# CARROT II: Collaborative Agent-based Routing and Retrieval of Text *

R. Scott Cost, Tim Finin, Srikanth Kallurkar, Hemali Majithia, Charles Nicholas, Yongmei Shi, Ian Soboroff
Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County
Baltimore, MD USA
email: {cost,finin,skallu1,hema1,nicholas,yshi1,soboroff}@csee.umbc.edu

20 September 2001

## Abstract

We describe CARROT II (**C2**), an agent-based architecture for distributed information retrieval and document collection management. In **C2**, agents provide search services over local document collections or information sources. A **C2** system can consist of an arbitrary number of agents, distributed across a variety of platforms and locations, and integrating widely varying information sources. **C2** agents advertise content-derived metadata that describes their local document store; this metadata is sent to other **C2** agents which agree to act as brokers for that collection - every agent in the system has the ability to serve as such a broker. A user query is sent to any **C2** agent, which can decide to answer the query itself from its local collection, or to send the query on to other agents whose metadata indicate that they would be able to answer the query, or send the query on further. Search results from multiple agents are merged and returned to the user. **C2** can be used to implement a wide variety of distributed information services, such as common single-stage web metasearch or Harvest-style brokering, as well as more sophisticated collection selection and ranking algorithms. **C2** differs from similar systems in that metadata takes the form of an unstructured feature vector, and that any agent in the system can act as a broker, so there is no centralized control. Furthermore, the agent architecture uses a sophisticated agent communication language for information exchange, opening the door to sophisticated coordination algorithms among **C2** agents. Search abilities are further enhanced with the integration of tools and ideas from the Semantic Web effort.

*This work supported in part by the United States Department of Defense.

## 1 Introduction

The integration of heterogeneous data sources is becoming increasingly important. One approach to routing queries within such an environment is to provide a logical 'front-end', which wraps the sources into one common framework. While this helps to solve the integration aspect of the problem, it presents an efficiency problem, in that all queries must follow the same path through the front end.

We have developed a large-scale, distributed query routing and information retrieval system that can serve as a valuable testbed for a variety of important research problems. Named CARROT II, as the successor of an earlier project (Collaborative Agent-based Routing and Retrieval of Text) [16, 14] (originally CAFE [15]), this system is composed of a flex-

ible hierarchy of query routing agents. These agents communicate with one another using KQML [17] and the Jackal platform [13], and may be distributed across the Internet. While all agents in the system are alike, they can each control widely varying information systems. Currently, they populate the nodes of our twelve-node Beowulf system. Agents interact with information sources via a well-defined interface. Queries presented to any agent in the system are routed, based on the content of the query and metadata about the contents of the servers, to the appropriate destination. Agents themselves are uniform and extremely simple. The flexibility of this system opens up a wide range of opportunities.

This system as implemented initially contains wrappers that extend several existing IR systems (e.g. MG [41], Telltale [29, 28]), as well as **C2**'s own, modest IR system. These wrappers present a very basic interface to the **C2** system for operating on documents and metadata. One initial goal of the project has been the integration of heterogeneous IR engines within a single search system. This will be followed by the extension of the use of these wrappers to cover a wide variety of information sources, including more structured search systems, databases, or even humans.

Agents are addressable via commands that are communicated in KQML. This means that a **C2** system can easily be created, configured and accessed by another information system, and so can be employed to extend the search capabilities of an existing project.

Such an endeavor presents several challenges. One of the most immediate problems that present is the form, representation and manipulation of metadata. Agents have metadata that represents the information content 'owned' by the various sources attached to the system, so the form and content of that metadata has an important impact on the systems ability to route queries effectively. It becomes an even more difficult problem when you consider that information sources may be heterogeneous, and may therefore have very dissimilar metadata representations. Another problem is how the metadata should be used by agents to actually route queries. Finally, since the system of agents will be dynamically reconfigurable, it may be necessary to move metadata from one node to another, or perhaps even combine it into a higher order object. Such schemes could be used to both affect query effectiveness and query throughput.

Related to the metadata question is the use of other information sources in informing the query routing algorithm. In addition to using local data about information source content, it may be possible to use information about past queries, the user or user community, just to name some examples, to enhance search performance. There are potential issues of learning here. **C2** agents are constructed with an open interface that allows them to dynamically integrate external 'experts' into the decision making process.

From the perspective of agent system deployment, we would like the nodes in the **C2** system to be able to coordinate closely with, and dynamically change their associations with, other agents. This means the ability to interact at a high level, in addition to efficiently moving potentially large volumes of information. We use the Jackal platform to support communication among agents in **C2** and to provide an interface to the outside world. In addition to its support for agent communication with KQML, we use its conversation management capabilities to specify and implement higher-level behaviors for the various negotiation and management tasks required within the agent system. This is supported by an ontology of the query routing and information retrieval problem domains.

Some interesting questions which have not been discussed but are also of interest include the topology of query routing systems, and the value of benefits of node mobility, and user modeling. Also, there is an interest in the management of standing queries; if queries are posed which persist in the system, how/where should they be managed, especially if the system is dynamic. Also, if the routing algorithm permits the spread of queries to multiple information sources, a results fusion problem presents itself when the data is collected from those sources.

