A Characterization Study of NCSTRL Distributed Searching

Naomi Dushay
Dept. of Computer Science
Cornell University
Ithaca, NY 14853-7501
naomi@cs.cornell.edu

James C. French
Dept. of Computer Science
University of Virginia
Charlottesville, VA 22903
french@cs.virginia.edu

Carl Lagoze
Dept. of Computer Science
Cornell University
Ithaca, NY 14853-7501
lagoze@cs.cornell.edu

Abstract

NCSTRL, the Networked Computer Science Technical Reference Library, is a federated digital library based on the Dienst architecture. One aspect of this architecture is distributed searching, with digital library queries being dispatched from query routers to globally distributed indexers that process them and return results. We studied user data for a two-month period at five query routers in order to characterize some key performance aspects of distributed searching in an operational digital library. This study uncovered the following characteristics. Query processing at NCSTRL servers involves significant time waiting for responses from indexers. Each indexer's availability and response times appear unique to each query router. Different indexers' availability and response times are not similar from the viewpoint of a single query router. Query router waiting time for indexers is larger than indexer processing time, implying that communication time over the network is significant. We close by examining the breakdown of NCSTRL queries: the number of fielded vs. non-fielded queries, and the complexity of these queries.

1 Introduction

Currently, most resource discovery on the World Wide Web employs an architecture where queries are processed at a single indexing site, which may be replicated for localized network access. Well known examples of this architecture are the commercial Web search engines such as Yahoo![1] and Infoseek[2]. While the utility of this architecture has been proven, it is limited by a number of key factors including scalability, lack of domain specificity, and intellectual property restrictions [7]. Consequently, future digital library systems will require mechanisms for effectively distributing queries across multiple search engines.

The Cornell Digital Library Research Group[3] is investigating and designing federated digital library architecture that employs such distributed searching. This architecture has three logical components related to distributed searching:

1) *Indexers* are servers that index metadata for digital library documents and process queries on that metadata, returning the URNs of documents that match the queries.

2) *User Interfaces* are servers that permit user entry of queries, dispatch them to query routers, and then format and display the results returned by the query routers.

3) *Query routers* are servers that dispatch queries (entered at user interfaces) to distributed indexers and combine the results received from those indexers.

These components and a few others form the foundation of the globally distributed Networked Computer Science Technical Reference Library (NCSTRL)[4], which provides the testbed for this paper.

---

[1] http://www.yahoo.com
[2] http://www.infoseek.com
[3] http://www.cs.cornell.edu/NCSTRL/CDLRG/cdlrg.htm

There are many dimensions to the distributed searching problem. Some of these have been extensively examined by the distributed database community, in particular the optimal distribution of indexing information across LANs and controlled WANs [3]. Areas that have received attention within the digital library community include query translation [2], content summarization for query routing [6], and protocols for meta-searching and metadata collection [1, 5].

Our research focuses on the issues related to routing queries based on performance and reliability both of network connections and indexers. The purpose of this study is to further our understanding of operational issues affecting distributed searching, so that more sophisticated searching methods can be developed and used if warranted.

This paper is organized as follows. The first two sections provide background for the remainder of the paper: in section 2, we describe the mechanisms for distributed searching in NCSTRL, and in section 3 we describe our data collection methodology. The next four sections are the heart of the paper, presenting distributed search data. Section 4 analyzes the contribution of indexer response time to total query router response time, section 5 analyzes the performance and reliability of index servers from the perspective of query routers, section 6 analyzes indexer performance independent of network latencies, and section 7 analyzes query complexity. We close with some concluding remarks in section 8.

2 Overview of Distributed Searching in NCSTRL

The Networked Computer Science Technical Reference Library (NCSTRL - pronounced "ancestral") is an operational digital library employing a distributed search architecture. The NCSTRL collection is globally distributed and made available through the Dienst [10]federated digital library architecture [12]. Dienst is an open architecture and protocol [4] for distributed digital libraries that was developed as part of the DARPA-funded Computer Science Technical Reports Project[5]. These characteristics – global distribution, open interface, and production availability – make NCSTRL an ideal testbed for distributed digital library research (indeed, NCSTRL is one of the collections in the DARPA-funded Distributed Integration Testbed[6]).

The Dienst architecture specifies the operational characteristics of semi-autonomous core digital library services. *Repositories* store and provide access to digital documents, the structure of which is described by the Dienst document model. *Indexers* process queries against the descriptive metadata for these documents. *Collection services* provide the mechanisms for federating these and other services into digital libraries. *User interface gateways* provide human access to these federated services. The Dienst architecture also describes an open, extensible protocol for communicating among and with these digital library services.

The NCSTRL collection consists of institutions, or publishing organizations, each of which (at a minimum) provides a repository of digital documents and descriptive metadata [11] for those documents. These institutions are a combination of Ph.D. granting computer science departments, ePrint repositories, electronic journals, and research institutions. The descriptive metadata for the documents in each repository is indexed at one or more indexers. As shown in Figure 1**,** indexers in NCSTRL are *overlapped*, rather than *replicated* or *disjoint*; that is, any two indexers may process metadata from both the same and different repositories. At the time of publication of this paper (December 1998), there were over 100 NCSTRL repositories and approximately 50 NCSTRL indexers worldwide.

[4] http://www.ncstrl.org
[5] http://www.cnri.reston.va.us/cstr.html
[6] http://www.cnri.reston.va.us/integration-testbed.html

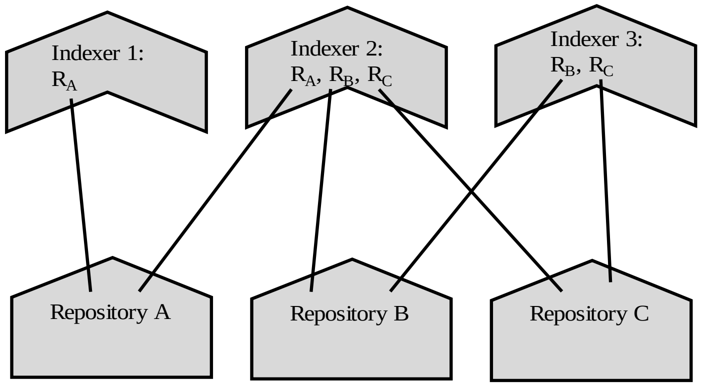

Figure 1 - NCSTRL indexing topology

While not formally defined in Dienst as a separate digital library service, in this paper we use the notion of a *query router* (QR), as described in [8]. The QR is effectively an intermediary between user interfaces (UI) and indexers. In operation, the UI passes user entered queries to the QR, from which it subsequently receives search results. The QR chooses indexers to process the query, sends the search to the indexers, merges the search results, and returns the results to the UI. The UI then formats the results and delivers them back to the user.

The QR determines how to route queries by using data that it derives from its periodic (every 30 minute) communication with the collection service. This collection service data specifies the distribution of indexing information (per institution) among the indexers -- for example, Stanford's documents may be indexed at I_1 and I_2, and Cornell's at I_2 and I_3. This interaction and the resulting routing of queries to specific indexers are illustrated in Figure 2.

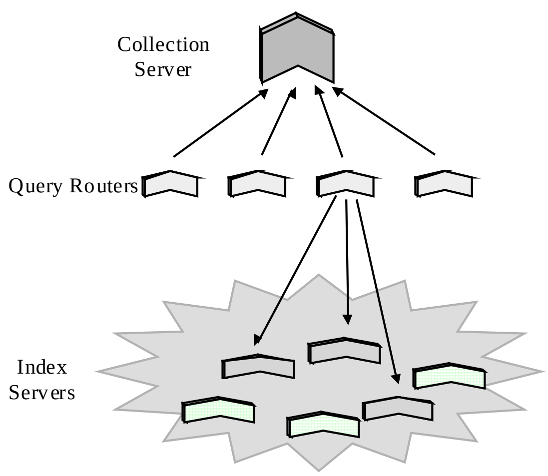

Figure 2 - Query routing based on collection service data

As is well known, global connectivity between Internet nodes varies dramatically. In order to improve search response times to users in a globally distributed digital library, NCSTRL indexers have been grouped into *connectivity regions* [9]. A connectivity region is defined as a set of nodes on the network that among them have low latencies and a resistance to failure relative to nodes outside of the region. Connectivity regions are implemented as part of the Dienst collection service. The collection service is distributed among a set of *regional meta servers* (RMS) – one per connectivity region – and a *master meta* *server* (MMS) from which each RMS derives its region-specific information. Every query router is associated with one connectivity region and thereby derives its *collection view* from the RMS for its respective region. The result is that, unless a failure occurs, each QR restricts its interactions to Dienst indexers within the same connectivity region. This regional configuration is illustrated in Figure 3.

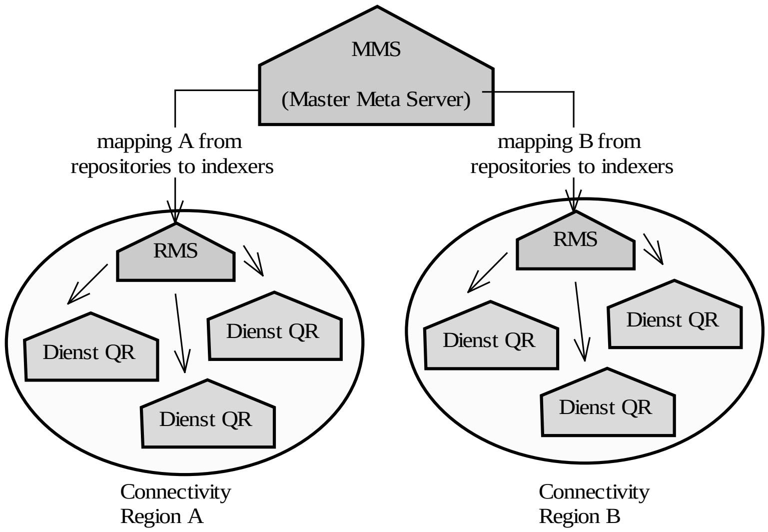

Figure 3 - Connectivity regions

Even with a controlled set of indexers limited to a QR's connectivity region, indexers sometimes fail to respond to the QR within a time deemed acceptable by users. Replication of NCSTRL metadata at multiple Dienst indexers allows for fault tolerance. Data from the collection server specifies an ordering of replicated indexing servers; one is designated as primary, another as secondary, and so on. If one indexer – the *primary* indexer – doesn't respond to a search by a timeout value (determined by a server configuration value), another indexer – the *secondary (*or tertiary, etc.) indexer – can be contacted so complete search results are delivered to the user.

Our experience with NCSTRL has shown that poor performance or failure of an individual indexer can persist over time, for example, a network or server failure usually is not repaired immediately. Rather than continuing to attempt contact with a failing indexer on successive searches, it is preferable that the QR "remember" the failure of the respective indexer and temporarily change the ordering of the indexers for a repository as given by the NCSTRL MMS.

To implement such behavior, Dienst QRs use a simple reliability algorithm. Each time an indexer fails a connection attempt from a QR or fails to respond before the search timeout, the respective QR increments a *failure counter* for that indexer; the failure counter is subsequently cleared when a query is successfully completed. This failure counter is then used as follows. When a QR chooses indexers from the mappings provided by the collection service, it applies a reliability test: *has this indexer failed to respond to a search request before the timeout for x consecutive times*? (where *x* is a configuration variable). If an indexer on the list fails this reliability test, the indexer is then demoted; i.e. a primary indexer becomes secondary and the secondary indexer is promoted to primary. After *z* seconds have passed, where *z* is another configuration variable, the indexer is promoted back to its normal position in the hierarchy.

The actual mechanics of a QR handling a search request is then as follows. When the QR receives a query from a UI, it selects indexers to contact by stepping through the ordered lists of indexers for each publishing institution in the query, looking for the first indexer on the list that isn't demoted and that doesn't fail the reliability test. Once the QR has an indexer for every publishing institution specified in the search query, the search is then sent out to the selected indexers – this is called *phase one* of the search. The QR tracks whether or not each indexer responds within the phase one search timeout, and accumulates search results. If one or more indexers fail to reply, there is another phase of searching, *phase two*, in which the

QR tries to use another indexer (chosen by continuing down the ordered list of indexers for the affected publishing institutions). Phase two searching, if it is necessary, ends when the indexers selected for phase two have all responded or when the phase two search timeout (another Dienst configuration variable) is reached, whichever comes first. The search results are then dispatched to the Dienst UI, which formats the results and delivers them to the user.

