Scalability Issues for Query Routing Service Discovery

Nicholas Gibbins
Intelligence, Agents, Multimedia
Dept. of Electronics and Computer Science
University of Southampton, United Kingdom
nmg@ecs.soton.ac.uk

Wendy Hall
Intelligence, Agents, Multimedia
Dept. of Electronics and Computer Science
University of Southampton, United Kingdom
wh@ecs.soton.ac.uk

ABSTRACT
In this paper, we discuss the relationship between mediator-based systems for service discovery in multi-agent systems, and the technique of query routing used for resource discovery in distributed information systems. We then construct a model of the query routing task which we use to examine the complexity and scalability characteristics of a number of commonly encountered architectures for resource or service discovery.

Keywords
Query routing, service discovery, resource discovery, mediator, middle agent, multi-agent system, scalability

1. INTRODUCTION
One of the most common problems encountered in multi-agent systems (MAS) is that of locating those agents which can provide a service that you need. This is variously described as the connection problem [6] or the *service discovery* problem. Multi-agent systems can be characterised by the flexible interactions of their component agents in ways which were not necessarily foreseen at design time[14]. A *service discovery service* is thus a necessary prerequisite for multi-agent systems, particularly those which involve ad-hoc interactions between agents which have no a priori awareness (in [10], the author notes that such services are essential components of a multi-agent system infrastructure). The scalability of such systems must therefore depend to some extent on the scalability of the means by which these systems perform the service discovery task. For this reason, we believe that the development of scalable techniques for service discovery is of key importance in the design of large scale multi-agent systems.

A key component of multi-agent systems which address the service discovery problem are the *mediators*, or *middle agents*. [5] These tackle service discovery by maintaining models of the capabilities of other agents, which are used to assess the likely ability of a given agent to satisfy a service request, with the goal that service requests are only passed to those agents which are best able to satisfy them.

The service discovery task is closely related to the *resource discovery* task, commonly found in distributed information systems such as the World Wide Web. The resource discovery task is that of searching a distributed information system in order to locate a resource (document, file, image) with certain characteristics that satisfy the query criteria. Resource discovery can be viewed as a specialisation of service discovery; a resource discovery search for a data object which satisfies some query criteria can be viewed as a service discovery search for an agent which can provide access to data objects which satisfy those criteria.[1]

The size and growth of the World Wide Web have demonstrated that resource discovery systems which rely on a single mediator (Altavista, for example) do not scale well, [15] yet agent systems commonly rely on a single mediator for service discovery. The evolution and scalability of systems for searching the World Wide Web offer valuable insights which are of use when designing scalable systems of mediators for agent service discovery.

In particular, recent developments in peer-to-peer file sharing (which perform the resource discovery task) have concentrated on truly distributed systems, of which Freenet [3] and Gnutella [1] are prime examples; in these systems, all servers interact as peers with no imposed structure. These systems are characterised by the ad-hoc nature of the interactions between their loosely coupled components, which makes them of some interest to the agent community.

In the next section, we give an overview of a technique known as *query routing*, which is one possible approach to distributing the service or resource discovery task, and describe the relation between query routing and the use of mediator agents. To date, there have been few complexity or scalability studies of query routing, and the majority of these have been empirical studies of particular systems. The results of these studies are not as generally applicable as might be desired, because the choice of dataset (the distribution of data within, and topology of the query routing system) affects the behaviour of the system to a great extent. We believe that there is a need for a more analytical study

[1] For clarity, we describe query routing systems in terms of resource discovery only in the remainder of this paper.

of these systems which would offer more generally applicable results.

In this paper we therefore present a simple model of a query routing system which we have used to examine the complexity and scalability of a number of system topologies which are commonly encountered in query routing or mediator systems.

2. QUERY ROUTING

Query routing is a type of informed distributed search that is commonly used for resource discovery, and was first defined in the WHOIS++ project [7]. Searching for resources in a distributed environment requires that a query be evaluated (logically, at least) on each server which holds resources in the distributed system. In a query routing system, the servers which process queries may contain descriptions in summary form of the contents of other servers. These summary descriptions are known as *forward knowledge*, and can be viewed as partial knowledge of the capabilities of other servers.

When a client wishes to search a query routing system, it does not exhaustively query each of the servers in the system (possibly by a broadcast), instead choosing to query a single server which then redirects and replicates the query through the remainder of the system (and so to the servers which hold the desired resources).

Forward knowledge is used to prune the search space by restricting the scope of the search (the servers to which the query is replicated or redirected) to only those servers which are likely to be able to satisfy the query. A server s in a query routing system is able to compare a query with a forward knowledge expression to determine whether the server s' about which it holds the forward knowledge will be able to satisfy the query. If the comparison indicates that s' is able to satisfy the query, s passes on the processing of the query to it.

We believe that query routing has much in common with the use of mediator agents for service discovery. The capability model of a mediator agent is equivalent to the forward knowledge of query routing servers, and is used for the same purpose, namely restricting the scope of search (other mechanisms for agent service discovery, such as the agent location mechanism presented in [23], use a per-agent list of acquaintances to replicate a query, but do not use capability models to restrict the search). Similarly, the notions of matchmaking and brokering introduced in [6] are equivalent to the notions of referral and delegation used in query routing systems.

