Aontas: The CaberNet Technical Abstracts Service

Paul Taylor

October 1995

Distributed Systems Group
Department of Computer Science
University of Dublin
Trinity College, Dublin 2, Ireland.
Fax: +353-1-6772204
Tel: +353-1-6081531
Email: pftaylor@dsg.cs.tcd.ie

Abstract

CaberNet is the ESPRIT network of excellence in distributed systems consisting of several European research groups. CaberNet has industrial affiliates who receive regular information about the research activities of CaberNet members. Most CaberNet members produce technical reports which are of interest to the industrial affiliates and other researchers world-wide. This document describes the design and implementation of a unified technical report service.

A contributing CaberNet site just has to make a bibliography available on a local machine. This bibliography is retrieved by a central site and any new or revised records are placed into a database. These records are processed by a professional library cataloguer who ensures that the information is relevant, complete and correct. The processed records may be searched over the world-wide-web and may be used to generate a summary of recent technical reports that is given to the industrial affiliates.

Document Status Draft
Distribution CaberNet
Document # TCD-CS-95-18

Contents

1 Introduction **2**

2 Design **2**

2.1 Bibliographic records 3
2.2 Document database 5
2.3 Document retrieval 6
2.4 Processing submitted records 7
2.5 Installation of processed documents 9
2.6 Report generation 10
2.7 The Dienst system 11
2.8 Other services 11

3 Implementation **12**

3.1 Document database 12
3.2 Perl packages 13
3.3 Document retrieval 18
3.4 CGI programs 18
3.5 Dienst integration 24
3.6 HTTP Server 25

4 Performance **27**

4.1 Parsing CS-TR records 27
4.2 Database operations 28
4.3 CGI programs 29

5 Future directions **29**

A Software requirements **30**

A.1 Server-side requirements 30
A.2 Client-side requirements 30

1 Introduction

CaberNet [3] is the ESPRIT Basic Research Network of Excellence in distributed computing systems architectures. Consisting of several internationally ranked European research groups who are researching the problems and opportunities of large-scale distributed systems, CaberNet's goal is to coordinate and strengthen the research and industrial linkages of these groups. CaberNet's industrial affiliates receive regular information about the research activities of the CaberNet members and CaberNet members benefit from the possibilities of technology transfer to industry.

Most CaberNet members are regularly producing technical reports (TRs) which may be of interest to the industrial affiliates and to other researchers world-wide with an interest in distributed computing systems. This document describes the design and implementation of a prototype technical reports service which provides a unified, remotely accessible, collection of TRs produced by CaberNet members.

Contributing CaberNet sites make available a file containing bibliographic records of recent TRs produced by that site. These records are collected by a central site and new or revised documents are retrieved. The records are then processed by a professional library cataloguer who can ensure that the information about the documents is relevant, complete, and correct. Accepted documents are made available over the World-wide-web (WWW) [18] using Cornell's Dienst [7] search engine. Summary reports of recent TRs are produced periodically to be sent to the industrial affiliates. These reports, which include the abstracts of the TRs, may be used to select interesting or relevant TRs for further investigation.

A distinguishing feature of this TR service over other similar services is the inclusion of the cataloguer. This has the effect of maintaining a high level of quality in the information presented over the WWW and to the industrial affiliates.

The remainder of this document is as follows. Section 2 describes the design of the TR server. A prototype implementation is covered in detail in section 3 and some performance figures for this implementation are presented in section 4. Finally, possible future directions are discussed in section 5.

2 Design

Figure 1 shows the main steps involved in the processing of the technical reports.

A publishing organisation (i.e., a contributing site) creates a file containing bibliographic records in the standard format which is placed on an FTP or HTTP server. This file is retrieved periodically and new or revised records are placed into a database. Each record includes information allowing the retrieval of a PostScript [15] version of the document and this is also retrieved and placed in the database. The records are then processed to ensure that they are relevant and that the information is correct and complete. Processed documents are either accepted or rejected and the placement of the documents in the database changes to reflect these decisions.

Once a number of documents have been processed, the accepted documents are moved

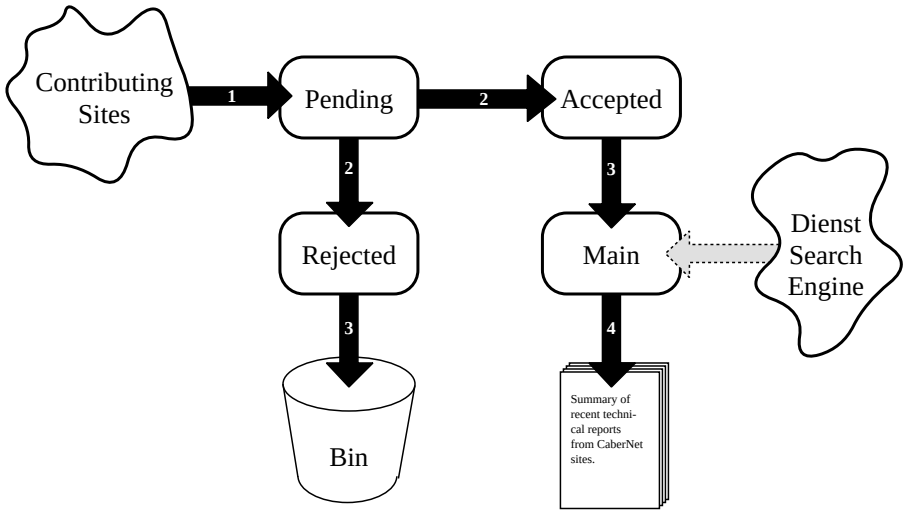

Figure 1: Steps in the processing of technical reports

to the main part of the database and made accessible over the WWW using the search engine and browsing facilities of Dienst. At this point the publishing organisations are given feedback on those documents which were accepted or rejected.

Periodically (e.g., every six months) a summary report of recently installed TRs may be produced. These reports contain information such as the documents' title, authors and abstract and may be sent to CaberNet's industrial affiliates. The reports may be used to select TRs of interest.

The remainder of this section is as follows. Section 2.1 describes the format for bibliographic records and section 2.2 describes the architecture of the document database. The retrieval, processing and installation of documents is described in sections 2.3, 2.4, and 2.5 respectively. Report generation is described in section 2.6 and the integration with Dienst is described in section 2.7. Finally, some other services are described in section 2.8.

2.1 Bibliographic records

The format of bibliographic records used by the TR service is defined by RFC-1807 [12] which superceeds RFC-1357 [6]. Within this document the format is known as the "Computer Science Technical Report" or CS-TR format.

A CS-TR record consists of a set of fields consisting of a *tag* followed by some data, the format of which is determined by the preceeding tag. Tags are identified by a word followed

by two colons ("::").

Each record has a *document identifier* which uniquely identifies the document. The document identifier is placed after the ID and END tags and consists of two parts seperated by two slashes ("//"): a publisher identifier and a report identifier. A publisher identifier uniquely identifies a publishing organisation which may be a research insistitute, a group within a University or a large project. Report identifiers are the identifiers assigned to documents by the publishing organisation.

For a complete description of the format, refer to RFC-1807, however the following is a description of the important parts of CS-TR records relating to this TR service:

1. The text of a record is written using the ISO-8859-1 encoding of the Latin-1 character set [10]. This allows the use of diacritics from most European languages as well as other symbols.

2. Each record should supply at least the following fields in addition to the mandatory fields: TITLE, AUTHOR, CONTACT, DATE, OTHER_ACCESS and ABSTRACT. For withdrawn records (see below), the ABSTRACT may be omitted.

3. A CONTACT field giving an email address should follow each AUTHOR field. At least the primary author of a document should have a CONTACT field.

4. The OTHER_ACCESS field is used to specify the location of a PostScript file for the document as a URL [2]. The file may be compressed as indicated by one of the suffixes: ".Z", ".gz", or ".z".

