QUERYING A NETWORK OF AUTONOMOUS DATABASES

Patricia Simpson
Rafael Alonso

CS-TR-202-89

January 1989

QUERYING A NETWORK OF AUTONOMOUS DATABASES

Patricia Simpson
Rafael Alonso

Department of Computer Science
Princeton University
Princeton, New Jersey 08544

Abstract

We consider the problem of enabling read-only access to a very large number of independent databases over a public network without compromising the autonomy or heterogeneity of the participants, both databases and queriers, who may number in the thousands or millions. From an examination of the network environment and its characteristics we derive general principles for designing applications involving autonomous systems, including statelessness, decentralization of control, and simple communication standards. Applying these principles, an architecture is developed for information exchange among diverse database systems. This architecture preserves autonomy by separating the functions of query interpretation, database discovery, and user interface into modules independently executable by any participant or subgroup of participants at any time. Subproblems arising in these three areas are discussed in detail, and solutions are presented.

Keywords: *databases, heterogeneity, autonomy, information retrieval, network applications*

January 19, 1989

QUERYING A NETWORK OF AUTONOMOUS DATABASES †

Patricia Simpson
Rafael Alonso

Department of Computer Science
Princeton University
Princeton, New Jersey 08544

1. Introduction

Over the next several years a computing environment will evolve which, while a natural extension of present-day technology, will be qualitatively different from existing environments. Its origins can be seen in three technological/societal trends: the personalization of computing via home- and office-based computers (PCs), the advent of the Integrated Services Digital Network (ISDN), and the ever-increasing quantity and variety of information stored primarily in digital rather than printed form. The result will be a vast network of independent systems, comparable in size to the present telephone network, in which any node can communicate with any other over a universally accessible carrier. Together these elements constitute an infrastructure for the next stage of our "information society".

With regard to the communication network itself (the carrier) we will assume only that it provides the lowest three levels of the ISO OSI reference model (the physical, data link, and network layers) and supports (at least) one-way connectionless communication. We make no particular assumptions about bandwidth or speed. Every node is assumed to have a unique network address. A message is sent to a node by handing its address and the message together to the carrier, which recognizes the address and routes the message to the recipient. Messages sent between two nodes do not pass through any others. The logical network, therefore, is fully connected and point-to-point, regardless of the topology of the physical network.

The nodes of this network already exist; they are autonomous computer systems owned by organizations and individuals who use them them to, among other things, store and manipulate information. Nodes may be heterogeneous in all respects — hardware, software,

† This research is supported by New Jersey Governor's Commission on Science and Technology Award No. 85-990660-6, and grants from IBM and SRI's Sarnoff Laboratory.

storage capacity, file structure, operating systems, database systems, peripheral devices, and the needs and activities of their owners. Many separate applications will exist on the network, and a given node may participate in any number of these. Applications are developed and implemented by the participants themselves.

Most applications developed for this environment will be extensions of communication and information-sharing activities currently engaged in more or less manually. These may include mail, voice mail, bulletin boards, publishing, education, entertainment, cooperative work, conferencing, dissemination of news and public service information, security and health monitoring, advertising, and commercial transactions, among others. One promising application is *information exchange* (IE), or topic-based querying of information located throughout the network.

In this paper we will describe an architecture for the information exchange application but, as few applications exist for this environment, it will be useful first to discuss the nature of the network and derive general principles for application design. This we do in the next section. In Section 3 we will apply these principles to the IE problem, outlining the issues specific to that application. Section 4 briefly presents the structure of a proposed architecture for IE, and Sections 5, 6, and 7 describe in greater detail solutions for the component problems of query expression and processing, node discovery, and querier interface, respectively.

2. Network Application Design Principles

The distinguishing characteristics of the network are its large scale and the autonomy and heterogeneity of its component computer systems. While some distributed computing research has dealt with these characteristics in isolation, when taken together they yield novel, even surprising, implications for application development which we now briefly discuss:

Application independence. The autonomy of all nodes must be preserved. Autonomy can be taken to mean freedom from any sort of control by other nodes and freedom to define as much as possible one's own view of the network and one's means of interaction with it. No node can require another to participate in a particular application or to behave in a particular way within it, and a node may enter or leave an application at any time. Logical topology will therefore vary among applications, evolving continually with the changing needs of participants. It is important that applications be kept separate from (independent of) each other and that a clean separation also exist between each application and the carrier so that new applications may arise and evolve without the need to coordinate these changes with the carrier.

Bottom-up development and operation. Applications should be designed to enable new nodes to join easily and independently, without disrupting the activity of other participants. Bottom-up mechanisms are therefore necessary for *discovering* other nodes as needed

and for gradually increasing one's level of participation in applications. This is in distinct contrast to conventional distributed system design, where an application is first developed and then distributed over known, pre-existing nodes. We will examine a specific instance of the node discovery problem in detail in our discussion of information exchange.