## 2 Related Work

In the past there have been attempts to introduce the concepts of agent-based information retrieval. Systems like SavvySearch [23] demonstrated a simple approach to querying web search engines and combining their results in a single ranked order.

Historically, Harvest was the first system to demonstrate the use of broker agents in distributed search. The Harvest system [6, 9, 8] is a distributed, brokered system originally designed for searching Web, FTP, and Gopher servers. In Harvest, "gatherer" agents collect metadata from information providers and deliver it to one or more brokers. Metadata objects are represented in Summary Object Interchange Format (SOIF), an extensible attribute-value-based description record; common attributes include author, host information, keywords, modification times, checksums, time-to-live, and file size. These records are mainly constructed using an information extraction system, but the records can also contain hand-written abstracts, category assignments, or the full text of a document [7]. Harvest pioneered the ideas of brokering, metasearch, replication, and caching on the Internet.

### 2.1 Distributed Information Retrieval

Information Retrieval in a distributed environment normally follows three steps [10]:

1. Information Source Selection: Select best information source per query
2. Query Processing: Send query to source(s) and return ranked list of documents
3. Results Fusion: Create single ranked list from ranked lists of all sources.

For retrieval from text one of the methods for information source selection is use of automatically generated metadata from the content. Comparing the query to metadata about the sources can reveal the possible relevance of each source to the query. CORI [11] and gGloss [19] are examples of such metadata in information source selection. The CORI model is based on inference networks. CORI creates a virtual document containing Document Frequency ($DF$) and Inverse Collection Frequency ($ICF$). The $ICF$ indicates importance of the term across the collections and is analogous to the Inverse Document Frequency ($IDF$), which is a measure of term importance in a single collection. gGloss takes another approach to database selection. gGloss creates a virtual document containing $DF$(s) and Term Frequency ($TF$), i.e. number of occurrences per document of unique terms of the collection. French et al. [18] showed that CORI performed better than gGloss in terms of retrieval effectiveness, however they could not provide a reason for CORI's better performance.

Gibbins and Hall [27] modeled query routing topologies for Resource Discovery in mediator based distributed information systems. Queries are routed by a referral (of a server) by the mediator or by a delegation of the query to the mediator. Liu [25] demonstrated query routing using processes of query refinement and source selection, which interleaved query and database source profiles to obtain a subset of good databases. User query profiles contained query scope and query capacity descriptions while source capability profiles contained content and capability descriptions. The sources were selected through serially pruning until a subset of sources (based on source description) that most match the query is obtained.

The final step in answering a query is fusing the the ranked list from the queried sources to obtain a single ranked list. Voorhees et.al. [36] showed the use of query training and query clustering to first query appropriate data sources and then merge the results. The query training approach used a dice biased by the number of documents still to be merged, whereas query clustering applied a factor to the results based on the importance of the source it was from and then rank based on the new scores. Aslam and Montague [1] showed that results fusion based on ranks alone can be as good as regular fusion techniques and that relevance scores are not required.

In general, there is a performance gain by distributing information, but distributed retrieval lags behind centralized retrieval in terms of retrieval effectiveness, i.e. percentage of relevant documents returned for a

query. However Powell et al. [30] showed that a distributed search can outperform a centralized search under certain conditions.

## 2.2 Semantic Web

The current web is primarily composed of pages with information in the form of natural language text and images intended for humans to view and understand. Machines are used primarily to *render* this information, laying it out on the screen or printed page. The idea behind the *semantic web* is to augment these pages with mark up that captures some of the meaning of the the content on pages and encodes it in a form that is suitable for machine *understanding*. This requires a new kind of markup languages, one that supports defining shared data models or ontologies for a domain and allows page authors to make statements using this ontology. The markup languages currently being used to explore this idea include RDF and DAML+OIL. Both languages are built on a foundation of XML and DAML+OIL is further layered on RDF.

The goal of DAML+OIL is to enable the transformation of the currently human-oriented web, which is largely used as a text and multimedia repository only, into a Semantic Web as envisioned by Berners-Lee [3, 2]. This process involves the augmentation of web pages with additional information and data that are expressed in a way that facilitates machine understanding [20, 22]. DAML+OIL is built upon the capabilities of an already existing syntactic language, the Extendable Markup Language (XML) [40], and of the Resource Description Framework and Resource Description Framework Schema (RDF/S) [37, 39, 38, 34]. These are XML applications that provide a number of preliminary semantic facilities required in the realization of the Semantic Web vision.

XML was developed by the World Wide Web Consortium (W3C) as a standard for alternative data encoding on the Internet that was primarily intended for machine processing. The XML standard provides the necessary means to declare and use simple data structures, which are stored in XML documents and which are machine-readable. Subsequently, the information made available in these documents can be processed or translated into additional XML documents to provide the appropriate form for human understanding, such as text-to-voice, graphics or HTML conversion. However, since XML is defined only at the syntactic level, machines cannot be relied upon to unambiguously determine the correct meaning of the XML tags used in a given XML document. Consequently, XML is not suitable as a desired language for representing complex knowledge.

