Investigating Link Service Infrastructures

David C. De Roure
Department of Electronics and Computer Science
University of Southampton,
Southampton, SO17 1BJ, UK
Tel: +44 23 8059 2418
E-mail: dder@ecs.soton.ac.uk

Nigel G. Walker
British Telecom Laboratories
Adastral Park,
Martlesham Heath
Ipswich IP5 3RE, UK
Tel: +44 1473 644853
E-mail: nigel.g.walker@bt.com

Leslie A. Carr
Department of Electronics and Computer Science
University of Southampton,
Southampton, SO17 1BJ, UK
Tel: +44 23 8059 4479
E-mail: lac@ecs.soton.ac.uk

ABSTRACT
Variations on the Distributed Link Service have now been deployed across a spectrum of hypermedia and multimedia projects. Although some implementations have utilised standard database technologies and hypermedia tools behind the scenes, most of the network services have been proprietary implementations. In this paper we discuss the motivation and requirements for a large scale, dynamic and open distributed link service using third party components, and explore the use of off-the-shelf services to provide the distributed infrastructure for link services. In particular we investigate HTTP, LDAP and Whois++ as candidate technologies.

KEYWORDS: open hypermedia, link service, distributed link service, directory services, query routing, LDAP, Whois++.

INTRODUCTION
At its simplest, a hypermedia link server takes a source anchor in a multimedia document and returns the possible destination anchors, obtained by interrogating a link database (henceforth a *linkbase*) for links containing that anchor. The anchors might identify specific locations or objects in particular multimedia documents; alternatively they might have broader applicability, matching content rather than position (so-called *generic* linking). The linkbase query might also be refined by the user's context, perhaps based on their profile, current role, task and location. Link services may be accessed before, during or after document delivery, and they may provide an interface for link creation and maintenance as well as retrieval.

Such link servers support configurable and extensible hyperstructure which enhances navigation in information applications. Content-based linking in particular provides powerful authoring functionality, integrating information retrieval with linking and supporting a more query-oriented mode of interaction [13]. Content-based navigation of an information space involving multimedia data involves specialised feature extraction and matching techniques [15] and these can be integrated within the link resolution framework.

In our early systems, link resolution was achieved by components in a standalone application, such as the filters of the Microcosm system [14]. Subsequently, it has been abstracted out as the duty of a link service that can be seen as a 'third party' network service. The original Distributed Link Service (DLS), presented in [4], supported multiple link databases on one DLS server, providing useful modularity in the hyperstructure. The extension to multiple servers was discussed in [8]. DLS architecture has evolved through a number of projects, notably the Open Journals project [5]. Meanwhile, the Open Hypermedia Protocol (OHP) is being developed to promote interoperability between hypermedia systems [6].

Ongoing developments in networking infrastructure bring new requirements to the link services, particularly with respect to distribution of the link information. In particular we need to provide link services to users with a spectrum of connectivity beyond the intranets that have characterised previous projects. This includes home users, users connecting via service providers and through firewalls, and mobile users with limited bandwidth and intermittent connectivity. In fact these more restrictive modes of connection often apply to the users who stand to benefit most from an effective link service, providing a customised information space to improve the effectiveness of their interaction. Longer term projects extend these requirements further with pervasive multimedia, including for example wearable computers and everyday artefacts capable of communication – in short, we are interested in what we call the *pervasive information fabric*, by which we mean the middleware for future information applications in this pervasive computing setting.

The open hypermedia principle that 'readers are authors' emphasises the linkbase update requirement – we are

Hypertext 2000, San Antonio, TX.

dealing not only with link resolution but with link authoring and maintenance of link integrity [7]. This implies a dynamic system with updates to the link information due to changes in the hyperstructure and also in document location and content. When the link information is distributed, as it must be to provide scalability in the contexts described above, maintenance of that information becomes a distributed systems issue. For example, if linkbase data is replicated for scalability, there is a linkbase coherency problem to be addressed when that data must be updated.

Another characteristic of new projects is that increasingly we work with multiple linkbases and linkservices, including 'legacy linkbases'. An individual linkbase typically contains the links applicable to a certain set of documents. This 'domain' of the database is likely to be a consequence of a combination of organisational, administrative and technical issues. For example, the following link databases have emerged across a variety of projects:

- Links associated with a given user;
- Links associated with a subject area;
- Links associated with a given administrative domain; e.g. department;
- Links supporting a specific task; e.g. teaching;
- Links associated with a given group of users; e.g. shared bookmarks;
- Links maintained by a given publisher of information;
- Links required specialised search algorithms, such as feature matching.

Some of these suggest a strict hierarchical arrangement, perhaps reflecting organisational hierarchy or subject taxonomy. However, in general the linkbases do not fall naturally into a hierarchy. They are also heterogeneous in the sense that they carry different kinds of information and might even respond to different types of query, with different search algorithms to deal with features of multimedia content. These examples emphasise the contrast between link services and network services such as the Domain Name Service (DNS), where information is partitioned across relatively homogeneous servers with network architecture, information architecture and even organisational architecture mirroring each other.

