# The Information Discovery Graph: Towards a Scalable Multimedia Resource Directory

Nathan R. Sturtevant, Nelson Tang, Lixia Zhang
*University of California Los Angeles*
*{nathanst, tang, lixia}@cs.ucla.edu*

## Abstract

*In this paper, we present the design, rationale, and basic mechanisms of the Information Discovery Graph (IDG), a scalable multimedia resource directory. Facing the fundamental challenge of scaling with both large amounts of resources and large numbers of users, the IDG is made up of a self-organizing hierarchy of information managers, each maintaining resource information for specific topics or areas. Multimedia data sources register themselves with the IDG system and keep the information up-to-date. Preliminary simulation results demonstrate the approach's promise. A number of open research issues are also addressed.*

## 1. Introduction

The Information Discovery Graph (IDG) is a component of the Semantic Multicast project [1]. The goal of Semantic Multicast is to facilitate distributed collaborations by selecting and filtering content available on the network. Selections use criteria such as user interest and network bandwidth, bringing appropriate material to the user's desktop. Semantic Multicast aims at providing a fast and intuitive method for connecting users with content that matches both their interests and their processing capabilities. Specifically, the IDG's role in this framework is to provide the directory and search mechanisms to locate any relevant content, such as video streams stored in a database or a real-time multimedia conference sessions on the MBone [2].

One of the principal goals of the IDG is to be able to handle large amounts of online multimedia information in a scalable fashion. Not only should users be able to find information quickly, regardless of the amount of information available, but the control protocol used to maintain the information directory should also have as little overhead as possible.

The IDG is based on a distributed cache, which can be browsed by topic or searched for specific content. By maintaining the resource information in a sorted mesh of caches, the user can find information quickly. In this paper we will discuss the existing tools and their limitations, and then explore the design of the IDG. Then, we will describe simulations of the IDG that demonstrate its correctness and usefulness, and finally, we will present ideas of future research.

## 2. Current information discovery tools

The IDG design aims at addressing the challenge of information search in large-scale networks. We expect that in the future Internet, orders of magnitude more multimedia contents will be online that cover a diverse range of topics. We therefore begin by exploring the current tools used for discovering online multimedia resources. In particular, we will highlight the limitations of the present tools and architecture.

There are two kinds of multimedia content available on the Internet today: stored multimedia content, and live (real-time) multimedia content. Currently, users find these two kinds of resources by using different sets of tools, and we will describe each set separately.

### 2.1. Stored multimedia content

Information about stored multimedia content is generally not propagated to users and is not categorized using any specific mechanism. In most cases users look for interesting content with a Web search engine. Existing Web search engines perform searches by either counting how often the search terms occur on a page, such as AltaVista's approach, or by matching search terms against a manually chosen category description, such as Yahoo's approach. In the first case, the content must be textual to permit effective searching; however, a large amount of multimedia content is not text-based. In the second case, the categories are fixed and are chosen by the search engine. In both cases the inherent problem of today's Web search engines is also introduced: poor scalability due to a centralized database. Scalability is a problem because crawlers of individual Web search engines must spend an increasing amount of time discovering the new contents that are added online everyday. The task becomes more difficult as the amount of content continues to grow. In

addition, over time many of the Web pages discovered by crawlers get moved to different locations, or deleted entirely. As a result, it has becoming increasingly common to encounter stale links from a search engine's reply. Furthermore, because each search engine works in isolation, a popular Web site often observes visits from multiple crawlers, one for each different search engine, resulting in increased network and Web server load. Finally, the lack of semantic content searching stems from the inability of the search engine to derive semantic meaning from the resource using its text matching algorithms. Web search engines provide a rudimentary mechanism for categorizing stored multimedia content, but it is not a sufficient architecture for handling the presumed growth of online multimedia content in the future.

More recently, another approach to bringing online multimedia content to users is by listing available content on a well-known web server using a proprietary directory, such as Broadcast.com. Broadcast.com uses a centralized, manually-categorized directory of multimedia sources, and users can retrieve interesting content by unicast. This approach suffers similar scalability problems.

