Bulletin of the Technical Committee on

Data Engineering

September 1996 Vol. 19 No. 3

IEEE COMPUTER SOCIETY
50 YEARS OF SERVICE • 1946-1996

Letters

The Bulletin of the Technical Committee on Data Engineering is published quarterly and is distributed to all TC members. Its scope includes the design, implementation, modelling, theory and application of database systems and their technology.

Letters, conference information, and news should be sent to the Editor-in-Chief. Papers for each issue are solicited by and should be sent to the Associate Editor responsible for the issue.

Opinions expressed in contributions are those of the authors and do not necessarily reflect the positions of the TC on Data Engineering, the IEEE Computer Society, or the authors' organizations.

Membership in the TC on Data Engineering is open to all current members of the IEEE Computer Society who are interested in database systems.

Letter from the Editor-in-Chief

Bulletin Related Items

Membership Renewal

Included on page 56 of the current issue is a Technical Committee Membership Application Form. This form must be filled in and forwarded to the Computer Society in order for you to continue as a member of the TC on Data Engineering in 1997. It is essential that you file this form in order for you to continue to receive the Data Engineering Bulletin.

Below is the message that I received from the Computer Society.

```
MEMBERSHIP RENEWAL BY DECEMBER 31, 1996 -- To serve our membership the
best way we can with limited resources, we must periodically ensure
that those of you who receive this newsletter and other information
continue to be interested in being part of TCDE.  Every few years, we
will ask that you renew your membership in the technical committee to
remain on the list.

Renew now by completing and returning the TC application form on page
56 by mail, fax, or email.  You only need to submit one form for
all TCs you belong to, as long as you check the appropriate boxes (up
to four).  Members who do not respond by December 31, 1996 will be
removed from the TC list(s).
```

You may also renew by visiting the IEEE Computer Society web site at

http://www.computer.org/tab/tcapplic.htm

and filling in the application there.

Editorial Appointment

I am happy to announce the appointment of Joe Hellerstein, from the University of California, Berkeley, as an Associate Editor. Joe graduated from Wisconsin only last year. His work in query processing and access methods has already established his reputation as an outstanding database researcher. Joe will be the editor of the December issue of the Bulletin.

About this Issue

The current issue focuses on information dissemination. This is a particularly timely issue as use of the world wide web has exploded. And information dissemination is one of the key roles played by the web. The issue is broader than only the web, discussing several of the fundamental technologies for information dissemination. I want to thank Mike Franklin for the fine job he has done in soliciting papers from diverse sources, exploiting the diverse technologies. While no single issue of any publication is likely to be fully comprehensive, Mike has succeeded in assembling an issue that provides excellent coverage of this very rapidly changing field.

David Lomet
Microsoft Corporation

Letter from the Special Issue Editor

In the past few years there has been an explosion in the number and variety of data-intensive applications being deployed. The World Wide Web is just one of the new ways that content (i.e., data) is being delivered to users at the office, at home, and even on the road. To the database specialist, these developments have been both gratifying, as a confirmation of the importance of data management issues, and unsettling, as we watch the information superhighway being paved with nary a transaction nor an equijoin in sight. While database systems will continue to play a significant role in storing and managing the "mission critical" data of most organizations, it is clear that many new applications are quite different from the ones that database systems have traditionally supported. Despite the differences, however, there is much benefit to be gained by applying database expertise to the many new types of data-intensive applications that are beginning to appear.

The focus of this issue is on one particular technology that is at the heart of many new applications, namely, data dissemination. Data dissemination can be loosely defined as the process of delivering data from a source or set of sources to a (typically larger) set of consumers. The articles in this issue cover topics ranging from communication primitives that can enable data dissemination, to architectures that incorporate it, to descriptions of applications that use it.

In the first paper, Azer Bestavaros and Carlos Cunha describe how traditional notions of locality break down on the Web due to the vast amount of information available and the ease of "surfing". They propose a dissemination-oriented approach for placing copies of frequently accessed web pages at key points in the network. Son Dao and Brad Perry describe the SATIN testbed developed at Hughes Research Laboratory to exploit the satellite-based DirecPC system, which is capable of providing multi-megabit delivery of data to millions of users in homes and elsewhere. The third paper, by Stan Zdonik and myself, describes the notion of a dissemination-based information system incorporating multiple modes of data delivery. The challenges that arise in the design of such a system are explored by using our Broadcast Disks method as an example.

David Glance provides a description of how the multicast support found in IONA Technologies' OrbixTalk messaging system can be used to support publish/subscribe communication in a CORBA environment. After describing the basic mechanism, Glance gives several compelling examples of types of applications that can be developed using such technology. Shashi Shekhar, Andrew Fetterer and Duen-Ren Liu, describe the Genesis project to develop an Advanced Traveler Information System (ATIS) in Minnesota. ATIS is a prime example of an application that is outside the traditional notion of database applications, but that can benefit greatly by the insights and techniques that have been developed by the database community. In the final paper, Tak Yan and Hector Garcia-Molina describe their SIFT system for disseminating USENET news articles across the Internet. SIFT, which began as a research project, has recently been commercialized. In this paper they describe the solutions they developed to address the scalability problems that arose with the increasing popularity of the system.

The papers included here provide a sampling of the kinds of new research and development opportunities that have recently arisen due to advances largely outside of the database area. Fortunately, as I believe these papers show, the database community has much to add to these important new applications. I would like to thank Stan Zdonik for his many ideas and discussions, and would also like to thank the Editor-In-Chief, Dave Lomet, for producing the Data Engineering Bulletin, which is an excellent resource for charting the current and future progress of the database field.

Michael Franklin
University of Maryland

Server-initiated Document Dissemination for the WWW

Azer Bestavros (best@cs.bu.edu)
Carlos Cunha (carro@cs.bu.edu)

Computer Science Department
Boston University, MA 02215

Abstract

In this paper we overview the merits of a data dissemination mechanism that allows information to propagate from its producers to servers that are closer to its consumers. This dissemination reduces network traffic and balances load amongst servers by exploiting geographic and temporal locality of reference properties exhibited in client access patterns. The level of dissemination depends on the relative popularity of documents, and on the expected reduction in traffic that results from such dissemination. We present results of log analysis and trace-driven simulations that quantify the performance gains achievable through the use of such a protocol.

1 Introduction

The effectiveness of client-based caching for very large distributed information systems (like the WWW) is limited. This was established in a comprehensive study of client-based caching for the Web, which was conducted by our Oceans group (`http://www.cs.bu.edu/groups/oceans/`). In that study [4], the effectiveness of session caching, machine caching, and proxy caching were established using a unique set of 5,700 client traces (almost 600,000 URL requests), which were obtained by intrumenting Mosaic as detailed in [6]. This study concluded that proxy caching is ultimately limited by the low level of sharing of remote documents amongst clients of the same site. This finding agrees with Glassman's predictions [9] and was further confirmed for general proxy caching by Abrams *et al* [1].

In order to explain the limited effectiveness of WWW client-based caching, we need to consider the locality of reference properties that contribute to an enhanced cache performance [2]. Access patterns in a distributed information system (such as the WWW) exhibit three locality of reference properties: temporal, geographical, and spatial. Temporal locality of reference implies that recently accessed objects are likely to be accessed again in the future. Geographical locality of reference implies that an object accessed by a client is likely to be accessed again in the future by "nearby" clients. This property is similar to the processor locality of reference exhibited in parallel applications [8]. Spatial locality of reference implies that an object "neighboring" a recently accessed object is likely to be accessed in the future.

If client-based caching is done on a *per-session* basis (*i.e.*, the cache is cleared at the start of each client session), then the only locality of reference property that could be exploited is the temporal locality of reference. The

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

	`www.cs.bu.edu`		`www.stones.edu`
	`Data Set 1`	`Data Set 2`	`Data Set 3`
Period	56 days	182 days	110 days
URL requests	172,635	585,739	4,068,432
Bytes transferred	1,447 MB	6,544 MB	112,015 MB
Average daily transfer	26 MB	36 MB	1,018 MB
Files on system	2,018	2,679	N/A
Files accessed (remotely)	974 (656)	1,628 (1,032)	N/A (1,151)
Size of (accessed) file system	50 MB (37 MB)	62 MB (42 MB)	N/A (402 MB)
Unique clients (10+ requests)	8,123	8,474	60,461

Table 1: Summary statistics for log data used in this paper

results of our client-based caching study [4] suggest that for a single client, the temporal locality of reference is quite limited, especially for remote documents. In particular, we found that even with a cache of infinite size, the average byte hit rate is limited to 36% and could be as low as 6% for some client traces. This poor performance could be attributed to the "surfing" behavior of clients, which implies that recently-examined documents are *rarely* revisted in the future.

If client-based caching is done on a *per-site* basis (*i.e.*, a common "proxy" cache is shared among all clients), then one would expect an improved cache performance as a result of the geographical locality of reference property. However, the results of our client-based caching study [4] suggest that for remote documents the amount of sharing between clients is limited. In particular, we found that even with an infinite proxy cache size, the average byte hit rate improves from 36% to 50%. Figure 1, which is reproduced from the data in [4] illustrates this point by plotting the rate at which the proxy cache must be inflated—a measure termed the *Cache Expansion Index* (CEI)—to maintain a constant *byte hit rate*. Figure 1 indicates that the CEI is proportional to the number of clients (N) sharing the proxy cache—it shows that for a large number of clients concurrently using the proxy cache, the CEI is linearly related to N and does not seem to level off.

It must be noted that Figure 1 does not imply that the relationship between CEI and N will continue to be linear for even larger values of N; it only implies that for the levels of concurrency likely to be aggregated at a single site through a proxy cache, the relationship *is* linear. To get a higher level of concurrency, we need to think about an economy of scale far beyond what a proxy cache at (say) an organization can provide.

The above discussion shows that temporal and geographical locality of reference properties for WWW access patterns are not strong enough to result in an effective caching strategy at a single client or site. However, what is yet to be answered is whether temporal and geographical locality of reference properties at a much larger scale (*e.g.*, thousands of clients) could be exploited. In the remainder of this paper, we provide an affirmative answer to this question.

2 The Premise of Dissemination

In order to be able to quantify the available locality of reference properties that could be exploited on the WWW, we collected extensive server traces from the HTTP server of the Computer Science Department at Boston University `http://www.cs.bu.edu` and from the HTTP server of the Rolling Stones web site `http://www.stones.com/`. These traces are summarized in table 1. Unless otherwise stated, our model validation and trace simulations were all driven by this data.

The highly uneven popularity of various Web documents has been documented in [6]. This study confirmed the applicability of Zipf's law [13, discussed in [12]] to Web documents. Zipf's law was originally applied to the relationship between a word's popularity in terms of rank and its frequency of use. It states that if one ranks the popularity of words used in a given text[1] (denoted by ρ) by their frequency of use (denoted by P) then $P \sim 1/\rho$. Note that this distribution is a parameterless hyperbolic distribution. *i.e.,* ρ is raised to exactly -1, so that the n^{th} most popular document is exactly twice as likely to be accessed as the $2n^{\text{th}}$ most popular document. Our data shows that Zipf's law applies quite strongly to *Web documents serviced by Web servers*. This is demonstrated in Figure 2 for all 2,018 documents accessed in the BU trace (data set 1). The figure shows a log-log plot of the total number of references (Y axis) to each document as a function of the document's rank in overall popularity (X axis). The tightness of the fit to a straight line is strong ($R^2 = 0.99$), as is the slope of the line: -0.95 (shown in the figure). Thus the exponent relating popularity to rank for Web documents is very nearly -1, as predicted by Zipf's law. Out of some 2000+ files available through the WWW server, the most popular 256KB block of documents (that is 0.5% of all available documents) accounted for 69% of all requests. Only 10% of all blocks accounted for 91% of all requests!

The above observation leads to the following question: How much bandwidth could be saved if requests for *popular* documents from outside the LAN are handled at an earlier stage (*e.g.,* using a proxy at the "edge" of the organization)? Figure 3 shows the percentage of the server load (measured in total bytes serviced) that would be saved if various block sizes of decreasing popularity are serviced by some other server.

A closer look at the logs of `http://www.cs.bu.edu`, reveals that there are three distinct classes of documents. Figure 4 shows the ratio of remote-to-local (and local-to-remote) accesses for each one of the 974 documents accessed at least once during the analysis period. Out of these 974 documents: 99 documents had a remote-to-local access ratio larger than 85%—we call these *remotely popular documents*; 510 documents had a remote-to-local access ratio smaller than 15%—we call these *locally popular documents*; 365 documents had a remote-to-local access ratio between 85% and 15%— we call the remaining 365 documents *globally popular documents*.

The classification of documents into globally, remotely, and locally popular (*i.e.*, based on temporal/geographical locality of reference) could be easily done by servers in order to decide which documents to disseminate and where such documents should be dessiminated.

An important question that may affect the performance of our dissemination protocol is related to the rate at which popular documents are updated. Notice that the more frequently these documents are updated, the more frequently the home server is to refresh the replicas at the proxies. In a related study, we monitored the *date of last update* of remotely, locally, and globally popular documents for a 6-month (26 weeks) period that started with the 56-day period of table 1, and continued for another 126 days). We observed that both remotely popular and globally popular documents were updated very infrequently (less than 0.5% update probability per document per day), whereas locally popular documents were updated more frequenlty (about 2% update probability per document per day).[2] This result is important because it suggests that those documents most likely to be disseminated are statistically the ones less likely to change. The significance of these findings was further investigated by Gwertzman and Seltzer in the context of WWW cache consistency [11]. It is important to note that our measurements and observations apply to information systems of *archival* nature (*e.g.* institutional Web pages). For information systems of *temporal* nature (*e.g.* stock market tickers, news feeds, *etc.*) it is likely that the most popular documents are those that change most frequently.

3 Dissemination Model and Experiments

In our model, information is disseminated from *home servers* (producers) to *service proxies* (agents) closer to *clients* (consumers). We assume the existence of a many-to-many mapping between home servers and service

[1]Zipf's law has subsequently been applied to other examples of popularity in the social sciences.
[2]Multiple updates to a document within one day were counted as *one* update.

proxies. A service proxy along with the set of home servers it represents form a *cluster*. We model the WWW (Internet) as a hierarchy of such clusters.[3]

Our notion of a service proxy is similar to that of a client proxy, except that the service proxy acts on behalf of a cluster of servers rather than a cluster of clients. In practice, we envision service proxies to be information "outlets" that are available throughout the Internet, and whose bandwidth could be (say) *"rented"*. Alternately, service proxies could be public engines, part of a national computer information infrastructure, similar to the NSF backbone. For the remainder of this paper we use *proxy* to refer to a service proxy.

Our model does not limit the number of proxies that could be used to "front-end" a particular server. Each server in the system may belong to a number of clusters, and thus may have a number of proxies acting on its behalf, thus disseminating its documents along multiple routes (or towards various subnetworks). A server is allowed to use (through bidding for example) a subset of these *proxies* to disseminate its data to clients.

In order to evaluate our dissemination protocol, we had to come up with a realistic clustering of the Internet to reflect the dissemination model introduced above and detailed in [3]. This clustering could be based on several criteria. For example, it could be based on geographical information (*e.g.*, distance in miles between clients as suggested in [10]). Alternately, it could be based on institutional/network boundaries (*e.g.*, assign one service proxy per institution or network). Our choice was to base the clustering on the structure of the routes between a server and its clients. Such a clustering (based on the distance between the server and client measured in *actual route hops*) allows us to quantify the savings achievable through dissemination because it makes it possible to measure the bandwidth saved in *Bytes* $\times$ *Hops* (BHops) units. We define a *BHop* to be equivalent to a single byte transfered over one hop. A reduction in BHops implies either a reduction in the total number of bytes transfered or a reduction in the transfer distance (in hops), or a reduction in both.

Using the `record route` option of TCP/IP, it is possible to build a complete tree originating at the server with clients at the leaves. We built such a tree for all *traceable* clients of `http://www.cs.bu.edu`. By a traceable client, we mean a client for which `traceroute()` succeeded in identifying a route that remained consistent over several traceroute experiments. Almost 90% of all bytes transferred from `http://www.cs.bu.edu` were transferred to traceable clients. For data set 2 of table 1, the constructed tree consisted of 18,000+ nodes.

Figure 5 shows a histogram of the number of clients (leaf nodes) versus the distance in hops of these clients from the `http://www.cs.bu.edu` server. This histogram shows three distinct client populations. The first population is within a distance of 2-3 hops and represents on-campus clients at Boston University. The second population is within a distance of 4-9 hops and represents clients on the New England Academic and Research network (NEARnet). The third is within a distance of more than 9 hops and represent WAN clients. For servers where most of the demand is generated by WAN clients, Figure 5 suggests that popular documents could be disseminated as much as 8-9 hops away from the server. Such a dissemination would result in large savings. Specifically, replicating the most popular 25 MB from `http://www.stones.com` on proxies 8-9 hops closer to clients would yield whoping network bandwidth savings of more than 8 GByte$\times$Hops per day.

We performed simulations of our dissemination strategy based on the structure[4] of the server-to-client routing tree discussed above. Our simulations were driven by the 26-week traces obtained from `http://www.cs.bu.edu` (data set 2 in table 1).

In our simulations, replicas of the most popular files on `http://www.cs.bu.edu` (the root of that tree) were disseminated every week down to proxies (internal nodes of the tree) *closer* to the clients. The location of such proxies depends on the demand during the previous month (4 weeks) from the various parts of the tree and aims to balance the load amongst those proxies.

The general dissemination model presented in [3] assumes that proxies are organized in a hierarchy. Our implementation of this idea was simpler, whereby the hierarchy was flattened by having the home server (the

[3]A detailed treatment of this model, including an optimal resource allocation strategy for proxies to ration their resources amongst competing home servers belonging to the same cluster can be found in [3].

[4]Analysis of `http://www.cs.bu.edu` logs suggests that the shape of the tree (especially internal nodes) and the distribution of load is quite static over time.

root of the tree) compute using the access patterns of all its clients the ultimate placement for the replicas. In other words, our simulations did not consider multi-level proxying (*i.e.*, the possibility of proxying a proxy) as would be possible under the general dissemination model.

In our study we assumed that any internal node is available as a service proxy. In a real system, this assumption may not be valid since internal nodes are routers, unliquely to be available as service proxies. In practice, we envision that the set of proxies available *"for rent"* by a particular server could be matched up to the optimum place obtained using our protocol.

In our simulations, we require clients to always request service from the home server. If the requested documents are available at proxies closer to the client, the home server forwards the request to the proxy that is closest to the client. This forwarding mechanism is supported by the HTTP protocol and has much less overhead than the more general hierarchical name resolution strategy described in [7].

In our simulations, we assume that the cost of propagating updates to proxies could be ignored. To establish the validity of this assumption, consider the ratio of the number of times a *popular* document is changed per unit time to the number of times that document is accessed per unit time. As noted earlier, popular documents are updated least. This means that the ratio of updates versus access will be very small (we measured this ratio at less than 0.001 for popular documents). Thus, the inaccuracy of our simulations (by not taking into account the cost of propagating updates) is very insignificant.

Figure 6 shows the percentage reduction in bandwidth (measured in $bytes \times hops$) that is achievable by disseminating a percentage of the most "popular" data available on the server. The horizontal axis shows the number of proxies to which this data was disseminated. Two curves are shown. The first assumes that the most popular 10% of the data ($\approx$ 5 MB) is to be disseminated, whereas the second assumes that the most popular 4% ($\approx$ 2 MB) of the data is to be disseminated. Obviously, a larger number of proxies results in a deeper dissemination, and thus larger bandwidth savings.

One observation from figure 6 is that most of the bandwidth saved through the use of dissemination is due to the dissemination of a very small amount of data. For example, increasing the level of dissemination by 150% (from 2MB per service proxy to 5MB per service proxy) results in a meager 6% additional bandwidth savings. Another observation from figure 6 is that most of the bandwidth saved results from using a small number of service proxies. For example, tripling the number of service proxies from 20 to 60 results in only 21% additional bandwidth savings. Finally, figure 6 shows that the payoff per replica decreases as the number of replicas increases.

