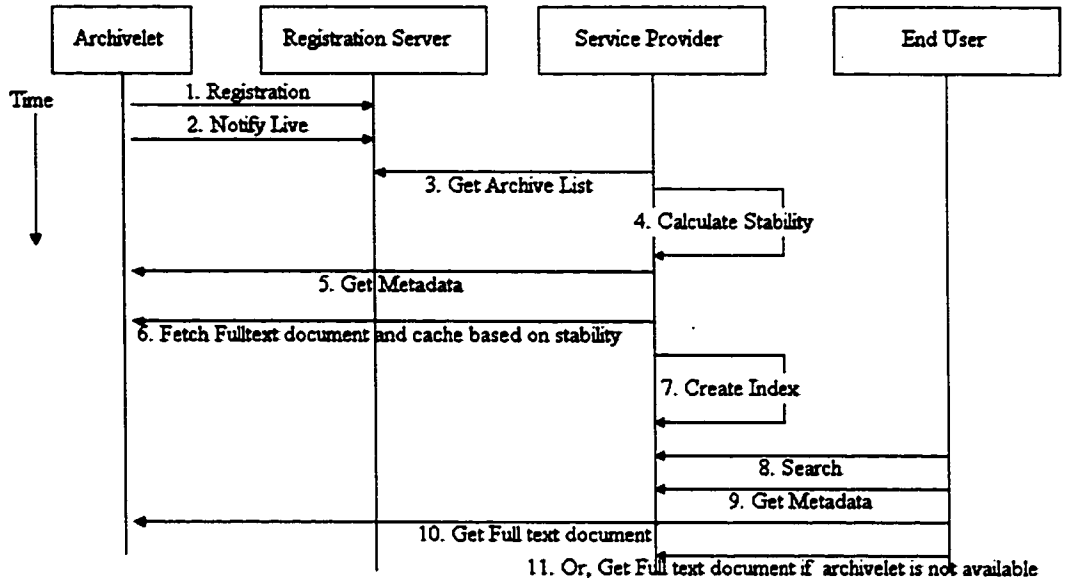

FIG. 7.5: Kepler process using cache.

In the optimal model, the harvester issues a request if and only if the data provider updates its content. In this case, the update cost is:

$$C_s = \sum_{j=1}^{M} \frac{T}{U(r_i)} \tag{13}$$

In the Kepler service, since the archivelet might be available for a very short period of time, the harvester must frequently issue requests to archivelets to discover new published data. For this reason, we expect a lower desired update latency (l), and we expect a large number of archivelets (M), each of which usually update in a very low update frequency. The basic model will not scale, but the optimal model provides much better performance. This is illustrated in Example 7.1.

Example 7.1. Assume 10,000 archivelets exist, and each one is active 30 minutes every day on the average. One archivelet publishes one new document every month. In order to keep data fresh, the harvester shall harvest each archivelet every 30 minutes. Using the basic model, the update cost per day is:

$$C_s = \frac{24 \cdot 60 \cdot 10000}{30} = 480,000$$

Using the optimal model, the update cost per day is:

$$C_s = \sum_{j=1}^{M} \frac{T}{U(r_i)} = \frac{24 \cdot 60 \cdot 10000}{30 \cdot 24 \cdot 60} = 333$$

The optimal model's performance is better than that of the basic model. However, it is difficult to predict up and update time of individual data providers. We propose a pull/push hybrid model to improve synchronization in the Kepler service.

7.6 SYNCHRONIZATION APPROACH FOR KEPLER SERVICE

In this section, we summarize the synchronization models in a systematic way in the Kepler service. There are three content delivery models to implement synchronization between data providers and service providers.

Pull Retrieval without prior coordination (e.g., as used by current robots and OAI-PMH)

Hybrid Push/Pull Retrieval after notification

Push Notification followed by a provider push.

In the pull model, a service provider requests immediate, synchronous delivery. The pull model is widely used in harvesting solutions such as Web Robots and OAI-PMH. The push and hybrid models require that a service provider actively listen for the notification, which adds implementation complexity to the service provider. Both models also require that a data provider keep a record of subscribed service providers and send replies in an asynchronous fashion. All three models individually can fulfill the metadata harvest task.

The major problem of the pull model is that service providers instead of data providers drive the harvest. This leads to a serious "Update Overhead" problem. Frequent crawling has to be done to synchronize the data providers and service providers. It is inefficient if the data providers seldom change during a harvest interval. On the other hand, without frequent crawling, service providers may become inconsistent with data providers. In the Kepler service, the number of archivelets potentially is very large and update frequency is very low. Sparse updates should be reflected on the service providers' side immediately because of the unstable nature of archivelets.

From our experiments, we conclude that a pull model is not suitable to implement the Kepler concepts built around the federation of individual archivelets. We propose to use asynchronous models for metadata harvesting. We believe the "Hybrid" and "Push" models will result in more efficient synchronization in the Kepler.

We extend the OAI-PMH with two additional verbs on the service provider side and one verb on the data provider side as a way to optimize the functioning of the OAI-PMH. The AddFriend and Notify verbs support push/pull hybrid model (Figure 7.6). The AddFriend verb informs the service provider of the existence of a data provider. The service provider then responds whether or not it will accept a push request. The Notify verb informs the service provider that a data provider is up/down or some new data is available. In addition, a PushMetadata verb is added to support the push model and to allow the data provider to push metadata directly to the service provider side. The process of pushing metadata does not rely on OAI-PMH.

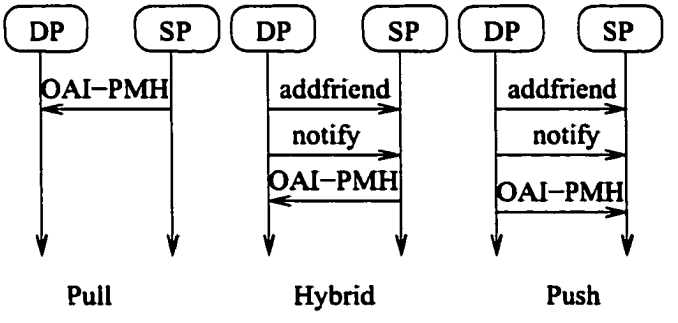

FIG. 7.6: Push, push and hybrid model

The syntax of the three added verbs follows the same HTTP request/XML response model as OAI-PMH, the request is restricted to HTTP POST to implement the metadata push.

7.6.1 Add a Friend

Summary: Request to be added as a friend

Request:
$?verb = AddFriend\&id = \&baseURL =$

Arguments:
id: Identifier of the archivelet.
baseURL: the baseURL to which the OAI-PMH requests are issued.

Response:
$< success > yes/no < /success >$
$< push > yes/no < /push >$

Explanation: The objective is to let the service provider know of the archivelet's existence. The service provider should decide whether or not to accept a push-based request based on whether (1) it has enough resources; and (2) the data provider is behind a firewall or NAT. This could be done by checking whether an HTTP request has the same IP address as the hostname in the baseURL. The service provider can later issue OAI-PMH requests to a registered archivelet or accept metadata pushed from it.

7.6.2 Notify

Summary: Notify is used for major events of an archivelet, including startup, shutdown and document update.

Request:
$?verb = Notify\&event = [start/stop/update]\&id = \&baseURL =$

Arguments:
event: the event to be notified
id: identifier of the archivelet.
baseURL: baseURL at the time of notification

Response:
$< success > yes/no < /success >$

Explanation: The archivelet should notify all registered service providers about its status when it starts, stops, or new documents are added and existing documents are

changed/deleted. The baseURL is re-submitted because the archivelet is inherently unstable and it may listen to different IP addresses and ports at each startup, as when an archivelet is connected by a dial-up network.

7.6.3 Push Metadata

Summary: Push Metadata

Request:

$?verb = PushMetadata\&contents =$

Parameters:
Contents: the pushed metadata.

Response:
$< success > yes/no < /success >$

Explanation: The archivelet pushes metadata to the service provider. The whole metadata set is identical to the GetRecord response in OAI-PMH and is put in the contents field. This way, the metadata exchanges could bypass the firewall and/or NAT.

7.7 IMPLEMENTATION

The prototype system we have implemented as a first feasibility step uses an LDAP-based registration system [134]. For the service provider, we have used a modified Arc. Arc uses an Oracle database to create the index for the harvested metadata. Using the OAI-PMH, the service provider harvests daily, asking for updates from the last successful harvest. It keeps a list of successful harvests with the registration service. The location of all registered archivelets is made available upon request from the registration service.

The publication tools consist of a simple display of the archive and a tool to specify metadata and upload files into the archivelet. The publication tools, together

Source DL:	Stanford DLI2
Identifier	http://171.64.75.199:2048/SIDL-WP-2001-0151.pdf
cached version:	oai.kepler.Stanford.DLI2.SIDL-WP-2001-0151.pdf
Title:	Multicasting a Web Repository
Creator:	Wang Lam
Creator:	Hector Garcia-Molina
Subject:	Computer Science
Description:	Web crawlers generate significant loads on Web servers, and are difficult to operate.
Contributor:	Multicasting a Web Repository
Discovery:	2001-02-01
Type:	submission to publication / working draft
Language:	en
Rights:	reserved
[redacted]	
ID:	oai:kepler/Stanford DLI2:SIDL-WP-2001-0151
DateStamp:	2001-05-07

FIG. 7.7: Cached document in Kepler service

with the client for the automatic registration process and for the interaction with the service provider (the OAI-PMH layer and the simplified web server), have been packaged together with the Java virtual machine (and necessary Swing classes) into a self-installing file that can be downloaded from the Kepler home page. Finally, in Figure 7.7, we show the display of one particular object found through Kepler service provider. The display shows selected metadata together with the URL of the document in the archivelet that will be either served by the web server of the archivelet if it is on-line, or by the service provider otherwise. However, in the latter case it will only be a cached copy and may not be the most recent one.

7.8 RELATED WORK

The driving force behind the development of Kepler was the need for author self-archiving. The resulting distributed archives can be federated into global "virtual" archives, citation-linked, and freely navigable by all. The author self-archiving could take the form of subject-, institution-, personal- , and publisher-based. ArXiv.org is a very successful subject-based self-archiving service. Since its inception in 1991, it has become a major forum for dissemination of results in physics and mathematics. The total number of submissions received during the first 10 years of operation is roughly 170,000. The submission rate continues to increase. ArXiv.org is based on a centralized server to which authors submit documents [39]. In contrast, e-Prints.org software package is an open source software created to support the institution-based self-archiving. It supports OAI-PMH and can be harvested by federation services [30]. Another widely-adopted self-archiving model is through a personal or institutional web site. A service such as ResearchIndex [38] retrieves research articles from these web sites and automatically builds the bibliographic and reference data from the articles. There is no interoperability or structured model underlying ResearchIndex, so the completeness of the collection is not guaranteed. The precision of search engines not aware of metadata is generally not as good as that of metadata-aware search engines. We believe all these methods are likely to remain in simultaneous use. Kepler provides another possible implementation to the vision of author self-archiving.

Peer-to-Peer networks represent a style of networking in which a group of computers can communicate directly with one another rather than through a central server. File sharing P2P networks such as Napster, Gnutella, Freenet and Fasttrack

have been widely used [104]. In the digital library domain, the LOCKSS (Lots Of Copies Keep Stuff Safe) is a prototype of a P2P network designed to preserve access to scientific journals published on the Web by coordinating libraries and publishers [113], the EDUTELLA [89, 1] uses JXTA framework [65] to exchange RDF metadata. While earlier P2P frameworks like Gnutella suffer from scalability problems due to their whole decentralized architecture [111], the recent arrival of FastTrack and openFT improves scalability by the introduction of a 2-tier system [37]. The first tier, referred to as SuperNodes, consists of fast connections to the network, and the second tier, referred to as Nodes consists of slower connections to the network. The SuperNodes index information distributed in Nodes, and they also provide routing and query services. This implements both a more scalable and reliable service.