5. If the WITHDRAW field is present, the document is considered to be withdrawn.

6. In general, the more information supplied in the record, the better.

There follows an example CS-TR record. Note that not all possible fields are represented. It is recommended that contributing sites follow the format of this record.

```
 BIB-VERSION:: CS-TR-v2.1
           ID:: TCD-DSG//TCD-CS-92-22
        ENTRY:: July 18, 1995
ORGANIZATION:: Trinity College Dublin. Distributed Systems Group
        TITLE:: Supporting Object Oriented Languages On The Comandos Platform
         TYPE:: Technical Report
     REVISION:: April 1992; added appedix
       AUTHOR:: Cahill, Vinny
      CONTACT:: Vinny Cahill <vjcahill@dsg.cs.tcd.ie>
       AUTHOR:: Horn, Chris
       AUTHOR:: Starovic, Gradimir
       AUTHOR:: Lea, Rodger
       AUTHOR:: Sousa, Pedro
         DATE:: May 1991
        PAGES:: 12
    COPYRIGHT:: Copyright © TCD 1991. All rights reserved. Permission
                is granted for any academic use of the report.
OTHER_ACCESS:: URL:file://ftp.dsg.cs.tcd.ie/pub/doc/TCD-CS-92-22.ps.gz
    RETRIEVAL:: Send a SAE to The Librarian, Department of Computer Science,
                Trinity College, Dublin 2, Ireland.
    RETRIEVAL:: This report is avaliable via anonymous FTP from
```

```
                ftp.dsg.cs.tcd.ie using the path
                /pub/doc/TCD-CS-92-22.ps.gz.
      KEYWORD:: Comandos
      KEYWORD:: Distributed Systems
      KEYWORD:: Object Orientation
   MONITORING:: Commission of the European Communities, Brussels, Belgium.
      FUNDING:: Esprit Project Comandos
     CONTRACT:: Esprit Project 891/1029
     LANGUAGE:: English
        NOTES:: Also available in Proceedings of the 1991 ESPRIT Conference.

ABSTRACT::

The Comandos project is designing and implementing a platform to
support distributed persistent applications. In particular the
platform supports the object oriented style of programming. An
essential requirement of the Comandos platform is that it must support
applications written in a variety of existing as well as new (object
oriented) programming languages. Moreover, the platform must support
interworking between different languages. Each language may naturally
have its own object model and execution structures implemented by a
language specific runtime system. Rather than forcing each language to
adopt a common object model and execution structures in order to
exploit the distribution and persistence support provided by the
Comandos platform, Comandos provides a generic runtime system on top
of which individual language's specific runtimes may be implemented.
In this paper we show how a language specific runtime for an existing
language such as C++ can be constructed above the Comandos generic
runtime.

END:: TCD-DSG//TCD-CS-92-22
```

2.2 Document database

The document database is used to hold data on TRs in addition to other information that may be of use in various parts of the system. The minimum data for each document is a CS-TR record. Documents are uniquely identified in the database by their CS-TR document identifier.

The database is logically tree structured. All data is accessible given a tuple of the form (*collection*, *docid*, *datafile*). At the highest level the database is split into several *collections*. A collection may be viewed as a holding place for a set of related documents (e.g., the *Reports* collection contains documents for reports of recent TRs) or as representing the state of a document (e.g., the *Pending* collection holds submitted documents that are awaiting processing). At the next level are the holding places for the documents themselves. For each document, there may be several types of data held in *data files*.

Apart from the CS-TR record, most documents also have a PostScript representation of the document. (The current system only supports PostScript.)

There are seven collections in the current system: *Accepted*, *Bibs*, *Deleted*, *Main*, *Pending*, *Rejected* and *Reports*. The purpose of each collection is described below.

Accepted Documents which have successfully passed the first processing phase are placed here after being in the *Pending* collection. This collection serves as a holding place for documents waiting to be installed into the *Main* collection.

Bibs This collection holds CS-TR records containing information about each publishing organisation. This information is used, for example when retrieving remote bibliographies and to send email to the publisher.

Deleted All records that have been deleted are placed here. This collection is special in that all records placed in this collection are no longer accessible. This provides a primitive backup facility.

Main This collection contains fully processed documents which may be searched using Dienst. Documents are placed here from the *Accepted* collection during the installation process.

Pending This collection holds documents that are awaiting processing. They are placed here during the collection process when the remote bibliographies are processed. After processing they are moved to either the *Accepted* or *Rejected* collections.

Rejected Documents which have failed the first processing phase are placed here after being in the *Pending* collection. Later, these documents will be deleted.

Reports When a report of recent TRs is produced the PostScript for the report along with a CS-TR record is placed in this collection.

Other collections may be added if they are needed.

The architecture of the TR database is based on the recommended architecture for Dienst document databases [11] with the addition of collections. The resulting architecture may be compared with Cornell's Digital Library architecture [16].

2.3 Document retrieval

Each publishing organisation is expected to maintain a bibliography of CS-TR records which can be retrieved given a URL. The selection of TRs in the bibliography, the creation of CS-TR records and the frequency with which new records are added is the responsibility of each organisation.

The *Bibs* collection contains a CS-TR record for each publishing organisation. This record gives details about the publishing organisation and must follow the following conventions:

- The document identifier in the ID and END fields must consist of the string "CABERNET//" followed by the publisher identifier (in upper-case) for the CaberNet site.
- The ORGANIZATION field must give a short textual name of the organisation publishing the documents. There should only be one such field.
- The CONTACT fields must specify email addresses which must be given in one of two formats: "Name <id@address>" or "id@address". These contact addresses are used for automatic and manual feedback via email. There may be several such fields.
- The OTHER_ACCESS field must specify a URL giving the location of the CS-TR bibliography. Only *http* and *ftp* protocols are supported. There should only be one such field.

- There is no special meaning for any of the other fields and they may be supplied if desired.

Below is an example of such a record:

```
 BIB-VERSION:: CS-TR-v2.1
           ID:: CABERNET//TCD-DSG
        ENTRY:: June 19, 1995
ORGANIZATION:: Trinity College Dublin.  Distributed Systems Group
        TITLE:: Bibliography of Technical Reports
         TYPE:: CS-TR Bibliography
       AUTHOR:: Taylor, Paul
      CONTACT:: Paul Taylor <pftaylor@dsg.cs.tcd.ie>
         DATE:: June 19, 1995
OTHER_ACCESS:: URL:ftp://ftp.dsg.cs.tcd.ie/pub/doc/cabernet/tr.bib

END:: CABERNET//TCD-DSG
```

Periodically (e.g., once a day, week or month) all bibliographies from contributing sites are processed. This involves iterating though the records in the *Bibs* collection and retrieving each remote bibliography. Each record in each of the bibliographies is processed and is inserted into the *Pending* collection if and only if (a) the record is currently not in *any* collection or (b) the record is in a collection but the REVISION fields of the two records are different. Note that if a document is being withdrawn, it must contain a WITHDRAW field *and* a new REVISION field.

When a record is placed in the *Pending* collection, an attempt is made to retrieve the PostScript version of the document specified by the URL given in the OTHER_ACCESS field of the record. Failure to retrieve the PostScript document is tolerated as it may be retrieved later when processing the records. If a bibliography contains any errors *none* of its records are inserted into the *Pending* collection.

When all bibliographies have been processed an email message is sent to the person responsible for processing the records containing a brief summary of those records that were inserted into the *Pending* collection. This serves as a means of notification since there may be long periods when no new records are inserted. This email is only sent if one or more records were inserted. A email message is also sent to the maintainer of the service containing a full report on the collection process, even if no records were inserted. This serves as a regular reminder that the collection service is running correctly.

No email is sent to the contributing sites at this point since the records have not been processed.

