The Active Information System (AIS): A Model for Developing Scalable Web Services

Sami Rollins, Robert C. Chalmers, Josep M. Blanquer, and Kevin C. Almeroth
Department of Computer Science
University of California
Santa Barbara, CA 93106-5110
{srollins, robertc, blanquer, almeroth}@cs.ucsb.edu

Abstract

The World Wide Web has become a primary venue for disseminating information to large numbers of users. From news stories, to music videos, to driving directions, more and more people are turning to the web to find information they need in their day-to-day lives. As web-based services become more complex, the traditional web model is becoming insufficient. There is an increasing demand for a model that supports large-scale, push-based data delivery combined with user-specific customization. In this paper, we describe our model, the *Active Information System* (AIS). The AIS is a model for web services that supports scalable information dissemination and dynamic user interaction. We evaluate our model by illustrating its use in developing a scalable, realtime auction application with functionality far beyond what is possible today.

1 Introduction

The World Wide Web is expanding to support a variety of services beyond static content distribution. Services such as online electronic commerce and distributed collaboration applications demand technological advances to support user interactivity, personalization, improved performance, and scalability. Moreover, the traditional pull-based method of content delivery is becoming insufficient. Information such as stock quotes changes periodically and should use *push-based* content delivery. In addition, some applications support user modification of disseminated information. An example might be a chat server where users continuously post new information. Waiting for the user to request or *pull* such information could result in missed updates if users do not poll frequently enough. The server could also be overloaded if users poll too frequently.

In this work, we focus on developing a model to implement scalable, push-based information dissemination applications. Our goal is to move away from traditional web techniques because they have a number of limitations. First, the repeated one-to-one (unicast) delivery of large information bases is wasteful of server and network resources. Second, the current web model supports only limited user-to-user interaction. Our approach attempts to overcome these limitations. At the heart of our model is an interactive, dynamic information base that allows users to interact not only with the system, but with each other. In addition, our model provides multicast-based support for large-scale data delivery to a large user base. In this way, the system can handle large numbers of requests without becoming overloaded.

Our model partially leverages the properties of the *Interactive Multimedia Jukebox* (IMJ)[1]. The IMJ is a novel architecture that addresses the issues of scalability and flexibility. The IMJ delivers on-demand programs via multicast and uses a WWW-based interface to schedule requests using a jukebox paradigm. Our goal is to generalize the IMJ model to develop a new model that achieves the IMJ's benefits of scalability and flexibility while supporting a variety of systems beyond streamed video-on-demand. We call our approach the *Active Information System* (AIS). To show the usefulness of our model, we describe its use in developing a realtime auction application. The capabilities of our AIS-based auction system far surpass functionality found in current online auctions.

This paper is organized as follows. Section 2 discusses the IMJ and its novel functionality. Section 3 describes the AIS model. Section 4 illustrates the implementation of a realtime auction. Section 5 gives an overview of other work in this area and Section 6 concludes the paper.

2 Interactive Multimedia Jukebox

The Interactive Multimedia Jukebox (IMJ)[1] defines a novel paradigm for multimedia content scheduling and dissemination. It is an Internet-based system that provides a hybrid between Video-on-Demand (VoD) and traditional broadcast television. Similar to other scalable VoD architectures[2], clients can use the system to schedule video content to be played using multicast[3]. Alternatively, users can view the schedule of programs requested by other users and choose to *tune in* to an already scheduled program.

The aim of the jukebox is to provide a scalable, flexible solution to multimedia content dissemination. The paradigm balances two metrics: *program delivery* and *program scheduling*. The number of channels where programs may be played is fixed avoiding the problem of allocating *n* sets of resources (e.g., server capacity, bandwidth) for *m*

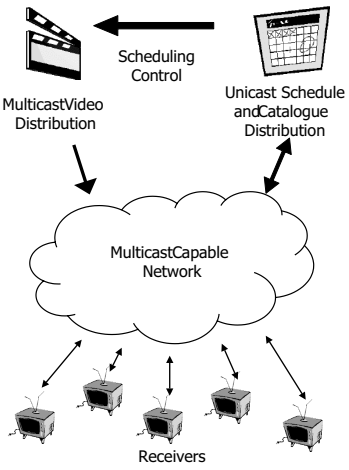

Figure 1. The IMJ architecture.

users. However, by allowing users to make requests, we achieve more flexibility than a broadcast solution. A limitless number of users may take advantage of the system at a given time. The most significant disadvantage is a longer waiting time before a user's preferred program is played. This is offset by the ability to watch any program already playing on an existing channel.