In *matchmaking* or *referral* (Figure 1), a client issues a query to a mediator which responds with a reference to a server which is able to answer that query. The client then communicates the same request directly to the indicated server.

In *brokering* or *delegation* (Figure 2), the client issues a query to the mediator as before, but the mediator then passes that query directly to a server which is able to satisfy it. When the mediator receives a response, it passes it back to the client.

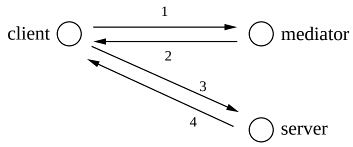

Figure 1: Matchmaking or referral

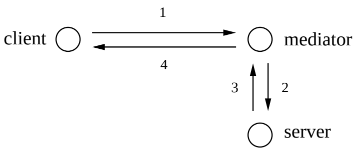

Figure 2: Brokering or delegation

The forward knowledge which is used by a mediator to generate referrals or delegate queries is the result of some a priori exchange of data between the server and the mediator. This communication may be classified in terms of two independent qualities of the exchange.

The first quality concerns the entity which initiated the data exchange; if the exchange were initiated by the mediator, we call this a pull model of forward knowledge distribution, and if it were initiated by the server, we call it a push model. The choice of push versus pull has an important effect on the latency of the forward knowledge; if the pull model is chosen, there is a tradeoff between the cost of the mediator's polls of the server (the more frequent, the more costly) and the potential obsolescence of the forward knowledge (infrequent polling may lead to increasingly out of date information).

Secondly, the forward knowledge is a summary of the contents of a server; this summarisation may be carried out by the server itself, or may be carried out by the mediator. In the former case, there is a potential problem if different summarisation techniques are applied by different servers because the mediator may not be able to combine those summaries. Summarisation by the mediator may avoid the problem of integrating dissimilar summaries, but is more costly to perform since the mediator must fetch the entire contents of the server in order to build the summary.

These considerations apply to query routing and mediator systems equally, but there is one important difference between the two types of system. In a query routing system, there is a frequent assumption made that a mediator may hold forward knowledge about the capabilities of another mediator as well as about resource servers (we could therefore consider query routing systems to be a generalisation of mediator systems).

In this case, the forward knowledge does not represent an ability to service a request per se, but rather to redirect the query to another entity which can process the request. An item of forward knowledge about a mediator is therefore a summary of the forward knowledge that mediator holds about other entities. When forward knowledge about mediators is used in a query routing system, a query may be passed through a chain of mediators before arriving at a server which can satisfy it.

For such a mediator chain to exist, each mediator must hold an item of forward knowledge about its successor which will satisfy the query; if this is not so, the processing of the query will terminate prematurely, that is, before it reaches the server at the end of the chain. If the mediators in the system are assigned forward knowledge randomly, there is no guarantee that a valid chain of mediators will exists for a given query.

Therefore, in our discussion of topologies for query routing (Section 4), we consider schemes for the distribution of forward knowledge in which, for all queries that can be satisfied by some server in a system, there is a valid chain from the initially queried mediator to those servers which can satisfy the query. Our discussion of the scalability of various query routing systems must not only include an analysis of the complexity of processing a query, but also of constructing the network of forward knowledge which is necessary for query processing to take place.

3. A QUERY ROUTING SEARCH MODEL

Our model of a query routing system comprises two parts; a model of the process by which queries are matched against forward knowledge expressions, and a model of the graph topology of a forward knowledge network. Together, these allow us to identify the necessary conditions for query satisfaction and lay a foundation for exploring the effects of different forward knowledge topologies on scalability.

3.1 Query/Forward Knowledge Matching

We have constructed a simple model for the matching process by combining and adapting a traditional set-theoretical model of information retrieval [22] and the description logic $\mathcal{ALU}$.[8]

The information retrieval model consists of three sets: a set R of resources, a set D of resource descriptions and a set Q of queries (see Figure 3). These sets are related by the function $X : R \to D$, which assigns descriptions to resources, the function $F : Q \to 2^D$, which maps a query onto the descriptions of relevant resources, and the function $T : Q \to 2^R$, which combines the previous two functions to model the overall mapping from queries to the sets of resources which are relevant to those queries.

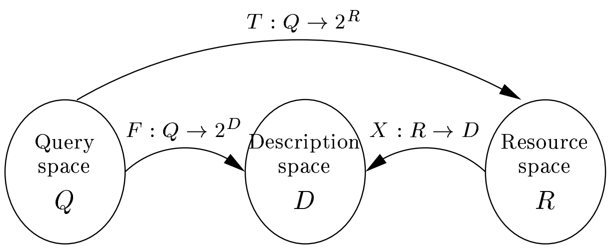

Figure 3: Set-theoretic IR Model

Our model is a modification of the traditional model which combines the query and description sets to give a unified set of resource descriptions; we treat queries as partial descriptions that incompletely describe several resources. Similarly, we treat the forward knowledge summaries as partial descriptions, because these by definition describe several resources.

