Hermes – A Notification Service for Digital Libraries

D. Faensen, L. Faulstich, H. Schweppe, A. Hinze, A. Steidinger
Institute for Computer Science
Freie Universität Berlin
hermes@inf.fu-berlin.de

ABSTRACT

The high publication rate of scholarly material makes searching and browsing an inconvenient way to keep oneself up-to-date. Instead of being the active part in information access, researchers want to be notified whenever a new paper in one's research area is published.

While more and more publishing houses or portal sites offer notification services this approach has several disadvantages. We introduce the Hermes alerting service, a service that integrates a variety of different information providers making their heterogeneity transparent for the users. Hermes offers sophisticated filtering capabilities preventing the user from drowning in a flood of irrelevant information. From the user's point of view it integrates the providers into a single source. Its simple provider interface makes it easy for publishers to join the service and thus reaching the potential readers directly.

This paper presents the architecture of the Hermes service and discusses the issues of heterogeneity of information sources. Furthermore, we discuss the benefits and disadvantages of message-oriented middleware for implementing such a service for digital libraries.

Categories and Subject Descriptors

H.3.6 [**Information Storage and Retrieval**]: Library Automation; H.2 [**Information Systems**]: Database Management

General Terms

Digital libraries

Keywords

alerting services, digital libraries, selective dissemination of information (SDI)

JCDL'01, June 24-28, 2001, Roanoke, Virginia, USA.

1. INTRODUCTION

The traditional way of using electronic publications is searching and browsing. The user must become active to find information. A much more appropriate usage pattern for a digital library would be the notification of readers whenever new material of interest becomes available. A service that provides such a value is called notification or alerting service (AS). The increasing number of scholarly publications emphasizes the need for sophisticated filtering capabilities of such an alerting service to prevent the drowning of users in a flood of irrelevant publications.

Some information providers such as publishing houses do offer alerting on their publications. This is definitely beneficial for end users. However, this bilateral 'user-provider' approach has significant disadvantages. Users have to know all relevant providers. They have to deal with a variety of interfaces, and must maintain their profiles at many sites. For privacy reasons users might not want to disclose their profiles of interest to arbitrary information providers. Duplicate notifications (e.g., occurring in the case when user receives notifications from both a publishing house and an abstracting and citation service) cannot be avoided. The capabilities to express users' interests (profiles) are poor. Typically, the user can just select a set of journals and then receives the table of contents by E-mail whenever a new issue of one of these journals appears. This leads to low precision and recall. Notifications from different providers clutter the mailbox since one rarely has the possibility to specify a notification schedule, and notifications are not integrated. Finally, small information providers (small publishers, universities, etc.) usually do not offer an alerting service.

An integrative alerting system that integrates the variety of providers and publication types and hides their specifics from the users avoids most of these problems. It should offer a unified interface and sophisticated profile expression capabilities including filtering as well as user-defined notification schedules. It must be open and easy to join for other providers which can then feed their bibliographical data into the alerting service and advertise their materials directly to an interested target group. Finally, it must be scalable to support millions of user profiles and tens of thousands of publications daily.

Today, huge amounts of valuable scientific literature are available on the Web. While users can retrieve those data from the publishers' Web sites it is unlikely that each provider is capable and willing to install or support the interfaces of an integrative alerting service. Instead, appropriate

wrappers for such Web sites can be employed to make their material available for profile-filtered notification.

Despite its obvious usefulness, only a small number of integrative alerting services have been developed for digital libraries.

Individual alerting services are offered by several publishing houses, such as Springer Link Alert[1] and Elsevier Contents Direct[2]. However, these services underly serious restrictions: they are mostly based on the publications offered by the particular publishing house only. Profiles can only be defined in a basic way, their definition is restricted to the selection of certain journals, no full-text retrieval is possible. Services by secondary publishers as Swets (service SwetsScan[3]) naturally cover a wider but still restricted selection of materials.

Some abstracting and citation services also offer notification services, such as the ISI services[4], Catchword[5], or UnCover Reveal[6]. These services are also restricted to the material offered by the hosting service. Thematically focused portal sites such as Neuroscion[7], BioMedNet[8] underly similar limitations. Most of these are commercial sources. We are convinced, however, that all metadata, including abstracts, will be freely available to the scientific community in the near future.