As a result, the W3C Consortium has developed RDF/S with the goal of addressing the XML deficiencies by adding formal semantics on the top of XML. These two standards provide the representation frameworks for describing relationships among resources in terms of named properties and values, which are similar to representation frameworks of semantic networks and rudimentary frame languages such as RDF Schema. Yet, both standards are still very restricted as a knowledge representation language due to the lack of support for variables, general quantification, rules, etc.

DAML+OIL is an attempt to build upon XML and RDF/S to produce a language that is better suited for building the Semantic Web. It follows the same path for representing data and information in a document as XML, and provides similar rules and definitions to RDF/S. In addition, DAML+OIL also provides rules for describing further constraints and relationships among resources, including cardinality, domain and range restrictions, as well as union, disjunction, inverse and transitive rules. DAML+OIL is, therefore, an effort to develop a universal Semantic Web markup language that is sufficiently rich to provide machines not only with the capability to read data but also with the capability to interpret and infer over the data. DAML+OIL will enable the development of intelligent agents and applications that can autonomously retrieve and manipulate information on the Internet and from the Semantic Web of tomorrow.

DAML+OIL is still an evolving language and may be joined by other expressive markup languages with well defined semantic foundations. Early in 2001, the W3C established an official Semantic Web activ-

ity[1] as the successor to the W3C Metadata Activity. It's goals are to continue the development of existing standards (e.g., RDF), to work with communities using semantic markup (e.g., the Dublin Core) and to explore and develop new languages and technologies to realize the semantic web vision.

# 3 Research Issues

There are a number of different issues which affect the design and operation of a system such as **C2**. Among the more significant are the form, creation and use of metadata, which provides a basis for all query routing decisions, the actual algorithms used for query routing, and the methods used for fusing results obtained from diverse sources. These issues and others are discussed in some detail below.

## 3.1 Metadata

Effective use of metadata is essential to the **C2** system. Currently, **C2** uses a vector-based representation of metadata which describes the contents of the local corpus (see [16]). Metadata, as well as corpus documents, are managed by the underlying IR engine (in **C2**, currently this is MG). This metadata is first order only, and is to be distinguished from information characterizing the set of collections known to a given agent (forward knowledge), or higher order metadata. **C2** metadata is derived from the distributed search work of Callan and Croft et al [11]. For the collection selection problem, or, determining the collection(s) (we use the terms collection and the agents that own them interchangeably) most suitable for a query, the issue of scalability determines the type of information stored in the metadata. For example, given a **C2** system of 1000 agents, in order for each of the agents to have information about all the other agents in the system, each agent would have to receive and store 999 metadata vectors from other agents. This will not scale for either a homogeneous or a heterogeneous collection. One way to solve this would be to derive a higher order metadata that represents *chunks* of the network. Designated agents would house such metadata. Agents would store the metadata vectors of only those agents that comprise that chunk. The higher order metadata would then be used to determine the top $n$ chunks and route the query to them. This technique is recursive in nature as the selected chunks would determine the top n agents contained in them and field them the queries.

The form of metadata used supports interoperability. A **C2** system in essence should be able to field queries to any agent that has been deemed suitable based on the document information by *looking* at the metadata. This should not be hampered by the underlying information source, may it be a N-gram based IR engine, a Search engine on the global internet or a RDF or DAML+OIL [12] based system.

In its simplest case a **C2** would comprise of a single agent responsible for all the documents. As we expand the system and setup **C2** agents on numerous sites document distribution plays an important role in retrieval effectiveness. As described, the documents a **C2** agent houses can be heterogeneous both in content and size. The space requirements for indexing metadata about a heterogeneous corpus is comparable to that of the documents. Homogeneity in a corpora provides more compact metadata as fewer terms can describe documents similar in content. Homogeneity in the documents can be obtained by creating logical sub-collections. Over time the agents cooperate and move similar looking documents into content based sub-collections, the content determined by the metadata. Homogeneity is a very important attribute for a dynamic system like **C2** to obtain higher retrieval effectiveness.

As we extend the system to integrate more varied and more dynamic information sources, we will face a challenge in incorporating them into our metadata representation and creation scheme. For example, how do we create the metadata representation, as a term-frequency vector, of all of the documents on the internet. There are a number of approaches that suggest themselves. One such would be to create manually a set of queries which represent your desired view of the information source. By posing those queries to the source, we get a set of 'representative documents', which can be indexed and described by **C2** metadata. This approach can be enhanced with techniques such

[1] See http://www.w3.org/2001/sw/

as query expansion.

### 3.2 Query Routing

The form of metadata in turn determines the distribution of the documents and routing of queries. Routing of queries to the appropriate corpus agent is the primary task of **C2** [25, 33]. Depending on the approach used, queries may be sent to one or multiple sources at each step. The latter is increasingly likely as sources become more and more varied, and their contributions to the final result differ significantly. One other thought which could be explored is the notion of search, or exploring various avenues independently or in sequence, until a satisfactory results set is obtained.