Microsoft Channel Definition Format (CDF) allows web publishers to personalize and streamline the delivery of information to their customers [28]. The synchronization problem in web crawling is also discussed in report of the W3C Distributed Indexing/Searching Workshop [115].

7.9 DISCUSSION

The Kepler service is significantly different from other file sharing P2P networks, it is based on the OAI-PMH and provides more efficient repository synchronization mechanism and better data freshness. The Kepler service has been deployed for more than one year at the Old Dominion University. Until May 2002, there have been 1181 downloads of the Kepler software. Kepler service provides a novel way to support author self-archiving, a number of services, such as peer-review, recommendation, and annotation services, can be built over the Kepler service.

CHAPTER 8

PROXY, GATEWAY AND CACHE SERVICE

The current and emerging applications based on metadata harvesting require a scalable and reliable infrastructure to support them. This chapter introduces the concepts of OAI-PMH proxies, OAI-PMH caches, and OAI-PMH gateways as tools for the optimization of the functioning of the data provider/service provider model underlying the OAI-PMH. The goal is to achieve interoperability, scalability, and reliability of OAI-PMH services. It also shows how various applications can exploit the services included in the proposed infrastructure. The concept of OAI-PMH proxy, cache, and gateway service is developed and refined by the author in cooperation with Tim Brody in the IAM (Intelligence, Agents, and Multimedia) group at Southampton University. The remainder of this chapter is organized as follows:

- Section 8.1 summarizes the problems faced in the metadata harvesting system.
- In Section 8.2, we present an overview of the optimized model and introduce the concept of OAI-PMH proxy, cache, and gateway.
- We discuss each of the subsystems in Sections 8.3–8.5.
- In Section 8.6, we discuss several working applications.
- Section 8.7 summarizes related works.

8.1 INTRODUCTION

The OAI-PMH uses HTTP-based request-response communication between a data provider and a service provider. The XML-formatted metadata is encoded in the HTTP response, which makes on-demand services possible. Using OAI-PMH, one data provider may be harvested by any number of service providers, each possibly implementing different services. These service providers can interoperate using the multiple-resolution capability (one identifier is resolved to multiple instances) based on unique identifiers. In OAI-PMH, the metadata is distributed and replicated in many different places and potentially provides a highly redundant and fault-tolerant system.

In the development of OAI-PMH based applications, we notice several challenges faced by OAI-PMH based applications, namely:

Data Provider and Metadata Quality During the testing of data providers, numerous problems were found. Not all archives strictly follow the OAI-PMH; many have XML syntax and encoding problems. With OAI-PMH, syntax for metadata is strictly defined (XML schema validation), and problems still appear. This problem has been discussed, and one solution is a robust harvester as described in Chapter 3.

Server Availability The stability and service from data providers are difficult to predict since many factors may influence data provider availability and efficiency [92]. If a large data provider is periodically unavailable, this can be a serious problem for harvesting. As also discussed in Chapter 3, recent research points out that a significant number of data providers could not be harvested [51].

Scalability OAI-PMH harvesting is resource-expensive to data providers, partially because the HTTP responses are dynamically generated, and data providers may need to keep current harvest sessions (harvesting may take several days for a large data set). Besides steps taken by individual data providers to improve services, a general infrastructure is required.

Linking Across Service Providers In OAI-PMH, data providers may be harvested by many service providers, each providing different services for the same record. Cross-service linking and data sharing can be achieved by using the unique OAI identifiers. Unique identifiers also allow the detection of record duplication.

In this Chapter we discuss an effort that addresses these problems using a variety of techniques. We present an architecture to achieve interoperability, scalability, and reliability by optimizing dataflow in the OAI-PMH model. This architecture introduces an OAI-PMH *proxy* concept that could improve data provider quality by fixing implementation problems just in time. An OAI-PMH *cache* service improves data availability and avoids bottlenecks through hierarchical harvesting. An OAI-PMH *gateway* translates operations from other resource discovery systems into operations

in OAI-PMH and vice versa. We also discuss how to build a series of services such as cross-archive linking based on the suggested architecture.

8.2 OVERVIEW

The need for an optimized model is motivated by the major challenges faced in the basic OAI-PMH model. The basic structure of OAI-PMH supports two roles: the service provider and the data provider. Multiple service providers may harvest multiple data providers at the same time. If one data provider has implementation problems (e.g., XML encoding), all service providers have to address these problems. If one data provider is unavailable, all service providers have to wait until the data provider comes up again, even if some service providers have already cached the data from the data provider.

Figure 8.1 illustrates the optimized model based on hierarchical harvesting. An OAI-PMH proxy dynamically forwards OAI requests to data providers. For example, it can dynamically fix common XML encoding errors and translate between different OAI-PMH versions. An OAI-PMH cache caches metadata and can filter and refine them before exposing them to service providers. It also serves as a simple cache that reduces the load on source data providers and improves server availability. An OAI-PMH gateway can convert the OAI-PMH to other protocols and applications. For example, the gateway could provide value-added services like automatic citation extraction, or conversion between different protocols (e.g. SOAP [13]) and OAI-PMH. An end-user service will present various services such as search and citation linking. Figure 8.1 illustrates how each layer may fetch data from any of its lower layers, depending on availability and service type.

8.3 OAI-PMH PROXY

From a harvester's point of view, the most convenient solution to incorrectly implemented data providers is to place a layer (i.e. a proxy) over source repositories that can be trusted to provide correct responses to the harvester's requests. The proxy can protect the network from erroneous and malicious clients, for example, a proxy can serve as the single access point for the outside world to data providers inside a firewall.

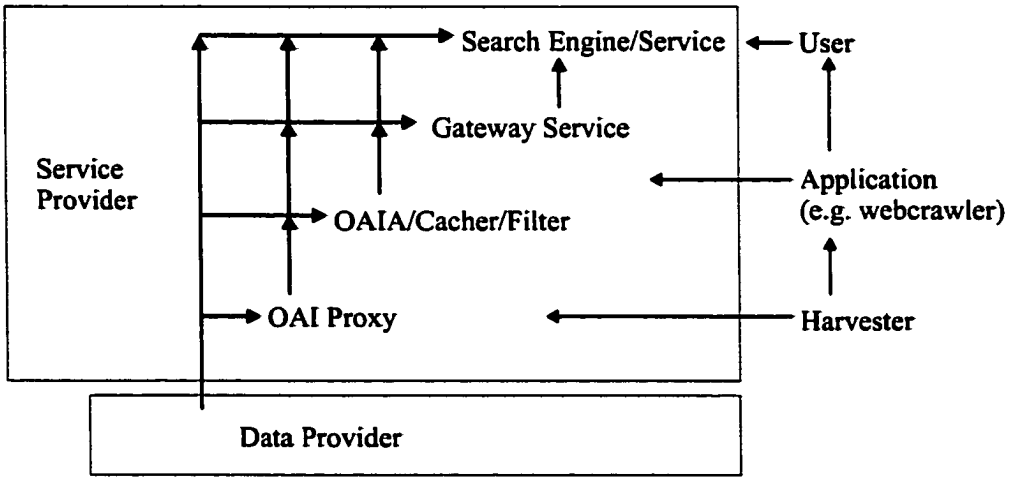

FIG. 8.1: Hierarchical harvesting model

An OAI-PMH proxy can either act as an HTTP proxy or be OAI-PMH-specific. As an HTTP proxy, it effectively becomes a transparent layer accepting HTTP requests and responding with HTTP responses. As an OAI-PMH-specific proxy, it must re-write request URLs; for an example of mapping a given subdirectory to a source base URL, see Table 8.1.

TABLE 8.1: OAI-PMH-specific style proxy requests

Request URL	Wrapped URL
Oai-proxy/cgi/proxy/cogprints	cogprints.soton.ac.uk/perl/oai
Oai-proxy/cgi/proxy/bmc	www.biomedcentral.com/oai/1.1/bmcoai.asp

An OAI-PMH proxy will fix the following errors:

Character Encoding OAI-PMH uses the Unicode's UTF-8 character encoding to support international character sets by using multiple bytes for non-English characters. As an OAI-PMH response is received from a repository, the proxy can replace any faulty character encoding that would normally cause an XML parser to fail.

XML Encoding The mark-up characters used in XML must be encoded when used in string data. Similar to recent web browsers, the proxy can use heuristics to determine whether a mark-up character is actually part of mark-up, or should be encoded.

how often the service provider should check that repository for updates.

A hierarchical OAIA structure can reduce this problem by avoiding duplication of the efforts of many service providers. For example, a service provider may want to provide an index of all the music manuscript repositories of a given country. That service provider can then expose the aggregated collection to an international service provider, saving the international service provider the effort of harvesting from every repository in every country.

8.4.3 Advantages Over HTTP Caching

An OAIA is similar to an HTTP cache; specifically, they both distribute load away from the server (the data provider) and closer to the client (the service provider).

An OAIA is, however, an active cache - it requests new records from the known repositories in advance. This means a repository's records will always be available from the cache to downstream harvesters, even if the repository itself is unavailable.

An important role for an aggregator is providing quick access to many smaller collections. By prefetching records from its source repositories, an OAIA can provide a downstream harvester with all the aggregated records in one session.

8.4.4 Datestamping

Incremental harvesting in OAI-PMH uses datestamping; that is, a harvester only needs to request records that are new or have changed since the last time it checked the repository.

With hierarchical harvesting, the OAIA must update the datestamp when it harvests a record – because the record is "new" to the OAIA. When a downstream harvester harvests from the OAIA, it will receive all the new records in the OAIA, even if the original datestamp of the record was before the date of harvest.

8.4.5 Identifiers

An OAIA can either maintain or change the *oai-identifiers* for records that it harvests (and re-exports).

By maintaining the record's *oai-identifier*, the OAIA can become a nearly transparent layer in a hierarchical system (nearly transparent, because it introduces a delay between a record being created by data providers, and it being harvested from

the OAIA). Maintaining *oai-identifiers* allows a harvester to change sources without causing inconsistent records.

OAI-PMH 2.0 introduces a provenance schema for use in the optional "about" field of records ("about fields" are for things that describe the metadata record). This schema allows the history of the record to be recorded: it stores the details of each OAI-PMH service that the record passes through as it goes down the hierarchy, from repository to eventual end-user service.

Provenance can be used to check the originality of the record, identify the change history of the record when it travels around the system, and extract the original datestamp.

8.4.6 Identifier Collisions

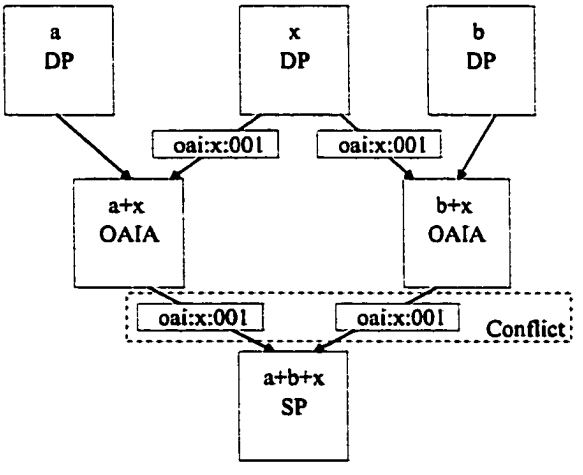

FIG. 8.2: Identifier conflict in hierarchical harvesting

When there is more than one path from a data provider to a service provider, the service provider may need to resolve a collision between two or more records with the same *oai-identifier*.

