Classifying Network Architectures for Locating Information Sources

Technical Report TRCS96-23

Version 0.1.1

R. Dolin D. Agrawal A. El Abbadi
Department of Computer Science
University of California
Santa Barbara, CA 93106

{rad, agrawal, amr}@cs.ucsb.edu

September 14, 1996

Abstract

This paper presents three broad classes of network architecture that support discovery of information sources. Relevant concepts such as query routing and the extraction, propagation, and retrieval of metadata are defined. Based on these concepts, different models of locating and querying relevant information sources are presented. Finally, we estimate several important characteristics of these models and classes as well as their expected scalability.

Keywords: resource discovery, distributed systems, scalability, metadata, internet, document database.

1 Introduction

Finding appropriate sources of information on the Internet increases in difficulty with the increasing number of Internet users, the number of sources providing electronic *documents*, the number of systems for locating these documents, and the volume of available data. An information source is any site in the network that provides electronic documents such as journal articles, books, hypertext pages, images, files, etc. Prior to digital networks, the number of potential information sources was relatively limited, mainly by locality; hence, a user could discover what was available by browsing individual sources. In a wide-area network such as the Internet, information resides in numerous heterogeneous systems, which are geographically distributed; each such distributed system may provide many information sources. Individual systems provide various discovery tools, e.g., Archie, Veronica, Yahoo. However, users can get easily lost in the vast space of documents that are indexed by these systems. No system-independent location facility currently exists. What was once a problem of limited choice has become a problem of too much choice. The existence of thousands of newsgroups, World-Wide Web (WWW) sites, FTP archives, and digital libraries renders traditional methods of locating good information sources infeasible.

Current WWW indexes, such as AltaVista [1], Lycos [8], and Yahoo [11], which are designed for locating information on the Internet, are limited in several ways. For example, they are limited to on-line, text documents, and they give no control to the sources of the documents to determine the frequency with which they are indexed. The *gGlOSS* text database discovery system [6] represents sources by vectors of term frequencies. It is similar, in this regard, to the SMART automated text retrieval system [9], replacing the two-dimensional term/document matrix by a term/collection matrix. SMART and *gGlOSS* are only two of several automatic text retrieval systems. Other notable systems include ConQuest [4], which uses a static semantic network based on dictionary definitions, and Latent Semantic Indexing (LSI) [2], which uses a dynamic semantic network based on word co-occurrences within documents in a collection.

Text Symbol	Diagram Symbol	Description
QG	$\circ$	Query Generator
S	$\triangle$	Information Source
I	$\Box$	Intermediary

Table 1: Objects Used in Model Diagrams

Content Routing [10] is a system in which queries get routed to the available servers based on the expected relevance of the server to the query. This system also propagates descriptions of server contents through a hierarchy of information servers. Harvest [3] automatically indexes documents within a source and distributes these indexes. It can be used to describe WWW sites, newsgroups, repositories of technical reports, etc. and includes subsystems for caching and replicating the indexes.

This paper presents a systematic description of possible network architectures that support the discovery of information sources, and analyzes their differences. The next section describes relevant concepts such as query routing and the extraction, propagation, and retrieval of metadata. Based on these concepts, different models of locating and querying relevant information sources are presented within three broad classes. Finally, we estimate several important characteristics of these models and classes as well as their expected scalability.

2 Terminology

We first describe the terms which will be used in the network models. We define a *document* operationally as an atomic entity which can be searched for, retrieved, and viewed, though not necessarily stored, as a single unit, electronically or otherwise. A document is not required to be stored atomically, nor is it necessarily always used atomically; although one can access sections of a document, a document must be available as an atomic entity. A *collection* refers to a well defined set of documents. A *query* refers to a document specification against which documents in a collection can be compared for relative similarity. A *query generator* is a system which electronically creates and transmits queries; they are represented as circles, $\circ$, in the model diagrams. A *search engine* is an interface which accepts as input a query about a particular collection and returns as output a *query result*, which is a subset of the collection which it has determined contain documents relevant to the query. The subset does not contain the actual documents, but rather pointers to, and possibly descriptions of, the documents. A single collection may have multiple search engines associated with it, but by definition a search engine can work on only a single collection (i.e. we do not require a collection to be an input parameter to a search engine). An *information source*, or *source*, refers to a particular [document collection, search engine] pair. Sources are represented as triangles, $\triangle$, in the model diagrams. We point out that *collection* refers only to a set of documents, while *source* refers more generally to a machine, including its documents, search engine, metadata, etc.

Metadata refers to the description of an object. We make a distinction between *document metadata* and *collection metadata*. The former is information about individual documents, while the latter is summary information about an entire collection and possibly other descriptive information about the machine, network parameters, etc. Unless otherwise stated, metadata will refer to *collection* metadata. Therefore, a query result is just the document metadata that would be returned by a search engine given a particular query. It is not necessarily returned directly by a search engine, but by any system which gives the same document subset.

Finally, an *intermediary*, drawn as a small or large box, $\Box$, in the model diagrams, is a logical machine on the network which provides a level of indirection between a request for information and the original supplier of the result, usually an information source. A *query intermediary* acts as a pseudo search engine; it takes queries as input and provides query results as output. However, a query intermediary differs from a search engine in that 1) it does not necessarily have direct access to any document collection, and that 2) it may return query results for several sources simultaneously. A *metadata intermediary* accepts a request for either document or collection metadata and supplies the relevant result. Intermediaries may store information locally, or may request information either from another intermediary or from a source. A summary of the basic design objects shown in the model diagrams is presented in Table 1.

