Greenstone: A platform for distributed digital library applications

David Bainbridge, George Buchanan, John McPherson,
Steve Jones, Abdelaziz Mahoui, and Ian H. Witten

Waikato University, New Zealand & Middlesex University, UK
{d.bainbridge, jrm21, s.jones, am14, i.witten}@cs.waikato.ac.nz
g.buchanan@mdx.ac.uk

Abstract. *This paper examines the issues surrounding distributed Digital Library protocols. First, it reviews three prominent digital library protocols: Z39.50, SDLIP, and Dienst, plus Greenstone's own protocol. Then, we summarise the implementation in the Greenstone Digital Libary of a number of different protocols for distributed digital libraries, and describe sample applications of the same: a digital library for children, a translator for Stanford's Simple Digital Library Interoperability Protocol, a Z39.50 client, and a bibliographic search tool. The paper concludes with a comparison of all four protocols, and a brief discussion of the impact of distributed protocols on the Greenstone system.*

Keywords: Distributed protocol, Z39.50, CORBA, graphical user interface support

1 INTRODUCTION

Use of the open source Greenstone Digital Library software is gathering pace. By mid-2000, more than a dozen libraries and universities had arranged access to the software to help meet their digital library needs. Since its release on Source Forge (*www.sourceforge.net*) last October, there have been thousands of downloads per month—but the actual level of use is hard to determine. One project that uses Greenstone extensively is HumanInfo, a Belgian-based NGO that regularly produces humanitarian aid digital library collections on CD-ROM, and distributes upwards of 10,000 copies of each collection within developing countries.

As the user base expands, the collective needs of users expand too. To step outside the mind set indoctrinated by: generate a new Web page in response to a user clicking on a button or hyperlink—the classical form for a digital library, if you will—Greenstone, like other digital library projects [4, 7], provides a protocol for fine-grained interaction with other programs [5]. The protocol is implemented using CORBA [8] and has been extended to support both the SDLIP and Z39.50 protocols.

The purpose of this paper is to demonstrate how distributed protocols promote a variety of distributed digital library applications. We select four applications, built within the Greenstone digital library framework, for discussion: a pilot digital library project for children, a translator for Stanford's Simple Digital Library Interoperability Protocol (SDLIP), a Z39.50 client, and a bibliographic search tool.

The digital library for children utilises a client (which in Greenstone is called a "receptionist") that provides a digital library environment specifically designed for primary school pupils. The receptionist supplies a different user interface from the standard Greenstone look and feel. Moreover, the protocol allows relevant collections served on other Greenstone sites to be seamlessly integrated into the children's work environment. The second example accepts requests from SDLIP clients and translates them into Greenstone protocol calls; data returned from Greenstone is then converted back to the appropriate SDLIP format.

The third illustration, a Z39.50 client, is another example of a receptionist, this time with the "standard" Greenstone look and feel. By incorporating into Greenstone the YAZ software package (*www.indexdata.dk*) the necessary protocol exchange can occur without changing any of the upper layers in the Greenstone code. This new "backend" can access any Z39.50 server; here we demonstrate its use with the Library of Congress's OPAC catalogue. In the final example we move to a Java client to support a rich dynamic graphical environment for user input and display, using the protocol to communicate with a Greenstone server.

The structure of the paper is as follows. We begin by reviewing three existing protocols used by digital library projects, followed by a summary of the Greenstone protocol (see [5] for a more detailed description). Next we describe the four selected applications that demonstrate different aspects of the Greenstone protocol and how it connects with other protocols. We conclude with a discussion that brings out the similarities and differences of all four protocols.

2 EXISTING PROTOCOLS

Three prominent protocols used in the digital library field are the ISO/ANSI/NISO approved Z39.50 protocol, Cornell University's Dienst protocol, and Stanford University's SDLIP.

2.1 Z39.50

Z39.50 specifies a wide-ranging protocol for information retrieval between a client and a database server [2]. Its origins stretch back to 1984, and three progressive versions of the specification were ratified by standards committees in 1988, 1992, and 1995. It is currently administered by the Library of Congress.

Defined as part of the application layer of the Open System Interconnection (OSI) Reference Model, message formats are specified using Abstract Syntax Notation One (ASN.1) and serialised for transmission over the OSI transport layer using Basic Encoding Rules (BER) [3]. The Transmission Control Protocol (TCP) is typically used for this.