Figure 8.2 shows how one record (with a unique *oai-identifier*) may appear twice to a harvester. Three repositories, a, b, and x, are being harvested by two aggregators, a+x and b+x. When the service provider a+b+x harvests from a+x and b+x, it will get duplicates for every record from the data provider x.

To resolve collisions, a service provider can either store both records or attempt to discard one. The following are some possible policies for record discarding:

Duplicate Records If colliding records are the same, or similar, the duplicates could be safely discarded.

Trusted Sources The service provider in Figure 8.2 may, for example, trust OAIA b+x more than OAIA a+x, in which case the service provider could discard or overwrite any colliding records harvested from OAIA a+x.

Most Recent It may be possible to distinguish the most recent (and hence most authoritative) record using the datestamps given by the aggregator's provenance data (e.g., OAIAs a+x and b+x in Figure 8.2).

8.5 OAI-PMH GATEWAY, VALUE-ADDED SERVICES

A gateway between two resource discovery systems translates operations from one system into operations in another system. An OAI-PMH gateway is responsible for converting OAI-PMH for use by other applications and vice-versa. Unlike the OAI-PMH cache and proxy, the gateway service does not necessarily retain the original data or OAI-PMH interface. The objective of a gateway is to extend OAI-PMH-compliant repositories to other protocols or applications; for example:

Protocol Broker A protocol broker could convert HTTP-based OAI-PMH requests to SOAP messages or extend OAI-PMH to a Web Service model.

Gateway for Crawlers A gateway for web crawlers could translate OAI-PMH-compliant repositories to a series of linked web pages, which allows web search engines that do not support the OAI-PMH to index the "Deep Web" contained within OAI-PMH-compliant repositories.

Value-Added Services A gateway could cache the full-text document and then provide value-added services, such as citation extraction, which can then be re-exposed through its own OAI-PMH interface.

Subject Gateway A subject gateway could help build a topic-specific service by harvesting records and then exposing them by subject criteria.

A gateway service may create a large overhead for data providers, especially if the gateway is designed to serve machine-based applications (e.g., web crawlers). This situation is where the OAI-PMH cache is relevant because the hierarchical structure

of the OAI-PMH cache will reduce the overhead for the source data providers to a minimum. At the same time, the gateway service itself could use flow control mechanisms, such as HTTP-throttle software, to reduce the overhead.

8.6 CASE STUDY

Several systems have been built to demonstrate the concepts of OAI-PMH proxy, cache, gateway services, and end-user services.

8.6.1 OAI-PMH Proxy

Our first experiment is an OAI-PMH-specific proxy that takes a URL of the format:

http: //foo.org/OAIProxy/{repositoryid}?{oai verb}

Its function is to filter XML encoding errors. This proxy relies on a preexisting mapping table between an OAI-PMH repository ID and a base URL. When an OAI-PMH request is issued, the proxy forwards the request to the corresponding data provider. The XML response is parsed by a DOM parser [133]; if any XML encoding errors exist, the proxy tries to delete bad records based on the detailed error message from the DOM parser. The proxy then returns the corrected XML response.

8.6.2 OAI-PMH Aggregation/Caching/Filtering

An OAI-PMH cache service has been explored in several experiments, including OAI Aggregator, Arc, and CiteBase. Both Arc and Citebase act as data providers disseminating Dublin Core metadata harvested from other data providers.

OAIA is specially designed to mirror OAI-PMH repositories. OAIA creates a duplicate of all available data from the source repositories, excluding the set hierarchy (with OAI-PMH 1.x, the set hierarchy can only be ascertained through exhaustive querying of each set).

OAIA is designed to facilitate OAI-PMH gateway services, which rely on fast, reliable access to OAI-PMH repositories. OAIA also acts as a gateway from legacy OAI-PMH implementations (1.x) to the most recent version of the OAI-PMH (2.0). As well as being able to harvest from any repository that has been OAI-PMH-compliant, OAIA converts the required Dublin Core metadata format to the most recent OAI-PMH version.

Given a new repository URL, OAIA issues an Identify request. The Identify response is stored so a service provider can retrieve from OAIA the source repository's data policies, etc. A ListMetadataFormats request is issued to find out which metadata formats are supported. Each record is then requested for each metadata format (either using the batch command ListRecords, or GetRecord, depending on the repository's reliability). The metadata is stored as it was received from the source repository, ready for a service provider to harvest. The record's datestamp is changed to the time the record was harvested by OAIA.

OAIA provides two views to harvesters of the records it has collected: a view of the aggregated collection, or a view of the individual repositories produced by a wrapped URL.

When an aggregated OAIA collection receives a ListMetadataFormats request, it lists all the metadata formats used by any of the harvested repositories (which may include variants of the same metadata format). When the same request is made to a wrapped repository, it lists only the metadata formats supported by that repository.

8.6.3 DP9 Gateway Service

DP9 is a gateway service that allows general search engines, (e.g. Google, Inktomi) to index OAI-PMH-compliant archives. DP9 does this by providing persistent URLs for records and converting them to OAI-PMH queries against the appropriate repository when the URL is requested. This service allows search engines that do not support the OAI-PMH to index the "deep Web" contained within OAI-PMH-compliant repositories.

Many DLs and databases are closed to general-purpose Web crawlers. The "deep Web" or "invisible Web" refers to vast repositories of content, such as documents in online databases, that general-purpose Web crawlers cannot reach. The deep Web content is estimated at 500 times that of the surface Web, yet has remained mostly untapped due to the limitations of traditional search engines [7]. On the other hand, many researchers use general-purpose search engines to locate research papers more frequently than they use specific DLs. A study about ResearchIndex query logs showed that only about 6% of the total number of sessions started with a search query from ResearchIndex itself, the majority of the sessions have been initiated by linking through a search engine such as Altavista or Google [78].

DP9 [26] is an open source gateway service that allows general search engines,

(e.g. Google, Inktomi) to index OAI-PMH-compliant archives. DP9 does this by providing consistent URLs for repository records, and converting them to OAI-PMH queries against the appropriate repository when the URL is requested. This allows search engines that do not support the OAI-PMH to index the "deep Web" contained within OAI-PMH-compliant repositories.

Internet search engines cannot index OAI-PMH collections since they are not aware of the OAI-PMH. We introduce an OAI-PMH gateway architecture to address this problem. Typically, a Web crawler indexes a web site by starting with a base HTML page and by following the links on this page to retrieve deeper pages on the Web site. To support this for an OAI-PMH collection, which only responds to OAI-PMH requests (and only in XML), we begin by dynamically creating a starting HTML page for an OAI-PMH collection. Although an individual data provider may have its own mechanism of creating this page, DP9 provides a general solution that fits all OAI-PMH compliant data providers. In DP9, the starting page for a data provider would be constructed by issuing an OAI-PMH ListIdentifier request and translating the response into a HTML format containing a series of links. A link on this HTML page, when invoked, would result in another OAI-PMH GetRecord request for a specific identifier. Again, the response for such a request would be translated into an HTML page with appropriate links. In other words, an HTML page presented to a Web crawler is a result of an OAI-PMH request, and the links on the Web page lead to other OAI-PMH requests. DP9 supports the resumption token and HTTP 503 status code "retry-after" and thus provides a basic flow control for large data providers. Note that the flexibility in the OAI-PMH allows different ways of constructing the HTML pages to expose an OAI-PMH collection. For example, the starting page could have been constructed using the OAI-PMH request ListRecords. The sequence of OAI-PMH requests we have used in our design was driven by what would be useful for crawlers.

DP9 uses links on Web pages that have the following format:
http://{hostname}/dp9/getrecord/{MetadataFormat}/{OAI_ID}
An example is:
http://arc.cs.odu.edu:8080/dp9/getrecord/oai_dc/oai:NACA:1917:naca-report-10

DP9 creates a series of ListIdentifiers pages for each archive with links to all individual records. These URLs are static and will be only activated when a HTTP

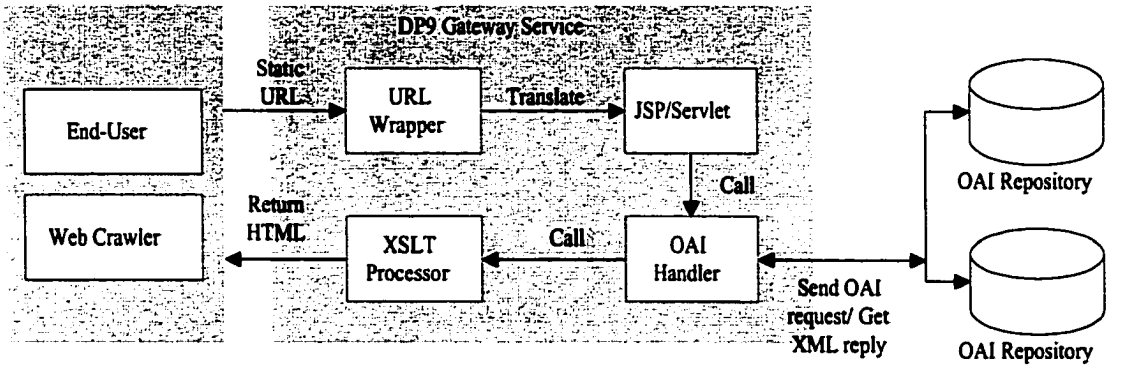

FIG. 8.3: DP9 architecture

request is received. DP9 provides an entry page and if a Web crawler finds this entry page, it may follow the links on this page and send requests to DP9. DP9 will then forward the request to corresponding data providers and process the returned XML records. Depending on the depth a crawler follows, it can index all records in a data provider.

DP9 consists of three main components (Figure 8.3): an URL wrapper, an OAI-PMH handler and an XSLT processor. The URL wrapper accepts the persistent URL and calls internal JSP/Servlet applications. The OAI-PMH handler issues OAI-PMH requests on behalf of a Web crawler. The XSLT processor transforms the XML content returned by the OAI-PMH archive to an HTML format suitable for a Web crawler. XSLT allows DP9 to support any XML metadata format simply by adding an XSL file. DP9 is based on Tomcat/Xalan/Xtag technology from Apache [50].

Some crawlers use the HTML meta tags to index a Web pages; so in addition to creating the user friendly HTML page, DP9 also maps Dublin Core metadata to corresponding HTML meta tags. For pages that are designed exclusively for robots navigation, a noindex robots meta tag is used.

Initial Results

We have collected 70 repositories with well over one million records. Considering Parallel Metadata Sets are supported by OAI-PMH leading to more references, potentially several millions of pages could be indexed by Web search engines. With DP9 now being deployed, thousands of documents in OAI-PMH collections have

been indexed by search engine such as Inktomi and Google. Web logs show that more than 1000 queries are issued from popular Web search engines each day. Figure 8.4 shows how the DP9 service is visited by users through general web search engine like Google; Figure 8.5 shows how the web robots that have visited the DP9 service in May 2002.

