Repository Synchronization in the OAI Framework

Xiaoming Liu*
Research Library
Los Alamos Research Laboratory
Los Alamos, NM, 87544
liu_x@lanl.gov

Kurt Maly Mohammad Zubair Michael L. Nelson
Computer Science Department
Old Dominion University
Norfolk, VA, USA
{maly,zubair,mln}@cs.odu.edu

Abstract

The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) began as an alternative to distributed searching of scholarly eprint repositories. The model embraced by the OAI-PMH is that of metadata harvesting, where value-added services (by a "service provider") are constructed on cached copies of the metadata extracted from the repositories of the harvester's choosing. While this model dispenses with the well known problems of distributed searching, it introduces the problem of synchronization. Stated simply, this problem arises when the service provider's copy of the metadata does not match the metadata currently at the constituent repositories. We define some metrics for describing the synchronization problem in the OAI-PMH. Based on these metrics, we study the synchronization problem of the OAI-PMH framework and propose several approaches for harvesters to implement better synchronization. In particular, if a repository knows its update frequency, it can publish it in an OAI-PMH Identify response using an optional About container that borrows from RDF Site Syndication (RSS) Format.

1. Introduction

Although now falling out of favor for the architecture of most digital libraries because of its well known limitations [6, 7] , distributed searching does have one strong advantage: the results are always an accurate assessment of the repository contents at the time of the search. Distributed searching has been eclipsed by the metadata harvesting model, as described by the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) [10].

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 and is a result of the metadata harvesting model. Frequent harvesting has to be done to synchronize the repositories and harvesters, but it is inefficient if data providers seldom change during a harvest interval. On the other hand, without frequent crawling, service providers may become inconsistent with data providers: not only can new records be missed, but deletions and modifications as well. While always undesirable, some applications (e.g. news feeds) are more sensitive to this situation than others (e.g. scholarly publications). The user must trust that the service provider has an accurate assessment of the contents of the repositories 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 repositories 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 needs to maintain hourly freshness for a satisfactory service; a web search engine may reharvest its indexed pages over a span of several months. We define the *acceptable latency* is the interval at which a service provider considers acceptable when it is synchronized with repositories. It is worth noting that originally, OAI-PMH 1.x only supported day granularity, but now OAI-PMH 2.0 supports the option of second granularity. However, the granularity of the protocol is not likely change the nature of the update frequency of a repository.

The metadata harvesting model is built on service providers "pulling" metadata from a set of repositories. After studying the harvest logs of Arc [14, 13], we conclude that most repositories have a steady change rate, but different repositories 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"

*This research was performed while the author was a doctoral student at Old Dominion University

rate based on the change rate of repositories. Several possible approaches may be used to determine the change of a repository:

Best Estimation The harvester estimates the record update frequency by learning the harvest history.

Syndication A data provider may describe its update frequency explicitly. Motivated by the work in RSS (RDF Site Syndication Format) [1] and other applications such as news syndication, we define an optional container in which a data provider can describe its update rate.

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.

Push Model Data providers may directly push updates to service provider side.

The remainder of this paper is organized as follows: Section 2 summarizes the previous and related work in synchronizing updates and measuring the freshness of web pages and digital libraries. We formally study the repository synchronization problem in Section 3. We study the update frequency of OAI-PMH compliant repositories in Section 4. Section 5 presents two algorithms to implement repository synchronization in the OAI-PMH framework. Section 6 presents approaches for determining repositories' update frequency.

2. Previous and Related Work

The original Harvest system from the University of Colorado [2] utilized a complex system of gathers and brokers to reduce the redundancy of typical web robot operation. Although no longer in use, portions of the Harvest system did evolve into a number of commercial products and it did provide the first working demonstration of incremental updates in a web environment.

Cho and Garcia-Molina [4, 5] gathered data from 270 web sites over a four month period and analyzed it by defining age and freshness metrics and by modeling the individual elements of a database as well as the database in its entirety. They then looked at the synchronization frequency and compared synchronization order and resource allocation policies. However, they were dealing with uncoordinated changes of web pages, which are different from the incremental harvesting model of the OAI-PMH.