### 2.2. Real-time content

Due to its time-sensitive nature, information about real-time multimedia conferences on the Internet, as opposed to stored multimedia content, is multicast to all users. Multicast takes place on the MBone, a virtual network built on top of the Internet where IP multicast has been deployed. Information about multimedia conferences is currently advertised using the Session Announcement Protocol (SAP) [3], which also outlines bandwidth guidelines for session announcements. These guidelines suggest using a fixed amount of bandwidth on a single well-known multicast address to periodically announce existing content. It is the responsibility of a content provider to advertise its content on this multicast address. Client programs, such as SDR [4], list content that has been announced. These advertisements time-out in SDR after some appropriate interval. There are no intermediate agents in the network, so there is no way to learn of the existence of real-time content except by waiting to hear directly from the sources.

This approach results in a number of very good features. SAP provides robustness and automatic error recovery due to its soft-state listings of content and lack of intermediaries. Additionally, the lack of intermediaries means direct source delivery and enables the user to directly select relevant content. However, these advantageous features also result in a few problems with SAP. First, when a new client initially comes online, it must wait patiently to learn about all existing announcements. Currently it can take up to 30 minutes to hear all available sessions [5], and this time can only be expected to grow as the number of content providers grows. The delay is due to the fixed bandwidth used by all session announcements and the extreme sensitivity to announcement packet losses. SAP dictates that only 200 bits per second of global bandwidth be used for content announcements; thus for new clients, a dropped announcement packet can result in a significant delay for learning of the existence of an ongoing session. Second, there is no ordering of listings within the single channel. Although provisions have been made in the Session Description Protocol (SDP) to allow keywords [6], few announcements currently use this option. Accordingly, a user must scroll through a long, flat list to find interesting content. When many more multimedia content providers exist, this approach will not be feasible.

We would like a resource directory that is scalable enough to support many orders of magnitude more resources than are currently supported. It should handle stored and real-time multimedia content equally well, have the simplicity and robustness of SDR, and overcome the lengthy client startup time and reduce sensitivity to packet loss. In addition, items should be grouped according to content, so relevant resources can be easily located. These goals motivate the design of the IDG.

## 3. IDG design

To describe the design of the IDG, we begin by describing the overall organization of the IDG. We then discuss specific features of the IDG design.

### 3.1. Overall organization

The IDG is a directory of multimedia content that helps users rendezvous with data sources with multimedia content in which the user is interested. The directory is organized around a taxonomy, which is a hierarchy based on the semantic information of whatever content data is available. Levels of the hierarchy represent a narrowing of focus of the semantic categories. Levels near the top of the hierarchy represent broad semantic categorizations, such as "sports" or "entertainment," and the categories lower in the hierarchy represent categories that are more specific. Figure 1 shows a high-level view of the IDG organization of a sample taxonomy.

The multimedia content is represented as a **data source**. A record for each data source is inserted into the IDG hierarchy at a semantic category that best categorizes the data source's content. This matching process is described in greater detail in Section 3.2.1. The data source itself is responsible for properly characterizing its content, as it has the most knowledge of the content.