2.4 Processing submitted records

Processing of CS-TR records involves taking each record in the *Pending* collection moving the record to either the *Accepted* or *Rejected* collections. Records may also be modified to add information, to change the ordering of fields, to correct mistakes or to change the format of the record. When editing the record it may be necessary to view the PostScript version of the document for more information.

If a record is correct and relevant, the record is moved to the *Accepted* collection. If, for some reason, the record cannot be processed or is found to be irrelevant, the record is moved to the *Rejected* collection.

Viewing collections

A means is provided to view the records in a particular collection. When viewing a collection, the information displayed for each record consists of the document identifier, the title and authors, the last modified date of the record and the PostScript document and the size of the PostScript document.

Both the order in which the records are displayed (based on the last modified data of the record) and the number of records displayed may be changed.

Searching for a document

It is possible to edit a record without browsing through a collection by specifying a document identifier. This is useful, if the exact location of the record is not known.

Editing Records

When a record is selected (e.g., when viewing a collection or searching for a particular record) the full text of the CS-TR record may be modified.

After modifying the record, the modifications must be committed. Committing will check the record for any errors and if none are present the modified version of the record will be written to the database. If there are any errors, these will be displayed and the record will not be written to the database. This prevents incorrectly formatted records from entering the database.

During editing of a record there are also options to view or print the PostScript of the document or to send email to the contact person for the publishing organisation (see below). If the PostScript for the document is not currently in the database, it is retrieved when the records is selected to be edited. The PostScript document may also be retrieved at any later time. This may be useful if, for example, the original PostScript contained some errors and had to be regenerated.

Moving and deleting records

When all modifications to a record have been made the record may be moved to another collection. From the *Pending* collection the record should be moved to either the *Accepted* or *Rejected* collections. However, records may be moved freely between any collections.

Records may also be deleted. Here, the record is actually moved to the *Deleted* collection but it will appear that the record is no longer in the database. This is a safety feature which allows deleted records to be recovered if the need arises.

Printing documents

While editing a record it is sometimes useful to have a hard copy of the PostScript document on hand. This can be used to make sure that the contents of the CS-TR record match the information in the document. Since this will usually only require the header pages (i.e., those pages that contain the title, authors and abstract), the printing service allows a selection of pages to be printed.

The printing service can also be for other purposes, for example to print out the summaries of recent TRs (see section 2.6).

The PostScript document may also be viewed on-line using a suitable PostScript viewer.

Mailing the publisher

When editing a record it may be necessary to obtain information that is not available from the record or the PostScript document. Thus, it is possible to send email to a contact addresses for the document's publishing organisation. The email addresses are obtained from the CONTACT fields in the publishing organisation's CS-TR record and not from the current record being edited.

Creating new records

New records may be created when viewing a collection by specifying the document identifier of the new record. The record is initially placed in the current collection and consists of a subset of the CS-TR fields. Only the fields BIB-VERSION, ID, ENTRY and END have default values filled in. The document identifier of the new record must not currently exist, otherwise the creation will fail.

2.5 Installation of processed documents

Once a number of records have been processed and placed into either the *Accepted* or *Rejected* collections, the documents can be installed. There are four steps in the installation process:

1. All records in the *Accepted* collection are moved to the *Main* collection. If any of the records being installed contain a WITHDRAW field, the record is processed as normal except that the PostScript for the document is deleted (the record remains in the *Main* collection).

2. All records in the *Rejected* collection are deleted (they are actually moved to the *Deleted* collection).

3. The accepted records are installed into the Dienst (see section 2.7 for details).

4. Email is sent to the contact addresses for publishing organisations informing them which documents have been accepted and rejected. This email is only sent if one or more of a publishing organisation's documents were accepted or rejected.

When a contributing site receives this email, all rejected records should either be removed from the bibliography or corrected in some manner so they will be accepted next time. Failure to remove rejected records will mean that they will be placed into the *Pending* collection the next time the bibliography is processed. Accepted records may be removed or replaced by the processed versions (see section 2.3). Once a document has been accepted the record will not be processed again unless a new revision of the record is created.

The purpose of the *Accepted* collection is to act as a buffer between the *Pending* and *Main* collections. It allows documents to be accepted as a batch rather than one at a time. The advantages of doing this is that contributing sites receive a single email rather than one email per document and the installation of documents into the search engine can be done in one go.

2.6 Report generation

An important aspect of the TR service is the ability to generate summary reports of recent TRs and distribute these reports either by surface mail or electronically to CaberNet's industrial affiliates. These reports contain information from the CS-TR records including the abstract and the document identifier and may be used by the industrial affiliates to select exactly those reports in which they are interested.

The decision of which TRs to include in a report is based entirely on whether a TR has been included in a report previously and only documents in the *Main* collection are considered. This method is simplistic but other ways of selecting reports are ineffective. For example, the DATE field could be used to select reports over a given period, from say May 1995 to October 1995. However since the DATE field is not mandatory and the order in which records are offered by contributing sites cannot be predicted, this method fails to be of any use. Another selection method could be to use the last modified time of CS-TR records. Normally, this would be the time when the documents are installed into the *Main* collection. However, CS-TR records may be edited even while they are in the *Main* collection so this method also fails to be of any use.

When a report is generated, a CS-TR record for the report is created and placed into the *Reports* collection. These records may be viewed and edited in the same way as any other records in the document database. Similarly the PostScript file for a report is included in the *Reports* collection and may be viewed and printed. The document identifiers for reports are of the form "CABERNET//REPORT-n" where n is the report number. Report numbers start at 1 and increase by 1 each time a new report is generated. Each document which has been included in a report has a data file containing the document identifier of the report.

A report contains the following information for each included document: title, list of authors, publishing organisation's name (taken from the information in the *Bibs* collection), abstract and the document identifier.

Reports may also be regenerated by specifying the document identifier of the report. This is provided in case there is a change in the way reports are formatted or in the information

content of a report.

2.7 The Dienst system

Dienst (Distributed Interactive Extensible Network Server for Techreports) [7, 11] is a server and protocol that provides distributed document libraries over the WWW. The Dienst document model provides both multiple formats (e.g., PostScript, OCR, etc.) and document parts (e.g., view the whole document or a particular page). Thumbnail images may be used to browse through a document. Keyword searches (over authors, titles and abstracts) are provided using inverted indices as well as the ability to browse documents in the library. Bibliographic records are in RFC-1357/1807 format.

In this TR service, Dienst is being used for its sophisticated searching and browsing capabilities. The Dienst server runs in "stand-alone" mode which means it does not provide access to other Dienst servers. In the future, full integration with other servers is possible.

Since the document database is designed with Dienst in mind and both systems share a common format for describing bibliographic records, integrating the two systems is straightforward. The Dienst server is configured to use the *Main* collection as its document library.

During the installation of documents from the *Accepted* collection to the *Main* collection, the inverted indices used by Dienst are extended to include the new documents. (The inverted indices are built from information contained in the CS-TR records.) To force the Dienst server to reload the new indices, a signal is sent to the server process.

The other part of the integration is making the searching and browsing facilities available via a HTTP server.

2.8 Other services

There are two other services which are useful to users responsible for maintaining a CS-TR bibliography: CS-TR validation and a means of retrieving processed CS-TR records.

Bibliographic record validation

A contributing site must produce a bibliography of CS-TR records for TRs at that site. When remote bibliographies are collected and placed in the *Pending* collection, the bibliographies must contain correct CS-TR records, otherwise the records are not accepted. The make sure that the contributing sites only supply correctly formated records the validation service should be used.

The CS-TR validation service is given the URL of a remote bibliography. The bibliography is retrieved and all the CS-TR records contained within are parsed. If any errors or warnings are found these are displayed. Error messages are displayed along with the line that caused the error.

Retrieval of CS-TR records

When CS-TR records are submitted by a contributing site, the records are processed, (hopefully) accepted and placed in the *Main* collection. If a revision of a record is created by a site, the revised record is placed into the *Pending* collection and must be processed again. If the previous version of the record required many changes, it can be frustrating to the person processing the record to have to redo all those changes. Thus, a means of retrieving processed records is provided.