Directory services are network services that provide access to information such as telephone numbers and email addresses. They are designed to meet many of the requirements above, and are configurable with domain-specific schema. As a linkserver access technology they lie on a spectrum between HTTP and OHP – HTTP is a ubiquitous protocol for accessing servers and imposes no constraint on the content, while OHP includes a data access protocol which itself is highly customised to linkbase access. Although HTTP has been used in existing link services (including DLS [4] and Chimera [2]), directory services appear to be a strong candidate for distributed link service infrastructure and this is the motivation for this study.

The Lightweight Directory Access Protocol (LDAP) [21] is an open, standard protocol for accessing information services, originally developed as a lightweight front end to the (heavyweight) ISO standard X.500 directory service. It has become an Internet standard and we have chosen LDAP for one of our experiments due to its wide availability. Whois++ [10][20] is a more experimental system with features extending beyond LDAP, and has led to the Common Indexing Protocol (CIP) which achieved RFC status in 1999 [1].

This paper describes our investigation of HTTP, LDAP and Whois++ as candidate technologies for distributed linkbases. The next section discusses query processing in a distributed context, and is followed by a description of our framework for comparing link service components. We then describe our experiences of using HTTP, LDAP and Whois++ to provide the link service infrastructure. After a discussion of replication issues, the paper concludes with a qualitative comparison and discussion of future work.

DISTRIBUTED LINKBASES AND QUERY ROUTING

Consider the situation in which the linkbase is distributed across multiple servers. When resolving a query, a server may, using its local knowledge, decide that the query should be sent to a different server. In HTTP there are two scenarios: it may propagate the query directly, as a Web proxy propagates a request down the chain towards a Web server, or it may send an HTTP redirect message back to the client. In directory services, the former is called *delegation* and the latter *referral*.

The delegation model is illustrated in figure 1. The responsibility for resolving the request from A is delegated by B to the server C, which may itself initiate further requests.

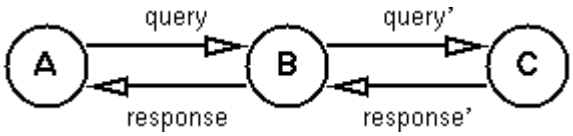

Figure 1: Delegation.

In the referral scenario, a link service request from A to link server B can result in A being referred by B to another link server C. This is illustrated in figure 2. There is no delegation; the responsibility remains with A, which contacts B and then C directly. Hence A retains control and B does not need the capability to call other servers and process results.

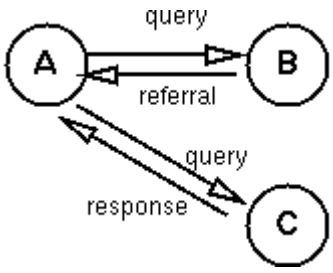

Figure 2: Referrals.

The referral scenario might not be advantageous if, for example, C is local to B and not to A, and so in general we conceive of hybrid approaches. A third scenario involves a request propagated from B to C and the results returned directly from C to A. This is feasible in an asynchronous message passing model, with peer-to-peer communication (as with email, for example), whereas we are focusing here on client-server models as reflecting current practice in directory services and the Web infrastructure.

Query Routing

In the examples above, a server relies on its local knowledge, either to respond to a request, or to identify other servers. The server can better perform this task if we augment this knowledge with summary information about other servers. Taking this a stage further, we can conceive of servers whose sole purpose is to carry knowledge about other servers and route requests accordingly. These then are techniques for *query routing*, the process by which a query converges on a set of servers that contain relevant data, in our case relevant links. This technique is also found in federated databases and metasearch engines [22]; the full process involves database selection, query evaluation and result merging. Note that we generally refer to requests rather than queries, in line with HTTP terminology but also to avoid the implication that transactions only involve information retrieval – we do not preclude the routing of update requests too.

Typically a query routing architecture involves the base servers (in our case link servers) at the leaves together with index servers which return referrals (e.g. B in figure 2). A referral directs the client at base servers that might be able to resolve the query, or other index servers which might carry relevant information to find an appropriate base server. These components are usually arranged in a mesh.

For server B to refer the client to server C, server B must have *forward knowledge* of C. This is some form of summary of the information carried by C or, in general, an indication of the type of queries that can be processed by C. For effective query routing, forward knowledge should enable a server to return referrals that route the query to all base servers capable of resolving the query. For efficiency it is also desirable that it returns referrals to as few 'unhelpful' servers as possible, and that referrals point directly to base servers rather than to index servers.

Forward knowledge could be preconfigured to 'manage' the routing, but for a flexible, dynamic system some form of server-to-server exchange of summary information is required. Consider the example of server B referring a client to server C – this is achieved by establishing a polling relationship between B and C, with B obtaining a summary object from C whenever C is updated.