**Users**, whether human beings or software agents, use the IDG to find data sources that serve relevant content

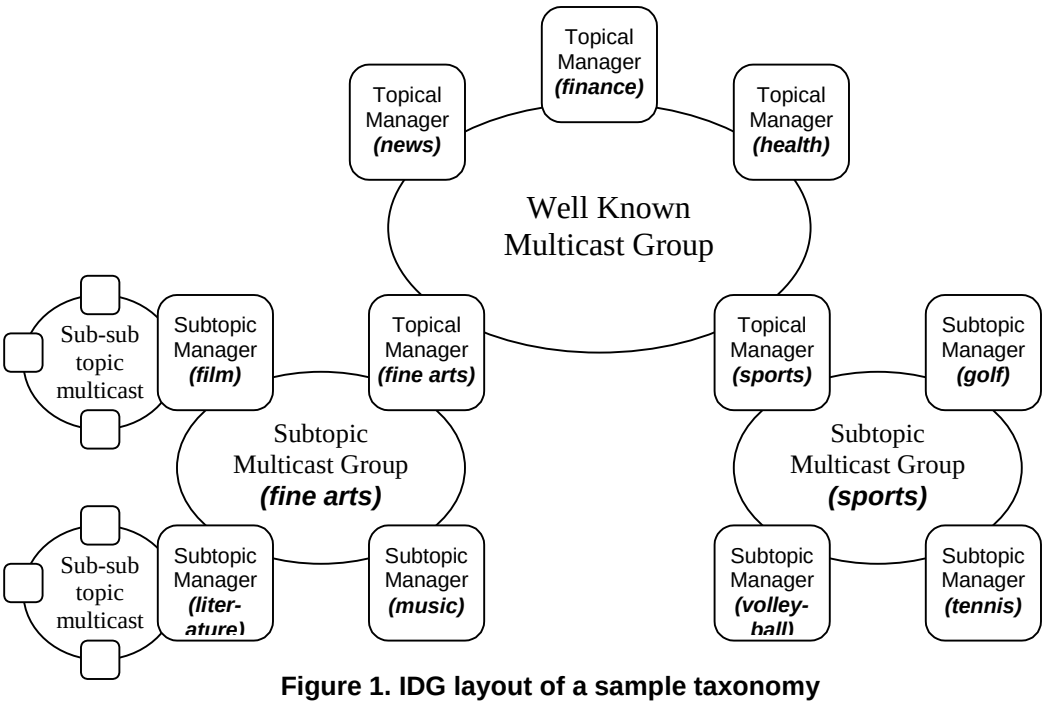


**Figure 1. IDG layout of a sample taxonomy**

data. A user can find relevant data sources in two different ways: the user may browse the IDG for listings of data sources of interest, or the user may send a query for the appropriate category. The exact mechanism is described below in Section 3.2.1. Once a user finds an appropriate data source, it can contact the source directly and receive the content; the IDG is used only to connect data sources and interested users.

**Managers** are responsible for maintaining the semantic categories of the hierarchical taxonomy. The hierarchy is implemented as a collection of multicast groups. A number of managers listen to each multicast group; some managers also listen to two groups, forming the interconnections between the levels of the hierarchy. The categories are merely listings of information of semantically-related data sources. The listings are stored by a manager, and each manager is responsible for one or more given semantic category. Note that managers are logical entities, not physical entities; thus, a single physical machine may house several separate managers.

## 3.2. IDG features

We now discuss some of the features of the IDG in greater detail. In particular, we point out the robust and fluid architecture that allows all users to register their own data sources as particular aspects of our architecture that makes it well suited for directory services.

**3.2.1. Querying the IDG.** Both users and data sources need to locate managers within the IDG; users look for managers to find interesting data sources, and data sources locate the most fitting manager with which to register themselves. This location process is the same for both users and data sources and can happen by one of two methods: manual traversal or querying a manager.

First, the IDG hierarchy can be manually traversed, and a relevant manager manually selected. Traversal means either passively listening to a group for manager descriptions (see Section 3.2.3), or it means asking a group for a description of the hierarchy. Both ways reveal a list of managers listening to a multicast group and to what subgroups, if any, they are connected. If the appropriate manager is not found in this group, the most appropriate subgroup can be selected and recursively traversed. This continues until an appropriate manager is found.

Second, a search query can be given to any manager. This search query attempts to find a match with a manager's semantic categories or keywords listings for its data sources. Managers can directly respond to the query, indicating that the manager can appropriately handle the request, or they may instead forward queries to a sub-multicast group. Additionally, since managers inform their multicast group peers of their descriptions, a manager has some knowledge of its peers, so its responses need not be for just its own categories or data sources.

For example, consider the sample taxonomy shown in Figure 1. Suppose a user wishes to view a broadcast of a golf tournament containing Tiger Woods and David Duval. A query can be given to any topical manager in the well-known multicast group. This manager can use its

cache to match the query with the sports manager, and it will forward the query to the sub-multicast group under the sports manager. At this level, the golf manager would pick up the query and respond directly to the user. This process is controlled by the matching mechanism between queries and managers, which is still under development.