There is also a number of services specialized in particular types of publications, such as Technical Report Servers (ArXiv[9], REPEC[10], NCSTRL[11]) that offer notification about their documents. Here, the focus of notification is, by definition, restricted.

Some alerting services are offered by libraries (e.g., CISTI Source[12]). Here, the material has a large spectrum covering several publishing houses and different types of publications. Nevertheless, these services are restricted to the material offered by the library.

When subscribing to more than one alerting service, the problem of duplicate notifications due to overlapping coverage arises. In particular, this applies to alerting services of different libraries. The problem of duplicate notifications can be addressed by an intermediate integrative alerting system.

In summary, there are currently a number of services with restricted material, accessible mostly by profiles with poor expressiveness. Even if single providers offer sophisticated profiles to their users they are still restricted to the provider's material.

The scientific research covering Internet-based alerting services that could be used for a digital library environment has a longer tradition. One of the earliest systems developed was SIFT [18], a tool for wide-area information dissemination, that is now commercially operated as InReference. SIFT was a monolithic service that did not support distribution.

Alerting services may be implemented in many ways. Since notification about events is a useful paradigm in different kinds of applications several distributed notification infrastructures have been developed in order to evaluate scalability, examples are JEDI [7], Elvin [16], Siena [4], and NiagaraCQ [5]. The profile definition languages supported by JEDI, Elvin and Siena are too restricted and not appropriate to support sophisticated full-text retrieval. NiagaraCQ supports XML-QL queries that select subtrees in an XML document. In contrast to such tree-based queries, filters in the Hermes application domain typically focus on 'flat' structures like bibliographical references.

The Continual Queries Project [15] investigates update monitoring problems. In this project, the Continual Queries (CQ) language was developed, which has been implemented in several prototypes. This language supports a sophisticated and detailed definition of profiles. CQ is a system that could be used as a basis for an alerting service for digital libraries, as well as other message-oriented middleware.

The academic projects on alerting services introduced above are not all fully implemented. None of these systems is actually used in a digital library context.

Many publishers offer free access to their bibliographic data, usually by sending them via E-mail or by making them available on their Web sites. Unfortunately, the format of this data is not standardized. The problem of extracting bibliographic data from publishers' Web sites can be addressed using methods from the field of semistructured data management [1]. *Wrappers* must be created that allow queries against the data offered by a Web site while hiding layout and structure of this site. Answering such queries means collecting data from multiple linked HTML documents and separating it from the HTML code in which it is embedded. Since HTML is a layout-oriented language, a-priori knowledge about the structure of a Web site and its documents must be employed in the construction of a wrapper. Several approaches such as [2, 3, 11] use regular-expression matching on HTML documents. However, regular expressions are not very robust with respect to layout variations and structural changes that occur frequently in Web sites. Rule-based approaches such as YAT [6] or HyperView [9] that operate on syntax trees of HTML documents provide a higher robustness. The HyperView system that is used in the Hermes project supports the maintenance of wrappers by using a multi-layered approach to separate the concerns of data *extraction* and data *conversion*.

The main contribution of this paper is an architecture for an alerting service which overcomes the heterogeneity problem and – from the user point of view - integrates all kinds of providers into a single source.

The paper is structured as follows: We discuss the issues of implementing an integrative alerting system in the following section. Section 3 presents the architecture of the Hermes system. Section 4 discusses the problems we encountered during development and propose some solutions. The paper concludes with a short outlook.

[1] Springer Link Alert, `http://link.springer.de/alert`
[2] Elsevier Contents Direct, `http://www.elsevier.nl`
[3] SwetsScan, offered by Swets `http://www.swets.nl/`
[4] ISI: Alerting Services, formerly Research Alert Direct, `http://www.isinet.com/prodserv/rad/radp.htm`
[5] Catchword: Internet Publishing Services, `http://www.catchword.com/`
[6] UnCover Reveal, `http://uncweb.carl.org/reveal/`
[7] Neuroscion, `http://www.neuroscion.com/`
[8] BioMedNet, `http://www.bmn.com/`
[9] arXiv.org e-Print archive, `http://www.arxiv.org/`
[10] Research papers in economics, `http://www.repec.org/`
[11] Networked Computer Science Technical Reference Library, `http://www.ncstrl.org/`
[12] Canadian Institute for Scientific and Technical Information, `http://www.nrc.ca/cisti/source/`

Table 1: Provider types

	cooperative	*non-cooperative*
active	sends notification with well-defined metadata	sends human-readable email
passive	allows AS download of well-defined metadata	makes metadata available at its Web site

2. ISSUES

The main problems that have to be solved when building an open integrating alerting system are

1. coping with the heterogeneity of information providers,
2. specifying the users' information needs (user profiles), and
3. efficiently matching the incoming notifications with the user profiles (filtering).

