

Middleware Support for Multicast-based Data Dissemination: A Working Reality

— A white paper —

Panos K. Chrysanthis*

Vincenzo Liberatore[†]

Kirk Pruhs[‡]

Abstract

Multicasting is an effective method to guarantee scalability of data transfer. Multicast applications range from the support of Content Delivery Networks to the relief of Internet hot spots. Much research has focused on isolated data management issues that arise in a multicast environment, including our own work on caching, scheduling, indexing, hybrid schemes, and consistency maintenance. Our goal is to investigate the integration of these research contributions and transfer them into a working software distribution that provides the middleware support of a data management layer to applications. Our architecture is flexible, can be shared across applications, and operates on top of existing and upcoming implementations of multicast protocols. The middleware will benefit distributed applications with a uniform, efficient, scalable, and state-of-the-art support for critical data management functionality. Furthermore, we will operate a distributed testbed for the evaluation of the middleware and applications.

Motivation

Any casual user of the Web knows that there is a critical need for scalable data transfer methods. Web servers and networks can be brought to their knees by hot-spot access patterns. Examples of Internet hot spots include news sites such as CNN after the 2000 presidential elections, the NASA site after the Mars Pathfinder landing, and the host site at the Nagano Olympics. In each case millions of clients attempted to simultaneously access the sites. Servers and networks were overwhelmed by the large number of requests, resulting in long delays to retrieve documents. One solution to this scalability problem is *multicast*. In these hot spot examples, many clients were interested in nearly the same data. Thus, a single server could have served all such requests by multicasting the hot data items to interested clients without any explicit request from the clients.

Our goal is to integrate data dissemination and multicast communication techniques into a working software distribution that provides the middleware support of a scalable multicast-based data management layer to applications. The time is ripe to exploit new and effective Internet multicast solutions to relieve the scalability problems of data-intensive distributed applications. Our approach will also impact data dissemination in wired and wireless local area networks, as well as satellite links, where broadcasting is the principal mode of communication.

*Dept. of Computer Science, University of Pittsburgh, Pittsburgh, PA 15260. E-mail: panos@cs.pitt.edu

[†]Electrical Engineering and Computer Science Department, Case Western Reserve University, 10900 Euclid Avenue, Cleveland, Ohio 44106-7071. E-mail: vx111@po.cwru.edu. URL: <http://vorlon.cwru.edu/~vx111/>.

[‡]Dept. of Computer Science, University of Pittsburgh, Pittsburgh, PA 15260. E-mail: kirk@cs.pitt.edu

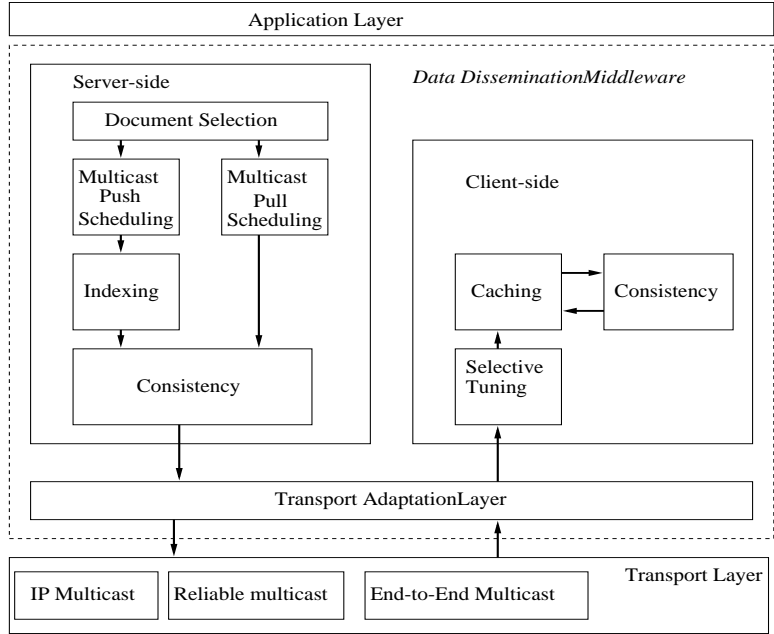


Figure 1: The middleware data dissemination architecture, and its relationship with the application and transport layers.

Data Management Issues in Multicasting

Multicast communication raises many data management issues that either do not arise in unicast communication, or that obviously require different solutions than the standard methods used in unicast settings. As an illustrative example consider the application of a highly scalable Web server. The objective of the Web server application is to scale to a large client population, and scalability will be accomplished by using the middleware. In the middleware, the server can disseminate data by choosing any combination of the following three schemes: *multicast push*, *multicast pull*, and *unicast push*. In multicast push the server repeatedly sends information to the clients without explicit client requests. (For example, television is a classic multicast push system). Multicast push is an ideal fit for asymmetric communication links, such as satellites and base station methods, where there is little or no bandwidth from the client to the server. For the same reason, multicast push is also ideal to achieve maximal scalability of Internet hot spots. Hence, generally multicast push should be restricted to hot resources. In multicast pull, the clients make explicit requests for resources, and the server broadcasts the responses to all members of the multicast group. If multiple clients request the same resource at approximately the same time, the server may aggregate these requests, and only broadcast the resource once. One would expect that this possibility of aggregation would improve user perceived performance for the same reason that proxy caches improve performance, that is, it is common for different users to make requests to the same resource. Multicast pull is a good fit for “warm resources” for which repetitive multicast push cannot be justified, while there is an advantage in aggregating concurrent client requests. Traditional unicast pull is reserved for cold documents. The end-user should not perceive that Web resources are downloaded with a variety of methods, as the browser and the middleware shield the user from the details of the multi-tier dissemination protocol.