Syntax	Semantics	Description
A	$A^{\mathcal{I}} \subseteq \Delta$	primitive concept
R	$R^{\mathcal{I}} \subseteq \Delta \times \Delta$	primitive role
$\top$	Δ	top
$\bot$	$\emptyset$	bottom
$\neg C$	$\Delta \setminus C^{\mathcal{I}}$	concept complement
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	concept conjunction
$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	concept disjunction
$\forall R.C$	$\{x : \forall y.R^{\mathcal{I}}(x,y) \Rightarrow C^{\mathcal{I}}(y)\}$	universal quant.
$\exists R$	$\{x : \exists y.R^{\mathcal{I}}(x,y)\}$	existential quant.

Table 1: Syntax and semantics of $\mathcal{ALU}$

We have modelled the matching process using the description logic $\mathcal{ALU}$; in our model, the domain of discourse, Δ, contains the resources about which queries are made, while the resource descriptions and queries are concepts of the logic (denoted by A, C or D in Table 1). The interpretation function $\cdot^{\mathcal{I}}$, which gives a denotational semantics for the logic by mapping concepts onto their extensions (subsets of Δ), corresponds to the query-resource mapping T in the IR model.

Thus, the resources which are relevant to a query expression are those which lie within the extension of that query, and the resources which are summarised by a forward knowledge expression are those that lie within its extension. It should be noted that this notion of relevance is an objective one, and more importantly, is a binary property; a resource either is or is not relevant to a query. This is at odds with many of the definitions of relevance in the literature (summarised in [18]), which are largely subjective in nature, being concerned with the user's self-assessment of their information need and the degree to which the retrieved resources satisfy this need, and classify a resource's relevance on a sliding scale.

However, subjective relevance is of most use in systems which offer full text searching on largely unstructured documents, but many of the query routing resource discovery or service discovery systems contain only highly structured resources such as ontological capability descriptions. With these sorts of resources, it is possible to give an objective definition of relevance based on structural matchings such as unification or subsumption.

In our model, the set of concepts whose extensions are singleton sets corresponds to the set D in the traditional model; these are the maximally specific resource descriptions which describe one and only one resource (these descriptions may be thought of as analogous to ground terms in Prolog). All of the other concepts of the logic correspond to queries or forward knowledge, being more general expressions which may describe more than one resource (compare these expressions with non-ground terms in Prolog). We construct a forward knowledge expression which describes a set of resources by taking the most specific generalisation (the disjunction, $\sqcup$) of the singleton concepts which are the descriptions of those resources.

A forward knowledge summary of a set of resources is therefore an expression which subsumes ($\sqsupseteq$, see (1)) all of the maximally specific descriptions of those resources. We may therefore use subsumption as a relevance test between a query and a maximally specific resource description; a resource is relevant to a query (ie. is in the extension of the query) if the maximally specific description of the resource is subsumed by the query.

$$C \sqsupseteq D \iff C^{\mathcal{I}} \supseteq D^{\mathcal{I}} \tag{1}$$

Choosing a relevance test for a query and a forward knowledge expression is more difficult, not least because there is some variety amongst existing query routing systems. One group of systems uses subsumption as a relevance test for forward knowledge; this group includes Nomenclator [21] and RWhois [26]. Another group of systems, which includes WHOIS++, tests to see if the two expressions share some elements of their extensions. We introduce the symmetric $\Box$ relation (2) to represent this relevance test.

$$C \Box D \iff C^{\mathcal{I}} \cap D^{\mathcal{I}} \neq \emptyset \tag{2}$$

This relation completes our simple model of resource descriptions, forward knowledge summaries and queries and the notion of relevance of the former to the latter.

3.2 The Forward Knowledge Graph

If a query that has been presented to a query routing system is to be satisfied, it must be possible to identify a chain of mediators between the initial server (where the query was asked) and the target server (where relevant resources are located) which can generate the necessary referrals or delegations. For these referrals to be generated, each of the forward knowledge expressions from which they were generated must be relevant to the query. As a precursor to examining the effects of forward knowledge distribution on the complexity of a query routing system, we can model the forward knowledge and identify some useful properties by considering the graph made by the forward knowledge.

For our model of the forward knowledge network, we have adapted the path calculus described in [4], and take a directed multigraph $G = \langle V, E\rangle$ whose nodes and edges are labelled with concepts from the description logic used to model the matching between queries and forward knowledge. We represent these labellings with the functions $\nu : V \to C$ (the node labels which summarise the resources held by a particular node) and $\lambda : V \times V \to C$ (the edge labels which represent the forward knowledge held by one node for another), where C is the set of concepts expressed in the description logic. Because λ is defined over the domain $V \times V$, we take $\lambda(u, v) = \perp$ if $\langle u, v\rangle \notin E$.