Accessing and retrieving heterogeneous data through a protocol in a way that promotes interoperability is a challenging problem. To address the broad spectrum of different domains where it might be used—such as bibliographic data, museum collection information, and geospatial metadata—Z39.50 includes a set of classes, called "registries," that provide each domain with an agreed-upon structure and attributes. Registries cover query syntax, attribute fields, content retrieval formats, and diagnostic messages. For example, content retrieval formats include Simple Unstructured Text Record Syntax (SUTRS) and the various MARC formats.

The Z39.50 protocol is divided into eleven logical sections (called "facilities") that each provide a broad set of services. The protocol is predominantly client driven; that is to say, a client initiates requests, and the server responds. Only in a few places does the server demand information from the client—for example, the Access Control Facility might require the client to authenticate itself before a particular operation is performed. Any server that implements the protocol must retain information about the client's state, and apportion resources so it can respond sensibly to clients using the Initialization Facility. Mandatory search capabilities include fielded Boolean queries, which yields a result set that can be further processed by the Sort and Browse Facilities or cancelled by the Result-set-delete Facility. Results themselves are returned through the Retrieval Facility. At any stage, the response to a request might be an error diagnostic.

Establishing which of the many Z39.50 options, registries, and domain-specific attributes are supported by a particular server is accomplished through the Explain Facility. The Extended Services Facility is a mechanism to access server functionality that persists beyond the duration of a given client-server session—for example, periodic search schedules and updating the database. The client-server session can be canceled immediately by either side through the Termination Facility.

2.2 Dienst

Dienst is one of the longest-running DL projects in the research community: its origins stretch back to 1992 [4]. It has three facets: a conceptual architecture for distributed digital libraries, an open protocol for service communication, and a software system that implements the protocol.

The protocol supports search and retrieval of documents, browsing documents, adding new documents, and user registration. Each of these is an individual service (with version control), borne over HTTP. A digital library collection involves a combination of these services. There are six categories of service: *repository services* store digital documents and associated metadata; *index services* accept queries and returns lists of document identifiers; *query mediator services* dispatch queries to the relevant index servers; *info services* return information about the state of a server; *collection services* provide information on how a set of services interact; and *registry services* store user information.

2.3 Stanford Infobus

Interoperation between distributed objects has been central to Stanford University's digital library project, the "Infobus," where many Infobus objects are in fact proxies to established information sources and services [6]. The original CORBA-based Digital Library Interoperation Protocol (DLIOP) has recently been superseded by the Simple Digital Library Interoperability Protocol (SDLIP), designed in collaboration with other North American research projects [7].

Emphasis has been placed in SDLIP on a design that is scalable, permitting the development of digital library applications that run on hand-held devices such as PalmPilots, in addition to workstation- and mainframe-based systems. There are two transport options: one CORBA-based, the other borne over HTTP; applications can mix these freely.

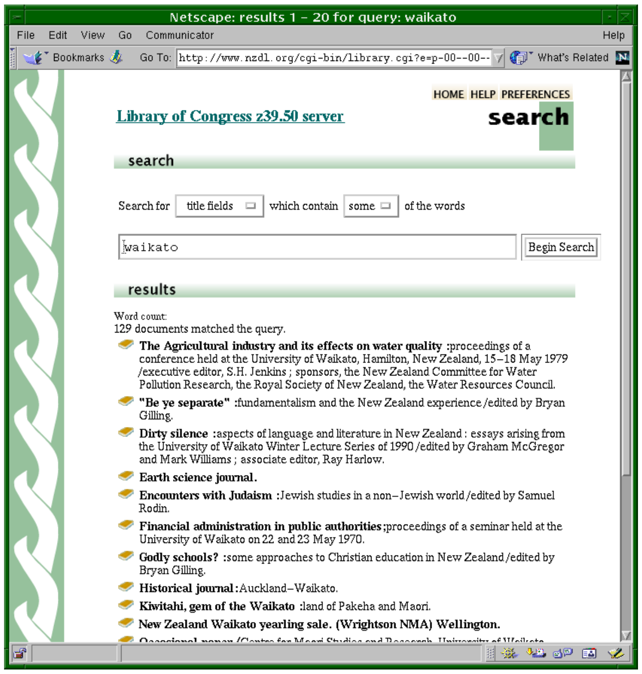
The protocol supports both state-keeping and stateless exchanges on the server side, as well as synchronous and asynchronous interactions between client and server. However, servers need not implement all these parts. It is up to a client to establish—using the protocol—what functionality is supported.