The results illustrated in figure 6 are based on a dissemination strategy that is done only once (on day one), based on analysis of the complete logs for all 26 weeks used to drive the simulations. Such an approach would be reasonable if the popularity profile is static. A more practical (and accurate) approach would be to periodically compute the popularity profile based on past access patterns. Figure 7 shows the performance results achievable using such an approach. In particular, it shows the reduction in bandwidth for four experiments, in which the dissemination was performed once a week using the logs of the last n weeks to compute the popularity profile, for n=1, 2, 3, and 5, respectively. For all of these experiments, the 10% most popular data (computed over the n-week period) was disseminated. Figure 7 shows that the longer the history used to compute the popularity profile, the better the performance, especially when the number of proxies is quite large.

An interesting observation from figure 7 regards the non-monotonicity of the performance gains with respect to the level of dissemination. In particular, while the general trend in figure 7 is for the performance to improve as the number of proxies is increased, there are many instances in which an increase in the number of proxies results in a minor performance degradation, especially when the history used to compute the popularity profile is short (*e.g.*, one week). This anomaly can be explained by noting that deploying a larger number of proxies results in a deeper dissemination and that the further a proxy is from the home server, the more lilkely it is for the load generated at that proxy to fluctuate. In other words, dissemination based on a smaller ratio of clients-to-proxies (which is the case when deep dissemination is attempted) is less effective. Recall that, at the limit when documents are propagated all the way to client proxies (*i.e.* closer to the leaves of the tree), dissemination reduces to client caching which has been shown earlier (in Section 1) to be of limited effectiveness due to the weak sharing amongst clients.

Figure 8 documents this observation by showing the load-variation index (on the Y axis) for the top 17 levels of the dissemination tree (on the X axis). This index is computed by normalizing the *change* in the percentage of load at a proxy from one week to the next using the Z-score technique.

In our simulation, the same data is disseminated to all proxies. Better results (and less succeptibility to variation in geographical locality of reference) are attainable if the dissemination strategy takes more advantage of the geographic locality of reference (by disseminating different data to different proxies based on the access patterns of clients served by each proxy).

4 Summary and Future Research

Current service protocols in large-scale distributed information systems are client-based; they do not use the knowledge amassed at servers concerning client access patterns. In a series of studies, we have demonstrated that such knowledge could be used effectively to perform dissemination. Using extensive trace simulations we have quantified the potential gains of our protocol. We showed that dissemination is most effective in reducing network traffic (by as much as 40-50%) and in balancing the load amongst servers. These gains are *above and beyond* what is achievable through client-based caching.

The basic premise of our work is that servers are in a unique position to decide *which* documents are worth replicating through dissemination, *how many* replicas should be disseminated, and *where* to place these replicas. This, however, does not imply that clients and their proxies do not (or should not) play a role in improving the performance of information retrieval and in alleviating communication bottlenecks. For example, our work is not a substitute for client-based caching; rather, it is a complement. The primary purpose of client-based caching is to reduce the latency of information retrieval, whereas the primary purpose of information dissemination is to reduce traffic and balance the load among servers. Of course, these purposes are not completely independent—a protocol whose primary purpose is to reduce traffic and balance load is likely to improve the latency of information retrieval. Nevertheless, the most reduction in latency is likely to be the result of a protocol whose primary purpose is just to do that (client-based caching in this case).

Another example of how our work complements other client-based protocols is related to the notion of dynamic server selection, introduced by Crovella and Carter in [5]. The primary purpose of a client-based dynamic server selection protocol is to determine "on the spot" which of several servers to use to retreive a document replicated on all of these servers. This selection is based on dynamic measurements that attempt to locate the server with the least congested route. Such a technique could complement dissemination by allowing servers to communicate with clients the location of all (or some) replicas that are deemed "close" to the client, while leaving to the client-based server selection protocol the final decision as to which one of these replicas to retrieve based on the dynamic conditions of the network.

Our experiments demonstrate the potential savings achievable through dissemination. More experiments are needed to generalize these findings by considering traces from a larger set of servers. Also, more experiments are needed to study the effects of dissemination from competing service proxies. Finally, it should be noted that our study has concentrated primarily on Web document retrieval. As the Web evolves in terms of the type of information and the manner in which this information is presented[5] more elaborate dissemination protocols may have to be developed.

Acknowledgments:
I would like to thank all members of the *Oceans* group `http://www.cs.bu.edu/groups/oceans` for the many discussions and feedback on this work. This work has been partially supported by NSF (grant CCR-9308344).

[5]For example, the increasing percentage of Common Gate Interface (CGI) queries, Java applets and scripts.

References

[1] Marc Abrams, Charles R. Standridge, Ghaleb Abdulla, Stephen Williams, and Edward A. Fox. Caching proxies: Limitations and potentials. In *Proceedings of the Fourth Interntional Conference on the WWW*, Boston, MA, December 1995.

[2] Virgilio Almeida, Azer Bestavros, Mark Crovella, and Adriana de Oliveira. Characterizing Reference Locality in the WWW. In *Proceedings of PDIS'96: The IEEE Conference on Parallel and Distributed Information Systems*, Miami Beach, Florida, December 1996.

[3] Azer Bestavros. Demand-based document dissemination to reduce traffic and balance load in distributed information systems. In *Proceedings of SPDP'95: The 7^{th} IEEE Symposium on Parallel and Distributed Processing*, San Anotonio, Texas, October 1995.

[4] Azer Bestavros, Robert Carter, Mark Crovella, Carlos Cunha, Abdelsalam Heddaya, and Sulaiman Mirdad. Application level document caching in the internet. In *IEEE SDNE'96: The Second International Workshop on Services in Distributed and Networked Environments*, Whistler, British Columbia, June 1995.

[5] Mark Crovella and Robert Carter. Dynamic server selection in the internet. In *Proceedings of the Third IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems (HPCS'95)*, August 1995.

[6] Carlos Cunha, Azer Bestavros, and Mark Crovella. Characteristics of www client-based traces. Technical Report TR-95-010, Boston University, CS Dept, Boston, MA 02215, April 1995.

[7] Peter Danzig, Richard Hall, and Michael Schwartz. A case for cashing file objects inside internetworks. Technical Report CU-CS-642-93, University of Colorado at Boulder, Boulder, Colorado 80309-430, March 1993.

[8] S.J. Eggers and R.H. Katz. A characterisation of sharing in parallel programs and its application to coherence protocol evaluation. In *Proceedings of the 15th International Symposium on Computer Architecture*, pages 373–383, May 1988.

[9] Steven Glassman. A caching relay for the world wide web. In *Proceedings of the First Interntional Conference on the WWW*, 1994.

[10] James Gwertzman and Margo Seltzer. The case for geographical push caching. In *Proceedings of HotOS'95: The Fifth IEEE Workshop on Hot Topics in Operating Systems*, Washington, May 1995.

[11] James Gwertzman and Margo Seltzer. World wide web cache consistency. In *Proceedings of the 1996 USENIX Technical Conference*, San Diego, CA, January 1996.

[12] Benoit B. Mandelbrot. *The Fractal Geometry of Nature*. W. H. Freedman and Co., New York, 1983.

[13] G. K. Zipf. *Human Behavior and the Principle of Least-Effort*. Addison-Wesley, Cambridge, MA, 1949.

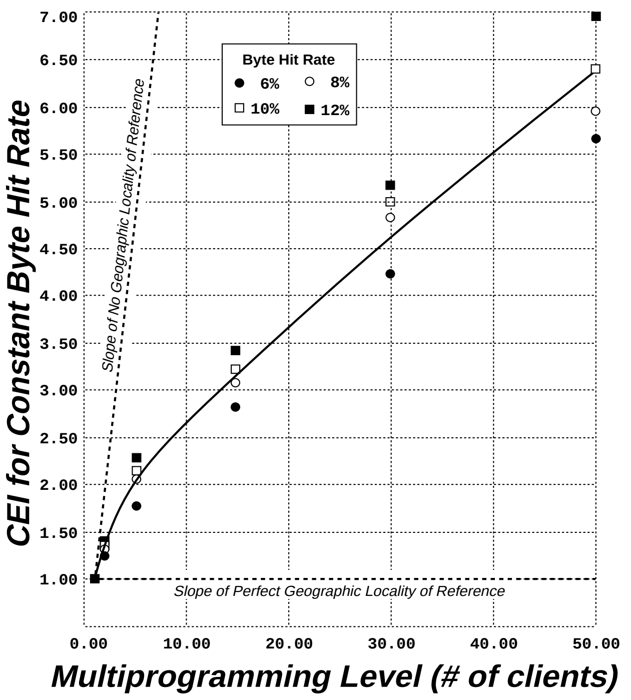

Figure 1: Cache size *vs* MPL for a constant hit rate

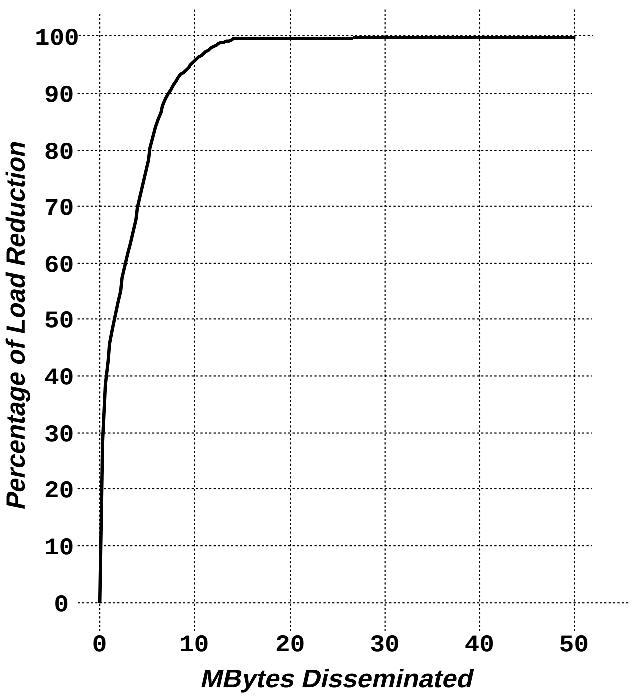

Figure 3: Load reduction due to popularity profile

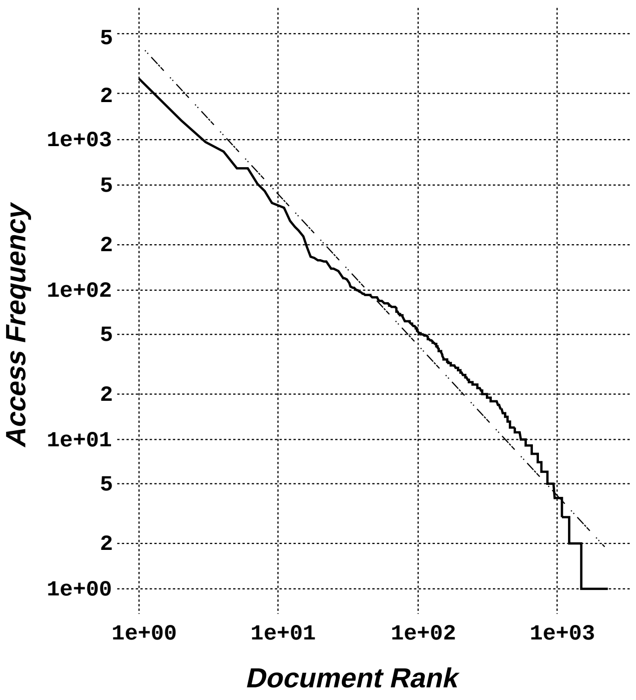

Figure 2: Popularity of documents versus rank

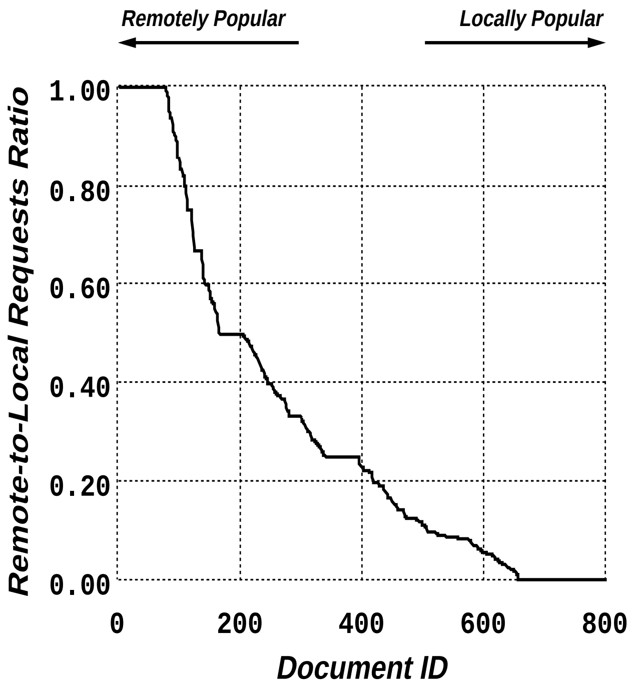

Figure 4: Local vs remote popularity of documents

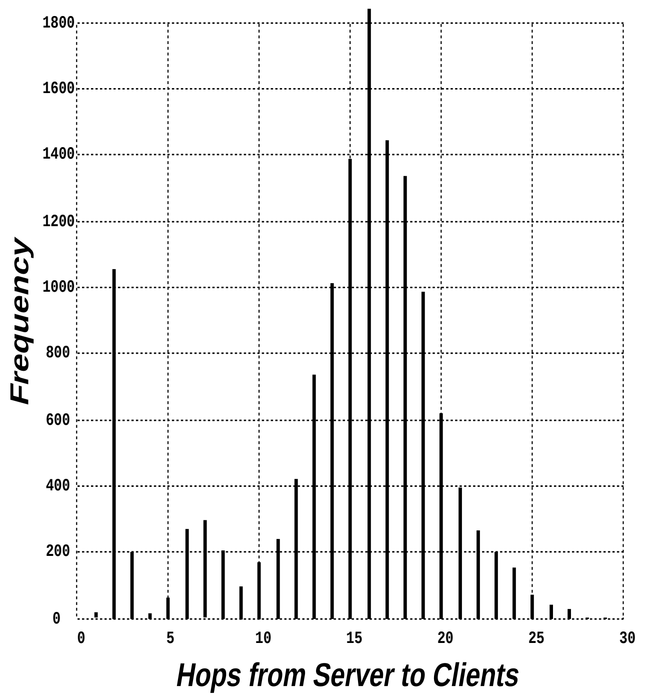

Figure 5: How far away are clients?

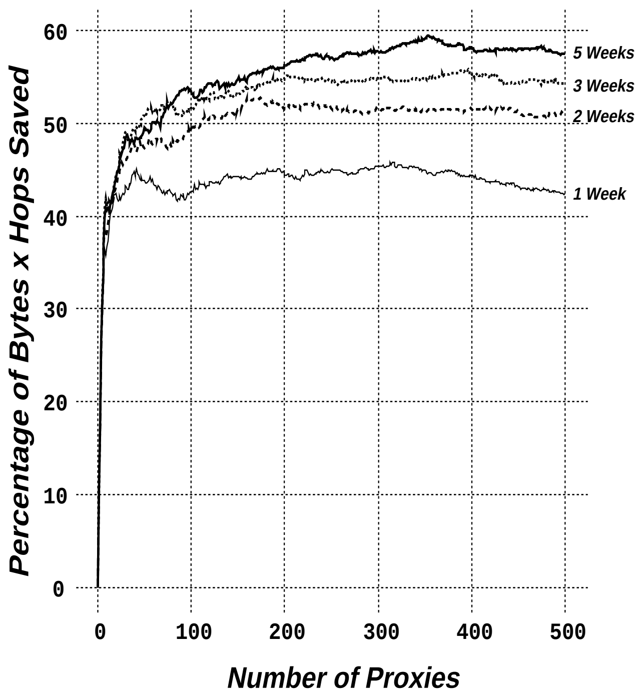

Figure 7: Performance succeptibility to history length

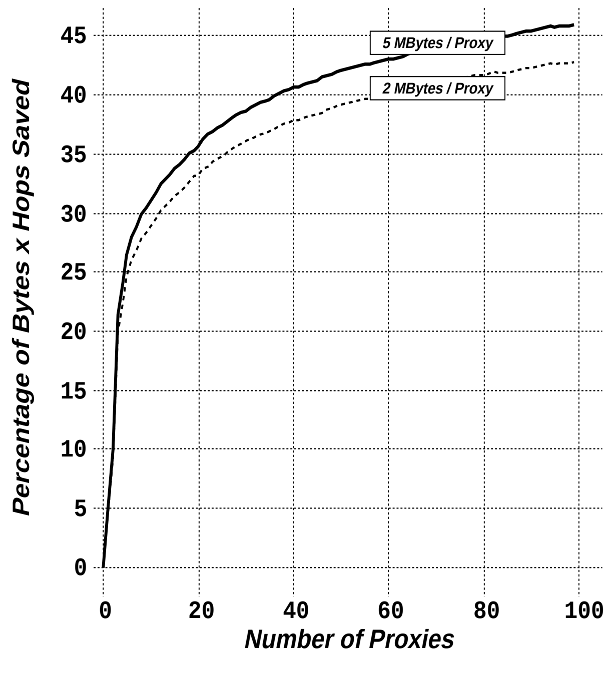

Figure 6: Bandwidth saved through dissemination

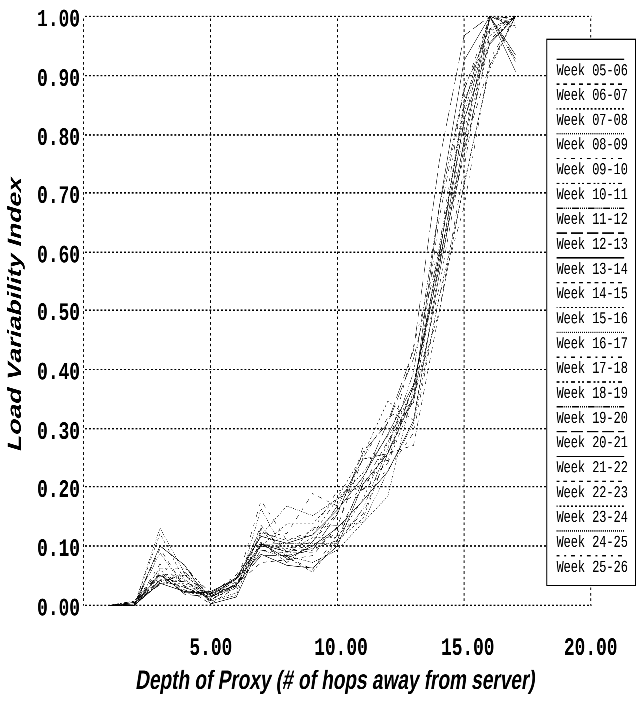

Figure 8: Load variability versus depth

Information Dissemination in Hybrid Satellite/Terrestrial Networks

Son Dao   Brad Perry
Information Sciences Laboratory
Hughes Research Laboratories
Malibu, CA 90265
{son,perry}@isl.hrl.hac.com

Abstract

In this paper we describe our Satellite and Terrestrial Information Networks (SATIN) testbed and its role in identifying and exploring issues in large-scale information dissemination. In particular, we explore the benefits of using satellite broadcast channels to enable information delivery in dynamic, mobile, and multimedia domains. We describe methods to evolve satellite uplink centers into information gathering centers that watch networked information sources and proactively construct packets of "similar" information for efficient broadcast dissemination. We describe two innovative dissemination principles that arise in SATIN applications: (1) smart push control *at the information gathering and broadcast dissemination centers; and (2)* seamless pull control *at the filtering and redistribution nodes. These two principles combine to demonstrate the effective use of broadcast channels for proactive data dissemination from large-scale, dynamic, and networked information sources.*

1 Introduction

The amount and diversity of information available across networks is exploding. Information sources change constantly, and users are faced with the daunting challenges of collecting, evaluating, navigating, and processing in this dynamic information universe. As the scale and rate of change for online information continues to grow, the dominant "mode" of information access must migrate from user-initiated, comprehensive searching to source-initiated dissemination of relevant information. As the importance and coverage of online information continues to grow, the dominant "mode" of information access must migrate from pre-established communication endpoints to anytime access anywhere on the globe. At Hughes Research Laboratories we are researching and developing novel methods to beneficially unite advancements in active data dissemination with advancements in global network connectivity.