Our network models describe several common activities. *Query propagation* or *query routing* is the process of passing a query from a query generator to either a source or a query intermediary. This action is expected to result in *result retrieval*, which returns a query result to the query generator. *Result merging* involves combining query results for a query sent simultaneously to multiple sources. Documents in a result set are often ranked within that set; result merging can include *rank merging* where the union of the multiple result sets are re-ranked for the entire union. Result merging can also include *duplicate detection*, in which an attempt is made to detect if the same document has been returned by multiple sources.

Metadata extraction denotes the process of deriving either document or collection metadata. Document metadata extraction can take place either at a source or at an intermediary which has direct access to a source's documents. Collection metadata extraction is generally done only directly at a source, though there is no requirement for this. *Metadata propagation* involves the 'pushing' of metadata by a source to any other machine that either stores it locally for long-term use or else uses it immediately to help resolve a query and then discards it. *Metadata retrieval* involves the 'pulling' of metadata by an intermediary or a query generator. The models that follow have examples of both long-term and short-term use of both document and collection metadata. An *update* is the process of bringing up-to-date all stored metadata across the entire network about a particular source.

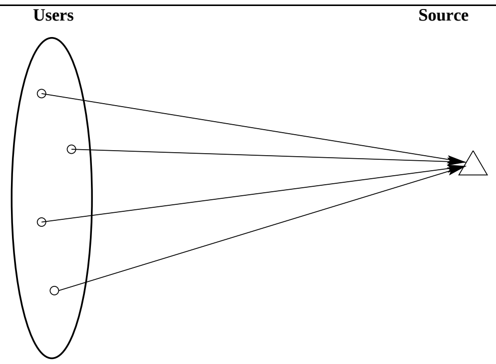
3 Network Models

This section describes three classes of network models, with increasing complexity, that encompass the discovery and querying of information sources. The first class of models uses no intermediate servers, so that the only communication is directly between query generators and information sources. The next level of complexity is obtained by placing a single intermediary between the query generators and the sources. The simplified model in this class is the system used by most current WWW index sites. The final level of complexity is obtained by allowing for multiple, communicating, intermediaries. The corresponding metadata for this class of models is potentially much more complex and carefully distributed. The models described below are summarized in Figure 1.

Figure 1 Model Classifications

- Class 1. Direct
  - Simple Model
  - Remote Model
  - Local Model
- Class 2. Single Intermediary
  - Brute-Force Model
  - STARTS Model
- Class 3. Multiple Intermediaries
  - Pharos Model

Figure 2 Simple Model Query



3.1 Direct

Within Class 1, the *Direct* models, we describe three architectures. These models are not intended to be realistic for a large-scale network, but rather exemplary of particular problems of scalability which will be discussed in Section 4.

The Simple Model is the most basic model, which contains one or more query generators but only a single information source. This model is shown in Figure 2. Queries are sent directly to the information source and the results are sent back directly to the generator of each query. Since we defined an information source as a unique [collection, search engine] pair, this model does not differentiate between simple, one-site sources, and complex, multi-site sources. We require only, from the perspective of the query generators, 1) that only unique query results are returned, and 2) that there is a single search engine query point for all of the (possibly distributed) data. Thus this model avoids any problems associated with result merging as discussed in Section 2.

The next model allows for multiple sources by simply adding them in and not worrying about which sources are queried nor how a query generator should merge the results. It is implicitly assumed that each query will be propagated to all sources. This model, the Remote Model, is depicted in Figure 3.

Neither the Simple Model nor the Remote Model include the use of any collection metadata. The next model, the Local Model, requires that each query generator pre-collect sufficient metadata about each source so that the best sources can be decided locally at the generator without first contacting any remote

Figure 3 Remote Model Query

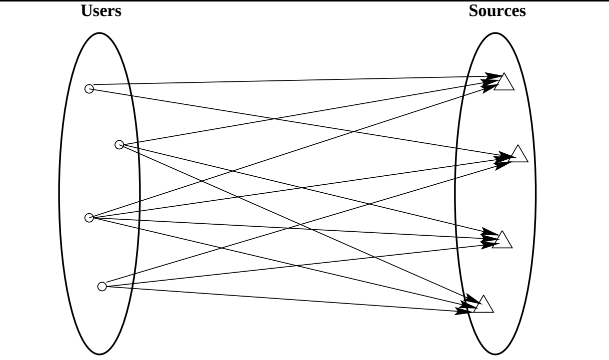
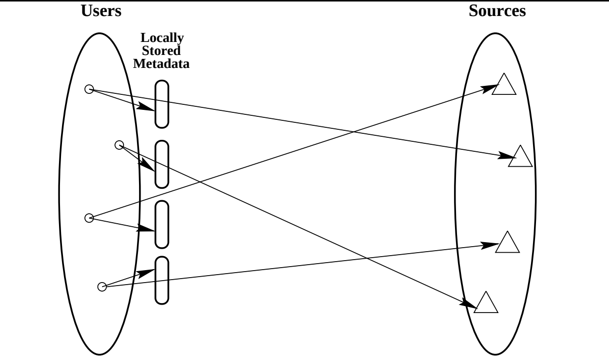