This notion of edge labels may be extended to label paths (sequences of adjacent edges) in the forward knowledge graph. A path should be labelled with the most general expression that will satisfy all of the edges in the path; if this expression is relevant to a query, sufficient referrals will be generated to allow the propagation of the query to the path end (which presumably holds resources which are relevant to the query). If we have a path $v_0 \leadsto^p v_n$ (Figure 4), the $\sqcap$ operator (Table 1) can be used as an extension operator to give a path label determined by:

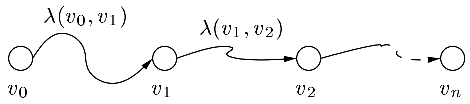

Figure 4: Path extension

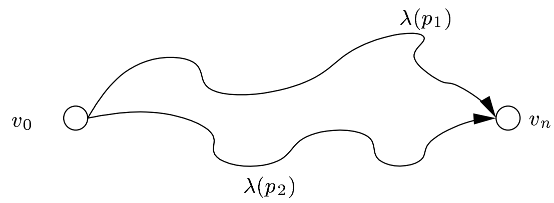

Figure 5: Path summary for parallel paths

$$\lambda(p) = \lambda(v_0, v_1) \sqcap \lambda(v_1, v_2) \sqcap \cdots \sqcap \lambda(v_{n-1}, v_n) \tag{3}$$

Although the edges would be traversed in order from $\langle v_0, v_1\rangle$ to $\langle v_{n-1}, v_n\rangle$ when a query is processed in a real system, $\sqcap$ is associative on C.

The forward knowledge graph may contain more than one path from the initial node to the target node, so we must also define a summary operator which can combine the labels of several paths in order to give an expression which denotes the queries which can be answered by the target node, starting at the initial node. For systems which use the symmetric extension-sharing relation ($\Box$) as their relevance relation, we can use the most specific generalisation ($\sqcup$) as our summary operator. The aggregate label $\lambda(v_0 \leadsto^* v_n)$ for all of the paths $v_0 \leadsto v_n$ is given by:

$$\lambda(v_0 \overset{*}{\leadsto} v_n) = \bigsqcup_{v_0 \leadsto^p v_n} \lambda(p) \tag{4}$$

However, $\sqcup$ is not immediately appropriate as a summary operator for those systems which use subsumption ($\sqsubseteq$) as their relevance relation. If we envisage a graph with two paths $v_0 \leadsto^{p_1} v_n$ and $v_0 \leadsto^{p_2} v_n$, labelled with λ (Figure 5), the summary label of the two paths is $\lambda(p_1) \sqcup \lambda(p_2)$. If we formulate a query $q \in C$ such that $q \sqsubseteq \lambda(p_1) \sqcup \lambda(p_2)$, $q \not\sqsubseteq \lambda(p_1)$ and $q \not\sqsubseteq \lambda(p_2)$, q cannot be satisfied by v_n from v_0 because there is no single path that can be traversed, despite what the summary label suggests. If we allow queries to be decomposed into smaller queries, $\sqcup$ does hold as a summary operator. If we formulate $q \in C$ and then break it down into $q_1, q_2 \in C$ such that $q = q_1 \sqcup q_2$, $q_1 \sqsubseteq \lambda(p_1)$ and $q_2 \sqsubseteq \lambda(p_2)$, each of the two parts of the query can be satisfied by traversing a different path to v_n from v_0.

The summary operator may also be used to combine diverging paths, so as to represent all of the routes which may be

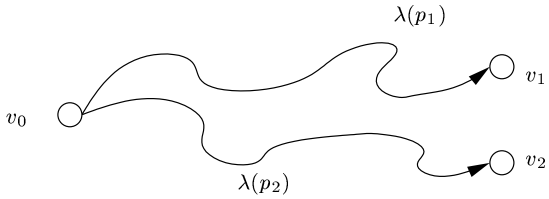

Figure 6: Path summary for diverging paths

taken from the initial node. For example, in Figure 6, the summary of the paths is $\lambda(p_1) \sqcup \lambda(p_2)$. Any query to which this expression is relevant will be able to traverse at least one of the paths from v_0.

4. SCALABILITY OF QUERY ROUTING TOPOLOGIES

The organisation of forward knowledge within a query routing search system plays a great part in the effectiveness, efficiency and scalability of the system. Forward knowledge should direct a query to the relevant servers with less effort than if the query were sent to all servers (an exhaustive search), and ideally the effort expended should grow more slowly than the number of servers as the system is expanded. Similarly, the effort required to construct a suitable network of forward knowledge must also be considered in a study of the scalability of the system.

The order implicit in this network of forward knowledge is also important; the majority of existing query routing search systems presuppose the servers to have been arranged in a hierarchical manner with an omniscient root server (or group of servers) indirectly aware of the contents of every other server. This has been demonstrated to scale well in systems such as the Domain Name System [19], but it requires that a degree of control be exerted over the servers to force them into a hierarchy. A different approach treats all servers as peers and models the interactions between them as if they were social acquaintances; the majority of servers know only about their close neighbours, but a few servers have knowledge of more distant servers. The resulting graph is a disordered one, but makes no assumptions about the existence of a central authority which controls the structure of the network.

In both types of system, the key aim is that a properly labelled path exists from the server by which a query enters the graph, to the server or servers which can satisfy that query. With ordered network topologies it is possible to provide simple rules by which forward knowledge is passed, whereas disordered networks benefit from flooding techniques akin to those used by conventional routing algorithms.