One particularly interesting advancement in wireless networking is the availability of high-bandwidth satellite communications as standard network components. Products such as DirecPC [5] allow a common desktop PC to connect to a small (24-inch) personal satellite receiving dish. This small dish immediately charges the PC with a 1 Mbit/s mobile and wireless data downlink. DirecPC represents just the surface of innovations that are rapidly bringing satellite network connectivity to an economic and individualized (or local group) level. With this fact in mind, we have developed the Satellite And Terrestrial Information Network (SATIN) testbed [4]. The SATIN testbed lets us explore the benefits and ramifications of interleaving satellite broadcast channels, in wired and wireless networks, for information delivery in dynamic, mobile, and multimedia domains.

In the remainder of this paper we give an overview of the SATIN testbed and highlight the technical issues it brings to active data dissemination. We then introduce the concepts of *smart push* and *seamless pull* control as the critical information processing components in a satellite-based dissemination network. Finally, we conclude with a summary of information dissemination in the SATIN testbed.

2 The SATIN Testbed

SATIN is a next-generation networking testbed under research and evaluation at Hughes Research Laboratories. The testbed mission is to continually gather state-of-the-art mobile and wireless networking components to research and demonstrate interoperative information access in such a hybrid environment. The following components currently highlight the SATIN testbed:

- Unix servers functioning as the information dissemination server that intelligently gathers and packages data to be broadcast over the Hughes DirecPC satellite.

- Mobile basestations implemented as a van equipped with a DirecPC receive dish and Windows-PC server. Furthermore, the PC has a cellular modem connection to the Internet to serve as a backlink to the dissemination servers.

- Fixed basestations implemented as stationary Windows-PC machine with a DirecPC receive dish and either a direct T1 or 28.8 modem backlink to the Internet and dissemination servers.

- Mobile clients (wireless laptops running either Linux or Windows) communicating with mobile or fixed basestations via a wireless LAN.

Our current prototype applications demonstrate integrated satellite and terrestrial dissemination of map, video, and structured data. The "modes" of information distribution currently supported include:

1. Subscription-based: Dissemination servers actively and automatically broadcast gathered information based on subscriptions of user interests.

2. Query-based: Individual clients or basestations submit ad-hoc requests to one or more dissemination servers. Results are broadcast to basestations for redirection to the requesting client *as well as* all potentially interested clients (based on profile and subscription information).

3. Propagation-based: Clients submit locally computed information to dissemination servers, via the basestations, for selective broadcast to all interested parties.

The SATIN testbed is designed to research innovative techniques for *rapidly deployable mobile networks in high-demand information applications.* Figure 1 depicts the general model of information dissemination studied in our SATIN testbed. This figure captures a model of actively pulling information from networked sources, intelligently packaging relevant information, autonomously pushing the information to wide areas of basestations, and intelligently redistributing information throughout wireless local area networks. One of the strengths of our research group is the multi-disciplined approach to these topics. In particular, we are studying how next-generation networks affect and enable effective dissemination and how large-scale dissemination affects and enables innovations in mobile and wireless networks. The first issue, that of network support enabling information dissemination, is the focus of the remainder of this paper.

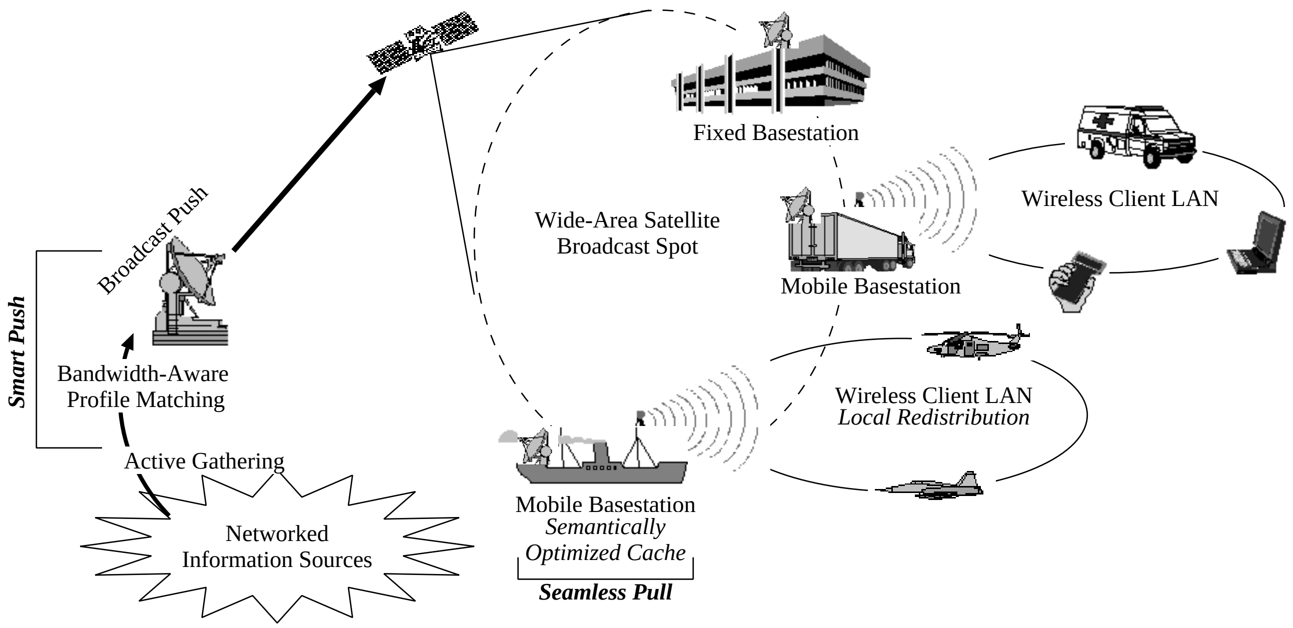

Figure 1: SATIN Architecture

3 SATIN – Technical Issues and Goals

The SATIN testbed enables rapidly deployable, mobile, and large-scale network connectivity. The innovative component is the use of satellite "channels" to reach mobile units, bridge disconnected subnets, and provide high bandwidth network broadcast. A closer look at the SATIN architecture identifies the role of satellite distribution in our current applications. The satellite broadcast channel is used to disseminate large amounts of information to a multitude of mobile, wireless, and/or fixed basestations. The basestations, in turn, redistribute selected information to mobile clients in the local area. Therefore, each satellite uplink center is the driving force pushing information from the sources, through the basestations, and to the mobile clients. There are two primary benefits we gain from the satellite distribution channel: (1) wireless access to clients and basestations anywhere in the satellite's "coverage"; (2) high-bandwidth broadcast access to all units (wireless or fixed) everywhere in the satellite's "coverage". The broadcast nature of satellite delivery is a characteristic to be "optimized" during information processing decisions. Indeed, it is not economical or computationally effective to use satellite broadcast to deliver information to individual receivers, one transmission at a time. Instead, we are faced with the following rule-of-thumb: *a satellite delivery channel is* `best used` *when the number of receiving dishes is* `maximized` *for each information package to be transmitted.* This maximization rule, though, must work within the finite limits of the satellite system – both the available bandwidth on the broadcast channel and the upper bound on the number of simultaneous receive dishes for any given transmission. With these issues in hand, we identify the following characteristics to be considered:

- If you are going to use a satellite delivery channel to distribute information to a (mobile or fixed) unit, then every other unit within the satellite's coverage should be prepared to accept broadcasts of pertinent, though unsolicited, information.
- When creating an information package to be transmitted on a satellite broadcast channel, the package should be "massaged" so as to be of interest to the greatest number of units in the current satellite coverage area.

We have been exploring applications where mobile clients submit *profiles* of their information needs to be monitored and collected in the information infrastructure. The basestations collect profiles from clients in the local area and submit profile sets to the dissemination servers. The dissemination servers collect profiles from the basestations and actively monitor and gather information from networked information sources. As "appropriate" information is found, it is aggregated into packages of similar data to be broadcast to the basestations. The basestations receive "relevant" packages and filter and decompose the information for redistribution to the mobile clients and/or caching for anticipated future client needs.

This active dissemination model requires information processing components for *smart push control* at the dissemination servers and *seamless pull control* at the basestations and mobile clients. In general, *smart push control* combines the satellite delivery bounds, the current basestation profiles, and the continually gathered information to construct the most appropriate information packages to be broadcast throughout the coverage area. *Seamless pull control* identifies the ability for basestations to dynamically accept unsolicited broadcast information and decompose it into objects ranked according to the perceived relevance of its local area of client needs. We are actively exploring methods for these interrelated control mechanisms, as elaborated below.

4 Smart Push and Seamless Pull Control

Figure 2 depicts the general modules of information processing in smart push and seamless pull control. Figure 1 is annotated with the "location" of each control entity in the overall network. As shown in the figure, smart push control bridges the gap between networks of information sources and satellite uplink managers. Seamless pull control, on the other hand, is located within the broadcast receiving basestations. Each basestation oversees a local area network of clients (a wired or wireless LAN). The primary function of the basestation is to collect the broadcast information most *relevant* to its local clients' needs. The basestation then redistributes high priority data to the clients and caches relevant data for anticipated future client requests. It is possible to duplicate the seamless pull control functionality in the actual clients and bypass the basestation completely. We do not promote this solution because clients are frequently disconnected and operating with limited resources.

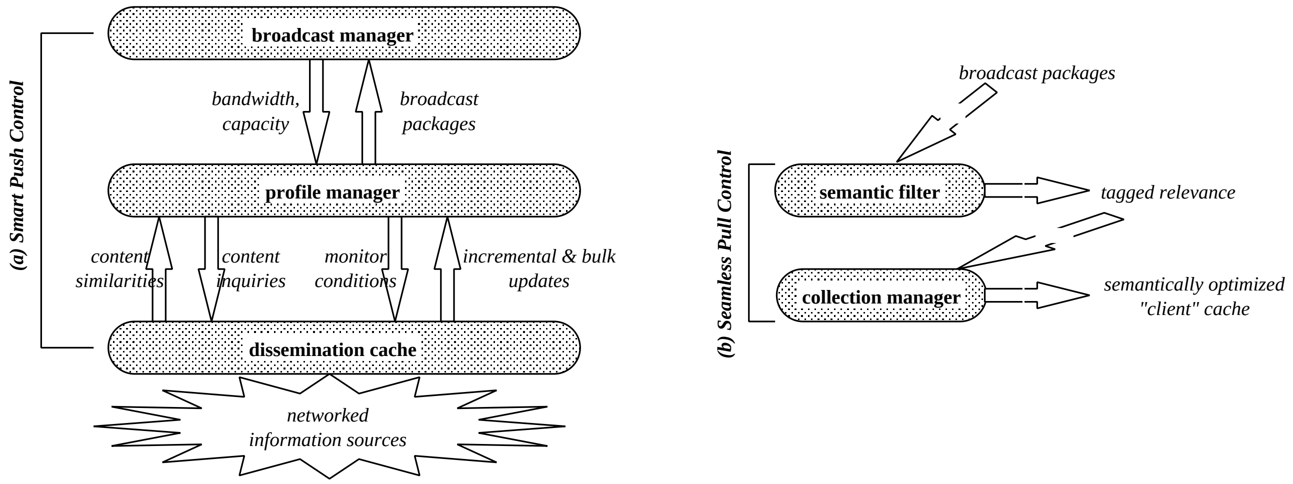

Figure 2: Smart Push and Seamless Pull

The goal of smart push control is to efficiently anticipate and disseminate the most appropriate data required to a multitude of dynamically and incrementally evolving mobile clients. We achieve this goal by developing innovative techniques for profile aggregation, neighborhood dissemination packets, and bandwidth-aware profile matching. A client profile is a query constructed from concepts and schema elements available in the dissemination servers. Two (or more) profiles are candidates for aggregation if (1) there exists a single query (the aggregate profile) that covers both candidate profiles; and (2) it is anticipated that the result set of the aggregate is semantically relevant to the information requested by each candidate. We recursively apply this "aggregation of profiles" so as to create a lattice of aggregates of varying client coverage, data coverage, and anticipated semantic relevance. By broadcasting the data for an aggregate profile to all associated clients, we are *proactively anticipating* and efficiently disseminating data required during the *incrementally evolving operation* of the mobile clients and networked information sources.

The goal of seamless pull control is to efficiently maintain a local data store, at each basestation, that is highly relevant to the local area of mobile clients. We achieve this goal with semantic filtering and adaptive garbage collection. Semantic filtering is responsible for "grabbing" packets off the broadcast network and annotating them with tags for local client interests. Adaptive garbage collection is responsible for discarding irrelevant data and intelligently clearing less relevant data when higher priority information needs cache space. These two techniques combine to assure that each basestation is actively caching and redistributing the information that is most relevant to the profiles of the local area of mobile clients.

4.1 Smart Push Control

For the case of smart push control (Figure 2(a)), the architecture centers around the interactions between the broadcast manager, profile manager, and dissemination cache. The broadcast manager is responsible for reacting to the load and priorities of the satellite uplink manager. The dissemination cache is responsible for collecting networked information required, or likely to be required, for dissemination to various clients in the field. The profile manager stores the set of all profiles currently under active management by the dissemination server.

A profile aggregation lattice is used to optimize two, often competing, criteria: (1) find the most general profile aggregate that covers a maximal amount of clients with a high degree of relevance; and (2) find the largest profile aggregate that does not exceed the available satellite bandwidth or restrictions on simultaneously receiving units. A profile is a database query, written in an object-based SQL supporting the concepts, schema, and media elements available in the dissemination cache. An aggregate profile generalizes two (or more) profiles by dropping conditions, generalizing condition values, and/or generalizing condition predicates. We build from the field of cooperative query answering [3, 2] to bootstrap the profile aggregation problem. Cooperative query answering has developed techniques to estimate the "relaxation error" [1] when rewriting the condition values in queries with `generalized` or `near` values. The relaxation error is the expected relevance of the results of the relaxed query to the intention of the original query. Similarly, an aggregate profile is the query that relaxes two subordinate profiles with a minimal amount of expected relaxation error. We have expanded this general principle into a methodology for constructing and maintaining weighted profile aggregation lattices for feature-based multimedia data collected from networked information sources.

We can perform a single broadcast with the result of an aggregate profile to simultaneously disseminate the information needs and incremental neighborhoods of all aggregated profiles. In general, given a set of updates to the dissemination cache and status of the broadcast manager, we can compute the best aggregate profile for constructing broadcast data packets. This "best profile" will be the aggregate that (1) covers the maximum the number of subordinate profiles (2) adheres to the availability and capabilities of the broadcast channels; and (3) maintains the highest degree of relevance to the subordinate profiles.

In Figure 2(a) we depict two types of updates to the dissemination cache. An incremental update corresponds to the periodic evolution of a monitored information source – updates are propagated into the cache for evaluation and dissemination. A bulk update corresponds to the "discovery" of a new information source. The identification of a new source has the effect of simultaneously inserting the entire source into the cache for immediate evaluation and dissemination. The problem of bulk updates, or *active source discovery and dissemination*, has not received much attention in previous work in data dissemination. In fact, it was not until the proliferation of networked information sources, such as the World Wide Web, that it was identified as a critical research topic. This "focused subsetting and dissemination of bulk data" remains an open topic to be adequately addressed in this field.

For incremental updates we can build from similar topics developed in text dissemination tools such as SIFT

[6]. In SIFT, the database is taken to be the set of profiles (or subscriptions for information) and each article entering the system is considered a query against the profile database. This approach is extremely effective for environments where the number of profiles is large and the arrival of dissemination data (i.e., updates) is incremental, periodic, and relatively small (in comparison to the size of the profile database). Our profile aggregation lattice mirrors the structure of the SIFT subscription index. Therefore, a SIFT-style breadth first search through the profile lattice is adopted to identify aggregates that cover sets of similar updates.

Effective data dissemination must optimize both profile matching and the capabilities of the broadcast channel(s). For this reason, we have introduced an interface between the broadcast manager and the profile manager. The purpose of this interface is to continually feed the profile matching processes accurate snapshots of the capabilities of the network. We are currently using bandwidth availability and receiver-count capabilities as the broadcast snapshots. Bandwidth availability influences the size of the broadcast packets constructed by the profile matching process. The receiver-count capabilities is a constraint on the number of units that can receive a single broadcast (due to signal constraints in the broadcast channels). These network snapshots are integrated into the semantic profile matching process to cut-off (or continue) matching when purely semantic criteria may warrant further matching (or no more matching).

4.2 Seamless Pull Control

For the case of seamless pull control (Figure 2(b)), the architecture centers around the semantic filter and collection manager at each basestation. The approach builds from the techniques for smart push control and exploits the semantic measures of nearness computed therein.

The semantic filter is responsible for grabbing broadcast packets off the network and annotating them with tags of local client interests. The semantic filter annotates each information packet with that data "exactly matching a client profile" and that data "in the neighborhood of a client profile." A broadcast packet contains two pieces of information: (1) the result set for the profile used as a basis for the broadcast; and (2) the subtree from the profile aggregation lattice rooted at the base profile. The subtree contains enough detail for each basestation to independently tag data sets with degrees of local relevance.

The collection manager is responsible for keeping a maximal amount of relevant information in a basestation's local store. We store the "neighborhood relevance" measures from the semantic filter with each broadcast packet. Given these relevance measures, the collection manager uses a combination of the standard LRU (Least Recently Used) garbage collection mechanism coupled with a bias towards maximal relevance to maintain a highly relevant and optimized basestation cache.

5 Summary

In this paper we have given an overview of the benefits and ramifications of satellite broadcast channels in large-scale information dissemination networks. We argued that a satellite broadcast channel is best used, and is often the best solution, in applications requiring (1) continuous and active dissemination of large amounts of information to large numbers of clients; and/or (2) rapid deployment and connectivity to wireless or mobile entities. We called out the issues of *smart push control* and *seamless pull control* as the principal information processing components enabling effective use of the satellite channels. Both control mechanisms require incorporating an understanding of semantic relevance between and among autonomous information sources and independent client subscription profiles. We are currently experimenting with the solutions for smart push and seamless pull control outlined herein. These experiments include measuring the performance and semantic relevance gained when using satellite-enabled smart push/seamless pull control for disseminating dynamic multimedia data.

References

[1] W.W. Chu, K. Chiang, C. Hsu, and H. Yau. Error-based conceptual clustering method for providing approximate query answers. *Communications of the ACM*, to appear, 1996.

[2] W.W. Chu, H. Yang, K. Chiang, and M. Minock. CoBase: a scalable and extensible cooperative information system. *Journal of Intelligent Information Systems*, 6, 1996.

[3] F. Cuppens and R. Demolombe. Extending answers to neighbour entities in a cooperative answering context. *Decision Support Systems*, 7, 1991.

[4] Son Dao. The Networking and Information Exploitation Department homepage. http://www.wins.hrl.com, 1996.

[5] Hughes Network Systems. DirecPC homepage. Technical Report http://www.direcpc.com/, August 1996.

[6] T.W. Yan and H. Garcia-Molina. Sift – a tool for wide-area information dissemination. In *Proceedings of the USENIX Technical Conference*, 1995.

Dissemination-Based Information Systems[*]

Michael Franklin
UMIACS and Dept. of Computer Science
University of Maryland
College Park, MD 20742 USA
franklin@cs.umd.edu

Stan Zdonik
Department of Computer Science
Brown University
Providence, RI 02912 USA
sbz@cs.brown.edu

Abstract

This paper examines the construction of distributed information systems that incorporate the ability to push data out to clients (i.e., dissemination) in addition to having clients pull data from servers. One key issue that distinguishes such dissemination-based information systems (DBIS) from more traditional ones is communications asymmetry. Asymmetry arises in many new applications due to both physical characteristics such as network bandwidths, as well as to workload characteristics. We outline a number of different data delivery mechanisms, and then focus on one particular dissemination mechanism that we have developed, called Broadcast Disks. Finally, we discuss issues that arise in the design of a DBIS architecture that can provide many different modes of data delivery.

1 Introduction

Ongoing advances in communications including the proliferation of the Internet and intranets, the development of mobile and wireless networks, and the impending availability of high bandwidth links to the home, have fueled the development of a wide range of new information-centered applications. Many of these new applications involve *data dissemination*, that is, the delivery of data from a set of *producers* to a (typically) larger set of *consumers*. Examples of dissemination-based applications include information feeds (e.g., stock and sports tickers or news wires), traffic information systems, electronic newsletters, and entertainment delivery.