2.1 Heterogeneity

First we define the different types of information providers an integrative alerting system has to handle.

2.1.1 A Classification of Information Providers

Information providers are any suppliers of scholarly information that make metadata (e.g., bibliographical data) available to the alerting service (AS).

We distinguish different types of providers as introduced in [12]. Providers can be either *active* or *passive*. Active providers offer their own AS. For instance, users of the Darwin AS[13] can subscribe to receive an E-mail notification whenever a new issue of a particular journal appears. Passive providers do not offer such a service and have to be queried for new material in a scheduled manner. An algorithm to optimize the query scheduling is introduced in [9].

Additionally, providers can be *cooperative* or *non-cooperative*. Cooperative providers provide their information in one of the standard formats that can be handled automatically by Hermes, and they implement at least one of the protocols supported by Hermes. Non-cooperative providers offer their information in a proprietary format. Specialized wrappers have to be written in that case. Possible combinations of provider types are shown in Table 1.

The distinction between *active* and *passive* is not an exact equivalent to *push vs. pull*. Mixed types occur as well. Consider, for instance, a provider that notifies (push) about new metadata at its FTP server (pull). A similar case is a passive provider together with an active external alerting system such as Darwin that polls the provider on a regular basis and sends notifications containing the URLs of new issues at the provider's Web site.

Passive, non-cooperative providers are most critical for an alerting service. They basically offer an HTML interface.

Active cooperative providers can join the Hermes system by registering and regularly submitting bibliographical metadata in a standardized format.

2.1.2 Heterogeneous Formats

Except for cooperative providers, the alerting system has to deal with various metadata formats. The degree of structure varies from mostly unstructured E-mail notifications in ASCII format over semistructured HTML pages on publisher Web sites to well-structured formats such as XML or the various formats used by citation management software.

[13] `http://darwin.inf.fu-berlin.de`

2.2 User Profiles

Users express their information needs in *profiles*. A profile is an aggregate of a *query* or *filter*, and a *notification policy*. While the query part defines *which* information the user wants to receive the notification policy specifies *how* it is to be delivered, for instance how often (e.g., daily or weekly), by which protocol (e.g., E-mail), and in which format (e.g., XML, HTML, plain text, BibTeX ...).

A query consists of a *Boolean* filtering expression and a *ranking* part. Both parts are optional. The Boolean query is an SQL-like simple attribute-value matching, e.g., "author = Smith" or "title LIKE '%alerting%' ". The set of attributes are the bibliographical metadata fields delivered by the information provider.

Bibliographical data that match the Boolean query can further be ranked according to the ranking part of the query. Such a ranking query is a text-retrieval-like term list (including phrases, proximity operators, or term weights) and a relevance threshold. Documents that are scored by Hermes with a relevance above that threshold are delivered to the user. Alternatively, the user can request the delivery of the top n relevant documents published during the notification period. If the Boolean query part is missing any incoming document is scored according to the ranking query. In this paper we focus on Boolean queries.

Users can create and maintain their profiles using a Web interface. Software clients such as reference managing tools access the service using a dedicated API.

2.3 Filtering Scalability

Incoming bibliographical metadata have to be matched against the user profiles. Obviously, scalability is an issue. While the frequency of incoming notifications in the application domain of digital libraries is usually low, the number of profiles can be very high. Since several 10,000 profiles can be deposited at the alerting service, an efficient matching of profiles and notifications is a crucial task.

3. ARCHITECTURE

Hermes consists of three main components as shown in Figure 1: The Observer, the Filter, and the Notifier.

3.1 Observer

The first task of an integrative alerting system is to collect information from a number of information providers and to produce a combined stream of events that can be filtered against the users' profiles. In the Hermes architecture, the component responsible for this task is called the *observer*.