Figure 4 Local Model Query

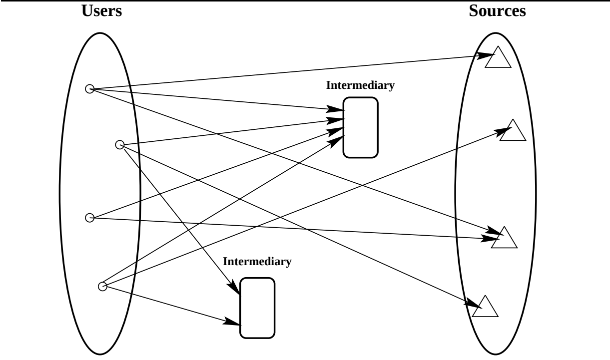


site. To handle a query, the only traffic generated in this model is that used to query the most relevant sources and retrieve their results. This model assumes that there is a standard method with which metadata can be retrieved and merged by the query generators. This model is shown in Figure 4.

3.2 Single Intermediary

Within Class 2, the *Single Intermediary* models, we describe two architectures: the Brute-Force (BF) Model and the STARTS Model [7]. Both these models maintain a single intermediary between the query generators and sources. This intermediary handles both query and metadata traffic, though there is no synchronization between its reception of a query and its (collection) metadata retrieval. Although there may be several intermediaries on the network simultaneously in this class of models, they do not communicate with each other. A query generator must decide which intermediary to use for a given query, though by definition, no method is provided for assisting in this decision. This uncertainty affects estimates of query network traffic, as will be discussed in Section 4.2.

Figure 5 Brute-Force Model Query



The simpler of the two models described in this class, the BF Model, is the system used by most, if not all, current WWW index sites and is depicted in Figure 5. In this model, an intermediary performs metadata extraction by gathering all documents available at all sources. Current implementations extract only limited document metadata to reduce storage space, and no collection metadata. In principle collection metadata could be extracted also. This process is driven completely by the intermediary; the sources have no control over the frequency of metadata collection nor over the type of metadata extracted. Queries are sent to the intermediary, which uses its pre-collected metadata to determine which of the documents it has analyzed are relevant, and then returns a (possibly ranked) result set to the query generator. Queries are not propagated to the sources, and results are not able to include information about documents which have not been analyzed directly by the intermediary.

A fairly new approach currently[1] being discussed among some of the major WWW index vendors is the STARTS Model [7], shown in Figure 6 on the following page. In this model, built on a *gGlOSS* [6] framework, each source performs its own (collection) metadata extraction. The intermediary then gathers this metadata rather than extracting its own. In principle, this procedure allows off-line documents to be included, since the intermediary is not required to have direct access to documents. Even though the intermediary retrieves the metadata, in principle this model allows the source to regulate its update frequency by including an expiration date field in the metadata. Although non-text metadata (see next section) is theoretically possible, the default metadata structure is based on lists of terms which appear in text document collections. Queries in the STARTS

[1]A STARTS Workshop occurred at Stanford University on August 1, 1996.

Figure 6 STARTS Model Query

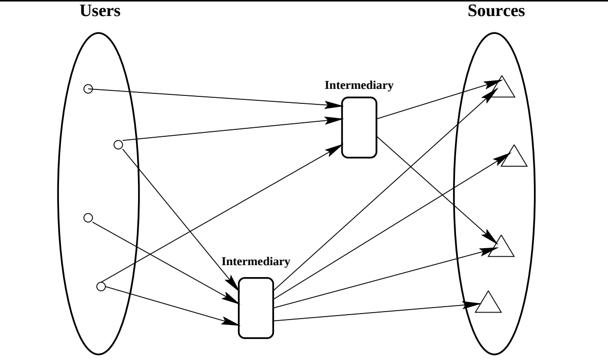
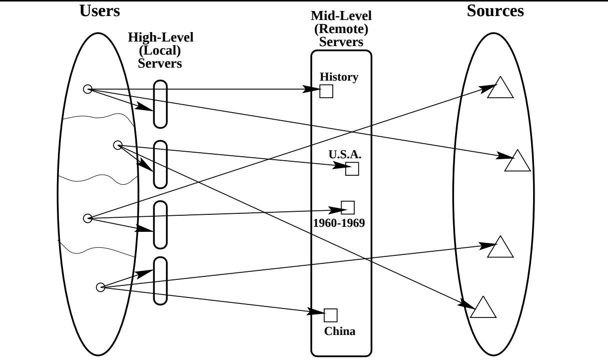


Figure 7 Pharos Model Query



Model are of a keyword matching nature, although the design allows for extensions to this query structure. When an intermediary receives a query, it analyzes its pre-collected metadata and chooses a set of relevant sources. The query is then propagated to these sources in a standard format and they return the results (document metadata retrieval), also in a standard format, back to the intermediary. The intermediary is then responsible for merging the results from all the queried sources and passing a single, ranked list of results back to the query generator.

The major differences that the STARTS Model has from the BF Model is that 1) it allows (actually, requires) each source to extract its own metadata, and 2) there are standard formats for information exchange (metadata, queries, and results). The STARTS Model does not require that an intermediary store document metadata as does the BF Model. Instead, it retrieves document metadata at query time, including, for example, an abstract where applicable. Therefore even though the query result size in the STARTS Model is perhaps roughly the same as in the BF Model, the query results are likely to be much more descriptive about the fewer documents returned. As a result, we expect that substantially fewer documents in the STARTS Model than in the BF Model are likely to be checked for relevance by having the query generator fetch the actual document, and that the *precision*[2] of the query result will be much higher.