Although dissemination-based applications share some common characteristics with other data-intensive applications such as query processing or decision support, the specific style of information flow that arises due to dissemination imposes a special set of demands on an information system architecture. One key aspect of dissemination-based applications is their inherent communications *asymmetry*. That is, the communication capacity or data volume in the downstream direction, (i.e., from servers-to-clients) is much greater than that in the upstream direction (i.e., from clients-to-servers). Dissemination can be both the cause of and solution to asymmetry in an information system. A stock ticker application is an example of the first case. It is asymmetric because the information flow is unidirectional: from the single (or small number of) source(s) where the ticker originates,

[*]This work has been partially supported by NSF grant IRI-9501353, ONR grant number N00014-91-J-4085 under ARPA order number 8220, and by a gift from Intel Corporation.

to the much larger set of clients that monitor the stock ticker. An example of the second case is a wireless network in which fixed base stations can transmit at a bandwidth that is much higher than that at which the mobile clients can transmit. In this latter example, data dissemination is a natural way to exploit the base stations' advantage in bandwidth.

In our work on the Broadcast Disk paradigm for data delivery [Zdon94, Acha95b], we have identified a number of ways that the asymmetric nature of data dissemination can impact the design of an information system. The issues that we have studied include broadcast scheduling and client cache management [Acha95a], client prefetching [Acha96a], and the dissemination of updates [Acha96b]. Our current work is focused on integrating the notion of Broadcast Disks and other forms of data dissemination into a coherent framework, which we refer to as a *Dissemination-Based Information System* (DBIS) architecture.

In what follows, we first examine the causes of communications asymmetry and their relationship to data dissemination. We next outline a number of different data delivery mechanisms, and then focus on Broadcast Disks as one example of such a mechanism. Finally, we discuss our notion of a DBIS and sketch some of the issues that arise in the design of such a system.

2 Communications Asymmetry

As stated previously, a key aspect of dissemination-based applications is that they are asymmetric in the amount of data transmitted from servers to clients versus from clients to servers. It is interesting to note, that while much existing communications infrastructure is symmetric (e.g., the Internet), most information intensive applications that use that infrastructure are asymmetric. Symmetric applications, of course, do exist. Examples include teleconferencing, email, and multi-user games. As bandwidth and connectivity continue to improve, however, networks are increasingly being used to deliver sophisticated content to communities of users. Content delivery is an asymmetric process regardless of whether it is performed over a symmetric channel such as the Internet, or over an asymmetric one, such as a cable television (CATV) network. Thus, regardless of whether or not one expects future networks to be symmetric or asymmetric, it is clear that many important applications of future networks will exhibit asymmetry. Techniques and system architectures that can efficiently support asymmetric applications will therefore be a requirement for future information systems.

Asymmetry can arise from a number of factors, both environmental and application-based. Each of these different types of asymmetry imposes a unique set of demands on a system architecture. Below, we identify several causes of asymmetry. In the subsequent section we examine the available options for data delivery and discuss how each of those options addresses the problems that can arise due to the different types of asymmetry.

2.1 Network Asymmetry

Perhaps the most obvious form of asymmetry occurs when the network bandwidth differs in the downstream (server to client) and upstream (client to server) directions. An example of an asymmetric network is the mobile case mentioned above — mobile computers are supplied with data at a high rate while mobile clients are unable to transmit or can do so only over a very slow cellular connection. Another important example can be found in technologies that are being developed to provide multi-megabit per second data delivery to the home. Three industry segments are vying to be the central provider of such services: the telephone companies (through ADSL and other technologies), the cable television companies (through the use of high-bandwidth cable modems), and direct broadcast satellite providers, such as Hughes' DirecPC. All three of these potential providers are planning, at least for the near term, to provide asymmetric bandwidth. For example, cable systems will allow home computers (or televisions) to be supplied with video and other data at speeds approaching five to ten megabits per second while the home machines will be able to transmit back to the server over channels of only tens of kilobits per second. Furthermore, in some situations, these slow "backchannels" must be shared among several households.

2.2 Client to Server Ratio

A system with a small number of servers and a very large number of clients can be viewed as another example of an asymmetric system because the server message and data processing capacity must be divided among all of the clients. For example, it is quite common for popular World-Wide-Web (WWW) servers to become "swamped" due to an unexpectedly large volume of requests. Such behavior is particularly likely when new features are announced, such as the availability of a new version of a popular software package. When a server becomes swamped, it will in the best case, simply refuse to accept additional connections. In the worst (but currently not unlikely) case, such a server will crash.

2.3 Data Volume

A third way that asymmetry arises is due to the volume of data that is transmitted in each direction. Information retrieval applications typically involve a small request message (e.g., containing a few query terms) and a much larger result. For such applications, the downstream bandwidth requirements for a given client are much higher than the upstream bandwidth requirements. For example, consider an application such as Intel's Intercast, in which URLs of related web pages are downloaded along with a television broadcast. If a viewer wishes to follow up on the related information, they can click on the appropriate URL, which could result in the downloading of significant amounts of data such as maps, related video clips, etc. In this case, the data sent upstream is quite small (i.e., it is basically the "key and mouse clicks") while the data sent downstream can be multiple megabytes or more.

2.4 Updates and Information Creation

A fourth type of asymmetry arises when the data to be sent to clients is frequently updated and/or new data of potential interest to clients is frequently created. In these situations, even though the backchannel might have a reasonably high bandwidth, the extremely high load that would be created by having clients poll to extract the latest updates creates an effective asymmetry. That is, it is not efficient (or in some cases, even feasible) to allow clients to continuously poll for new information as such polling places a high load on both the network and the server(s) as well as on the clients themselves.

2.5 Implications of Asymmetry

These different types of asymmetry all impose somewhat different demands on an information system. Network asymmetry places severe limitations on the use of the client-to-server backchannel, either in terms of the size of messages that can be sent, the number of messages that can be sent, or both. A high client/server ratio also imposes restrictions on backchannel usage, and in addition, makes it expensive for the server to send individual reply messages to each client. Data volume asymmetry implies that a significant amount of upstream bandwidth will go unused in a symmetric network. Finally, in the case of updates and new information, clients could spend significant portions of their own resources (in addition to those of the backchannel and server(s)) if they are required to poll continuously for new data. The frequency of such polling also has the potential to impact the integrity (i.e., currency) of data that is cached at clients.

3 Options for Data Delivery

A dissemination-based information system must incorporate a number of different styles of data delivery in order to efficiently cope with the various types of asymmetry described in the previous section, as well as to support symmetric applications. In addition, it is important to provide different types of data delivery so that the system can be optimized for various criteria. For example, even in an asymmetric network where it is infeasible to allow all clients to use the client-to-server backchannel, it may be desirable to provide upstream bandwidth to those users who are willing to pay a premium for it.

In this section, we identify several characteristics that can be used to compare data delivery mechanisms. Figure 1 shows the three main characteristics we consider: (1) push vs. pull; (2) periodic vs. aperiodic; and (3) point-to-point vs. one-to-N. The leaves of the tree in the figure show how several common mechanisms relate to these characteristics.

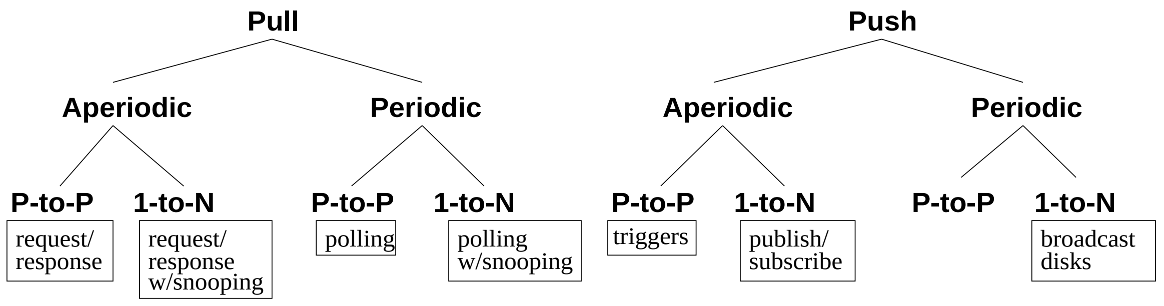

Figure 1: Data Delivery Options

3.1 Client Pull vs. Server Push

The first distinction we make among data delivery styles is that of “pull vs. push”. Current database servers and object repositories manage data for clients that explicitly request data when they require it. When a request is received at a server, the server locates the information of interest and returns it to the client. This *request-response* style of operation is *pull-based* — the transfer of information from servers to clients is initiated by a client pull. In contrast, data dissemination involves sending information to a client population, possibly in advance of any specific request. This is referred to as a *push-based* approach since information transfer is initiated by a server push.

The tradeoffs between push and pull revolve around the costs of initiating the transfer of data. A pull-based approach requires the use of a backchannel for each request. Furthermore, the server must be interrupted continuously to deal with such requests and has limited flexibility in scheduling the order of data delivery. Also, the information that clients can obtain from a server is limited to that which the clients know to ask for. Thus, new data items or updates to existing data items may go unnoticed at clients unless they periodically poll the server.

Push-based approaches, in contrast, avoid the issues identified for client-pull, but have the problem of deciding which data to send to clients in the absence of specific requests. Clearly, sending irrelevant data to clients is a waste of resources. A more serious problem, however, is that in the absence of requests it is possible that the servers will not deliver the specific data that are needed by clients in a timely fashion (if ever). Thus, the usefulness of server push is dependent on the ability of a server to accurately predict the needs of clients. A solution to this problem to allow the clients to provide a *profile* of their interests to the servers. A popular mechanism for providing such profiles is known as a *publish/subscribe* protocol (e.g., [Oki93]). Using publish/subscribe, clients subscribe to given classes of information by providing a set of expressions that describe the data of interest. These subscriptions form a profile — they are in effect, a continuous query. The information providers publish information that is tagged in a way that allows the matching of the data with the subscriptions of the users. When new data items are created or existing ones updated, the items are disseminated to the subscribers whose profiles match the items. The distributed system infrastructure is responsible for delivering data to all relevant subscribers. While publish/subscribe is useful for delivering new or modified data to clients, it can not be used to efficiently deliver previously existing data to clients that subsequently realize they need it. Such data is most easily obtained through a client pull.

3.2 Aperiodic vs. Periodic

Both push and pull can be performed in either an aperiodic or periodic fashion. Periodic delivery is performed on a regular, repeating schedule, while aperiodic has no such schedule. For example, an application that sends out stock prices on a regular basis is an example of periodic push, whereas one that sends out stock prices only when they change is an example of aperiodic push. Using periodic push, a set of data items is sent out repeatedly according to a prespecified schedule. Since the pattern repeats, a client that misses a data item on one period of the pattern, it can get it the next. In contrast, using aperiodic push, data are sent out, but not according to any particular schedule. This type of data delivery is what would arise in a publish/subscribe system.

Periodic push is useful for cases in which clients might not be available at all times or might be unable to react to what has been sent. Unavailability can occur in mobile settings where clients can become disconnected. It might also occur in cases in which the client processing load is high, thus preventing the continual servicing of the incoming data stream. On the other hand, aperiodic push assumes that clients are either always listening and able to respond to what is being sent or that missing some information is not crucial to the application. Aperiodic push has the advantage of making more effective use of the downstream communication channel, as the choice of items to deliver can be adjusted more quickly to the changing needs of an application.

Aperiodic pull arises in a system that uses the traditional request/response style of data delivery. Data is pulled from the server to clients whenever clients request it, for example, on a page fault or a cache (buffer) miss. In contrast, periodic pull arises when a client uses polling to obtain data from servers. The client requests data on a regular schedule. Such polling may be useful in certain applications such as remote-sensing, but as discussed previously, can be expensive in terms of server and client processing as well as backchannel usage.

3.3 Point-to-point vs. One-to-N

The third characteristic of data delivery mechanisms we identify is whether they are based on point-to-point or one-to-N communication. Using point-to-point communication, data is sent from a data source (e.g., a single server) to one other machine, while one-to-N communication such as broadcast or multicast allows multiple machines to receive the data sent by a data source. The tradeoffs between these approaches depend upon the commonality of interest of the clients. Using broadcast or multicast, scalability can be improved by allowing multiple clients to receive data sent using a single server message. Such benefits can be obtained, however, only if multiple clients are interested in the same items. If not, then scalability may actually be harmed, as clients may be continually interrupted to filter data that is not of interest to them.

Two types of one-to-N data delivery can be distinguished: multicast and broadcast. With multicast, data is sent to a specific subset of clients. Multicast is typically implemented by sending a message to a router that maintains the list of recipients. The router reroutes the message to each member of the list. Since the list of recipients is known, it is possible to make multicast reliable; that is, network protocols can be developed that guarantee the eventual delivery of the message to all clients that should receive it. In contrast, broadcasting sends information over a medium on which an unidentified and unbounded set of clients can listen. This differs from multicast in that the clients who may receive the data are not known *a priori*.

3.4 Classification of Delivery Mechanisms

It is possible to classify some existing data delivery mechanisms using the framework described above. Such a classification is shown in Figure 1.

- *Request/Response* - Traditional request/response mechanisms use aperiodic pull over a point-to-point connection. If instead, a one-to-N connection is used, then clients can "snoop" on the requests made by other clients, and obtain data that they haven't explicitly asked for. It is interesting to note, therefore, that in the one-to-N case, request/response looks like a pull-based approach to the client that initiates a request, but appears to be push-based to any other clients that are snooping on the broadcast.
- *Polling* - In some applications, such as remote sensing or control applications, a system may periodically send requests to other sites to obtain status information or to detect changed values. If the requested information is returned over a point-to-point channel, then this pull-based approach is what is usually considered to be polling. If the information is returned over a one-to-N link, then as with request/response, other clients can snoop to obtain data as it goes by.
- *Publish/Subscribe* - As stated previously, publish/subscribe protocols are becoming a popular way to disseminate information in a network. Publish/subscribe is push-based; data flow is initiated by the data sources (i.e., servers), and is aperiodic, as there is no predefined schedule for sending data. Publish/subscribe protocols are typically performed in a one-to-N fashion, but a similar protocol can be used over a point-to-point channel, as is done for triggers or notifications in an active database system.
- *Broadcast Disks* - The last category we examine is the combination of periodic and push. Periodic push has been used for data dissemination in many systems such as TeleText [Amma85, Wong88] and DataCycle [Herm87, Bowe92]. Clients that need to access a data item that is pushed periodically can wait until the item appears. In this way, periodic push can be thought of as a storage device whose average latency for an item is half the period at which that item repeats [Acha95a]. As with aperiodic push, periodic push can also be used with both point-to-point and one-to-N channels, but one-to-N is likely to be much more prevalent. We discus this method of data delivery in more detail in the following section.

4 Broadcast Disks

Having described the notion of data dissemination and outlined the different styles of data delivery that can be employed in a DBIS, we now focus on one particular approach that we have developed, called Broadcast Disks [Zdon94, Acha95b]. Using Broadcast Disks, a broadcast program containing all the data items to be disseminated is determined and this program is transmitted in a periodic manner. As stated above, the repetitive nature of the broadcast allows the medium to be used as a type of storage device; clients that need to access a data item from the broadcast can obtain the item at any point in the broadcast schedule that the item is transmitted. Periodic broadcast has been studied by a number of researchers (e.g., [Amma85, Herm87, Giff90, Imie94]). The Broadcast Disk model differs from previous data broadcasting work, however, in that it integrates two main components: 1) a novel "multi-level" structuring mechanism that allows data items to be broadcast non-uniformly, so that bandwidth can be allocated to data items according to the importance of those items; and 2) client-side storage management algorithms for data caching and prefetching that are tailored to the multi-disk broadcast. As is discussed below, these client-side policies differ in significant ways from those that would be used for the other styles of data delivery (e.g., pull-based, or aperiodic push-based).

4.1 Server-Side Broadcast Scheduling

The multi-level approach to broadcast scheduling used in Broadcast Disks allows data items to be placed on sub-broadcasts of varying size and periodicity. An arbitrarily fine-grained memory hierarchy can be created by constructing a multi-level broadcast such that the highest level is relatively small and repeats relatively frequently, while subsequent levels are progressively larger and repeat progressively less frequently. Such a broadcast program is analogous to a traditional memory hierarchy ranging from small, fast memories (e.g., cache) to larger, slower memories (e.g., disks or near-line storage). By placing data items on faster levels, the access time for higher-priority data can be reduced at the expense of increasing the latency for lower-priority items. For instance, in a highway traffic information system, information on accidents or potential delays can be placed on faster levels, while less time critical information (such as local point-of-interest listings) can be placed on slower levels. In this way, newly arriving motorists can receive data about incidents more quickly than they could if all information was broadcast in a uniform (i.e., flat) manner.

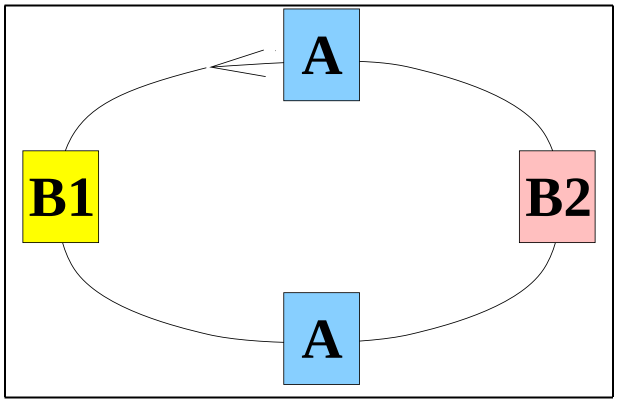

Figure 2: Multidisk with 3 distinct pages on 2 disks

Figure 2 shows a simple example broadcast of three pages arranged on two disks. The page labeled A is on a disk that is spinning twice as fast as the two pages that are labeled B1 and B2. The pages B1 and B2 are said to be on a slower spinning disk that is twice the size of the fast disk. The desirable characteristics for a multi-level broadcast program have been outlined in [Acha95a]. Briefly, a good broadcast program is periodic, has fixed (or nearly fixed) inter-arrival times for repeated occurrences of a data item, and allocates bandwidth to items in accordance with (but not in direct proportion to) their access probabilities. Multiple levels are superimposed on a single broadcast channel by interleaving the data items of the various levels in a manner that results in the desired relative frequencies. For example, the multidisk shown in Figure 2 would be generated by repeatedly broadcasting the pattern :“A, B1, A, B2, ...”. The algorithm used by the server to generate the broadcast program requires the following inputs: the number of levels , the relative frequency of each level and assignments of data items to the levels on which they are to be broadcast. The algorithm is described in detail in [Acha95a]. More recently, algorithms for automatically determining these inputs have been proposed [Baru96].

4.2 Profiles, Feedback, and Client-side Storage Management

In order to decide how to structure a broadcast schedule, the server must use its best knowledge and synthesis of the needs of the clients that require disseminated data. There are two potential pitfalls with this approach. First, the server is likely to have inaccurate data about the needs of the clients, as these needs can change quickly and are often hard even for the clients themselves to predict. Second, even given perfect knowledge of the access profiles of the clients, the server must *synthesize* an “average” profile on which to base its broadcast schedule; this average profile is unlikely to be optimal for any *individual* client. A key to solving this problem is the use of client storage resources for caching and prefetching data items. By intelligent management of local storage, a client can (to a large extent) isolate itself from many of the mismatches between its local priorities and the global priorities determined at the server.

We have developed implementable caching and prefetching algorithms to address this issue and have compared them to ideal policies [Acha95a, Acha96a]. The policies take into account both the local access probability (using heuristics such as LRU or usage frequency) and the expected latency of a data item; items that reside on slower levels have higher average latencies. The caching policies [Acha95a] favor slow-level data items, allowing clients to use their local storage resources to retain pages which are important locally, but are not sufficiently popular across all clients to warrant placement on a faster level. Broadcast Disks presents an excellent opportunity for prefetching because objects continually flow past the receivers. We have also developed client *prefetching* algorithms [Acha96a]. These algorithms are more dynamic than the caching policies because they take into consideration the time-to-arrival for items at a given instant. The prefetching policies are able to retain items in a client's cache during those portions of the broadcast cycle where their time-to-arrival is at its highest, which results in a significant performance improvement.

Intelligent client cache management goes a long way towards relaxing the need for strict accuracy in user profile management. The feedback from user requests and status information delivered over a backchannel, however, can provide important information to help maintain the profile information at server. Algorithms for dynamically maintaining profiles and adjusting the broadcast accordingly are thus, important directions in our current work.