This multi-phase searching algorithm is illustrated by the example in Figure 4. In the example, the QR has received a query from the UI that specifies publishing institutions *a*, *b*, *c* and *d*. Based on the mapping from publishing institutions to indexers that the QR received from the collection service, and based on the accumulated failure counters the QR maintains for indexers, the QR chooses three indexers for phase one. Search results for publisher *a* are routed to indexer 1, those for publishers *b* and *d* are routed to indexer 2, and those for publisher *c* are routed to indexer 3. By the phase one search timeout, indexers 2 and 3 have responded, but indexer 1 has not. The QR updates its failure counter for indexer 1 and then initiates a phase two indexer for publisher *a*. Again using data supplied by the collection service, which indicates that indexer 4 should be used as publisher *a*'s secondary indexer, the query is now sent to indexer 4. The QR receives a response from indexer 4 before the phase two search timeout, so the QR now has results for all publishing institutions specified in the query, even though indexer 1 failed to respond before the phase one search timeout. (Even though the notion of tertiary, etc. indexers is included in the architecture, it is not implemented. Thus, if the secondary indexer had failed, the user would have received a notification of an incomplete result set.) Results from indexers 2, 3 and 4 are merged together by the QR and finally are sent to the UI.

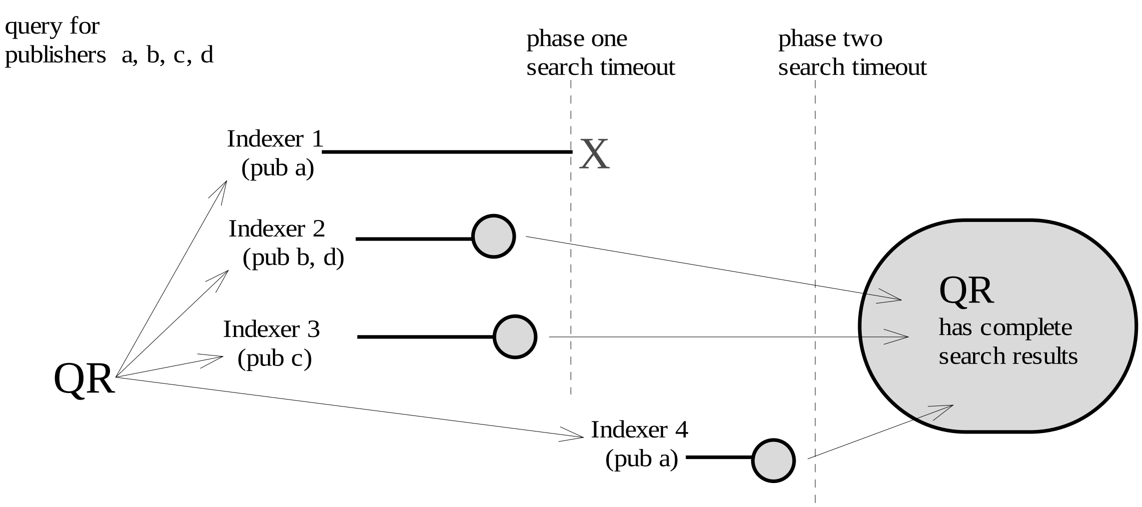

Figure 4 - Phase one and phase two searching

For the purpose of this paper, the QR's tasks can be split into two categories. The first of these can be called QR processing time, which includes 1) choosing indexers to be contacted, 2) sending search requests to indexers, 3) merging results from the indexers, as well as 4) communicating with the UI. The second of these is the time the QR spends waiting for indexers to respond. A key part of this study is to examine whether the time a QR spends waiting for indexers to respond is a significant part of the total time NCSTRL spends at QRs.

3 What data did we examine?

In this study, we are particularly concerned with NCSTRL operational data that illuminates certain key factors pertaining to distributed searching performance. The questions we investigate are:

1) Whether the time the QR waits for indexer responses is a significant part of the total time the QR spends processing a query.

2) If an indexer's view of its own availability and response time is significantly different from a QR's view of the same indexer availability and response time.

3) If the data a QR gathers about indexer performance should be shared among other QRs.

We also examine the ancillary topic of NCSTRL query complexity to inform research we are planning for the future.

While NCSTRL has been and still is a unique testbed for researching issues such as these, it is also a production system. Its features have evolved over time in the form of new versions, but individual sites have not fully coordinated their evolution to these new versions and the level administration of servers at individual sites varies. In addition, we discovered through our research some shortcomings in the logs generated Dienst servers, which are worth noting for future researchers in this area. For example, while we know that NCSTRL has connectivity regions, we cannot be sure that a given QR was in a particular connectivity region during the period of our study, nor can we be certain that the QR was not moved from one connectivity region to another. Moreover, we cannot detect when a QR passed over an indexer because it was demoted or because it did not pass the reliability test; we can only examine data for actual attempts to contact indexers.

3.1 What relevant information is in the Dienst logs?

Recall that an individual Dienst server is usually comprised of a UI (which includes QR functionality), an indexer, and a repository. All of these Dienst services produce Dienst log messages. Dienst logs can contain four types of messages relevant to searching, as shown in Figure 5.

```
     7 Apr 97 15:11:36 GMT TRANSACTION:: 128.84.218.120 GET
     /Dienst/UI/2.0/Query/?boolean=and&author=lagoze&title=&abstract=&all=all&name=.
A
     7 Apr 97 15:20:16 GMT TRANSACTION:: 128.84.218.120 GET
     /Dienst/UI/2.0/QueryNF?keywords=lagoze.

     7 Apr 97 15:11:57 GMT TRANSACTION:: 128.84.254.169 GET
B    /Dienst/Index/2.0/SearchBooleanCallback?host=martin.cs.cornell.edu&port=2000&co
     okie=891961902&timeout=20&boolean=or&title=lagoze&author=lagoze&abstract=lagoze
     &authority=ncstrl.naomipub1&authority=ncstrl.naomipub2.

C    7 Apr 97 15:11:57 GMT STATISTICS:: Search Collection: 1 hits in 0 seconds.

     7 Apr 97 15:12:00 GMT STATISTICS::
D    boolean=or&title=lagoze&author=lagoze&abstract=lagoze cs-tr.cs.cornell.edu 80 6
     8  ncstrl.cc.vt.edu 8080 0 6  elib.stanford.edu 80 0 7  ncstrl.cc.vt.edu 8090 0
     7 www.cs.ucla.edu 8001 0 9  ncstrl.mit.edu 80 0 10 lite.ncstrl.org 3803 0 8
     cs.gmu.edu 8080 0 5 .
```

Figure 5 - Dienst log messages

The meaning of these messages is as follows:

A. The log entries for fielded and non-fielded queries entered by a user through the UI at this Dienst server.

B. The log entry for an indexer request received at this server from a Dienst QR.

C. The log entry recorded when the Dienst indexer finishes processing a search query. These messages are not present in Dienst logs for versions of the Dienst software older than v4-1-0.

D. The search statistics log entry recorded when a QR has finished processing a query. It lists the search criteria followed by the list of indexers called and their response data (or lack thereof) for this query.

The Dienst logs only indicate elapsed time in seconds; this is something Dienst inherits both from Perl (the language in which Dienst servers are implemented) and from the design decision to allow easy porting of Dienst to different varieties of Unix. Note also that for all the elapsed time data in this paper, the whole numbers are floor values -- a response time of 5 means that the indexer responded in 5 or more, but less than 6, seconds. Thorough descriptions of how the log data was processed are in Appendices A, B, C and D.

3.2 What Dienst logs did we examine?

Our data is from the logs of five NCSTRL Dienst servers and spans the period from March 1, 1997 through April 30, 1997. The five sites were:

1. *http://www.ncstrl.org* -- the home page of NCSTRL, located at Cornell University. This is the only Dienst server in our list that does not have all four layers -- it has a UI and a QR, but not an indexer or a repository layer. We refer to this Dienst server as NCSTRL in the remainder of this paper.

2. *http://cs-tr.cs.cornell.edu* -- the Cornell University Department of Computer Science Dienst server. We refer to this Dienst server as CS-TR in the remainder of this paper.

3. *http://lite.ncstrl.org:3803* -- the Dienst server at the University of Virginia. It is also the primary indexer for all NCSTRL participants that only provide a Dienst repository (and no indexer – "lite" sites in NCSTRL parlance). In March and April of 1997, this server was running Dienst-4-0-b0, which put all the log data in one big file, rather than in daily files. The relevant large log file was split apart into daily logs by Perl script that examined timestamps of log messages.
   - The log for March 1, 1997 seems incomplete.
   - The log for April 26, 1997 seems incomplete.
   - The logs for April 27-30, 1997 are empty.

   We refer to this Dienst server as LITE in the remainder of this paper.

4. *http://sunsite.berkeley.edu* -- the University of California at Berkeley Dienst server. We refer to this server as BERKELEY in the remainder of this paper.

5. *http://www.ics.forth.gr/TR* -- the Institute of Computer Science, Foundation for Research and Technology - Hellas (ICS-FORTH) Dienst server.
   - The log for April 4, 1997 was not used, as it was corrupted.
   - The log for April 12, 1997 seems incomplete.
   - There was no log for April 13, 1997.

   We refer to this Dienst server as FORTH in the remainder of this paper.

We reiterate that data in our study reflects non-demoted indexers only: we examined indexer performance from the QR view only when an actual attempt was made to contact the indexer. We also reiterate that our data reflects connectivity regions that existed at the time of the log entries. In theory, the QRs only attempted to contact NCSTRL indexers that had low latencies and a high resistance to failure, relative to other connectivity regions.

4 QR view: how significant is the wait time for indexers?

After the Dienst QR receives a query from the UI, it determines which indexers to contact for search results, sends a search request to those indexers, waits for results, and then merges results and delivers them to the UI. In this section, we examine the significance of wait time for indexers from the QR point of view by comparing how long the QR waited for indexers with the total QR processing time.

Sections 4.1, 4.2, and 4.3 pertain to 32,352 queries answered by the five QRs in our study between March 1, 1997 and April 30, 1997. See Appendix A for a full explanation of how the logs were analyzed to obtain this data.

4.1 QR wait time for indexers

First we examine how long the five QRs in our study wait for search results from indexers. This information is culled from the search statistics messages in the logs (see "D" in Figure 5), which allows us to determine the minimum number of seconds the QR waited for indexers to respond. Recall that for all the data below, as with all the elapsed times reported in this paper, the whole numbers are floor values -- a response time of 8 means the QR waited less than 9 seconds, but 8 or more seconds for indexers to respond. For further details on how the logs were analyzed, see Appendix A.

In Table 1**,** the missing values in the column headed "low", (the smallest amount of time the QR waited for responses from indexers, according to our data) are from search statistics log messages that only have "Server Timed Out" messages for indexer response times.

Table 1 - Minimum number of seconds QR waits for indexer responses

QR	no. obs	high	low	mean	median	mode	std dev
cs-tr	14977	25		8.4	8	6	4.1
ncstrl	10800	99		10.8	10	9	4.5
berkeley	5788	98		11.5	11	12	6.1
lite	637	20	2	9.2	9	3	4.9
forth	150	40	2	13.8	11	3	10.3
all QRs	32352	99		9.8	9	9	4.9

Table 1 indicates that the mean wait time for indexers for all QRs in our study is 9.8 seconds. However, there is a wide range of known minimum wait times, from 2 seconds to 99 seconds, and we can see from the frequency distribution (Table 2) and the standard deviations that the data is not tightly grouped near the mean. In fact, the standard deviation for all QRs is 4.9 seconds, exactly half of the mean wait time of all QRs. Note also that the mean wait time for CS-TR is only 8.4 seconds, while the mean for the FORTH QR wait time is 13.8 seconds. Thus different QRs vary considerably in their response times.

Nevertheless, 29,904 out of 32,352 observations for all QRs, or 92%, have a minimum wait time for indexers of less than 16 seconds.

Table 2 - Number of observations for Table 1

secs	cs-tr	ncstrl	berkeley	lite	forth	all QRs
0	3	16	222	0	0	241
1-5	4169	599	332	192	38	5330
6-10	6394	5105	1976	200	36	13711
11-15	3874	4082	2483	160	23	10622
16-20	310	616	454	85	15	1480
21-25	227	261	198		15	701
26-30		96	79		8	183
31-35		11	20		9	40
36-40		6	6		6	18
41-45		5	1			6
46-50		0	6			6
51-55		0	2			2
56-60		0	2			2
61-65		1	0			1
66-70		0	0			0
71-75		1	1			2
76-80		0	1			1
81-85		0	0			0
86-90		0	0			0
91-95		0	3			3
96-100		1	2			3

4.2 Total time QR takes to process query

Recall that QR processing includes determining which indexers to contact for search results, sending search requests to those indexers, waiting for results, merging results and delivering them to the UI. We now examine the total number of seconds the QR spends processing a query, including the wait for indexer responses. Appendix A explains how we infer this information from the logs.

Table 3 shows that the mean time spent processing queries for all QRs was 22.3 seconds. As with the QR wait times for indexer responses (Section 4.1), there was a wide range of values, from a maximum of 126 seconds to a minimum of 1 second, and the data is not tightly grouped.