Although X.500 has a concept of a referral, LDAP (version 2) does not support this. Hence for our request routing experiments we have employed Whois++ [10][20]. The distributed indexing model of Whois++ has evolved into the more general Common Indexing Protocol (CIP) [1], providing an orthogonal mechanism for exchange of summary objects which are not necessarily oriented around template based databases.

COMPARISON FRAMEWORK

The link services developed in various projects have been tested and evaluated in several ways. Generally we have favoured the use of exemplar applications as case studies. By contrast, here we seek an application-independent approach, a generic framework that can be used to support comparison of alternative link service implementations. Although in this paper we emphasise qualitiative comparison, quantitative experiments should also be possible.

Part of our methodology is to use the Web as our control experiment: we abstract links from the Web, reinsert them using the link service interactively, then compare the performance of the resulting system with the original Web interaction. This enables us to compare performance and scalability: for example, we expect a reduction in performance while wishing response time to be suitable for interactive use, and the issues of scaling with respect to linkbase size, numbers of clients and wider area access need to be understood. The second part of our methodology is to see that we achieve the benefits of the enhanced navigation provided by the open hypermedia approach, including generic linking.

Although we have chosen to derive the linkbase data from the Web, in making this decision we are conscious that the Web hyperstructure might not cover all linking practices and we must continue to be informed by other hypermedia systems. Hence we call upon additional sources of linkbase data in order to bootstrap our experiments, including the linkbases described in the introduction. We also investigate response to updates by editing linkbases manually or generating new linkbases.

To import sets of HTML documents into the framework, we use a robot which walks a web site, saving link data in a relational database and making local copies of the documents with the anchors (HREFs) rewritten according to the requirements of the experiment. The SQL database facilitates the subsequent generation of multiple linkbases.

The current implementation is based on *libwww* from W3C and runs on FreeBSD and linux. It uses the *mysql* database, with *www-sql* to facilitate interaction with the database. This has proven to be a very versatile combination of tools.

For the replication of Web functionality we use two HTML rewrite scenarios. The first preserves the original appearance of the Web links but forces link resolution via a link service, with the queries specified as URIs embedded in the HTML document. The second is a mechanism for naming anchors, which can then be used in a separate link database. The methods can be used in combination. An example of each of these rewrite methods is given below:

Rewrite method 1

HREFs are rewritten to invoke a service that is part of the experiment, so that Web links still appear in a Web browser. For example, in the file `foo.html` on server `www.deroure.org`

```
<A HREF="bar.html#thing">here</A>
```

is rewritten to

```
<A HREF="S1234">here</A>
```

and a new base URI identifying the service is provided.

Rewrite method 2

HREFs are replaced by NAMEs, so that each source anchor can be denoted by a URI. For each such rewrite, a pair of URIs is generated, corresponding to the source and destination anchors of the original link. For example, in the file `foo.html` on server `www.deroure.org` the anchor

```
<A HREF="bar.html#thing">here</A>
```

is rewritten to

```
<A NAME="S1234">here</A>
```

generating link number 1234 with source anchor

```
http://www.deroure.org/foo.html#S1234
```

and destination anchor

```
http://www.deroure.org/bar.html#thing
```

The experimental framework also features a Web proxy which rewrites fragments of HTML as documents pass through in order to insert links on-the-fly, without waiting to parse the entire document.

Extension to content-based techniques

The framework described above enables the user to follow a link from a location in a document only if there was an original link from that location. As discussed in the introduction, generic linking enables the user to determine available links based on the *content* at the current location. In the case of text, a simple content-based technique is to use a word selected by the user as the basis for a query; this could, for example, provide a context-sensitive glossary facility. In the general multimedia context, content-based techniques require feature extraction followed by interrogation of a link database using matching techniques specific to the feature – this is discussed in the next subsection.

There is a crude but effective technique for linking from individual words and it further illustrates the first rewrite technique described above: every word can be rewritten as an anchor with an HREF which initiates the appropriate query. For example

```
deroure
```

is rewritten to

```
<A HREF="words?deroure">deroure</A>
```

This is a somewhat extreme scenario that causes massive HTML inflation (and very blue text!) but has proven invaluable. For example, it was used successfully in a tool to assist in creating linkbases to link speech recordings with their text transcripts. In the authoring process, the user listened to the audio and clicked on words so that the server could record the time at which they occurred; subsequently, clicking on the same words caused the audio to jump to that point. This serves as a reminder that we are interested in both link creation and retrieval.

Fortunately it is possible to achieve a more flexible approach taking advantage of contemporary browser facilities, so that the user can mark an arbitrary region of text and this is then submitted automatically to an engine which in turn queries the link service. This sort of integration has been achieved successfully in other projects (such as [12]) using Javascript, Java and DOM (the W3C Document Object Model).