As results are obtained from the various corpus agents in the system, they must somehow be fused into one consistent view.

### 3.3 Results Fusion

In typical distributed retrieval situations, document scores may vary depending on the collection. It is essential to merge the results obtained from each of these collections into an accurate single ordering. The results fusion or results merging problem has been an important research issues for some time [35, 26, 42, 24]. A solution to the results fusion problem in a **C2** system would have to take into account the fact that a **C2** system uses disparate IR systems, that is the techniques used to determine the similarities between the documents and queries can vary. In most **C2** scenarios one can expect that the document collections to be disjoint, i.e. the agents do not have documents in common. Also the document distribution can be non-uniform in terms of the document size and number of documents.

Additionally, in integrating heterogeneous information sources, we face the problem of fusing disparate types of results. For example, a classic IR system might return a ranked list of documents, while another might return data about the query, unordered results, etc. It may be necessary to select from among a variety of result types, fuse them intelligently, or return a multi-modal response which incorporates all or some. For example, in some scenarios the results returned may take the form of a multipart 'portfolio', with raw, statistical, high-level, or perhaps visual components.

## 4 Architecture

A **C2** system is essentially a collection of coordinated, distributed agents managing a set of possibly heterogeneous IR resources. These agents each perform the basic tasks of collection selection and query routing, query processing, and data fusion. In order to effect this coordination, some amount of underlying structure is required.

There are three components to the **C2** architecture. The central work of **C2** is performed primarily by a distributed collection of **C2** agents. There is also a network of infrastructure agents which facilitate communication and control of the system. Finally, a small set of support agents facilitates access to the system, and coordinates its activities. Each of these components is described in detail below.

### 4.1 C2 Agents

The **C2** agent is the cornerstone of the **C2** system. Its role is simple; manage a certain corpus, accept queries, and either answer them itself, or forward them to other **C2** agents in the system. In order to do this, each **C2** agent can create and distribute metadata describing its own corpus to other **C2** agents. All **C2** agents are identical, although the IR systems they manage may vary dramatically.

#### 4.1.1 Control Aspect

The **C2** agent is Java agent. It is responsible for one node in the system, which means one information source, such as an instance of MG. It must process all incoming and outgoing queries and results messages. In addition, it must manage the nodes metadata store, and negotiate with other nodes for the exchange of metadata.

To communicate with other agents in the system, **C2** agents use Jackal as a communication tool.

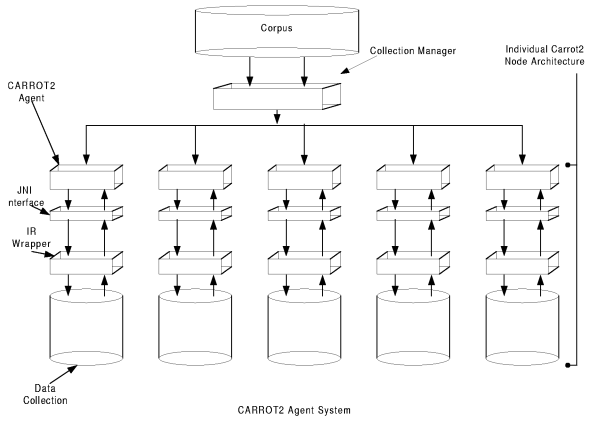


Figure 1: Individual **C2** nodes.

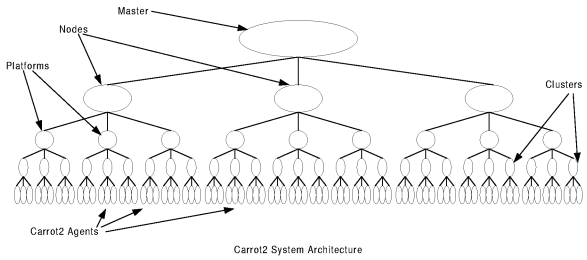


Figure 2: **C2** Architecture: Infrastructure hierarchy, terminating in a collection of **C2** nodes.

Jackal provides a communication infrastructure for the agents to communicate in KQML.

The **C2** agents must also interface with the local information source. In our current instantiation, this is a classic IR engine, such as MG. A standard interface provides methods for manipulating documents and metadata and handling queries. The agent maintains a catalog of metadata which it manages with MG. The catalog contains information about the metadata objects stored.

## 4.2 IR Aspect

A **C2** agent interfaces with an information source which may be an ordinary IR package, or a database manager.

The **C2** system currently has a wrapper that interfaces with the WONDIR[2](in-house) IR engine. It can however be extended to support other types of Information sources. As mentioned earlier, metadata is derived from the document collection. The metadata takes the form of a vector of the unique "N-grams" of the collection and a sum of their number of occurrences across all documents in the collection. Hence unlike Harvest, **C2** metadata describes the agent's collection of documents, not a single document. **C2** uses such metadata for source selection per query. The motivation for such a form of metadata comes from relative ease of use, low cost of generation and the ability to aggregate metadata, such that a single vector may contain metadata about multiple agents.