4.3 Data Consistency

We have extended the mechanisms described above in order to incorporate data item updates and have studied their effect on performance [Acha96b]. As in the previous work, the design considerations divide into client-side and server-side issues. The server must decide how to modify its broadcast program in order to most effectively communicate updates to the client. The server can use both invalidation and propagation. Timely invalidation (i.e., notification of updates) is the minimal requirement in order to enable the preservation of data consistency. The server can also choose to propagate the new values of updated items in order to enhance performance. The client must perform appropriate actions to bring its cache contents back to a good state after an update. For example, the clients can prefetch items that have been invalidated from its cache. We have implemented this *automatic* prefetching and have integrated it with the regular (non-update) prefetching algorithms. Alternatively, a client can choose to ignore an invalidation if it determines that retaining a stale copy of a data item is preferable to having no copy of that item.

Two different consistency models have been studied — immediate and periodic (i.e., based on the period of the broadcast). Our studies have shown that under both models, for low to moderate update rates it is possible to approach the performance of the case in which no updates occur. In other words, the broadcast mechanism can be made quite robust in the face of updates. One reason for this is that when invalidated items are automatically prefetched by clients, the broadcast program itself can act as a propagation medium. Explicitly propagating an item, therefore, is useful only if clients are likely to need to access that item long before it would naturally appear in the broadcast; otherwise, the propagation wastes bandwidth.

5 Towards an Integrated DBIS

One of the key lessons of the Broadcast Disks work has been the importance considering server-side, client-side, and network issues when developing a data dissemination technique. As described in Section 3, the periodic push approach used by Broadcast Disks is just one potential technique for delivering data in a DBIS. Our current work is focused on developing a more general dissemination-based information system architecture that is capable of supporting a wide range of applications across many varied environments (e.g., mobile networks, satellites, wide-area networks). The key to achieving this flexibility is to combine the various data delivery techniques in a way that achieves the most efficient use of the communication channels, while providing clients with responsive access to data.

We view an integrated DBIS as a system in which the links between the computing elements vary in character: from standard pull-based connections to periodic broadcast-based links. The choice of the data delivery mechanism to use between two given nodes will be made based on the properties of the available communications channels and applications as well as economic concerns. These factors may be stable or may change over time, requiring the DBIS to adapt dynamically. A key point is that the character of a link should be of concern only to the nodes on either end. For example, the fact that an information provider receives data from a broadcast disk as opposed to a request/response protocol should make no difference to clients of that provider.

In an integrated DBIS, there will be three types of nodes: (1) *data sources*, which provide the base data that is to be disseminated; (2) *clients*, which are net consumers of information; and (3) *information brokers*, that acquire information from other sources, add value to that information (e.g., some additional computation or organizational structure) and then distribute this information to other consumers. By creating hierarchies of brokers, information delivery can be tailored to the needs of many different users.

The goal is to provide the architectural components that can be used to construct a DBIS. A builder of an information resource would make use of a library of these components to construct the interfaces to their service. Example components include a broadcast generator, a set of dissemination services, a client cache manager, a client prefetcher, a backchannel monitor, etc. In the following, we briefly outline some of the fundamental issues in the design of an integrated DBIS:

- *Bandwidth Allocation* - For a given link, policies are needed for allocating bandwidth among the various data delivery mechanisms.

- *Push Scheduling* - For the push-based approaches, intelligent scheduling is necessary in order to obtain the maximal benefit from the available bandwidth. Scheduling must also take into account the likelihood and distribution of transmission errors. Also, for periodic push, the broadcast should include index and/or schedule information that describes the objects that are to appear in the upcoming broadcast. Such information allows clients to minimize the amount of time and/or processing that they devote to monitoring the broadcast and can aid in storage management decisions.

- *Client Storage Management* - Clients must allocate their storage resources among the data obtained through the various delivery mechanisms. Furthermore, as stated earlier, different methods of data delivery impose differing demands on the policies for client caching and prefetching. Furthermore, in some cases (e.g., mobility), storage management must also take into account the likelihood of disconnection and of data becoming stale due to updates or expiration.

- *User Profiles and Feedback* - Profiles of client needs are key for making allocation, scheduling and other policy decisions at both clients and servers. The form of the profiles will be important to achieve the most effective use of the medium. For example, access probabilities are one specific representation of the client needs. The server must also have effective models for combining client profiles. The integration of a *backchannel* from clients to servers is needed to allow for updating profiles and making additional requests.

- *Security Issues* - Another set of important issues that must be addressed revolves around the security and privacy concerns that arise in any distributed information system. The emphasis on one-to-N commuincation in a DBIS, however, increases the significance of such issues.

- *Consistency Issues* - The final issue we list here is the maintenance of data consistency, particularly in the face of possibly intermittent connection. Two types of consistency must be considered. First, guarantees on the timeliness of individual data items must be provided if required by the clients. Second, *mutual consistency* across multiple items will be required in some instances. All types of consistency must be provided in a flexible manner, so that tradeoffs between consistency and responsiveness can be made on a case-by-case basis.

6 Summary

The increasing ability to interconnect computers through internetworking, mobile and wireless networks, and high-bandwidth content delivery to the home, has resulted in the proliferation of information delivery applications. A key attribute of many such applications is their inherent asymmetry. These applications present new challenges for data management throughout all components of a distributed information system. We have proposed the notion of a dissemination-based information system that integrates many different data delivery mechanisms and described some of the unique aspects of such systems. We also described the Broadcast Disks data dissemination paradigm and showed how the choice of data delivery mechanism effects both clients and servers. Our current work is aimed at integrating Broadcast Disks with a pull-based request/response mechanism. We view this work as a step towards the development of a more general DBIS architecture.

References

[Acha95a] S. Acharya, R. Alonso, M. Franklin, S. Zdonik, “Broadcast Disks: Data Management for Asymmetric Communications Environments”, *Proc. ACM SIGMOD Conference*, San Jose, CA, May, 1995.

[Acha95b] S. Acharya, M. Franklin, S. Zdonik, “Dissemination-based Data Delivery Using Broadcast Disks”, *IEEE Personal Communications*, 2(6), December, 1995.

[Acha96a] S. Acharya, M. Franklin, S. Zdonik, “Prefetching from a Broadcast Disk”, *12th International Conference on Data Engineering*, New Orleans, LA, February, 1996.

[Acha96b] S. Acharya, M. Franklin, S. Zdonik, “Disseminating Updates on Broadcast Disks”, *Proc. 22nd VLDB Conference*, Mumbai (Bombay), India, September, 1996.

[Amma85] M. Ammar, J. Wong, “The Design of Teletext Broadcast Cycles”, *Perf. Evaluation*, 5 (1985).

[Baru96] S. Baruah and A. Bestavros, ”Pinwheel Scheduling for Fault-tolerant Broadcast Disks in Real-time Database Systems”,Technical Report TR-96-023, Boston University, August, 1996.

[Bowe92] T. Bowen, G. Gopal, G. Herman, T. Hickey, K. Lee, W. Mansfield, J. Raitz, A. Weinrib, “The Datacycle Architecture”, *CACM*, 35(12), December, 1992.

[Giff90] D. Gifford, “Polychannel Systems for Mass Digital Communication”, CACM, 33(2), February, 1990.

[Herm87] G. Herman, G. Gopal, K. Lee, A. Weinrib, “The Datacycle Architecture for Very High Throughput Database Systems”, *Proc. ACM SIGMOD Conf.*, San Francisco, CA, May, 1987.

[Imie94] T. Imielinski, S. Viswanathan, B. Badrinath, “Energy Efficient Indexing on Air”, *Proc. ACM SIGMOD Conference*, Minneapolis, MN, May, 1994.

[Oki93] B. Oki, M. Pfluegl, A. Siegel, D. Skeen, “The Information Bus - An Architecture for Extensible Distributed Systems”, *Proc. 14th SOSP Conference*, Ashville, NC, December, 1993.

[Wong88] J. Wong, “Broadcast Delivery”, *Proceedings of the IEEE*, 76(12), December, 1988.

[Zdon94] S. Zdonik, M. Franklin, R. Alonso, S. Acharya, “Are ’Disks in the Air’ Just Pie in the Sky?”, *IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, December, 1994.

Multicast Support for Data Dissemination in OrbixTalk

David Glance Ph.D.
IONA Technologies Pty. Ltd.
Suite 15, 189 St George's Terrace
Perth, WA 6000, Australia
+61 9 322 4222
`davidg@iona.com.au`

1 Introduction

Traditional mechanisms of inter-object communication have focused on a point-to-point request/reply approach with one object communicating with one other object. In many instances however, there is the requirement for a many-to-many form of object communication, or what is also commonly referred to as, Publish/Subscribe style of communication.

In Publish/Subscribe communication, publishers "publish" information onto a network, labeling the information using a hierarchically structured name. Subscribers "subscribe" to this information by using the same name. The information referred to by that name will be then delivered to the subscriber. Since subscribers and publishers share information only by referring to the information label and not the particular source or destination of the information, they can remain completely anonymous. The emphasis here is the information itself and not any given source of this information.

Another feature of publish/subscribe communication is that it emphasizes a "push-based" style of information dissemination which is different from the standard "pull-based" form which is implemented by databases queries and the World Wide Web for example. The former style is better suited to dynamic data (such as news bulletins, stock market prices, etc.) whereas the pull-based communication style is better suited to static data (such as a client's address for example).

An example of publish/subscribe communication is the use of email groups. A publisher can send an email to a particular group (say, news@iona.com) without being aware of who is a member of that group. By joining the email group, subscribers can receive mail on the topic of news. Of course, in email groups, information is sent with the email to identify the sender, but this is simply convention .

Another example of the use of this form of communication is objects sharing stock market price information. The price of a stock may originate from more than one object (different markets for example) which is published using a name such as //IBM/PRICE (the naming convention is that used by OrbixTalk - see below). Objects interested in this price information, such as market price displays, may receive these messages by simply subscribing to the information using this name.

IONA has implemented a multicast messaging system known as OrbixTalk which allows a publish/subscribe form of communication. This has been implemented using TCP/IP's IP Multicast providing a fast, efficient and scalable service. OrbixTalk multicast messaging is asynchronous, and allows suppliers and consumers to be to-

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

tally decoupled. OrbixTalk has been used within IONA's Object Request Broker, Orbix to implement the Event Services specified within the Object Management Group's Common Object Request Broker Architecture (CORBA).

2 Information is identified by Topic Names

OrbixTalk allows objects to supply or consume messages by referring to the messages using a message "Topic Name". This name is hierarchical with the format based on the Uniform Resource Locator of the World Wide Web. For example, objects may use the topic name "otrmp://STOCK_PRICE/IBM" to supply and consume IBM stock price messages. The "otrmp" portion of the topic name refers to the protocol, OrbixTalk Reliable Multicast Protocol. Any number of objects may be suppliers ("talkers") or consumers ("listeners"). Each part of the name hierarchy may be "wild-carded" by using a "$*$". E.g. "otrmp://STOCK_PRICE/$*$" will match all stock prices. Each portion of a Topic Name may be any length and their may be up-to five parts to a name (this has been arbitrarily set at present).

By using the topic name, suppliers and consumers are decoupled from each other since they do not have to be aware who the suppliers of information are and likewise, the suppliers are not aware of the consumers of that information. The advantage of this is that it is the consumer that decides what information it needs not the supplier. Thus if the functionality of the consumers change, the suppliers do not have to change with them.

3 The multicast protocol

The OrbixTalk reliable multicast protocol is based on TCP/IP's IP Multicast extension which uses UDP. Since UDP is inherently unreliable, OrbixTalk implements reliability using a negative acknowledgment scheme which operates in the following manner. Messages are allocated a sequence number and the fragmented into configurably sized packets. The packets are then sent out on an IP multicast address (how this address is assigned is dealt with below) and stored by the sender for possible re-transmission. The consumer receives the packets and assembles them in the correct order. If the consumer detects a missing packet, it re-requests the packet from the sender by sending a negative acknowledgment. The sender, on receiving a negative acknowledgment, retransmits the requested packets. The consumer will re-send the negative acknowledgment a certain number of times if it does not receive the missing packets. The sender keeps up to a configurable number of messages for re-transmission and for a certain period of time before they are deleted.

4 Topic name to IP Multicast Group Address translation

Topic names are actually mapped to IP multicast groups which are represented as class D IP addresses in the range 224.0.0.0 to 239.255.255.255. A daemon process, the OrbixTalk Directory Enquiries daemon is responsible for taking a topic name and allocating an IP multicast group. Since IP multicast groups actually map to multicast Ethernet addresses, a large amount of filtering of uninteresting packets is done at the hardware level and not by the IP or application software. In reality, the process is not quite as efficient as this since interfaces have limits to the number of multicast groups that can be supported at any one time. This means that topic names are necessarily multiplexed on any given multicast group. To increase the efficiency of the topic name multiplexing that occurs on a multicast group, the OrbixTalk Directory Enquiries daemon translates the topic name into unique integers for efficient and speedy comparison.

5 Guaranteed Delivery of OrbixTalk Messages

OrbixTalk provides a mechanism to persistently store messages sent by a supplier and thus provide guaranteed delivery to consumers. This protocol is known as the OrbixTalk Store and Forward protocol (otsf). A process called the OrbixTalk Message Store is responsible for receiving messages from talkers and storing them persistently on disk. On receipt of a message from a talker, the messages are stored in a sequential indexed file and then a positive acknowledgment is sent back to the sender. The talker process records the successful delivery of the message in an individual state file.

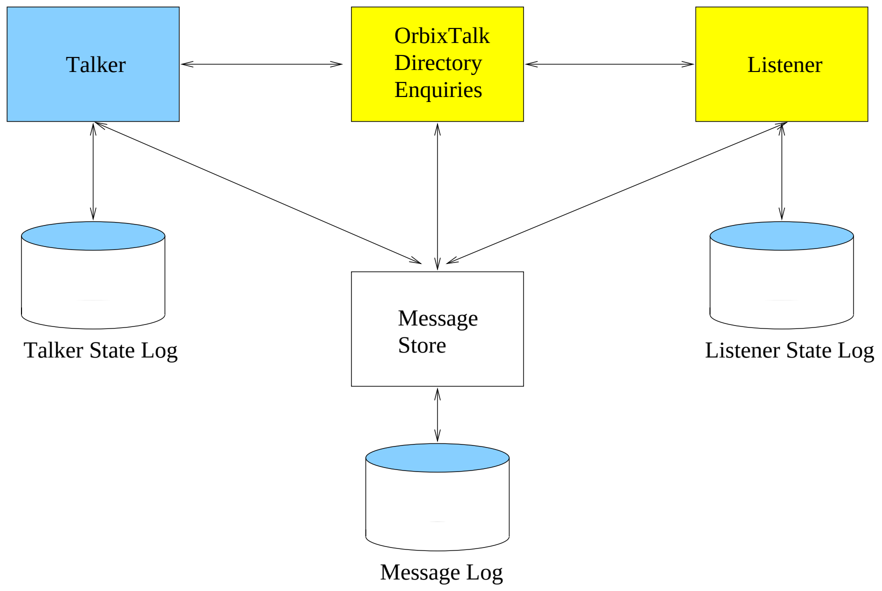

Figure 1: OrbixTalk Message Store Components

Once the message is stored in the Message Store log, the message is forwarded to all listeners using the OrbixTalk Reliable Multicast Protocol. The listener checks that the message is the next correct message that it was expecting. If it is, the listener passes the message to the user application and then updates its individual state log marking the message as successfully received. If the listener detects a gap in the message sequence (i.e. it has lost a message or messages), it can ask for a replay of the missing message(s) from the Message Store.

On start-up, a listener will check its state log and may optionally request all messages that it missed whilst not running. Optionally, this can be narrowed to a specific Topic Name or messages from a specific application. Message Store processes are identified by unique Topic Names (e.g. //OrbixTalk/MessageStore/1). Talkers and listeners are also given unique names which are used to identify the application's message state file. Having the application state controlled by the applications themselves rather than the Message Store process greatly simplifies the processing of the Message Store. At present, the Message Store log is implemented as a series of sequential files. As the file grows to a configurable size, a new log is opened. The Message Store will keep rolling-over logs to enable old logs to be archived. By using a negative acknowledgment scheme of message distribution from the Message Store process to the listeners, message delivery is optimized and is much faster and more scalable than if a positive acknowledgment, point-to-point scheme had been adopted (as is common with other store-and-forward implementations).

6 Future Enhancements of OrbixTalk

6.1 Transactional Delivery of OrbixTalk Messages

Work is in progress to enhance OrbixTalk to support transactional guarantees of message delivery. Using the CORBA Object Transaction Service, OrbixTalk will be able to send messages as part of a transaction using the 2-phase commit protocol. In this way, messages may be sent within a transaction which actions on databases or other resources. If a transaction is committed, the messages are guaranteed to have been stored by the Message Store process. Likewise, if a transaction aborts, the messages will not be stored on the log.

Within this transaction, the Message Store and the OrbixTalk library act as resource managers controlling the updates to the talker state log and the Message Store log. On commit of the transaction, the message is committed to the Message Store log and the update to the talker's state log is made. In this context, acknowledgments are unnecessary.

On the listener side, once a message is delivered, the application may start a distributed transaction. Here, on commit, it is the information to the state log that is committed. In this way, the listener can ensure exactly once delivery of the message to the listener application. Should the message delivery be aborted, that message is not delivered again.

6.2 OrbixTalk Administration

Administration facilities will be enhanced to allow easy configuration of OrbixTalk applications. Users, will be able to browse Message Stores and view the contents of messages. Messages may be held, deleted, or resent. Reconciliation between senders and receivers of messages may be carried out.

It is planned that the administration and monitoring facilities of OrbixTalk will be implemented using the wider administration system being implemented for Orbix with gateways into SNMP and CMIS/CMIPS.

6.3 Fault-Tolerance

The Message Store process will be capable of being run in fault-tolerant hot-standby mode. In this mode, two Message Store processes will run with one as the primary process, and the other as the secondary process. Both processes will receive messages from talkers and both will update separate logs. However, only the primary process will acknowledge the receipt of the messages and will forward them onto the listeners. The secondary will take over as primary when it detects the failure of the primary process.

To ensure that the secondary process's logs are consistent with the primary's, the primary will send periodic check-point messages to the secondary to inform it of the current sequence numbers for all topics being stored. The secondary process will be able to replay messages from the primary.

7 OrbixTalk and the CORBA Event Services

OrbixTalk has been used within Orbix to implement the CORBA Event Services. In the event services, CORBA talks about suppliers and consumers which may communicate with each other indirectly using event channels. Event Channels are defined in IDL (Interface Definition Language) and are thus conceptually objects themselves. Communications via these event channels may use either the push or pull models. The push model is controlled by the supplier and is similar to the form of communication outlined in previous sections above. In the pull model, the consumer may request an event from the supplier. Events are supplied via an event channel and may be either typed or untyped.

To take as an example, we will construct a simple push supplier (i.e. a talker) which will push one event onto an event channel and a push consumer (a listener) which will consume that event from the event channel.

7.1 Push Supplier

Referring to the code shown in Figure 2, the following steps are taken to code a simple push supplier:

1. The push supplier initializes the OrbixTalk manager.
2. It then binds to an event channel by using the CORBA event channel administration facility, identifying the event channel by using an OrbixTalk Topic Name.
3. Having obtained the event channel, a reference to the supplier administration object is obtained and used to obtain a reference to the proxy push consumer interface of the event channel (to the push supplier, the event channel acts as a proxy for the push consumer). The push supplier then connects to the proxy push consumer.
4. An untyped variable (CORBA::any) is then constructed and it is then pushed onto the event channel.
5. The OrbixTalk events are then processed to allow the communication to take place.

```
main(int argc, char ** argv) {

OrbixTalk_ptr   aOrbixTalkManager;

CosEventChannelAdmin::EventChannel_var         aEventChannel;
CosEventChannelAdmin::ConsumerAdmin_var        aConsumerAdmin;
CosEventChannelAdmin::SupplierAdmin_var        aSupplierAdmin;
CosEventChannelAdmin::ProxyPushConsumer_var    aProxyPushConsumer;

// [1] initialise OrbixTalk

aOrbixTalkManager = OrbixTalk::initialise();

// [2] bind to the event channel

aEventChannel = CosEventChannelAdmin::EventChannel::_bind("otrmp//IBM/PRICE");

// [3] Get a reference to the event channel supplier administration channel
// and register as a push supplier

aSupplierAdmin = aEventChannel->for_suppliers();
aProxyPushConsumer = aSupplierAdmin->obtain_push_consumer();
aProxyPushConsumer->connect_push_supplier(0);

CORBA::any  aAnyVariable;

// [4] construct an any variable (this is an untyped event)

aAnyVariable <<= 32.5;

aProxyPushConsumer->push(aAnyVariable);

// [5] process events for a short time, then exit

CORBA::Orbix.processEvents(5000);

}
```

Figure 2: Push Supplier Code

7.2 Push Consumer

Referring to the code shown in Figure 3, the following steps are taken to code a simple push consumer:

1. A class is constructed which inherits from the Basic Object Adapter implementation of the PushConsumer class.
2. The PushConsumer class receives the push event and then prints out the event.
3. The push consumer initializes the OrbixTalk manager.
4. It then binds to an event channel by using the CORBA event channel administration facility, identifying the event channel by using an OrbixTalk Topic Name (note that this is the same name used by the push supplier above).
5. Having obtained the event channel, a reference to the consumer administration object is obtained and used to obtain a reference to the proxy push supplier interface of the event channel (to the push consumer, the event channel acts as a proxy for the push supplier). The push consumer then connects to the proxy push supplier.
6. The OrbixTalk events are then processed to allow the communication to take place.