Extension to other media types

The techniques described above are specific to HTML and related formats. In practice, a link service is used with a wide variety of content types and it is therefore necessary to address this in the experimental framework. In general there are two kinds of information which may form part of a query for available links: the *location* in a multimedia document and the *content* at that location; this is augmented by information about the context of the query, including for example the name of the document, the user, their role and location. These information sources are depicted in figure 3.

Positions can take various forms; for example a byte offset or range of bytes, a point or region in an image, a time or duration in a stream [17]. Within a web browser we can use the image map mechanism to generate position information for images. We have previously demonstrated a generic media player application for use outside a browser: clicking the *link* button generates a source URI based on the current document URI and the current play time [3].

Basic hypermedia facilities can be expressed using position information. However, once a fragment of

multimedia content has been identified through the user interface by whatever mechanism, it can be submitted to a feature extractor to obtain a content representation suitable for matching. It is possible that this representation will be textual in which case it may be possible to use a standard link service. Position information is sometimes textual too, for example if objects have identifiers; e.g. in VRML.

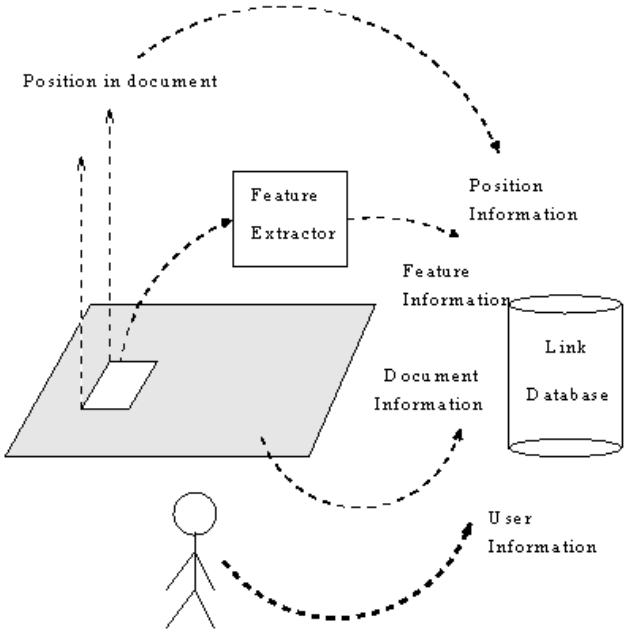

Figure 3: Information sources

However, in general the processing of position information (such as an interval or region) or extracted features (perhaps a colour histogram) requires special processing according to content type. In addition, the matching process may need to be parameterised, perhaps to provide a threshold or to guide query expansion, and the results may carry additional information about the quality of the match.. This means the link service may need to deal with arbitrary queries and route them to specialised matching engines, and potentially support aggregation of ranked results.

For our experiments we use the melodic contour engine [3][11] as an example of a content-based technique. A contour is an established abstraction of a sequence of musical notes, indicating the direction of note transitions but not absolute pitches or sizes of intervals. Contours have the advantage that they are represented as text strings (with a small alphabet) and are therefore easy to work with as a feature. They can be readily derived from MIDI files. The engine is fairly typical of other content-specific engines in accepting search parameters and returning ranked results. In fact the engine uses query expansion rather than fuzzy matching, in order to provide response times suitable for interactive use.

HTTP EXPERIMENT

The basic model of a DLS linkserver is that a source anchor is provided as an HTTP GET request and the linkserver responds with a set of destinations extracted from its local link data [4]. Hence when the user clicks on a Web link, an "available links" display is produced. In fact, for links with one destination it is possible to use a redirection to provide an 'autofollow' facility, maintaining the illusion of following the original link. Queries can be based on individual words or regions of text, and this extends to the whole document – in fact DLS can function in Web proxy mode, inserting links on-the-fly using its local link data.

The first rewriting method tests the basic model. The user notices no difference in the interface but the link service is exercised in the process of link following. With the linkserver local to the client on the network, we observe no appreciable delay in link following except for linking to destinations within the same document.

The proxy approach can apply much greater stress to the link service. With a DLS linkserver itself in proxy mode, every word in an HTML document is looked up locally in the link data as it flows through the proxy, and the destination links inserted. Despite the apparent computational intensity of this approach, we observe that it works very effectively, with no significant delay on web pages of typical sizes. This is partly achieved by loading the linkbase data into a runtime data structure for fast access; however, live linkbase updates are then complicated by the need to maintain consistency between the runtime and persistent forms of the linkbase data.

However, in the comparison framework with the proxy decoupled and interrogating the linkserver via HTTP for every word (rather than through an internal API), the delay is several seconds for average Web documents. This clearly exceeds acceptable interactive response times. Using the second rewrite method, the proxy can be used to insert only the links that were present in the original document; the slowing effect is still noticeable on pages that would otherwise load instantly.

By using HTTP we gain advantage from the Web infrastructure. We can configure the system to use multiple linkservers via the mappings provided in the HTTP server configuration; we can pull remote linkbase data over to the local linkserver using HTTP and taking advantage of the Web caching. By making linkserver queries appear as requests for cacheable documents, we can even cache the results of those queries, though this does require any relevant user context to be explicitly encoded in the URI.