Stateless interaction. Applications requiring participants to keep track of the state of their interaction will not scale when many nodes participate in many applications. In addition, since autonomous nodes cannot be *required* to answer or acknowledge any particular message, a node cannot rely upon another to engage in multi-stage interaction. This implies a rather severe restriction, that communication consist of one-way, self-contained messages — analogous to letters posted through the mail. This constraint may be relaxed by agreement among nodes willing to forego absolute autonomy in exchange for a more sophisticated level of interaction; but, in general, applications should be designed to run statelessly if possible.

Decentralized control. In conventional distributed systems "decentralized control" is generally taken to mean control by more than one node, usually via some cooperative protocol such as token-passing or voting. Among autonomous entities, however, the notion must be taken a step further to preclude cooperation in the usual sense since cooperative algorithms require all parties to maintain state. Needless to say, the existence of reliable global information cannot be assumed. True decentralized control also precludes a reliance on specialized nodes such as name servers. Servers may exist, but if an application *depends* on their existence the autonomy of its participants will be reduced.

Decentralization is perhaps the most novel implication of autonomous networks. The application designer is not at liberty to build yet another *system*, using resources appropriated from the nodes in a synchronized or cooperative way, but merely mechanisms to enable communication *among* separate systems that intend to remain separate. Each node maintains locally all of the programs and data structures it needs for the application, so that never is the cooperation of any other particular node required.

Separation of function. The requirements of scale and heterogeneity dictate that each node retain responsibility for defining its own owner-to-application interface. This allows the interface to be customized by and for each participant, but requires that communication standards among nodes be simple and flexible. In the information exchange application, for example, the query entry interface will reside at the querying node and database access will be performed by the database owner. The communication stream between the two is a thin one, consisting of only simple, compact messages. Applications should be structured so as to permit separate functions to be executed as separate independent modules; this will maximize autonomous operation and minimize message traffic on the network.

In short, this is a very different environment than that of "classical" distributed systems, offering new potential and posing new challenges.

2.1. Related Work

Research in several different areas has addressed problems related to information exchange. Heterogeneous distributed database management systems promote interoperability of pre-existing databases, generally by constructing a global schema acceptable to all participants. Thus participants sacrifice autonomy and changing or expanding the system entails substantial effort and cooperation. Federated database systems [Heimbigner85] support negotiable autonomy but require a homogeneous data model and a centrally administered federation dictionary. Neither approach scales well to large numbers of participants, especially those participating in multiple DDBs or federations.

Several attempts have been made to provide a unified query interface to multiple retrieval systems [Marcus81, Moulinoux83, Williams86], including in some cases assistance in selecting the system to which queries are directed. In every case the interface is uncustomizable; users still must subscribe to the service and learn its query language. Adding new sources is difficult, which limits systems to only a few large databases.

The idea of implementing retrieval enhancements on queriers' own processors was introduced in [Jamieson79]. In this architecture customization is possible but, as queriers retain the burden of communicating in the host database query language, this does not scale to multiple source systems. The telesophy system *TSI* [Schatz87] presents a friendly query/browsing interface to multimedia information located at multiple sites but imposes centralized indexing and a homogeneous retrieval model, making no provision for acquiring information from autonomous sources.

The resource discovery problem was discussed in [Schwartz88] and an architecture was presented that is suitable for a large number of participants. It relies, however, on a hierarchy of specialized agents and coordinated behavior among them. Attribute-based naming likewise scales well only in a centralized, or at least hierarchical, environment. The information exchange architecture that we will present here is the first proposed design that allows an arbitrarily large number of truly heterogeneous systems to share data without sacrificing their autonomy.

3. Information Exchange

Information exchange may be thought of as the extension of traditional information retrieval (IR) — wherein many queriers access a single information source via a query interface defined by that source — to a broader environment of many independent systems, most of which have some shareable information and all of which may play the role of both querier and source at various times. Whereas the component databases may use different data models, data types, and access methods, the IE application is concerned with *text* information and supports access by *topic* — that is, by its subject matter or content. This allows us to draw on the results of nearly three decades of IR research and experience in text retrieval,

perhaps laying some groundwork for more advanced applications. The unit of retrieval is the *document*, a human-readable string of text. A query is an expression of desired document content, and a response to a query consists of one or more documents that are deemed *relevant* to the topics expressed in the query.

The aim of IE is to enable automated access to all computerized sources. As such, it bridges the gap between IR, characterized by high-quality automated access to a small number of isolated information bases, and manual information seeking, characterized by haphazard access to a rich diversity of resources. IE acknowledges the natural diversity, inconsistency, and redundancy inherent in the information of a society, and so makes no attempt to guarantee the consistency or semantic quality of responses. Information retrieval, whether manual or automated, is always an *inexact* process [Kraft85], a point that must be borne in mind throughout this discussion. Because a query is only an approximate expression of an information need, and the text of a document an approximation of its semantic content, the retrieval process is approximate at best.