For both manual traversal and querying, users and data sources need not start at the top-level well-known multicast address. Since the upper levels of the hierarchy are not likely to change very often, users may cache their knowledge of the hierarchy and start traversals or queries at lower, more appropriate levels. This caching of the IDG hierarchy provides scalability; users traversing or querying the hierarchy will start at levels much closer to likely data sources, reducing the amount of traffic generated by each user. If the user's cache has stale hierarchy data, the user can simply return to the top-level well-known multicast address. However, since this is the starting point for new users as well, it is imperative to take as much traffic out of the upper hierarchy levels as possible to support scaling of the number of users of the IDG.

**3.2.2. Data source registration.** SAP calls for data sources to broadcast an advertisement of their content to the well-known multicast address on a periodic basis. Our IDG design takes a slightly different approach; data sources do not make announcements to any multicast groups, but rather they register themselves with a manager. (In our design, managers may also share data source registrations with other managers if the content spans multiple semantic topics, allowing for more effective user searches for interesting data. However, the details of this sharing have not been fully explored, and we leave it as an item for future research.)

When a data source registers with a manager, the manager acknowledges the registration and creates a soft-state connection with the data source. The manager will eventually time-out the data source's registration, so the data source must periodically refresh its registration with its manager, which the manager will acknowledge. This process prevents the buildup of stale entries to data sources that are no longer available, and it also allows data sources to know that their managers are still alive, providing a measure of robustness.

It is important that the announcement from a data source be descriptive enough to allow proper classification by managers. The data source itself will be best able to provide such a description. Thus, the responsibility for an accurate characterization of the multimedia data is on the creator of the data source.

**3.2.3. Description announcements.** Since a data source does not directly advertise its content to users, its manager is responsible for doing the advertisement. Managers periodically send their descriptions to the multicast group(s) to which they are connected. These descriptions identify the manager and its place in the hierarchy and also include a list of keywords to characterize the data sources that have registered with the manager.

This serves two purposes: it provides a way for users and other managers to passively learn the IDG hierarchy, and it serves as a heartbeat to other managers to support robust failure recovery. The first aspect lets users learn the IDG hierarchy without having to send queries to any multicast groups or managers; users that monitor a group will eventually hear all manager descriptions for that group at a minimal cost. This very closely approximates the operation of SAP, and thus we can reap the same benefits of simplicity that this design entails. For example, a client program might passively collect information about the IDG while the user is idle. However, our design improves upon SAP by using managers to aggregate and simplify data source announcements into a single, more compact manager description. Users can always get full information on data sources by contacting the manager directly. We thus reduce the global multicast traffic both by using smaller periodic announcement packets and by only sending lengthy data source information, via unicast, to users who request it.

All managers in a multicast group will cache broadcast descriptions in order to share their knowledge of the IDG hierarchy. This means any manager can accurately respond to queries. The manager description also serves as a heartbeat, which is needed to support a robust failure recovery scheme. If a manager is not heard from after some interval, other managers in the same group(s) will assume the manager has failed. At this point, a failure recovery mechanism starts. Another manager in the multicast group (or the "lower" group, if the failed manager bridged two groups) will be elected to take over for the failed manager. "Taking over" entails assuming control of the semantic categories for which the failed manager was responsible. Data sources that were registered with the old manager will re-register with the new manager taking over for the old one. Note that at this point, the new manager may try to recruit more managers to help properly distribute the load (see Section 3.2.4).

**3.2.4. Hierarchy fluidity.** The IDG hierarchy is completely dynamic; it changes as the registered data sources come and go. This means that if a manager reaches some threshold of load or has too many data sources registered, it may create a new semantic subcategory and delegate another manager to handle that subcategory. It will then divest itself of data sources that better fit the subcategory, instructing them to re-register themselves with the new manager. Similarly, if a manager finds itself below some threshold of load or data sources,

it may decide that its category is too specific and is no longer needed. In this case, it will pass any remaining data sources on to a manager of a more general category and then terminate itself. This design scheme provides maximal flexibility and scalability for the IDG; we do not need to fully specify the entire manager hierarchy in advance, as the managers will adapt to whatever data sources there are. As the amount of online multimedia content increases, one can simply add more resources to the information managers pool, and the IDG protocol will utilize as many managers as the work load demands.