LDAP EXPERIMENT

While HTTP provides a mechanism for querying linkbases and the Web infrastructure provides a means of replicating link data to improve retrieval performance, there is no automatic support for use of multiple link servers and aggregation of query results from multiple

sources. However, we do wish to work with multiple servers, both to support diverse link data and to provide servers which are local to clients for efficient access. These requirements appear to be closely matched to directory services. Instead of querying the service with a name to retrieve an email address, we wish to query it with a source anchor to retrieve a destination. LDAP, with its configurable schema and support for integrating multiple servers, is a strong candidate for such a system. It also has the attraction of widespread support, including provision for LDAP access built into Web clients.

Our system consists of LDAP servers extended with templates for link objects. The simplest experimental link object is defined as follows:

```
item cis "Link ID"         cn
item url "Source anchor"   source
item url "Dest anchor"     destination
```

i.e. a link ID string and URIs representing the source and destination anchors. The implementation uses the *Open LDAP* source (version 1.2) based on the LDAP implementation from University of Michigan. The system was configured with experimental link schema, and employed the 'slurpd' mechanism for replicating dynamically updated directory data. It was tested from Netscape Communicator 4.08 and 4.5, which have support for LDAP including a user interface that can be customised for new attributes, and an HTTP to LDAP gateway (*web500gw*) distributed with the Open LDAP release.

The experiment in reconstructing Web link structure via the link service provided confirmation of the viability of using LDAP in this role. Latency issues are much as with HTTP, with individual queries to locally networked link services not causing noticeable delay, but repeated access (as with the proxy) causing delays that are significant in terms of interactive response times. With two servers we successfully demonstrated cloning one LDAP server and also configuring one server to delegate to another.

LDAP has support for expressing complex queries and for filtering results, and we believe these to be useful for constraining the search space using information about the context of the query. We also found the LDAP system to be effective for generic linking from text documents, as it has some simple matching algorithms built in. For example, substring matching enables multiple links to be identified from a single query word and context. A 'sounds like' mechanism also proved useful and we anticipate that this will be useful in applications involving text transcription of the spoken word.

Content-based searches are possible by introducing other attributes but, with standard LDAP servers, the service is most effective if these map naturally to simple text attributes. For example, the contour engine uses 14 letter strings on an alphabet of three symbols (U, D, R) which can easily be handled and searched for within LDAP. On the other hand, we have developed highly specialised algorithms for pitch contour matching that, for example, account for common errors.

Note there is a distinction between restrictions of the LDAP protocol itself and the limited functionality of an LDAP server implementation. The protocol is quite general and can handle searches based on tokens or text, so content-based searching could be incorporated within an extensible LDAP server. Alternatively, a content-based retrieval engine could be given an LDAP interface.

WHOIS++ EXPERIMENT

LDAP (version 2) does not support referrals. Whois++ [10][20], although still a more experimental system than LDAP, does have this support. Hence Whois++ implements query routing, and a form of 'forward knowledge' is needed to support this, with servers polling each other for summary information.

Whois++ adopts a template model with attribute-value pairs. Two template formats for links have been evaluated. The first is based on the database schema used in [9], with host, path and document as separate attributes (and a status attribute for repair and the housekeeping of ‘garbage collection’). The second is based on the summary object interchange format (SOIF), introduced in the Harvest project, with source and destination URIs and available metadata; this format effectively treats links as one would individual documents.

In the prototype interface, following a link resulted in a display of ‘available links’ as usual in DLS (via the browser but in a separate window), but now including ‘available link servers’ corresponding directly to the referral responses to the query; i.e. a series of SERVER-TO-ASK responses. The user can then submit the query to any of these servers. This interface, shown in figure 4, makes the query routing explicit and is probably most appropriate for system maintainers and link database authors; we expect many applications to use an agent rather than a human in this loop.

Our original query routing infrastructure was the Digger software from Bunyip Information Systems, which supports Whois++ and a version of CIP. This was ported to PHP3.0 (an HTML-embedded scripting language) and used both as a link database and as an index server, separately and in combination. An existing link service (early DLS, with a flat file linkbase) was then integrated by adding a script to generate a summary from the link data, and the link service protocol extended with a new method to obtain the summary; a new summary was automatically generated on linkbase update. We have subsquently experimented with two further Whois++ implementations, in Java and PERL.

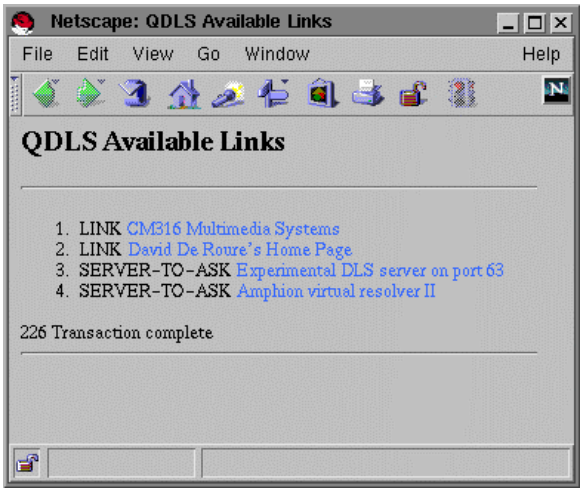

Figure 4: The Available links interface.