Based upon experience with the IMJ, we have identified four novel functions of the IMJ that can be generalized and extended to support a broader range of applications.

Catalogue Dissemination. As the catalogue of programs grows and as the information available about each item extends beyond a simple text description, a number of scalability issues arise. Distributing the entire catalogue each time a user requests it will place too much demand on vital resources such as network bandwidth. The challenge is to find ways to reduce the need to send the entire catalogue each time it is requested, or to satisfy multiple requests for the catalog with one transmission. Several options exist including creating a multicast stream for the catalog itself; segmenting the catalog into static parts; or only delivering the results of a user-selected query.

Request Processing/Scheduling. To provide a level of flexibility over the standard Pay-Per-View or traditional broadcast television model, users may issue requests to schedule a program of interest. The scheduler is designed with the ability to support multiple scheduling schemes: from first-come-first-served to schemes allowing users to vote on which programs they would like to have scheduled[4]. As the number of users making requests grows, the scheduling scheme may be modified to balance the tradeoff between the system resources required and the time a user must wait to watch a scheduled program.

Schedule Dissemination. Even though the catalogue of programs is a *static* information set, where the schedule of programs is dynamic, as the number of users monitoring the schedule gets large or the schedule itself grows in size, the same distribution scalability problems become an issue. Therefore, we can take advantage of similar distribution techniques. For example, only part of the schedule needs to be delivered if a user is interested in only a subset of channels. However, a difference between catalogue and schedule dissemination is that schedule dissemination requires incremental updates for all users. This implies that a push-based system is much more efficient than requiring users to frequently re-load web pages.

Video Streaming. The purpose of the IMJ is to allow users to request a service, in this case the playing of a video stream. The schedule determines when the service is to occur, and creates the event that triggers the initiation of the service. Streaming video is a costly operation that requires a great deal of resources, primarily network bandwidth and server capacity. The IMJ uses multicast delivery to solve resource bottleneck problems. By satisfying many users with one stream, good scalability can be provided. If this model can be extended beyond video to other objects, other services based on this model can benefit.

3 Active Information Systems

We define the behavior of the IMJ as that of an *Active Information System* (AIS). From an abstract point-of-view, the IMJ provides a user with the ability to dynamically request services based upon information gathered while browsing a static library of information. A request for static information generates a set of multimedia objects to be transmitted. Requests may also initiate the start of an activity. Interest in these activities, coordinated around a scheduled time, may be shared by any number of users.

There are a number of existing web-based services that follow the AIS model. For example, a large-scale chat server might first provide users with a catalogue of chat rooms. Chat rooms can be long-lived general discussions or they can be specific groups moderated by some well-known person. Users request to join a particular discussion and the activity is the exchange of messages among all members. In this case, the activity is not even centrally delivered by the AIS system.

While we recognize that large-scale, web-based applications such as chat servers have been implemented and deployed, the typical solution to problems of scale is to simply buy more hardware. Web sites such as eBay employ colossal server farms to provide constant availability to an ever increasing client base. With more hardware also comes the responsibility of developing more complex and robust software to deal with issues such as consistency between replicated databases. Our claim is that by implementing these kinds of services using the scalable AIS model, both the hardware and software requirements can be greatly reduced. Using the same infrastructure, the AIS-based application can service a larger number of users than a traditional system. Furthermore, not only can scalability be improved, but the user experience can also be significantly extended. For example most online auctions do not have realtime auctions in which bidders can “see” each other. The remainder of this section describes the framework for a generic AIS and then how it can be applied to existing web-based services.

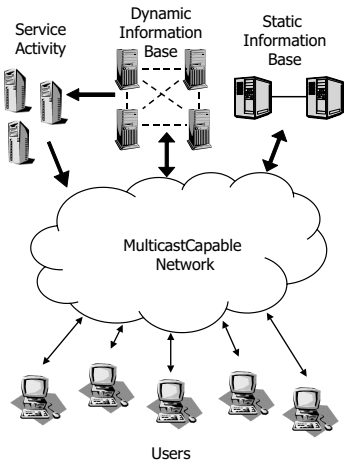

Figure 2. The architecture of a generic AIS.