Although the CORBA Event Services does not specify a communications protocol that should be used in implementing the Event Services, other vendors have used the Internet Inter-ORB Protocol (IIOP). To ensure interoperability between Orbix and other ORBs, an IIOP/OrbixTalk gateway will be provided.

This gateway will also be used to implement the Orbix Event Services in JAVA (although a direct port of OrbixTalk to JAVA is another possibility).

8 Uses of OrbixTalk

It is envisioned that there will be a multitude of application uses for both the reliable and guaranteed delivery modes of OrbixTalk.

8.1 Making the World Wide Web Dynamic

One example would be enhancement of the World Wide Web (WWW). By using Topic Names to identify message streams, an information space made up of messages may be constructed in much the same way that the World Wide Web is an information space made up of documents. The WWW is implemented using a “pull” model which is static in its configuration. OrbixTalk would allow browsers to have links which access information dynamically, irrespective of location using a “push” model of information dissemination.

For example, a page could be opened which allows news headlines to be sent as they occur. To do this today, the user has to typically “refresh” the page - i.e. re-request it. This would be achieved by subscribing to the topics that you wished to see e.g. //NEWS/SPORT/BASEBALL/SCORES. This would start by fetching the latest scores and then the page would update dynamically each time a new score came in.

The addition of a communication system such as OrbixTalk to browsers together with a language such as Java and Java ORBs increases the sophistication of this environment for building industrial strength applications.

```
// [1] Set up a push consumer class which will receive the push and disconnect

class PushConsumerClass : public CosEventComm::PushConsumerBOAImpl
{
public:
  void push(const CORBA::any & aAnyVariable, CORBA::Environment & aEnvironment);
  void disconnect_push_consumer(CORBA::Environment & aEnvironment);
};


// [2] Print out the received push event

void PushConsumerClass:: push(const CORBA::any & aAnyVariable,
CORBA::Environment & aEnvironment)
{
   char * aPrice;

   aAnyVariable >>= aPrice;

   cout << "Received a price: " << aPrice << endl;
}


void PushConsumerClass:: disconnect_push_consumer(CORBA::Environment &
aEnvironment)
{
  cout << "Got a disconnect " << endl;
}
main(int argc, char ** argv) {

OrbixTalk_ptr    aOrbixTalkManager;

CosEventChannelAdmin::EventChannel_var          aEventChannel;
CosEventChannelAdmin::ConsumerAdmin_var         aConsumerAdmin;
CosEventChannelAdmin::ProxyPushSupplier_var     aProxyPushSupplier;

CosEventComm::PushConsumer_var aPushConsumer = new PushConsumerClass;

// [3] initialise OrbixTalk

aOrbixTalkManager = OrbixTalk::initialise();

// [4] bind to the event channel

aEventChannel = CosEventChannelAdmin::EventChannel::_bind("otrmp//IBM/PRICE");

// [5] Get a reference to the event channel consumer administration channel
// and register as a push consumer

aConsumerAdmin = aEventChannel->for_consumers();
aProxyPushSupplier = aConsumerAdmin->obtain_push_supplier();
aProxyPushSupplier->connect_push_consumer(aPushConsumer);

// [6] wait for 60 seconds for an event

CORBA::Orbix.processEvents(60000);

}
```

Figure 3: Push Consumer Code

8.2 Financial Systems

Market data distribution has always been a prime example of broadcast-based information dissemination systems. A price arriving on a trading floor may be disseminated to a number of dealer positions, updating their screens in real-time. OrbixTalk, by its use of IP Multicast, makes the network communication more efficient.

For example, the price of IBM's stock as quoted on the New York stock exchange may be referenced using a Topic Name of "otrmp//STOCK/IBM/NY". The application receiving information about the IBM price from the NY stock exchange would publish this information using this topic name. Applications wishing to access information concerning IBM's stock price, would subscribe to this information using the same topic name.

Another application may be receiving information about the IBM stock price as it is traded on the Frankfurt stock exchange. It could publish this information using the Topic Name of "otrmp//STOCK/IBM/FRANKFURT". An application wishing to receive information concerning IBM's stock price on all exchanges is able to wild-card the final part of the name, i.e. "otrmp//STOCK/IBM/$*$". This subscriber would then receive information on IBM's NY price and its Frankfurt price.

Again in the financial dealing room setting, other financial objects can be treated in the same way, such as trades, orders, position information, etc.

8.3 System Administration

Use of OrbixTalk allows application information to be disseminated as events using a push model. Application attributes may be pushed out for systems administration monitoring applications to collect and display. Gateways are being constructed to convert OrbixTalk messages into SNMP and CMIS/CMIP format.

OrbixTalk is ideal also for "dynamic discovery" of objects located on a network by using well-known Topic Names to allow objects to be interrogated as to their location, name and in fact, any other attribute of interest.

8.4 Implementation of a Plug'n'Play System Architectures

OrbixTalk allows objects to communicate anonymously with loose coupling. This means that an object may contribute information without regard for which objects are going to receive the information. Likewise, an object may listen to messages without regard to their source (or sources). In this way, objects are totally decoupled. It is the consumer that decides what information it needs rather than the supplier determining this for it. New consumers may be added or removed without changing the objects that supply information.

For example, an order entry system may accept an order for an automobile spare part. Once the order is accepted, this information may be of use to a number of systems such as accounting (there is now an order outstanding and invoices must be sent), warehouse (the part needs to be found) and management information (the management need to see how the sale of each of the parts is doing). In a traditional architecture, the order entry system would have to know about all of the other systems requiring the information about the order (i.e. it would be tightly coupled to the other systems) and would typically send this order to those systems using a request/reply mechanism. Using OrbixTalk, the order entry system just publishes the order without concern for which systems require that information. The warehouse, accounting and management information systems just subscribe to receive the order information.

Now add an extra system to the architecture, say a just-in-time ordering system which, on receipt of an order, sends a request for a part from the central warehouse. By using OrbixTalk, this system can be added without changing the order entry system, or any other component, it simply subscribes to the order information to receive it. This is where the true power of OrbixTalk and its publish/subscribe style of communication lies.

Further Reading

[1] OrbixTalk White Paper: http://www.iona.com/Orbix/Talk/WhitePaper/OrbixTalk.html

[2] Reliable Multicast Protocols: http://research.ivv.nasa.gov/projects/RMP/links.html

Genesis: An Approach to Data Dissemination in Advanced Traveler Information Systems [*]

Shashi Shekhar and Andrew Fetterer
Department of Computer Science, University of Minnesota
Minneapolis, MN 55455
shekhar@cs.umn.edu fetterer@cs.umn.edu

Duen-Ren Liu
Institute of Information Management, National Chiao Tung University
Hsinchu, Taiwan 300, R.O.C.
dliu@iim.nctu.edu.tw

Abstract

Genesis and ATIS are being developed under the umbrella of the Intelligent Transportation Systems to facilitate various kinds of travel, including daily commuting to work via private/public transportation and visiting new places and points of attraction. Travelers are naturally mobile, and the most effective way to disseminate information to travelers is with wireless communication, which is being explored by the Genesis project in Minnesota. The travelers can use personal communication devices including pagers and portable computers (e.g. Apple Newton) to avail themselves of the ATIS services provided by Genesis. This extended abstract presents an overview of the goals and preliminary design of Genesis, and illustrates the data-dissemination needs of ATIS. This paper describes the application domain, rather than evaluating candidate solutions.

1 Introduction

Advanced Traveler Information Systems (ATIS) is one facet of Intelligent Transportation Systems (ITS) [2, 3] , which are currently being developed to improve the safety and efficiency of automobile travel. ATIS assist travelers with planning, perception, analysis and decision-making to improve the convenience, safety and efficiency of travel [4, 5, 6, 7, 8] . ATIS applications create a shared resource for efficient data integration and dissemination. As shown in Figure 1, ATIS obtain information from different sources, including traffic reports, scheduled traffic events, sensors, and maps. Periodic sensor data might lead to high update rates. The clients of the database include travelers on the road, mobile persons with hand-held or portable PCDs (personal communication devices), and users who access information via computers at home, the office, shopping mall or information center. A large

[*]This paper is a shortened version of the paper which appeared in the NSF MOBIDATA Workshop [1] on Mobile and Wireless Information Systems, Nov. 1994.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

number of travelers will query databases over the wireless communication channel to seek traffic information and driving advisories. Data dissemination is the core of ATIS.

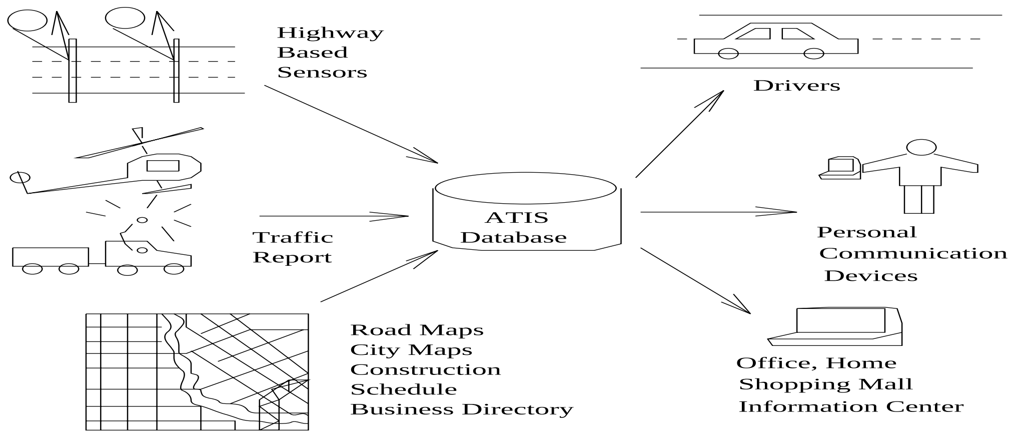

Figure 1: Data Sources and Data Dissemination using an ATIS database

1.1 ATIS Services

ATIS provides up-to-the minute information on weather and road conditions, detours, construction zones, bus schedules and parking. Travel information is available before and during your trip. ATIS performs a variety of functions, including navigation using digital maps; route selection and guidance; information on services such as gas stations, restaurants, and hospitals; and real-time traffic information through communication between travelers and the Advanced Traffic Management System (ATMS). These functions are categorized into six services and depending on system architecture, each service may be configured to be run on either the mobile clients or on the remote server.

The **Traveler Information Service** includes the business directory and travel information databases which are integrated with the road-map database to provide information about tourist attractions, hotels, restaurants, and other useful information. Each user will have a personal profile maintaining default travel-time preferences, alarm parameters and advisory-recognition preferences. Travel-time preferences include desired trip start or stop time and travel duration. Advisory parameters include travel duration, expected bus delay, real-time, and planned event advisories. User-specific information such as preferred routes and trip origin/destination are also maintained.

Pre-trip Travel Information Service provides information that is useful before the trip begins, such as road and weather conditions. The trip-planning service is also provided by allowing the user to set trip routes using roadway or MTC (Metropolitan Transit Commission) bus routes, to plan the trip schedule, and to receive estimated travel duration and known advisories for the planned trip. A planned-event advisory provides the traveler with a list of scheduled events which may interfere with normal travel routines. Events reported by the fixed-end server include construction, lane closures, maintenance and special events. Pre-trip planning also includes fixed bus information such as the bus schedule, fare and bus-route information. Delays for specific bus routes are also disseminated.

The **Route-Guidance Service** recommends the most favorable routes from a starting location to any chosen destination. The “favorable” route may represent the shortest distance, minimum travel time, etc. This service guides the user from a location to the destination. A trip-planning function allows the user to maintain trips using primary and alternate routes to destinations, to receive estimated travel durations and known advisories for the planned trip, etc. Trip scheduling allows the user to select a planned trip as the active trip, receive current and estimated travel duration and known advisories for the active trip, etc. Trip status allows the user to receive travel duration, real-time, planned-event, and bus-exception advisories. Figure 2 shows the Trip Management function [9] in Genesis. Route optimization determines ”favorable” routes from the vehicle’s current position to a chosen destination. The route guidance service also supports route evaluation capabilities, which evaluate a set of alternate routes between origin and destination, based on the current travel-time, congestion, restrictions and other attributes of the transportation network. The in-vehicle processor is able to receive traffic information on incidents, including travel times on affected routes, from the Traffic Management Center (TMC). The system determines if the traveler’s selected route is affected, calculates a new route if necessary, and informs the traveler that a revised route is available.

The **En-route Traveler Advisory Service** provides information available en-route such as construction zones, traffic congestion, traffic incidents, weather conditions and detours. Incidents are received and displayed using real-time location-specific incident information from the fixed-end server. Incident information includes potential hazards or delays due to unusual occurrences such as traffic accidents or excessive congestion. Incident reports include travel-duration advisories and real-time advisories. A travel-duration advisory is the dissemination of the difference between the current travel duration of a link and the historical average travel duration of the link for the current time period. A real-time advisory is the dissemination of current abnormal conditions for specific links.

The **Emergency Notification and Personal Security Service** uses pagers, cellular phones or callboxes for “mayday” purposes. When initiated by the traveler, or initiated automatically in case of accident, this feature will help summon emergency assistance and provide the vehicle location.

The **Information Retrieval and Display Service** retrieves information from the TMC and displays that information on the user interface. Communications functions support the data transmissions between the fixed-end database server and PCDs. Several approaches to wireless communication are being explored; including Infrared and RF-beacon technologies, FM sideband technology, Mobile satellite services, Cellular phone technology and Radio-Frequency (RF) data communication networks [8, 10] . The user interface communicates with the user (traveler), accepting requests for service and delivering driving directions and other information. Direction screens present simple, highly stylized graphics that can be taken in at a glance. The system can be augmented with a digitized or synthesized voice for traffic information and route guidance. Alternatively, head-up displays provide more flexible and useful guidance.

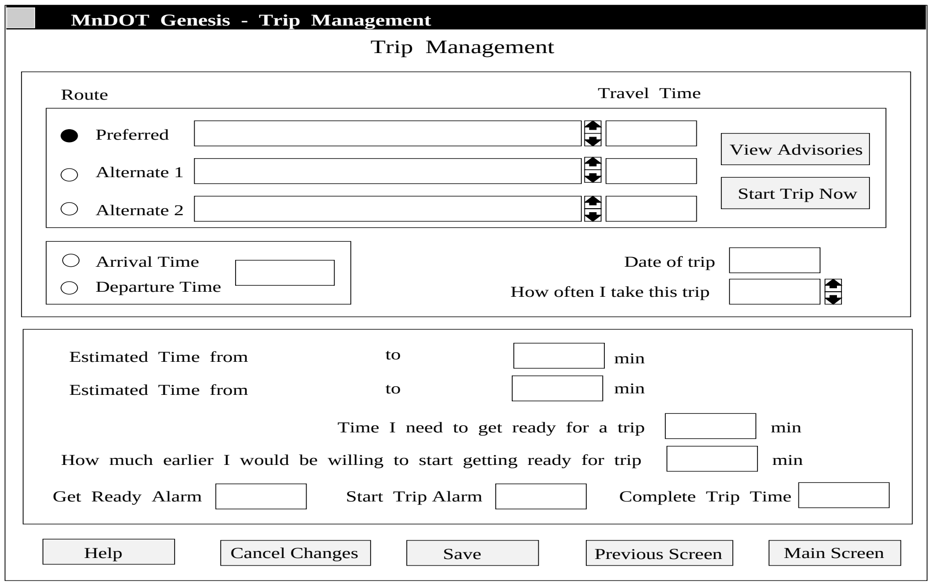

Figure 2: Trip management function in Genesis

1.2 Scope and Outline

This paper attempts to describe an important application which may showcase data dissemination issues in ATIS. We focus on describing the application rather than evaluating the technology.

Section 2 describes the Genesis system. Section 3 presents data management in Genesis. Query processing issues in Genesis are discussed in section 4. Finally, section 5 summarizes our discussions.

2 The Genesis System

In this section, we describe the components of the Genesis system. Genesis [11] is an ATIS operational test project developed in Minnesota. It is a partnership between the University of Minnesota, the Minnesota Department of Transportation, and private sector companies. The goal of Genesis is to test the effectiveness of an advanced portable traveler information service to provide comprehensive real-time travel data. Traveler information, including trip planning, bus transit, traffic and parking, etc., will be provided via a family of fully portable personal communications devices(PCDs). Initially, three types of devices are being evaluated: an alphanumeric pager, a notebook computer, and a personal digital assistant. PCDs are being evaluated in a multi-phase operational test throughout the Minneapolis - St. Paul metropolitan area. The first phase is a pilot project, providing reports on incidents only. Later phases will examine the provision of additional information via PCDs.

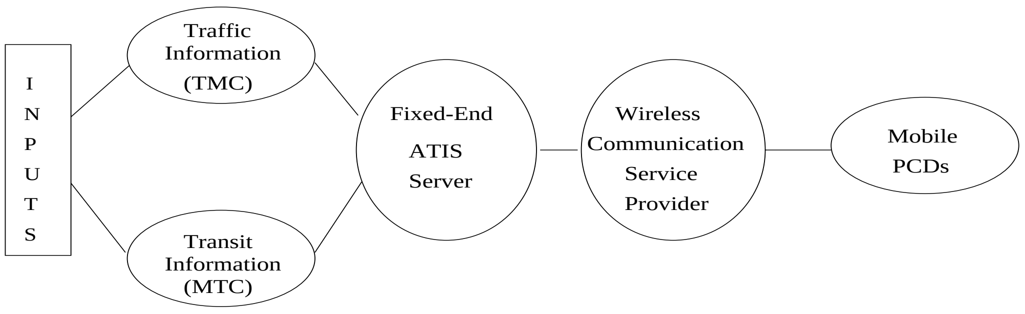
INPUTS
Traffic Information (TMC)
Transit Information (MTC)
Fixed-End ATIS Server
Wireless Communication Service Provider
Mobile PCDs

Figure 3: Genesis Operational View

The Genesis system includes data collection stations such as the Traffic Management Center and Metropolitan Transportation Commission, along with the fixed-end ATIS database server storing historic travel times, bus schedules and fares, and other static information; the wireless communication service provider and mobile PCDs (Personal Communication Devices). Figure 3 shows an operational view of Genesis [9] .

The TMC is the communications control and computer center for managing traffic on Twin Cities Metropolitan Area Freeways. The TMC is equipped with both video and radio monitoring and broadcasting equipment, in addition to traffic management workstations. The TMC uses networks of sensors, communication channels, and information sources to acquire traffic data. The MTC collects data about current bus locations (public transportation) and schedule exceptions.

The fixed-end ATIS server is the functional data processing element. Real-time data is collected, converted, processed and stored by the fixed-end server. Data is disseminated to the mobile clients via a communication service. Data dissemination occurs at the request of mobile clients or is sent by the fixed-end server as a result of a pre-determined data transaction.

The communication service provider has the responsibility of disseminating data from the fixed-end server to mobile clients (PCDs) and of transmitting requests from the clients to the server. PCDs will, through a tightly coupled interface, communicate with the Genesis fixed-end server via the communication service provider. PCDs are divided into three categories: a alphanumeric pager, an application specific hand-held PDA (Personal Digital Assistant), and an off-the shelf PDA.