The retrieval service is given a document identifier search pattern and displays all records matching this pattern in the *Main* collection. The search method may be either an exact match (to retrieve a single record) or a regular expression (to retrieve several records). The latter method can be used to retrieve all records from a particular publishing organisation by using a regular expression of the form "^*pub-id*//" where *pub-id* is the identifier of the publishing organisation. Records from other collections may also be retrieved.

3 Implementation

This section describes a prototype implementation of the TR service which will be used to evaluate both the TR service as a whole and details of the design and implementation.

The prototype is written in Perl [17] on SunOS 4.1.3 and should be portable to any UNIX platform. The implementation of the database architecture is described in section 3.1. Several packages providing useful routines have been written. The interfaces to these packages are given in section 3.2. The program responsible for the collection and processing of remote bibliographies is described in section 3.3.

Those parts of the service concerned with the processing, installation, validation and retrieval of CS-TR records are implemented as CGI [5] programs accessed using HTML forms [14] over the WWW. The advantage of this approach is that it avoids the necessity of distributing code (to the contributing sites and to the person responsible for processing the records) by distributing only the *interface*. The service is easily made available on a large number of platforms since all that is required is a suitable WWW client. Section 3.4 gives detailed descriptions of each of the CGI programs that make up the TR service.

The integration of this TR service with the Dienst server is explained in section 3.5. The use of CGI programs and Dienst in the TR service requires the use of a HTTP server. The requirements and configuration of the server is discussed in section 3.6.

Note that the implementation requires that the root directory of the service is contained in the environment variable "TRS_ROOT" (including the HTTP server and the Dienst server).

3.1 Document database

The document database is implemented on a UNIX directory hierarchy. The top-level directory contains an index and a subdirectory for each collection.

The index maps document identifiers to collection names and contains an entry for each

document in the database. Each entry is a line of the form "*docid* = *collection*". The index is stored in a file called "INDEX".

Below each collection subdirectory are the document subdirectories. These are implemented as a two-level hierarchies formed by the publisher and report identifier parts of the document identifiers. In each document subdirectory are the data files which have the form "*report-id.suffix*" where *suffix* identifies the type of the data in the file (e.g., CS-TR records have the suffix "bib" and PostScript files have the suffix "ps").

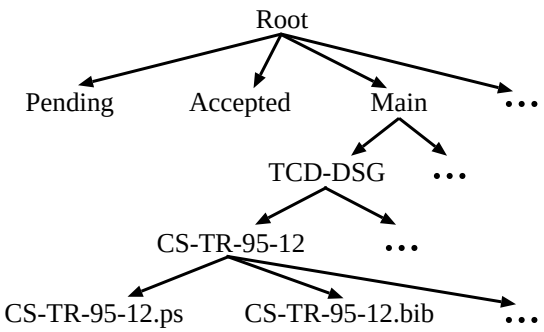

Figure 2: Implementation of the document database

As an example, if the document whose identifier is "TCD-DSG//CS-TR-95-12" is in the *Main* collection, the CS-TR record for this document would be in the file ".../Main/TCD-DSG/CS-TR-95-12/CS-TR-95-12.bib" (see figure 2).

3.2 Perl packages

There are five Perl packages: "bibdb" provides access to the document database, "config" contains configuration settings, "cstr" provides routines to parse CS-TR records, "htmldoc" provides routines to format information in HTML, "mail" provides email facilities, and "web" contains miscellaneous HTML and WWW routines.

Perl package "bibdb"

This package provides routines to browse and modify the document database. All routines are atomic in the sense that they either succeed or fail and leave the database untouched. Protection against concurrency is provided at the level of individual routines and is done using file locking. A routine may be either *read-only* or *modifying*. Multiple "read-only" routines may be active at once, but a modifying routine has exclusive access to the database. Modifying routines are generally less efficient than read-only routines since they must update the database index file. If an errors occurs in any of the routines, an error message is contained in the variable `$error`; otherwise the variable `$error` is empty.

contents(*collection*) – Returns a list of the document identifiers belonging to documents in the collection *collection*. The document identifiers are not returned in any particular order. If an error occurs, an empty list is returned (this is also the return value for an empty collection so the `$error` variable should be consulted). (Read-only.)

delete_record(*docid*, *collection*) – Removes the document identified by *docid* in the collection *collection* from the database. All datafiles belonging to the document are deleted. Returns non-zero if successful; otherwise returns zero. (Modifying.)

doc_path(*docid*, *collection*) – Returns the base path of data files for the document identified by *docid* in the collection *collection*. A particular data file path may be obtained by appending a string of the form "*.suffix*" to the returned path. If an error occurs, returns an empty string. (Read-only.)

locate_collection(*docid*) – Returns the name of the collection in which the document identified by *docid* is located. If an error occurs, returns an empty string. (Read-only.)

move(*docid*, *from*, *to*, *no_index*) – Moves the document identified by *docid* from the collection *from* to the collection *to*. All data files belonging to the document are moved. If *no_index* is non-zero the entry for *docid* is removed from the index file making it appear that the documement has been deleted. Returns non-zero if successful; otherwise returns zero. (Modifying.)

read_record(*docid*, *collection*) – Reads and parses the CS-TR record for the document identified by *docid* in the collection *collection*. This routine uses the "cstr" package (see below) to do the parsing, so the data from the record is available via the "cstr" package variables. If no errors occured returns non-zero; otherwise returns zero. If a zero is returned and `$error` is empty, that means an error occurred during the parsing of the record. (Read-only.)

record_path(*docid*, *collection*) – Returns the path for the CS-TR record for the document identified by *docid* in the collection *collection*. If any errors occurred, returns an empty string. (Read-only.)

write_record(*docid*, *collection*, *record*) – Writes the CS-TR record contained in *record* for the document identified by *docid* in the collection *collection*. The CS-TR record is assumed to be correct. Returns non-zero if successful; otherwise returns zero. (Modifying.)

Perl package "config"

This package contains variables used to configure the system. The most important ones are:

$www_host & **$www_port** – The fully qualified name of the host and port respectively to which the HTTP server is connected.

%NAME – Long and short names for the service and a pseudo publisher identifier for the service.

%COLLECTIONS – The set of database collections each of which is mapped to a short textual description.

%EMAIL – Email addresses for the maintainer and the person responsible for processing the records.

%URL – URLs for various parts of the system.

%PROG – Full pathnames for programs used by the system.

%PRINTERS – The set of printers to which PostScript documents may be printed. Each printer is mapped to a textual description of the printer.

%DEFAULT – Various default values.

Perl package "cstr"

This package provides extensible routines to parse CS-TR records and bibliographies. Information extracted from records is available via variables.

parse(*filename***)** – Reads and parses the contents of the file *filename* which is assumed to contain a set of CS-TR records (see **parse_fh** for more details). Returns non-zero if the file was read and parsed successfully; otherwise returns zero.

parse_fh(*fh***, *filename*)** – Reads and parses the set of CS-TR records from the file-handle *fh* (*filename* is the name of the file associated with the file-handle).

During parsing, the data parts of each field are collected into list and scalar *tag-variables* based on the name of the tag associated with the field. Thus, for a field called *tag*, all its data parts for the current record are in the list `@`*tag* and the last seen data part is in the scalar `$`*tag*. These variables are always in lower-case and any dashes in the tag names are converted to underscores in the variable names. The complete text of the record is collected and formatted in the variable `$record`.

For each field in a CS-TR record there is a mapping to a *tag-routine*. This mapping may be changed so that user defined tag-routines are called instead of the default ones. This is done by modifying the associative array `%TAG` (all field names are in upper-case). The interface to all tag-routines is

tag-routine(*tag*, *data*, *filename*, *lineno*)

where *tag* is the name of the field, *data* is the value of the field, *filename* is the name of the file being parsed, and *lineno* is the line number of the field. To ensure correct behaviour, user tag-routines should call the associated default tag-routine before doing anything else.