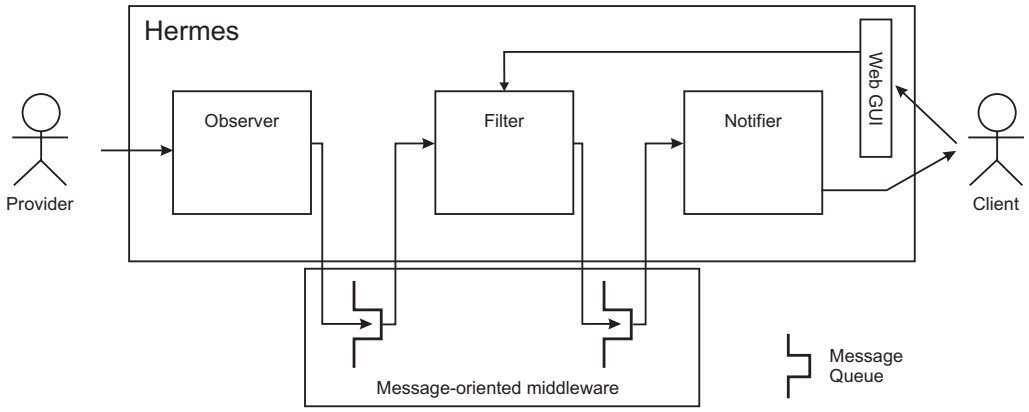

Figure 1: Architecture of the Hermes system

3.1.1 Observer Architecture

The architecture of the observer component (Fig. 2) treats the different types of information providers discussed above in a unified way. Each information provider is handled by a dedicated *wrapper* component. A wrapper consumes a stream of notifications in a provider-specific format and produces a stream of citations in the internal format used by Hermes.

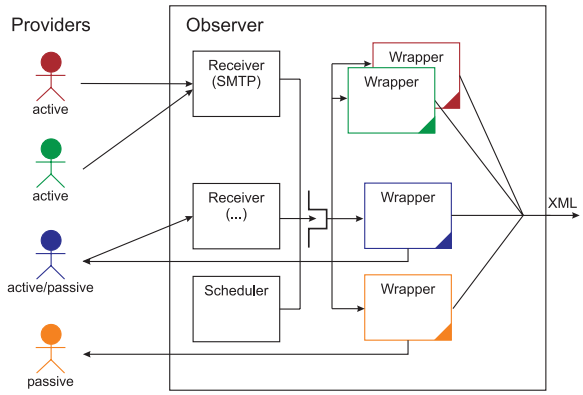

Figure 2: Architecture of the Observer.

In the case of a *cooperative* provider, the wrapper just forwards the incoming notifications since they are already in the internal format. For *non-cooperative* providers the wrapper has to translate the notification into the internal format.

For active providers, the information contained in a notification is sufficient to generate citations. In the case of mixed active/passive types of information providers the wrapper retrieves the material advertised in the notification from an external server of the information provider. For completely passive providers, a synthetic notification is generated by the scheduler component within the observer that causes the wrapper to retrieve new material from the provider.

Since notifications can be sent by the providers using different protocols, the observer contains a component called *receiver* for each protocol. For instance, there exists a receiver in the Hermes system that handles E-mail sent using the SMTP protocol and forwards the content of an E-mail message to the wrappers.

All incoming notifications including those generated by the scheduler form a single stream. Each wrapper subscribes to the notifications originating from the provider it is responsible for (not shown in Fig. 2). This can be easily implemented on top of a message-oriented middleware.

3.1.2 Wrappers

Bibliographical material that can be obtained from non-cooperative providers is typically in a form that resembles a journal's table of contents (TOC). For instance, some publishing houses allow users to browse the TOC of their journals on their Web sites. Others send E-mails containing the journal TOC in plain text. Since the format of the TOC varies between providers, dedicated wrappers are necessary to extract the metadata.

The Hermes system currently uses two types of wrappers: individually coded converters for E-mail notifications of several active providers, and a generic wrapper for the HyperView system [9] that in turn contains several rule-based wrappers for different publisher Web sites.

The first type of wrappers converts E-mails containing tables of contents into citations. Since most publisher's E-mail notifications are highly unstructured, every wrapper has to be implemented individually using a general purpose programming language.