Table 3 - QR total time (in seconds)

QR	no. obs	high	low	mean	median	mode	std dev
cs-tr	14977	112	1	17.3	15	20	11.4
ncstrl	10800	126	2	24.9	22	19	13.8
berkeley	5788	123	6	30.0	28	26	14.0
lite	637	96	2	21.2	19	3	14.5
forth	150	98	2	44.1	53.5	66	25.7
all QRs	32352	126	1	22.3	20	20	13.9

Since we were unable to distinguish between one and two phase searches in the logs in most cases (see Appendix A), our data contains both types of searches. Recall that a Dienst QR sends a query to a set of indexers, and if all of those indexers did not respond before the phase one timeout value, then the QR chooses another set of indexers for a second phase of searching. The QR then waits until it has results from all phase two indexers or until the phase two timeout, whichever comes first. The amount of extra time a second phase adds to the QR total time is dependent on three things:

1) The phase one timeout.

2) How long it takes the QR to choose phase two indexers.

3) How phase two ends: how long it takes for all phase two indexers to respond and/or the phase two timeout.

None of this information can be derived from the Dienst logs, so it is unknown for this study. Nevertheless, while it is possible for the maintainer of a Dienst QR to change phase one and phase two timeout values, it is likely that these stayed constant for the duration of our study. It is also probable that the time the QR spent choosing phase two indexers remained fairly constant for our study. It is therefore reasonable to expect a bimodal distribution of total times for an individual QR, with the first peak representing one phase searches and the second peak representing two phase searches.

Table 4 - Number of observations for Table 3

secs	cs-tr	ncstrl	berkeley	lite	forth	all QRs
0	0	0	0	0	0	0
1-5	1881	349	0	116	25	2371
6-10	3294	761	247	64	14	4380
11-15	2396	1628	483	77	1	4585
16-20	2231	2066	875	88	2	5262
21-25	2618	1909	723	52	1	5303
26-30	771	1297	1211	77	1	3357
31-35	338	777	625	61	2	1803
36-40	235	423	355	34	0	1047
41-45	1128	441	258	21	4	1852
46-50	34	553	548	35	14	1184
51-55	15	298	253	3	21	590
56-60	9	132	95	1	10	247
61-65	9	54	30	5	8	106
66-70	6	44	26	1	45	122
71-75	5	17	20	1	0	43
76-80	3	18	9	0	1	31
81-85	2	9	8	0	0	19
86-90	0	5	9	0	0	14
91-95	1	8	0	0	0	9
96-100	0	1	1	1	1	4
101-105	0	3	3			6
106-110	0	1	6			7
111-115	1	2	1			4
116-120		1	1			2
121-125		1	1			2
126-130		2				2

In the case of the CS-TR QR, for example, the frequency distribution table in Table 4 shows one peak at 6-10 seconds with 3,294 observations, and a smaller peak of 1,128 observations at 41-45 seconds. Similarly, for NCSTRL, the first peak has 2,066 observations at 16-20 seconds and the second, smaller peak has 553 observations at 46-50 seconds. The BERKELEY data is not as cleanly bimodal, but if we view the 723 observations at 21-25 seconds as an anomaly, then the 1,211 observations at 26-30 seconds is the first peak and the 548 observations at 46-50 seconds is the second peak**.** Note that the second peak represents the average of all phase two searches. The lower value for the second peak for CS-TR implies that a smaller proportion of queries timed out on that machine.

The FORTH data in Table 4 appears trimodal. The first peak is at 1-5 seconds with 25 observations, the second peak is at 51-55 seconds with 21 observations, and the third peak is at 66-70 seconds with 45 observations. One possible explanation for three peaks instead of two could be that a significant number of phase two searches ended due to phase two timing out: the first peak is for one phase searches, the second peak represents two phase searches that had all phase two indexers respond, and the third peak represents two phases searches that timed out in phase two. The spread of data is also reflected in the standard deviation of 25.7 seconds for FORTH (Table 3) – almost double that of the other QRs.

The LITE data in Table 4 is not clearly unimodal, bimodal or trimodal – while there is a clear peak of 116 observations at 1-5 seconds, the distribution is somewhat erratic out to 46-50 seconds. Perhaps timeout values for phase one changed during the two-month observation period, or perhaps there is some other cause. Despite the spread of data indicated in the frequency distribution, the standard deviation for LITE is 14.5 seconds (Table 3), which is close to the other QRs' standard deviations.

The lowest mean total QR time is for CS-TR at 17.3 seconds, while the highest mean is FORTH's at 44.1 seconds (Table 3**)**. Not only are the means for the individual QRs far apart, but the standard deviations are large as well. Even so, the CS-TR standard deviation of 11.4 seconds implies that FORTH mean is more than two CS-TR standard deviations away. Or, we could say that the CS-TR mean is more than one FORTH standard deviation (25.7 seconds) away from the FORTH mean.

Interestingly, the data for all QRs combined has essentially a unimodal distribution. The interval with the highest frequency was 21-25 seconds, with 5303 observations. The mean time to process queries for all QRs is 22.3 seconds and the standard deviation for all QRs is 13.9 seconds. 25,258 observations out of 32,352, or 78%, are processed in less than 31 seconds.

4.3 QR wait time for indexers as a proportion of total QR time

In section 4.2 we examined the total time the QR spent processing each query, including choosing indexers, sending search requests to the indexers, waiting for search results, merging the results and delivering them to the UI. In section 4.1, we examined the minimum time the QR waited for indexers to respond. By comparing these two measurements, we can explore the significance of QR wait time for indexers by viewing it as a proportion of QR total time. We employ two methods of comparison below.

4.3.1 Individual observation percentages

This method examines the minimum QR wait time as a percentage of the total QR processing time for each QR observation we examined in the logs. For further details on how we analyzed the logs for this information, see Appendix A.

Examining the individual observation's total QR time spent waiting for indexers (see Table 5), we see that mean percentages for individual QRs range from BERKELEY's 41.2% to CS-TR's 61.5%. For all QRs, the mean percentage of QR total time spent waiting for indexers is 54.2%. So on average, the QR spends roughly half of its time waiting for indexers to respond.

Table 5 - % of QR total time due to waiting for indexers

QR	no. obs	high	low	mean	median	mode	std dev
cs-tr	14977	100	0	61.5	66.7	100	26.2
ncstrl	10800	100	0	51.0	50.0	66.7	19.3
berkeley	5788	91.5	0	41.2	41.9	50	16.1
lite	637	100	4.5	58.7	64.3	100	27.7
forth	150	100	5.8	45.0	35.2	75	29.8
all QRs	32352	100	0	54.2	52.2	100	23.8

Table 5 shows a high of 100% for QRs CS-TR, NCSTRL, LITE and FORTH: all of these QRs have instances of the wait time for indexers equaling (within one second) the total QR processing time. Since we know the QR wait time measurement is a minimum, we can be certain that there are instances of QR processing time, other than waiting for indexers, taking less than one second. Again, this illustrates that waiting for indexers can take a significant proportion of total QR time.

Table 6 - Number of observations for Table 5

%	cs-tr	ncstrl	berkeley	lite	forth	all QRs
0	3	16	222	0	0	241
1-5	14	0	0	1	0	15
6-10	156	22	17	18	7	220
11-15	249	133	95	25	22	524
16-20	493	212	166	32	10	913
21-25	763	390	356	38	13	1560
26-30	750	758	513	19	11	2051
31-35	872	1059	625	27	11	2594
36-40	734	1135	709	24	8	2610
41-45	663	906	648	29	5	2251
46-50	959	1002	843	25	7	2836
51-55	481	600	418	19	3	1521
56-60	790	1063	485	35	6	2379
61-65	453	822	366	37	2	1680
66-70	891	920	224	69	2	2106
71-75	1255	750	77	55	14	2151
76-80	1249	405	16	37	4	1711
81-85	1077	148	3	29	5	1262
86-90	973	137	3	25	5	1143
91-95	294	34	2	7	2	339
96-100	1858	288		86	13	2245

We see low percentages of 0% for CS-TR, NCSTRL, and BERKELEY, a low of 4.5% for LITE and a low of 5.8% for FORTH in Table 5**.** These numbers do not necessarily imply that there are instances where QR wait time for indexers was insignificant compared to QR total time; instead, it is likely that these low percentages correspond to instances where our measurement of the QR wait time is too low. Recall that QR wait times are a minimum, as our Dienst log analysis might not detect two phase searches or might assign non-responding indexers a wait time of zero instead of the phase timeout (See Appendix A). Even so, we can see from Table 6 that these low percentages for individual observations comprise a very small proportion of the data. Only 476 observations out of 32,352, or less than two percent, have less than an 11% proportion of QR total time attributed to waiting for indexers to respond.

Actually, given the frequency distribution of percentages shown in Table 6 and given the large standard deviations (ranging from 16.1 for BERKELEY to 19.8 for FORTH, and an overall standard deviation of 23.8), it seems the percentages of QR total time spent waiting for indexers vary widely. However, given that QR wait times are a minimum, the values should be interpreted as minimum percentages: it is likely that some of the percentages are higher than we could infer from the Dienst logs. Since the mean percentage measurement for all individual observations for all QRs is 54.2%, this method of comparison corroborates the notion that the time the QR spends waiting for indexers to respond is significant.

4.3.2 Percentage from means

Another way to explore the proportion of QR total time spent waiting for indexers is to compare the mean number of seconds all QRs spending waiting for indexers to respond to search requests with the mean number of seconds all QRs spend processing requests.

Table 7 - Analysis of QR wait time statistics

QR	mean min. QR wait time for indexers	mean total QR time	% total due to wait time
cs-tr	8.4	17.3	49
ncstrl	10.8	24.9	43
berkeley	11.5	30.0	38
lite	9.2	21.2	43
forth	13.8	44.1	31
all QRs	9.8	22.3	44

The mean of the minimum time a QR spends waiting for indexers to respond is copied into Table 7 from Table 1 in section 4.1, and the mean total QR time in Table 7 is from Table 3 in section 4.2. The percent total due to wait time column is computed by taking the mean minimum QR wait time as a percentage of the mean total QR time. We can see from Table 7 that this method shows that for all QRs combined, 44% of the total QR time is spent waiting for indexers, with values ranging from 31% for the FORTH QR to 49% for the CS-TR QR. Again, this indicates the wait time for indexers to respond to search requests is a significant part of total QR processing time.

5 QR view of indexers

Having shown in section 4 that QR wait time for indexers is a significant part of QR processing time, we now wish to examine the QR view of indexer performance. Recall that after the Dienst QR chooses indexers to process a query, it contacts those indexers and waits for replies. From the QR perspective, there are two interesting results from each attempt to contact an indexer: 1) did the indexer respond to the QR, and 2) how quickly did the indexer respond, given that it responded. We explain how we obtained this information from the Dienst logs in Appendix B.

Note that our study only examines actual attempts QRs made to contact indexers – the observations below represent QRs' attempts to contact indexers that are not demoted and that are chosen from the ordered mapping of publishing institutions to indexers within the QR's connectivity region (see section 2). Thus our study may show indexers as more reliable than they were during the period of study – the NCSTRL heuristic adaptations to network and other conditions have already eliminated many poorly performing indexers from the QR's communication attempts.

The data below shows that indexer speed and reliability appear different to each QR, and it also shows that indexers located very near each other on the Internet do not necessarily appear to have similar performance from the QR point of view. This implies that it is not useful for QRs to share indexer performance information or to aggregate that information across indexers. Even so, we hope our characterization of these measurements will inform future algorithms for QR indexer choice, perhaps by aiding the QRs in making reasonable predictions of individual indexer performance.

5.1 QR view: did the indexer respond?

In this section, we examine whether or not an indexer responded each time that a QR contacted it. For complete details on what was considered a response and what was considered a non-response, see Appendix B. The column heading "ratio" in this section's tables is the number of responses for an indexer divided by the number of attempts the QR made to communicate with the indexer. This ratio is an estimate of the server availability, that is, the probability that the server will respond if it is contacted.

5.1.1 CS-TR QR – indexer response ratios

For the two-month observation period, the CS-TR QR processed 17,240 queries, the largest number of any of the five QRs studied. In processing these queries, CS-TR made 160,392 attempts to communicate with individual indexers. These results are shown in Table 8.