3.1. Issues

In this section we identify and discuss some of the issues that arise in the context of the information exchange application. We cover basic concepts here, and leave a detailed discussion of problem solutions to the remaining sections of the paper.

Common language. The need for stateless interaction suggests a "one-shot" question/answer protocol between nodes. That is, rather than navigational queries (browsing through a source's information) or iterative queries (repeated query refinement), queries must be self-contained, each expressing a complete information request. † Sources may then respond to any query without regard to previous responses, order of receipt of queries, or prior knowledge about the querier. To achieve this, a mutually understood query language is needed that is both expressive enough to convey precise information requests and simple enough to be manageable by many different kinds of source and querier systems.

While at first it appears impossible to search the network without a globally defined list of acceptable or recognized topics represented in the network, there is in fact a pre-existing body of topic identifiers that can be used: the words and phrases of natural language. Natural language is, after all, what most documents are composed of and it is the only language which all participants can agree represents information. Basing the common query language on ordinary English words will allow maximum expressiveness without necessitating a global dictionary or placing artificial limits on participants' vocabularies. (Note that IE does not involve natural language *understanding* at all, but will accept arbitrary words in queries and

† Browsing and query refinement need not be completely sacrificed, as both can be performed at the querying site once a body of information has been retrieved. Remote retrieval, however, should be stateless.

treat them as character strings for the purposes of indexing and searching).

Querying and Translation. The functions of query formulation and query processing should be separated, so that the querier does not *directly* process queries on other nodes, but can only *ask* that they be processed. Each querier should take responsibility for producing "good" queries that express the request accurately. Each source should likewise produce the best response it can, given a particular query. Nodes vary greatly in their abilities; by keeping communication standards simple and allowing participants to define their own roles as much as possible, nodes remain free to evolve and upgrade independently of other application participants. The quality of interaction between any two nodes, then, is determined by their own capabilities.

Because ease of participation is an important objective of this application, it is undesirable to require sources to modify their data storage structures or access mechanisms in order to participate. We therefore would like to ensure that it is possible for any source to *translate* incoming queries into its own database query language, so that they may then be executed via the existing query interface. Translation is completely under the control of the queried node, as is the decision of whether to respond at all. (An autonomous source may certainly choose to be "invisible" to certain queriers, or at certain times, by ignoring queries it receives.) In Section 5.2 we will discuss translation by both text-oriented and non-text databases.

Querier interface. Within a network of thousands of nodes, the ability of each participant to see a *unified* view of the network is important. A drawback of present retrieval systems is that each has its own command and/or query language, vocabulary, signon procedures, storage structures, and so forth, all of which must be learned by each user before interaction can begin. This is part of the reason that most people never access any computerized information service at all. However, a unified view does not imply standardizing query languages and storage structures across the entire network — just the opposite. By keeping interaction as simple as possible, queriers should be able to *customize* their query interfaces to individual needs and preferences, while sources retain their existing access mechanisms, merely adding a software layer to interpret incoming queries. Ideally one should be able to query external sources via the same user interface used to access locally held information, and to integrate document contents into the existing local storage system as responses are received. These goals will be attainable to greater or lesser degrees, depending on the specific capabilities of each querying node.

External indexing. In the case of IE the node discovery problem entails learning what sources exist, where they are located, and what kind of information each provides. Queriers need this knowledge in order to direct queries to appropriate sources. We call this the *external indexing* problem, as it is essentially a matter of indexing, by topic, a body of information which is not held or controlled by the indexer. It entails the bootstrapping problem of discovering unknown network addresses in the absence of location servers of any type as well as the

performance issue of choosing only good sources to query. It is clearly not feasible for a querier to publish a query throughout the entire network in an attempt to find sources capable of responding to it; the network would be flooded with messages and sources would be inundated with queries they could not answer. Nor is a network-wide global index admissible, as global cooperation would be required to build and maintain it.

We can model a solution for the external indexing problem on human communication practices. Everyone knows some set of information resources (people, publications, and organizations) and has a rough idea of what topics each is knowledgeable about. When I have a question which my known sources cannot answer, I ask them to recommend other sources; that is, I ask them who *they* know. In this way my source set enlarges and I can expect eventually to learn of someone capable of answering my question. By remembering their areas of expertise, I can direct questions only to sources likely to be able to answer them. Nodes can build personalized external indexes in much the same manner.

Perhaps the most difficult aspect of designing an IE application involves decentralizing control. Information must be searchable without a consistent global storage structure. Topics must be expressible without an agreed-upon classification system or vocabulary. Sources must be discoverable without a global directory or reliable servers. The architecture described in the following section operates decentrally and satisfies the goals and constraints we have discussed.