4.1 Efficiency Measures

Information retrieval research has traditionally focussed on effectiveness as a measure for the assessment of information retrieval systems, so concentrating on improving the accuracy of the results that such a system returns. Indeed, efficiency is commonly considered to be a lesser concern to effectiveness because a system which returns poor quality results is of little use, regardless of how quickly it returns those results. A recent study [9] noted that one recent collection of seminal research papers [24] did not contain a single paper on efficiency considerations.

The advent and rise of distributed information retrieval has changed this state of affairs to some extent. Web search engines are under considerable pressure to deliver their results quickly; the process of evaluating a query is perceived by many users as being no more complex than that of following a link. Similarly, the impact that the information retrieval system has on its environment (its use of network resources while building its index and processing queries, for example) is vitally important. A web search engine which places a heavy load on the network would be deemed unacceptable and antisocial.

Existing studies of information retrieval efficiency have made use of measurements such as the time taken to process queries in seconds or the size of an index in bytes. While wholly appropriate for an empirical study of an existing system, they are less so when modelling the behaviour of a hypothetical system. We use more abstract complexity measures which, while strongly related to the real-world measurements, are more amenable to a study which does not consider the added modelling complexities of network latency and the like.

4.1.1 Query Complexity

When a query is submitted to the system, the number of request and response messages which are generated between system components (the message traffic) is a measure of the communication complexity of query processing. The rate at which this grows relative to the number of servers in the network gives an indication of its scalability. In addition, the individual message traffic for each server in the system can be used to show the presence of bottlenecks, servers which contribute disproportionately to the global message traffic. We do not explicitly specify whether the system operates by means of referrals or by delegation since a single referral generates as many messages as a single delegation. However, a system which operates by referrals is less prone to making redundant queries (where a server is queried more than once) because the state of the query is stored in one location (the client), making it easy to remove duplicates. A system which uses delegation could be expected to send more redundant messages, but the exact effect cannot be quantified without more detailed knowledge of the exact structure of the forward knowledge graph and the query being asked.

We consider our hypothetical systems to operate in synchronous parallel rounds, one round being the time for each entity in the network to communicate with one or more of its neighbours. The running time (time complexity) for query processing is the number of rounds required until the client receives an answer to its query.

4.1.2 Control Complexity

Although the message traffic generated by submitting queries can be used to measure system load under normal operations, it would not be possible to process queries at all if there were no forward knowledge in the system. This forward knowledge must be built up by passing messages which contain forward knowledge summaries between servers.

The forward knowledge graph affects the query message traffic by constraining the types of referral which may be issued, but the more information the graph contains, the more messages must be exchanged initially in order to construct the graph. These messages are equivalent to the control messages which allow a communications network to build its routing tables; the number of control messages sent by the system gives the communication complexity of the forward knowledge building operation. We consider only those control messages sent when a system is started; control messages sent as the result of dynamic changes over time to the resources held by the servers are not considered.

As with query message traffic above, we measure time complexity by counting the number of rounds taken for the system to complete its task. In this case, the system must *converge*, reach a state where each server has sufficient forward knowledge that all resources that are relevant to a query are reachable).

4.1.3 Routing Table Size

If the number of control messages shows the communications overhead inherent in building the forward knowledge graph, the size of the routing tables held by each server as a result of the control messages above shows the space required to store the routing information. A system's control message traffic could be largely redundant, so a server would not retain each message it received; this measure gives the size of the 'useful' control information.

4.2 Ordered Mediator Networks

4.2.1 Single Mediator

This simplest hierarchical network for query routing consists of a set of servers which hold the resources in the system and a single mediator which contains a summary of the contents of each server (as shown in Figure 7). Queries are submitted to the mediator, which uses its forward knowledge to propagate the query to the relevant servers. This arrangement is used by the file-sharing service Napster and most Internet meta-search engines.

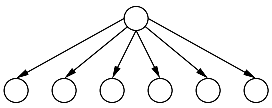

Figure 7: Single Mediator

We model this system as a directed graph $\langle V, E\rangle$ in which there is a distinguished $v_0 \in V$ which is the single mediator and all edges in E are of the form $\langle v_0, v\rangle$. The contents of the servers (the elements of the set $V \setminus \{v_0\}$) are denoted by the vertex labelling $\nu : V \to C$. Each data server passes a summary of its contents to the mediator, so $\forall v \in V, v \neq v_0, \lambda(\langle v_0, v\rangle) = \nu(v)$. It is therefore simple to show that, given a query $q \in C$ which can be satisfied by the contents of a server, there exists a path (edge) from the mediator to that server whose label also satisfies q.

When a query which may be satisfied by a data server is submitted to the system, the generated query message traffic is constant with increasing system size (one request and response with the mediator and the chosen server), as is the running time. Although query complexity is low, the mediator is a bottleneck, since all queries presented to the system must be processed by it. The number of control messages pushed to the mediator scales as $O(|V|)$, the number of servers in the system. The number of entries in the routing table held by the mediator also scales as $O(|V|)$.