Table 8 - Indexer response ratios for CS-TR QR

indexer	attempts	resp	non-resp	ratio
cs-tr.cs.cornell.edu:80	14868	14748	120	0.99
cs.nyu.edu:80	1869	1135	734	0.61
dri.cornell.edu:80	14172	14119	53	1.00
ei.cs.vt.edu:8090	13653	13462	191	0.99
lite.ncstrl.org:3803	10370	5457	4913	0.53
ncstrl.cc.vt.edu:8080	16957	14925	2032	0.88
ncstrl.cc.vt.edu:8081	1115	440	675	0.39
ncstrl.cc.vt.edu:8090	15824	15004	820	0.95
ncstrl.cs.cornell.edu:8090	13057	11397	1660	0.87
www.cc.gatech.edu:81	3076	2097	979	0.68
www.cs.dartmouth.edu:80	2510	2185	325	0.87
www.cs.uiuc.edu:80	2785	2735	50	0.98
www.cs.umass.edu:80	13993	12474	1519	0.89
www.cs.umd.edu:80	2322	1433	889	0.62
www.cs.utah.edu:80	910	852	58	0.94
www.icase.edu:80	13761	12473	1288	0.91
www.ics.forth.gr:7000	5162	4598	564	0.89
www.tc.cornell.edu:80	13988	13893	95	0.99
all indexers	160392	143427	16965	0.89

Since the CS-TR QR is located at Cornell University, we would expect indexers located at Cornell to perform well, as they are unlikely to be more than one or two network hops away. The "local" indexer at *cs-tr.cs.cornell.edu:80* responded 14,748 out of 14,868 times, giving a ratio of 0.99, while the best ratio, showing as 1.00, was delivered by *dri.cornell.edu*, with only 53 non-responses out of 14,172 attempts. Note that the local indexer on the machine, which should have no network communication problems with itself, is not the most reliable indexer. Regarding other indexers located at Cornell, *www.tc.cornell.edu:80* also had an excellent ratio of 0.99, responding 13,893 out of 13,988 times, however, *ncstrl.cs.cornell.edu:8090* only responded 11,397 out of 13,057 times, for a ratio of 0.87. Thus, within the *cornell.edu* domain, there are indexer response ratios ranging from 0.87 to 1.00.

Indexer *ncstrl.cc.vt.edu:8081* had the worst ratio of 0.39, but interestingly, the indexer on the same machine at port 8080 had a ratio of 0.88. It's possible that the processes for port 8090 were not as stable as those for port 8080, or perhaps the attempts to communicate with port 8090 occurred at times of high load on the Dienst server so it wasn't able to respond quickly enough. The indexer at *ei.cs.vt.edu*, presumably located near to *ncstrl.cc.vt.edu* on the network, had a ratio of 0.99, with only 191 non-responses out of 13,653 attempts. Again, this illustrates that network location is not a sole predictor of indexer performance. Overall, out of the 160,392 attempts the CS-TR QR made to contact indexers, 143,427 were responses, giving an overall ratio of 0.89 responses / attempts. This finding is consistent with availability measures taken by others. [14] measured the availability of web servers and found it to be approximately 95%. The availability of Internet servers has also been measured between 85% and 91% by [13].

5.1.2 NCSTRL QR – indexer response ratios

The NCSTRL QR processed the second largest number of queries: 12,745 of them. Two of the processed queries were dropped because we were unable to parse the search statistics messages. For the remaining 12,743 processed queries, NCSTRL made 113,517 indexer communication attempts. We dropped 6 of these attempts from our results due to response times over two minutes (there was clearly some server problem) – refer to Appendix B for more information on how the logs were processed. We therefore examined 113,511 attempts NCSTRL made to communicate with indexers. These results are shown in Table 9.

Table 9 - Indexer response ratios for NCSTRL QR

indexer	attempts	resp	non-resp	ratio
cs-tr.cs.cornell.edu:80	11632	11514	118	0.99
cs.nyu.edu:80	646	360	286	0.56
dri.cornell.edu:80	11635	11520	115	0.99
ei.cs.vt.edu:8090	11186	11020	166	0.99
lite.ncstrl.org:3803	7190	2628	4562	0.37
ncstrl.cc.vt.edu:8080	17275	14848	2427	0.86
ncstrl.cc.vt.edu:8081	1804	0	1804	0.00
ncstrl.cc.vt.edu:8090	11700	11005	695	0.94
ncstrl.cs.cornell.edu:8090	2851	1712	1139	0.60
ncstrl.cs.cornell.edu:8099	589	589	0	1.00
ncstrl.cs.uiuc.edu:80	194	194	0	1.00
www.cc.gatech.edu:81	1890	1598	292	0.85
www.cs.dartmouth.edu:80	1449	1165	284	0.80
www.cs.uiuc.edu:80	223	0	223	0.00
www.cs.umass.edu:80	10445	9255	1190	0.89
www.cs.umd.edu:80	332	66	266	0.20
www.cs.utah.edu:80	223	0	223	0.00
www.icase.edu:80	10912	10685	227	0.98
www.tc.cornell.edu:80	11335	11247	88	0.99
all indexers	113511	99406	14105	0.88

The NCSTRL Dienst server is also located at Cornell University, and the NCSTRL QR contacted two "local" indexers. Indexer *ncstrl.cs.cornell.edu:8099* responded 589 out of 589 times, for a perfect ratio of 1.00. However, the indexer on the same machine at port 8090 only responded 1712 out of 2851 times for a ratio of 0.60. Again, we can only guess that the discrepancies in these ratios are caused by the indexer calls occurring at different times or by varying reliability of processes associated with different ports.

As with the QR at CS-TR, NCSTRL attempted to contact *ncstrl.cc.vt.edu:8080* the most times, and again that indexer did not have the highest ratio of responses to attempts – only 0.86. The uniqueness of the QR view is apparent when we realize that NCSTRL tried to contact *ncstrl.cc.vt.edu* 17,275 times to CS-TR's 16,957 attempts, while CS-TR had far more queries to process: 17,240 for CS-TR vs. 12,745 for NCSTRL. In other words, these two closely located QRs sent different proportions of their queries to the same indexers due to differences in the demotion/promotion performance data at each QR and possibly due to differences in the mapping of publishing institutions to indexers.

Another indication that indexer performance appears differently to different QRs, despite proximity on the network, comes up when we examine the three indexers that never responded to the NCSTRL QR. *ncstrl.cc.vt.edu:8081* did not respond to NCSTRL once in 1,804 attempts, but it responded 440 times out of 1,115 attempts to CS-TR. Both *www.cs.uiuc.edu:80* and *www.cs.utah.edu:80* responded 0 out of 223 attempts for NCSTRL, but for CS-TR they responded 2,735 out of 2,785 attempts and 852 out of 910 attempts, respectively.

5.1.3 BERKELEY QR – indexer response ratios

We were able to parse all but one of the 7,085 queries processed by the BERKELEY QR. Out of 69,492 attempts to communicate with indexers for the remaining 7,084 queries, we examined 69,483 of them, dropping 9 due to responses times over two minutes. These results are shown in Table 10. See Appendix B for more information on how the logs were processed.

Table 10 - Indexer response ratios for BERKELEY QR

indexer	attempts	resp	non-resp	ratio
cs-tr.cs.cornell.edu:80	6068	5793	275	0.95
cs.nyu.edu:80	966	480	486	0.50
dri.cornell.edu:80	6044	5745	299	0.95
ei.cs.vt.edu:8090	5943	5631	312	0.95
lite.ncstrl.org:3803	4232	1370	2862	0.32
ncstrl.cc.vt.edu:8080	8904	7667	1237	0.86
ncstrl.cc.vt.edu:8081	724	0	724	0.00
ncstrl.cc.vt.edu:8090	6117	5581	536	0.91
ncstrl.cs.cornell.edu:8090	3686	1048	2638	0.28
ncstrl.cs.cornell.edu:8099	1547	1547	0	1.00
ncstrl.cs.uiuc.edu:80	458	458	0	1.00
sunsite.berkeley.edu:80	228	16	212	0.07
www.cc.gatech.edu:81	1311	788	523	0.60
www.cs.dartmouth.edu:80	1190	813	377	0.68
www.cs.uiuc.edu:80	608	0	608	0.00
www.cs.umass.edu:80	5900	4763	1137	0.81
www.cs.umd.edu:80	872	247	625	0.28
www.cs.utah.edu:80	608	0	608	0.00
www.icase.edu:80	5896	5482	414	0.93
www.ics.forth.gr:7000	2275	1797	478	0.79
www.tc.cornell.edu:80	5906	5626	280	0.95
all indexers	69483	54852	14631	0.79

The "local" indexer for the BERKELEY QR, *sunsite.berkeley.edu:80*, responded only 16 out of 228 times for a ratio of 0.07. This is extremely poor performance for an indexer that should be on the same LAN. However, we do not distinguish between poor network connectivity from the QR to the indexer and problems with the indexer, so perhaps this indicates that the BERKELEY Dienst server had corrupted indexes or some other problem affecting indexing during our two month observation period. We note that no other QR in our study contacted *sunsite.berkeley.edu:80* as an indexer, which implies that poor or erratic performance for BERKELEY was noted by the NCSTRL maintenance team and they removed the BERKELEY indexer from the NCSTRL MMS's ordered list of indexers for the BERKELEY repository.

As with the NCSTRL QR, *ncstrl.cc.vt.edu:8081*, *www.cs.uiuc.edu* and *www.cs.utah.edu* did not respond to any communication attempts by the BERKELEY QR. And similarly with NCSTRL and CS-TR, the indexers at ports 8080 and 8090 of *ncstrl.cc.vt.edu* had response ratios over 0.85 (0.86 and 0.91 respectively), once again disproving the notion that network location and machine availability are sufficient to explain availability of indexers.

5.1.4 LITE QR – indexer response ratios

The LITE QR processed 796 queries with 6,363 attempts to communicate with indexers. Results are shown in Table 11.

The "local" indexer for the LITE QR, *lite.ncstrl.org:3803*, responded to 396 out of 636 communication attempts for a ratio of 0.62. This is poor, but the CS-TR, NCSTRL and BERKELEY QRs all had worse ratios for *lite.ncstrl.org:3803* as an indexer: 0.53, 0.37, and 0.32 respectively. It is not clear whether the differences in the ratios are caused by network connectivity issues or by indexer availability at the specific times the indexer was contacted. In fact, we cannot distinguish the cause from the Dienst logs.

Table 11 - Indexer response ratios for LITE QR

indexer	attempts	resp	non-resp	ratio
cs-tr.cs.cornell.edu:80	574	567	7	0.99
cs.nyu.edu:80	97	73	24	0.75
dri.cornell.edu:80	544	539	5	0.99
ei.cs.vt.edu:8090	549	545	4	0.99
lite.ncstrl.org:3803	636	396	240	0.62
ncstrl.cc.vt.edu:8080	456	427	29	0.94
ncstrl.cc.vt.edu:8081	3	0	3	0.00
ncstrl.cc.vt.edu:8090	590	585	5	0.99
ncstrl.cs.cornell.edu:8090	575	557	18	0.97
www.cc.gatech.edu:81	200	102	98	0.51
www.cs.dartmouth.edu:80	98	85	13	0.87
www.cs.uiuc.edu:80	91	88	3	0.97
www.cs.umass.edu:80	546	516	30	0.95
www.cs.umd.edu:80	98	83	15	0.85
www.cs.utah.edu:80	75	69	6	0.92
www.icase.edu:80	539	521	18	0.97
www.ics.forth.gr:7000	145	114	31	0.79
www.tc.cornell.edu:80	547	544	3	0.99
all indexers	6363	5811	552	0.91

We note that the indexer the LITE QR tried to contact the most was the "local" indexer, while the CS-TR, BERKELEY and NCSTRL QRs all attempted to contact *ncstrl.cc.vt.edu:8080* most often. The LITE QR also contacted port 8090 on *ncstrl.cc.vt.edu* more than other ports, unlike CS-TR, BERKELEY and NCSTRL which all contacted port *8080* the most. As with BERKELEY and NCSTRL, however, the LITE QR did not get any responses from indexer *ncstrl.cc.vt.edu:8081*, while ports 8080 and 8090 had ratios above 0.85 (0.94 and 0.99, respectively).

5.1.5 FORTH QR – indexer response ratios

The FORTH QR only answered 203 queries, all of which we could parse, and furthermore only had 744 attempts to communicate with indexers, one of which had a response time greater than two minutes, leaving 743 attempts for us to examine. None of the indexers called by the FORTH QR are used by any of the other QRs because FORTH is in a different connectivity region (see section 2). We are unwilling to extrapolate much from the FORTH data due to the small number of observations, but we present the information in Table 12 for completeness.

Table 12 - Indexer response ratios for FORTH QR

indexer	attempts	resp	non-resp	ratio
alexandra.di.uoa.gr:80	127	44	83	0.35
carolan.dsg.cs.tcd.ie:1996	120	0	120	0.00
dienst.csi.forth.gr:80	200	197	3	0.99
medoc.informatik.uni-hamburg.de:80	4	4	0	1.00
www-ir.inf.ethz.ch:80	4	3	1	0.75
www.ics.forth.gr:7000	141	137	4	0.97
www.sztaki.hu:80	4	3	1	0.75
www.sztaki.hu:8001	143	91	52	0.64
all indexers	743	479	264	0.64

5.1.6 QR view of indexer response ratios -- conclusions

We will now look at response ratios for indexers from the five QRs in our study in order to draw some conclusions. In Table 13, "ncstrl" is marked with an asterisk because the NCSTRL QR is at port 80 on the machine at *ncstrl.cs.cornell.edu*, while the NCSTRL indexer is at port 8090 at *ncstrl.cs.cornell.edu*.