3.1 The AIS Model

Section 2 identifies the four components and respective functions of the IMJ. Our challenge is to generalize the components of the IMJ and the functions they support to formalize a generic architecture for an AIS system. To support the widest set of applications, the AIS defines five components. The AIS components are analogous to the IMJ components, however the IMJ scheduler is divided into two components, one to manage static information, and one to manage dynamic information. We describe the AIS components shown in Figure 2:

1. **Scheduler**: The scheduler receives/processes requests to view or change the dynamic information base.
2. **Back-end**: The back-end receives and processes requests for static content.
3. **Event Server**: The event server executes an application-specific service activity.
4. **Proxy**: The proxy components act as application-layer multicast relay sites in the network and can provide caching, aggregation, content tailoring, etc.
5. **Client** - The client provides user interaction with the system through an application-specific interface.

The AIS also generalizes the functions of the IMJ to support a wider application base. The functions of the AIS are as follows:

Static Content Distribution. The AIS model assumes a stable back-end information source such as a database. Users begin interacting with the system by issuing requests for information from that source. In the IMJ, this was the program database. Information requests may be straightforward such as requesting all database information, or may be more sophisticated. For example, a user may request only a subset of available information, e.g. all PG rated movies. The basis of information delivery in the AIS model is the one-to-many delivery of content using multicast. This mechanism is not only applicable for the system's primary service but can also be applied for information dissemination. The challenge is find ways to aggregate the many user requests without having to constantly transmit information that is not needed and without having to make users wait while other requests are batched. Given a large enough user base, this will not be a problem. A number of multicast-based dissemination strategies are possible given a set of requests and their responses. Beyond answering queries in realtime, a low bandwidth multicast channel can be created. Users join this group when browsing a site and it is used to pre-fetch objects likely to be requested. The fundamental goal is to avoid having to dedicate system resources to satisfy a realtime user request.

Input Processing. Input processing consists of accepting and processing requests from the user. In the IMJ, these requests were to schedule playout of a particular program. Allowing the user to provide input to the application greatly increases the flexibility of the application itself. The goal of processing the request is to alter or add to the dynamic content distributed by the application. In most cases, the exact type of user requests and the processing required is application specific. Ideally, the actions of one user should positively affect other users. In the case of the IMJ, all users may watch the program scheduled by a single user. In another example, a distance learning application, the user request might be a real question, something that all students participating in the lecture would like to hear answered. These types of scenarios are surprisingly common in web applications that attempt to draw together a community of users.

Dynamic Content Distribution. The AIS model is deemed "active" because user requests create dynamic changes in the application. In the IMJ, this was the dynamic schedule of programs. AIS-style applications are dependent upon user requests. When the state of the application changes, the update must be propagated to all users. Unlike most traditional web-services, which expect users to check for new updates by re-loading, AIS uses a push-based technique. The traditional pull-based approach is limiting in a number of ways. First, realtime applications suffer arbitrary delays based on how willing the user is to frequently check for updates. And second, additional load is placed on the system because users are constantly checking for updates. A large enough user population makes this kind of system wholly unscalable. In the AIS model, updates are sent to interested users of the system as *events*. This ensures that (1) updates are delivered efficiently without wasting network bandwidth and (2) users receive updates as soon as possible. Again, using multicast to deliver these events is an effective technique for achieving scalability.

Service Activity. The service provided by an AIS application may include anything from multimedia content distribution (as in the case of the IMJ) to the sale of an item in a realtime auction. This piece of the framework is intentionally left open such that we can support a variety of applications. However, as with the IMJ, the underlying goal is still scalability. Therefore, we would like to service as

many users as possible by using a minimal allocation of resources. Again, in many applications, the use of multicast is appropriate to service all users with a single set of resources, e.g., one video stream. For some applications, like a chat room, the functionality can effectively be achieved using a completely distributed system. The real function of the AIS system is simply to coordinate users and build a community with something in a common interest.

3.2 AIS-based Web Services

To illustrate the flexibility of the AIS model, Table 1 gives a brief overview of some of the potential applications that may be implemented using the AIS.

One of the keys to the AIS model is the use of push-based communication. It makes the AIS particularly useful for the web environment where a pull-based model can be limiting, especially for realtime applications. For example, a service such as an eBay-style auction cannot take place in realtime. To see bid updates, users must visit the eBay web site and actively request a bid update. This limitation exists with most web-based services. Users who poll too frequently place unnecessary demands on the servers while users who do not poll frequently enough may miss updates. However, the use of a push-based model would eliminate many of these problems.

As the web evolves, a scalable, push-based model for web services will become increasingly necessary. The AIS model solves many of the problems of flexibility and scale faced by many existing web applications. In addition, as the services offered on the web become more diverse, we can use our model to develop those services bottom up. Our goal is to develop a basic framework that an application programmer can use to develop an AIS-style application.