4. Architecture for an Information Exchange Network

Our architecture stresses autonomy by separating the functions of indexing, query generation, and query processing into independent activities carried out by individual nodes at their own initiative. The software components needed to perform these functions are shown in Figure 1. Note that, although the activities of querier and source are shown at separate nodes, any node may in reality incorporate all the components so that it may play either role.

Queries are composed locally (at the querying site) with the help of local software, and converted into a simple common query language for transmission to one or more source sites. We will refer to this common language as *query normal form*, or QNF. In this way queries can be generated autonomously, and the querier need not learn the command language, query language, or file structures of the sources it consults. In Section 5 we will discuss the format of queries and of responses.

The querier is also responsible for determining which sources are to be queried, using a locally held partial index of external resources. This index is built up gradually, as the need arises, and consists of network addresses of source nodes together with condensed descriptions of the information each holds. We will describe a decentralized mechanism for external indexing in Section 6, and in Section 7 present some ideas for creating customized user interfaces to the indexing and querying functions.

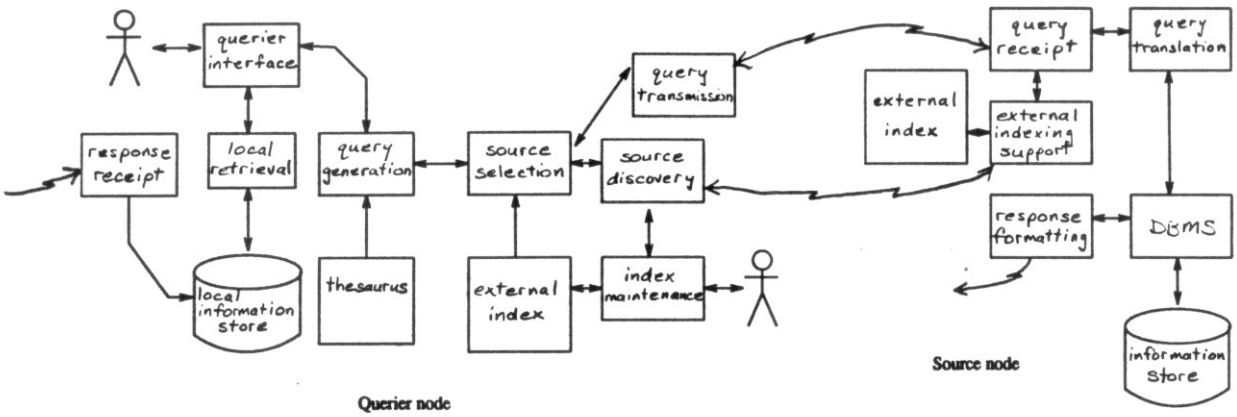

Figure 1. Information exchange modules

A source's role in the external indexing function is to make available to potential queriers a *description*, or summary, of its sharable information (but not the information itself). We call this item a *node descriptor*, and propose a format for it that is compatible with QNF; this allows a querier to search its external index for descriptors that "match" an intended query.

Source nodes are responsible for processing queries, and so must be capable of interpreting query normal form. A preprocessing operation translates each QNF query into one or more queries in the host language, which are then executed. The resulting responses should be returned to the querier in decreasing order of relevance (as determined by the source). Some postprocessing may be necessary, first to format the responses as human-readable documents, then to relevance-rank them. Translation and response formatting are discussed in the next section.

5. Queries and Responses

Conventional IR is based on the assumption that each document is indexed under a set of *terms* (generally natural-language words or phrases) drawn from a vocabulary of index terms. This set is called a document descriptor. Boolean IR systems accept queries consisting of terms joined by the Boolean connectives AND, OR and NOT, such as (*sculpture* OR (*art* AND (*stone* OR *marble*))) AND NOT *Italy*. Each term retrieves the set of documents indexed by it, and these sets are combined as requested to yield a set of documents matching the query.

Term-vector IR systems accept queries consisting of a set of terms, with no connectives. A *similarity measure* is employed to determine, for each document in the collection, its closeness to the query, so that the response consists of a list of all documents, ordered by

decreasing similarity. There are many similarity measures in use [Salton81]. Perhaps the most straightforward is the vector product; queries and documents are viewed as vectors in T-space where T is the number of index terms in the vocabulary. Vector elements are taken to be 1 if a term is present and 0 if absent. Documents are in this case ranked by the number of terms held in common with the query. Alternatively, *weighted* terms may be permitted in document descriptors, queries, or both, indicating varying presence or importance of terms. The vector product then ranks documents by a weighted combination of their terms' presence and importance.

Since the quantity and quality of information on a topic will vary greatly among sources, it is important that queriers receive ranked responses. If many responses are generated, the better ones can then be examined first. This facility requires weights, and as the combination of weighted terms with Boolean connectives entails subtle complexities and even anomalies of interpretation [Bookstein81], we will use the weighted term set as the basis of query normal form.