In terms of recreating the Web linking, Whois++ performed much as LDAP. However, the advantage of the referral model is that the routing decisions are made closer to the client and we believe this will be significant in some applications.

Forward knowledge for a link service

Effective routing relies on good forward knowledge, and the issue of forward knowledge for a link service is a research issue in itself. The summary object (also called a *centroid*) comprises:

- A list of the templates in the server;
- The attributes in the templates;
- A list of unique tokens for each attribute.

With the summary containing all the unique anchors in a linkbase, exact string matching requires queries with complete anchors; a more sophisticated matching algorithm might determine referrals based on partial matches. When we break each anchor into hostname, pathname and document name attributes, the summary information comprises three lists of unique tokens; i.e. hosts, paths and document names. Referrals then expand the search space to include base servers with anchors containing matching components. The trade-off is that referrals to inappropriate servers become likely (so-called negative referrals).

Query (request) routing is attractive for integrating diverse link services, and since some of this diversity comes from working with multimedia content we must extend the notion of forward knowledge to include different media types. For pitch contours we split the contour data into two engines, each with a summary comprising several thousand contours. This is a highly inefficient form of forward knowledge: we would prefer to be able to describe the partitioning by a pattern or simple predicate, and the example suggests that not all media types are suited to extraction of forward knowledge. This exercise also uncovered another aspect that is relevant here: the engine deals with approximate matching by query expansion (a 'fuzzy query') and we identified the need to decouple the query expander from the lookup code so that the generated queries can be routed independently.

The reader is also referred to [18] for a useful discussion of content-routing. A generalisation of Whois++ to the Common Indexing Protocol (CIP) has now achieved RFC status [1]. CIP is not limited to directory services and arbitrary summary objects are possible. These extensions are attractive in our particular application, particularly for multimedia content.

REPLICATION ISSUES

One of the attractions of using a directory service within a link service is the potential to address our requirement of providing distributed information with good availability and of maintaining coherency when the information is updated. For example, a service such as DNS is designed for availability and performance but infrequent update, whereas some of our applications involve frequent dynamic update.

To investigate both scalability and the interplay between the query routing architecture (e.g. index server locations) and the network architecture, we are conducting simulations. This permits the experiment to be scaled to much larger numbers of servers. For example, we consider a scenario with linkbases on home PCs interconnected by low bandwidth links to a higher speed backbone.

The simulator was built in Lisp, which deals comfortably with attribute-value pairs and symbolic representation of forward knowledge. A request consists of a set of attribute-value pairs, and the index server compares the set of attributes (e.g. user, document, anchor) with each linkbase summary in turn. The common attributes, if any, are each checked for the value in the query being a member of the set of possible values associated with that attribute in the linkbase; if the value exists for all common attributes then a referral is generated. The simulation enables us to explore optimisation of the search strategy, incorporating an explicit representation of the network with associated wide area transmission costs.

As a simple example, figure 5 shows the results of an experiment exploring a heuristic for optimising link resolution. Forward knowledge tables are automatically replicated in the direction of hosts which send numerous requests, moving one hop whenever the number of requests exceeds a threshold.

This particular simulation involves 20 link servers, interconnected by a fixed network, subjected to up to 200 initial random queries. The X axis shows the number of requests submitted by clients to the service and the Y axis show the plot of the total distance covered by all messages to resolve that number of requests (the 'total query mileage'), in arbitrary units. Points indicated by squares were recorded without automatic table replication;

triangles show the results with automatic replication enabled. Multiple runs result in different results due to the random test data, and with this heuristic the wide variation in the results with replication shows over-sensitivity to the initial sequence of queries.

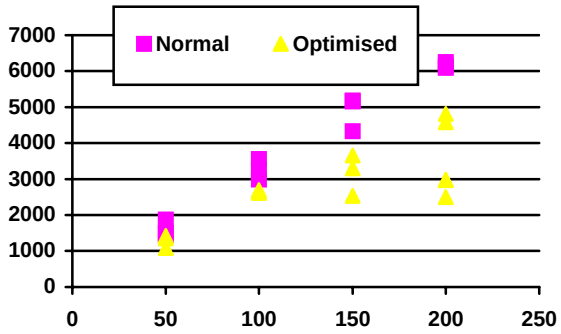

Figure 5: Query transport costs with and without table replication.

The simulator has enabled us to investigate hybrid models which include delegation and referrals. For example, one server can hide multiple servers behind it and respond as a single virtual server, including provision of aggregated summary information, dealing with updates and concealing specific location information.