Since HTML pages are more structured than typical E-mail notifications, a higher-level approach based on rules is used for extracting bibliographical data from publisher Web sites: The generic HyperView wrapper consumes XML notifications from the Darwin alerting system that contain references to new journal issues. Each reference contains a URL pointing to the table-of-contents (TOC) page of this journal issue at the publisher's Web site.

The HyperView wrapper responsible for the particular publisher Web site loads the TOC page from the given URL and applies a set of graph-transformation rules to the syntax graph of the page. If necessary, hyperlinks on the page are followed by loading additional pages. As a result, a graph is created that contains the pure data extracted from the analyzed pages. This graph has a source-specific structure.

At the second stage, another set of rules is applied to this graph to transform it into a source-independent representa-

tion of the extracted table of contents. Finally, a third set of rules rewrites the bibliographical data of each article into the syntax tree of an XML document that contains the citation in the internal format used by Hermes. This citation is then returned by the HyperView system and posted by the generic HyperView wrapper to the outgoing stream of citations.

Compared to a hand-coded wrapper in a general-purpose programming language, a HyperView wrapper is smaller and easier to understand and maintain since HyperView rules are on a much higher conceptual level.

3.2 Filter

The Filter's responsibility is the comparison of the bibliographical data with the query part of the user profiles (see 2.2). For simple queries using only Boolean filtering Hermes can use a message-oriented middleware (MOM).

For the first filtering step it is not necessary to respect document structure. Therefore, the filterable attributes are extracted from the XML representation using a SAX parser and assigned to filterable message header fields. For further processing the XML representation is included in the message.

3.3 Notifier

The Notifier retrieves the messages buffered for a user according to the user-defined schedule as specified in the notification part of the user profile. The XML payload of the retrieved messages is transformed to the format preferred by the user. Transformation to various formats is performed using XSL stylesheets.

The transformed notification is delivered to the client via the client's preferred protocol. Currently, only E-mail delivery is implemented.

3.4 Message-Oriented Middleware

The communication between the components is performed using a message-oriented middleware (MOM). This has the advantage that components are relatively independent of each other. Another benefit is that profile matching can be performed by the MOM. The MOM is capable of filtering messages according to a selector that is applied against the message header fields. The query parts of the user profiles are stored as *subscriptions* at the MOM. The MOM filters messages and buffers them until the next notification is due.

The MOM is accessed by the Java Message Service (JMS) API [14].

4. PROBLEMS AND SOLUTIONS

In this section we discuss some of the issues encountered during the design and implementation of Hermes in more detail, namely heterogeneity and scalability.

4.1 Wrappers *vs.* Cooperative Providers

Wrappers for information sources are based on a-priori knowledge of the structure and formats of these sources. This knowledge is typically incomplete because in most cases it must be obtained by a reverse-engineering process. Hence, there is no guarantee that a wrapper can handle all data available from an information provider. Moreover, this knowledge can rapidly become outdated since the information provider is autonomous and can choose to change its structure at every moment without notice.

E-mail notifications by publishers are intended for human users rather than for machine input, so they are often unstructured. This requires heuristics like counting the percentage of single letters in a line to distinguish the list of authors from the title of an article. It is a matter of fact that such heuristics are unsharp and can produce incorrect results.

In the case of information extraction from Web sites, one has to deal with semistructured HTML documents that typically provide more hints to find and separate different data items. The HyperView system used in the Hermes system allows to write rules that are robust to some extent with respect to structural variations such as inserts, deletes, or reorderings on a page. This holds as long as the navigation path used to access the data on a page is not affected by the changes. Unfortunately, major changes in the layout will break a wrapper in most cases.

Since wrappers can break, they have to be monitored permanently. Heuristics can be used to detect, e. g., if a wrapper produces no citations for a new journal issue. An alerting system must support recovery, i.e., after fixing a wrapper it must be possible to feed a notification or parts of it to this wrapper again.

Although our wrappers turned out to be quite stable over several months, it is preferable to cooperate with information providers. In fact, the Hermes project has an agreement with a major scientific publisher to deliver bibliographical data as a cooperative provider.