Table 13 - Indexer response ratios between studied servers

	cs-tr		ncstrl*		berkeley		lite		forth	
	QR cs-tr sees indexer	indexer cs-tr as seen by QR	QR ncstrl* sees indexer	indexer ncstrl* as seen by QR	QR berkeley sees indexer	indexer berkeley as seen by QR	QR lite sees indexer	indexer lite as seen by QR	QR forth sees indexer	indexer forth as seen by QR
cs-tr	0.99	0.99	0.99	0.87	0.95	-	0.99	0.53	-	0.89
ncstrl*	0.87	0.99	0.60	0.60	0.28	-	0.97	0.37	-	-
berkeley	-	0.95	-	0.28	0.07	0.07	-	0.32	-	0.79
lite	0.53	0.99	0.37	0.97	0.32	-	0.62	0.62	-	0.79
forth	0.89	-	-	-	0.79	-	0.79	-	0.97	0.97

The data columns in Table 13 are shown in five groupings of two columns each – one grouping for each site in our study. The meaning of the two columns within each grouping is as follows, using the first grouping as an illustrative example. The first data column, headed "QR cs-tr sees indexer," shows the indexer response ratios as perceived by the CS-TR QR for each of the indexers in our study. For example, the ratio of responses to failures for the NCSTRL indexer as perceived by the CS-TR QR was 0.87. These figures are taken from Table 8 in section 5.1.1. The second data column, headed "indexer cs-tr as seen by QR," shows the indexer response ratios for the CS-TR indexer, as perceived by the QRs indicated. These figures are taken from the tables in sections 5.1.1, 5.1.2, 5.1.3, 5.1.4, and 5.1.5. The double column groupings for other sites in our study are derived similarly. Missing values in Table 13 occur when a QR didn't call a particular indexer, or (similarly) when an indexer wasn't called by a particular QR. For example, the BERKELEY indexer is only called by the BERKELEY QR, implying the MMS's ordered list of indexers for the BERKELEY repository did not include the BERKELEY indexer for any but the BERKELEY QR (see section 5.1.3).

Examining the data for CS-TR, we see that from the perspective of the CS-TR, NCSTRL and LITE QRs the CS-TR indexer has a response ratio of 0.99, while from the perspective of the BERKELEY QR the CS-TR indexer has a response ratio of 0.95 (FORTH didn't poll the CS-TR indexer during the study period). So the CS-TR indexer has a response ratio ranging from 0.95 to 0.99 for the QRs in our study. Meanwhile, the response ratios of the other indexers in our study, from the perspective of the CS-TR QR, range from 0.53 for LITE to 0.89 for FORTH, and a ratio of 0.99 for the indexer on the same Dienst server. So the CS-TR indexer appears more available to other Dienst QRs than the indexers at those sites appear to it. The implication is that the CS-TR QR was well maintained during the course of our study – the machine was adequate to the processing tasks demanded of it, and support staff ensured the machine was up as much as possible.

The LITE QR polls the same four indexers in our study as the CS-TR QR, with response ratios ranging from 0.62 for the LITE indexer to 0.99 for the CS-TR indexer. The LITE indexer is also contacted by the same QRs as the CS-TR indexer, and the response ratios range from 0.37 for the NCSTRL QR to 0.62 for the LITE QR. So in contrast to CS-TR, indexers appear more reliable to the LITE QR than the LITE indexer appears to other Dienst QRs. The implication is that during the course of our study, the LITE machine was either poorly maintained or not up to the processing demanded of it. In fact, the LITE machine was underpowered during the course of this study -- the machine had large indexes, a high demand, and only one processor. It was in fact upgraded in early summer 1997, soon after the study period.

Another indication that the reliability of the LITE indexer was poor is in the asymmetry between CS-TR and LITE: the QR at CS-TR has a response ratio of 0.53 for the LITE indexer, while the LITE QR has a ratio of 0.99 for the CS-TR indexer. We know that the QR view of an indexer depends both on network connectivity and indexer reliability. In this case, it would appear that the network connectivity between LITE and CS-TR is good, otherwise the LITE QR could not have a ratio of 0.99 for the CS-TR indexer. But the low ratio for the CS-TR QR view of the LITE indexer implies poor reliability of the LITE indexer, given that the network connection is good. This is born out by the low ratios reported for the LITE indexer by all QRs.

Table 13 also shows that the BERKELEY and LITE QRs view their own indexers as the least reliable of any of the servers in our study, buttressing the notion that the QR view of indexer reliability depends on more than good network connectivity and availability of the server. Server load, index sizes, search characteristics – many factors could contribute to an indexer not responding to a QR before the search timeout; unfortunately, we cannot diagnose the cause from the Dienst logs.

We can also conclude that indexer response ratios do not appear the same to different QRs. For example, the NCSTRL indexer has a response ratio of 0.28 for the BERKELEY QR, and a response ratio of 0.97 for the LITE QR. This discrepancy might be caused by communication attempts being made at different rates and at different times, or it might be caused by Internet topology. Regardless, the QR's view of indexer availability is unique. This corroborates the notion that the QR view of the availability of an indexer, rather than an absolute view of the availability of an indexer (that doesn't take network connectivity into account), will be the most useful for predicting indexer performance. The high spread of response ratios in the data also underlines the need for an adaptive and flexible fault tolerant distributed searching algorithm that incorporates indexer redundancy.

5.2 QR view: indexer response times, given that the indexer responded

This section analyzes response time data in seconds each time an indexer responded to a QR. For complete details on how response time data was gathered from the logs, see Appendix B.

Recall that the data below is the floor value of response time in seconds -- a response time of 8 means the indexer responded in less than 9 seconds, but in 8 or more seconds.

5.2.1 CS-TR QR – indexer response times, given that the indexer responded

The CS-TR QR had 143,427 responses from indexers; these results are shown in Table 14.

The "local" indexer on the same Dienst server as the CS-TR QR had a mean response time of 1.9 seconds., but that was not the fastest indexer. The smallest mean response time was 0.3 seconds from *www.cs.utah.edu:80* for 852 responses. It's difficult to believe that Internet connectivity is slower within a LAN than across the Internet, so we conclude that the Dienst server at CS-TR takes longer to search its own indexes on average than some other Dienst indexers, perhaps because the index files at CS-TR are large. We are reminded that Internet topology is not the only factor in the QR view of indexer response times.

Note that the indexer at *ncstrl.cc.vt.edu:8090* had a mean response time of 4.2 seconds to the CS-TR QR, while the indexer on port *8081* at *ncstrl.cc.vt.edu* had a mean response time of 13.3 seconds. The Dienst logs do not provide enough information to explain this. Perhaps port *8081* was contacted at times of higher load on the server, perhaps the machine at *ncstrl.cc.vt.edu* is tuned in a way that affects processes for port *8090* and *8080* more favorably than those for port *8081*, or perhaps the index files for the different ports are of different sizes. Whatever the cause, it highlights the notion that different indexers appear different to a QR, even if they are in the same Internet location.

In addition to the wide range of actual response times, from 25 seconds to 0 seconds, there was a wide range of mean response times, from 0.3 seconds for indexer *www.cs.utah.edu* to 13.3 seconds for *ncstrl.cc.bt.edu:8081*. The spread of means shows it would be ineffectual to combine performance data from different indexers in order to describe the QR view of indexer behavior.

Table 14 - Indexer response times for CS-TR QR

indexer	no. obs	high	low	mean	median	mode	std dev
cs-tr.cs.cornell.edu:80	14748	15	0	1.9	2	2	0.9
cs.nyu.edu:80	1135	14	0	2.8	1	0	3.4
dri.cornell.edu:80	14119	22	0	3.2	3	3	1.4
ei.cs.vt.edu:8090	13462	14	0	1.8	2	2	1.3
lite.ncstrl.org:3803	5457	18	0	7.2	7	7	3.3
ncstrl.cc.vt.edu:8080	14925	25	0	5.1	4	3	3.9
ncstrl.cc.vt.edu:8081	440	25	0	13.3	13	13	7.2
ncstrl.cc.vt.edu:8090	15004	18	0	4.2	3	3	2.5
ncstrl.cs.cornell.edu:8090	11397	15	0	4.4	4	2	3.1
www.cc.gatech.edu:81	2097	15	0	2.7	2	2	2.3
www.cs.dartmouth.edu:80	2185	15	0	7.0	6	6	2.5
www.cs.uiuc.edu:80	2735	14	1	3.0	3	3	1.0
www.cs.umass.edu:80	12474	15	0	4.7	4	2	3.1
www.cs.umd.edu:80	1433	15	1	10.9	11	12	2.8
www.cs.utah.edu:80	852	12	0	0.3	0	0	1.1
www.icase.edu:80	12473	14	0	2.0	2	1	2.1
www.ics.forth.gr:7000	4598	24	0	5.5	5	4	2.7
www.tc.cornell.edu:80	13893	14	0	1.7	2	2	1.0
all indexers	143427	25	0	3.6			

5.2.2 NCSTRL QR – indexer response times, given that the indexer responded

The NCSTRL QR had 99,406 responses from indexers; these results are shown in Table 15.

Table 15 - Indexer response times for NCSTRL QR

indexer	no. obs	high	low	mean	median	mode	std dev
cs-tr.cs.cornell.edu:80	11514	26	1	5.1	5	5	1.4
cs.nyu.edu:80	360	17	0	5.1	4	4	3.6
dri.cornell.edu:80	11520	26	2	7.0	7	7	1.9
ei.cs.vt.edu:8090	11020	26	0	5.0	5	5	1.6
lite.ncstrl.org:3803	2628	65	0	9.8	10	9	3.2
ncstrl.cc.vt.edu:8080	14848	59	1	8.6	8	7	4.5
ncstrl.cc.vt.edu:8090	11005	99	2	8.2	8	7	3.2
ncstrl.cs.cornell.edu:8090	1712	22	2	7.9	7	6	2.9
ncstrl.cs.cornell.edu:8099	589	21	3	8.7	9	6	3.0
ncstrl.cs.uiuc.edu:80	194	15	6	8.6	8	7	1.8
www.cc.gatech.edu:81	1598	36	1	6.9	6	5	2.6
www.cs.dartmouth.edu:80	1165	19	5	10.7	10	9	2.0
www.cs.umass.edu:80	9255	65	0	6.8	6	5	3.1
www.cs.umd.edu:80	66	21	7	13.0	13	14	2.0
www.icase.edu:80	10685	26	0	4.7	5	4	1.8
www.tc.cornell.edu:80	11247	26	0	5.0	5	5	1.5
all indexers	99406	99	0	6.6			

The mean response times for indexers to the NCSTRL QR ranged from 4.7 seconds for *www.icase.edu:80* to 13.0 seconds for *www.cs.umd.edu:80*. Since both the NCSTRL Dienst server and the CS-TR Dienst server are located at Cornell University, it is interesting that the CS-TR QR had twelve indexers with a lower mean response time than 4.7 seconds, the lowest mean response time for the NCSTRL QR. Moreover, we see that NCSTRL's mean response time for every indexer was higher than CS-TR's (e.g. 6.8 seconds for a response from *www.cs.umass.edu* to NCSTRL and 4.7 seconds for a response from *www.cs.umass.edu* to CS-TR). It's unlikely these differences are related to network topology, however, it is possible that the CS-TR QR contacted the indexers at different times than the NCSTRL QR, which could have caused different network latencies as well as different indexer response times.

The two indexers located on the same machine as the NCSTRL QR had mean response times of 7.9 seconds (*ncstrl:8090*) and 8.7 seconds (*ncstrl:8099*). Many indexers not located on the same LAN had a smaller mean response time to the NCSTRL QR, such as *www.cs.umass.edu:80* at 6.8 seconds. Again, this points to performance factors beyond the state of the network.

The standard deviations for indexer response times to the NCSTRL QR lie in a smaller range than those for the CS-TR QR: all range from 1.4 seconds to 3.2 seconds except *ncstrl:8080* at 4.5 seconds and *cs.nyu.edu:80* at 3.6 seconds. However, we still have a large range of mean response times, from 4.7 seconds for *www.icase.edu* to 13.0 seconds for *www.cs.umd.edu*. Once more, a single QR's view of individual indexer performance data indicates that each indexer should be viewed as a separate entity.

5.2.3 BERKELEY QR – indexer response times, given that the indexer responded

We collected 54,852 responses from indexers for the BERKELEY QR; these results are shown in Table 16.