3 Data Management in Genesis

In this section, we describe the data and queries in the Genesis system. Some of the data and queries are not included in the current stage of Genesis design, but are common in ATIS systems.

3.1 Data

ATIS information data includes driving and public-transportation data that is available to the traveler or commuter. Those data are categorized into the following.
Digital Road Maps are represented as a collection of road segments. Each road segment has a beginning node and an ending node. Since road maps are embedded in a geographical space, each node has x, y coordinates. There are about 120,000 road intersections and 300,000 road segments in the Twin Cities metropolitan area. Data set sizes are 20 Mbytes for the metropolitan area. The road-map of interest to Genesis includes all of the roads in the seven counties around the Twin-Cities area.
Transit Information includes Schedules, Route Maps, Fares, Bus Stop Schedules, Real-Time Status, Messages, Park-and-Ride Parking, Downtown HOV parking, Ridesharing, Elderly and Handicapped accommodations, and Transit Itinerary Planning. The dynamic data (parking, rideshare, bus delay, etc) is sent from the central server at about 30 Kbytes per minute for the metropolitan area.
Traffic and Roadway Information includes Incidents, Construction and Detours, Highway Segment Information, Highway Trip Information, Highway Itinerary Planning, etc. Data for for simple sensor data and incident information is sent from the central server at about 10 Kbytes per minute.
The **Business Directory** contains information about hotels, restaurants, and services, etc. Data set sizes are 50 Mbytes for administrative units of the metropolitan area.

3.2 Queries and Triggers

User queries include requests for traveler service information, pre-trip travel information, route guidance, en-route traveler advisories and emergency notification. Typical ATIS queries are shown in Table 2.

In Genesis, events such as traffic accidents, traffic congestion and road hazards need to be constantly monitored. Since events do not occur predictably, it is more appropriate to model them as triggers such that the system can warn travelers once the system detects an incident. Typical triggers are shown in Table 3. Such Genesis functions as real-time advisory and trip status can be modeled as triggers.

A real-time advisory is the dissemination of current abnormal conditions for specific road segments or for a route which is usually the primary route selected by the traveler. The type of conditions reported are: traffic incidents, traffic congestion, weather conditions and road hazards. For example, the traveler would like the system to inform him/her if there is any traffic congestion on his/her usual route home.

4 Performance Issues in Genesis

Several different strategies for supporting query processing, advisories, and communications are being considered. We list some of the strategies in this section, without any particular preference or evaluation.

Travel Time   *Find the travel time on I-35W between Hwy. 62 and Diamond Lake Road.*
Construction Query   *Which roads are under construction in the downtown area?*
Shortest Path   *Find the shortest travel-time path from the EE/CSci building to the Mall of America?*   *Display the locations of all county parks within the Twin Cities area, along with the minimum travel time routes from the EE/CSci building to them.*
Route Evaluation   *Evaluate the travel time from Home to the EE/CSci building by preferred major route I-494.*   *Evaluate the travel time for my three preferred routes from Home to the EE/CSci building.*
Emergency Service Request   *I have a flat tire, dispatch a tow-truck to I-394 and Ridgedale Drive*

Table 2: ATIS Queries

Congestion   *Whenever any traffic congestion occurs on the primary route, inform me.*
Alternative Routing   *If the travel duration of an alternate route is less than primary travel duration by a differential threshold, then suggest an alternate route as the new primary.*

Table 3: ATIS Triggers

4.1 Spatial Network Management

A spatial network is a special kind of graph with nodes located in a two-dimensional or three-dimensional Euclidean space. The spatial network database is often used to manage rapidly changing information (e.g. traffic flow on road-networks, load on utility network edges) to support interactive decision-support systems for time-critical decision-making. The database should be able to carry out network operations within a reasonable response time, preferably by minimizing response time. For example, emergency road service providers can benefit from the efficient computation of routes based on an accurate map, using current information about the state of traffic on the highway network. To be efficient, a spatial network database system needs to (1) support efficient data storage and access [12]; (2) provide algorithms for point-to-point route computation on very large graphs (10^7 nodes) [13, 14, 15]; (3) model spatial networks as continuous entities [16]; and (4) parallelize spatial queries to improve performance [17].

4.2 Strategies to Handle Triggers

As described in section 3.2, triggers are used to model events that need to be constantly monitored. We discuss alternative strategies to handle triggers below.

- Triggers are placed at the server site. Once the server monitors any trigger, it either transmits warning or advisory messages to the specific traveler or broadcasts the warning/advisory messages.
- Triggers are cached in the client (PCD) site. The server broadcasts incident data and the client caches broadcast data. The client checks the triggers for any effect on the primary route and provides warnings or

advisories if the trigger is activated.

- A hybrid of the above two methods is to put common and important triggers (e.g. severe weather conditions or earthquakes) in the server site, and to put traveler-specific triggers (e.g. If traffic congestion occurs on the primary route, then suggest alternate routes) in the client site.

4.3 Data Dissemination

In considering the periodic broadcasting of dynamic traffic information,the broadcast period (T) needs to be carefully chosen. A longer T lowers the communication load but provides less accurate (up-to-time) traffic information to the client. In addition, since ATIS often covers very wide geographic areas and since many vehicles participate, the data volume to be broadcast will be very large. We discuss the following possible approaches to reduce data volume and thus reduce the communication load.

Decompose Geographically : Travelers in the northern area might not need traffic information in the southern area. Therefore, server coverage could be geographically decomposed into different zones or cells, such that each zone has a particular server site to serve travelers in that zone. The, only traffic information in that particular zone needs to be broadcast. This scheme needs to handle the situation where travelers enter different zones or travel across nearby zones.

Checkpointing : Traffic changes are broadcasted at checkpoints. Traffic information is examined at checkpoints to determine traffic changes, i.e., when traffic difference exceeds thresholds. Increasing thresholds will reduce communication load, but at the same time decrease the accuracy of traffic information. Checkpointing operates every t times ($t \ll T$).

5 Summary

ATIS requires large amounts of data to be continuously collected, stored and disseminated. Throughput requirements and the capacity of the wireless communications channel will dictate many of the other parameters. Due to the massive number of travelers and the large geographic area to be covered, an ATIS will require high throughput demands on the wireless communications system. Channel-capacity requirements for real-time traffic information should be investigated. Efficient wireless communication architectures and technologies need to be explored to support full-scale ATIS features.

Genesis project functionality has been subsumed by the Guidestar program at the Minnesota Department of Transportation. Data dissemination is handled by Guidestar's Trilogy project [18, 3].

Communication overhead, caching strategies, data currency, scalability, computer cost and service charge, etc., need to be carefully evaluated in designing ATIS. In addition, efficient routing algorithms [15, 12, 14] should be investigated for minimum travel time or other selectable criteria that take into account current traffic conditions and traffic network travel times.

ATIS is a very promising application requiring wireless data dissemination. Since it is limited by current technology, the current stage of Genesis will only test some ATIS features. As technology advances , the functions and capabilities of Genesis will improve to accommodate advances in technology and changing user requirements. In the future, a wide variety of personal and business related ATIS services will be fully provided based on the success of mobile database technology.

Acknowledgment

Genesis represents cooperation between the public and private sector and academia [11] . Public sector institutes like the Minnesota Department of Transportation and the Federal Highway Authority are involved in the project, along with private sector companies such as IBM Federal Systems Company, BRW, JHK, Battelle, Barrientos and Associates, Motorola, etc.

References

[1] S. Shekhar and D. R. Liu. “Genesis and Advanced Traveler Information Systems (ATIS) : Killer Applications for Mobile Computing”. In *NSF MOBIDATA Workshop on Mobile and Wireless Information Systems*, 1994.

[2] “Intelligent Vehicle Highway Systems Projects”. Department of Transportation, Minnesota Document, March 1994.

[3] The Diehold Institute for Public Policy Studies Inc. *”Transportation Infrastructures: The Developement of Intelligent Transportation Systems”*. Praegeo, 1995.

[4] W. C. Collier and R. J. Weiland. “Smart Cars, Smart Highways”. *IEEE Spectrum*, pages 27–33, April 1994.

[5] J. H. Rillings and R. J. Betsold. “Advanced Driver Information Systems”. *IEEE Trans. on Vehicular Technology*, 40(1):31–40, February 1991.

[6] J. L. Buxton and et. al. “The Travelpilot: A Second-Generation Automative Navigation System”. *IEEE Trans. on Vehicular Technology*, 40(1):41–44, February 1991.

[7] R. von Tomkewitsh. “Dynamic Route Guidance and Interactive Transport Management with ALI-SCOUT”. *IEEE Trans. on Vehicular Technology*, 40(1):45–50, February 1991.

[8] A. M. Kirson. “RF Data Communications Considerations in Advanced Driver Information Systems”. *IEEE Trans. on Vehicular Technology*, 40(1):51–55, February 1991.

[9] “GENESIS : Personal Communication Device”. GENESIS 191A321 Document, 1993.

[10] S. Elliot and D. Dailey. *”Wireless Communications for Intelligent Transportation Systems”*. Artech House, 1995.

[11] James L. Wright, R. Starr, and S. Gargaro. “GENESIS - Information on the Move”. In *Proc. of Annual IVHS America Conference*, pages 334–336, 1993.

[12] S. Shekhar and D. R. Liu. “CCAM : A Connectivity-Clustered Access Method for Aggregate Queries on Transportation Networks : A Summary of Results”. In *Proc. of the 11th Intl Conference on Data Engineering*. IEEE, March 1995, (Extended version to appear in IEEE TKDE).

[13] S. Shekhar, A. Fetterer, and B. Goyal. FAST Routing for Large Spatial Networks. Technical Report 96-046, University of Minnesota, Department of Computer Science, August 1996.

[14] S. Shekhar, A. Kohli, and M. Coyle. “Path Computation Algorithms for Advanced Traveler Information System”. In *Proc. of the Ninth Intl Conference on Data Engineering*, pages 31–39. IEEE, April 1993.

[15] Shashi Shekhar and Andrew Fetterer. Path Computation in Advanced Traveler Information Systems. In *Proc. ITS America, also at http://www.cs.umn.edu/research/shashi-group/abstract/its96.abs.html*, 1996.

[16] S. Shekhar, M. Coyle, B. Goyal, D.R. Liu, and S. Sarkar. Experiences with Data Models in Geographic Information Systems. Technical Report 96R-001, University of Minnesota, Department of Computer Science. Also available at http://www.cs.umn.edu/research/shashi-group/paper_ps/cacm.new.ps, 1996. To Appear in the Communications of the ACM.

[17] S. Shekhar, S. Ravada, and et. al. “Load-Balancing in High Performance GIS: Partitioning Polygonal Maps”. In *Proc. of 4th Symposium on Spatial Databases, SSD’95. To appear in IEEE TKDE*, 1995, available via WWW, http://www.cs.umn.edu/research/shashi-group/abstract/ssd_95.abs.html.

[18] ITS America. *National ITS Program Plan*, volume 1 and 2. ITS America, Washington D.C., 1 edition, March 1995.

Efficient Dissemination of Information on the Internet

Tak W. Yan
Healtheon Corporation
87 Encina Avenue
Palo Alto, CA 94301

Hector Garcia-Molina
Department of Computer Science
Stanford University
Stanford, CA 94305

Abstract

An information dissemination server accepts long-term queries that represent user interests, collects new documents from underlying sources, matches the documents against the queries, and continuously updates the users with relevant information. In this paper we describe the SIFT (Stanford Information Filtering Tool) dissemination server, which was implemented at Stanford and is now operated commercially by InReference. SIFT disseminates USENET Netnews as well as other articles. As of April 1996, the server had more than 18,400 users world-wide, processing over 80,000 articles against some 40,100 standing queries daily. While the server was implemented and evaluated, we investigated techniques for efficient dissemination, including query processing, removal of duplicate information, and distributed information dissemination. In this paper we describe the problems addressed and some of the solutions we developed.

1 Introduction

Information dissemination (a.k.a. selective dissemination of information [Sal68], information filtering [LT92], routing) is a powerful information finding mechanism in wide-area environments. In an information dissemination system, a user submits a long-term *profile* consisting of a number of queries to represent his information needs. The system then continuously collects new documents from underlying information sources, filters them against the user profile, removes information deemed redundant to the user, and finally delivers relevant information to him.

A very simple kind of information dissemination service has long been available on the Internet: mailing lists (see e.g., [Kro92]). Hundreds of mailing lists exist, covering a wide variety of topics. The user subscribes to lists of interest to him and receives messages on the topic via email. He may also send messages to the lists to reach other subscribers. A problem with mailing lists is that they provide a crude granularity of interest matching. A user whose information need does not exactly match certain lists will either receive too many irrelevant or too few relevant messages. The USENET News (or Netnews) system (see, e.g., [Kro92]), an electronic bulletin board system on the Internet, is similar in nature to mailing lists. While Netnews is extremely successful with millions of users and megabytes of daily traffic, it is not very effective as a dissemination system. Like mailing lists, the coarse classification of topics into newsgroups means that a user subscribing to certain newsgroups may not find all articles interesting, and also he will miss relevant articles posted in newsgroups that he does not subscribe to.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

The SIFT (Stanford Information Filtering Tool) system implemented at Stanford (and as of June 1996 in commercial operation by InReference), offers a more powerful dissemination service that lets users define their interests at a much lower granularity level. The system distributes USENET Netnews and articles from various mailing lists, and accepts profiles using two standard information retrieval query languages. For example, a user may submit the profile “gateway AND Oracle” (assuming Boolean queries are used), and any article received by the server that contains these words will be forwarded to the user. The user will receive related articles from expected sources (e.g., the database interest group in netnews), and will also receive articles that may have been missed with limited systems, perhaps an article in a hospital newsgroup where a gateway to an Oracle patient database is being evaluated. It is interesting to note that traditional person to person email can also be (roughly) covered within a general dissemination model: A user John Doe wishing to receive mail addressed to him simply submits a profile requesting all articles with the words “John” and “Doe” in the “To” field. A sender simply writes the names of the recipients in the “To” field. This of course assumes that the profile language understands fields in articles, and that an access control mechanism would prevent articles from being delivered to inappropriate persons.

Before SIFT, a number of powerful dissemination services had been suggested and tested. The main focus of those efforts was on the filtering *effectiveness*, with the goal of providing fine-grained interest-matching using information retrieval (IR) [Har93, Har94, Har95, WF91], rule-based [Mal87], artificial intelligence [She94, Ste93, BS95], or relational techniques [GNOT92].

On the other hand, the main goal of SIFT was the *efficient* dissemination of large volumes of information across many profiles. Indeed, SIFT was one of the first dissemination services to be used by large numbers of people on the Internet, and performance quickly became the central issue in the running system. Some basic SIFT history illustrates this point. The system has been accessible to the public since February 1994, when we announced it in two Netnews newsgroups. Fueled by word-of-mouth publicity, traditional magazines and newsletters, as well as electronic channels such as mailing lists, the number of SIFT users has been increasing ever since, and as of April 1996, we had over 18,400 users world-wide, and more than 40,100 long-term queries. The rates of growth are approximately 680 users per month and 1,500 queries per month. At the same time, the generation rate of Netnews articles has also been increasing. Initially the system processed an average of some 30,000 articles a day, and as of April 1996, the number is over 80,000 articles a day. To handle the dissemination of an ever-increasing volume of information to an ever-growing user population, it is apparent that efficient techniques are necessary.

This paper presents an overview of issues arising in the design and implementation of a large-scale information dissemination system such as SIFT. We assume a reference architecture of an information dissemination system depicted in Figure 1. The system consists of one or more intermediate *information dissemination servers*. Such servers, well-known to both providers and users, accept queries from the users, collect documents from sources, match documents against user queries, and relay relevant information to users.

2 Indexing Queries

The first issue we address is what query processing techniques a server can use. This is critical because each server has to handle a large number of long-term queries and a very high rate of document generation. Assume that the server supports the boolean retrieval model [Sal89] for interest matching. Further assume that it has three queries, $Q_1 = a \wedge b$, $Q_2 = a \wedge c$ and $Q_3 = f$. (The notation $Q_1 = a \wedge b$ means that query Q_1 is subscribing to documents that contain both words a and b.) Say a new document D with words a, b, e has arrived. One “brute force” way to process D is to build a hash table for D that lets us quickly tell if it has a given word. We can then run through the queries and check them. For example, we check that D has a and b and so we send D to the person that posted Q_1.

The above scheme is simple, but not very efficient. For each new document, we have to evaluate *all* queries

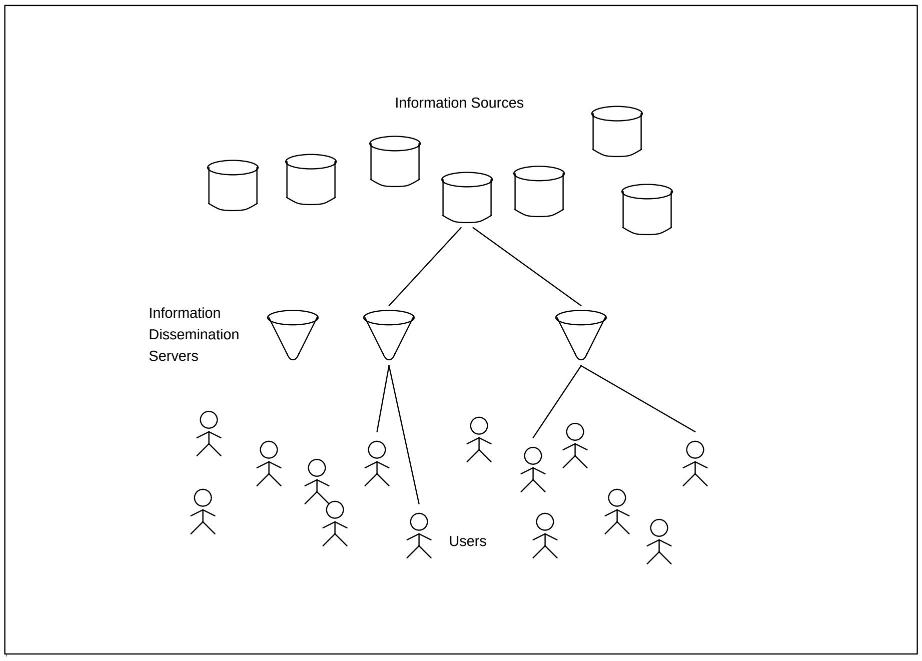

Figure 1: Reference Architecture of an Information Dissemination System

against it. To improve, we may pre-process the long-term queries; a novel idea is to build an *inverted index* [Sal89] on queries. Conventionally, inverted indexes are built on data, or documents. Here we reverse the roles of documents and queries. An inverted index on queries associates every word with a list of queries that contain it. Returning to our example, if we build an index on Q_1, Q_2, and Q_3, then the word a will have a list containing queries Q_1 and Q_2, the list of b will have query Q_1, that of c will have query Q_2, and that of f will have Q_3. When we receive a document, we check it against the index of queries. In IR, to process a query against a document index, we perform set operations on the lists of the queried words, and the result of the operations is the answer to the query. For example, the $\wedge$ (AND) operator is processed by intersecting the lists; the $\vee$ (OR) operator is processed by merging the lists [Sal89]. In processing inverted lists of queries, we cannot use set operations directly. In our example, the intersection of the lists for the words in D (i.e., a, b, and e) gives us nothing (the list of e is empty). And if we merge the lists, we get both Q_1 and Q_2, but Q_2 does not match D. However, all is not lost. The result of the union of the lists contains queries that potentially match the document: each shares at least one word with the document. The merged list is a superset of the desired answer, and so we can screen out the superfluous queries by checking them against the document, using the hash table as described above. With this strategy we avoid checking all queries, i.e., queries that do not contain a, b, or e (Q_3 in our example) will not be considered.

In [YGM94c, YGM94b], we explore the idea of indexing queries. We address questions such as: Are there better alternatives to the basic solution outlined above? How much more efficient are these indexing techniques than the brute force approach? How do we extend the idea to deal with IR techniques like stemming, truncation, and thesaurus, and to other retrieval models? The results in these papers indicate that there are several powerful indexing schemes beyond the ones described here, and that they can potentially reduce the amount of processing by orders of magnitude. Selected query indexing techniques have been implemented in the SIFT system.