Origin					
Direct address / Bookmarks:			12282	22 %	12644	12.2 %
Links from an Internet Search Engine :			31994	57.3 %	32053	31 %
- Google	31585					
- Other search engines	230					
- Yahoo	123					
- Web.de	47					
- MSN	29					
- Virgilio	9					
- Netscape	7					
- Meta Miner	7					
- Euroseek	6					
- Lycos	4					
- Voila	4					
- MetaGer	1					
- Excite	1					
Links from an external page (other web sites except search engines):			3163	5.6 %	3821	3.6 %
- http://arc.cs.odu.edu/simple.html		1511				
- http://arc.cs.odu.edu/mytop.html		510				

FIG. 8.4: The search log of Arc and DP9, most hits are directed from general web search engine (May-2002)

DP9 is a gateway service, it does not cache the OAI-PMH records and only forwards requests to corresponding data providers. This insures DP9's records are always up-to-date; however, its quality of service is highly dependent on the availability of data providers. On the other hand, an aggressive crawler using DP9 can rapidly send requests without regard for the load they are placing on the data providers. The robot exclusion protocol [53] at the data provider site will not be observed

Robots/Spiders visitors

Robots	Hits	Last visit
Googlebot	29768	31 May 2002 - 03:39
larbin	7716	23 May 2002 - 07:39
Mercator (Not referenced robot)	7257	30 May 2002 - 01:03
WISENutbot (Not referenced robot)	4706	31 May 2002 - 23:49
ia_archiver (Not referenced robot)	1774	31 May 2002 - 03:53
Voila (Not referenced robot)	758	23 May 2002 - 14:26
Unknown robot (Not referenced robot)	257	30 May 2002 - 19:03
Wget	116	12 May 2002 - 19:14
Scooter	112	21 May 2002 - 05:33
Inktomi Slurp	43	31 May 2002 - 19:59
IBM_Planetwide	18	24 May 2002 - 08:27
Tcl W3 Robot	12	27 May 2002 - 14:44
Road Runner: The ImageScape Robot	10	22 May 2002 - 19:44
ComputingSite Robi/1.0	6	28 May 2002 - 09:07
Calif	6	18 May 2002 - 17:20
Internet Shinchakubin	3	19 May 2002 - 06:33
Fish search	3	03 May 2002 - 04:53
Pioneer	3	25 May 2002 - 10:01
ARIADNE	2	27 May 2002 - 08:52
Harvest	1	06 May 2002 - 08:39

FIG. 8.5: The robot visitors to DP9 (May-2002)

because the requests come from DP9, an OAI-PMH service provider. We are studying the possibility of using an OAI-PMH mirror/caching mechanism such as OAI-PMH Aggregator [15] and HTTP throttle software [48] to relieve the overhead on data providers.

DP9 also provides an easy way to build services for OAI-PMH-compliant repositories. Indexing tools such as HtDig [49] and GreenStone [132] are designed to index websites, they could be used to build searching services for OAI-PMH collections with DP9 support.

8.6.4 Other Gateway Services

Another gateway service is the reference extraction module in CiteBase. CiteBase extracts the bibliography from arXiv.org documents and exposes them by an additional OAI-PMH interface. These data are then harvested by ODU to build its own citation service.

CiteBase adds reference data by harvesting new records from a repository's OAI-PMH interface, then separately downloading the full text for parsing. The parsed bibliography is added to the existing metadata and is used by CiteBase or harvested by other services.

The CiteBase concept could be extended from the current supported repository arXiv to a general service for any full-text scientific repository - assuming tools can be developed to parse the bibliography.

8.6.5 End-User Services

Both Archon and CiteBase implement a cross-archive search interface; Archon focuses more on harvesting heterogeneous collections and builds an interactive search interface based on harvested metadata, and CiteBase concentrates on automatic reference extraction. Both applications may harvest from the same repository (e.g. arXiv.org) and implement different services for the same record. With the quick adoption of OAI-PMH, we believe this will become a common situation. We implemented two prototypes for cross-linking between Archon and CiteBase (Figure 8.6).

The first approach is to re-expose value-added metadata through an OAI-PMH interface. Using this method, Archon harvests citation data from CiteBase, APS,

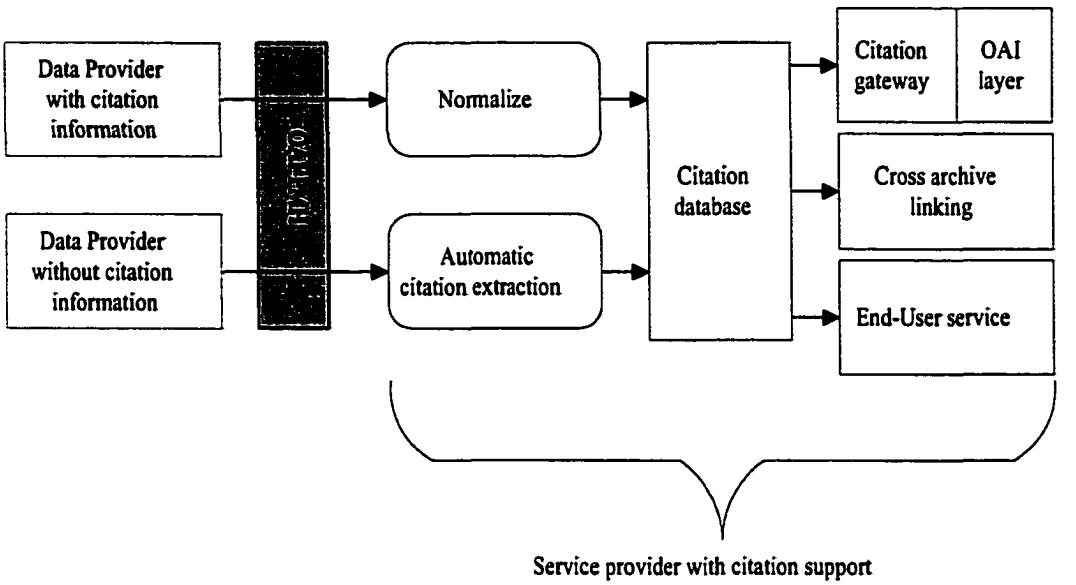

FIG. 8.6: Cross archive citation link

CERN, and other sources. It then builds a cross archive linking service, for example, a citation in APS may lead to a document in CiteBase and vice-versa. Another prototype is based on dynamic linking: both services link to a broker page, and the broker page dynamically checks whether or not a service exists for a specific record. If so, it adds a link to the corresponding service provider. In order to know which records are available in advance, the broker issues an OAI-PMH GetRecord lookup to the target service (which has an OAI-PMH export). Based on the reply, the broker knows whether a record is harvested. We envision that a DP may also link to this broker page for additional services for its data.

8.7 RELATED WORK

8.7.1 Caching and Replication

HTTP proxy and cache distribute load, reduce network traffic and access latency, and protect the network from erroneous clients. There are two basic approaches for web cache implementation: a passive cache and an active cache. The passive cache only loads a data object as a result of a client's request to access that object; the active cache employs some mechanism to prefetch data in advance of a request by a

client [17].

Mirror software is designed to duplicate a directory hierarchy between two machines. It avoids copying files unnecessarily by comparing the file timestamps and file sizes before transferring.

8.7.2 Hierarchical Harvesting

The earlier Harvest project explored the concept of the hierarchical arrangement of object caches and focused on the content extraction for general web documents [12]. After the OAI-PMH was released, both Arc and CiteBase explored the issues of hierarchical harvesting in OAI-PMH service providers. The Open Digital Library project [120] uses the OAI-PMH sets concept for OAI-PMH metadata filtering.

8.7.3 Unique Identifiers

Identifier is a powerful tool for communication within and between communities. For example, the Handle system [122] and DOI (Digital Object Identifier) [97] provide a mechanism for implementing naming systems for arbitrary digital objects. The multiple-resolution capability becomes important in the OAI-PMH community, as metadata may be widely replicated and modified, and many different services will be implemented on the same metadata records. An "intelligent" resolution service should be able to deliver different outcomes to a resolution request dependent on user-specified requirements.

In OAI-PMH, a unique identifier unambiguously identifies an item within a repository. The format of the unique identifier must correspond to that of the URI syntax. Individual communities may develop community-specific URI schemes for coordinated use across repositories. However, the unique identifiers may conform to a recognized URI scheme with greater scope. The *oai-identifier* schema, especially, describes a specific, recommended implementation of unique identifiers which repositories may adhere to; *oai-identifiers* should have global scope and guaranteed global uniqueness. The *oai-identifier* has been widely accepted in implementation of OAI-PMH 1.x and is further refined in the version 2.0 of the protocol by introducing a globally unique OAI URN. All our implementations use the *oai-identifier* schema and rely on its uniqueness.

8.7.4 Citation Linking

Citation linking is the general term for hypertext linking the reference lists (the bibliography) in research articles to the cited articles. In recent years, citation linking has been extensively developed in Open Citation Project, OpenURL and other projects [47, 16, 8, 125, 124]. With the wide acceptance of OAI-PMH, new challenges are raised about cross-archive (i.e. cross collection) linking and cross-service linking. With various data providers providing metadata of different qualities and formats, cross-archive linking is necessary to integrate them into one unique linking environment. Similarly, the distributed and highly redundant OAI-PMH architecture allows different services to be built which, with context sensitive and dynamic cross-service linking, could potentially be integrated. Such integrated services might provide citation analysis for forward links (to articles that have referenced the current article), impact factors, co-citation analysis, and novel navigation methods [18].

8.8 DISCUSSION

OAI-PMH proxies, OAI-PMH caches, and OAI-PMH gateways optimize the functioning of the data provider/service provider model underlying the OAI-PMH. To demonstrate the usability of this framework, we have built several prototype services. These demonstration systems and source codes are available at the web sites of both ODU and the Southampton group.

CHAPTER 9

CONCLUSIONS AND FUTURE WORK

9.1 CONCLUSIONS

Digital library interoperability is essential for building services that will enable users to discover information from multiple libraries through a single unified interface. The difficulty of creating a large distributed searching service is the motivation behind the OAI-PMH to create federated digital libraries based around the concept of metadata harvesting. This dissertation examines the application of the metadata harvesting approach in DL federation. It answers the following questions: Whether or not metadata harvesting provide a realistic and scalable solution for DL federation; How to synchronize data providers with service providers; How to build services over harvested metadata; and how to create a scalable and reliable infrastructure to support metadata harvesting-based applications.

This research has successfully met the objectives as stated in Chapter 2. First, we present a layered architecture of metadata harvesting system and its major components. We introduce the concept of harvesting service, centralized federation service, replicated federation service, registration service, proxy, cache, and gateway service. Secondly, a series of systems, including Arc, Kepler, TRI, Archon, DP9, NCSTRL, Celestial, and OAI-PMH proxy, are developed based on the proposed architecture. We also present major issues involved in constructing a metadata harvesting system. Thirdly, these services are deployed on the Web, implementing publicly available services allows us to demonstrate that metadata harvesting is a realistic approach to implementing digital library federation, and our approach is practical since we have no control over the user community. The developed software are used in a variety of applications and DL deployments. Finally, by performing various experiments and evaluating results, we are able to verify the concept of metadata harvesting, answer open questions, and address major issues in metadata harvesting approach. We study a series of performance criteria, including server availability, reliability, metadata variability, parallel metadata implementation, and update frequency. These criteria are useful for building both data providers and services in metadata harvesting applications.

9.2 FUTURE WORK

We now briefly discuss potential areas for future work. In Chapter 2, we introduced the OAI-PMH protocol, it fits well with the REST (REpresentational State Transfer), a phrase coined by Roy Fielding in his dissertation attempting to describe the Web's architectural style in a rigorous enough fashion to make the principles both comprehensible and extensible [32]. The most important contrast model is RPC [22], in particular, SOAP is the de-facto standard for XML based protocol. There has been some discussion about the desirability of implementing the OAI-PMH in the direction of SOAP or REST, but there are not enough supporting experiments and experiences on this topic to make the OAI community assured about this move.