5.1. Query Normal Form

A QNF query is of the form <T,N,W> where T = $\{t_1/w_1 , \ldots , t_n/w_n\}$ is a set of weighted terms, N is a positive integer indicating the maximum number of documents desired, and W is the minimum acceptable weight of a document, where a *document weight* is taken to be the sum of the weights of the query terms under which the document is indexed. Weights are rational numbers in the interval (0,1]. The constraints N and W combine with the term weights to give the querier some control over the ordering and quantity of the retrieved documents that will be returned. By incorporating control information (N, W, and weights) into the query itself, some of the benefits of interactive access can be attained in a noninteractive environment. A QNF query includes an upper constraint on quantity, a lower constraint on quality, and a means of ranking responses.

It should be stressed that the querier does not have knowledge of, or control over, the actual retrieval mechanisms used. A particular source may or may not be able to meet the W constraint or rank documents as accurately or as finely as the querier intends, simple though these requirements are. For this reason it seems futile to require more elaborate queries; the greatest common denominator of query languages across such a diversity of systems is low and will likely remain so. The only feature common to queries in all text retrieval systems is the use of natural language terms, and even then indexing vocabularies may differ widely among sources (and queriers). This does not appear to be a problem, however. IR systems that process natural language queries have been successfully implemented [Bernstein84, Doszkocs83], and two provisions of our architecture to be discussed later — the use of thesauri in query generation and translation, and the use of node descriptors to select sources — will mitigate vocabulary incompatibility problems.

5.2. Translation

It is not possible to describe query translation in all its details, since there may be as many different query translators as there are sources. It is possible, however, to develop general translation procedures for certain well-defined classes of source system. Standard IR systems, designed specifically for text retrieval, are well-suited to QNF query processing. In [Simpson88] algorithms were presented which accurately translate QNF for processing by Boolean retrieval systems and by term-vector systems that compute the vector product, using either binary or weighted document descriptors.

Non-text-oriented systems present greater difficulty. The first task is to define the unit of retrieval, in the absence of documents per se. For example the IE interface to a relational DBMS might define a tuple to be a "document" for purposes of responding to external queries. The semantic content of a tuple, however, is not completely contained within its attribute values; much of it is hidden in attribute and relation identifiers, in the data dictionary's descriptions of those identifiers and of data types, in the ways that local programs use the data, and in written documentation external to the database. In this environment the relational source must compose a meaningful document from each response tuple before transmitting it to a remote querier. This might be done by defining for each relation a *template* into which tuple values may be inserted. The completed template is human-readable and can be treated as a document. For example, the tuple:

sched_flight:

flno	freq	orig	dest	1cap	Bcap	Ccap	dep	arr	m	remarks
117	*G*	*EWR*	*LAX*	*12*	*0*	*132*	*1105*	*1330*	*1*	*none*

may be converted to the document:

> "Flight 117, operating daily except Tues. and Wed., is scheduled to depart from Newark Airport, Newark, NJ at 11:05 a.m. local time and arrive at Los Angeles International Airport, Los Angeles, CA at 1:30 p.m. local time. This flight carries 12 passengers in first class, none in business class, and 132 in coach seating. One meal is served."

Standard DBMSs are ill-equipped to perform retrieval-by-similarity and ranking. Their operation is akin to that of Boolean IR systems in that they retrieve a precisely specified set of items, with the added complications of multiple files (relations), named fields (attributes), and "hidden" semantics as mentioned above. We are investigating several methods for converting QNF to a relational query language such as SQL.

The first step in translation is the conversion of query terms to identifiers recognized by the DBMS. A simple method is to construct a priori an inverted index or thesaurus associating relation names and attribute names with acceptable synonyms. For example, if a source

whose database schema consists of the relations

sched_flight: flno , freq , orig , dest , 1cap , Bcap , Ccap , dep , arr , m , remarks
equipment: plno , flno , date , cond

receives the query

{ < movie/w_1 , flight/w_2 , from/w_3 , Newark/w_4 , to/w_5 , L.A./w_6 > , N , W }

it may use a synonym index such as that shown in Figure 2 to determine that the relation **sched_flight** and the attributes **flno**, **orig**, and **dest** may be relevant.

flno	flight,number
plno	plane,airplane,number
dest	destination,arrive,to
m	meal,breakfast,lunch,dinner
...	...
sched_flight	flight,schedule,flno,freq,orig,dest,1cap,Bcap,Ccap,dep,arr,m,remarks
equipment	equipment,plane,airplane,plno,flno,date,cond

Figure 2.