3 Duplicate Removal

From our experience of operating SIFT, we realize that a very important task of an information dissemination system, besides content-matching, is *duplicate removal*; i.e., ensuring that no redundant information is delivered to the users. Due to a variety of reasons described below, duplicates are very common in digital documents. The existence of such redundant information impairs the usefulness of the system. If a document is delivered to a user, its exact/close copies will certainly reach the user over and over again. It is imperative that duplicate documents are detected and not delivered.

The foremost reason for the rise of duplicate documents is that digital documents can be reproduced with extreme ease and at almost no cost. For example, in Netnews, a user may cross-post a news article in many newsgroups. He may report it a few days later, again to multiple newsgroups. Further, there is no loss in the quality of the copies, and thus they can be replicated again. For example, a user who has made a copy of the article may report it in yet some other newsgroups or channels (e.g., mailing lists). In an experiment carried out to investigate the degree of duplication in the Netnews articles sent out by SIFT (detailed in [YGM95a]), we found that on average some 18% of articles received by a user overlap 80% or more in content (e.g., number of sentences) with some articles seen previously. We believe this is a major drawback diminishing the value of the system. Indeed, SIFT users have complained about this problem.

By a duplicate, we mean not just an exact copy as described above, but in general a document closely similar in content to some other document and not giving users any extra information. For example, in a traditional library catalog system, duplicate bibliographical records referring to the same technical reports are very common [HR79, WM79, Goy87, BH91, Rid92, ORO93]. The reasons leading to the existence of such duplicate records are mainly human input errors or inconsistencies, such as different practices in record creation, translation differences, and typographical errors. In forthcoming digital libraries, documents are not limited to short, structured bibliographical records, but rather full-text documents existing in different media types, with complex inter-document relationships among them. This rich content gives rise to more sources of duplication. One major source is the different media formats in which a document may exist. For instance, a technical report may be written in $\LaTeX$, and converted into DVI and postscript. It may also be converted into plain text or HTML. The hardcopy may be scanned in as images and then converted to text via optical character recognition (OCR). All these may be made available on-line. Another source of duplication is versioning. A document may undergo a number of versions in its life-span. For example, a technical report may have a short and a full version. A user, after reading the short version, may find the full version a duplicate.

To provide duplicate removal, we realize that individual users may have different requirements for duplicate elimination. A user, depending on what documents he has read previously, may consider a document a duplicate while another user does not. He may also want to specify how close a document must be to a seen one for it to be a duplicate. For example, a user may find an article not very interesting and want to remove any duplicate more than 70% similar in content. On the other hand, if a user receives an interesting document, he may want to remove only identical copies. Thus, the removal of duplicates operates on a per user, per document basis – each document read by a user generates a request for duplicate removal. As we can imagine, the number of such requests is tremendous in a large-scale system; maintaining and processing them undoubtedly requires efficient techniques. In [YGM95a] we discuss these issues in more depth and we present various schemes for efficiently removing duplicates in a dissemination system.

4 Distributed Information Dissemination

To really cope with the scale of information dissemination in a wide-area environment, we need a distributed system consisting of multiple servers, as in Figure 1. A key question then is how to replicate and distribute queries and documents among the servers to achieve high performance and availability. To illustrate, suppose a document is sent to every server, and a query is posted at only one server. The number of queries at each server is low, but the document arrival rate is high. Further, the availability of the service is low, since if a server goes down, the users it serves miss documents. The opposite way is to replicate a query at all servers, and send a document to any one of them. This way, availability is high – if a server goes down, documents can be rerouted to any other server. However, the number of queries is high at each server, and the cost of updating a query is high also.

There are intermediate solutions in between these two. If we denote the set of servers that a query x is posted at by P_x, and the set of servers that a document y is sent to by D_y, then to ensure that a query does not miss a document, we must guarantee that P_x intersects D_y for every x and y. This parallels the idea of *quorum consensus* in replicated data management. In [YGM94a], we formalize the correspondence and, given the options for replication and distribution, we study the tradeoff between communication costs, document delivery times, reliability, and other parameters. To validate these results, a distributed version of SIFT has been implemented.

Once a document is matched to a set of queries at a server, there is the additional problem of sending the document to the users that posted the queries. In general, there could be a very large number of users that need to receive the document. Without efficient document distribution mechanisms, much wide-area network (WAN) traffic would be generated. (The problem is related to message broadcast, e.g., [DGH$^+$87].) To illustrate, say we have a collection of users at an institution that have posted queries. A straightforward strategy for a server is to send matching documents directly to users. The disadvantage of this approach is that if a given document happens to be of interest to many users at the institution, many copies will be sent over the WAN. An improvement may be to have a *local distribution site* at the institution. Now, if a server discovers that there is a document that matches any query from that institution, only one copy needs to be sent to the distribution site, which then distributes the document to the relevant user(s) locally. This takes advantage of not only the geographical locality of the users, but also the *interest locality*: users from the same institution often share common interests, and consequently it is often that a document matches a number of users at the same time. In [YGM94a], we look at this mechanism in detail and evaluate its benefits.

5 An Overview of SIFT

Besides serving as a testbed for our experimentation, SIFT is also a publicly accessible information dissemination system. Let us briefly illustrate how SIFT works with an example (refer to Figure 2). Suppose a user is interested in underwater archaeology. He sends an email subscribe request to a SIFT server specifying the query "underwater archaeology," and optionally parameters that control, for example, how often he wants to be updated or how long his subscription is for. He may alternatively access the SIFT server via the web, using a web browser fill out a form with the subscription information. (Before he subscribes, he may test run his query against an index of a sample document collection.) His subscription is stored in the subscription database. As the SIFT server receives new documents, the filtering engine will process them against the stored subscriptions, and notifications will be sent out by email based on the user-specified parameters. A more recent version of SIFT prepares for the user a personal web page filled with pre-matched hits for his profile. This way, the user does not receive the hits by email; instead he visits his personal web page to read the matching articles.

Using SIFT, we have set up a server for selectively disseminating Netnews articles. The user accesses the Netnews SIFT server via email or the web. In February 1994, we publicized the Netnews server in two newsgroups; within ten days of the announcement, we received well over a thousand queries. The number of users and queries kept increasing ever after, and as mentioned earlier, reached 18,400 users and 40,100 queries by April 1996. Since it became increasingly expensive for an academic department to maintain such a large-scale system, in April 1996 we licensed SIFT to InReference Inc., which operates the service Reference.COM at http://www.reference.com. Since June 1996 InReference operates a modified and improved SIFT system, and the number of subscribers continues to grow. A detailed description and performance study of the original SIFT system can be found in [YGM95b].

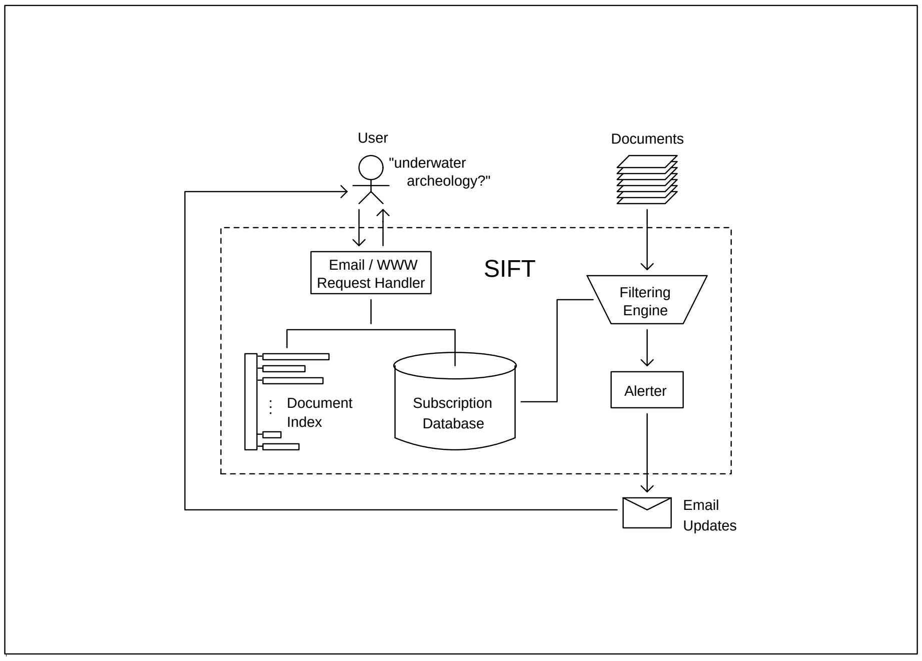

Figure 2: An Overview of SIFT

6 Conclusions

Large-scale information dissemination is becoming increasingly important. In addition to existing services available from commercial information providers such as Dialog and traditional libraries, new services and systems are emerging on the Internet, including Reference.COM, InfoSeek Personal Newswire, and Mercury NewsHound. Many sources of information are covered, such as news wires, USENET News, and mailing lists. Foreseeably, with the rapid growth in the popularity of the Internet as a source of information, these systems will become more and more popular, and the volumes of traffic will be higher and higher. We believe the efficient information dissemination techniques like the ones we have studied here will be very applicable to these systems.

References

[BH91] J. Bunge and J. Handley. Sampling to estimate the number of duplicates in a database. *Computational Statistics & Data Analysis*, 11(1):65–74, 1991.

[BS95] M. Balabanovic and Y. Shoham. Learning information retrieval agents: experiments with automated web browsing. In *Proceedings of the AAAI-95 Spring Symposium on Information Gathering from Heterogenous, Distributed Environments*, 1995.

[DGH$^+$87] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Principles of Distributed Computing*, 1987.

[GNOT92] D. Goldberg, D. Nichols, B. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communication of the ACM*, 35(12):61–70, 1992.

[Goy87] P. Goyal. Duplicate record identification in bibliographic databases. *Information Systems*, 12(3):239–42, 1987.

[Har93] D. Harmon. *Proceedings of the 1st Text Retrieval Conference*. National Institute of Standards, Maryland, 1993.

[Har94] D. Harmon. *Proceedings of the 2nd Text Retrieval Conference*. National Institute of Standards, Maryland, 1994.

[Har95] D. Harmon. *Proceedings of the 3rd Text Retrieval Conference*. National Institute of Standards, Maryland, 1995.

[HR79] T. Hickey and D. Rypka. Automatic detection of duplicate monographic records. *J. Libr. Automn*, 12(2):126–42, 1979.

[Kro92] E. Krol. *The Whole Internet User's Guide & Catalog*. O'Reilly & Associates, Sebastopol, California, 1992.

[LT92] S. Loeb and D. Terry. Editors. Special Section on Information Filtering. *Communications of the ACM*, 35(12):26–81, 1992.

[Mal87] T.W. Malone. Intelligent information sharing systems. *Communications of the ACM*, 30(5):390–402, 1987.

[ORO93] E. O'Neill, S. Rogers, and W. Oskins. Characteristics of duplicate records in OCLC's online union catalog. *Library Resources & Technical Services*, 37(1):59–71, 1993.

[Rid92] M. Ridley. An expert system for quality control and duplicate detection in bibliographic databases. *Program*, 26(1):1–18, 1992.

[Sal68] G. Salton. *Automatic Information Organization and Retrieval*. McGraw-Hill, New York, 1968.

[Sal89] G. Salton. *Automatic Text Processing*. Addison Wesley, Reading, Massachusetts, 1989.

[She94] B.D. Sheth. A learning approach to personalized information filtering. Technical Report Master Thesis, Massachusetts Institute of Technology, 1994.

[Ste93] C. Stevens. Knowledge-based assistance for handling large, poorly structured information spaces. Technical Report Ph.D. Thesis, CU-CS-640-93, University of Colorado at Boulder, 1993.

[WF91] M.F. Wyle and H.P. Frei. Retrieval algorithm effectiveness in a wide area network information filter. In *Proc. ACM SIGIR Conference*, pages 114–22, 1991.

[WM79] M. Williams and K. MacLaury. Automatic merging of monographic databases – identification of duplicate records in multiple files: the iucs scheme. *J. Libr. Automn*, 12(2):156–68, 1979.

[YGM94a] T.W. Yan and H. Garcia-Molina. Distributed selective dissemination of information. In *Proc. Parallel and Distributed Information Systems*, pages 89–98, 1994.

[YGM94b] T.W. Yan and H. Garcia-Molina. Index structures for information filtering under the vector space model. In *Proc. International Conference on Data Engineering*, pages 337–47, 1994.

[YGM94c] T.W. Yan and H. Garcia-Molina. Index structures for selective dissemination of information under the boolean model. *ACM Transactions on Database Systems*, 19(2):332–64, 1994.

[YGM95a] T. Yan and H. Garcia-Molina. Duplicate removal in information dissemination. In *Proc. VLDB Conference*, 1995.

[YGM95b] T. Yan and H. Garcia-Molina. SIFT – a tool for wide-area information dissemination. In *Proc. 1995 USENIX Technical Conference*, pages 177–86, 1995.

VLDB ‘97

Preliminary Conference Announcement

23rd International Conference on Very Large Data bases

Athens, Greece

August 25-29, 1997

PROGRAM CHAIRS

GENERAL PC CHAIR

Matthias Jarke
Informatik V, RWTH Aachen, Ahornstr. 55
52072 Aachen, Germany
email: jarke@informatik.rwth-aachen.de
Ph: (+49 241) 802 1500 FAX: (+49 241) 888 8321

EUROPE

Klaus Dittrich
Institut fur Informatik, Universitaet Zurich
Winterthurerstr. 190
CH 8057 Zurich, Switzerland
email: dittrich@ifi.unizh.ch
Ph: (+41 1) 257 4312 FAX: (+41 1) 363 0035

AMERICAS

Mike Carey
K55/801, IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120, USA
email: carey@almaden.ibm.com
Ph: (+1 408) 927 2732 FAX: (+1 408) 927 4304

MIDDLE- AND FAR-EAST

Fred Lochovsky
Department of Computer Science
Hong Kong Univ. of Science and Tech.
Clear Water Bay Rd., Kowloon, Hong Kong
email: fred@cs.ust.hk
Ph: (+852) 2358 6996 FAX: (+91) 2358 1477

INDUSTRIAL/COMMERCIAL TRACK CHAIR

Pericles Loucopoulos
UMIST, Dept. of Computing
PO Box 88, Manchester M60 1QD
United Kingdom
email: pl@sna.co.umist.ac.uk
Ph: (+44 161) 200 3332 FAX: (+44 161) 200 3364

TUTORIAL CHAIRS

Yannis Ioannidis
Department of Computer Sciences
University of Wisconsin
Madison, WI 53706, USA
email: yannis@cs.wisc.edu
Ph: (+1 608) 262-7764 FAX: (+1 608) 262 9777

Tamer Ozsu
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2H1
email: ozsu@cs.ualberta.ca
Ph: (+1 403) 492 2860 FAX: (+1 403) 492 1071

PANEL CHAIRS

Ramez Elmasri
Dept. of Computer Science and Engineering
University of Texas at Arlington
Arlington, TX 76019
email: elmasri@cse.uta.edu
Ph: (+1 817) 272 3785 FAX: (+1 817) 272 3784

Arie Segev
Walter A. Haas School of Business
University of California
Berkeley, CA 94720, USA
email: segev@csr.lbl.gov
Ph: (+1 415) 642 4731 FAX: (+1 415) 642 2826

Scope

For 23 years, VLDB has served the database community as its major truly international conference all over the world. From an arena where just research results are presented, VLDB has evolved to a forum where researchers and practitioners exchange ideas and experiences, without sacrificing quality ensured by rigorous refereeing. The VLDB '97 conference will continue and strengthen this trend.
To further the goal of stimulating new research directions, we solicit papers and panel proposals on speculative and futuristic topics. We also encourage papers on novel and challenging applications of database technology in designing and building complex information systems.

Topics of Interest

The topics of interest include but are not limited to:

Temporal and Spatial Databases	Optimization and Performance
Parallel and Distributed Databases	Persistent Object Systems
Multimedia Databases	Database Languages
Storage Management	View Management and Data Warehousing
Active Databases	Knowledge Base Management Systems
Text Databases	Engineering and Scientific Databases
Heterogeneous and Federated Databases	Graphical Query Languages and Tools
DBMS Architectures	Concurrency Control and Recovery
Data Consistency, Integrity and Security	Database Mining
Data Models and Database Design	Database Evolution and Migration
Mobile Databases	Internet Information Servers
Interaction Between DB and IR Systems	User Interfaces
Database Benchmarks	Logic and Databases
Implementation Reports/Case Studies	Transactional Workflow

Paper Submission

Six copies of original papers not exceeding 5000 words (double spaced pages) should be submitted to the appropriate program co-chair depending on the geographic region in which the authors of a paper reside. The submissions must be received by the "drop-dead" deadline of 21 February 1997; there will be no grace period. In addition to the paper hardcopy, submit a 250 words abstract with title, authors, and track (research or industrial/commercial) **one week before the official deadline** via email to the appropriate co-chair.

Industry/Commercial Track

We encourage submission of papers describing work of significant interest to industry, particularly in applications and areas providing interesting directions for the development of DBMSs. We seek to provide a forum to discuss experiences in applying DBMSs to real-life situations. Papers intended for this track will receive special consideration by the Program Committee. They should be clearly marked as industrial/commercial papers and submitted to the regular PC chairs corresponding to the author's geographic region. In the evaluation of the papers submitted to this track, the committee will consider novelty, technical quality and the value of reported results to developers and users of information systems. Questions concerning possible submissions to this track should be directed to the Track Chair.

Panels and Tutorials

Panel and tutorial proposals should be submitted to the respective chairs. Panels should address exciting new and controversial issues rather than being short paper sessions. Tutorial proposals should clearly identify the intended audience which should be broader than a small research community.

Important Dates

Paper, Panel, Tutorial, Industry/Applications submission:	**21 February 1997** (firm deadline)
Notification of acceptance:	**9 May 1997**
Conference:	**25-29 August 1997**

IEEE COMPUTER SOCIETY

This form is also available on the Web at http://www.computer.org/tab/tcapplic.htm

PLEASE RETURN FORM TO:

IEEE Computer Society
1730 Massachusetts Ave., N.W.
Washington, DC 20036-1992
Phone: +1-202-371-0101
Fax: +1-202-728-9614
k.neal@computer.org

Technical Committee & Council Membership Application

Name: ____

First (Given) Middle (Patronymic) Last (Family) Suffix

Prefix: ❐ Mr. ❐ Ms. ❐ Miss ❐ Mrs. ❐ Dr. ❐ Prof. ❐ Other

Organization: ____

Office Address/Mailstop: ____

____

City State/Province Country Postal Code

Office Phone(s): ____________________ Office Fax(es): ____________________

Email Address(es) (**Internet Accessible**): ____

Web Address(es)/WWW: ____

Home Address: ____

____

City State/Province Country Postal Code

Home Phone(s): ____________________ Home Fax(es): ____________________

IEEE Membership Number: ____________________

p I am a member of the Computer Society
Please note: Only current Computer Society members are eligible to receive Technical Committee newsletters.

PLEASE SELECT UP TO FOUR OF THE FOLLOWING: (Only the first four you select will be recorded.)

TECHNICAL COMMITTEES

p TCCX - Complexity in Computing
p TCCM - Computational Medicine
p TCCA - Computer Architecture
p TCCC - Computer Communications
p TCCE - Computer Elements
p TCCGM - Computer Generated Music
p TCCG - Computer Graphics
p TCCL - Computer Languages
p TCCP - Computer & System Packaging
p TCDE - Data Engineering
p TCDA - Design Automation
p TCDP - Distributed Processing
p TCECBS - Engineering of Computer-Based Systems
p TCFT - Fault-Tolerant Computing
p TCMS - Mass Storage Systems
p TCMF - Mathematical Foundations of Computing
p TCMM - Microprocessors & Microcomputers
p TCMARCH - Microprogramming & Microarchitecture
p TCMC - Multimedia Computing
p TCMVL - Multiple-Valued Logic
p TCOS - Operating Systems & Applications Environments
p TCPP - Parallel Processing
p TCPAMI - Pattern Analysis & Machine Intelligence
p TCRT - Real-Time Systems
p TCSP - Security & Privacy
p TCSIM - Simulation
p TCSA - Supercomputing Applications
p TCTT - Test Technology
p TCVLSI - VLSI

TECHNICAL TASK FORCES

p TFDL - Digital Libraries
p TFYF - YUFORIC: Youth Forum in CS&E

TECHNICAL COUNCILS

p TCSE - Software Engineering