3.3 Multiple Intermediaries

Within Class 3, the *Multiple Intermediaries* models, we define three extensions to standard metadata and then bring them together in a single design, Pharos. The first extension is *hierarchical* metadata, which means that the metadata is organized in a tree-like structure. Information at a parent node in the tree contains some form of summary of the information of all its children, though possibly with less detail. The second extension is that of *shared* metadata. Sharing metadata means that intermediaries can pass metadata between themselves in an organized manner. Thus, depending on the architecture, an intermediary can collect specific metadata about many sources; other systems can then retrieve that specialized information if and when they need it without having to communicate with the separate sources directly. The last extension is that of *non-text* metadata, which describes non-text aspects of documents and collections. For example, documents that include images, sounds, and maps may not be best described by words or terms associated with them. Many of these non-text documents, as well as many text documents, are also characterized by, for example, image feature-vectors, geographical coordinates, temporal information, etc. Given these metadata extensions, we describe a new model, Pharos, shown in Figure 7. Pharos is described in detail in [5].

Pharos is designed to select a small set of (highly relevant) sources from among a very large set. While Pharos separates the query activity (query propagation and result retrieval) from the metadata propagation, as in the previous models, it differs from them in that it synchronizes this activity with the metadata retrieval. Metadata is extracted by the source, as in the STARTS Model, but here the metadata is composed of a small amount of non-hierarchical information and a relatively large amount of hierarchical information defined by one or more *information taxonomies*. The metadata is then split into two levels: the *high-level* metadata consists of the non-hierarchical information and the top portion of the hierarchical metadata, and the *mid-level* metadata consists of the lower portion of the hierarchi-

[2] *Precision* is the fraction of returned documents that are relevant to a query.

cal metadata. There are two types of intermediaries: high-level servers and mid-level servers.[3] These levels correspond to the levels in the metadata. Both types of intermediaries contain information about all sources, although multi-source metadata records are also possible in this model. The high-level servers are massively replicated, identical copies, and designed to reside locally, similar in nature to USENET news servers. The mid-level servers, however, are only sparsely replicated (if at all), and each contains unique topic-based metadata about all sources (again, multi-source records are possible). A source propagates its metadata as needed to the relevant intermediaries. Because Pharos works with any type of information taxonomy, a source can extract non-text information about its collection and incorporate it as a standard component of its metadata.

When a query is generated, there is an iterative series of metadata retrieval. The query is first matched against the local high-level metadata and the sources are filtered to a greatly reduced subset (from, say, $\sim 10^6$ potential sources available on the network to, say, $\sim 10^3$ relevant sources for the specific query). The query generator then requests more metadata about this subset from the relevant topic-based mid-level servers, situated remotely on the network. These servers supply greater detailed information about this filtered list of sources, thus further reducing the number of potentially relevant sources (say from $\sim 10^3$ to $\sim 10^1$).

Once a small set of sources is selected, Pharos assumes that the sources are then queried directly; that is, there are no query intermediaries in Pharos as there are in the STARTS Model. However, the equivalent functionality for query result merging could be built into the query generator's user interface. There is nothing about having multiple intermediaries that precludes such activity. While Pharos does not directly address this issue, keyword text-based results can be merged using the STARTS Model methodology. For other types of documents, result merging remains a more difficult problem.

4 Model Comparison

Some of the criteria under which we might compare these different models and classes, especially from an implementation point of view, are scalability, flexibility, design complexity, and evaluation methodology. Increases in design complexity should be warranted by corresponding increases in either scalability or flexibility. The performance of these models depends on factors such as the number of queries, the number of sources, and the size of a collection. We attempt to estimate several important characteristics of each model as well as the three classes in general. We have selected several model parameters and estimated their corresponding order of magnitude values. We then use these parameters to derive equations for network traffic, storage requirements, and the number of accesses. Next, we use the sample values to check that these equations are realistic and to identify potential problems. Finally, we discuss how these characteristics are likely to change with the future growth of the Internet.

4.1 Model Parameters

Table 2 on the following page lists all model parameters used in our derivations. All values in this and the following tables are written in $\log_{10}(x)$. We first list the values used to describe quantities such as the number of documents in a collection and the number of sources. Next we list the values used to describe, in bytes, quantities such as the size of a document and the size of a query result. This table includes estimates of what might be considered a parameter's small, large, and typical order of magnitude. For example, N_D, the number of documents in a single collection, is estimated to range from 10^2 to 10^8, and a typical collection is estimated to contain 10^4 documents.[4]

Several of these parameters are specialized and need some further explanation. In Pharos, queries are based on one or more topics chosen from several information classifications, and the number of topics chosen per query, N_{QI}, affects how many mid-level intermediaries are involved in handling a query: one intermediary per topic. We differentiate S_{DT}, the size of an ASCII text document, from S_D, the size of a non-specific document, because architectures such as the BF and STARTS Model are designed for text documents; ASCII documents tend to be smaller than, for example, images.

All sizes in Table 2 after S_R are related to metadata. It is difficult to estimate these sizes; as an example, we derive two of them. For text-based vector analysis [6, 9], the size S_{CTS} of the full collection metadata at a source is a function of the number of unique words, t, in its collection, which is generally a function of the total number of words in the collection, N. For $t = f(N)$, clearly $f(N)$ is greater than

[3] Low-level servers are considered to be the sources themselves.