There are five special tag-routines that are called during parsing: `<START-BIB>` is associated with the start of the file; `<END-BIB>` is associated with the end of

the file; `<START-RECORD>` is associated with the start of a record; `<END-RECORD>` is associated with the end of a record; `<ERROR>` is associated with error occurrences; and `<WARNING>` is associated with warning occurrences. For the latter two, the *data* parameter contains the error or warning message; for all the other special tag-routines the *data* parameter is empty.

Any errors encountered during parsing are collected in the list `@ERRORS`. All errors are in the form "*filename*: *lineno*: *message*". Similarly, warnings are collected in the list `@WARNINGS`. All warnings are in the form "*filename*: *lineno*: warning: *message*".

Returns non-zero if the file was read and parsed successfully; otherwise returns zero.

split_docid(*docid*) – Returns a list consisting of the two parts of the document identifier *docid*: publisher identifier and report identifier. If any errors occur, a list with two empty strings is returned.

Perl package "htmldoc"

This package provides routines to write HTML for CS-TR records and other purposes. This is useful to achieve a standard look for HTML pages.

action_bar(*docid*, *collection*, *actions*) – Outputs an action bar for the document identified by *docid* in the collection *collection*. There are four possible actions which may be contained in the list *actions*: "Edit" makes a link to edit the CS-TR record for the document; "Mail" makes a link to send email to the contact address for the document's publisher; "Print" makes a link to print the document; and "View" makes a link to down-load the PostScript for the document.

cstr_errors(*file*, *limit*) – Displays parse errors and warnings in CS-TR records and bibliographies. This assumes that one of the parsing routines in the "cstr" package has just been called either directly or indirectly via the routine "bibdb'read_record". *file* is the name of the file containing the CS-TR records and *limit* is the maximum number of errors to display (a value of zero means no limit). The messages are displayed alongside the line that caused the error or warning.

display_doc(*docid*, *collection*) – Displays information on the document identified by *docid* in the collection *collection*. This assumes that one of the parsing routines in the "cstr" package has just been called either directly or indirectly via the routine "bibdb'read_record". The information displayed consists of the document identifier, the document's title, the list of authors, the last modified date of the CS-TR record and PostScript file and the size of the PostScript file (if any).

menu_bar(*topic*) – Displays a menu bar with links to important pages such as the home page, Dienst search page, database top-level page etc. A link is also made to the on-line help information on the topic *topic*.

Perl package "mail"

This package provides a single routine to send email messages.

send(*subject*, *message*, *from*, *recipients*) – Sends an email message to the addresses in the list *recipients*. *subject* is the subject field of the message, *from* is the sender's email address and *message* is the body of the message. Returns non-zero if no errors occurred; otherwise returns zero. Any error output from the mail program (i.e., "sendmail") is collected in the variable `$error`.

Perl package "web"

This package provides miscellaneous WWW routines.

ReadParse – Stephen Brenner's routine to collect CGI paramaters into variables.

compressed(*path*) – Returns non-zero if *path* represents a compressed file; otherwise returns zero. This is done by examining the suffix of *path*.

decode_entities(*text*) – Returns a modified version of *text* with all Latin-1 entities of the form "`&name;`" converted to the ISO-8859-1 encoding for that entity.

decode_html(*text*) – Returns a modified version of *text* in which HTML entity encodings of the form "`&name;`" and "`&#num`" are substituted with the character they represent.

encode_html(*text*) – Returns a modified version of *text* in which characters from the upper end of the Latin-1 character set and the characters `&`, `<` and `>` are substituted with entity encodings of the form "`&#num;`" and "`&name;`" respectively.

extract_url – Returns a URL extracted from the last CS-TR record parsed by one of the "cstr" package parsing routines. Returns the URL if one was found; otherwise returns an empty string.

footer – Outputs a standard HTML footer consisting of the name of the service and an email address to contact for further information.

headers(*title*, *progname*) – Outputs HTTP responce headers and HTML header information. *title* is the title of the page being produced; *progname* is the name of the program (usually `$0`). If the program is a non-parsed-header program (see section 3.4) as indicated by the occurrence of the string "nph-" in *progname*, suitable HTTP response headers are output and buffering is turned off on standard output. This is followed by a MIME document type header and HTML for the "TITLE" and "H1" elements.

retrieve_document(*url*, *path*, *uncompress*) – Retrieves the document specified in the URL given in *url* and places it in the file given by *path*. If *uncompress* is non-zero, the file is also uncompressed. Returns non-zero if the file was successfully retrieved and written; otherwise returns zero. (Note that if the document is empty, the file is removed and zero is returned.)

3.3 Document retrieval

The program "collector" is responsible for retrieving remote bibliographies and placing new and revised records in the *Pending* collection. The program should be run periodically (e.g., once a week), for example as a "cron" script or using the "at" program.

The program works in three phases. In phase one all the CS-TR records in the *Bibs* collection are processed. For each record, the CS-TR bibliography specified in the URL contained in the record is retrieved and written to a data file with the suffix "cstr".

Phase two processes all CS-TR bibliographies that were successfully retrieved. Each bibliography is parsed and information contained in each of the records is collected. This information consists of the URL for the PostScript document, the contents of the REVISION field and the complete text of the record. The bibliography is not processed if it contains any errors.

Once a bibliography has been completely processed, each record is examined to see if it should be placed in the *Pending* collection. The record is inserted if it does not already exist in the database or if it does exist but the REVISION fields of the two records are different. If a record is inserted, an attempt is made to retrieve the documents's PostScript file.

In phase three, email messages are sent to the person responsible for processing the records and to the service maintainer. The former email is only sent if one or more records were placed in the *Pending* collection. The contents of the email contain the list of documents inserted and an indication of whether they were new records or revisions of previous records. The latter email is always sent and consists of a verbose status report.

The program "get_url" is responsible for retrieving remote documents (e.g., CS-TR bibliographies and PostScript files) given a URL. The program is based on Perl programs developed by Oscar Nierstrasz and Lee McLoughlin.

3.4 CGI programs

There are eight CGI programs supporting the TR service: "db" for viewing and editing the document database; "help" for accessing on-line help information; "install" for installing accepted documents into the *Main* collection; "mail" for sending email; "print" for printing all or parts of a document; "report" for generating reports of recent TRs; "request" for retrieving processed CS-TR records; and "validate" for validating CS-TR records and bibliographies.

CGI parameters may be input using both the "GET" and "POST" HTML form methods.

Some of the programs may be accessed by anybody while others require authentication. It is assumed that the HTTP server performs the authentication and passes the identifier of the user to the programs via the environment variable "REMOTE_USER". For these programs, the user "browse" is allowed read-only access.

CGI programs come in two forms: standard and *non-parse-header* (NPH) form [1, 5]. NPH programs are distinguished from standard programs by the occurrence of the string "nph-" in the program name. In a NPH program, it is the program itself and not the server which must output a HTTP response header. Also, the output of NPH programs is

not sent indirectly via the server. This means that NPH programs are potentially faster and the results may be output without being buffered. The latter feature is important as it allows a WWW client to display a page as the data is received thus reassuring the end user that something is happening. Each CGI program described below comes in both forms, implemented using a symbolic link.

CGI program "db"

Synopsis: Provides the ability to view and edit the document database.

Parameters:

collection – The name of a database collection.
docid – A document identifier.
limit – A positive integer (default: 0).
move_to – The name of a database collection.
order – One of three possible values: "None", "Ascending", or "Descending".
record – The text of a CS-TR record.
retrieve – One of two possible values: "on" or "off".

Description: This program works in three different ways depending on the parameters **docid** and **collection**. If neither is supplied, a top-level page is output giving the list of collections and the number of documents in each collection.