Index entries may also exist for field values which are very standardized or coded. For example, since there is a finite number of airports, the thesaurus may associate each airport code with the equivalent terms expected in queries:

orig,dest	EWR	Newark, NJ, N.J., New Jersey, New York, NY, N.Y.
orig,dest	LAX	Los Angeles, LA, L.A., CA, California

The query-derived information now looks as shown in Figure 3.

relations:

relname
sched_flight

attributes:

attrname
flno

values:

value	**attrname**
EWR	orig
EWR	dest
	LAX
	LAX

unrecognized terms:

term
movie

Figure 3.

Clearly not every query term usable as a value can be recognized as pertaining to a specific attribute, as ''Newark'' was, in this example, associated with ''orig'' and ''dest''. A source may ignore unrecognized terms such as ''movie'' or it may search for them as a value in every field in the database. Between these two extremes, it may restrict searching to relations and/or attributes already identified as relevant. In any case no attribute need be searched whose data type differs from the perceived type of the query term in question. In this example, let us suppose that searching for ''movie'' is restricted to relevant relations (''sched_flight'' only) and attributes of type ''variable-length text strings'' (''remarks''). The resulting SQL-like query is:

```
SELECT * FROM sched_flight
    WHERE orig = 'EWR' OR orig = 'LAX'
        OR dest = 'EWR' OR dest = 'LAX'
        OR remarks = '*movie*'
```

The resulting response tuples must be assigned weights by which they can be sorted. A straightforward method is to compare the contents of the response tuples with the original query via the synonym index. It may also be possible to keep track of weights throughout the proceess of response relation generation.

This translation procedure is probably sufficient for many sources, but it does have drawbacks. In the worst case, it may generate a selection condition for every query-term value of every attribute in every relation. It also contains no provision for approximate matching, using similarity rather than strict equality as a search criterion. We are currently investigating the use of supplementary similarity functions, one per data type, which would (for example) recognize ''Tuesday'' as more similar to ''Wednesday'' than to ''Saturday'' and define similarity between numeric values as inversely proportional to their difference.

6. External Indexing

A necessary component of an information system of any kind, from flat file system to knowledge management system, is a mechanism for *locating* items of interest — for indexing into the information base so that wanted items may be retrieved. Since a querier must send its query to a specific address (or addresses), it needs, in effect, a procedure that takes as input a proposed query and returns a list of addresses of sources capable of answering that query. Such a procedure, to work perfectly, would need access to the *complete* information contents of every source. As this is not possible, we will settle for access to a *summary* of the contents of some subset of sources in an attempt to identify sources *likely* to be useful. The set of sources that a querier ''knows about'' are simply those for whom the querier has an address and a content summary stored in its external index. We will call those sources the querier's *neighbors*, thereby defining a logical network topology specific to the application.

We propose a bottom-up mechanism whereby each participant can construct and maintain for itself a customized external index containing whatever portion of the network "index" it finds important or useful. This index dynamically modifies itself to accommodate changing needs. The content summaries which it comprises we will call node descriptors (NDs).

6.1. Node Descriptors

Format and Use. A node descriptor is a weighted term set. This is the same format as QNF queries, and is compatible with the term vectors used as document descriptors in many IR systems. An ND may, in fact, be thought of as the descriptor of the "document" that is an entire database (and a query as a descriptor of an information need or of a hypothetical ideal document).

The primary operation on an external index is the similarity matching of a proposed query to the NDs in it. Matching yields a ranking of the NDs' suitability as sources for the query in question. Each querier should therefore store the index in a data structure that reflects its logical structure as a *set* of sets of weighted terms and on which the matching operation may be efficiently implemented. The indexing function can be performed completely without intervention or the querier may control it directly by browsing through the index to choose sources manually. The degree of automation is at the discretion of the querying node.

The term-based form of NDs ensures that vocabulary differences between querier and source are reconciled before the query is sent, since a query will not match an ND well unless the terms used in each overlap significantly. This eases the source's burden of translation and increases the likelihood that every source queried will return a relevant response. The querier can use a thesaurus to automatically augment a proposed query with other, related terms for the purpose of improving the source selection process.

ND Acquisition. When an index does not contain enough (or any) sufficiently good sources, more NDs must be solicited from neighbors. This process too can be fully automated. NDs may be acquired both *actively*, by requesting copies of them from neighbors' indexes, or *passively*, by receiving unsolicited "announcements" from sources, known or unknown. They may be *implicitly* sought in the process of choosing sources for a specific query or *explicitly* sought for the purpose of expanding one's index to support future queries. Requests may be *unrestricted*: "Please send me all NDs you have" or *restricted* by topic or address: "Please send me NDs on topic X", "Please send me your ND", or "Please send me Y's ND if you have it". As NDs are received, unwanted ones are filtered out manually or by means of a user-specific interest profile and the remainder are merged into the recipient's index.