[4] The 1987 Annual Report of the Library of Congress lists approximately 86 million items in its collection; INSPEC contains well over a million articles.

Symbol	Small	Large	Typical	Description
N_D	2	8	4	Number of documents in a single collection
N_{RD}	1	4	2	Number of relevant documents from a single collection
N_S	2	7	5	Number of sources
N_{BS}	0	2	1	Number of 'best' sources
N_{RS}	0	5	2	Number of relevant sources (i.e. sources with any relevant documents): assumed to be $\sim N_S/500$
N_{QG}	2	8	6	Number of query generators
N_{II}	1	4	2	Number of independent intermediaries (Class 2)
N_{MLS}	1	3	2	Number of unique mid-level servers (Pharos)
N_{IM}	0	2	0	Number of duplicate mid-level servers per 'topic' (Pharos)
N_{HLS}	2	5	3	Number of high-level servers (Pharos)
N_{HLF}	1	4	3	Number of sources selected by high-level filter (Pharos)
N_{QI}	0	1	0	Number of taxonomy nodes in a Pharos query
S_D	2	8	5	Size of a non-specific document
S_{DT}	2	6	4	Size of a text (ASCII) document
S_C	4	14	9	Size of a collection of non-specific documents: $N_D * S_D$
S_{CT}	4	12	8	Size of a collection of text (ASCII) documents: $N_D * S_{DT}$
S_Q	1	3	2	Size of a query
S_M	1	4	3	Size of a query result (returned metadata record) for a single document (e.g. [title, author, publication date, keywords, abstract])
S_R	1	8	5	Size of a query result (all relevant documents) from a single source: $N_{RD} * S_M$
S_{CTS}	4	9	6	Size of full (text) collection metadata at a source (STARTS): see Equations 1–3
S_{CTI}	6	14	11	Size of full collection metadata for all sources, at an intermediary (STARTS): see Equation 4
S_{TN}	0	2	1	Size of information for each node in a taxonomy (Pharos)
S_{CPS}	4	6	5	Size of full collection metadata at a source (Pharos): S_{TN} * (nodes/taxonomy) * (taxonomy/source)
S_{HPS}	1	4	3	Size of full collection high-level metadata at a source (Pharos): extracted from $\sim$25 depth-1 nodes of the taxonomy trees
S_{MPS}	2	5	4	Size of full collection mid-level metadata at a source (Pharos): extracted from $\sim 25^2$ depth-2 nodes of the taxonomy trees
S_{HLI}	4	10	8	Size of metadata at a high-level intermediary for all sources (Pharos): $\sim S_{HPS} * N_S$
S_{MLI}	5	11	9	Size of metadata at a mid-level intermediary for all sources (Pharos): $\sim S_{MPS} * N_S$

Table 2: Model Parameters (all values are $\log_{10}(x)$; sizes are in bytes)

$\log(N)$ and less than N. Salton [9] gives $t = kN^{\beta}$, for constants k and β, $10 \leq k \leq 20$, and $0.5 \leq \beta \leq 0.6$. We take $k = 15$ and $\beta = 0.55$. Next, we must estimate N as a function of S_{CT}, the size of a collection of ASCII text documents. Salton points out that, while average word length for distinct English words is 8.1 characters, the average word length in ordinary English text (with many repeated words) is 4.7 characters. Since we are not assuming ordinary English text necessarily (or even English, let alone text), and we want to allow for multi-word *terms*, we take the average number of bytes per term, L, as 6. Finally, assuming that each term requires at least three 4-byte numbers in the metadata[5], we derive the following estimate:

$$\begin{aligned} S_{CTS} &\gtrsim (12 + L) * t && (1) \\ &\approx [((12 + L) * k)/L^{\beta}] * (S_{CT})^{\beta} && (2) \\ &\approx 100 * (S_{CT})^{0.55} && (3) \end{aligned}$$

with $t = k * N^{\beta}$ and $N = S_{CT}/L$.

An intermediary is required to maintain this metadata for each source, regardless of whether the terms from a source have already appeared in an intermediary's global list of unique terms. Thus

$$S_{CTI} \approx S_{CTS} * N_S \tag{4}$$

where S_{CTI} is the size of metadata at an intermediary and N_S is the number of sources from which the intermediary has collected information.

4.2 Analysis

Estimates for several characteristics of each model are given in Estimate Tables 3–6, where all expressions are derived from the model parameters described in the previous section. Values are derived by applying the constant 'typical' values in Table 2 to the expressions in each Estimate Table. Any column which is not applicable to the corresponding model is marked 'N/A'; such is the case, for example, of metadata quantities for the Simple Model, which has no metadata. This is different from columns which potentially apply to a model but for which the particular model has a zero quantity, marked '-none-'.