## 4. Overhead model

In order to compare our design to the architecture currently in place, we need some sort of metric to compare. We look at two such metrics. First, we consider the time it takes a user to find a relevant data source. Then, we compare the global multicast bandwidth used.

### 4.1. SAP search time

For globally announced sessions, SAP dictates that only 200 bps are used for all announcements. Sources announce their sessions at minimum once every 5 minutes, and announcement times increase directly with the number of sessions. To calculate the time it takes for a user to find the session in which they are interested, we suppose that a user has some sense of what they are interested in. The user will then listen for announcements until the most interesting session announcement arrives. Since it is difficult to predict what the "most interesting session" is, we use the worst case of waiting until all sessions arrive, assuming no packet loss.

More formally, given

$N$ = number of announcements
$R$ = max(300, (8**N***ad_size*)/*limit*), in seconds
*limit* ranges from 200bps to 2Kbps; see [3]

Then, the search time for SAP is

$$ST_{sap} = N*R$$

Thus, the search time is linear in the number of announcements being made.

### 4.2. IDG search time

Using the IDG, the time to get an interesting session is the amount of time it takes to locate a manager containing interesting sessions plus the time to retrieve all data source listings from that manager. The time to locate an interesting manager is the cost of a path from the root of the IDG to that manager. Along that path, we will retrieve all manager listings at each level. Given an average branching factor of $b$ in the IDG and a depth $d$ search, the cost of reaching some particular manager is $b \cdot d$. The exact time to get to this manager will vary by some constant factor. This factor will account for the bandwidth available to the end user and the delay in processing any requests. Once the best manager has been located, it sends the session descriptions to the user.

Thus, given

$B$ = branching factor of IDG (managers per level)
$D$ = depth of manager with "interesting" results
$K$ = number of announcements per manager

Then, the search time for the IDG is given as

$$ST_{idg} = B*D+K$$

To make our calculations, we assumed that each manager will have 50 data sources registered with it, and that there will be 5 managers at each particular level. This is a worst case bound; the amortized cost of searching should actually be less in most cases, since the user will often look in the same locations of the IDG that he or she already has cached from previous searches. (This idea of spatial caching is described further in Section 5.1.) The comparison of search time between the IDG and SAP is shown in Figure 2.

Thus, in the IDG, the search time depends on the depth of the deepest manager, which, in a tree, is logarithmically related to the number of data sources.

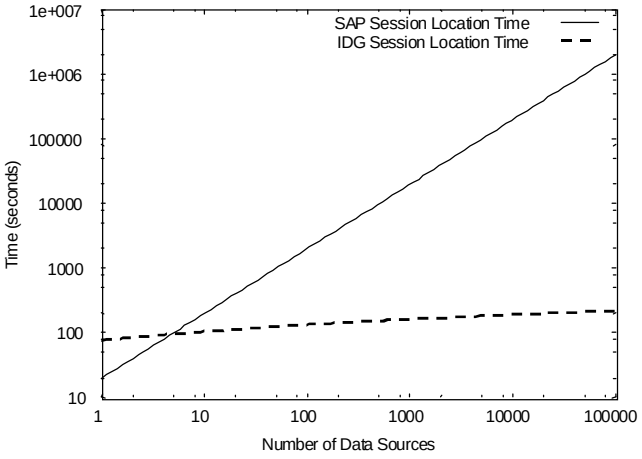


**Figure 2. Search time comparison**

### 4.3. Global multicast bandwidth comparison

SAP uses a constant amount of global multicast bandwidth: 200 bps. However, defining a constant value in a protocol does not lend itself to scaling well. Moreover, we argue that if there is enough bandwidth to

support many large multicast sessions, that some fraction of that bandwidth should be available to a session directory. We could put such a modification into SAP, but this would only be a temporary fix to the problem, as it would still neglect the issue of organization.