4.2.2 Hierarchical Mediators

Although the system above is capable of routing queries to relevant data servers, it is unlikely to scale well as the number of data servers increases. The limiting factors are most likely to be the size of the mediator's routing table and the disproportionate load that it is under.

It is natural to expand the system to a multi-layer hierarchy in order to address the first of these concerns (X.500 [13] and RWhois [26] being good examples of this). The lowest layer of the system consists of servers which pass forward knowledge up the tree to mediators that summarise it and pass it in turn to their parent. Clients present their queries to the system as a whole by sending them to the root mediator, which forwards the query to those second-level mediators that might be able to pass it on to appropriate data servers (and so on, until the query reaches the servers).

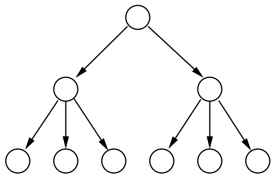

Figure 8: Hierarchical Mediators

Let $G = \langle V, E\rangle$ be a directed layered graph such that the set V of vertices is partitioned into the sets $L_0, L_1, \ldots, L_n$ and all edges $e \in E$ are of the form $\langle v, w\rangle, v \in L_i, w \in L_{i+1}$. The set L_0 contains the root elements of the graph; for this system we take L_0 to be a singleton, so the graph is single-rooted. We denote the root vertex by v_0. For convenience we define the function $c(v) = \{x \in V : \langle v, x\rangle \in E\}$ which maps a vertex onto its children and the function $p(v) = x, \langle x, v\rangle \in E$ which maps a vertex onto its parent. The vertices in V are labelled with the function $\nu : V \to C$, which describes the contents of the servers in L_n with a summary of their resources and describes all mediators as having no content. As in the previous case, the edges are labelled with the function $\lambda : E \to C$. The function differs from its previous definition by adding a summary of the edge labels from the layer below.

$$\lambda(u, v) = \nu(v) \sqcup \bigsqcup_{x \in c(v)} \lambda(v, x) \tag{5}$$

If we take a query $q \in C$ for which some data server $v \in L_n$ contains a relevant document (which we can tell if $q \sqsubseteq \nu(v)$), there will be a chain of forward knowledge from the root server to this data server which will satisfy q if $q \sqsubseteq \lambda(v_0 \rightsquigarrow v)$.

When a query which may be satisfied by a single data server is submitted to the system, the generated query message traffic and query running time scale as $O(\log|V|)$, the depth of the hierarchy. The number of control messages sent initially scales as $O(|V|)$, as each server and mediator passes forward knowledge toward the root while the control running time scales as $O(\log|V|)$. The mean number of entries in the routing table held by each mediator, including the root, is constant with increasing system size, being equal to the breadth b of the hierarchy (the number of direct children of a node).

As with the single mediator system, the root mediator is a bottleneck because it must process all messages. We can partially rectify this situation by allowing control messages which pass down the hierarchy as well as up, so that servers have forward knowledge about the capabilities of mediators (so doubling the number of control messages and the control running time). We represent this by labelling the transpose graph such that:

$$\lambda(v,u) = \lambda(v,p(v)) \sqcup \bigsqcup_{x \in c(v) \setminus \{u\}} \lambda(v,x) \tag{6}$$

This allows queries to be started at leaf servers and referred upwards until they reach the root of the subtree which contains the relevant resources, and reduces the number of queries which are handled on the root server by a constant factor b (assuming an even distribution of queries).

4.2.3 Complete Mediator Networks

In this system, all the servers know about the contents of all of the other servers (see Figure 9). The message traffic generated by queries in this system is constant (4 messages) and takes constant time to process. Each server requires a routing table of size $O(|V|)$ and the control message traffic scales as $O(|V|^2)$ because the graph contains $|V|(|V|-1)$ edges. The running time for control messages is constant (one round).

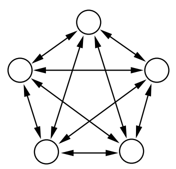

Figure 9: Complete Network of Mediators

4.2.4 Mediator Councils

Councils are a cross between complete networks and hierarchies that were introduced in [16] (Figure 10). The servers are divided into $|V|/b$ groups of b members each. Within a group, there is complete forward knowledge; every server knows about the contents of every other server. Each group is represented in a council by a chosen member from that group. This pattern is repeated in the councils; a council is also fully connected and makes representation to a higher level council, and so on.

Provided that $b \ll |V|$, a council has the same control complexity for queries and control message as a hierarchy. As

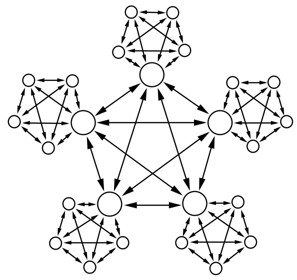

Figure 10: Mediator Councils

b grows, the effort required to exchange control messages with group peers grows and the control message complexity approaches $O(|V|^2)$, that of a complete network (the running time is $O(\log|V|)$ regardless of b). The query message complexity and running time for a council are $O(\log|V|)$). For each group in which a server participates, it will have a constant b entries in its routing table. In the worst case a server will be a representative in groups at all levels; here the server's routing table size will scale as $O(\log|V|)$. Also, because such a server handles a disproportionately large number of messages (being responsible for gatewaying messages between subtrees), it is a bottleneck, though not to as great an extent as the root server in a hierarchical network (here, the load is shared with its peer servers).