There are four parts (called "interfaces") to the protocol: searching, result access, metadata, and delivery. The *search* interface initiates a search. In a synchronous, stateless exchange the client waits until all results are returned, but in a synchronous, state-keeping one only some of the results need be returned as part of the search—the rest can be accessed through the *result access* interface. A server that supports asynchronous searches must by nature also be state-keeping. When results to an asynchronous query become available, the server uses the *delivery* interface to notify the client. Finally, the *source metadata* interface provides a mechanism for clients to discover the functional capabilities of a server (including version number control), the collections stored there, and the metadata fields present in a particular collection.

3 THE GREENSTONE PROTOCOL

The Greenstone protocol is closely integrated with the digital library architecture, which supports full access to multimedia data: a text query to retrieve a book; a sung fragment of tune to retrieve a music score. Like the previous approaches the protocol adopts a client-server model, although the term "receptionist" is used instead of "client" to emphasise the role that this component plays in the architecture. The protocol is divided into three areas: General, Filtering and Documents. Since last reported [5], we have migrated from a Perl-based remote procedure call mechanism to the more general CORBA framework; the underlying functionality, however, remains essentially the same.

General operations available to a client include obtaining a list of collections offered by a server, testing to see whether a particular collection is running, and obtaining information specific to a collection—such as how many documents it contains, and when it was last updated. The filtering mechanism supports both


Netscape: results 1 – 20 for query: waikato
File Edit View Go Communicator Help
Bookmarks Go To: http://www.nzdl.org/cgi-bin/library.cgi?e=p-00--00-- What's Related
HOME HELP PREFERENCES
search
Begin Search

Library of Congress z39.50 server

search

Search for [title fields] which contain [some] of the words

waikato

results

Word count:
129 documents matched the query.

- **The Agricultural industry and its effects on water quality** :proceedings of a conference held at the University of Waikato, Hamilton, New Zealand, 15–18 May 1979 /executive editor, S.H. Jenkins ; sponsors, the New Zealand Committee for Water Pollution Research, the Royal Society of New Zealand, the Water Resources Council.
- **"Be ye separate"** :fundamentalism and the New Zealand experience /edited by Bryan Gilling.
- **Dirty silence** :aspects of language and literature in New Zealand : essays arising from the University of Waikato Winter Lecture Series of 1990 /edited by Graham McGregor and Mark Williams ; associate editor, Ray Harlow.
- **Earth science journal.**
- **Encounters with Judaism** :Jewish studies in a non–Jewish world /edited by Samuel Rodin.
- **Financial administration in public authorities**;proceedings of a seminar held at the University of Waikato on 22 and 23 May 1970.
- **Godly schools?** :some approaches to Christian education in New Zealand /edited by Bryan Gilling.
- **Historical journal**:Auckland–Waikato.
- **Kiwitahi, gem of the Waikato** :land of Pakeha and Maori.
- **New Zealand Waikato yearling sale. (Wrightson NMA) Wellington.**

Fig. 1. Greenstone interface to the Library of Congress using protocol Z39.50.

searching and browsing. Filters provide an element of dynamic configurability to the protocol through an enumerated list of types that package together specific options. Finally, the Document support provides access to the content of individual documents.

The protocol is stateless, or—to be more accurate—designed for a stateless server. While this simplifies some matters, and meshes well with our digital library architecture, it imposes overheads elsewhere. We return to these in the final section, which compares the various protocols.

4 APPLICATIONS

We now describe four applications that demonstrate the use of distributed digital library protocols in Greenstone.