To compare the global multicast bandwidth used by SAP to the amount used by the IDG, we assume that each multicast group uses a constant amount of bandwidth, as SAP does for the single multicast group. If each manager contains a fixed number of data sources, the number of multicast groups in the IDG will be linearly related to the number of managers. Since the number of managers is also proportional to the number of data sources, we determine that the global multicast bandwidth grows as a small fraction of the number of data sources. This was confirmed by our simulation results (see Section 4.4 below).

Again, for

$K$ = announcements per manager
$N$ = number of announcements
$M$ = number of managers = $N/K$

Our total global bandwidth used is

$BW = M*(200\text{bps})$

This is on the order of the number of data sources, which is unacceptable for general deployment. We describe future work measures that will reduce the global multicast bandwidth in Section 5.

### 4.4. Simulation results

We implemented our IDG model in Parsec, a scalable simulation language developed at UCLA. We used this simulation to get a preliminary indication of how well our model works. We have not yet refined the simulation to allow simulations involving thousands of managers and data sources. The simulations we have run involved 45, 90, 135, and 180 entities. In these simulations, 22% of the entities were managers, 66% were data sources, and the remaining 12% were routers. Routers served to simulate packet flow in the network.

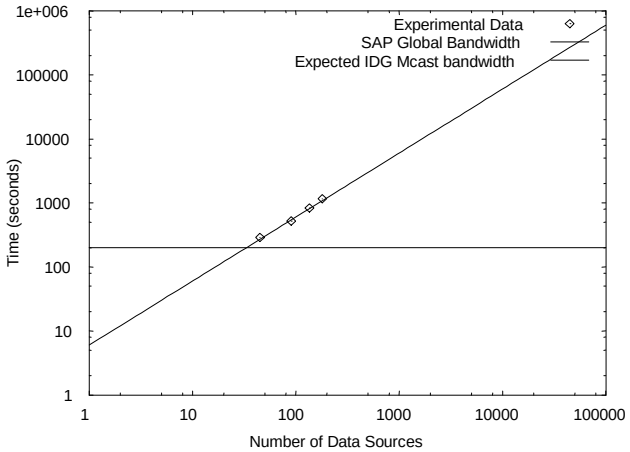


**Figure 3. Multicast bandwidth usage**

To simplify the experiments, when a manager had 2 or more data sources, it would begin to expand the manager hierarchy. Since data sources do not contribute to the global multicast bandwidth, this effectively simulates many more data sources in the IDG. Each simulation was run for 10,000 seconds, and the multicast bandwidth used was recorded. As seen in Figure 3, our simulations so far exactly match the expected bandwidth usage.

## 5. Current status and future work

In addition to the simulation, we have developed a prototype implementation of the IDG in Java, including the infrastructure as well as a simple user front-end (see Figure 4). We have also created a special manager to encapsulate SDR sources, which could be used for incremental deployment. Our implementation provides the core functionality of our design, but it needs to be developed further. In addition, we list below a number of issues deserving of future research.

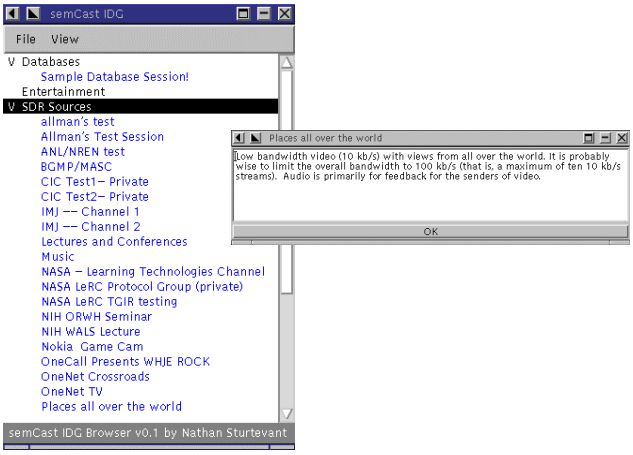


**Figure 4. Java GUI**

### 5.1. Scaling in spatial dimension

The IDG fixes the delays in locating interesting data sources by letting the managers maintain a sorted mesh of caches. These caches are temporal caches; they allow efficient searches based on semantics and save having to wait to hear information directly from content source. However, these caches do not address the spatial locality of a user or data source. It may be that a source is providing content for only a limited region, so the announcement should be registered locally, and not sent

globally. Also, it is undesirable to have every topic group multicast globally, or to have a client make many queries to managers widely distributed across the Internet.