4.3 Disordered Mediator Networks

While ordered, hierarchical networks are a simple abstraction for studying the behaviour of query routing systems, it should be noted that similar real world systems are, for the most part, not ordered. We base our models of disordered systems of forward knowledge on two families of random graph, *small-world networks* [25] and *scale free networks* [2], which have been used to characterise complex networks such as acquaintance networks and the hyperlink structure of the World Wide Web.

Small-world networks are ordered graphs (in [25], ring lattices, but could also be rectangular lattices as used in [23]) in which each edge may be rewired at random with a given probability, giving a graph which has a large degree of local order, but also contains a few 'long distance' edges. In contrast, a scale free network starts with a small complete graph and adds vertices one at a time, linking them to existing vertices at random, weighted by the degree of the existing vertices so that new vertices are preferentially linked to well-connected vertices. This rich-get-richer behaviour results in a graph with a very few well-connected vertices, a slightly larger number of less well-connected vertices and so on. Although they are constructed in different ways, these two graph families share the property of having a low graph diameter (scales as $\log|V|$).

We will assume that the servers in our systems are not clustered by interest or ability, but form neighbourhoods based on some real world or network distance criteria. The edges in the disordered graph show basic mutual awareness between servers. These edges may be chained using traditional network routing techniques to produce shortest paths to the other servers. We consider two common families of routing algorithms, *Distance Vector* (also known as Bellman-Ford) and *Link State*.

There is a fundamental difference between network routing and query routing, namely the treatment of addresses. In network routing, an address is a unique object used to identify a server. The routing process takes an address and constructs a path from the source to the destination server which bears that address; each server's routing tables are indexed by these destination addresses. In contrast, query routing takes a query expression, which does not uniquely identify a server, and attempts to construct paths from the source to each destination server which can satisfy the query expression. The routing tables may therefore contain more than one entry for a given expression, but this presents a problem while the routing tables are being constructed; do two entries with the same expression refer to the contents of different servers (and so should be retained as separate entries), or are they both references to the same server (and should be contracted into a single entry)?

This ambiguity may be resolved by including both the server's content summary (or capability description) and its name in the routing table, but this brings to light a different question. If a server's routing table contains the name of a data server, why can't the query take a shortcut and be sent to the destination directly without having to traverse the path generated by the routing process? If the query is sent directly to the destination, none of the non-routing knowledge accumulated by the intermediate servers (namely, cached answers) may be brought to bear on the query.

4.3.1 Distance Vector

Distance Vector (DV) is a distributed routing algorithm used widely on the Internet (as an implementation called the Routing Information Protocol [12]). Each server sends periodic updates to its neighbours on its estimates of path lengths to each destination and maintains a routing table containing the shortest paths to those destinations (routing table size is $O(|V|)$). When the system converges, the query message traffic and running time are proportional to the graph diameter (ie. scale as $O(\log |V|)$), but DV is an extremely expensive and poorly scaling algorithm in terms of control messages, creating traffic of $O(|V||E|)$ and converging in $O(|V|)$ rounds.[17]

4.3.2 Link State

Link State (LS) algorithms, of which Open Shortest Path First [20] is an example, flood a description of the local network topology to all the servers in the system, each of which calculates a routing table based on this complete topological knowledge. The query complexity and running time are as that for DV, but the control complexity is that of the flooding operation. The message traffic for flooding is given in [11] as $\Theta(|E| \log |V|)$ with convergence in $O(\log |V|)$ (assuming a graph diameter that scales as $\log |V|$), but it is possible to use other algorithms that accomplish the same task with more amenable complexity (such as the Name-Dropper algorithm introduced in that paper, which has communications complexity of $O(|V| \log^2 |V|)$ and converges in $O(\log^2 |V|)$ rounds).

5. CONCLUSION

In this paper, we have examined the complexity and scalability of a number of network topologies for use in query routing systems for resource or service discovery (see Table 2 for a summary), and to this end have constructed a suitably abstract model of the query routing process.

Of the ordered networks we have studied, the council offers a reasonable compromise between the efficiency of a purely hierarchical system and the bottleneck at its root, provided that the group size is kept much smaller than the overall system size. However, ordered systems have the drawback that it may be costly or politically inexpedient to organise such a system; scalable ordered systems rely on some centralised components for their operation, and the responsibility for running these may be too large to entrust them to an 'ordinary user' (the administrator of a simple leaf server, for example). The centralised components make these systems vulnerable to deliberate attempts to disrupt service, as evinced by the recent closure of the Napster service.

Existing fully distributed systems for peer-to-peer file sharing, such as Gnutella and Freenet, have relied on message flooding not only for creating routing information but also for processing queries, which has lead to scalability problems. Our adaptation of network routing algorithms to query routing suggests that this may be a better approach (when using a link state algorithm, at least) for creating scalable multi-agent systems which use query routing for service or resource discovery.