Table 16 - Indexer response times for BERKELEY QR

indexer	no. obs	high	low	mean	median	mode	std dev
cs-tr.cs.cornell.edu:80	5793	97	1	5.1	5	4	3.0
cs.nyu.edu:80	480	21	0	6.2	5	4	3.5
dri.cornell.edu:80	5745	92	2	6.3	6	5	2.9
ei.cs.vt.edu:8090	5631	97	0	5.4	5	5	3.1
lite.ncstrl.org:3803	1370	57	2	10.6	11	10	3.4
ncstrl.cc.vt.edu:8080	7667	98	1	8.1	6	4	5.9
ncstrl.cc.vt.edu:8090	5581	92	2	7.6	7	6	3.8
ncstrl.cs.cornell.edu:8090	1048	38	1	7.9	7	6	3.8
ncstrl.cs.cornell.edu:8099	1547	94	2	8.3	8	7	4.2
ncstrl.cs.uiuc.edu:80	458	21	3	7.2	6	6	2.4
sunsite.berkeley.edu:80	16	14	5	7.7	7.5	5	3.0
www.cc.gatech.edu:81	788	91	2	6.8	6	5	4.4
www.cs.dartmouth.edu:80	813	22	3	9.9	9	9	2.5
www.cs.umass.edu:80	4763	94	1	7.8	7	6	3.9
www.cs.umd.edu:80	247	37	6	12.0	12	14	3.2
www.icase.edu:80	5482	97	1	5.1	4	4	3.0
www.ics.forth.gr:7000	1797	97	3	10.9	9	8	6.1
www.tc.cornell.edu:80	5626	92	1	5.2	5	4	2.7
all indexers	54852	98	0	6.8			

As with the other QRs, BERKELEY does not see the indexer on its own machine as the fastest indexer. The mean response time for the BERKELEY indexer was 7.7 seconds, and there were nine other indexers with smaller mean response times.

The range of mean response times is similar to that of the NCSTRL QR, with a minimum mean response time of 5.1 seconds for both the CS-TR and *www.icase.edu* indexers and a maximum mean response time of 12.0 seconds for *www.cs.umd.edu*. We observe that the indexer response times for the NCSTRL QR have more in common with the indexer response times for the BERKELEY QR than with the indexer response times at the CS-TR QR, despite the similar Internet locations of the NCSTRL and CS-TR Dienst servers. Again we are reminded that the QR view of indexer performance is unique and cannot be explained by network topology alone.

5.2.4 LITE QR – indexer response times, given that the indexer responded

5,811 indexer responses to the LITE QR were collected; these results are shown in Table 17.

Table 17 - Indexer response time for LITE QR

indexer	no. obs	high	low	mean	median	mode	std dev
cs-tr.cs.cornell.edu:80	567	16	0	2.4	2	2	1.8
cs.nyu.edu:80	73	19	0	4.8	3	3	4.8
dri.cornell.edu:80	539	13	0	2.8	2	2	2.1
ei.cs.vt.edu:8090	545	18	0	1.9	1	1	2.0
lite.ncstrl.org:3803	396	20	0	10.2	10	9	4.4
ncstrl.cc.vt.edu:8080	427	19	0	4.3	3	3	3.6
ncstrl.cc.vt.edu:8090	585	18	0	2.5	2	0	2.9
ncstrl.cs.cornell.edu:8090	557	19	0	3.8	3	0	3.2
www.cc.gatech.edu:81	102	9	0	1.4	1	0	2.0
www.cs.dartmouth.edu:80	85	17	0	5.8	5	6	3.6
www.cs.uiuc.edu:80	88	11	0	1.0	0	0	1.8
www.cs.umass.edu:80	516	19	0	4.7	4	2	4.1
www.cs.umd.edu:80	83	16	1	9.9	10	11	3.1
www.cs.utah.edu:80	69	4	0	0.3	0	0	0.7
www.icase.edu:80	521	18	0	1.1	0	0	2.6
www.ics.forth.gr:7000	114	19	2	6.6	5	4	3.5
www.tc.cornell.edu:80	544	80	0	1.2	0	0	3.8
all indexers	5811	80	0	3.4			

The LITE QR's highest mean response time was for the indexer on the same machine, which is curious. We also see that LITE's view of itself as an indexer (mean 10.2 seconds) was slower than the NCSTRL QR's view (9.8 seconds) or the CS-TR QR's view (7.2 seconds) of LITE as an indexer. However, the NCSTRL and LITE QRs had the same median of 10 seconds and mode of 9 seconds for LITE as an indexer. Again, this underscores the notion that different QR views of the same indexer are different.

As with the other QRs, we note the wide range of mean response times, from 0.3 seconds for the *www.cs.utah.edu* indexer to 10.2 seconds for the LITE indexer.

5.2.5 FORTH QR – indexer response times, given that the indexer responded

Table 18 - Indexer response times for FORTH QR

indexer	no. obs	high	low	mean	median	mode	std dev
alexandra.di.uoa.gr:80	44	40	3	12.0	10	4	8.6
dienst.csi.forth.gr:80	197	39	2	8.5	4	3	9.8
medoc.informatik.uni-hamburg.de:80	4	23	5	10.8	7.5		8.3
www-ir.inf.ethz.ch:80	3	20	4	13.0	15		8.2
www.ics.forth.gr:7000	137	38	3	9.8	8	3	7.1
www.sztaki.hu:80	3	10	6	8.3	9		2.1
www.sztaki.hu:8001	91	26	5	12.4	11	7	5.6
all indexers	479	40	2	10			

The FORTH QR only had 479 responses from indexers. Again, we are unwilling to extrapolate much from the FORTH data due to the small number of observations, but we present the information in Table 18 for completeness.

5.2.6 QR view of indexer response times, given that the indexer responded -- conclusions

We will now draw some conclusions by looking at mean indexer response times as viewed by the five QRs in our study. In Table 19, “ncstrl” is marked with an asterisk because the NCSTRL QR is at port 80 on the machine at *ncstrl.cs.cornell.edu*, while the NCSTRL indexer is at port 8090 at *ncstrl.cs.cornell.edu*.

Table 19 – Mean indexer response times between studied servers

	cs-tr		ncstrl*		berkeley		lite		forth	
	QR cs-tr sees indexer	indexer cs-tr as seen by QR	QR ncstrl* sees indexer	indexer ncstrl* as seen by QR	QR berkeley sees indexer	indexer berkeley as seen by QR	QR lite sees indexer	indexer lite as seen by QR	QR forth sees indexer	indexer forth as seen by QR
cs-tr	1.9	1.9	5.1	4.4	5.1	-	2.4	7.2	-	5.5
ncstrl*	4.4	5.1	7.9	7.9	7.9	-	3.8	9.8	-	-
berkeley	-	5.1	-	7.9	7.7	7.7	-	10.6	-	10.9
lite	7.2	2.4	9.8	3.8	10.6	-	10.2	10.2	-	6.6
forth	5.5	-	-	-	10.9	-	6.6	-	9.8	9.8

The data columns in Table 19 are shown in five groupings of two columns each – one grouping for each site in our study. The meaning of the two columns within each grouping is as follows, using the first grouping as an illustrative example. The first data column, headed “QR cs-tr sees indexer,” shows the mean indexer response times as perceived by the CS-TR QR for each of the indexers in our study. These figures are taken from Table 14 in section 5.2.1. The second data column, headed “indexer cs-tr as seen by QR,” shows the mean indexer response times for the CS-TR indexer, as perceived by the QRs indicated. These figures are taken from the tables in sections 5.2.1, 5.2.2, 5.2.3, 5.2.4, and 5.2.5. The double column groupings for other sites in our study are derived similarly. Missing values in Table 19 occur when a QR didn’t call a particular indexer, or (similarly) when an indexer wasn’t called by a particular QR. For example, the BERKELEY indexer is only called by the BERKELEY QR, implying the MMS’s ordered list of indexers for the BERKELEY repository did not include the BERKELEY indexer for any but the BERKELEY QR (see section 5.1.3).

It is easy to see from Table 19 that different QRs record varying response times for the same indexer – the QR view of each indexer is unique. For example, the CS-TR indexer has mean response times ranging from 1.9 seconds for the CS-TR QR to 5.1 seconds for the NCSTRL and BERKELEY QRs. Similarly, the NCSTRL indexer has response times ranging from 3.8 seconds for the LITE QR to 7.9 seconds for the BERKELEY and NCSTRL QRs. These differing mean response times from the QR view imply that sharing QR view indexer performance data may not be useful.

Another interesting feature shown in Table 19 is the similarity between the mean indexer response times as viewed by the NCSTRL and BERKELEY QRs. Given that the NCSTRL Dienst server is located at Cornell University in Ithaca, New York and the BERKELEY Dienst server is located in Berkeley, California, it is surprising how similar these response times are. It is even more surprising to note that the CS-TR Dienst server is also located at Cornell University in Ithaca, New York, but the NCSTRL QR response times are most similar to the distant BERKELEY QR. This underlines our previous conclusion: sharing this sort of data among QRs may not be useful and certainly involves more than knowledge of Internet topology.

Another conclusion we can draw from Table 19 is that the fastest indexer is not necessarily the indexer on the same machine as the QR. The CS-TR, NCSTRL, BERKELEY and LITE QRs all see CS-TR as the fastest indexer, despite the network travel required by all by the CS-TR QR. In fact, all of section 5.2 indicates that indexers located nearby on the Internet can be slower than indexers located further away.

We note that the CS-TR QR sees the LITE indexer with a mean response time of 7.2 seconds, while the LITE QR sees the CS-TR indexer with a mean response time of 2.4 seconds. There is a similar asymmetry between the NCSTRL QR's mean response time for the LITE indexer (9.8 seconds) and the LITE QR's mean response time for the NCSTRL indexer (3.8 seconds). With the limited information available from the Dienst logs, we cannot tell if these results were caused by server load, index size, network conditions or any number of other factors affecting indexer performance. Nevertheless, these asymmetries probably indicate that the LITE machine was underpowered during the course of our study, as inferred in section 5.1.6.

6 Indexer view: indexer processing time

We now examine the indexer view of indexer processing time. This differs from the QR view of indexers explored in section 5 in that the indexer view of itself has no network component. We want to explore the differences between the indexer view and the QR view of indexers to further our understanding of the uniqueness of the QR view.

The NCSTRL Dienst server does not act as an indexer, and during the period of our study, the LITE Dienst server did not have a version of the Dienst software that included indexer performance data in the logs. We therefore only examine indexer processing time for CS-TR, BERKELEY and FORTH. For complete information on how we got this information from the logs, see Appendix C.

6.1 CS-TR indexer

6.1.1 indexer view of CS-TR indexer

The Dienst server at CS-TR was called as an indexer 45,660 times for the two-month period of our study. Recall from Figure 5 that for each transaction an indexer should record two log messages – one for the *SearchBoolean* message that initiates the indexer query processing ("B" in Figure 5) and one that is the *Search Collection* message indicating the end of the transaction ("C" in Figure 5). One *SearchBoolean* transaction line had an error, but there were only 45,654 *Search Collection* lines in the CS-TR logs for the period studied – we expected 45,660 *Search Collection* messages. *629 Search Collection* lines had errors, and there was one unmatched *Search Collection* line, leaving 45,024 lines for indexer statistics. Results are in Table 20 and Table 21. For further information on how the logs were analyzed for indexer view data, see Appendix C.

According to Table 20, the mean processing time for all calling QRs is 0.1 seconds. But since an indexer processing time of 0 seconds actually means "less than one second", the mean of 0.1 seconds tells us that in the vast majority of cases, the known indexer processing time was under one second. This is readily apparent from Table 21, which indicates that 41,525 out of 45,024 observations on the CS-TR indexer, or 92%, are less than one second. The data is fairly consistent across calling QRs: all but three of the calling QRs had a mean response time of 0.1 seconds or less, and all but one calling QR had a median of 0 seconds and a mode of 0 seconds.