Table 3 on the next page gives estimates for remote network traffic generated by a single query and by a single update. We assume that network traffic is sent efficiently (e.g. text compression is ~50%), but this does not greatly affect order-of-magnitude estimates. We specify *remote* traffic because we do not include in these estimates any traffic assumed to travel between machines within the same local area. This excludes, for example, traffic in the Pharos Model between a query generator and its corresponding local high-level metadata server. Table 4 gives estimates for long-term metadata storage requirements. Metadata storage is required at the query generator in the Local Model only. However, Pharos stores some metadata at high-level servers within a query generator's local area; there are a total of N_{HLS} such servers, and this storage requirement is listed as residing at the query generator. In our numerical estimates, we assume that each Class 2 intermediary retrieve metadata from every source. In reality, some intermediaries collect more than others, and as a result, many queries are propagated to multiple intermediaries by users trying to find as good information as is practical. The amount of duplicate query generation is difficult to estimate and we do not attempt to do so. Tables 5 and 6 give the total number of accesses generated assuming that each query generator issues a single query, and each source issues a single update, respectively. Thus the total number of queries and updates are determined by the number of query generators, N_{QG}, and sources, N_S, respectively.

As an example, we derive the Single Query Expression for the STARTS Model in Table 3. The query, of size S_Q, is first propagated to the intermediary, which matches it against its metadata and selects the best sources to query directly. These N_{BS} sources are then handed the query. So far the query has generated $S_Q * (1 + N_{BS})$ bytes of remote network traffic. The next action is the result retrieval process. Each queried source will send back its own query result assumed to be of size S_R. The intermediary in STARTS will perform only limited result merging: rank merging, but no duplicate detection. Thus, the same amount of traffic sent to the intermediary is bundled together and re-sent to the query generator, for a total result retrieval of $(2 * S_R * N_{BS})$ bytes. While one might consider the factor of 2 to be superfluous, it is negligible in the order-of-magnitude estimate anyway. Thus, the total amount of traffic generated by a single query in the STARTS Model is $(S_Q * (1 + N_{BS})) + (2 * S_R * N_{BS})$ bytes, as shown in the table. In fact, in all the models, traffic caused by queries is dominated by multiplicative factors of S_R, with the best being $S_R * N_{BS}$ in STARTS and Pharos.

[5] The metadata may require, for example, the number of documents in which each term occurs, the total number of occurrences of the term in the entire collection, fields in which the word occurs, etc.

Model	Single Query		Single Update		Comments
	Expression	Typical Value	Expression	Typical Value	
Simple	$S_Q + S_R$	5	N/A	N/A	No metadata, single-source query only
Remote	$N_S * (S_Q + S_R)$	10	N/A	N/A	No metadata
Local	$N_{BS} * (S_Q + S_R)$	6	$N_{QG} * S_{CTS}$	12	
BF	$S_Q + (N_{RS} * S_R)$	7	$N_{II} * S_{CT}$	10	Query result assumes no duplicate detection
STARTS	$(S_Q * (1 + N_{BS})) + (2 * S_R * N_{BS})$	6	$N_{II} * S_{CTS}$	8	
Pharos	$(2 * N_{HLF} * N_{QI} * S_{TN}) + (N_{BS} * (S_Q + S_R))$	6	$(N_{HLS} * S_{HPS}) + (N_{IM} * S_{MPS})$	6	

Table 3: Estimates of Remote Network Traffic (in bytes; all values are $\log_{10}(x)$)

Model	Combined Query Generators	Combined Intermediaries	Typical Values		Comments
			Average per Storage Site	Total	
Simple	-none-	N/A	-none-	-none-	No metadata, single-source query only
Remote	-none-	N/A	-none-	-none-	No metadata
Local	$N_{QG} * S_{CTI}$	N/A	11	17	Assumes STARTS-style metadata (i.e. text only)
BF	-none-	$N_{II} * (N_D * N_S * S_M)$	11	13	Assumes smaller S_M
STARTS	-none-	$N_{II} * S_{CTI}$	11	13	
Pharos	$N_{HLS} * S_{HLI}$	$N_{IM} * S_{MLI}$	8	11	

Table 4: Estimates of Metadata Storage Requirements (in bytes; all values are $\log_{10}(x)$)

Model	Combined (Remote) Intermediaries	Combined Sources	Typical Values			Comments
			Average per		Total	
			Intermediary	Source		
Simple	N/A	N_{QG}	N/A	6	6	Single-source query only
Remote	N/A	$N_{QG} * N_S$	N/A	6	11	
Local	N/A	$N_{QG} * N_{BS}$	N/A	2	7	
BF	N_{QG}	$(N_{QG} * N_{RS} * N_{RD})/1000$	4	2	7	Assumes relevance check of top 0.1% documents
STARTS	N_{QG}	$N_{QG} * N_{BS}$	4	2	7	
Pharos	$N_{QI} * N_{QG}$	$N_{QG} * N_{BS}$	4	2	7	

Table 5: Estimates of Query Accesses (one query per generator; all values are $\log_{10}(x)$)

Model	Combined (Remote) Intermediaries	Typical Values		Comments
		Average per Intermediary	Total	
Simple	N/A	N/A	N/A	No metadata
Remote	N/A	N/A	N/A	No metadata
Local	$N_{QG} * N_S$	5	11	Query generator is intermediary
BF	$N_D * N_{II} * N_S$	9	11	
STARTS	$N_{II} * N_S$	5	7	
Pharos	$(N_{HLS} * N_S) +$ $(N_{MLS} * N_{IM} * N_S)$	5	8	

Table 6: Estimates of Update Accesses (one update per source; all values are $\log_{10}(x)$)

4.3 Scalability Estimates

We first discuss our estimates assuming the parameters take on values that might be currently considered typical, as was listed in Table 2. After that, we increase three key values and re-compare some of the models.