ND Generation. It is the responsibility of each source to generate an ND suitable for transmission to prospective queriers. This process can be automated to a large extent. In effect a *summarization function* must be defined that, applied to a database or document collection, yields a set of descriptive terms with meaningful weights. This is related to the automatic indexing problem, which has been studied extensively and can provide guidelines for developing good summarization functions. For example, it has been demonstrated that full-text documents are best indexed by terms that appear in few documents of the collection but that appear many times within each of those few; such terms are said to be good *discriminators* [Salton81]. Likewise, the best indexing terms for a document collection should be those appearing in many of its documents but in few other databases. A non-text database can build its ND from a combination of data dictionary entries, representative data values, and manually entered terms.

It is a necessary consequence of autonomy that the quality of ND produced is controlled only by its source. NDs can vary greatly in their length, vocabulary, level of detail and coverage, and accuracy. The meanings of terms and weights are not standardized across sources. Queriers may therefore find it useful to modify the NDs they hold to reflect their actual experiences with sources over time. Queriers can control the up-to-dateness of their indexes by occasionally requesting fresh copies of NDs; in addition, a source may choose to send a current copy of its ND to any querier who submits a query containing unrecognized terms.

6.2. Optional Indexing Features

There may well be criteria other than topic by which queriers will wish to choose sources. The customized nature of the external index allows them to index on these criteria as well, which may include cost, speed, reliability, trustworthiness, and other subjective measures of quality. To implement this, quality criteria may be weighted by their importance to the querier, just as query terms are assigned weights. Their weighted values for the various sources can then be factored into the source selection/ranking process.

Sources too may supply other information about themselves besides an ND, pertaining to either the information they hold (its structure, presentation, timeliness, quality) or services they provide that are not part of the basic IE application (clipping service, subscription to a hypertext interface, etc.) Queriers may make use of this information if capable.

6.3. Index Maintenance

A potential difficulty with this architecture is that indexes can become quite large, possibly too large for a PC's storage media or for rapid processing. In this case a node may have to treat its index as a cache and institute an acceptable replacement policy to make the best use of limited space. The size of individual NDs can be reduced by removing terms that are of no interest to the querier. A large index can be partitioned by subject and stored in off-line

storage (e.g., floppy disks). Any of these solutions can be implemented individually by any node. Alternatively, queriers of similar interest may decide to form a cluster, partitioning among themselves the mutually interesting NDs. Some nodes may decide to specialize, acting as index servers to provide comprehensive and up-to-date indexes in their areas of expertise, or even as access intermediaries that build their NDs by consolidating those of other sources, then forward incoming queries to appropriate sources and direct the responses back to the original querier. Although queriers may make use of such services, it is important to note that the architecture itself does not *require* reliance on servers or other third parties.

While it may appear that individual indexes are a redundant, and therefore inefficient, solution to the external indexing problem, the decentralization of indexing appears to be the only way to truly preserve autonomy. It should be remembered that any given node will only retain that portion of the index of interest to it, which can be expected to be very small relative to the entire network index.

7. Querier Interface

The querier interface is implemented entirely at the querying node. Its purpose is to assist the querier in all IE-related tasks — constructing and modifying queries, querying and maintaining the external index, selecting sources, updating auxiliary data structures such as the thesauri, filters, profiles, and ND annotations, and viewing and filing responses.

Although for the sake of generality we must assume untrained users who require a more or less transparent interface to the network, more sophisticated users may prefer to exchange some transparency for greater control. In fact a continuum of transparency is possible, and the choice of interface is made independently by each node. A sophisticated user may understand and interact with all of the components shown in Figure 1 while a naive user prefers to view querying as a black-box operation, delegating to the software every detail except actually typing in queries.

Standard automatic indexing techniques [Salton86] can be used to convert arbitrary natural-language prose requests into QNF queries, as mentioned in [Simpson88]. This suggests a query interface suitable for even very naive users whereby the querier supplies a "sample response", an example of the sort of document being sought. The words in the sample become the basis of the QNF query, once low-content "stopwords" and other terms known to be poor discriminators have been automatically removed. Term weights may be assigned based on their frequency in the sample and/or their discrimination value as determined by examining the external index. Or the querier can assign weights via a graphic interface, adjusting a lever or dial icon on the screen for each term.

A local thesaurus can be used to expand a proposed query before matching it against the external index, a process which increases the reliability of the matching procedure and allows the querier to disambiguate polysemic terms by associating them with additional terms

related to the intended meaning. The thesaurus can be expanded manually when desired, and the source selection software may prompt the querier to supply synonyms for query terms the first time they are used or for unrecognized terms in newly acquired NDs.

The querier should be able to browse through the external index to view NDs, addresses, and source quality annotations. Sources may even be chosen manually in lieu of similarity matching; or matching may be used just to narrow the field of candidate sources, after which their descriptions are presented visually to the querier for final selection. When sources have supplied additional information with their NDs, the source selection procedure may display this to the querier.