If **collection** is supplied and **docid** is not, the output is a view of the specified collection. For each document in the collection, the document's identifier and title are displayed along with the last modified time of the CS-TR record and an indication of whether the PostScript file for the document is available. Each document identifier may be selected to edit that document. The **limit** parameter specifies the maximum number of documents to display (0 means no limit). The **order** parameter can be "None", "Ascending" or "Descending" and refers to the ordering of the documents based on the last modified date of the CS-TR record.

If **db** is supplied and **collection** is not, the collection containing the specified document is located and the CS-TR record for the document is edited (see below).

If both **db** and **collection** are supplied, the CS-TR record for the document is edited. This allows the text of the record to be modified and then committed to the database. Also, the document may be moved to another collection or deleted. If **record** is supplied, it is first checked for errors. If there are any, these are displayed; otherwise the value of **record** is written to the database as the CS-TR record for the document. However if **record** is "new" a new record is created (providing **docid** does not exist in the database). If **move_to** is supplied, the document is moved to the specified collection. Finally, if the PostScript file for the document does not exist or the value of **retrieve** is "on", the PostScript file is retrieved from the URL specified in the CS-TR record.

Note that while editing a record, characters in the upper part of the Latin-1 character set may be entered using the form "&*name*;", where *name* is the Latin-1 entity name.

Output: The output consists of a standard header, a menu bar, the body and a standard footer. For the top-level page, the body consists of a list of collections and a form to edit a CS-TR record given a document identifier. When viewing a collection, the output consists of the list of documents followed by a form to alter the order and limit of the list. There is also a form to specify the document identifier of a record to be created. When editing a record, the output consists of information on the document and a form for editing the CS-TR record. The form allows modification of the text of the record and the specification of a new collection for the record.

Access: This program is protected using basic authentication. Only privileged users may change the state of the database; the user "browse" may only browse documents.

CGI program "help"

Synopsis: Displays help information on a specific topic.

Parameters:

topic – A text string.

Description: This program is used to access on-line help information on a specific topic that is specified in the **topic** parameter. The topic string is converted to lower-case and all non alpha-numeric characters are removed. The resulting string names a help-file located in a special directory. The help-file consists of HTML and may also contain embedded Perl commands bracketed with hash characters ("`#`"). These commands, which must be contained within a single line, are substituted with the result of evaluating the command. If the command is of the form "`&topic('a', 'b')`", the result is a HTML link to the help topic *a* anchored to the text *b*.

Output: The output consists of a standard header, a menu bar, a form for inputting a new help topic, the information on the selected help topic (if any) and a standard footer.

Access: No access restrictions.

CGI program "install"

Synopsis: Installs accepted documents into the *Main* collection.

Parameters:

action – One of two possible values: "Install documents" or empty (default).

Description: This program is used to install documents from the *Accepted* collection into the *Main* collection so they may be searched using the Dienst software. If **action** is empty, a list of publishing organisations is produced. For each publisher, the number of document in the *Accepted* and *Rejected* collections belonging to that publisher is also listed.

If **action** is "Install documents" the following four actions are done:

1. All documents in the *Accepted* collection are moved to the *Main* collection and all documents in the *Rejected* collection are deleted (moved to the *Deleted* collection).
2. An email message is send to the contact address for each publisher for which at least one document was either accepted or rejected.
3. The accepted documents are installed into the Dienst system (see section 3.5 below for details).
4. A HTML document which lists the current publishing organisations is produced.

Output: The output consists of a standard header, a menu bar, the list of publishers, the status of the four actions (if these are done) and a standard footer.

Access: This program is protected using basic authentication. Only privileged users may install documents.

CGI program "mail"

Synopsis: Sends an email message.

Parameters:

action – One of two possible values: "Send Message" or empty.
from – Email address.
publisher – A publishing organisation's identifier.
subject – A short text string.
text – A (possibly long) text string.
to – Comma separated list of email addresses.

Description: This program is used to send email. If **action** is "Send Message" a message is composed using the parameters **subject**, **from** and **text** and is sent to the users specified in **to** (the parameters **to**, **from** and **text** must not be empty). If **publisher** is specified, the message is sent to the contact address specified in the CS-TR record for the named publisher held in the *Bibs* collection. The email message is sent using the "sendmail" program. A footer identifying the source of the message is appended to the body of the message.

Output: The output consists of a standard header, a menu bar, a form for specifying parameters (if **action** is empty), the status of sending the message (if **action** is "Send Message"), and a standard footer.

Access: This program is protected using basic authentication. Only privileged users may send email.

CGI program "print"

Synopsis: Prints all of or part of a PostScript document.

Parameters:

action	– One of three possible values: "Print Document", "Show Print Queue", or "Search for Document".
collection	– The name of a database collection.
docid	– A document identifier.
new_docid	– A document identifier.
pages	– A selection of pages in the format of the "psselect" program.
printer	– A text string.

Description: This program provides a way of printing a document identified by the parameters **docid** and **collection**. If **action** is "Print Document" and a valid document has been selected, the document is printed to the printer specified in **printer**. If **pages** is specified, the program "psselect" is first used to extract the desired selection of pages and the result is printed. If **action** is "Show Print Queue" the print queue for the selected printer is shown. If **action** is "Search for Document" the document identifier identifier by **new_docid** is selected.o

Output: The output consists of a standard header, a menu bar, information on the selected document (if any), a form for submitting requests, the result of a request (if any) and a standard footer.

Access: This program is protected using basic authentication. Only privileged users may submit the action "Print Document". The user "browse" may perform all other actions.

CGI program "report"

Synopsis: Generates a new report or regenerates an existing report of recent TRs.

Parameters:

action – One of three possible values: "Generate new report", "Regenerate report", or "List current reports".
number – An integer value greater than zero.

Description: If the **action** parameter is "Generate new report", a report of recent TRs is produced. A CS-TR record for the new report is created and placed in the *Reports* collection along with the PostScript for the report. The reports selected consist of all documents in the *Main* collection which have not previously been included in an existing report. When a document has been included in a report, the document identifier of the report is stored in a report data file in the documents directory using the suffix "report". If **action** is "Regenerate report" the report identifier by **number** is regenerated. This involved finding all documents whose report file contains the identifier of the report being regenerated.

When a document is included in a report, the document identifier of the report is placed in a data file with the suffix "report". Generation of a new report selects those documents with no "report" data file; regeneration of an existing report selects those documents whose "report" data file contains the document identifier of the report being regenerated.

Reports are generated by creating a LaTeX file which is processed and converted to a PostScript file. The LaTeX document uses the style file "isolatin1.sty" so that Latin-1 characters are rendered correctly. The file "preamble.tex" is prepended to the LaTeX document; this file contains macros for formatting the information from the CS-TR records.

Output: The output consists of a standard header, a menu bar, a form for the three actions, the status of the report generation (if a report was generated), a list of current reports (in descending order of report number) and a standard footer.

Access: This program is protected using basic authentication. Only privileged users may generate or regenerate reports. The user "browse" may view current reports.

CGI program "request"

Synopsis: Extracts CS-TR records from a collection based on a document identifier search pattern.

Parameters:

collection – The name of a database collection (default "Main").
pattern – A text string.
type – One of two possible values: "exact" or "regexp" (default "regexp").

Description: The "request" CGI program searches the database collection specified in the parameter **collection** for CS-TR records whose document identifier matches the search pattern in the parameter **type**. Two types of search patterns are supported. If the parameter **type** is "exact" there must be an exact match; if **type** is "regexp" there match is based on a Perl regular expression. The former search type returns at most one record while the latter may return more than one record.

Output: The output consists of a standard header, a menu bar, a search form, the results of the search (if **pattern** is supplied), and a standard footer.

Access: No access restrictions.

CGI program "validate"

Synopsis: Validates CS-TR records and bibliographies.