Labrinidis [8] studied update scheduling problem in web materialization, where dynamic pages are cached at the web server and constantly updated in the background, resulting in fresh data accesses on cache hits. A Quality of Data (QoD) metrics was defined to evaluate how fresh the data served to the users is. An update scheduling algorithm was designed to maximize overall QoD by allocating more resources to frequently used pages. This study also focuses on individual web pages.

The RSS syndication module provides hints to aggregators and others picking up this RSS feed regarding how often it is updated [1]. The RSS is widely used in news aggregation services. RSS is roughly the converse of the OAI-PMH model: RSS provides a single URL that is guaranteed to always have the most recent contents, and the OAI-PMH provides a single URL through which a harvester can dynamically discover the contents of a repository during different points in time.

The proposed "HTTP Distribution and Replication Protocol" (DRP) [17] 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 [16], to identify the changes of content.

In the Open Citation Project, Brody [3] studied how often papers are changed and updated after initial submission and how extensive the changes are in arXiv.org eprint archive. It explained in more detail about the update pattern in an individual digital library.

3. Metrics for Update Frequency and Freshness

The OAI-PMH is based on a coordinated model in which 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. For smaller web sites (100s of pages), the difference between the OAI-PMH model and that of standard web crawlers is negligible. The power of the OAI-PMH model is realized for large digital library web sites (several 1000s of pages or more).

We formally define several metrics to measure the update frequency of data providers. Let $\{r_1, ..., r_M\}$ be the M repositories 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 largely irrelevant to the metrics that follow. We will therefore denote the observation times as $\{t_1, ..., t_n\}$, where $t_{j+1} = t_j + \Delta t$. For any t_j, let t_c denote the nearest update time before t_j.

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

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

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:

$$I(r_i;t_j) = \begin{cases} 0 & if\ S(r_i;t_j) == 0 \\ j-c & if\ S(r_i;t_j) == 1 \end{cases} \quad (1)$$

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

$$U(r_i) = \frac{\sum_{j=1}^{n} I(r_i;t_j)}{\sum_{j=1}^{n} S(r_i;t_j)} \quad (2)$$

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

$$FRQ(r_i) = \frac{1}{U(r_i)} \quad (3)$$

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

$$AVG(r_i) = \frac{\sum_{j=1}^{n} R(r_i;t_j)}{\sum_{j=1}^{n} S(r_i;t_j)} \quad (4)$$

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

$$F(r_i;t_j) = \begin{cases} 1 & if\ r_i\ is\ up-to-date\ at\ time\ t_j \\ 0 & otherwise \end{cases} \quad (5)$$

Freshness of Service Provider: The freshness of the service provider H at time t_j is:

$$F(H;t_j) = 1/M \sum_{i=1}^{M} F(r_i;t_j) \quad (6)$$

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 repositories. This latency could range from several seconds to several months, depending on different applications.

Based on how synchronization resources are allocated to repositories, we can classify synchronization policies as uniform model and adaptive model.

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

$$C_d(r_i) = \frac{T}{l} \quad (7)$$

$$C_s = \frac{TM}{l} \quad (8)$$

In the adaptive 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.

$$C_d(r_i) = \frac{T}{U(r_i)} = T * FRQ(r_i) \quad (9)$$

$$C_s = \sum_{i=1}^{M} \frac{T}{U(r_i)} = \sum_{i=1}^{M} T * FRQ(r_i) \quad (10)$$

From formula (8) and (10), we can derive that the acceptable latency, number of data providers, and repository update frequency play important roles in calculating the update cost. We illustrate these metrics by two examples.

Example 3.1 In a typical digital library application (i.e., scholarly publications), a daily latency should satisfy most requirements, if the observance period of T equals one day, from formula (8) we can derive $C_s = M$.