The **C2** metadata is different from the *CORI virtual document* in that CORI uses document frequency, i.e. the number of documents the term has occurred in the collection. **C2** uses term frequency, i.e. the sum of the number of occurrences of the unique terms over all documents. In many if not most collections, a large percentage of terms have occurrences of 0 or 1. By storing the sum of the terms **C2** metadata adds more weight to terms that appear more often in the collection either in more documents or in fewer documents with larger occurrences per document.*CORI* would favor terms that occurred only in more documents.

The same query operation can now be performed on both documents and metadata. A query operation returns a similarity score using $TF * IDF$ based cosine similarity [32]. Querying a collection returns a ranked list of the documents sorted by their similarity scores. For querying metadata collection $IDF$ is replaced by the $ICF$ (see Section 2). On average from empirical observations the metadata is 8-10% of the size of the document collection. The agent that creates the metadata attaches its signature information to the vector.

## 4.3 C2 Infrastructure Agents

In order to support the successful inter-operation of potentially very many **C2** agents, we have constructed a hierarchical infrastructure (See Figure 2).

[2]Word or N-gram based Document Indexing and Retrieval

At present, this infrastructure is largely dormant while **C2** is in operation, serving primarily to facilitate the orderly startup and shutdown of the system, and provide communications support.

The infrastructure is hierarchical, and is controlled by a single Master Agent, which may be located anywhere on the network. At startup, the Master Agent is instructed as to the number of agents required, and some factors regarding their desired distribution. These include the number of physical nodes to be used, as well as the degree of resource sharing at various levels.

The Master Agent starts a Node Agent on each participating machine, and delegates to it the task of creating a subset of the required agents. The node will be divided into Platforms, or independent Java Virtual Machines, each governed by a Platform Agent. The Node Agent creates an appropriate number of Platforms, and again delegates the creation of a set of agents.

Within each Platform, the Platform Agent creates a set of Cluster agents. The purpose of the Cluster Agent is to consolidate some of the 'heavier' resources that will be used by the **C2** agents. Primarily, this means communication resources. A Cluster Agent maintains a single instance of Jackal. Each Cluster Agent creates a series of **C2** agents; these run as subthreads of the Cluster Agent. Because most agents will be dormant at any one time, we could allow a **C2** agent to be assigned more than one collection, creating a set of 'virtual' agents, but we have not explored this to date.

### 4.4 C2 Support Agents

While the agents in the **C2** system work largely independently, a small set of support agents serve to coordinate the system's activities. These are the Agent Name Server, Master Agent, Logger Agent and Collection Manager.

A Master Agent manages the operation of the system through the hierarchy of infrastructure agents. This could be operated by a user, or act as an interface to another client system.

An Agent Name Server provides basic communication facilitation among the agents. Through the use of Jackal, **C2** employs a hierarchical naming scheme, and the burden of that is distributed through a hierarchy of name servers.

A Logger Agent monitors log traffic, and allows the system to assemble information about the details of operation. This information can then be used to feed monitors or visualization tools.

Finally, a Collection Manager facilitates the distribution of data and metadata. The Collection Manager is arguably on of the most important agents in the system. It determines which collection of documents or information source will be assigned to each agent, how each agent will distribute its metadata, and what set of agents will be visible outside the system.

## 5 C2 Operation

The instantiation and operation of a **C2** system has a number of distinct phases. Most of these phases occur during startup, and significantly impact the behavior of the system.

### 5.1 Partitioning

Before a **C2** system can be fielded, a decision must be made about how the target collection will be partitioned. There are a number of factors that may come into play here. Some may reflect the physical location of the collection or collection components, or the optimal physical distribution of documents. Others may involve the clustering of collection components into topical or other semantically relevant groupings.

Once established, the collection partition has a significant impact on the distribution of agents and collection assignment.

### 5.2 Distribution of Agents

The number of **C2** agents to be created is determined by the partitioning scheme. We next need to determine how to distribute and arrange these agents across available physical platforms. This is done automatically by the **C2** infrastructure, and is influenced by a number of system level factors.

## 5.3 Collection Assignment

Once they are ready, **C2** agents are assigned subcollections by the Collection Manager. This assignment is based on the original partitioning scheme. A major factor in the decision to do assignment this was the anticipation of a dynamic environment in which agents enter and leave the system, and in which collections are added, deleted or modified as well. It is also desirable to allow for the exchange of corpus elements from agent to agent in order to effect load balancing and metadata refinement/clustering.

## 5.4 Metadata Distribution

The distribution of Metadata has a profound impact on the system's ability to route queries effectively, and determines the 'shape' of the **C2** system. There are an endless number of metadata distribution schemes that can be used. For example, each agent could broadcast it's metadata to every other agent in the system. Under this scheme, any agent receiving a query would have complete (and identical) knowledge of the system, and be able in theory to find the optimal target for that query. Once the system reaches a certain size, however, this scheme is no longer practical, for a number of reasons. Other schemes involve sending metadata to certain designated 'brokers', or using more mathematical, quorum-based distributions. It should be clear that the distribution scheme chosen is very closely tied to the algorithm used to route queries, and to the choice of agents we expose externally.