Table 20 - Indexer processing time for CS-TR indexer

calling QR	no. obs	max	min	mean	median	mode	std dev
128.122.27.18	177	1	0	0.0	0	0	0.2
128.173.40.200	3	0	0	0.0	0	0	0.0
128.42.1.132	467	1	0	0.1	0	0	0.2
128.84.152.49	396	2	0	0.1	0	0	0.3
139.91.151.4	1	0	0	0.0	0		
147.27.12.1	50	0	0	0.0	0	0	0.0
192.76.247.102	1	0	0	0.0	0		
192.84.227.1	95	2	0	0.1	0	0	0.3
DBVideo.Stanford.EDU	3390	7	0	0.1	0	0	0.3
DIENST.SRV.CS.CMU.EDU	531	2	0	0.1	0	0	0.3
WHEAT.TC.CORNELL.EDU	87	2	0	0.1	0	0	0.3
a.cs.uiuc.edu	1039	1	0	0.1	0	0	0.3
antares.inf.ethz.ch	5	0	0	0.0	0	0	0.0
brahma.sics.se	3	0	0	0.0	0	0	0.0
cashew.cs.tamu.edu	93	3	0	0.1	0	0	0.3
cs-tr.cs.cornell.edu	14831	13	0	0.1	0	0	0.5
dienst.iei.pi.cnr.it	16	1	0	0.1	0	0	0.3
ei.cs.vt.edu	46	1	0	0.0	0	0	0.2
flounder-f.icase.edu	644	1	0	0.1	0	0	0.2
flounder.icase.edu	67	1	0	0.0	0	0	0.2
guild.cs.cornell.edu	2	0	0	0.0	0	0	0.0
hpschlichter10.informatik.tu-muenchen.de	167	1	0	0.0	0	0	0.2
inf.informatik.uni-stuttgart.de	69	1	0	0.0	0	0	0.1
kernighan.cs.umass.edu	223	3	0	0.1	0	0	0.3
larc.cs.Virginia.EDU	582	3	0	0.1	0	0	0.3
maya.iei.pi.cnr.it	1	0	0	0.0	0		
ncstrl.cs.cornell.edu	11560	3	0	0.1	0	0	0.3
pasteur.ics.uci.edu	489	3	0	0.1	0	0	0.3
pita.cs.umd.edu	793	10	0	0.2	0	0	0.7
pooof.cs.utah.edu	44	1	0	0.0	0	0	0.2
researchsmp2.cc.vt.edu	6	1	0	0.2	0	0	0.4
rzfspc51.informatik.uni-hamburg.de	62	1	0	0.0	0	0	0.1
siwenna.cc.gatech.edu	2	0	0	0.0	0	0	0.0
tex.iei.pi.cnr.it	4	1	0	0.5	0.5	0	0.6
tr.cs.washington.edu	736	3	0	0.1	0	0	0.3
ubkaaix6.ubka.uni-karlsruhe.de	61	1	0	0.0	0	0	0.2
windsor.cs.dartmouth.edu	70	1	0	0.1	0	0	0.2
www-ucpress.Berkeley.EDU	5920	5	0	0.1	0	0	0.3
www.bmstu.ru	70	3	0	0.0	0	0	0.4
www.cs.wisc.edu	2221	6	0	0.1	0	0	0.3
all calling QRs	45024	13	0	0.1	0	0	0.4

Table 21 - Number of observations for all calling QRs for Table 20

secs	no. obs.
0	41525
1	3231
2	166
3	64
4	12
5	9
6	4
7	3
8	2
9	0
10	1
11	4
12	2
13	1

6.1.2 QR views of CS-TR indexer compared with indexer view

Table 22 illustrates the difference between the CS-TR indexer view of itself and the QR view of the CS-TR indexer. It is immediately apparent that the indexer view shows the same mean response time for all four calling QRs in our study (the FORTH QR did not call CS-TR as an indexer during the two-month period), while the QR views of the indexer vary. Thus the indexer view of response time does not appear to be very useful as a means of illustrating QR view response times.

Table 22 - QR and indexer views of mean times for CS-TR indexer

calling QR	indexer view	QR view
cs-tr	0.1	1.9
ncstrl	0.1	5.1
berkeley	0.1	5.1
lite	0.1	2.4

According to Table 22, the CS-TR indexer has a mean response time of 0.1 seconds for the four QRs in our study that call this indexer. CS-TR as a QR perceives the mean response time for the CS-TR indexer as 1.9 seconds, so there is approximately a two-second discrepancy between the mean processing time of the CS-TR indexer view and the CS-TR QR view. The discrepancy can be explained by indexer overhead not included in the *Search Collection* messages (see Appendix C), and by connection time for the CS-TR indexer to receive an *http* protocol request and to send a response back to the CS-TR QR.

The NCSTRL QR, which is also located in the Cornell University Computer Science department, had a mean response time of 5.1 seconds for the CS-TR indexer during our study. Likewise, the BERKELEY QR recorded a mean response time of 5.1 seconds for the indexer. The network travel times, the connection times and indexer overhead account for nearly all of the mean response times from the NCSTRL and BERKELEY QR view, as compared with less than one second response times from the indexer view.

The LITE QR recorded a mean response time of 2.4 seconds for the CS-TR indexer. This is a smaller mean response time than that for NCSTRL, which seems odd since NCSTRL is closer to CS-TR on the network. This reminds us that the QR view of an indexer's response time is unique to each QR – network latency and connection times will vary from one QR to another, so they are best tracked by the QRs themselves.

6.2 FORTH indexer

6.2.1 indexer view of FORTH indexer

The FORTH indexer was contacted 281 times during our period of study. It had no *SearchBoolean* transaction lines or *Search Collection* lines with errors, and there were no unmatched *Search Collection* lines, so we have statistics for all 281 indexer uses. Results are shown in Table 23 and Table 24.

Table 23 - Indexer processing time for FORTH indexer

calling QR	no. obs	max	min	mean	median	mode	std dev
alexandra.di.uoa.gr	60	4	0	0.7	0	0	0.9
dienst.csi.forth.gr:10001	158	8	0	0.6	0	0	1.0
dienst.csi.forth.gr:10002	40	1	0	0.4	0	0	0.5
dienst.csi.forth.gr:10003	13	2	0	0.6	0	0	0.8
dienst.csi.forth.gr:10004	5	1	0	0.4	0	0	0.5
dienst.csi.forth.gr:10005	1	1	1	1.0	1		
dienst.csi.forth.gr:10006	1	2	2	2.0	2		
www.sztaki.hu:2000	3	2	1	1.7	2	2	0.6
allQRs	281	8	0	0.6	0	0	0.9

Table 24 - Number of observations for all calling QRs for Table 23

secs	no. obs.
0	160
1	91
2	20
3	6
4	3
5	0
6	0
7	0
8	1

According to Table 23, the FORTH indexer processed 160 out of 281 observations, or 57%, in less than one second. 251 out of 281 observations, or 89%, were processed in less than two seconds. There were only ten observations taking three or more seconds. The data is consistent across calling QRs: all but three of the calling QRs had both a median and mode of zero seconds, and there were only five total observations for the three remaining calling QRs. There is similar consistency of the mean indexer processing time, with two means at 0.4 seconds, two means at 0.6 seconds, one at 0.7 seconds, and the remaining means for individual calling QRs 1.0 seconds or higher, but again, these are the three calling QRs that have only five data points among them.

6.2.2 QR views of forth indexer compared with indexer view

We're not sure which of the calling QRs listed in Table 23 correspond exactly to the FORTH QRs in our study, so we cannot make any inferences about the difference between the FORTH view of itself as an indexer and the FORTH QR view of the FORTH indexer.

6.3 BERKELEY indexer

6.3.1 indexer view of BERKELEY indexer

The BERKELEY QR was the only Dienst server to call BERKELEY as an indexer during the two-month period of our study. We surmised in section 5.1.3 that the NCSTRL maintenance team excluded the BERKELEY indexer from the BERKELEY publisher mappings due to poor performance. According to the *SearchBoolean* messages in the logs, BERKELEY called itself as an indexer 227 times. There were five *Search Collection* statistics messages with errors, leaving 222 observations for us to analyze. Results are shown in Table 25 and Table 26.

Table 25 - Indexer processing time for BERKELEY indexer

calling QR	no. obs	max	min	mean	median	mode	std dev
www-ucpress.Berkeley.EDU	222	11	0	0.7	0	0	1.6
allQRs	222	11	0	0.7	0	0	1.6

Table 26 - Number of observations for all calling QRs for Table 25

secs	no. obs.
0	138
1	60
2	11
3	3
4	2
5	1
6	1
7	1
8	3
9	1
10	0
11	1

Table 25 shows the mean processing time for the BERKELEY indexer was 1.9 seconds, and Table 26 indicates that 198 out of 222 observations, or 89%, occur in less than two seconds.

6.3.2 QR views of BERKELEY indexer compared with indexer view

Table 27 - QR and indexer view of mean times for BERKELEY indexer

calling QR	indexer view	QR view
berkeley	0.7	7.7

In Table 27, we see that the BERKELEY QR shows mean response time of 7.7 seconds for the BERKELEY indexer, compared with the 0.7 second mean for the indexer view. The seven second discrepancy between the QR view and the indexer view includes indexer overhead not included in the *Search Collection* messages (see Appendix C) and connection time for the BERKELEY indexer to communicate via the *http* protocol with the BERKELEY QR. Seven seconds is a long time for these tasks, but recall from Table 10 in section 5.1.3 that only 16 out of 228 calls to the BERKELEY indexer responded to the BERKELEY QR before the search timeout. Clearly there was some sort of difficulty with the BERKELEY indexer during the period of our study.

7 Query complexity

This is an ancillary topic that doesn't bear directly on the other work in this paper, but we include it for its potential relevance to subsequent predictive work.

We analyzed search criteria in the search statistics log messages for all QRs in our study. In order to quantify query complexity, we examined the split of non-fielded (*i.e.*; those that don't specify a bibliographic category) vs. fielded (*e.g.*; *author equals "Hopcroft")* queries for each QR, how many search terms were used in non-fielded queries and the number of fields used in fielded queries. For details on the log analysis, see Appendix D.

7.1 Fielded vs. non-fielded queries

Table 28 shows the total number of queries answered by each QR broken down into fielded and non-fielded queries. See Appendix D for details.

Table 28 - Fielded vs. non-fielded queries

QR	total	fielded	non-fielded	% fielded	% non-fielded
cs-tr	17238	7331	9907	43	57
ncstrl	12743	5457	7286	43	57
berkeley	7084	2499	4585	35	65
lite	796	576	220	72	28
forth	203	169	34	83	17
all QRs	38064	16032	22032	42	58

There are more non-fielded than fielded queries overall: 58% non-fielded vs. 42% fielded queries. The QRs with the bulk of the data, CS-TR and NCSTRL, both have 57% non-fielded queries. The BERKELEY QR has a higher percentage of non-fielded queries, 65%, while the QRs with the least data, LITE and FORTH, have more fielded than non-fielded queries.

7.2 Non-fielded queries – number of terms

In order to quantify complexity of non-fielded queries, we examined how many terms were in each non-fielded query -- how many words were joined by boolean operators "and" or "or." Results are shown in Table 29 and Table 30. Appendix D has further details on this analysis.

12,485 out of 22,032, or 57%, of non-fielded queries had only one search term. Since there were 38,064 total queries, 33% of the total queries were non-fielded queries of one term. From a user's perspective, a one term non-fielded query is the simplest non-null query possible: a third of the queries in our study were this simple.

For all QRs, there were only seven non-fielded queries with ten or more terms, and all of the QRs had similar distributions of number of terms, with median and mode of one term for all except the LITE QR with a median of two terms.

Table 29 - Number of terms in non-fielded queries

QR	no. obs.	max	min	mean	median	mode
cs-tr	9907	23	1	1.7	1	1
ncstrl	7286	11	1	1.6	1	1
berkeley	4585	44	1	1.7	1	1
lite	220	6	1	1.7	2	1
forth	34	3	1	1.2	1	1
all QRs	22032	44	1	1.7	1	1

Table 30 - Number of observations for Table 29

# terms	cs-tr	ncstrl	berkeley	lite	forth	all QRs
1	5602	4093	2659	104	27	12485
2	2900	2220	1275	86	6	6487
3	924	695	434	22	1	2076
4	224	165	119	3		511
5	157	52	50	4		263
6	47	31	17	1		96
7	26	20	13			59
8	11	8	6			25
9	8	0	1			9
10	4	1	9			14
11	2	1	0			3
12	0		1			1
13	1		0			1
…	…		…			…
23	1		0			1
…			…			…
44			1			1

7.3 Fielded queries – number of fields

In order to quantify complexity of fielded queries, we counted how many of the fields *author*, *title*, *abstract* and *name* in the query had non-null values. Note that it is possible (but rare) for NCSTRL queries to have values in other fields, hence the four fielded queries listed with zero fields. Results are shown in Table 31 and Table 32. See Appendix D for further details on how the logs were analyzed for this data.

Table 31 - Number of fields in fielded queries

QR	no. obs.	max	min	mean	median	mode
cs-tr	7331	4	0	1.2	1	1
ncstrl	5457	4	0	1.2	1	1
berkeley	2499	4	0	1.2	1	1
lite	576	3	0	1.1	1	1
forth	169	2	1	1.4	1	1
allQRs	16032	4	0	1.2	1	1

Table 32 - Number of observations for Table 31

# fields	cs-tr	ncstrl	berkeley	lite	forth	all QRs
0	1	1	1	1	0	4
1	6196	4654	2126	494	107	13577
2	1037	727	330	78	62	2234
3	95	71	41	3		210
4	2	4	1			7

Of the 16,032 fielded queries, 13,577, or 85%, had search terms in only one field. This corresponds to 36% of the total number of queries in our study, 38,064. The simplicity of the fielded searches also depends on the number of search terms in each field, but we did not break down the query data to get this information. However, since we know that a non-fielded query searches the *author*, *title*, and *abstract* inverted index files, it is possible that fielded queries of one or even two fields are actually "simpler" for the NCSTRL indexers to process than non-fielded queries.