4.2 Bibliographical Formats and Interoperability Protocols

Cooperative providers offer their metadata in a format that can be automatically processed by Hermes. The delivered data set should at least contain all the information against which the queries can be defined, and a link to the document's full-text. For journal papers this data format is the Majour Header DTD [8].[14]

Active cooperative providers submit their data by sending an E-mail or using the HTTP POST method. Data from passive cooperative providers is loaded by FTP or HTTP GET. In the case of passive providers scheduling of Observer activation must be configured. Since in the domain of electronic journals changes on the provider site occur in relatively stable intervals the Observer activation intervals can be precisely tuned. As mentioned above (3.1) the Observer can easily be extended supporting additional protocols (e.g., Z39.50 or CORBA). Currently, no error handling in the data exchange protocol is implemented.

Different protocols have been proposed for delivery of bibliographic data. One of them is defined by the CrossRef project[15], an initiative to provide a centralized source to obtain object identifiers of electronic publications. The idea is to support publishers or authors to link their bibliography entries directly to the reference's full-texts – even if the full-text is located at a different provider's site. CrossRef defines

[14]The Majour Header is an SGML DTD. Hermes transforms SGML documents to conform to XML. Some minor changes are made to the Majour Header, e.g., the addition of a tag <aloc> containing the location (URL) of the article's full-text.

[15]CrossRef, `http://www.crossref.org`

a protocol for submission of bibliographical data that can be easily adopted by Hermes. Bibliographic data are formatted according to the `doibatch.dtd`, an XML DTD for batch submission. Transfer of the formatted data is done via HTTP POST. An HTML-formatted diagnostic message is returned. A detailed failure report is sent by E-mail. A DTD for this diagnostic error message is available.

The CrossRef format allows to submit a minimal set of bibliographical data that is sufficient to identify a document. Unfortunately, this data is not sufficient to provide personalized filtering. For example, including an article title is optional, and the DTD contains no elements for keywords and abstract. However, many publishers deliver data to CrossRef. Therefore, Hermes will support the CrossRef protocol in the near future to provide an efficient way to let publishers join the service. The only modifications that need to be made are the addition of elements to carry the document's abstract and keywords.

Recently, the Open Archive Initiative (OAI)[16] published a protocol for metadata harvesting [17]. It allows *data providers* like technical report servers, publishing houses, libraries etc. to make their data accessible by *service providers* that build value-added services for these data (e.g., alerting services). The OAI protocol is a *pull* protocol that allows service providers to request metadata records from the data providers. It is based on HTTP and XML and is designed to be so simple that an experienced developer can implement it "within a day of work". Metadata (e.g., bibliographical data) are usually delivered in a Dublin Core format (domain-specific schemata can be defined). Requests allow a simple selection by *set* (semantics of sets are not defined in the protocol), and/or by *date*.

It can be expected that the OAI protocol will be implemented by a significant number of data providers. We will therefore apply the protocol as a means to integrate *passive cooperative* providers.

4.3 Filtering

There are two alternatives for matching document metadata against user profiles: (i) implement the matching algorithms yourself, or (ii) make use of existing infrastructures. We compared two implementations of filtering. The first one uses a message-oriented middleware (MOM). The second is based on tables in a relational database system.

A MOM has the capability not only to perform basic message filtering according to a message selector but includes transaction support as well as message buffering. Therefore, Hermes is built on top of such a MOM. In addition to the filtering capabilities the application of a MOM results in a loose coupling of the components and increases the stability, as failure of a single component does not affect the whole system.

However, the application of a MOM has a number of disadvantages:

- Filtering is restricted to Boolean queries (message selectors). Ranking is not supported.
- The number of subscriptions (i.e., queries) is limited.

The latter is attributed to the fact that incoming messages are assigned immediately to the subscribers that are interested in it to avoid a long delay. This approach does not scale well (for instance, in one product the number of subscribers is limited to 1024).

In [10] an implementation of messaging on top of a relational database is proposed. The solution handles the subscription rules as tuples of the relations, and the publication of a message as a trigger that performs a `select ... from ... where` statement to find all interested subscribers. While this approach is convenient for most applications of messaging systems it restricts the subscription rules to make use of only a small set of comparison operators. However, substring search, which is most important in the field of digital libraries, cannot be supported.

We therefore experimented with a simple implementation of a message queue based on a relational database system that takes advantage of some knowledge of the application domain. The structure of the messages is known in advance, the publication frequency is low, and a message delay up to a few hours is tolerable for users.