Example 3.2 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 uniform model, $C_s = 1440 * M$, the uniform model will not scale. In contrast, in the adaptive model, $C_s = 24 * M$, it promises better efficiency.

The algorithms for both models are described in Section 5. In the adaptive model, the harvester essentially allocates more resources to active repositories; the prerequisite is that harvester must know the update interval of the repositories. The approaches to decide update interval are discussed in Section 6.

4. Study of Update Frequency

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 repositories 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 repositories, if a data provider allowed the metadata to change based on usage, annotations, or reviews as specified in the NSDL project [9], the required harvesting would likely become significant.

In this Section, we present our experimental results that show how OAI-PMH-compliant repositories change. We address the following questions:

- Does the data provider change at a constant rate?
- How often does a data provider change?

We run the Arc harvester once a day to harvest approximately 100 OAI-PMH-compliant repositories. The datestamps of harvested records are kept in a database. The change rate covers new data, modified data, and deleted data.

Table 1 lists the monthly average update rate of records in selected e-print archives. They are randomly selected from production OAI-PMH-compliant e-print services (we consider production services demonstrate a more realistic trend), the complete table is available at [11]. 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.

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 datestamp granularity. In Table 2, we list the daily average update interval, average update rate, standard deviation of update interval, and Coefficient of Variation (C.O.V.). The complete table is in [11]. It shows that most of them have a relatively small C.O.V. In Figure 1, we summarize the result of this analysis. In the figure, the horizontal axis represents the average update interval of repositories, and the vertical axis shows the fraction of repositories changed at the given average interval. We can observe that less than 10% of the repositories change daily, while about 70% of the repositories change monthly or longer.

In summary, OAI-PMH repositories (especially E-print archives) change at a steady rate overall, and the rates vary dramatically from site to site.

Because the OAI-PMH supports the features of incremental harvesting, with current number of OAI-PMH compliant repositories (about 100), 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 fresh harvesting cycle over all participating repositories.

5. Synchronization Algorithm for Harvesters

In Section 4, we conclude that the OAI-PMH model of synchronization works well for current OAI-PMH repositories (out of about 100 repositories, 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 in the NSDL project .
- If the OAI-PMH is used in some applications which require rapid updates (granularity of an hour or less), such as news or mailing lists.

In Section 3, we define two synchronization models: uniform and adaptive. The uniform model is implemented in the Arc harvester. The adaptive model is based on the features that most repositories 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, in the optimal case, the harvester can be notified whenever a data provider changes.

5.1. Uniform Model

In the uniform model, we repeatedly synchronize the repositories in the same order. We describe the uniform model more formally in Figure 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 repositories 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.

5.2. Adaptive Model

In this model, the harvester changes its synchronization rate based on the average repository update interval (Figure 3). Here, the harvester adjusts its harvesting frequency

Table 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 [11]

archive	02/01	02/02	02/03	02/04	02/05	02/06	02/07	02/08	02/09
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

Table 2. Repository update interval (09/30/2001-09/30/2002)