Not surprisingly, the median and mode for all QRs was one field for all fielded queries. The mean number of fields ranged only from 1.1 to 1.4, with an overall mean of 1.2 fields for all fielded queries, indicating the similarity of the distributions of number of terms in the queries across different QRs.

8 Conclusions and future work

We have shown that the QR view of an indexer is a useful one: there is a significant difference between the performance of an indexer from the perspective of a QR and the indexer's performance from the perspective of the indexer itself. Moreover, different QRs perceive the performance of the same indexers differently. Lastly, the time a QR spends waiting for indexers to respond is significant, implying that the overall system performance can be improved if this waiting time can be reduced. We are currently looking at techniques to reduce the QR waiting time.

Dienst QRs do not have the best response records from the "local" indexer on the same machine, and network proximity and server speed alone are unable to explain response time differences between indexers. We surmise that there is a temporal nature to the data, related perhaps to when exactly the QR contacted the indexer and the snapshot states of the network, the QR server and the index server.

We believe it is possible for the QR to reduce its wait time for indexers with an adaptive algorithm that uses past performance data to predict performance. The algorithm needs to take advantage of data gathered at the QR and of redundancy in available indexers, and needs to be both flexible and fault tolerant. We intend to explore the following methods of analyzing past performance data for indexers in order to make predictions about indexer responses and response times: averages, low pass filters, and timed low pass filters.

We also learned that many NCSTRL queries are simple, giving validity to the notion of simulating load in NCSTRL with a simple query, as might be done in a future predictive performance study.

Acknowledgments

This work was supported by DARPA grant MDA 972-96-1-006 with the Corporation for National Research Initiatives, DARPA contract N66001-97-C-8542, and NSF grant CDA-9529253. This paper does not necessarily represent the views of CNRI, NSF, or DARPA. The authors are grateful to David Fielding for his contributions.

Appendix A – details of log analysis for significance of QR wait time for indexers

Recall from Figure 5 that there are two Dienst log messages pertaining to QR processing, the User Interface *Query* transaction message and the *search statistics* message. A *Query* log message ("A" in Figure 5) indicates that a query was received at the UI. A *search statistics* log message ("D" in Figure 5) indicates that the QR finished processing a query, whether or not each indexer polled responded, and the number of seconds the QR waited for each indexer. As with all Dienst log messages, there is a timestamp indicating when these messages were written to the log. Unfortunately, the logs do not provide any explicit matching of these two log messages; however, the match can be inferred in most cases with a three step process.

First we analyzed all 38,069 *search statistics* lines in the logs for all five QR in our study, less the three lines we were unable to parse properly. We determined the minimum number of seconds the QR waited for all indexers to respond by examining the response times of all indexers in the *search statistics* log message. If no indexer was repeated, and if there were more indexers than authorities indicated in the search, then the wait time was set to the maximum of all the known indexer response times for this *search statistics* line. If an indexer was repeated, or if there were more authorities than indexers, then we knew we had a two phase search and used a different algorithm to come up with minimum wait time. Since the known two phase searches were dropped later, we won't address that algorithm here. Appendix B goes into more detail on interpreting indexer results from search statistics log messages.

We can only deduce a minimum amount of time the QR waited for indexers because we can't definitively determine whether each search statistics message represents a one phase or a two phase search. In the case of two phase searches, the QR waits until the phase one timeout, then determines which indexers to contact for phase two, and then waits for results from all the phase two indexers or for the phase two timeout. We do not know how long the phase one or the phase two timeout is, nor can we always tell which indexers were polled in phase two. At this time it is only important to note that the wait time we infer from the logs is a minimum.

The second step in the process to match *Query* and *search statistics* messages was to determine which *Query* messages occurred prior to the inferred QR wait time for indexers. We looked through the daily Dienst log containing the relevant *search statistics* message and culled out all *Query* messages occurring prior to the inferred beginning of the QR wait time for indexers.

The third and last step in the matching process was to do a crude matching of a query argument string (or a piece of one) against the *Query* messages occurring at an eligible time. If there was a single *Query* transaction message occurring at an eligible time that matched the string, then we knew we had an exact match and we kept the observation. There were 32,690 such cases. There were 5,247 cases with multiple *Query* messages at an eligible time and that matched the string. We dropped all of these because we had no way to determine which *Query* message should match which *search statistics* message. We also dropped the 128 *search statistics* messages that didn't pass our crude string match, and the one observation that had no *Query* lines in the log occurring at the right time, probably due to some logging error.

Exact matching of *Query* messages to *search statistics* messages is important for two reasons. First, we subtract the *search statistics* timestamp from the *Query* timestamp to get the total QR processing time for a query. Clearly if we are not confident of the match, then our data for total QR processing time would not be reliable either. We do note that there is some QR processing occurring before the *Query* message is written to the logs and after the *search statistics* message is written to the logs, but we treat this unknown processing time as a) unobtainable and b) insignificant compared to the known QR processing time. The second reason exact matching of *Query* messages to *search statistics* messages is important is because we want to examine how much of the total QR time is spent waiting for indexers. For each observation with an exact match, we computed this by dividing the time the QR waited for indexer responses by the total QR time.

We noticed that in twelve instances, the logs indicated that a QR had waited more than two minutes for an indexer to respond. We also had 25 instances of QR processing time (QR total time – QR wait time for indexers) greater than two minutes. This small number of observations with large times would have skewed our results, so we ignored these *search statistics* lines for the purposes of determining the significance of network costs to the QR.

We also dropped 321 exact matches that were known two phase searches and ended up with 32,352 *search statistics* lines for which we could not deduce two phases in the search, which matched exactly one *Query* message, and which did not have a wait time or a QR processing time greater than two minutes.

We were surprised to discover that in some cases the logs had timestamps out of order. This is most likely caused by one Dienst process locking the log file while another is waiting to write to it. In most cases, these timestamps were only a second or two off, but in the case of the FORTH log for April 04, 1997, there were so many timestamps vastly out of order, we decided to drop the entire FORTH log for that day from our analysis.

Appendix B – details of log analysis for QR view of indexer performance

Indexer performance from the QR view is determined from *search statistics* log messages (see "D" in Figure 5). In this log message indexer results are separated from the query arguments. If we were unable to separate the query arguments from the indexer results, as happened in three instances out of 38,069, we ignored the search statistics message.

Indexer results have four parts in most cases: a hostname, a port number, the number of hits, and the response time in seconds (as a floor value). Sometimes there is a fifth part, which is an error message of some kind. Here are some examples:

```
host port 5 7
host port 0 5 The+search+keyword+"li"+matched+too+many+words,+try+a+longer+string
host port –1 1 Invalid+response+from+server:+-500+
host port –1 Server+Timed+Out
host port –1 6 Could+not+connect+to+its+server
```

We counted the first three instances in the above examples as responses (successful transactions) because the QR got a responsefrom the indexer. In the last two cases above, the QR received no communication from the indexer (unsuccessful transactions), so we counted these as non-responses.

We tracked responses and non-responses for each indexer for each QR. In the case of responses, we also tracked the response times. As with the log analysis of the significance of QR wait time for indexers, we drop indexer response times greater than two minutes. However, we do not drop ALL indexer observations from the search statistics lines with these large response times: only the particular indexer observation is dropped. This means that the individual indexer performance data may have more observations represented in this context than the collective indexer performance data represented in the context spelled out in Appendix A.

We make no distinction between one phase and two phase searches here, so again, we are looking at more data than was ultimately studied via the methods in Appendix A. It is true that the search timeout affects response rate, as waiting longer may allow more indexers to respond before the timeout. But since we have no way to infer the search timeouts accurately from the logs, and since the timeout values could have changed during the two month period of our study without our knowledge, we do not concern ourselves with the issue of whether phase two search timeouts are different than phase one search timeouts. This is the only possibly relevant issue from the standpoint of individual indexer performance from the QR view.

The response times indicated in the logs represent the time between when the QR received a response from an indexer and when the Dienst QR started listening for responses from indexers (as soon as it has finished sending *SearchBoolean* messages to all indexers for a query).

Note that we ignored a QR's demotion and promotion of indexers (see section 2) because we examined response rates only when the QR was attempting to contact an indexer – the QR had already chosen the indexer for query processing.

Appendix C – details of log analysis for indexer performance

Indexer processing time is determined from *Search Collection* log messages (see "C" in Figure 5). These messages are only present in Dienst logs generated by Dienst code version 4-1-0 and above, which excluded the LITE logs. At first we had hoped to match *Search Collection* log messages with *SearchBoolean* log messages (see "B" in Figure 5), but since there is no way to be sure of correct matching, we ultimately abandoned this idea.

We dropped 629 *Search Collection* messages with errors from the CS-TR logs and 5 *Search Collection* messages with errors from the BERKELEY logs. We were unable to detect a prior *SearchBoolean* message to match with a *Search Collection* message in one instance for the CS-TR logs – we left in this vestigial logic from our first approach to this analysis. All other *Search Collection* messages were analyzed – we noted the calling QR and the number of seconds indicated to process the *SearchBoolean* Dienst request.

The times in the *Search Collection* messages do not fully capture the indexer processing time, as they don't reflect a) the time to accept the socket connection to receive the http protocol request, b) the time to fork a Dienst process to handle the indexer request, c) the time to prepare the arguments for processing, or d) the time to return the results to the calling QR via a socket connection. *Search Collection* message times basically reflect the time to actually search inverted indexes at the Dienst server.

Appendix D – details of log analysis for characterization of queries

In order to examine query complexity in NCSTRL, we examined all queries answered by the five QRs in our study -- we analyzed search criteria in *search statistics* log messages (see "D" in Figure 5). Of the 38,069 *search statistics* lines in the logs, we analyzed queries for 38,064 of them – five lines were rejected because we could not separate search criteria from the indexer data or because the search criteria was corrupted.

After separating the query strings from the indexer statistics, we broke the search criteria into name-value pairs. Non-fielded NCSTRL queries in *search statistics* messages are distinguished by having the same value for "title," "author" and "abstract" fields. When we identified a non-fielded query, we counted how many words were joined by boolean operators "and" or "or" or another representation of these boolean operators, such as "&".

The queries that did not have the same value for "title," "author" and "abstract" fields were non-fielded queries. We counted the number of fields in fielded queries with field names "author," "title," "abstract," or "name." It is possible for valid NCSTRL queries to have values in other fields as well, but the vast majority only uses these four field names.

References

[1] "Information Retrieval (Z39.50): Application Service Definition and Protocol Specification," ANSI/NISO 1995.

[2] C.-C. Chang and H. Garcia-Molina, "Evaluating the Cost of Boolean Query Mapping," presented at Second ACM International Conference on Digital Libraries, 1997.

[3] W. Chu, "Optimal File Allocation in Multiple Computer Systems," *IEEE Transactions on Computers*, October , 1969.

[4] J. Davis and C. Lagoze, "Dienst protocol version 5.0," 1997; http://www.cs.cornell.edu/lagoze/dienst/protocol5.htm.

[5] L. Gravano, C.-C. Chang, H. Garcia-Molina, and A. Paepcke, "STARTS: Sanford Proposal for Internet Meta-Searching," presented at ACM SIGMOD International Conference on Management of Data, 1997.

[6] L. Gravano, H. Garcia-Molina, and A. Tomasic, "The Effectiveness of GlOSS for the Text-Database Discovery Problem," presented at ACM SIGMOD International Conference on The Management of Data, 1994.

[7] C. Lagoze, “From Static to Dynamic Surrogates: Resource Discovery in the Digital Age,” in *D-Lib Magazine*, 1997.

[8] C. Lagoze and D. Fielding, “Defining Collections in Distributed Digital Libraries,” *D-Lib Magazine*, November 1998.

[9] C. Lagoze, D. Fielding, and S. Payette, “Making Global Digital Libraries Work: Collection Service, Connectivity Regions, and Collection Views,” presented at ACM Digital Libraries '98, Pittsburgh, 1998.

[10] C. Lagoze, E. Shaw, J. R. Davis, and D. B. Krafft, “Dienst Implementation Reference Manual,” Cornell University Computer Science, Technical Report TR95-1514, May 1995.

[11] R. Lasher and D. Cohen, “A Format for Bibliographic Records,” Internet Engineering Task Force, RFC 1807, June 1995.

[12] B. M. Leiner, “The NCSTRL Approach to Open Architecture for the Confederated Digital Library,” *D-Lib Magazine*, December 1998.

[13] D. D. E. Long, J. L. Carroll, and C. J. Park, “A Study of the Reliability of Internet Sites,” presented at Proceedings of the Tenth Symposium on Reliable Distributed Systems, 1991.

[14] C. L. Viles and J. C. French, “Availability and Latency of World Wide Web Information Servers,” *Computing Systems*, 8 (1), pp. 61-91, 1995.