In Chapter 3, we study the features of current OAI-PMH compliant repositories. This study is preliminary and further study is essential with more digital libraries becoming OAI-PMH compliant. A major contribution of OAI-PMH is to enable digital libraries to expose their metadata for public study, and a richer service requires richer metadata set out of the scope of basic Dublin Core. A more extensive study over parallel metadata formats will improve the standardization of complex metadata sets across digital libraries.

In Chapter 3, we also discuss the registration service. A registration service plays an essential role for a large number of data providers and service providers. OAI-PMH tries to re-use the DNS name to avoid a centralized registration service, and it suggests a "list-friends" model to discover interesting data providers by chatting among neighbors.

- In OAI-PMH 2.0, organizations are asked to choose namespace-identifier values which correspond to a domain-name that they have registered and are committed to maintaining. Domain name registration is used to avoid the need for any additional registration service for oai-identifiers. DNS-based identifiers guarantee global uniqueness without the need for OAI registration.

- A decentralized "list-friends" model is introduced in OAI-PMH 2.0. Data providers keep an optional up-to-date list of befriended data providers, i.e. pointing at other data providers. That list-of-friends would then be made accessible to service providers via the OAI-PMH. In this approach, service providers would have to find an (some) initial point(s) of access (some initial data providers), and could then hop from data provider to data provider to

collect locations of additional data providers. Because the approach relies on data providers pointing at others, it is likely that it would result in unreachable data providers, just like the URL-concept leads to unreachable pages on the web. It is not clear how well or how reliably this model will work.

In Chapter 4, we study the repository synchronization. However, our study was based mainly on experiments rather than theoretical proofs. One interesting research direction would be to formalize the repository synchronization with a strict mathematical model and design an optimal synchronization metric based on that model.

In Chapter 5, we study how to build a federated search interface over heterogeneous repositories. Our approach still makes individual archives visible to the searcher, but from our study it is becoming evident that one unified interface, which exploits rich source metadata and is transparent to participating archives, is feasible. Controlled values are widely used in many archives, and for fields such as type and language, we could map the data to a standard without significant manual effort, by using approximate word matching and other algorithms [35]. The interactive search can also be improved by using reverse-engineered text categorization [36] that is used to supply mappings from an ordinary language vocabulary to a specialist vocabulary.

In chapter 7, we discuss the Kepler framework for individual publishers. Kepler shows promise in changing the current publication model. The Kepler framework can be further developed to cover the annotation services, recommendation services, and peer-review services. It is a challenging issue to encourage authors to use Kepler software and exploit its potential.

In Chapter 8, we discuss the OAI-PMH gateway service. The DP9 service is promising in making the "hidden Web" [7] visible to general web crawlers. The gateway service could be improved with flow-control [48] and the "robot bait" concept, and similar gateway services for other protocols will make a wide range of resources interoperable with the Web. On the other hand, general web search engines also suffer serious freshness and update frequency problems. The fresh and incremental harvesting concept can also be used by the search engine community to create a crawler-friendly Web.

BIBLIOGRAPHY

[1] B. Ahlborn, W. Nejdl, and W. Siberski. OAI-P2P: A peer-to-peer network for open archives. In *Workshop on Distributed Computing Architectures for Digital Libraries. ICPP 2002*, pages 462–468, Vancouver Canada, August 18-21 2002.

[2] H. Anan, X. Liu, K. Maly, M. L. Nelson, M. Zubair, J. French, E. Fox, and P. Shivakumar. Preservation and transition of NCSTRL using an OAI-based architecture. In *the Proceedings of the Second ACM/IEEE Joint Conference on Digital Libraries*, pages 181–182, Portland OR, July 14-18 2002.

[3] Statistics metadata variablity in OAI-PMH repositories. `http://arc.cs.odu.edu/stat`.

[4] Source code of Arc available by SourceForge. `http://oaiarc.sourceforge.net`.

[5] W. Arms. *Digital libraries.* MIT Press, Cambridge, MA, 1999.

[6] G. Beged-Dov, D. Brickley, R. Dornfest, I. Davis, L. Dodds, J. Eisenzopf, D. Galbraith, R. Guha, K. MacLeod, E. Miller, A. Swartz, and E. van der Vlist. RDF Site Summary 1.0 Modules: Syndication, 2000. `http://purl.org/rss/1.0/modules/syndication/`.

[7] M. K. Bergman. The deep web: Surfacing hidden value. *Journal of Electronic Publishing*, 7(1), 2001. `http://www.press.umich.edu/jep/07-01/bergman.html`.

[8] D. Bergmark. Automatic extraction of reference linking information from online documents. Technical Report 2000-1821, Computer Science Department, Cornell University, 2000.

[9] D. Bergmark. Collection synthesis. In *Proceedings of the Second ACM/IEEE Joint Conference on Digital Libraries*, pages 253–262, Portland OR, July 14-18 2002.

[10] T. Berners-Lee. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor.* HarperCollins, New York, NY, 1999.

[11] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, May, 2001.

[12] C. M. Bowman, P. B. Danzig, D. R. Hardy, U. Manber, and M. F. Schwartz. The Harvest information discovery and access system. *Computer Networks and ISDN Systems*, 28(1-2):119–125, 1995. `http://citeseer.nj.nec.com/article/bowman95harvest.html`.

[13] D. Box, D. Ehnebuske, K. G., A. Layman, N. Mendelsohn, H. Nielsen, S. Thatte, and D. Winer. Simple Object Access Protocol (SOAP) 1.1. Technical Report NOTE-SOAP-20000508, W3C, 2000. `http://www.w3.org/TR/SOAP/`.

[14] T. Bray, J. Paoli, and C. M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0 - W3C recommendation 10-February-1998. Technical Report REC-xml-19980210, W3C, 1998. `http://www.w3.org/TR/2000/REC-xml-20001006`.

[15] T. Brody. Celestial aggregator service, 2002. `http://celestial.eprints.org/`.

[16] P. Caplan and W. Y. Arms. Reference linking for journal articles. *D-Lib Magazine*, 5(7/8), 1999. `http://www.dlib.org/dlib/july99/caplan/07caplan.html`.

[17] A. Chankhunthod, P. B. Danzig, C. Neerdaels, M. F. Schwartz, and K. J. Worrell. A hierarchical internet object cache. In *USENIX Annual Technical Conference*, pages 153–164, 1996.

[18] C. Chen and L. Carr. Trailblazing the literature of hypertext: Author co-citation analysis (1989-1998). In *Hypertext '99*. The Association for Computing Machinery, 1999. `http://www.ecs.soton.ac.uk/~lac/ht99.pdf`.

[19] J. Cho. *Crawling the Web: Discovery and maintenance of large-scale web data.* PhD thesis, Department of Computer Science, Stanford University, 2001.

[20] J. Cho and H. Garcia-Molina. Synchronizing a database to improve freshness. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 117–128, 2000.

[21] J. Clark. XSL Transformations (XSLT) version 1.0. Technical Report REC-xml-19980210, W3C, 1998. `http://www.w3.org/TR/xslt`.

[22] G. Coulouris, J. Dollimore, and T. Kindberg. *Distributed Systems: Concepts and Design (3rd Edition)*. Addison-Wesley Publication Corporation, 2000.

[23] J. M. B. Cruz and T. Krichel. Cataloging economics preprints: an introduction to the RePEc project. *Journal of Internet Cataloging*, 3(2/3), 1999.

[24] J. Davis and C. Lagoze. NCSTRL: Design and deployment of a globally distributed digital library. *Journal of the American Society of Information Science*, 51(3):273–280, 2000.

[25] B. Desai. Supporting discovery in virtual libraries. *Journal of the American Society for Information Science*, 48(3):190–204, 1997.

[26] DP9 service. `http://dlib.cs.odu.edu/dp9`.

[27] N. Dushay, J. C. French, and C. Lagoze. A characterization study of NCSTRL distributed searching. Technical Report TR99-1725, Cornell University Computer Science Department, 1999.

[28] C. Ellerman. Channel Definition Format (CDF). Technical report, W3C, 1997. `http://www.w3.org/TR/NOTE-CDFsubmit.html`.

[29] R. Elmasri and S. Navathe. *Foundamentals of database systems*. Addison-Wesley Publishing Company, 1994.

[30] Eprints.org self-archiving software. `http://www.eprints.org/`.

[31] O. Etzioni. The World Wide Web: Quagmire or gold mine? *Communications of the ACM*, 39(11):65–68, 1996.

[32] R. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000.

[33] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. Technical Report Internet RFC 2616, IETF, 1998. `http://www.ietf.org/rfc/rfc2616.txt`.

[34] J. C. French, A. L. Powell, F. Gey, and N. Perelman. Exploiting a controlled vocabulary to improve collection selection and retrieval effectiveness. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, pages 199–206, Atlanta, Georgia, USA, 2001.

[35] J. C. French, A. L. Powell, E. Schulman, and J. L. Pfaltz. Automating the construction of authority files in digital libraries: a case study. In *Proceedings of research and advanced technology for digital libraries, first European conference*, pages 55–71, Pisa, Italy, 1997.

[36] F. C. Gey, M. Buckland, A. Chen, and R. Larson. Entry vocabulary – a technology to enhance digital search. In *Proceedings of the First International Conference on Human Language Technology*, San Diego, USA, 2001.

[37] Gift. giFT interface protocol documentation, 2002. `http://gift.sourceforge.net/docs/?document=interface.html`.

[38] C. L. Giles, K. Bollacker, and S. Lawrence. CiteSeer: An automatic citation indexing system. In I. Witten, R. Akscyn, and F. M. Shipman III, editors, *Digital Libraries 98 - The Third ACM Conference on Digital Libraries*, pages 89–98, Pittsburgh, PA, June 23–26 1998. ACM Press.

[39] P. Ginsparg. Creating a global knowledge network. In *Proceedings of Second Joint ICSU Press - UNESCO Expert Conference on Electronic Publishing in Science*, Paris, 2001.

[40] R. Goldman and J. Widom. Interactive query and search in semistructured databases. In *Proceedings of the International Workshop on the Web and Databases*, 1998.

[41] L. Gravano, K. Chang, H. Garcia-Molina, C. Lagoze, and A. Paepcke. STARTS:stanford proposal for internet meta-searching. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 207–218, 1997.

[42] N. Green, P. Ipeirotis, and L. Gravano. SDLIP + STARTS = SDARTS: A protocol and toolkit for metasearching. In *Proceedings of the First ACM+IEEE Joint Conference on Digital Libraries*, pages 207–214, Roanoke VA, June 24-28 2001.

[43] S. Griffin. NSF/DARPA/NASA digital libraries initiative -a program manager's perspective. *D-Lib Magazine*, 5(7/8), 1998. `http://www.dlib.org/dlib/july98/07griffin.html`.

[44] S. Harnad. The self-archiving initiative. *Nature*, 410:1024–1025, 2001.

[45] S. Harnad and L. Carr. Integrating, navigating and analyzing eprint archives through open citation linking (the OpCit project). *Current Science (special issue honour of Eugene Garfield)*, (79):629–638, 2000.

[46] S. Harnad, L. Carr, and T. Brody. How and why to free all refereed research from access- and impact-barriers online, now. *High Energy Physics Libraries Webzine*, 4, June 2001. `http://library.cern.ch/HEPLW/4/papers/1/`.

[47] S. Hitchcock, L. Carr, Z. Jiao, W. Hall, C. Lagoze, and S. Harnad. Developing services for open eprint archives: globalisation, integration and the impact of links. In *Proceedings of the 5th ACM Conference on Digital Libraries.* The Association for Computing Machinery, 2000. `http://opcit.eprints.org/dl00/dl00.html`.