One area that still requires investigation is the effect of updates on system complexity. For simplicity, we have made the assumption that the capabilities of agents in the system remain static, and that no new agents join once the forward knowledge network has been constructed, but this is unlikely to be the case in a real world system. For this reason, we need to study the complexity of strategies for the propagation of incremental updates in the forward knowledge network of the topologies we have examined in this paper.

6. REFERENCES

[1] Gnutella homepage. `http://gnutella.wego.com/`, 2001.

[2] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(509), 1999.

[3] I. Clarke. A distributed decentralised information storage and retrieval system. BA dissertation, Division of Informatics, University of Edinburgh, 1999.

[4] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[5] K. Decker, K. Sycara, and M. Williamson. Middle-agents for the internet. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, 1997.

[6] K. Decker, M. Williamson, and K. Sycara. Matchmaking and brokering. In *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS-96)*, 1996.

Topology	Control traffic	Control time	Query traffic	Query time	Routing table
single	$O(\|V\|)$	constant	constant	constant	$O(\|V\|)$
hierarchy	$O(\|V\|)$	$O(\log \|V\|)$	$O(\log \|V\|)$	$O(\log \|V\|)$	constant
complete	$O(\|V\|^2)$	constant	constant	constant	$O(\|V\|)$
council	$O(\|V\|)$	$O(\log \|V\|)$	$O(\log \|V\|)$	$O(\log \|V\|)$	constant
distance vector	$O(\|V\|\|E\|)$	$O(\|V\|)$	$O(\log \|V\|)$	$O(\log \|V\|)$	$O(\|V\|)$
link state	$O(\|E\| \log \|V\|)$	$O(\log \|V\|)$	$O(\log \|V\|)$	$O(\log \|V\|)$	$O(\|V\|)$
link state w/ name-dropper	$O(\|V\| \log^2 \|V\|)$	$O(\log^2 \|V\|)$	$O(\log \|V\|)$	$O(\log \|V\|)$	$O(\|V\|)$

Table 2: Complexity summary

[7] P. Deutsch, R. Schoultz, P. Faltstrom, and C. Weider. Architecture of the Whois++ service. RFC 1835, Internet Engineering Task Force, Aug. 1995.

[8] F. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In G. Brewka, editor, *Principles of Knowledge Representation and Reasoning*, Studies in Logic, Language and Information, pages 193–238. CSLI Publications, 1996.

[9] O. Frieder, D. A. Grossman, A. Chowdhury, and G. Frieder. Efficiency considerations for scalable information retrieval servers. *Journal of Digital Information*, 1(5), Dec. 1999.

[10] L. Gasser. MAS infrastructure definitions, needs and prospects. In *Proceedings of the Workshop on Infrastructure for Scalable Multi-Agent Systems at the Fourth International Conference on Autonomous Agents*, 2000.

[11] M. Harchol-Balter, T. Leighton, and D. Lewin. Resource discovery in distributed networks. In *Proceedings of the Eighteenth Annual ACM Symposium on Principles of Distributed Computing*, pages 229–237, 1999.

[12] C. Hedrick. Routing Information Protocol. RFC 1058, Internet Engineering Task Force, 1988.

[13] ITU. Information Technology – Open Systems Interconnection – The directory: Overview of concepts, models and services. ITU Recommendation X.500, International Telecommunication Union, Nov. 1993.

[14] N. R. Jennings. On agent-based software engineering. *Artificial Intelligence*, 117(2):277–296, 2000.

[15] S. Lawrence and C. L. Giles. Accessibility of information on the web. *Nature*, 400:107–109, July 1999.

[16] M. Lejter and T. Dean. A framework for the development of multiagent architectures. *IEEE Expert*, 11(6):47–59, 1996.

[17] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[18] S. Mizzaro. How many relevances in information retrieval? *Interacting With Computers*, 10(3):305–322, 1998.

[19] P. Mockapetris. Domain names - concepts and facilities. RFC 1034, Internet Engineering Task Force, Nov. 1987.

[20] J. Moy. OSPF version 2. RFC 1247, Internet Engineering Task Force, July 1991.

[21] J. Ordille. Internet Nomenclator Project. RFC 2258, Internet Engineering Task Force, Jan. 1998.

[22] G. Salton. Mathematics and information retrieval. *Journal of Documentation*, 35(1):1–29, Mar. 1979.

[23] O. Shehory. A scalable agent location mechanism. In *Intelligent Agents VI - Proceedings of the Sixth International Workshop on Agent Theories, Architectures and Languages (ATAL'99)*, volume 1757 of *Lecture Notes in Artificial Intelligence*, pages 162–172, 1999.

[24] K. Sparck Jones and P. Willett, editors. *Readings in Information Retrieval*. Morgan Kaufmann, 1997.

[25] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

[26] S. Williamson, M. Kosters, D. Blacka, J. Singh, and K. Zeilstra. Referral Whois (RWhois) Protocol v1.5. RFC 2167, Internet Engineering Task Force, June 1997.