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

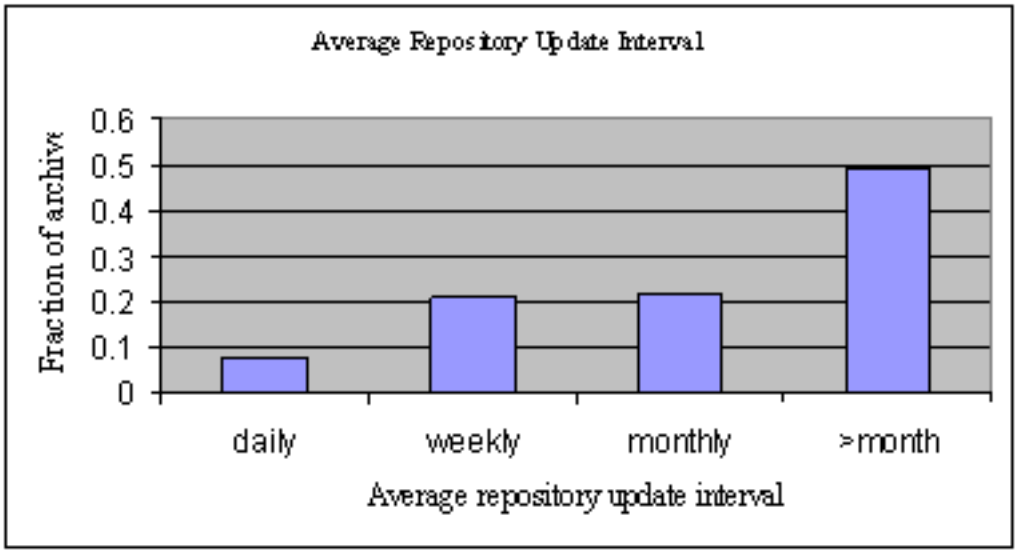

Figure 1. Average repository update interval of OAI-PMH repositories

```
Algorithm 1  UniformModel Synchronization
Input : ArchiveList = {a_1, a_2, ..., a_n}
LastHarvestTime = {t_1, t_2, ..., t_n} = null
Procedure
for(i = 1; i ≤ n; i++){
        t_i = getresponsetime(a_i);
        historical_harvest(a_i);
 }
while(true){
      for(i = 1; i ≤ n; i++){
            responsetime = getresponsetime(a_i);
            fresh_harvest(a_i, t_i);
            t_i = responsetime;
      }
      sleep(pre_defined_interval);
}
```

Figure 2. Algorithm for the Uniform Model

```
Algorithm 2  AdaptiveModel Harvesting
Input : ArchiveList = {a_1, a_2, ..., a_n}
AverageUpdateInterval = {u_1, u_2, ..., u_n}(such as 1 day, etc.)
LastHarvestTime = {t_1, t_2, ..., t_n} = null
Procedure
for(i = 1; i ≤ n; i++){
      t_i = getresponsetime(a_i);
      historical_harvest(a_i);
 }
while(true){
      for(i = 1; i ≤ n; i++){
          if(currenttime − t_i ≥ u_i){
              responsetime = getresponsetime(a_i);
              fresh_harvest(a_i, t_i);
              t_i = responsetime;
          }
      }
      sleep(pre_defined_interval);
}
```

Figure 3. Algorithm for the Adaptive Model

based on updated frequency of individual data providers, a more frequently updated data provider is allocated more resources.

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. Furthermore, beyond the framework of OAI-PMH, a data provider can notify the service provider about its update, or directly push updated metadata to service provider side.

6. Approaches For Determining Update Frequency

There are four approaches to decide update frequency of a data provider.

Best Estimation The harvester estimates the record update frequency by learning the harvest history.

Syndication A data provider may describe its update frequency explicitly.

Subscribe/Notify A data provider may notify a service provider whenever its content is changed.

Push Model Data providers may directly push updates to service provider side.

In the above approaches, best estimation and syndication can be integrated into current OAI-PMH model seamlessly; subscribe/notify and push models extend or modify OAI-PMH model, but they are useful in special cases such as the Kepler service [12, 11].

6.1. 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 logs of Arc, we conclude in Section 4 that many current OAI-PMH compliant repositories present a constant update rate, and the rates vary dramatically from site to site.

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 the update cost. Typically, the freshness requirement is decided by the application.

6.2. Syndication

In OAI-PMH, the response to an *Identify* request may contain locally defined description containers that can be used

```
<?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>
```

Figure 4. XML schema for syndication

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 3.

The RSS (Rich Site Summary) syndication module provides syndication hints to aggregators and others picking up RSS feeds regarding how often it is updated. The RSS defines three parameters to define update frequency of a data source: UpdatePeriod (Describes the period over which the data provider is updated), UpdateFrequency (Describe the frequency of updates in relation to the update period), and UpdateBase (Defines a base date to be used in concert with updatePeriod and updateFrequency to calculate the publishing schedule) [1]. These parameters provide syndication hints to aggregators and others picking up RSS feeds 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."