4 Case Study: AIS for Real-time Auctions

We now attempt to evaluate the merits and drawbacks of the AIS model by describing a design for a realtime auction using concepts in the model. We first look at the design of the application and then evaluate how well the functionality decomposes into AIS functions.

4.1 Designing an Online Auction

Most online auctions follow the *open outcry* or *English* model. In this type of auction, participants submit bids for a given item. The other bidders see the bids submitted and may choose to submit a higher bid for the given item. In online auctions, there is no realtime delivery of new bids. Therefore, current systems must span days or even weeks.

Wellman and Wurman indicate that there have been no successful attempts to conduct **realtime** open outcry auctions online[5]. There are a number of challenges associated with developing a system that would provide the appropriate facilities for a realtime auction. First, it is imperative that the system be push-based. Users must be notified of a bid as soon as it occurs. Waiting for a user to visit a web page to check bids may result in users missing updates

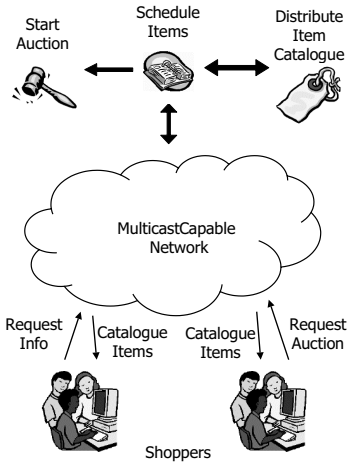

Figure 3. The first AIS distributes the catalogue of items and manages the schedule.

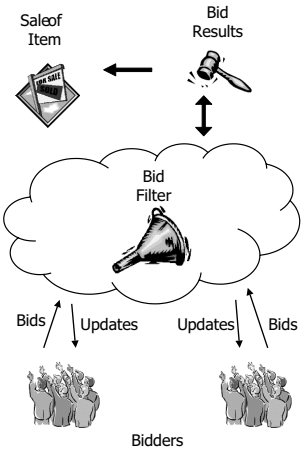

Figure 4. The second AIS conducts the auction.

and thus missing a bidding opportunity. Moreover, scaling an auction that uses this kind of system will be nearly impossible, especially if many items are being sold simultaneously.

Figures 3 and 4 illustrate our approach to developing an online auction. The implementation we describe requires two instantiations of the AIS. The first instantiation distributes a catalogue of items and creates the order for items to be auctioned. When an auction is scheduled to begin, the first instantiation of the architecture triggers the start of the second. The second instantiation conducts the auction by processing bids and distributing them to all interested auction participants. The following section evaluates the benefit of using the AIS for an auction application.

4.2 Evaluation of AIS

The novelty of the AIS model is that it provides an inherent scalability mechanism for web-based applications. It is also flexible enough to support a variety of applications. We now look at how well AIS supports the auction application, scalability of static content distribution, and the benefit of static allocation of resources.

By designing an auction using the AIS model, we

Application	Static Content	Input Processing	Dynamic Content	Service Activity
Chat World	Catalogue of chat rooms	Post a message	Messages posted by users	n/a
Digital Classroom	Lecture notes, web tours, and archived content	Lecture replay requests, real-time questions, requests for supporting materials	Student questions, class handouts, real-time test questions.	Creating additional streams of classroom streams of content
News-on-Demand	News categories	Subscription to a service	Schedule of stories	Cotent delivery
Stock Ticker	List of stocks and related news	Subscription to items of interest	Order and frequency of updates	Contet delivery
Realtime Auction	Catalogue	Start of auction	Auction schedule	Running an auction

Table 1. Examples of AIS-style applications.

have discovered which parts of the model support the application. First, the push-based nature of the AIS model is key to implementing a realtime auction. Without a push-based information dissemination model, the realtime constraint cannot be met and an auction application ends up more in the vein of eBay style auctions. Second, the modular design of the model provides a level of flexibility. If the application were to change to include a vote-based auction schedule scheme, the design of the auction would require only minimal changes.

Beyond flexibility, another advantage of the AIS model is scalable static content delivery. Web sites such as eBay generally employ one-to-one delivery of catalogue information. Each user must generate a separate request for a desired item and the item itself is sent once for every user. This is not at all scalable. First, the server is loaded by processing many queries, possibly the same query multiple times. This is especially expensive if the user has issued a complex query. In addition, network bandwidth becomes a bottleneck, especially if the reply is large. As the database of items grows, queries become more complex and results grow as well. This could easily be the case with an auction database. Sellers may post videos or three dimensional representations of the items they wish to auction. Therefore, even if a query returns only a few results, the response object may still be very large.