4.3.1 Typical Current Values

We intuitively expect that Class 1 should show the greatest scalability problems. Each model in this class breaks down in a different way. Clearly the Simple Model is not feasible for many technical (let alone political) reasons. If the characteristics we are estimating are the most relevant ones, we expect that some of these reasons will show up in one or more of the Estimate Tables 3–6. The Simple Model does not seem to show a problem in the amount of remote network traffic shown in Table 3. This is unrealistic because, for example, if we assume that all documents on the network were placed at one monolithic source, the query result size would likely be substantially larger than for more typical sources. In Table 3, though, we assume the same result size for all queries. Table 5, however, shows that sources in both the Simple and Remote models receive orders of magnitude more query accesses than in the other models because each query is propagated to every source. The last model in this class, the Local Model, shows problems with, for example, the amount of traffic needed to perform a single update. The model also does poorly in terms of storage and total update accesses. Total storage is a problem because the model must assume that each query generator has its own copy of all metadata in order to be in Class 1.

The models discussed in Class 2 are the closest to present working systems. It is important to distinguish behavior evidenced in a particular model from behavior inherent in the class. Both BF and STARTS, for example, have high storage requirements, as seen in Table 4. However, this is a result of the fact that the size of the collection metadata used in these models is a function of the size of the collection. Pharos uses a classification-based metadata structure whose size is independent of the collection size, and thus yields smaller storage values. Such a metadata structure is not inconsistent with Class 2. The BF Model is particularly problematic in the number of update accesses, shown in Table 6. This is because the model requires that the intermediaries separately fetch each document from a source rather than fetching a single, albeit large, collection metadata record as in the STARTS Model. STARTS achieves the reasonable goal that each intermediary is accessed no more than once per update. If the BF Model does not rank the documents in its result set, users must fetch many documents in order to determine their relevancy. Even with a ranked list, however, a lack of document duplicate detection would require needless document fetches. As mentioned in Section 3.2, we expect that STARTS provides better precision and more document metadata than BF, and therefore results in even less document fetches. We account for this in our estimates by assuming that the top 0.1% of the documents returned by an intermediary in the BF Model is checked for relevance, and therefore included in the query access count in Table 5.

One of the advantages of Class 3 in general is that it allows intermediary metadata to be less than linear with respect to the number of sources; this is not possible in Class 1 or 2. Several resource discovery models [3, 5, 6, 10], including Pharos, allow intermediaries to collect information from other intermediaries, including metadata describing a collection of *collections*, rather than just a collection of *documents*. An appropriate network hierarchy of intermediaries could arbitrarily reduce the size of the metadata, at the cost of loss of discrimination between different sources. Furthermore, grouping sources behind par-

Figure 8 Scalability Issues

- Increasing; Degrades Performance:
  - Number of Users
  - Number of Sources
  - Data Volume
- Increasing; Improves Performance:
  - Network Bandwidth
  - Storage Capacity
  - Processing Power
- Constant; Design Criteria:
  - Query Response Time
  - Query Result Size

ticular intermediaries reduces the number of update accesses per intermediary, at the cost of maintaining more of them.

The only Class 3 model we have included is Pharos. Pharos performs well in the estimates for several reasons. Network traffic is minimized for updates for the same reason that storage is minimized; as stated, the metadata size is independent of the collection sizes. Pharos stratifies metadata within a collection. Because of this layering, it can send pieces of metadata selectively to different intermediaries; this allows a large number of sources to be coarsely described in a compact way. This type of metadata splitting is possible only if the information itself is hierarchically structured. It can be utilized only within Class 3, since its benefit arises from having intermediaries hierarchically arranged in a way that matches the hierarchy within which a collection is classified.

4.3.2 Typical Future Values

We have chosen typical values for our estimates based more or less on the current (1996) Internet environment. It is generally expected that several parameters will experience fairly rapid growth over the next few years, some degrading system performance and others improving it, as shown in Figure 8. However, we expect that factors relating to computer-human interaction cannot drastically change if a resource discovery system is to continue to be useful. In particular, query response time and the result size need to remain fairly constant. While we do not attempt here to directly model query response time, it is likely to be greatly affected by factors such as the network traffic and number of accesses required by each model. The query result size is directly related to the amount of network traffic generated by a single query, as shown in Table 3. Therefore, it is important to recognize which models are likely to experience future scalability problems with these values based on parameter dependencies. We ignore the Class 1 models in this section since they were already problematic with the previous parameter values.

First, we change the three model parameters which most closely relate to the number of users, the number of sources, and the data volume: N_{QG}, N_S, and N_D, respectively. We increase the first two by two orders of magnitude and the third by one, yielding approximately 10^8 query generators, 10^7 sources, and an average of 10^5 documents per collection. These changes require changes in seven other parameters: N_{RS}, S_C, S_{CT}, S_{CTS}, S_{CTI}, S_{HLI}, and S_{MLI}.

We now allow for two cases of the models to handle these increases. In the first, Case 1, we keep the number of intermediaries the same; in the second, Case 2 we increase them. Increasing the number of intermediaries in Class 2 is a simple matter of increasing N_{II}, which we do by two orders of magnitude. For Pharos, we have two coordinated sets of intermediaries. The mid-level servers are the most similar to the intermediaries in the other models, and therefore for Case 2 we increase N_{IM} by two orders of magnitude. The high-level servers, however, act more like USENET news servers and are expected to serve a local area network. Moreover, any replication of this type of service within a local area is irrelevant to our model, which is looking only at remote traffic. Therefore, we increase the number of high-level servers, N_{HLS}, only one order of magnitude; that is, we roughly expect that the increase in the number of high-level servers grows approximately as the square root of the growth of the number of users.