Queriers may define *filters* to select which incoming responses and NDs to store and which to discard. Filtering is primarily by topic but may reflect other considerations such as sender's identity or size in bytes. A querier *profile* may be used to instruct the querying software to periodically seek external information on topics of continuing interest. Records must be kept of what queries have been sent to whom and what responses were received and when. The user is able to query these records at any time. Responses should be temporarily indexed under the queries that elicited them, until the querier is ready to view them and assign those of interest to permanent storage.

8. Conclusions

We have discussed a new form of computer-communication network which we expect will develop over the next several years, one populated by millions of independent computer systems. We postulate that the network will serve primarily to enhance many forms of communication among individuals and organizations. While the characteristics of this environment — autonomy, heterogeneity, and large size — pose challenges to application design, we have presented an architecture for topic-based querying of heterogeneous databases that meets the constraints imposed by these characteristics.

The issues arising from the information exchange application were discussed and several component problems were identified, including: choice of language for communicating queries, customization of querier interfaces to suit both experienced and naive users, the design of query translation procedures for implementation by diverse source database systems, and discovery of sources in the network by topic. Solutions were outlined for each of these problems that follow the principles of statelessness, separation of function, and decentralization.

References

[Bernstein84]

Bernstein, Lionel M. and Williamson, Robert E. "Testing of a Natural Language Retrieval System for a Full Text Knowledge Base", *Journal of the American Society for Information Science*, vol. 35, no. 4 (July 1984), pp. 235-247.

[Bookstein81]

Bookstein, A. "A Comparison of Two Systems of Weighted Boolean Retrieval", *Journal of the American Society for Information Science*, vol. 32, no. 4 (July 1981), pp. 275-279.

[Doszkocs83]

Doszkocs, Tamas E. "From Research to Application: The CITE Natural Language Information Retrieval System", *Research and Development in Information Retrieval, Proceedings*, Berlin, May 1982, G. Salton and H.-J. Schneider, eds., pp. 251-262. Springer-Verlag, New York, 1983.

[Heimbigner85]

Heimbigner, Dennis and McLeod, Dennis. "A Federated Architecture for Information Management", *ACM Transactions on Office Information Systems*, vol.3, no. 3 (July 1985), pp. 253-278.

[Jamieson79]

Jamieson, S. H. "The Economic Implementation of Experimental Retrieval Techniques on a very large scale using an Intelligent Terminal", *Proc. of the Second International Conference on Information Storage and Retrieval*, Dallas, September 1979, *ACM SIGIR Forum*, vol. 14, no. 2, pp. 45-51.

[Kraft85]

Kraft, Donald H. "Advances in Information Retrieval: Where Is That /#*&@¢ Record?", *Advances in Computers*, vol. 24, Marshall C. Yovits, ed., pp. 277-318.

[Marcus81]

Marcus, Richard S. and Reintjes, J. Francis. "A Translating Computer Interface for End-User Operation of Heterogeneous Retrieval Systems. I. Design. II. Evaluations", *Journal of the American Society for Information Science*, vol. 32, no. 4 (July 1981), pp. 287-317.

[Moulinoux83]

Moulinoux, C., Faure, J. C. and Litwin, W. "MESSIDOR: A Distributed Information Retrieval System", *Research and Development in Information Retrieval, Proceedings*, Berlin, May 1982, G. Salton and H.-J. Schneider, eds., pp. 51-61. (Lecture Notes in Computer Science, vol. 146, Springer-Verlag, New York, 1983.)

[Salton86]

Salton, Gerard. "Another Look at Automatic Text-Retrieval Systems", *Communications*

of the ACM, vol 29, no. 7 (July 1986), pp. 648-656.

[Salton81]

Salton, G., Wu, H. and Yu, C. T. "The Measurement of Term Importance in Automatic Indexing", *Journal of the American Society for Information Science*, vol. 32, no. 3 (May 1981), pp. 175-186.

[Schatz87]

Schatz, Bruce R. "Telesophy: A System for Manipulating the Knowledge of a Community", *Globecom Tokyo 87 (Proc. of the IEEE Global Telecommunications Conference)*, vol. 2, Tokyo, November 1987, pp. 1181-1186

[Schwartz88]

Schwartz, Michael F. "The Networked Resource Discovery Project: Goals, Design, and Research Efforts", U. of Colorado Technical Report no. CU-CS-387-88, May 1988.

[Simpson88]

Simpson, Patricia. "Query Processing in a Heterogeneous Retrieval Network", *Proc. of the 11th International Conference on Research & Development in Information Retrieval*, Grenoble, June 1988, pp. 359-370.

[Williams86]

Williams, Martha E. "Transparent Information Systems Through Gateways, Front Ends, Intermediaries, and Interfaces", *Journal of the American Society for Information Science*, vol. 37, no. 4 (July 1986), pp. 204-214.