According to the RSS specification, we define an optional XML schema (Figure 4), which can be used by a data provider to describe its update rate. We do not address how a repository comes to know its update rate (it is beyond the scope of the OAI-PMH discussion), nor do we address inaccurate or deceitful repositories (cf. [15]).

Similar to best estimation, a syndication container may not be accurate, or a data provider is not able to provide a reasonable syndication container at all. However, a harvester can still improve freshness by allocating more resources to frequently changed repositories.

6.3. 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.

The subscribe/notify promises optimal results. It is useful in a repository with low and irregular update rate, such as archivelets in Kepler framework. However, this model requires that a service provider actively listen for the notification, which adds implementation complexity to the service provider. It also requires that a data provider keeps a record of subscribed service providers, and harvester issues OAI-PMH requests in an asynchronous fashion.

We extend the OAI-PMH with one additional "Notify" verbs on the service provider side as a way to optimize the functioning of the OAI-PMH. The repository issues a Notify request to inform the service provider that some new data is available. The harvester issue asynchronous OAI-PMH requests based on its status.

6.4. Push Model

Repositories may directly push updates to service providers. In this case, however, repositories decide when and what to push. That would require a data provider to keep the harvesting status. In implementation, the service provider supports a new "PushMetadata" verb, the metadata is attached in a HTTP POST request.

The push model provides additional benefits of bypassing firewalls and/or NAT (Network Address Translation) proxies. Similar to subscribe/notify model, it promises a nearly optimal synchronization. However, in the push model, the repository has additional complexity of saving the status of how it is harvested; in the subscribe/notify model, the communication is asynchronous, the status information is saved in harvester side.

7. Conclusions

The synchronization of harvesters and repositories is the key issue the OAI-PMH resolves. Compared to the general web crawlers, OAI-PMH optimizes the repository synchronization by supporting the incremental and selective harvesting. With more OAI-PMH repositories and an intention to support a wide range of applications, it is imperative to study the efficiency of a harvester and freshness of a service provider. We define some metrics for describing the synchronization problem in the OAI-PMH. Current update frequency of OAI-PMH compliant repositories is studied and several new methods are introduced to improve the efficiency of a service provider staying synchronized with large numbers of data providers. We introduce an optional *About* container (adopted from the RSS) for the OAI-PMH *Identify* response that allows a repository to advertise its update frequency.

References

[1] 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/`.

[2] 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`.

[3] T. Brody. Mining the social life of an eprint archive. http://opcit.eprints.org/tdb198/opcit/.

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

[5] 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.

[6] 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.

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

[8] A. Labrinidis and N. Roussopoulos. "Update Propagation Strategies for Improving the Quality of Data on the Web". In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB'01)*, Rome, Italy, Sept. 2001.

[9] 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.

[10] 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`.

[11] X. Liu. *Federating Heterogeneous Digital Libraries by Metadata Harvesting.* PhD thesis, Department of Computer Science, Old Dominion University, 2002.

[12] X. Liu, K. Maly, and M. Zubair. Enhanced Kepler framework for self archiving. In *Workshop on Distributed Computing Architectures for Digital Libraries. ICPP 2002*, pages 455–461, Vancouver Canada, August 18-21 2002.

[13] X. Liu, K. Maly, M. Zubair, and M. L. Nelson. Arc - an OAI service provider for digital library federation. *D-Lib Magazine*, 7(4), 2001. `http://www.dlib.org/dlib/april01/liu/04liu.html`.

[14] X. Liu, K. Maly, M. Zubair, and M. L. Nelson. Arc: An OAI service provider for cross archive searching. In *Proceedings of the ACM/IEEE Joint Conference on Digtial Libraries*, pages 65–66, Roanoke VA, June 24-28 2001.

[15] C. Lynch. When documents deceive: Trust and provenance as new factors for information retrieval in a tangled web. *Journal of the American Society for Information Science and Technology*, 52(1):12–17, 2001.

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

[17] 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`.