Agents receive instructions on metadata distribution from the Collection Manager.

Metadata Distribution can take place in three different modes: global, local, or flood. In the global mode the metadata is sent to one global broker agent. In the local mode the metadata is sent to one agent on each node. In the flood mode the each agent knows the number of agents in the system and sends metadata to each of them. There is mapping of the agent numbers to names. For link-based referral, the agent looks at the documents it is loading, and compiles a list of the URLs referred to in the documents. Then it sends a request to the collection manager, asking for the names of the agents that hold those documents.

## 5.5 Query Processing

Once the system is running, the Collection Manager becomes the primary or initial interface for outside clients. A client first contacts the Collection Manager to get the name or names of **C2** agents that it may query. The names in this set will be determined by above decisions. The client should then target queries at randomly selected agents from the given set. For example, in the classic **C2** mode, the names of all **C2** agents in the system will be given, and the client will randomly chose agents from that list to accept queries. It is also possible to model more restricted or brokered architectures by limiting the list to only one or a few agents, which would then feed queries to the remainder of the system.

In response to an incoming query, an agent must decide whether the query should be answered locally, forwarded to other agents, or a combination of the two. The first thing it will do is compare the query to its local metadata collection, to determine the best destination. Based on the results, it may send the query to the single best source of information, or it may chose to send it to several. On of those sources may be its own local IR engine. Once answers are computed and received, the results are forwarded back to the originator of the query. If more than one information source is targeted, the agent faces the problem of fusing the information it receives into one coherent response.

Queries may be routed through a number of different agents before finally being resolved; this depends on the scheme used for metadata distribution and the routing algorithm (See Figure 3). For example, the simplest scheme is to have each agent broadcast its metadata to every other agent in the system. The corresponding routing algorithm would be to route to the best information source. Since all agents have the same metadata collection and employ the same algorithm, a query will be forwarded at most once before being resolved. For schemes which employ a more efficient distribution of metadata, or possibly higher order metadata, queries may pass through many **C2** agents before finally being resolved.

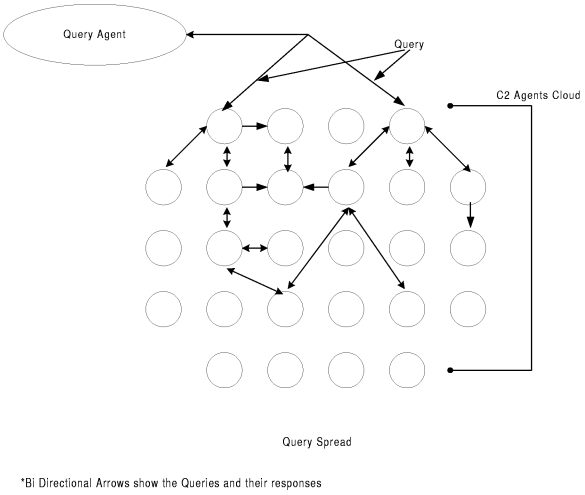


Figure 3: Queries routed through **C2** nodes.

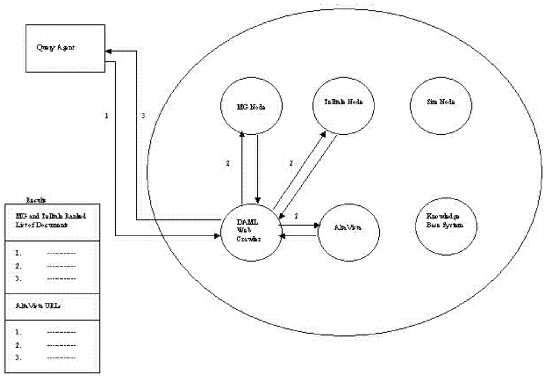


Figure 4: Hypothetical scenario involving classic IR and web-based information sources.

# 6 Example Scenario

Having described the architecture of the C2 system, we present a hypothetical scenario to illustrate it's use. In this case, we show the the integration of some classic IR systems, world wide web search/crawler engines, and an expert systems.

Consider the nodes in the C2 system shown in Figure 4. Each node represents a C2 agent. There are three IR agents: an MG node, a Telltale node and a WONDIR node. In addition, there is a web search node (wrapping AltaVista), and a web crawler, both of which have been extended to make use of DAML+OIL semantic markup. Also, the system contains one high-level knowledge base/reasoning system. The MG, Telltale and WONDIR nodes manage local, disjoint corpora of text documents.

In our sample scenario, an external client would like to pose a query which is largely text, with some embedded DAML+OIL markup. Having obtained a list of available entry points from the collection manager, it randomly selects the crawler node, and submits the query. Note that it does not matter which node initially handles the query. Based on the raw text portion of the query, the crawler node decides that it can be best handled by the MG node. Because the Telltale node was also a very close match, however, it chooses to send the query to both. In addition, recognizing the semantic markup present in the query, it chooses to forward the query to the DAML+OIL-enhanced web search engine as well.