RESULTS

Table 1 presents a summary of our initial findings regarding the suitability of HTTP, LDAP and Whois++ to provide a link service.

We found the LDAP model to be well suited to link server access. One of the key reasons for building hypertext using a DLS is that there is value in having groups of links together, so they can be easily searched according to various criteria. Directories are more suited to offering such search facilities than a HTTP server storing links: the 0HTTP route requires development of many additional tools. Whois++ extends the directory service with referrals, which we have found to be an effective model in link services. However, Whois++ is not yet widely supported.

Consistently, intense link resolution (as required by the proxy) is sufficiently slow that it has a negative effect on the interactivity of the system. This is largely due to network latency and to an extent can be overcome by formulating multiple requests in one transaction. The established solution is to take the resolution out of the interaction loop by displaying links asynchronously. Another idea is to look to directory services as providers of linkbases rather than individual links. In other words, the query causes the directory service to generate a custom linkbase, and this is accessed locally by the resolver (via an API rather than as a network service).

CURRENT WORK

In our experiments we have not fully addressed a number of aspects including the management model (who owns what information and what security model do we need?) and the user interface (can the referrals be dealt with by a personal assistant agent?) These will be explored through case studies, which should also provide more evidence about the design of forward knowledge and on the balance between delegation and referral.

We are currently investigating a more general approach to accommodate the useful ideas in query routing. In multiagent systems, agents are autonomous and must announce their availability and capabilities to other agents; in fact there are formal models for agents carrying information about other agents, which suggest a basis for query routing. Furthermore, there is a well established technique of 'wrapping' legacy systems to integrate them with agent systems. Hence our approach is to take an agent framework and construct agents to perform the link resolution tasks [16].

	Advantages	Disadvantages
HTTP	Near-universal support Automatic replication through caching of link data Integrates multiple servers through HTTP redirects Possibility of caching query results	Difficult to process queries on multiple servers and aggregate results
LDAP	Wide support Integrates multiple servers Retrieves multiple entries Servers have support for replication	No referrals, distributed information tree must be managed Default matching algorithms suitable for generic linking in text but not other media
Whois++	Best support of query routing Extended to multimedia content summaries through Common Indexing Protocol	Not widely supported

Table 1. Qualitative comparison of link services

In the second section we mentioned the client-server model *vs.* asynchronous message passing. Mapping these ideas to internet protocols, we note that client-server is advantageous when connections can persist (i.e. the results travel back along the same TCP/IP connections used by the query) but the asynchronous model is advantageous when faced with intermittent connectivity or very long transactions. Since we are partly motivated by the need to support information applications in pervasive computing systems, intermittent connectivity is a key issue. Hence we are also considering asynchronous message passing models.

Future work based on these experiments will include provision of a distributed link service for mobile users, and application to temporal media. We will also address the metadata issues in the hyperstructure; initial work with RDF is in progress. We are also investigating use of the Handle system [19] as another candidate infrastructure. The LDAP model, Whois++ and CIP are all still evolving, and we will continue to track their development.

The experimental work has given an interesting insight into the problem: in separating the hyperstructure and then working with it as distributed information, we create a distributed information management problem which resembles our original problem working with distributed documents! In particular, following forward knowledge is like following a link; forward knowledge maintenance resembles link maintenance. This suggests a question: can we use the infrastructure that we have created in order to manage the structure at a higher level; e.g. use links between index servers? The display in figure 4 is a good example of this – instead of available links we have available link servers. This reflective approach is also a subject of future work.

SUMMARY

We have demonstrated that directory services can be used within link service infrastructures. A simple service such as LDAP provides a basic functionality which, for example, is well matched to text based media, as in our experiment of reproducing Web linking after extracting linkbases, and to simple generic linking. We believe that in some circumstances LDAP may be better suited to linkbase generation than individual link resolution. LDAP also has basic replication facilities that assist in the larger scale distributed context. It is a good candidate for some link services, particularly as it enjoys wide support.

The query routing model of Whois++ (and CIP) appears particularly well suited, and the forward knowledge model is useful in integrating our diverse resources. The challenge is an appropriate definition of forward knowledge for the problem at hand, and how to extend this to multimedia content. We believe that a great attraction of the referral architecture is that processing of referrals is the responsibility of a system component which is separate from the link servers, providing a useful separation of concerns – it means decisions can be made closer to the user.

ACKNOWLEDGMENTS

This work was conducted jointly between the University of Southampton and BT Laboratories, and is partially supported by EPSRC awards LinkMe and HyStream. The 'query by humming' engine and linking media player were developed by Steven Blackburn. The agent framework was developed by Luc Moreau and colleagues in the Multimedia Research Group. Particular thanks are due to Danius Michaelides, Samhaa El-Beltagy and Nick Gibbins, and also to Mark Thompson who is conducting the current research using asynchronous message passing with support from IBM under the University Partnership Programme. The authors are grateful to the reviewers for their useful feedback.