In the auction application, scalability is achieved by having the result set from every query multicast to the entire user population. This means that a single query has the potential to satisfy a number of users yielding a substantial gain over the traditional unicast model. When a user issues a query, the client application first checks the local cache. If the request can be satisfied by the local cache, the result is displayed. The tradeoff in this case is the number of results a user must cache versus the number of results in which the user is interested. We make the assumption that there will be a substantial overlap in user interest. There will be a subset of *hot* items that many users are interested in and the remainder of the items will be *cold*[6]. However, the best gain occurs when multiple users are interested in the exact same result. In addition, performance could be improved by developing a set of cache heuristics. Profiles, which may be user-defined or discovered over time, indicate which results are most likely to be accessed.

5 Related Work

Our research draws from two main areas. In this section, we first look at research efforts that have looked at creating scalable web services and where they fall short. We follow with a discussion of information dissemination architectures and evaluate the functionality provided by current architectures. The AIS seeks to unite these two concepts by providing a scalable Internet service model that supports large scale data dissemination.

5.1 Internet Services

Fox and Brewer motivate the need for a scalable solution to providing Internet services with 24x7 availability to an ever growing user base[7]. A number of projects have approached this problem. TACC[8] and MultiSpace[9] both look at designing a cluster-based solution to provide incremental scalability. The focus of both projects is largely the design of a protocol for distributing workload among cluster nodes. In contrast, our effort focuses on developing a model for application development that will provide inherent scalability at all levels, from the hardware infrastructure to the network. In fact, both TACC[8] and MultiSpace[9] can be thought of as complimentary to our effort. We could employ their cluster-based techniques to implement our schedulers, event servers, and back-end system.

Smart Clients[10], part of the WebOS project[11], takes a slightly different approach by looking at how to use client side Java applets to load balance and provide fault tolerance. The aim is to distribute the tasks providing greater flexibility while maintaining the transparency of a server side solution. Our model advocates a similar idea and can leverage many of the techniques developed in this work to join clients to the proxy tree in a distributed, scalable manner. The benefit of the AIS model is that it provides a more complete solution, developing a model for both client and server scalability. WebOS provides a larger view by developing a file system and security model. However, the focus is still not on large scale data dissemination. The goal of the AIS is not only to provide the foundation for developing web services, but to overcome the limitations of the traditional unicast, pull based web model when applications become data intensive.

5.2 Data Dissemination

Large scale data distribution has been studied to some extent. However, many related projects focus largely on a single application. For example, SIFT[12] and Tapestry[13] both focus largely on email filtering and dissemination. Salamander develops an information dissemination architecture that looks largely at delivery semantics such as Quality of Service parameters[14]. The model itself does not directly address issues of scale in the information delivery service. Finally, while these models provide the user with the ability to *subscribe* to the service itself, they do not develop a notion of user *request* to dynamically alter the state of the system. Alternatively, the AIS model provides a scalable, dynamic model that allows users to alter the state of the system in real-time.

A number of projects have looked at broadcast of large information bases[15, 16]. The Broadcast Disks project[17] looks at disseminating data by cyclically broadcasting objects to clients. A similar project looked at cyclic, multicast delivery of web pages to clients[18]. Clients listen to a broadcast channel until they receive all sub-pieces of the information they are interested in. An extension to the Broadcast Disks project investigates the balance between push versus pull data delivery[19] allowing clients to explicitly request information rather than always waiting until the server schedules the item to be sent. Finally, DBIS[20] summarizes this work by providing a classification of information storage and dissemination systems. While this work focuses largely on delivery of static web pages, we focus on static as well as dynamic content. Not only may the content from external sources change, the users themselves can modify the content that is delivered. Our model goes one step further to support data reads and writes by the users rather than simply reads.

6 Concluding Remarks

This paper describes the development of the Active Information System, a model for data intensive, dynamic, web-based applications. By generalizing the concepts developed in the Interactive Multimedia Jukebox, we have created a model suitable for building a variety of Internet-based information dissemination applications. To illustrate the feasibility of using our model, we have looked at its advantages and disadvantages by describing the design of a real-time auction. We are encouraged by the model's support of the auction application. We anticipate the use of AIS for a variety of web services. The AIS provides a model for scalable, interactive applications. Implementation of new applications and services should be straightforward given that AIS-based systems are based on underlying communication mechanisms and basic functionality that are simple in the abstract sense.