Either TTL or administrative scoped multicast can be useful tools in addressing the issue of scaling in spatial dimension. A good outline of administrative scoping can be found in [5]. This reference also shows how scoping can be used to help support localized multimedia sessions. The same concept should apply to IDG design. A local manager, or set of local managers, could both manage locally scoped data sources and provide a cache of interesting global sessions.

### 5.2. Reduction of multicast bandwidth

Our design has made an exact tradeoff with the SAP architecture. By reducing user search time, the IDG has increased the multicast bandwidth used. Our original efforts were directed towards optimizing the session location experience. While the quality of search has increased, the overhead costs are larger than we like. We plan to see how, by changing our growth assumptions, we can fix this issue. For instance, it may not be the case that managers hold a constant number of data sources, which changes the dynamics of how the overhead grows. There also may not be a fixed number of managers in each level of the tree. These issues depend partly on how the taxonomy is defined to organize the managers. Ultimately, we hope that our global bandwidth usage will be logarithmically related to the number of data sources.

### 5.3. Kohonen nets

In the current design, a new data source is inserted into the IDG wherever the managers determine it is most appropriate. The managers determine this by using heuristics such as keyword matching. This may or may not be adequate for real usage. As society changes with time, new areas of interest can quickly arise. It might be desirable to have a system that could automatically detect and incorporate such changes. One type of neural network, the Kohonen net [7], has been used to classify Usenet newsgroup articles [8]. Under this system, the Kohonen network placed articles together on a feature map based on their relevance to each other. This system worked offline to produce the results. We are experimenting to see if an online system could be developed to enhance the IDG with a dynamically growing and adjusting mesh of keywords for content categorization. Initial experiments suggest that the information contained within a data source description is inadequate for this purpose, although other similar methods may be more successful.

### 5.4. Database search methods

We have done little research on what search methods can or will be used by managers to handle queries. As another aspect of the Semantic Multicast project, researchers at the University of Illinois at Chicago are investigating using relevance feedback to improve database queries [9]. Exactly what, if any, query mechanisms are supported by managers has yet to be decided. If a manager contains only a few data sources, it is reasonable to send the user all data sources and let the user filter them. However, allowing the user to search using characteristics such as session bandwidth requirements and broadcast times would also be useful.

### 5.5. Security

Security is a common issue in any open system, and is yet to be well addressed in our design. There are several areas in which the IDG infrastructure is vulnerable to attack. First, as one can do to a search engine with a web page, a user can falsely insert thousands of session announcements across the IDG. While the open design for making announcements cannot prevent this, we can at least curb this behavior by requiring some authorization before announcements are submitted. The IDG is also open to attack if someone attempts to “hijack” a session announcement. Public key encryption can help establish “ownership” of a session announcement and prevent this problem. Additionally, a data source can detect a malicious manager by querying the manager and checking to see if its own description is returned. Both Web search engines and SAP can suffer similar attacks, and we acknowledge the need for good security in any session directory. Many security issues have been discussed in [10], and the ideas therein are applicable to the IDG.

## 6. Related work

As we stated earlier, the fundamental goal of this work is to explore the design space of a scalable, robust, effective, and efficient information discovery service infrastructure. The most widely used tool for information discovery today is Web search engines. As we pointed out in Section 2, however, all existing Web search engines suffer from two common defects: the inability to derive adequate semantic meaning from the content; and, more fundamentally, the poor scalability of a centralized system. Given the phenomenal growth of the Internet over the last few years, which predicts a corresponding phenomenal growth in online content, we believe that a distributed information infrastructure will be necessary to provide scalable information discovery services.