In the Web server application, the *document selection* unit periodically gathers statistics on document popularity. Once statistics have been collected, the server partitions the resources into *hot*, *warm*, and *cold* documents. When a client wishes to request a Web document, it either downloads it from a multicast channel or it requests the document explicitly depending on whether the document is hot or not. The server also broadcasts an *index* of sorted URIs which quickly allows the client to determine whether the requested resource is in the hot broadcast set. On the whole, the client determines the multicast channel, downloads the appropriate portions of the index, and determines whether the resource is upcoming along the cyclic broadcast. If the request is not in the hot broadcast set, the client can make an explicit request to the server, and simultaneously starts to listen to the warm multicast channel if one is available. If the page is cold, the requested resource is returned on the same connection. If the page is warm, the client waits on the warm multicast channel until the requested resource is transmitted. The *multicast pull scheduling* component resolves contention among client request for the use of the warm multicast channel and establishes the order in which pages are sent over that channel.

In multicast push, the server periodically broadcasts hot resources to the clients. The server chunks hot resources into nearly equal-size pages that fit into one datagram and then cyclically sends them on a single or on a layered multicast channel along with index pages. The frequency and ordering of the pages within the multicast push channel are determined by the *multicast push scheduling* component. Upon receipt of the desired pages, the client can buffer them to reconstruct the original resource and can *cache* resources to satisfy future request. The set of hot pages is cyclically multicast, and so received pages are current in that they cannot be more than one cycle out-of-date. Furthermore, certain types of *consistency* semantics can be guaranteed by transmitting additional information along with the control pages.

Our Approach: Middleware for Broadcast Data Dissemination

This project incorporates both applied and fundamental research investigations. On the applied research side, it focuses on the design and implementation of middleware for multicast data dissemination that transparently provides applications with a flexible array of data management services. These services include document selection, scheduling, indexing, caching, and consistency maintenance. The outline of our architecture is shown in Figure 1 (the transport layer is any one of the protocols that is available independently of this project, and the objective of the transport adaptation layer is to enable the middleware to interact with different types of multicast transport within a uniform interface). This approach has the benefit of unifying several algorithms and techniques from data management as services into one software distribution. As a result, established and new methods from data management will be more generally available to application developers.

On the fundamental research side, beside new algorithms and optimizations for each component, this project aims to provide an insight into the functional synergy of the different components. An integrated approach highlights gaps in the state of the art, and leads to new research problems and solutions. In particular, this middleware exposes the performance and functional trade-off between existing data management algorithms and existing or proposed multicast protocols. Consequently, our approach suggests new research issues in the field of middleware technology.

Conclusion

In summary, our collaborative project between the University of Pittsburgh and Case Western Reserve University will explore fundamental questions and the associated middleware development issues to support multicast-based data dissemination. The project consists of two parts:

- Applied research to prototype and experiment with a new middleware layer that provides data management solutions in a multicast environment. We would implement this prototype on top of a generic multicast transport API that will allow us to support and experiment with diverse multicast protocols on real and simulated IP networks.
- Fundamental research on the development of new data management techniques and on the interaction between data management components, and between the data management middleware and the underlying multicast protocol.

References

- [1] R. Agrawal and P. K. Chrysanthis. Efficient data dissemination to mobile clients in e-commerce applications. In *Proc. of the Third IEEE Int'l Workshop on Electronic Commerce and Web-based Information Systems*, June 2001.
- [2] J. Emonds and K. Pruhs. Broadcast scheduling: When fairness is fine. In *Proc. of the ACM/SIAM Symposium on Discrete Algorithms Experiments (SODA)*, 2002.
- [3] B. Kalyanasundaram, K. Pruhs, and M. Velauthapillai. Scheduling broadcasts in wireless networks. *Journal of Scheduling*, 2002.
- [4] S. Khanna and V. Liberatore. On broadcast disk paging. *SIAM Journal on Computing*, 29(5):1683–1702, 2000.
- [5] V. Liberatore. Caching and scheduling for broadcast disk systems. In *Proc. of the 2nd Workshop on Algorithm Engineering and Experiments (ALENEX 00)*, pages 15–28, 2000.
- [6] V. Liberatore. Broadcast scheduling for set requests. In *DIMACS Workshop on Resource Management and Scheduling in Next Generation Networks*, 2001.
- [7] E. Pitoura and P. K. Chrysanthis. Exploiting versions for handling updates in broadcast disks. In *Proc. of the 25th Int'l Conference on Very Large Data Bases*, pages 114–125, Sept. 1999.
- [8] E. Pitoura and P. K. Chrysanthis. Scalable processing of read-only transactions in broadcast push. In *Proc. of the 19th IEEE Int'l Conference on Distributed Computing Systems*, pages 432–441, June 1999.