A message queue consists of three database tables: a *message* table containing messages (bibliographical data), a *subscriber* table that stores message selectors, and the actual *queue* table where messages are assigned to the subscribers. The message selectors are conditional expression strings that can be applied in the SQL `where` statement to select messages from the *message* table.

Message publication is the insertion of message data into the *message* table. To update a queue two alternatives are conceivable. The most obvious one is the implementation of a trigger for each subscriber that is executed after each insertion of a message. The trigger would match the message with the subscriber's selector and insert a tuple of message id and subscriber id in the *message queue* table (or directly notify the client). Obviously, this approach does not scale. Even with a low frequency of 1 message per second the execution of more than 10,000 triggers is beyond the scope of current database implementations. The alternative of having a single trigger execute a `select` for each subscriber's selector does not solve the problem. Instead, we take advantage of the fact that updating the queue can be deferred for hours. Once in a while (e.g., once a day, which is enough for notification on scientific publications) the *queue* is updated by iterating through all subscribers, selecting the messages that are of interest for the subscriber, and inserting the resulting tuples of message id and subscriber id into the *queue*. Selection and insertion can be performed by a single SQL statement within a short execution time. The time required to update the queue for 10,000 messages and 10,000 subscribers with a message selectivity 0.1 % is in the order of 3 hours (Oracle 8.1.6, Sun Enterprise 450 with 2 processors and 1 GB main memory). A subscriber receives its messages by performing a natural join on the *message* and *queue* tables and selecting the messages that are dedicated to the subscriber (as indicated by the subscriber id in the *queue*).

An API to access the queues that conforms to a subset of the JMS API can easily be implemented.

Existing infrastructures like database systems or message queuing systems allow the easy implementation of simple Boolean filtering. However, for text documents (bibliographical data can be seen as text documents regarding title and abstract) this kind of query leads to low precision and recall. More sophisticated filtering is therefore necessary.

[16] `http://www.openarchives.org/`

4.4 Ranking

As mentioned above simple attribute-value matching as it is performed by the available message-oriented middleware products is far from satisfying the users' needs. Since most sources of bibliographical data provide document abstracts in addition to author, title, etc., one can achieve much higher precision by applying classical information retrieval methods.

In information retrieval document relevance is usually measured by two parameters. The term frequency tf, an indicator of how often the query terms occur in the document, positively influences the relevance scores. The inverse document frequency *idf* indicates the relevance of the search terms with respect to the document collection (in how many documents of the collection does each search term occur?) and influences the score negatively. The problem is that in an alerting service documents are 'transient events' and therefore no document collection exists. To mimic such a collection a 'virtual' collection can be built: All incoming documents are inserted into that virtual collection. The *idf* is computed for that virtual collection. Since access to older documents is not required it is not necessary to keep the actual documents in that collection. Instead it is sufficient to store the term statistics. Topics covered by an alerting service can evolve over time. It may therefore improve the filtering effectiveness to let the term statistics of the virtual collection evolve as well.

5. CONCLUSIONS AND OUTLOOK

Alerting systems offer a necessary means to cope with the ever-increasing amount of scholarly literature. Existing alerting systems in this field are mostly proprietary and often of limited coverage and functionality. Scholars need an open integrative alerting system that (i) combines bibliographical metadata from various sources to maximize coverage and that (ii) offers sophisticated filtering and notification facilities to achieve a high selectivity and usability.

In this paper we have introduced an open integrative alerting system, the Hermes alerting service for digital libraries. An instance of the service is available to the public at `http://hermes.inf.fu-berlin.de`. This instance is currently covering several hundred scientific journals from major publishers, including our project partners Springer and Nature, as well as the technical report servers NCSTRL and ArXiv. Typically, such an alerting system would be operated by a scientific library as a service for its users.

The main issues that are addressed by the Hermes project are the heterogeneity of the information providers, and information filtering according to the user's need.
For the provider integration Hermes follows two alternative approaches. For providers that disclose their bibliographic metadata but not in a well-defined format, Hermes provides appropriate wrappers. Since wrappers are subject to failure due to format changes, Hermes allows providers to join the alerting service with little effort by implementing a simple protocol for data delivery.
Filtering can be implemented with low effort on the basis of message-oriented middleware or relational database systems, but this is restricted to simple Boolean filtering.