4.1 Z39.50 receptionist

Figure 1 shows the result of searching the Library of Congress's publicly available catalogue of bibliographic records for titles that include the word "Waikato" (the name of our university and geographical district). The interaction style follows the standard Greenstone interface. After selecting the field to search—from the

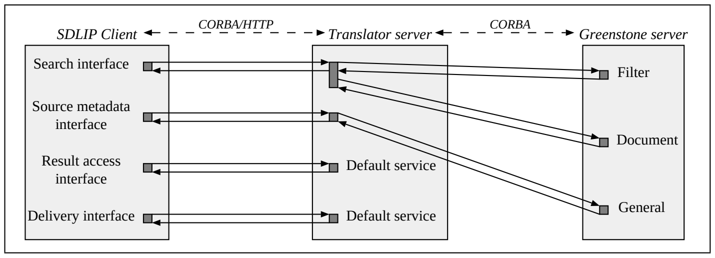


Fig. 2. How SDLIP protocol calls are mapped to the Greenstone protocol.

choices *any fields*, *title*, and *author*—and whether *some* or *all* of the words must be included, a search is initiated by pressing the "Begin Search" button. This loads a new page (shown) that repeats the query settings for the given query at the top and includes matching entries below. Clicking on the "book" icon beside a matching entry produces a new page giving the full catalogue entry.

Due to the complexity of the Z39.50 bibliography registry, "title" metadata covers various different fields. However, for brevity, this system shows only one of these fields for each matching entry. Thus the display may not include the words in the query. For example, the second entry in Figure 1, "Be ye separate," does not specifically mention "Waikato" (although it is clearly related to New Zealand). However, the term does appear in the full citation, as will be revealed by clicking on the book icon.

The interaction between Greenstone and a Z39.50 server is a follows: using the freely available YAZ package, calls to the *General* part of the Greenstone protocol are translated into *Initialization* and *Explain Facility* calls; *Filtering* maps to the *Search*, *Sort* and *Browse* facilities (although presently we only use *Search*); and Greenstone *Document* requests use the *Retrieval* facility.

4.2 SDLIP protocol translator

The protocol translator maps Stanford's SDLIP protocol calls pertaining to stateless synchronous interaction to Greenstone protocol calls. The translator runs as a server in its own right, and Figure 2 shows it acts as an intermediary, accepting SDLIP requests transmitted either through CORBA or HTTP, and passing them on to Greenstone's CORBA-based protocol. Written in Java, the translator server implements the intersection of the Greenstone protocol and SDLIP's *search* and *source metadata* interfaces.

The *search* interface maps to Greenstone's Filter and Document operations, while *source metadata* maps to various calls from the General part of the Greenstone protocol. The remaining interfaces and services, such as the result access interface and the delivery interface, are set up to return trivial, default behavior, because they have no counterpart in a direct mapping to a synchronous stateless service.

Client:

```
weka% java SimpleClient http://kiwi.cs.waikato.ac.nz:8282 "music style"
DOCUMENT: 1
        http://purl.org/metadata/dublin_core#Title
          = "Computer Graphic Aided Music Composition"
DOCUMENT: 2
        http://purl.org/metadata/dublin_core#Title
          = "Schenker s Theory of Tonal Music -- Its Explication ..."
DOCUMENT: 3
        http://purl.org/metadata/dublin_core#Title
          = "Andre Tchaikovsky Meets the Computer: A Concert ..."
DOCUMENT: 4
...
```

Server:

```
kiwi% java SdlipToGsdl http://www.nzdl.org hcibib 8282
GreenstoneCorba Init on www.nzdl.org OK
hcibib OK
hcibib is public? ... yes
Starting DASL/HTTP server transport on port: 8282

[SDLIP/DASL Server Transport] request from: weka.cs.waikato.ac.nz
Query is:
<a:basicsearch xmlns:a="DAV:">
        <a:select>
                <a:allprop/>
        </a:select>
        <a:where>
                <a:contains>music style</a:contains>
        </a:where>
        <a:limit>
                <a:nresults>10</a:nresults>
        </a:limit>
</a:basicsearch>

Query string = music style
Title: Computer Graphic Aided Music Composition
Title: Schenker s Theory of Tonal Music -- Its Explication ...
Title: Andre Tchaikovsky Meets the Computer: A Concert ...
Title: ...
```

Fig. 3. Example use of the SDLIP to Greenstone translator.

Figure 3 shows the result of running the sample command-line driven SDLIP client available for download from the Infobus Web site. At the top we see diagnostic output from the SDLIP client; at the bottom is the diagnostic output from the SDLIP to Greenstone translator. We assume the existence of a Greenstone server (output not shown) whose location is specified when the translator server is started.