The results are shown in Table 7. This table shows the re-computed estimates of all values in Tables 3–6 as a result of the increases in N_{QG}, N_S, and N_D. For each model, the column labeled 'Original' shows the values from the original tables. The column labeled 'Case 1' shows the Case 1 values, where the number of intermediaries is left as it was. The column labeled 'Case 2' shows the values for which the number of intermediaries was increased as previously discussed. Certain scalability problems become evident in this table. For example, because the metadata in BF and STARTS grows with the size of the collection as well as with the number of intermediaries, the amount of traffic generated by a single update grows by as much as three orders of magnitude. In Pharos, the increase is dominated by the extra number of high-

Estimate	BF			STARTS			Pharos		
		New Values			New Values			New Values	
	Original	Case 1	Case 2	Original	Case 1	Case 2	Original	Case 1	Case 2
Network Traffic (Table 3):									
Query	7	9	9	6	6	6	6	6	6
Update	10	11	13	8	9	11	6	6	7
Metadata Storage (Table 4):									
Avg	11	14	14	11	14	14	8	10	10
Total	13	16	18	13	16	18	11	13	14
Query Accesses (Table 5):									
Avg Int	4	6	4	4	6	4	4	6	4
Avg Src	2	4	4	2	2	2	2	2	2
Total	7	11	11	7	9	9	7	9	9
Update Accesses (Table 6):									
Avg Int	9	12	12	5	7	7	5	7	7
Total	11	14	16	7	9	11	8	10	11

Table 7: Estimates for Large Parameters (all values are $\log_{10}(x)$)

level servers, and so the result is much smaller: only one order of magnitude. Pharos also scales well with respect to storage; this is not only because, as before, the metadata size per source is constant, but also because only a portion of it is sent to any particular intermediary. Finally, STARTS scales the best for the number of update accesses, since it requires only one access between each source and intermediary, and has relatively few intermediaries.

5 Conclusion

We have presented a classification of network architectures for locating information sources. These models have been grouped into three broad classes: Direct, Single Intermediary, and Multiple Intermediaries. We discussed the relevant parameters for these models, derived estimates for several of their characteristics, and compared them based on these estimates. Each class has been shown to have certain scalability characteristics for all its models. We believe that each increase in design complexity among the classes and models is warranted by the corresponding increase in scalability. In general, the estimates we derived indicate the need for collection metadata whose size is not a function of the size of the collection. Moreover, utilizing multiple intermediary models that rely on metadata summaries can avoid a linear growth of metadata size with respect to the number of sources. Limiting the growth of collection metadata in these two ways should more easily accommodate the expected expansion of the Internet.

More work needs to be done in better establishing which scalability factors relate to the general classes, and in further classifying the architectures in more detail, including, for example, metadata structure. In the future, we hope to compare these models in terms of flexibility, such as query methodology, data diversity, source selection criteria, and so on. We are currently working on simulating these systems in order to more realistically compare them. This work includes developing an evaluation methodology which is consistent between all models, given the different query methodologies and results between them.

References

[1] AltaVista, http://www.altavista.digital.com/, 1996.

[2] Berry, M.W. & Dumais, S., "Using Linear Algebra for Intelligent Information Retrieval," SIAM Review, December 1995; and http://www.cs.utk.edu/~library/TechReports-/1994/ut-cs-94-270.ps.Z.

[3] Bowman, C.M.; Danzig, P.B.; Hardy, D.R.; Manber, U.; and others, "The Harvest Information Discovery and Access System", Proceedings of the Second International World-Wide Web Conference, pp. 763-771, Chicago, Illinois, October 1994; and http://harvest.cs.colorado.edu/.

[4] ConQuest Software, Inc., "ConQuest Guide for Developers" (Proprietary), Version 4.0, Columbia, MD, 1994.

[5] Dolin, R.; Agrawal, D.; Dillon, L.; El Abbadi, A.; "Pharos: A Scalable Distributed Ar-

chitecture for Locating Heterogeneous Information Sources", Technical Report TRCS96-05, Computer Science Department, University of California, Santa Barbara, 1996; and http://www.cs.ucsb.edu/TRs/TRCS96-05.ps.

[6] Gravano, L. & García-Molina, H., "Generalizing *GlOSS* to Vector-Space Databases and Broker Hierarchies", Proceedings of the 21st VLDB Conference, Zurich, Switzerland, 1995.

[7] Gravano, L.; Chang, K.; García-Molina, H.; Paepcke, A., "STARTS: Stanford Protocol Proposal for Internet Retrieval and Search", http://www-db.stanford.edu/~gravano-/starts.html, 1996.

[8] Lycos, http://www.lycos.com/, 1996.

[9] Salton, G., *Automatic Text Processing*, Addison-Wesley Publishing Company, Inc., Reading, MA, 1989.

[10] Sheldon, M.; Duda, A.; Weiss, R.; O'Toole, J.; Gifford, D., "Content Routing for Distributed Information Servers", Proceedings of the 4th International Conference on Extending Database Technology, 1994.

[11] Yahoo FAQ, http://www.yahoo.com/docs/info-/faq.html, 1996.