[48] A. Howe. Apache mod_throttle/3.1.2. `http://www.snert.com/Software/mod_throttle/index.shtml`.

[49] Htdig www search engine software. `http://www.htdig.org`.

[50] Apache jarkata project. `http://jarkata.apache.org`.

[51] A. Kent. OAI harvester crawling status, 2002. `http://www.mds.rmit.edu.au/~ajk/oai/interop/summary.htm`.

[52] T. Koch, H. Neuroth, and M. Day. Renardus: Cross-browsing european subject gateways via a common classification system (DDC). In *IFLA satellite meeting: Subject Retrieval in a Networked Environment*, OCLC, Dublin, Ohio, USA, 2001. `http://www.lub.lu.se/~traugott/drafts/preifla-final.html`.

[53] M. Koster. The web robots page. `http://www.robotstxt.org/wc/robots.html`.

[54] M. Koster. Robots in the Web: threat or treat? *ConneXions*, 9(4), April 1995.

[55] C. Lagoze and J. R. Davis. Dienst - an architecture for distributed document libraries. *Communications of the ACM*, 38(4):47, April 1995.

[56] C. Lagoze, W. Hoehn, D. Millman, W. Arms, S. Gan, D. Hillmann, C. Ingram, D. Krafft, R. Marisa, J. Phipps, J. Saylor, C. Terrizzi, J. Allan, S. Guzman-Lara, and T. Kalt. Core services in the architecture of the National Science Digital Library (NSDL). In *Proceedings of the Second ACM/IEEE Joint Conference on Digital Libraries*, pages 201–209, Portland OR, July 14-18 2002.

[57] C. Lagoze and H. Van de Sompel. The Open Archives Initiative: Building a low-barrier interoperability framework. In *Proceedings of the ACM/IEEE Joint Conference on Digtial Libraries*, pages 54–62, Roanoke VA, 2001.

[58] C. Lagoze, H. Van de Sompel, M. Nelson, and S. Warner. The Open Archives Initiative Protocol for Metadata Harvesting, version 2.0. `http://www.openarchives.org/OAI/openarchivesprotocol.html`.

[59] R. Lasher and D. Cohen. A format for bibliographic records. Technical Report Internet RFC 1807, IETF, 1995. `http://www.faqs.org/ftp/rfc/rfc1807.txt`.

[60] O. Lassila and R. Swick. Resource Description Framework (RDF) model and syntax specification. Technical Report REC-rdf-syntax-19990222, W3C, 1999. `http://www.w3.org/TR/REC-rdf-syntax/`.

[61] S. Lawrence, G. L., and K. Bollacker. Digital libraries and autonomous citation indexing. *IEEE Computer*, 32(6):67–71, 1999.

[62] Z. Lee. The NSF National Science, Technology, Engineering, and Mathematics Education Digital Library (NSDL) program. *D-Lib Magazine*, 7(11), 2001. `http://www.dlib.org/dlib/november01/zia/11zia.html`.

[63] M. Lesk. *Practical Digital Libraries: Books, Bytes and Bucks*. Morgan Kaufmann, July 1997.

[64] A. Y. Levy, A. Rajaraman, and J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings of the 22nd International Conference on Very Large Data Bases*, pages 251–262, 1996.

[74] C. Lynch. Metadata harvesting and the Open Archives Initiative. *ARL Monthly Report*, 217, 2001. `http://www.arl.org/newsltr/217/mhp.html`.

[75] W. Ma, B. Shen, and J. Brassil. Content services networks: The architecture and protocol. In *Proceedings of the WCW'01*, Boston, MA, 2001.

[76] M.-H. Maa, S. L. Esler, and M. L. Nelson. Lyceum: A multi-protocol digital library gateway. Technical Report NASA TM-112871, NASA, 1997.

[77] D. Mahoney and M. D. Giacomo. Flashpoint @ LANL.gov: A simple smart search interface. *Issues in Science and Technology Librarianship*, 2001. `http://www.library.ucsb.edu/istl/01-summer/article2.html`.

[78] M. Mahoui and S. J. Cunningham. Search behavior in a research-oriented digital library. In *Proceedings of the Fifth European Conference on Digital Libraries - ECDL2001, LNCS 2163*, pages 13–24, Darmstadt, Germany, September 4-9 2001.

[79] K. Maly, M. Zubair, H. Anan, D. Tan, and Y. Zhang. Scalable digital libraries based on NCSTRL/ Dienst. In *Proceedings of the Fourth European Conference on Digital Libraries - ECDL 2000*, pages 169– 179, Lisbon, Portugal, 2000.

[80] K. Maly, M. Zubair, and X. Liu. Kepler - an OAI data/service provider for the individual. *D-Lib Magazine*, 7(4), 2001. `http://www.dlib.org/dlib/april01/maly/04maly.html`.

[81] K. Maly, M. Zubair, X. Liu, M. Nelson, and S. Zeil. Structured course objects in a digital library. In *Proceedings of the Third International Symposium on Digital Libraries (ISDL 99)*, pages 89–96, Tsukuba, Japan, September 28-29 1999.

[82] K. Maly, M. Zubair, M. L. Nelson, X. Liu, H. Anan, J. Gao, J. Tang, and Y. Zhao. Archon - a digital library that federates physics collections. In *DC-2002: Metadata for e-Communities: Supporting Diversity and Convergence*, October 13-17 2002.

[83] M.Baldonado, C. Chang, L.Gravano, and A.Paepcke. The stanford digital library metadata architecture. *International Journal of Digital Libraries*, 1(2), 1997.

[84] M. McClelland, D. McArthur, S. Giersch, and G. G. Challenges for service providers when importing metadata in digital libraries. *D-Lib Magazine*, 8(4), 2002. `http://www.dlib.org/dlib/april02/mcclelland/04mcclelland.html`.

[85] L. McLoughlin. Mirror software. `http://sunsite.doc.ic.ac.uk/packages/mirror/`.

[86] Emory university: Metaarchive service. `http://www.metaarchive.org`.

[87] K. Moss. *Java Servlets*. McGraw-Hill Companies, Inc, Boston, MA, 1999.

[88] MySQL reference manual. `http://www.mysql.com/doc/home.html`.

[89] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmer, and T. Risch. Edutella: A p2p networking infrastructure based on RDF. In *Proceedings of the 11th World Wide Web Conference*, Hawaii, USA, 2002.

[90] M. L. Nelson. *Buckets: Smart Objects for Digital Libraries*. PhD thesis, Computer Science Dept, Old Dominion University, August 2000.

[91] M. L. Nelson. Smart objects and dumb archives: Insuring the long term integrity of digital information. In *Proceedings of Exploration 2000*, pages 247–261, Philadelphia, PA, December 12-15 2000.

[92] M. L. Nelson and B. D. Allen. Object persistence and availability in digital libraries. *D-Lib Magazine*, 8(1), 2002. `http://www.dlib.org/dlib/january02/nelson/01nelson.html`.

[93] M. L. Nelson, K. Maly, S. Shen, and M. Zubair. NCSTRL+: Adding multi-discipline and multi-genre support to the Dienst protocol using clusters and buckets. In *Proceedings of Advances in Digital Libraries 98*, pages 128–136, April 22-24 1998.

[94] Network Development and MARC Standards Office, Library of Congress. MARC in XML. `http://www.loc.gov/marc/marcxml.html`.

[95] Network Development and MARC Standards Office, Library of Congress. MARC standard. `http://www.loc.gov/marc/`.

[96] Network Development and MARC Standards Office, Library of Congress. MARC to Dublin Core crosswalk. http://www.loc.gov/marc/marc2dc.html.

[97] N.Paskin. DOI: Current status and outlook. *D-Lib Magazine*, 5(5), 1999. http://www.dlib.org/dlib/may99/05paskin.html.

[98] NSDL. Technical infrastructure white paper. version 2.0, 2002. http://www.smete.org/nsdl/workgroups/technical/nsdl_tech_arch2.0.doc.

[99] An XML schema to represent MARC records. http://www.openarchives.org/OAI/2.0/guidelines-oai_marc.htm.

[100] OAI-PMH registration page in Cornell. http://www.openarchives.org/data/registerasprovider.html.

[101] OLAC: Open Language Archives Community. http://www.language-archives.org/.

[102] Budapest open access initiative. http://www.soros.org/openaccess/.

[103] Oracle InterMedia server reference manual. http://otn.oracle.com/docs/products/oracle8i/doc_index.htm.

[104] A. Oram. *Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology.* O'Reilly & Associates, 2001.

[105] A. Paepcke, R. Brandriff, G. Janee, R. Larson, B. Ludaescher, S. Melnik, and S. Raghaven. Search middleware and the simple digital library interoperability protocol. *D-Lib Magazine*, 6(3), 2000. http://www.dlib.org/dlib/march00/paepcke/03paepcke.html.

[106] A. Paepcke, C. K. Chang, T. Winograd, and H. Garcia-Molina. Interoperability for digital libraries worldwide. *Communications of the ACM*, 41(4):33–43, April 1998.

[107] L. Page and S. Brin. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the Seventh International World-Wide Web Conference*, April 1998.

[108] Public library of science. http://www.publiclibraryofscience.org.

[109] A. Powell and J. C. French. Growth and server availability of the NCSTRL digital library. In *Proceedings of 5th ACM International Conference on Digital Libraries (DL 2000)*, pages 264–265, San Antonio, TX, June 2-7 2000.

[110] G. Reese. *Database programming with JDBC and Java.* O'Reilly & Associates, Sebastopol, CA, 2000.

[111] J. Ritter. Why GNUtella can't scale. no, really. `http://www.darkridge.com/~jpr5/doc/gnutella.html`.

[112] R. Rivest. The MD5 message-digest algorithm. Technical Report Internet RFC-1321, IETF, 1992. `http://www.ietf.org/rfc/rfc1321.txt`.

[113] D. S. H. Rosenthal and V. R. Herbert. Permanent web publishing. In *Proceedings of USENIX Annual Technical Conference*, 2000. `http://lockss.stanford.edu/freenix2000/freenix2000.html/`.

[114] SAX: Simple API for XML. `http://www.saxproject.org/`.

[115] M. Schwartz. Report of W3C distributed indexing and searching workshop, 1996. `http://web3.w3.org/Search/9605-Indexing-Workshop/`.

[116] E. W. Selberg. *Towards Comprehensive Web Search.* PhD thesis, Computer Science Department, University of Washington, 1999.

[117] R. Shi, K. Maly, and M. Zubair. ynamic interoperation of non-cooperating digital libraries. In *Proceedings of Digital Library - IT Opportunities and Challenges in the New Millennium*, Beijing, China, 2002.

[118] P. Shivakumar. Sample NCSTRL usability evaluation report. Technical Report TR02-08, Virginia Tech Computer Science Technical Report, 2002.

[119] H. Suleman. Enforcing interoperability with the Open Archives Initiative repository explorer. In *Proceedings of the ACM/IEEE Joint Conference on Digtial Libraries*, pages 63–64, Roanoke VA, June 24-28 2001.

[120] H. Suleman and E. A. Fox. A framework for building open digital libraries. *D-Lib Magazine*, 7(12), 2001. `http://www.dlib.org/dlib/december01/suleman/12suleman.html`.

[121] H. Suleman, E. A. Fox, and M. Abrams. Building quality into a digital library. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, pages 228–229, San Antonio, TX, 2000.

[122] S. X. Sun and L. Lannom. Handle system overview. Technical Report Internet Draft. draft-sun-handle-system-09.txt, IETF, 2002. `http://www.handle.net/overview-current.html`.

[123] H. Thompson, D. Beech, M. Maloney, and N. Mendelsohn. XML Schema 1.1. Technical report, W3C, 2001. `http://www.w3.org/TR/xmlschema-0/`.