Parameters:

action – One of two possible values: "Validate Bibliography" or "Validate Record".
limit – A positive integer (default: 0).
record – Text of a CS-TR record.
url – A URL (protocols *http* and *ftp* only).

Description: This program is used to validate either a single CS-TR record or a bibliography of CS-TR records and to display any errors encountered. If **action** is "Validate Bibliography" the bibliography specified in the URL in the parameter **url** is retrieved and validated. If **action** is "Validate Record" the text specified in **record** is validated. If any errors are found, these are displayed alongside the line that caused the error. A limit on the number of errors displayed may be specified in **limit**.

Output: The output consists of a standard header, a menu bar, the result of parsing the record or bibliography, a form for submitting requests and a standard footer.

Access: No access restrictions.

3.5 Dienst integration

The Dienst software requires local customisation to enable it to access the collection of documents. Routines are provided to:

1. traverse the document collection and call a routine for each document found;

2. given the path for a document's directory, return the document identifier for that document;

3. given a document identifier, return the document directory for that document; and

4. various other small routines dealing with the structure of document identifiers.

Implementation of these routines is simple because the architecture of the document database is based on Dienst recommendations and both systems use the same format for document identifiers.

Dienst also needs configuration for items such as the hostname and port for the HTTP server, the email address of the maintainer and the list of publishing organisations. Since both systems share a common implementation language (i.e., Perl), this configuration is simple since much of this information is already present in the package "config". The list of publishing organisations is created during installation of documents into the *Main* collection and written to a file which is included by the Dienst software.

Dienst allows browsing based on years. However, the information about which year a document was published in comes from the document identifier and not from the CS-TR record. This is done because years are typically encoded into report identifiers. In this service, the format of report identifiers varies between publishing organisation's so this feature of Dienst has been disabled.

During the installation process, apart from creating the list of publishers, the Dienst server needs to be informed that new documents have been installed. Doing this first requires extending the inverted indices used by Dienst and then getting Dienst to reload these indices.

The inverted indices are build using the program "build-inverted-indexes.pl" which can takes a list of document identifiers (which may be contained in a file), reads the CS-TR record for each document and uses this information to extend the indices. To force the Dienst server to reload the indices, the signal SIGUSR2 must be delivered to the server process. Also, sending the signal SIGUSR1 forces the server to reload the configuration code (this is necessary since the list of publishers may have changed). Finding the process identifier of the server processes is done by searching the Dienst log for the last occurrence of the string "Dienst started as process *pid*" where *pid* is the required process identifier.

3.6 HTTP Server

The HTTP server requires integration with the TR service and with the Dienst server. Any HTTP server may be used if it:

1. implements the CGI (the ability to run NPH programs is desirable but not mandatory); and

2. provides basic authentication for "GET" and "POST" methods.

The information here shows the configuration required for version 3.0 of the CERN server [4]. In the following, assume that *root* is the path of the top-level directory of

the service (i.e., the value of the "TRS_ROOT" environment variable). The configuration assumes that the HTTP server is used exclusively for the TR service, though integration with an existing server is possible.

The CERN server configuration file requires the following general settings:

```
ServerRoot root/httpd
HostName   hostname
Port       port
AccessLog  root/logs/httpd-log
ErrorLog   root/logs/httpd-errors
PidFile    root/logs/httpd-pid
```

where *hostname* and *port* are the hostname and port of the HTTP server as specified in the "config" Perl package.

A rule must be added to enable CGI programs to be executed. All CGI programs are in directory "cgi-bin" and those requiring authentication are placed in the subdirectory "auth". (The reason for this separation is that the CERN server requires that all protected CGI programs have an entry in an ACL file and there is no way of specifying an entry with no ACL.) The following three lines specify the "protection setup file" for the protected CGI programs the "Exec" rule for all CGI programs:

```
Protect /TRS/auth/*   root/httpd/trs.auth
Exec    /TRS/auth/*   root/cgi-bin/auth/*
Exec    /TRS/*        root/cgi-bin/*
```

Next are the "Exec" rules for the Dienst server:

```
Exec    /TR/*         root/dienst/Kernel/nph-dienst_stub.pl
Exec    /Server/*     root/dienst/Kernel/nph-dienst_stub.pl
Exec    /Document/*   root/dienst/Kernel/nph-dienst_stub.pl
Exec    /MetaServer/* root/dienst/Kernel/nph-dienst_stub.pl
Exec    /Submit/*     root/dienst/Submit/submit.pl
Exec    /Misc/*       root/dienst/Misc/*
```

Finally, there are two "Pass" rules. The first is used when down-loading PostScript documents and the second is used for ordinary HTML documents (the Dienst HTML documents are included in this root directory using symbolic links.)

```
Pass    /DB/*         root/db/* Pass    /*            root/doc/html/*
```

The protection setup file "trs.auth" is used to specify information used in authenticating access to the protected CGI programs:

```
AuthType      Basic
ServerId      Technical-Report-Service
PasswordFile  root/httpd/passwd
GroupFile     root/httpd/group
GetMask       trs
```

Only members of the group "trs" are allowed access to the protected CGI programs. This group, in addition to the privileged users, contains the user "browse" and this user does not require a password.

The ACL file "cgi-bin/auth/.www_acl" directory ensures that all privileged CGI programs are authenticated for "GET" and "POST" methods. The file contains just one line:

```
* : GET,POST : trs
```

4 Performance

This section presents performance figures for selected parts of the TR service. All figures were obtained using the "gettimeofday" system call on a lightly loaded SPARC Classic workstation running SunOS 4.1.3 and version 5.0 of perl. The programs, libraries and the document database were on a local disk.

Figures are presented for parsing CS-TR records (section 4.1) and database operations (section 4.2).

4.1 Parsing CS-TR records

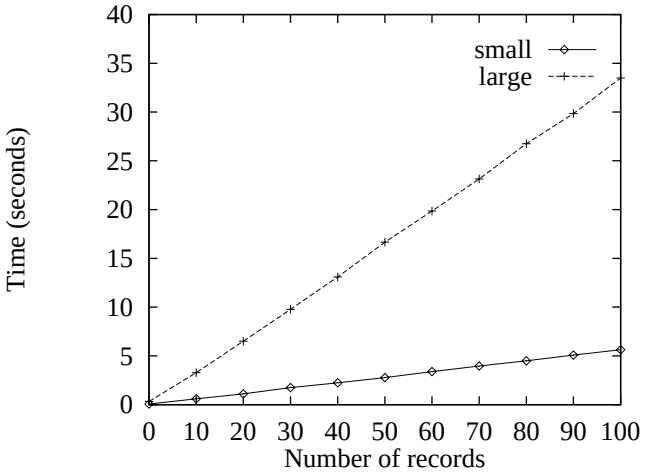

Figure 3: Parsing CS-TR records

Figure 3 shows times for parsing two types of CS-TR records (using the "parse" operation from the "cstr" package):

1. "small" — the smallest possible record consisting of the four mandatory fields ID, BIB-VERSION, ENTRY and END (size: 69 bytes).
2. "large" — a typical record consisting of 29 fields including an abstract (size: 1696 bytes).

The figures represent the time to parse a file consisting of a number of records. Each file was parsed several times and the average elapsed time was used.

The times are high (e.g., approximately 330ms for the large records) due to the extensible nature of the "parse" operation and the way in which information from the record is collected. Both of these are done using the Perl "eval" statement which is costly.