When the client program is run, it first connects to the SDLIP server specified on the command line (the translator, in our case) and then calls the *search* interface with the remaining command line arguments stored as the query. The translator server accepts these arguments and sets up a Filter call to emulate the call to SDLIP `search`. If the property list supplied by the SDLIP call specifies document text, a second call to Greenstone is made, this time using the Document part of the protocol, to access the necessary information. The data obtained from these calls is then collated, and returned encoded as XML.

The translator example is intended for demonstration purposes only. A more sophisticated—and ultimately more desirable—approach is to enhance the translator with state keeping capabilities. Just because the Greenstone server does not keep state does not mean that state-based SDLIP interactions cannot be

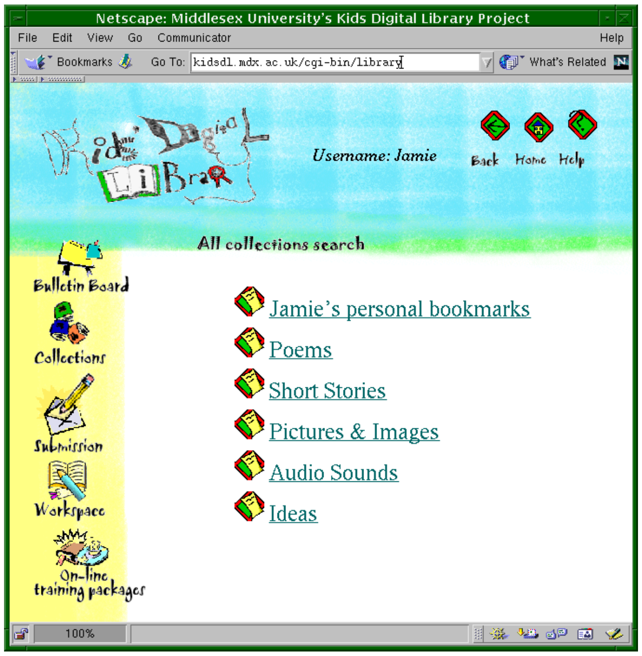


Fig. 4. The Middlesex University's Kids Digital Library environment.

supported. For instance, when query results are returned from the CORBA call to Greenstone, the translator server can store the result locally and assign it a result set identifier. It can use this to support subsequent calls to the result access interface—including query refinement.

4.3 A digital library for children

Figure 4 shows the home page of the Kids Digital Library, part of Middlesex University's digital library project. The vertical column in the center gives the collections available to the user. On the left are support services: a workspace for creative writing; a submission process for completed stories and poems; a bulletin board where selected works are discussed and annotated; and on-line training packages to help users learn about the digital library environment.

The receptionist asks the user to log in before reaching this page; in this case the user is `Jamie`, shown at the top of the page. There is also a special account for the class teacher, with extra functionality provided by the receptionist for updating collections with new stories and so forth. Authentication is not part of the protocol; instead it is built into the receptionist's software architecture.

From the home page a pupil can view the various collections on offer or access the support services mentioned above. *Poems* and *Short Stories* are collections of finished works by the pupils, vetted by the teacher. The collections are searchable

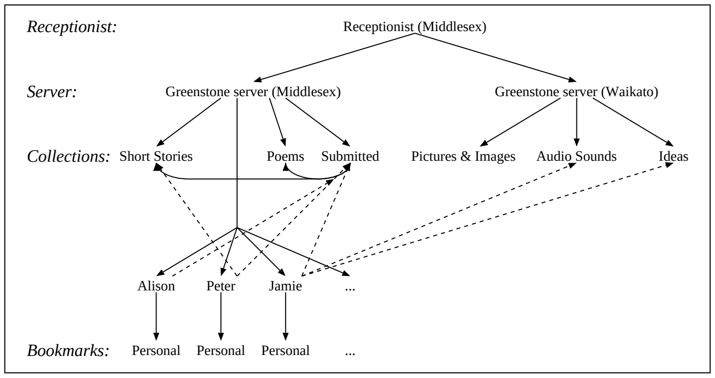


Fig. 5. How the Greenstone protocol is used to provide Middlesex University's Kids Digital Library environment.

by full text, author, and title; browsable by author and title. *Pictures & Images*, *Audio Sounds*, and *Ideas* are collections pulled together from various sources to provide resources and ideas for the pupils; they too have searchable and browsable structures. Finally, the *Personal Bookmarks* collection, which is specific to the particular user, is formed from the user's bookmarks file and includes the downloaded content of each Web page mentioned. The collection is fully indexed, and browsable by title and subject folder.