REFERENCES

1. J. Allen and M. Mealling, “The Architecture of the Common Indexing Protocol (CIP)”, RFC 2651, Internet Engineering Task Force, August 1999.

2. K. M. Anderson, “Integrating Open Hypermedia Systems with the World Wide Web”, in Proceedings of the 8th ACM Conference on Hypertext, Southampton, UK, April 1997. pp. 157-156.

3. S. Blackburn and D. DeRoure, “A tool for content based navigation of music”, in Proceedings of ACM Multimedia 1998, Bristol, UK, Sep. 1998, pp. 361-368.

4. L. Carr, D. DeRoure, W. Hall, and G. Hill, “The Distributed Link Service: A tool for Publishers, Authors and Readers”, The Web Journal 1(1) (Proceedings of the Fourth International World Wide Web Conference, Boston, USA, Dec. 1995), O'Reilly and Associates, pp. 647-656.

5. L. Carr, W. Hall and S. Hitchcock, “Link Services or Link Agents?”, in Proceedings of the Ninth ACM Conference on Hypertext and Hypermedia, Pittsburgh, Pennsylvania, USA, Jun. 1998, pp113-122.

6. H. C. Davis, D. E. Millard, and S. Reich, “OHP - communicating between hypermedia aware applications”, in Proceedings of the Workshop 'Towards a New Generation of HTTP', held in conjunction with the 7th International WWW Conference (J. Whitehead, ed.), Irvine, CA, USA, University of California, Irvine Department of Information and Computer Science, Apr. 1998.

7. H. C. Davis “Referential Integrity of Links in Open Hypermedia Systems”, in Proceedings of the 9th ACM Conference on Hypertext and Hypermedia, Pittsburgh, USA, June 1998. pp. 207-216.

8. D. DeRoure, L. Carr, W. Hall, and G. Hill, “A distributed hypermedia link service”, in Third International Workshop on Services in Distributed and Networked Environments, IEEE, Macau, June 1996, pp. 156-161.

9. D. DeRoure, W. Hall, S. Reich, A. Pikrakis, G. Hill, and M. Stairmand, “An open framework for collaborative distributed information management”, in Seventh International World Wide Web Conference (WWW7), Computer Networks and ISDN Systems, vol. 30, Brisbane, Australia, Apr. 1998, pp. 624-625.

10. P. Deutsch, R. Schoultz, P. Faltstrom and C. Weider, “Architecture of the WHOIS++ service”, RFC 1835, Internet Engineering Task Force, August 1995.

11. A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith, “Query by humming - musical information retrieval in an audio database”, in Proceedings of ACM Multimedia 95, San Francisco, California, Nov. 1995.

12. K. Gronbaek, N. O. Bouvin and L. Sloth, “Designing Dexter-based hypermedia services for the World Wide Web”, in Proceedings of the 8th ACM Conference on Hypertext, Southampton, UK, April 1997. pp. 146-156.

13. W. Hall, “Ending the tyranny of the button”, IEEE Multimedia, vol. 1, Spring 1994, pp. 60-68.

14. G. J. Hill, R. J. Wilkins and W. Hall, “Open and reconfigurable hypermedia systems: A filter-based model”, Hypermedia, vol. 5, no. 2, 1993, pp. 103-118.

15. P. H. Lewis, H. C. Davis, S. R. Griffiths, W. Hall, and R. J. Wilkins, “Media-based navigation with generic links”, in Proceedings of the 7th ACM Conference on Hypertext, New York, Mar. 1996, pp. 215-223.

16. Moreau et al, “SoFAR with DIM Agents: An Agent Framework for Distributed Information Management”, Fifth International Conference on The Practical Application of Intelligent Agents and Multi-Agents, Manchester, UK (to appear).

17. S. Newcomb, N. Kipp and V. Newcomb, “The HyTime hypermedia/time-based document structuring language”, Communications of the ACM, 34(11), November 1991, pp 67-83.

18. M.A. Sheldon, A. Duda, R. Weiss and D.K. Gifford, “Discover: a resource discovery system based on content routing”, in Third International World Wide Web Conference, Computer Networks and ISDN Systems, vol.27, no.6, 1995, pp 953-72.

19. S. X Sun, L. Lannom, “Handle System Overview”, Internet Draft draft-sun-handle-system-03 (work in progress), July 1999.

20. C. Weider, J. Fullton and S. Spero, “Architecture of the Whois++ Index Service”, RFC 1913, Internet Engineering Task Force, Feb 1996.

21. W. Yeong, T. Howes and S. Kille, “Lightweight Directory Access Protocol”, RFC 1777, Internet Engineering Task Force, March 1995.

22. J. Xu, Y.Y. Cao, E.P.Lim and W.K. Ng “Database Selection Techniques for Routing Bibliographic Queries”. In Proceedings of the 3rd ACM International Conference on Digital Libraries (DL’98), Pittsburgh, June 1998.