In the future we will focus on the following issues: improvement of filtering quality, scalability by distributed execution, duplicate elimination, and relevance feedback.

Improving the filtering quality can be achieved by applying the methods of text retrieval to alerting.

Hermes is designed for scalability. Its loosely coupled components can each be deployed in multiple instances and thus share their work. Related projects [4] have shown that cooperating alerting services can improve scalability. Consider an instance of an alerting service that receives metadata from publishers and propagates it to topic-specific instances. Further research is necessary to find methods to compute the *covering relations* [4] of profiles in a digital libraries.

Duplicate elimination will become necessary once we add information providers with overlapping coverage. Due to different metadata formats, heuristics have to be used to identify different citations of the same publication.

Relevance feedback will allow users to achieve a higher selectivity by grading and returning notifications to the alerting systems. For this purpose we plan to adapt existing approaches in classical information retrieval to relevance feedback.

6. ACKNOWLEDGMENT

The Hermes Project is funded by the German Research Ministry (BMBF) within the Global Info initiative.

7. REFERENCES

[1] S. Abiteboul. Querying semi-structured data. In *ICDT*, volume 6, pages 1–18, 1997.

[2] N. Ashish and C. Knoblock. Wrapper generation for semi-structured internet sources. In *Proc. Workshop on Management of Semistructured Data*, Tucson, 1997.

[3] P. Atzeni and G. Mecca. Cut & paste. In *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 12–15, Tucson, Arizona, 1997.

[4] A. Carzaniga. *Architectures for an Event Notification Service Scalable to Wide-area Networks*. PhD thesis, Politecnico di Milano, Milano, Italy, Dec. 1998.

[5] J. Chen, D. DeWitt, F. Tian, and Y. Wang. NiagaraCQ: A scalable continuous query system for internet databases. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, 2000.

[6] S. Cluet, C. Delobel, J. Siméon, and K. Smaga. Your mediators need data conversion. In L. M. Haas and A. Tiwary, editors, *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*. ACM Press, 1998.

[7] G. Cugola, E. Di Nitto, and A. Fuggetta. Exploiting an event-based infrastructure to develop complex distributed systems. In *Proceedings of the 20th International Conference On Software Engineering (ICSE98)*, Kyoto, Japan, Apr. 1998.

[8] European Workgroup on SGML. MAJOUR Header DTD, version 1.1. Software.

[9] L. C. Faulstich and M. Spiliopoulou. Building HyperView wrappers for publisher web-sites. *International Journal on Digital Libraries*, 3(1):3–18, 2000.

[10] J. Freytag, F. Leymann, D. Roller, and M. Stillger. Publish/subscribe functions based on object-relational features. Humboldt University Berlin, Computer Science department, 2000, in preparation.

[11] J. Hammer, H. Garcia-Molina, J. Cho, R. Aranha, and A. Crespo. Extracting semistructured information from the web. In *Proc. Workshop on Management of Semistructured Data*, Tucson, 1997.

[12] A. Hinze and D. Faensen. A unified model of internet scale alerting services. In Hui and Lee [13].

[13] L. C.-K. Hui and D. L. Lee, editors. *Internet Applications. 5th International Computer Science Conference, ICSC'99, Hong Kong, China, December 1999*, volume 1749 of *Lecture Notes in Computer Science*. Springer, 1999.

[14] *Java Message Service API*. `http://www.javasoft.com/products/jms/`.

[15] L. Liu, C. Pu, and W. Tang. Supporting internet applications beyond browsing: Trigger processing and change notification. In Hui and Lee [13].

[16] B. Segall and D. Arnold. Elvin has left the building: A publish/subscribe notification service with quenching. In *Proceedings of the Australian Unix and Open Systems User Group Conference (AUUG97)*, 1997.

[17] H. Van de Sompel and C. Lagoze. The Open Archives Initiative protocol for metadata harvesting. Protocol Specification 2001-01-21, Open Archives Initiative, Jan. 2001. `http://www.openarchives.org/OAI/openarchivesprotocol.htm`.

[18] T. W. Yan and H. García-Molina. SIFT - a tool for wide-area information dissemination. In *USENIX 1995 Technical Conference on UNIX and Advanced Computing Systems, Conference Proceedings*, pages 177–186, New Orleans, Louisiana, Jan. 1995. USENIX Association, Berkeley, CA, USA.