The idea behind the *Personal Bookmarks* collection is this. Pupils browse the Web using a variety of strategies for finding information pertinent to their work, bookmarking relevant pages. Upon activating "rebuild this collection" through a hyperlink on the Greenstone page for the Personal Bookmarks collection, new pages are downloaded and any existing pages that have changed are updated to form the latest version of their Personal Bookmarks collection.

Figure 5 shows the underlying structure of the Kids Digital Library environment. Unbeknownst to the user, collections are accessed from two servers: one local, the other remote. Small collections that are rebuilt frequently, such as personal bookmarks, short stories and the bulletin board, are served from the main site in Middlesex. The larger collections—intended as a source of inspiration—do not change so rapidly and are served remotely from the Waikato Greenstone site, which has more resources dedicated to supporting digital library collections. For example, in Figure 5 Jamie is accessing the *Sounds* and *Ideas* collections in Waikato as a basis for creative writing, and submitting the composition for the teacher's perusal.

4.4 A Java application

We now turn to an interactive application written in Java. Figure 6 shows a bibliographic search tool that uses a citation's year and matching relevance score to graphically lay out the query result set. The result set is further enhanced

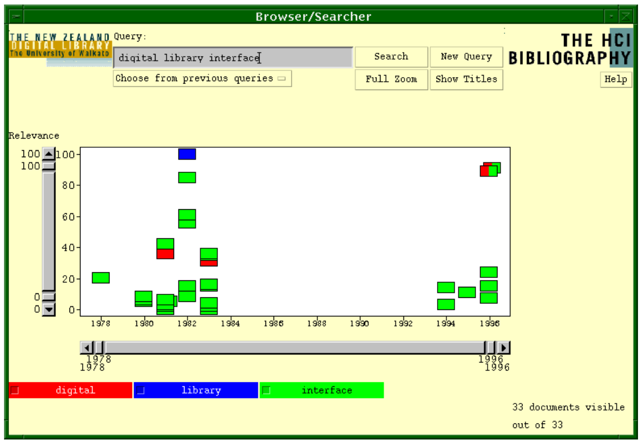


Fig. 6. A bibliographic search tool based on the Greenstone protocol.

through the use of colour: each word in the query is assigned a colour, and matching citations that include that word are displayed using that colour. In the case of a document containing more than one of the query terms, the box is divided into vertical coloured strips. The scroll bars adjacent to the graphics display area allow the user to zoom and pan around the search set; clicking on a particular document box pops up a new window that includes its full citation.

When the bibliographic tool is started, a Greenstone server is specified, along with the particular collection to use. The bibliographic tool requests year metadata, relevance score, and term frequency in addition to document identifier (which is included as standard) when a search is invoked through the protocol's Filter operation. This is sufficient information to generate the graphical display of the result set. When a user clicks on a document box, a new call over the protocol is made to request all metadata for the given document identifier. State information is kept client side.

5 DISCUSSION

We have summarised the main features of four digital library protocols: Z39.50, Dienst, SDLIP, and Greenstone. Demonstrated by four examples, we have also discussed the distributed nature of the Greenstone digital library system, as supported by its protocol. Here we compare and comment on the similarities and differences of the protocols.

All support searching, browsing, and document retrieval. Text searching is relatively well understood—all four protocols support ranked and Boolean queries, with a rich array of options: fielded search, stemming, case matching, and so forth. The main detail for choice is the query syntax used; here Z39.50 and SDLIP are notable in their use of existing and/or emerging standards.

The role of browsing (normally closely associated with metadata) in a digital library is less clear, and support for this varies. Here, Greenstone's communication appears more general than the others, supporting, through its Filter mechanism, hierarchical browsing—it is not clear from the literature if the other protocols support this.

The final core service—document retrieval—is also well supported in the four protocols. Here we see protocols defining models of document structure, and enumerating document formats and types. Arguably, Dienst provides the richest functionality, with its ability to export logical structure in a variety of MIME types.