In a more complex scenario, the queries could have been extended further, but we will assume that they are answered by the nodes listed above, and the results are returned to the crawler node. Since MG and Telltale are similar systems in some ways, it is possible for the agent to fuse the results obtained from these two into one ranked document list. The Web agent is a very different system, however. The crawler agent assembles a multi-part response to the client, which contains both the ranked list of documents obtained from the IR engines, and a ranked list of relevant URLs obtained from the web agent.

The architecture of the the C2 system brings up various issues which have not been addressed-for example, how to start a system of heterogeneous C2 agents. Another important aspect is that of the metadata pool which governs the decision involved in routing the query. Due to this nature of propagation of the query, we can describe the C2 system as a classic example of a distributed information retrieval system over the world wide web.

# 7 Future Work

Having constructed the core architecture, with support for several classic IR engines, we plan to move work forward primarily in three different but related directions. The first involves improving the basic system itself; exploring ways to make the system more robust, flexible, and easier to use. The second will involving experimenting with the variety of architectures that **C2** makes possible. That is, varying the routing algorithms, distribution models, extending our current notion of metadata, etc. Finally, we will explore ways in which the system can be extended to incorporate or interact with other technologies, and in particular,ideas and standards from the Semantic Web effort [22, 21, 3].

# 8 Conclusion

In this paper, we have described our work with the **C2** agent-based architecture for distributed information retrieval and document collection management. Because of it's flexible design, **C2** can be used to implement a wide variety of distributed information services, such as common single-stage web metasearch or Harvest-style brokering, as well as more sophisticated collection selection and ranking algorithms. This flexibility enables it to facilitate the integration of a variety of information sources, including classic IR systems, Semantic Web-based search engines, structured databases and more. It is our hope that use of this platform will prove to be a powerful and effective approach to query answering in a distributed and possibly heterogeneous environment, and also facilitate fruitful research in many of the areas highlighted above, including metadata, query routing algorithms and data fusion.

# References


[1] J. A. Aslam and M. Montague. Models for metasearch. In *ACM SIGIR*, pages 276–284, 2001.

[2] T. Berners-Lee and M. Fischetti. Weaving the web: The original design and ultimate destiny of the world wide web by its inventor. *Harper, San Francisco*, 2001.

[3] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.

[4] D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, B. Zhao, and J. Kubiatowicz. Oceanstore: An extremely wide-area storage system.

[5] K. D. Bollacker, S. Lawrence, and C. L. Giles. Citeseer: An autonomous web agent for automatic retrieval and identification of interesting publications. In *Proceedings of the Second International Conference on Autonomous Agents (Agents '98)*, Minneapolis, 1998. ACM Press.

[6] C. M. Bowman, P. B. Danzig, D. R. Hardy, U. Manber, and M. F. Schwartz. Harvest: A scalable, customizable discovery and access system. Technical Report CU-CS-732-94, Department of Computer Science, University of Colorado, Boulder, 1994.

[7] C. M. Bowman, P. B. Danzig, D. R. Hardy, U. Manber, and M. F. Schwartz. The harvest information discovery and access system. In *Proceedings of the Second International Conference on the World-Wide Web*, pages 763–771, 1994.

[8] C. M. Bowman, P. B. Danzig, D. R. Hardy, U. Manber, and M. F. Schwartz. The Harvest information discovery and access system. *Computer Networks and ISDN Systems*, 28(1–2):119–125, 1995.

[9] C. M. Bowman, C. Dharap, M. Baruah, B. Camargo, and S. Potti. A File System for Information Management. In *Proceedings of the ISMM International Conference on Intelligent Information Management Systems*, March 1994.

[10] J. Callan. *Advances in Information Retrieval*, chapter Distributed Information Retrieval, pages 127–150. Kluwer Academic Publishers, 2000.

[11] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In E. A. Fox, P. Ingwersen, and R. Fidel, editors, *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–28, Seattle, Washington, 1995. ACM Press.

[12] R. S. Cost, T. Finin, A. Joshi, Y. Peng, C. Nicholas, H. Chen, F. P. Lalana Kagal, Y. Zou, and S. Tolia. ITTALKS: A case study in the Semantic Web and DAML. In *International Semantic Web Workshop (SWWS) – Infrastructure and Applications for the Semantic Web*, 2001.

[13] R. S. Cost, T. Finin, Y. Labrou, X. Luan, Y. Peng, I. Soboroff, J. Mayfield, and A. Boughannam. Jackal: A Java-based tool for agent development. In J. Baxter and C. Brian Logan, editors, *Working Notes of the Workshop on Tools for Developing Agents, AAAI '98*, number WS-98-10 in AAAI Technical Reports, pages 73–82, Minneapolis, Minnesota, July 1998. AAAI, AAAI Press.