An earlier research effort to build a scalable Internet information discovery service is Harvest [11]. The design

of Harvest includes a few major components: Gatherer, Broker, Index Subsystem, and Replicator. *Gatherer* runs at content providers' sites to summarize local content for efficient export to the *Broker*. The Broker collects inputs from all sources and uses the *Index Subsystem* to build a space-efficient index database. To make the index widely available, the *Replicator* builds a weakly consistent, wide-area file system to hold the broker index database. In short, Harvest offers information discovery service by replications of a central index database, while our IDG design aims at building a *distributed* search database to scale well with the Internet growth.

Similarly, the Domain Name System (DNS) is the largest distributed database currently deployed on the Internet. DNS service provides an essential component in daily network operation, and the design has scaled adequately with the Internet growth. The DNS system is composed of a very large set of autonomous servers, each in charge of name-to-address mapping services of a different domain. The system achieves robustness by server replication, and it achieves scalability by heavy use of caching, which is based on the assumption that name-to-address mapping changes slowly. One undesirable, but perhaps necessary, drawback of the DNS system is that the entire system is *manually configured*. Network operators decide on the exact structure of DNS name space, configure the servers for each name domain, manually input server addresses as the way to set up the interconnection among servers, and manually maintain the name database.

IDG design has borrowed a few key features from the successful design of DNS, but also with one fundamental difference. The IDG system is made up of a large set of autonomous information managers, each in charge of different topical contents. For efficient operation, the IDG also plans on heavy use of caches of query results that do not change frequently. However, IDG is also fundamentally different from DNS by moving away from manual configuration. Instead of a fixed name space, IDG's hierarchical mesh of semantic information will gradually adjust itself as the information contents are added or deleted. Instead of configuring a server hierarchy, IDG will draw information managers from a pool of autonomous servers made available by network providers and clients and place them at proper points in the semantic structure as the demand warrants. In short, IDG design aims at building a self-configured, continuously re-organizing server infrastructure to achieve the goal of scalability and robustness.

## 7. Conclusions

We have presented the initial design and simulation results of the Information Discovery Graph, one component of the Semantic Multicast project. The IDG is responsible for providing directory services for multimedia sessions, both stored and real-time, and for organizing the sessions according to semantic content. Data sources can register with the IDG, and users can browse or query for sessions by semantic content. The IDG overcomes limitations of current multimedia directories (conventional Web search engines and the MBone's SDR tool) to provide a robust and scalable directory service. Our simulation confirms our model in terms of the global multicast bandwidth overhead incurred by IDG. Having developed the framework and a basic model, we can move forward with our ideas of future research on spatial caching, scoping, applying AI techniques to improve searching heuristics, using relevance feedback to improve querying, and exploring security issues raised by a global deployment of the IDG.

## 8. References


[1] Semantic Multicast project, *http://www.wins.hrl.com/projects/semcast/*.

[2] MBone home page, *http://www.mbone.com*.

[3] M. Handley, "SAP: Session Announcement Protocol", Internet Draft *draft-ietf-mmusic-sap-00.txt*, Nov. 1996.

[4] SDR, *http://north.east.isi.edu/sdr/*.

[5] A. Swan, S. McCanne, and L. Rowe, "Layered Transmission and Caching for the Multicast Session Directory Service", *Proceedings of ACM Multimedia '98*, Sep. 1998.

[6] M. Handley and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, Apr. 1998.

[7] T. Kohonen, "Self-Organizing Maps", *Series in Information Sciences*, vol. 30, 2nd edition, Springer-Verlag, Heidelberg, 1997.

[8] T. Honkela, S. Kaski, K. Lagus, and T. Kohonen, "Newsgroup Exploration with WEBSOM Method and Browsing Interface", *Technical Report A32*, Laboratory of Computer and Information Science, Helsinki University of Technology, Espoo, 1996.

[9] Semantic Multicast project, internal documentation.

[10] P. Kirstein, G. Montasser-Kohsari, and E. Whelan, "SAP Security Using Public Key Algorithms", Internet Draft *draft-ietf-mmusic-sap-sec-04.txt*, Mar. 1998.

[11] C.M. Bowman, P.B. Danzig, D.R. Hardy, U. Manber, and M.F. Schwartz, "The Harvest Information Discovery and Access System", *http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/Searching/schwartz.harvest/schwartz.harvest.html*