While not a core requirement of a digital library implementation (as defined, for example, in [1]) all four protocols include functionality to establish the services offered and options supported by a server. This enables more general clients to be written that configure themselves dynamically in response to different situations and, we believe, reflects a level of maturity in DL protocol design.

Other important elements are version control and authentication. Version control is handled externally in Z39.50 by ratified standards. In SDLIP and Dienst it is built into the protocol: a more ambitious aim, with the onus on clients to resolve versioning conflicts. Time will tell how successful this approach is; however, current signs are encouraging. In Greenstone there is no explicit version control. With the protocol tied so closely to the software architecture, this is not as limiting as might first appear. Within the application program interface there is a certain latitude for backwards compatible extensions and the filtering mechanism—the main part of the protocol that is likely to change—has purposely been designed to be extensible. This is backed up by the Filter mechanism that includes calls to list the filter types supported and the options they take.

Although a framework for authentication is part of the Dienst protocol, how it is implemented is a detail left up to the service provider. In Z39.50, authentication is more rigorously defined by its Access Control Facility, which is in stark contrast to Greenstone, that has none. Here authentication is enforced through the receptionist, as seen in the Kids Digital Library example. In SDLIP, there is no mention of authentication [7]. Presumably a client end implementation, similar to Greenstone's, is feasible. Alternatively, a more encompassing security check might be imposed by the transport layer when a client connects to a server.

5.1 Considerations for DL Systems

As we have demonstrated, one digital library system is capable of supporting more than one protocol; at present, in the case of Greenstone, three different protocols are supported. Using the distributed protocols, we have also shown that they can be utilised not only to support *remote services* through one interface, but *novel interfaces* as well, in the form of graphical environments. The Kids' Digital Library shows the high degree of sophistication which can be achieved by utilising both benefits.

Greenstone is internally separated into two components; the "collection server" which provides *services* on one side, and a "receptionist" which accesses the services through an *interface* on the other. This has made it particularly adaptable to supporting both distributed traditional web-based access and more heavyweight and richer graphical environments from one server program. Thus the component architecture already reported in [5] is further validated.

6 CONCLUSIONS

Digital library protocol design is at an interesting stage. While several alternative designs have emerged with varying degrees of complexity, from the elaborate Z39.50 to the simple but tightly prescribed Greenstone protocol, the different designs are not incompatible. As the SDLIP to Dienst proxy [7] and the SDLIP to Greenstone translator presented here both demonstrate, interoperability is alive and well. Furthermore, seemingly irreconcilable differences in protocol design, such as state-keeping and stateless, can often be overcome by appropriate programming support.

References

1. R. M. Akscyn and I. H. Witten. *Report on the First Summit on International Cooperaton on Digital Libraries.* 1998. Available on-line at <ks.com/idla-wp-oct98>.
2. ANSI/NISO. *Information Retrieval (Z39.50 version 3): Application Service Definition and Protocol Specification (ANSI/NISO Z39.50-1995).* NISO Press, Bethesda, MD, 1995. Available on-line at <lcweb.loc.gov/z3950/agency/document.html>.
3. J. Larmouth. *ASN.1 Complete.* Morgan Kaufmann, 1999.
4. C. Logoze and D. Fielding. Defining collections in distributed digital libraries. *D-Lib Magazine*, 4(11), Nov. 1998. Available on-line at <www.dlib.org/dlib/november98/lagoze/11lagoze.html>.
5. R. McNab, I. Witten, and S. Boddie. A distributed digital library architecture incorporating different index styles. In *Proc. IEEE International Forum on Research and Technology Advances in Digital Libraries*, pages 36–45, Santa Barbara, California, 1998. IEEE Computer Society Press.
6. A. Paepcke, M. Baldonado, C.-C. K. Chang, S. Cousins, and H. Garcia-Molina. Using distributed objects to build the stanford digital library infobus. *Computer*, 32(2):80–87, Feb. 1999.
7. A. Paepcke, R. Brandriff, G. Janee, R. Larson, B. Ludaescher, S. Melnik, and S. Raghavan. Search middleware and the simple digital library interoperability protocol. *D-Lib Magazine*, 6(3), Mar. 2000. Available on-line at <www.dlib.org/dlib/march00/paepcke/03paepcke.html>.
8. D. Slama, J. Garbis, and P. Russell. *Enterprise CORBA.* Prentice Hall, 1999.