[14] R. S. Cost, I. Soboroff, J. Lakhani, T. Finin, E. Miller, and C. Nicholas. TKQML: A scripting tool for building agents. In M. Wooldridge, M. Singh, and A. Rao, editors, *Intelligent Agents Volume IV – Proceedings of the 1997 Workshop on Agent Theories, Architectures and Languages*, volume 1365 of *Lecture Notes in Artificial Intelligence*, pages 336–340. Springer-Verlag, Berlin, 1997.

[15] G. Crowder and C. Nicholas. Resource selection in CAFE: An architecture for network information retrieval. In *Proceedings of the Network Information Retrieval Workshop, SIGIR 96*, August 1996.

[16] G. Crowder and C. Nicholas. Metadata for distributed text retrieval. In *Proceedings of the Network Information Retrieval Workshop, SIGIR 97*, 1997.

[17] T. Finin, Y. Labrou, and J. Mayfield. KQML as an agent communication language. In J. Bradshaw, editor, *Software Agents*. MIT Press, 1997.

[18] J. C. French, A. L. Powell, J. P. Callan, C. L. Viles, T. Emmitt, K. J. Prey, and Y. Mou. Comparing the performance of database selection algorithms. In *SIGIR*, pages 238–245, 1999.

[19] L. Gravano and H. Garcia-Molina. Generalizing gloss to vector-space databases and broker hierarchies. In *In Proceedings of the 21st VLDB Conference*, Zurich, Switzerland, 1995.

[20] J. Heflin, J. Hendler, and S. Luke. SHOE: A prototype language for the semantic web. *Linkping Electronic Articles in Computer and Information Science, ISSN 1401-9841*, 6, 2001.

[21] J. Hendler. Agents and the semantic web. *IEEE Intelligent Systems*, 16(2):30–37, March/April 2001.

[22] J. Hendler and D. McGuinness. The DARPA agent markup language. *IEEE Intelligent Systems*, 15(6):72–73, November/December 2000.

[23] A. E. Howe and D. Dreilinger. SAVVYSEARCH: A metasearch engine that learns which search engines to query. *AI Magazine*, 18(2):19–25, 1997.

[24] M. E. C. Leah S. Larkey and J. Callan. Collection selection and results merging with topically organized u.s. patents and trec data. In *Conference of Information and Knowledge Management*, 2000.

[25] L. Liu. Query Routing in Large Scale Digital Library Systems. *ICDE, IEEE Press'99*, 1997.

[26] C.-C. K. C. Luis Gravano and H. Garcia-Molina. STARTS: A Stanford proposal for internet Meta-Searching. *ACM SIGMOD '97*, 1997.

[27] W. H. Nicholas Gibbins. Scalability issues for ruery routing service discovery. In *Second Workshop on Infrastructure for Agents, MAS and*

*Scalable MAS at the Fourth International Conference on Autonomous Agents*, pages 209–217, 2001.

[28] C. Pearce and E. Miller. *Advances in Intelligent Hypertext*, chapter The TELLTALE Dynamic Hypertext Environment Approaches to Scalability. Lecture Notes in Computer Science. Springer-Verlag, 1996.

[29] C. Pearce and C. Nicholas. TELLTALE: Experiments in a dynamic hypertext environment for degraded and multilingual data. *Journal of the American Society for Information Science*, June 1994.

[30] A. L. Powell, J. C. French, J. Callan, M. Connell, and C. L. Viles. The impact of database selection on distributed searching. In *SIGIR*, pages 232–239, 2000.

[31] M. Rabinovich, I. Rabinovich, R. Rajaraman, and A. Aggarwal. A dynamic object replication and migration protocol for an internet hosting service. In *International Conference on Distributed Computing Systems*, pages 101–113, 1999.

[32] G. Salton, C. Yang, and A. Wong. A vector space model for automatic indexing. *Communication of the ACM*, pages 613–620, 1975.

[33] M. A. Sheldon, A. Duda, R. Weiss, and D. K. Gifford. Discover: A resource discovery system based on content routing. In *3rd International Conference on WWW*, 1996.

[34] S. Staab, M. Erdmann, and A. Maedche. Ontologies in rdf(s). *Linkping Electronic Articles in Computer and Information Science*, 6, 2001. ISSN 1401-9841.

[35] E. M. Voorhees, N. K. Gupta, and B. Johnson-Laird. The collection fusion problem. In *Text REtrieval Conference*, 1994.

[36] E. M. Voorhees, N. K. Gupta, and B. Johnson-Laird. Learning collection fusion strategies. In *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Fusion Strategies, pages 172–179, 1995.

[37] W3C Working Group. W3C resource description framework (rdf), October 1998. site: http://www.w3c.org/RDF.

[38] W3C Working Group. W3C resource description framework model and syntax specification, February 1999.

[39] W3C Working Group. W3C resource description framework schema (rdfs), March 1999. site: http://www.w3.org/TR/rdf-schema/.

[40] W3C Working Group. extensible markup language (xml), October 2000. site: http://www.w3.org/XML.

[41] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Van Nostrand Reinhold, 1994.

[42] R. R. Yager and A. Rybalov. On the fusion of documents from multiple collection information retrieval systems. *Journal of the American Society for Information Science*, 49(13):1177, November 1998.