[124] H. Van de Sompel and O. Beit-Arie. Generalizing the OpenURL framework beyond references to scholarly works- the Bison-Fut model. *D-Lib Magazine*, 7(7/8), 2001. `http://www.dlib.org/dlib/july01/vandesompel/07vandesompel.html`.

[125] H. Van de Sompel and O. Beit-Arie. Open linking in the scholarly information environment using the OpenURL framework. *D-Lib Magazine*, 7(3), 2001. `http://www.dlib.org/dlib/march01/vandesompel/03vandesompel.html`.

[126] H. Van de Sompel, T. Krichel, M. L. Nelson, P. Hochstenbach, V. Lyapunov, K. Maly, M. Zubair, M. Kholief, X. Liu, and H. O'Connell. The UPS prototype: An experimental end-user service across E-Print archives. *D-Lib Magazine*, 6(2), February 2000.

[127] H. Van de Sompel and C. Lagoze. The Open Archives Initiative Protocol for Metadata Harvesting, version 1.1. `http://www.openarchives.org/OAI/1.1/openarchivesprotocol.htm`.

[128] H. Van de Sompel and C. Lagoze. The Santa Fe Convention of the Open Archives Initiative. *D-Lib Magazine*, 6(2), 2000.

[129] H. Van de Sompel and C. Lagoze. Notes from the interoperability front: A progress report on the Open Archives Initiative". In *Proceedings of 6th European Conference on Research and Advanced Technology for Digital Libraries*, pages 144–157, Rome, Italy, 2002.

[130] A. Van Hoff, J. Giannandrea, M. Hapner, S. Carter, and M. M. The HTTP distribution and replication protocol. Technical Report NOTE-DRP, World Wide Web Consortium, 1997. `http://www.w3.org/TR/NOTE-drp`.

[131] S. Weibel, J. Kunze, C. Lagoze, and M. Wolfe. Dublin Core metadata for resource discovery. Technical Report Internet RFC-2413, IETF, 1998. `http://www.ietf.org/rfc/rfc2413.txt`.

[132] I. H. Witten, D. Bainbridge, and S. J. Boddie. Greenstone open-source digital library software. *D-Lib Magazine*, 7(10), October 2001. `http://www.dlib.org/dlib/october01/witten/10witten.html`.

[133] L. Wood, V. Apparao, S. Byrne, M. Champion, S. Isaacs, I. Jacobs, A. L. Hors, G. Nicol, J. Robie, R. Sutor, C. Wilson, and L. Wood. Document Object Model (DOM) level 1 specification. Technical Report REC-DOM-Level-1-19981001, W3C Recommendation, 1998. `http://www.w3.org/TR/REC-DOM-Level-1/`.

[134] W.Yeong, T. Howes, and S. Kille. Lightweight Directory Access Protocol. Technical Report Internet RFC-1777, IETF, 1995. `http://www.ietf.org/rfc/rfc1777.txt`.

[135] International standard, ISO 23950: "Information Retrieval (Z39.50): Application service definition and protocol specification". `http://lcweb.loc.gov/z3950/agency/markup/markup.html`.

[136] M. Zubair, K. Maly, I. Ameerally, and M. L. Nelson. Dynamic construction of federated digital libraries. In *Proceeding of WWW9 Conference*, Amsterdam, The Netherlands, 2000.

APPENDIX A

METADATA VARIABILITY OF OAI REPOSITORIES

This appendix lists the number of records harvested and the number of distinct subject, type, format, and language fields used in each archive. The complete explanation is in Section 5.2.

TABLE A.1: Metadata variability of OAI-PMH-compliant repositories (to April 3, 2002)

Archive	No of Records	Date	Subject	Type	Format	Language
8657690236	798	Y	53	1	0	0
AIM25	3962	N	2424	1	1	0
anlc	5	Y	2	1	2	0
anu	114	Y	22	6	0	0
aps	422	N/A	5	0	180	0
arXiv	182996	Y	121	1	0	12
bmc	220	Y	0	13	0	1
caltechCSTR	504	Y	8	2	0	0
caltecheerl	140	N	1	1	0	0
caltechETD	30	Y	10	1	4	1
cav2001	111	Y	103	1	0	0
CBOLD	89	Y	136	1	20	3
CCSDthesis	99	Y	16	1	0	0

Archive	No of Records	Date	Subject	Type	Format	Language
LTRS	2629	N	70	117	0	0
mathpreprints	76	N/A	0	0	0	0
mit.etheses	6288	Y	1339	1	5	0
MONARCH	490	Y	471	11	0	6
NACA	7492	N	0	7483	0	0
NCSTRL	21213	N	0	60	0	7
ndltd	6	Y	4	2	0	0
Nottingham	41	Y	9	5	0	0
NSDL-DEV-CU	2559	N	1735	8	857	9
OpenVideo	1658	Y	389	1	3	2
ota	1245	Y	0	1	44	52
perseus	1394	N/A	0	1	0	0
physdoc	407	Y	397	1	8	13
rdn	387	N/A	674	0	0	24
RIACS	35	Y	5	1	0	0
sammelpunkt	109	Y	29	11	116	0
sceti	47	N	130	0	0	1
scout	50	N	175	0	0	0
SUUB	125	N	97	1	0	2
tkn	321	N	401	25	2	0
Tropicos	517400	Y	0	0	0	0
UBC	2	Y	3	3	1	2
UDLAthesis	95	Y	59	2	1	3
uiLib	29443	N	3353	5	688	4
UKETD	26	Y	9	1	1	0
UKOLN-ejournals	113	N/A	0	0	0	1
USF	28	Y	13	1	2	1
UUdiva	1536	Y	7387	1	2	6
VTETD	3138	Y	170	1	1	1
yea	86	Y	279	2	32	9

APPENDIX B

USAGE OF PARALLEL METADATA

This appendix lists parallel metadata usage in OAI-PMH repositories, the complete explanation is in Section 3.2.4.

TABLE B.1: Usage of parallel metadata in OAI-PMH repositories (to August, 2002)

Archive	Metadata Format
1111	oai_dc
ackarch	oai_dc
AIM25	oai_rfc1807
AIM25	oai_dc
AlanTest	olac
AlanTest	oai_dc
anlc	olac
anlc	oai_dc
ans	oai_vracore
ans	oai_dc
anu	oai_dc
applebytest	olac
applebytest	oai_dc
ArchiveLyon2	oai_dc
arXiv	oai_dc
arXiv	oai_rfc1807

Archive	Metadata Format
arXiv	arXiv
arXiv	arXiv_Old
bmc	bmc_article
bmc	bmc_bibl
bmc	oai_dc
caltechcstr	oai_dc
caltechEERL	oai_dc
caltechETD	oai_rfc1807
caltechETD	oai_marc
caltechETD	oai_etdms
caltechETD	oai_dc
cav2001	oai_dc
cbold	oai_olac
cbold	oai_dc
CCSDarchiveSIC	oai_dc
CCSDJeanNicod	oai_dc
CCSDthesis	ccsd_tel
CCSDthesis	oai_dc
CDLCIAS	oai_dc
CDLDERM	oai_dc
cdlib1	oai_dc
CDLTC	oai_dc
celebration	oai_dc
cogdata	olac
cogdata	oai_dc
cogprints	oai_dc
conoze	oai_dc
CPS	oai_dc
CSTC	ims1_2_1
CSTC	ims1_1
CSTC	oai_cstc
CSTC	oai_dc
CyberTheses	oai_dc
DavidRumseyCollection	oai_dc
DLCommons	oai_dc
dlpscoll	dlxs_bib
dlpscoll	oai_dc

Archive	Metadata Format
DUETT	oai_rfc1807
DUETT	oai_marc
DUETT	oai_etdms
DUETT	oai_dc
EarlyMandarin	olac
EarlyMandarin	oai_dc
EKUTuebingen	oai_dc
eldorado	oai_dc
ELibBSU	oai_dc
epsilondiss	oai_dc
epubWU	oai_dc
ETDIndividuals	oai_etdms
ETDIndividuals	oai_dc
ethnologue	olac
ethnologue	oai_dc
Formosan	olac
Formosan	oai_dc
GenericEPrints	oai_dc
hofprints	oai_dc
hsss	oai_dc
ibiblio	oai_dc
in2p3	oai_in2p3
in2p3	oai_dc
jhjhjh	olac
jhjhjh	oai_dc
JTRS	oai_dc
lacito	olac
lacito	oai_dc
lcoa1	oai_marc
lcoa1	oai_dc
LSUETD	oai_rfc1807
LSUETD	oai_marc
LSUETD	oai_etdms
LSUETD	oai_dc
LTRS	oai_dc
MathPreprints	oai_dc
mit	oai_rfc1807
mit	oai_dc

Archive	Metadata Format
MONARCH	oai_dc
NACA	oai_dc
ncstrlh	oai_rfc1807
ncstrlh	oai_dc
NUIM	oai_dc
OpenVideo	oai_dc
pastel	oai_dc
perseus	olac
perseus	perseus
perseus	oai_dc
physdoc	oai_dc
PKP	oai_rfc1807
PKP	oai_marc
PKP	oai_dc
RIACS	oai_dc
RUGNL	oai_dc
sammelpunkt	oai_dc
sceti	oai_marc
sceti	oai_dc
scoil	olac
scoil	oai_dc
sil	olac
sil	oai_dc
SinicaCorpus	olac
SinicaCorpus	oai_dc
stevenbird	olac
stevenbird	oai_dc
SUUB	oai_dc

Archive	Metadata Format
TalkBank	olac
TalkBank	oai_dc
tkn	oai_dc
tropicos	oai_darwin_core
tropicos	oai_dc
UBC	oai_rfc1807
UBC	oai_marc
UBC	oai_etdms
UBC	oai_dc
uiLib	uiuc_dcq_rdf
uiLib	oai_dc
UMIMAGES	oai_dc
UniversityOfNottingham	oai_dc
UnivOldenburgBIS	oai_dc
UUdiva	oai_dc
VTETD	oai_rfc1807
VTETD	oai_marc
VTETD	oai_etdms
VTETD	oai_dc

APPENDIX C

RECORDS UPDATE RATE (MONTHLY)

This appendix lists monthly records update rate of OAI-PMH Repositories ($R(r_i; t_j), \Delta t = 1month$). The complete explanation is in Section 4.3.