References

[1] K. Almeroth and M. Ammar, "The interactive multimedia jukebox (IMJ): A new paradigm for the on-demand delivery of audio/video," in *Proceeding of the Seventh International World Wide Web Conference*, (Brisbane, AUSTRALIA), Apr. 1998.

[2] T. Little and D. Venkatesh, "Prospects for interactive video-on-demand," *IEEE Multimedia*, pp. 14–23, 1994.

[3] S. Deering and D. Cheriton, "Multicast routing in datagram internetworks and extended lans," *ACM Transactions on Computer Systems*, vol. 8, pp. 85–111, May 1990.

[4] A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling policies for an on-demand video server with batching," in *Proceedings of ACM Multimedia '94*, (San Francisco, CA, USA), Oct. 1994.

[5] M. Wellman and P. Wurman, "Real time issues for internet auctions," in *Proceedings of the IEEE Workshop on Real-Time E-Commerce Systems*, (Denver, CO, USA), June 1998.

[6] G. Zipf, *Human Behavior and the Principle of Least Effort*. Reading, MA: Addison-Wesley, 1949.

[7] A. Fox and E. Brewer, "Harvest, yield, and scalable tolerant systems," in *Proceedings of the Seventh Workshop on Hot Topics in Operating Systems*, (Rio Rico, AZ, USA), Mar. 1999.

[8] A. Fox, S. Gribble, Y. Chawathe, E. Brewer, and P. Gauthier, "Cluster-based scalable network services," *Operating Systems Review*, vol. 31, pp. 78–91, Dec. 1997.

[9] S. Gribble, M. Welsh, E. Brewer, and D. Culler, "The multispace: an evolutionary platform for infrastructural services," in *Proceedings of the USENIX Annual Technical Conference 1999*, (Monterey, CA, USA), June 1999.

[10] C. Yoshikawa, B. Chun, P. Eastham, A. Vahdat, T. Anderson, and D. Culler, "Using smart clients to build scalable services," in *Proceedings of the USENIX Annual Technical Conference 1997*, (Anaheim, California, USA), Jan. 1997.

[11] A. Vahdat, T. Anderson, M. Dahlin, and D. Culler, "Webos: Operating system services for wide area applications," in *Proceedings of the Seventh IEEE Symposium on High Performance Distributed Computing*, (Chicago, Illinois, USA), July 1999.

[12] T. Yan and H. Garcia-Molina, "Sift - a tool for wide-area inforamtion dissemination," in *Proccedings of the 1995 USENIX Technical Conference*, (New Orleans, LA, USA), Jan. 1995.

[13] D. Goldberg, D. Nichols, B. Oki, and D. Terry, "Using collaborative fi lerting to weave and information tapestry," *Communications of the ACM*, vol. 35, pp. 61–70, Dec. 1992.

[14] G. Malan, F. Jahanian, and S. Subramanian, "Salamander: A push-based distribution substrate for internet applications," in *Proceedings of USITS 1997*, (Monterey, CA, USA), Dec. 1997.

[15] S. Jiang and N. Vaidya, "Scheduling data broadcast to impatient users," in *Proceedings of the ACM international workshop on Data engineering for wireless and mobile access*, (Seattle, WA, USA), Aug. 1999.

[16] K. Stathatos, N. Roussopoulos, and J. Baras, "Adaptive data broadcast in hybrid networks," in *Proceedings of the 23rd VLDB Conference*, (Athens, Greece), 1997.

[17] S. Acharya, M. Franklin, and S. Zdonik, "Dissemination-based data delivery using broadcast disks," *IEEE Personal Communications*, vol. 2, Dec. 1995.

[18] K. Almeroth, M. Ammar, and Z. Fei, "Scalable delivery of web pages using cyclic best-effort (udp) multicast," in *Proceedings of IEEE INFOCOM 1998*, (San Francisco, CA, USA), June 1998.

[19] S. Acharya, M. Franklin, and Z. S., "Balancing push and pull for data broadcast," in *Proceedings of the ACM SIGMOD Intl. Conference on Management of Data 1997*, (Tucson, AZ, USA), May 1997.

[20] D. Aksoy, M. Altinel, R. Bose, U. Centemel, M. Franklin, J. Wang, and S. Zdonik, "Research in data broadcast and dissemination," in *Proceedings of 1st International Conference on Advanced Multimedia Content Processing*, (Osaka, Japan), Nov. 1998.