4.2 Database operations

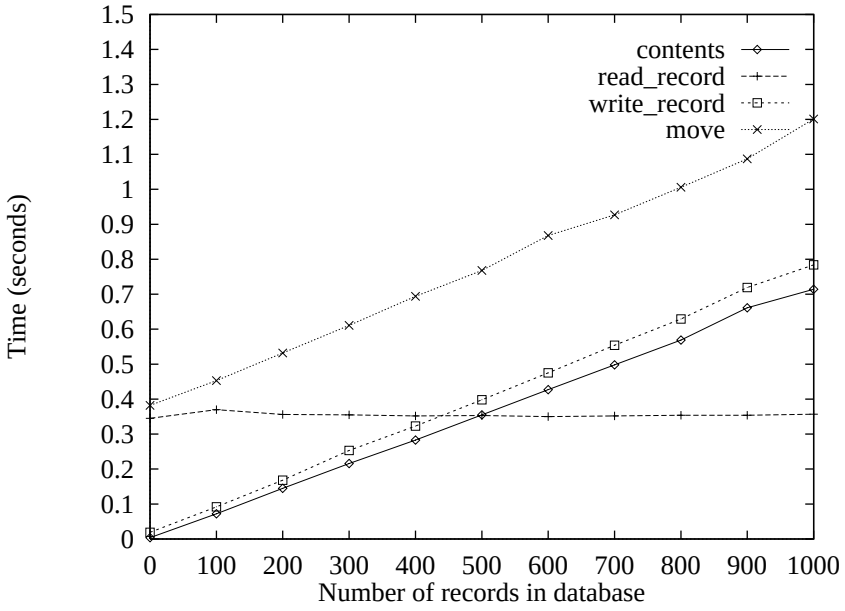

Figure 4: Database operations

The times for four database operations from the package "bibdb" are shown in figure 4. The operations are "contents", "read_record", "write_record", and "move".

For each operation, the figures were obtained by calling the operation several times and taking the average of the elapsed time. The number of records in the database was varied from 0 to 1000.

The times for "contents", "write_record" and "move" increase linearly as the number of records in the database increases. This is due to the index file which is read for all read-only operations and read and written for all modifying operations.

The "read_record" operation is constant because it does not read the index file. The average time for this operation is 355ms which is due to the fact that the "parse" operation is called.

The times for the "move" operation are high because it is a combination of "read_record" and "write_record".

4.3 CGI programs

5 Future directions

There are many ways in which the current TR server could be improved. Some ideas are given below.

Security The "Basic" form of authentication provided by HTTP 3.0 is not secure [1] (i.e., passwords are transmitted from the client to the server as plain-text). Further protection could be provided by only allowing access from certain machines. However, to achieve a more secure implementation would require either a better form of authentication or not using the WWW to access the document database.

More flexible WWW clients The flexibility of the access to the document database is limited by the power of WWW clients. For example, CS-TR records are edited within the "TEXTAREA" HTML element. It would be better if the client knew that CS-TR records were being edited so that formatting would be easier. One possible avenue is to make use of the ability to down-load code from the server to the client as is done in the HotJava [9] WWW client.

Efficiency Some parts of the implementation need to be improved regarding performance (see section 4). Some of these problems could be attributed to the design of a component while others are due the the interpreted nature of Perl. Once the prototype has settled it may be worthwhile rewriting the system in C or C++.

Non-WWW based interface The decision to use a WWW based interface to access the document database needs to be reevaluated after the prototype has been in use for some time. It this decision turns out to be problematic (e.g., due to security concerns or lack of flexibility at the client side) non-WWW based interface will have to be considered. One possible alternative would be to use a graphical interface implemented with Tcl/Tk [13].

Email services Currently documents are submitted by making a bibliography available via a URL. This could be extended by allowing people to email CS-TR records to the server. The email facility could also be used to perform searches, document retrieval, validation, etc.

Search engines There is no reason why other search engines cannot be used instead or alongside the Dienst search engine. Also, installed records could be placed into other distributed bibliography systems such as Refdbms [8]. This would require converting the CS-TR record to *refer* format.

Other formats The current implementation only supports PostScript as the format for documents. This could be extended to support other formats such as DVI, OCR etc.

Fault tolerance The operations on the document database (i.e., those provided by the Perl package "bibdb") could be made to tolerate faults better. This could range from using a full transaction facility to managing faults and concurrent accesses better.

A Software requirements

Below is a list of software requirements for the server side (i.e., the site running the HTTP server) and the client side (i.e., users such as the cataloguer and researchers searching the database).

A.1 Server-side requirements

- A HTTP server. In addition to serving HTML documents, the server must implement the CGI and (NPH programs should also be supported but is not strictly necessary) and provide authentication (e.g., CERN httpd, NCSA httpd).
- The Dienst server and utilities.
- Compression and decompression software (e.g., the standard UNIX utilities "compress" and "uncompress" or the GNU gzip package).
- The LaTeX document processing system and a converter from the DVI format output by LaTeX to PostScript (e.g., "dvips").
- The "psselect" program for extracting a selection of pages from a PostScript document.
- The UNIX utilities "lpr" and "lpq" to print files and show the contents of a print queue.
- The "sendmail" program.
- The "perl" programming language. (Dienst only works with versions 4.x of Perl.)
- Miscellaneous UNIX utilities such as "cp" and "rm".

A.2 Client-side requirements

- A HTML client which should support a limited subset of HTML 3.0 (particularly tables and multiple "submit" buttons).
- A PostScript viewer (e.g., "ghostview").

References

[1] T. Berners-Lee, R. Fielding, and H. Frystyk. Hypertext transfer protocol - HTTP/1.0. [ONLINE], August 1995. Available at http://www.w3.org/hypertext/www/protocols/http1.0/draft-ietf-http-v10-spec-02.ps.

[2] Tim Berners-Lee, Larry Masinter, and Mark McCahill. Uniform resource locators (URL). RFC-1738, December 1994.

[3] CaberNet: Computing architectures for basic European research. [ONLINE], July 1995. Available at http://cabernet.esprit.ec.org/cabernet/.

[4] CERN httpd 3.0 guide. [ONLINE], October 1994. Available at http://www.w3.org/pub/WWW/Daemon/User/.

[5] The common gateway interface specification version 1.1. [ONLINE], September 1995. Avaliable at http://hoohoo.ncsa.uiuc.edu/cgi/interface.html.

[6] Danny Cohen. A format for e-mailing bibliographic records. RFC-1357, July 1992.

[7] James R. Davis and Carl Lagoze. A protocol and server for a distributed digital technical report library. Technical Report TR94-1418, Cornell University, Ithaca, NY 14853, June 1994. Available at http://cs-tr.cs.cornell.edu:80/TR/CORNELLCS:TR94-1418.

[8] Richard A. Golding, Darrell D. E. Long, and John Wilkes. The *refdbms* distributed bibliographic database system. In *Proceedings of Usenix Winter Technical Conference*, San Francisco, CA, January 1994. Usenix Association.

[9] James Gosling and Henry McGilton. *The Java Language Environment: A White Paper*. Sun Microsytems, May 1995.

[10] *ISO 8859-1:1987 Information processing – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1.* International Standards Organisation, 1987.

[11] Carl Lagoze, Erin Shaw, James R. Davis, and Dean B. Krafft. Dienst: Implementation reference manual. Technical Report TR95-1514, Cornell University, Ithaca, NY 14853, May 1995. Available at http://cs-tr.cs.cornell.edu:80/TR/CORNELLCS:TR95-1514.

[12] Rebecca Lasher and Danny Cohen. A format for bibliographic records. RFC-1807, June 1995.

[13] John Ousterhout. *Tcl and the Tk Toolkit*. Addison-Wesley, 1994.

[14] Dave Raggett. HyperText markup language specification version 3.0. [ONLINE], March 1995. Available at http://www.w3.org/hypertext/WWW/MarkUp/html3/CoverPage.html.

[15] Ed Taft and Jeff Walden. *PostScript language reference manual*. Addison-Wesley, Reading, MA, 2^{nd} edition, 1985.

[16] William Turner. The document architecture for the cornell digital library. RFC-1691, August 1994.

[17] Larry Wall. *The Perl Programming Langauge*. O'Reilly & Associates, 1990.

[18] World Wide Web Consortium. The world wide web initiative: The project. [ONLINE], September 1995. Available at http://www.w3.org/hypertext/WWW/TheProject.html.