TABLE C.1: Monthly records update rate of OAI-PMH repositories (from 2002-01 to 2002-09)

archive	J-02	F-02	M-02	A-02	M-02	J-02	J-02	A-02	S-02
ACL	N/A	N/A	N/A	2149	0	0	0	0	0
AIM25	26	81	617	151	223	134	11	341	122
CCSDJeanNicod	N/A	N/A	N/A	N/A	3	52	9	2	37
CCSDarchiveSIC	N/A	N/A	N/A	N/A	9	16	4	0	4
CCSDthesis	21	26	23	10	12	44	6	20	24
CDLCIAS	0	1	1	1	0	0	0	0	0
CPS	28	10	9	2	11	15	2	1	6
CSTC	0	2	3	4	6	0	1	0	0

archive	J-02	F-02	M-02	A-02	M-02	J-02	J-02	A-02	S-02
DLCommons	0	2	7	25	3	1	1	0	1
DUETT	8	10	5	5	28	2	83	31	22
EKUTuebingen	1	8	37	429	44	46	38	47	17
EarlyMandarin	N/A	N/A	N/A	N/A	1	0	0	0	0
Formosan	N/A	N/A	N/A	N/A	1	0	0	0	0
GenericEPrints	N/A	1	0	0	0	0	0	0	0
HKUTO	270	81	76	89	89	29	4	8	1
HUBerlin	16	6	1	2	1	2	0	0	3
HUBerlin.de	21	8	19	17	10	6	18	0	5
JTRS	0	2	1	2	1	0	0	0	0
LSUETD	28	2	3	141	6	40	72	9	15
LTRS	12	39	13	12	26	35	39	4	21
MONARCH	5	8	12	8	1	2	1	1	1
NACA	4	6	2	0	2	0	0	0	0
NCSTRL	0	0	0	0	0	0	0	0	0
NSDL-DEV-CU	1	2	5	0	1	0	2	2	2
NUIM	N/A	N/A	N/A	N/A	5	2	3	1	0
Nottingham	0	1	0	1	0	0	0	0	0
OpenVideo	0	603	217	182	192	258	0	0	147
PKP	3	2	0	0	4	0	65	0	0
RIACS	0	1	0	0	1	0	0	0	2
RUGNL	N/A	N/A	N/A	7	7	2	0	3	0
RePEc	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	2E+05
Rumsey	N/A	N/A	N/A	N/A	N/A	6571	0	0	704
SUUB	0	10	87	161	127	57	77	30	52
SinicaCorpus	N/A	N/A	N/A	N/A	1	0	0	0	0
UBC	N/A	2	0	0	0	0	0	0	0
UDLAthesis	0	0	0	0	0	0	0	0	0
UKOLN-ejournals	0	0	0	0	0	0	0	0	0
UMIMAGES	N/A	N/A	N/A	6025	24	20	16202	749	5898
UUdiva	N/A	N/A	N/A	N/A	N/A	N/A	1	0	1714
VTETD	16	25	10	84	115	52	51	78	45
anlc	4137	0	0	0	0	0	0	0	0
anu	49	41	1	15	70	51	1	1	25

archive	J-02	F-02	M-02	A-02	M-02	J-02	J-02	A-02	S-02
arXiv	7744	3198	3874	3089	3605	3672	4462	4181	4505
artiste	N/A	1159	0	0	0	0	0	0	0
arxiv.org	N/A	N/A	N/A	N/A	N/A	2E+05	4282	4345	4583
bmc	50	20	5	11	68	3	0	0	5
caltechCSTR	0	0	1	3	1	1	44	47	3
caltechETD	4	2	1	1	9	10	1	5	76
caltecheerl	3	1	5	0	28	34	20	0	1
cdlib1	3	3	4	6	0	4	79	78	1
cds.cern.ch	N/A	N/A	N/A	N/A	N/A	N/A	N/A	19647	469
cogdata	N/A	N/A	N/A	N/A	N/A	1	0	0	0
cogprints	1370	19	10	11	8	40	15	41	11
conoze	37	29	21	9	27	12	5	4	0
dispute	0	0	94	0	0	0	64	0	0
dlpscoll	1621	12	0	629	1386	0	109	0	11221
eldorado	99	20	12	45	2	17	17	5	2
epsilondiss	N/A	N/A	N/A	13	12	3	0	3	1
epubwu	N/A	N/A	N/A	N/A	1	1	0	0	0
glasgow	3	1	0	2	8	1	0	0	0
hopprints	6	2	13	1	0	0	0	0	4
hsss	0	4	0	0	7	1	0	0	0
ibiblio	N/A	N/A	N/A	N/A	N/A	N/A	1	380	1
in2p3	180	140	276	57	90	110	108	52	141
informedia	71	0	0	0	0	0	0	0	0
ioffe	N/A	N/A	337	0	0	0	0	0	0
lacito	0	31	0	0	1	0	2	0	0
lcoa1	0	7756	0	0	0	0	1	522	2
lcoa1.loc.gov	N/A	N/A	N/A	N/A	N/A	1E+05	4417	193	15772
lib.umich.edu	1621	0	48995	4099	730	0	45665	0	11221
ltrs.larc.nasa	12	40	31	22	42	35	71	31	24
mathpreprints	5	6	3	3	20	40	12	7	12
mit.etheses	46	86	142	119	189	63	75	124	82
naca.larc.nasa	4	6	3	0	2	0	0	0	0
ndltd	1	0	0	0	0	0	0	0	0
oai.library.uiuc	0	0	0	96853	0	0	0	0	0
oai.sunsite.utk	N/A	N/A	N/A	N/A	N/A	N/A	2	351	94
ota	0	0	0	0	0	0	0	0	0
pastel	N/A	N/A	16	18	29	3	0	0	1

archive	J-02	F-02	M-02	A-02	M-02	J-02	J-02	A-02	S-02
perseus	0	0	0	0	0	0	0	0	0
sammelpunkt	N/A	1	25	0	0	0	17	0	12
scielo	781	321	488	420	225	571	168	258	198
scoil	N/A	N/A	N/A	N/A	1552	0	0	0	0
tkn	0	0	0	55	2	0	0	0	0
uiLib	1086	0	27564	96149	0	0	0	0	0
unimelb.edu.au	N/A	N/A	N/A	N/A	N/A	N/A	1	46	32
xtcat.oclc.org	N/A	1E+06	0	0	0	0	0	0	0
yea	0	0	0	0	0	0	0	0	0

APPENDIX D

UPDATE INTERVAL (DAILY)

This appendix lists update interval of OAI-PMH repositories, the complete explanation is in Section 4.2 and Section 4.3. The observance interval is day, or ,$\Delta t = 1$ *day*

$AVG(r_i)$: Average Update Rate

$U(r_i)$: Average Update Interval

$stdv(I(r_i))$: Standard Deviation of Update Interval

$C.O.V.(I(r_i))$: Coefficient of Variation of Update Interval

TABLE D.1: Repository update interval of OAI-PMH repositories (09/30/3001-09/30/2002)

archive	$AVG(r_i)$	$U(r_i)$	$stdv(I(r_i))$	$C.O.V.(I(r_i))$
AIM25	39.43	6.16	10.2	1.66
anlc	912.75	67.5	64.53	0.96
anu	5.19	6.06	12.12	2
arxiv.org	145.32	1	0	0
bmc	3.19	3.25	11.93	3.67
caltechCSTR	5.18	13.77	24.47	1.78
caltecheerl	6	13.22	17.78	1.34
caltechETD	3.25	9.53	14.1	1.48

archive	$AVG(r_i)$	$U(r_i)$	$stdv(I(r_i))$	$C.O.V.(I(r_i))$
CCSDarchiveSIC	1.58	7.16	17.2	2.4
CCSDJeanNicod	3.57	4.5	5.53	1.23
CCSDthesis	1.85	3.12	6.38	2.04
CDLCIAS	3	73	92.05	1.26
cdlib1	6.52	12.55	111.91	8.92
cds.cern.ch	873.43	2.35	3.42	1.46
cogprints	20.07	3.8	4.13	1.09
conoze	3.51	6.8	8.59	1.26
CPS	1.58	3.7	7.15	1.93
CSTC	1.4	22.33	106.93	4.79
dispute	22.57	30.29	93.5	3.09
DLCommons	3.78	19.67	30.18	1.53
dlpscoll	1737	32.33	111.86	3.46
DUETT	3.18	4.59	4.7	1.02
EKUTuebingen	7.33	3.73	9.28	2.49
eldorado	7.93	5.04	9.48	1.88
epsilondiss	2.07	10.73	18.1	1.69
epubwu	1	134	105.76	0.79
glasgow	1.79	22.29	24.68	1.11
HKUTO	5.23	2.65	62272.87	23462.58
hopprints	2.42	29.08	42.32	1.46
hsss	1.09	21.36	219.99	10.3
HUBerlin	1.68	6.87	1438.23	209.41
HUBerlin.de	1.95	4.32	4871.18	1127.22
ibiblio	63.5	12.33	11.35	0.92
in2p3	7.89	1.71	2.92	1.71
informedia	23.14	49.43	571.92	11.57
ioffe	66.5	94.5	93	0.98
JTRS	1.4	60.4	173.98	2.88
lacito	8.5	60	90.31	1.51
lcoa1	1037.12	44.38	47.74	1.08
lcoa1.loc.gov	7648.2	11.6	8.53	0.74
lib.umich.edu	8640.85	19.31	108.76	5.63
LSUETD	4.08	4.09	8.14	1.99
LTRS	2.36	2.81	7.12	2.53
ltrs.larc.nasa.gov	2.43	2.15	5.41	2.52
mathpreprints	1.54	4.33	5.82	1.34
mit.etheses	7.33	2.03	3.3	1.62
MONARCH	1.25	6.55	36.03	5.51
NACA	24.61	19.44	18.55	0.95
naca.larc.nasa.gov	24.37	18.42	17.79	0.97

archive	$AVG(r_i)$	$U(r_i)$	$stdv(I(r_i))$	$C.O.V.(I(r_i))$
NCSTRL	2723.25	84.5	70.06	0.83
Nottingham	1.6	35.5	55.9	1.57
NSDL-DEV-CU	2.15	18.05	2531.85	140.27
NUIM	1.25	16.5	10.75	0.65
oai.library.uiuc.edu	24481.5	82	74.39	0.91
oai.sunsite.utk.edu	74.17	12.83	13.97	1.09
OpenVideo	69.65	13.19	23.5	1.78
ota	2.06	36.88	81.07	2.2
pastel	2.36	7.57	17.58	2.32
perseus	2	364	5454.73	14.99
PKP	18.5	67.75	365438.63	5393.93
RePEc	44	2.38	1.29	0.54
RIACS	1.56	40.22	55.48	1.38
RUGNL	1.89	19.44	18.26	0.94
Rumsey	704	110	25	0.23
sammelpunkt	4.15	16.54	33.75	2.04
scielo	411.08	30.33	100.93	3.33
SUUB	7.46	4.46	13.25	2.97
tkn	14.25	43.25	111.57	2.58
torc9.cs.utk.edu	1.67	2	0.82	0.41
UDLAthesis	3.67	119	1891.9	15.9
uiLib	41237.67	80.67	60.23	0.75
UKOLN-ejournals	25	364	502.53	1.38
UMIMAGES	1526.2	10.6	9.07	0.86
unimelb.edu.au	4.59	5.29	10.97	2.07
UUdiva	857	43.5	71.92	1.65
VTETD	3.5	2.11	56.5	26.77
www.open-video.org	45.75	8.4	14.22	1.69
yea	49	341	563.75	1.65

APPENDIX E

QUERY LOGS OF ARC AND NCSTRL

TABLE E.1: Number of queries of Arc (from 2001-03 to 2002-07)

Month	Number of queries
2001-03	897
2001-04	1201
2001-07	891
2001-08	1271
2001-09	1549
2001-10	1773
2001-11	1642
2001-12	1827
2002-01	1654
2002-02	2324
2002-03	2404
2002-04	2627
2002-05	2206
2002-06	1809
2002-07	1694

TABLE E.2: Number of queries in NCSTRL (from 2001-10 to 2002-08)

Month	Number of queries
2001-10	1897
2001-11	2340
2001-12	6196
2002-01	8212
2002-02	3636
2002-03	7691
2002-04	7939
2002-05	7411
2002-06	6755
2002-07	6020
2002-08	6480

VITA

Xiaoming Liu
Department of Computer Science
Old Dominion University
Norfolk, VA 23529

ADDRESS
814 Westmoreland Ave, APT 80,
norfolk, VA, 23508

EDUCATION

B.S. Computer Science, July 1994, ShanDong University
M.S. Computer Engineering, March 1997, Shanghai Jiaotong University
Ph.D Computer Science, December, 2002, Old Dominion University

Typeset using LaTeX.