"Data In Your Face": Push Technology in Perspective*

Michael Franklin
University of Maryland
franklin@cs.umd.edu

Stan Zdonik
Brown University
sbz@cs.brown.edu

Abstract

Push technology has recently generated a tremendous amount of media attention, commercial activity, and controversy. The wide range of opinions on push is understandable given that it represents a major departure from the way distributed information systems have traditionally been built. Adding to the noise, however, is confusion about the basic principles of push and where it fits in to the world of data delivery. For example, many discussions on the topic blur the distinction between push and broadcast. We argue that this confusion stems from two fundamental causes: First, push is just one dimension of a larger design space of data delivery mechanisms. Second, networked information systems can employ different data delivery options between different sets of information producers and consumers. In this short paper we characterize the design space for dissemination-based information systems and applications, and show how current "push" solutions fit into this space. We then use this framework highlight how the implementation of current Internet-based push solutions differs from the appearance that they present to users.

1 Introduction

Push technology stems from a very simple idea. Rather than requiring users to explicitly request (i.e., "pull") the information that they need, data can be sent to users without having them specifically ask for it. The advantages of push are straightforward. The traditional pull approach requires that users know *a priori* where and when to look for data or that they spend an inordinate amount of time polling known sites for updates and/or hunting on the network for relevant sites. Push relieves the user of these burdens. The problems of push are also fairly obvious. Push transfers control from the users to the data providers, raising the potential that users receive irrelevant data while not receiving the information they need. These potential problems can arise due to issues ranging from poor prediction of user interests to outright abuse of the mechanism, such as "spamming". The "in-your-face" nature of push technology is the root of both its potential benefits and disadvantages.

Push technology has been around in various forms for as long as people have been communicating. Examples range from newspapers, to telephones, to radio and television, to E-mail. Early work on using computer networks for pushing data was performed in the 1980's. The Boston Community Information System at MIT [Giff90], Teletext systems for distributing data over broadcast media [Amma85, Wong88], and the Datacycle database machine [Herm87], are all examples of systems that incorporated some form of push technology. Recently, however, the combination of push technology with the Internet and Web (sometimes referred to as *Webcasting*) has generated a ground swell of excitement, commercial activity, and controversy.

1.1 The Push Phenomenon

In February 1996, PointCast made its client software available for free downloading over the Internet, setting off a wave of interest in push technology. The idea was appealing: rather than using your idle desktop machine as a display ground for flying toasters, PointCast would turn it into an active information terminal that would display headlines, weather forecasts, stock prices, sports scores, etc., with the appearance of having real-time updates. By specifying a *profile*, users could indicate their interests to the system, and the display would be tailored to these interests.

For anyone who tried the software, the reaction was immediate; this represented a paradigm shift in the way one could think about using the Internet as an information delivery tool. Push technology on the Internet represented a new and untapped medium. The computer trade press became in-

*This work has been partially supported by Rome Labs agreement number F30602-97-2-0241 under DARPA order number F078, by the NSF under grant IRI-9501353, and by research grants from Intel and NEC.

To appear in the ACM SIGMOD International Conference on the Management of Data, Seattle, WA, June, 1998.

undated with articles about push technology and dozens of companies touting push-based solutions arrived on the scene. A new jargon of data delivery was developed, with terminology borrowed from broadcast media. Users of push technology could *tune* into *channels* that contained *broadcasts* of information on particular topics.

By the end of 1996, the excitement had spilled over into the mainstream press. A steady stream of articles about push technology appeared in venues such as the *New York Times* and the *Wall Street Journal*.[1] In February 1997, *Business Week* magazine published a Special Report section entitled "A Way Out of the Web Maze", which argued that Webcasting could solve many of the Web's problems, such as information overload and the inability for users to find the data they need. Similar sentiments were echoed by numerous vendors and technology pundits.

The peak of the media hype for push technology was reached in March of 1997 when the cover article of *Wired* magazine blared: "Push! Kiss your browser goodbye". This article began by declaring: "Remember the browser war between Netscape and Microsoft? Well forget it. The Web browser itself is about to croak. And good riddance.". While the article was certainly provocative and clearly overstated, the argument it made was simply that push technology would change the Web from a passive library of information into a networked, immersive medium for information and entertainment delivery. Despite this simple message, the article seemed to epitomize both the promise of push technology and the potential for overselling its virtues.

1.2 The Inevitable Backlash

Around the time of the *Wired* article, the voices of dissent began to make themselves heard. A March 1997 *New York Times* CyberTimes article by James Gleick stated: "... the promotion of Push is the silliest piece of puffery to waft along in several seasons. ... The failure of Push is preordained.". A July 1997 article in the on-line net-zine *webmonkey* (published by the same company that publishes *Wired*), was entitled simply "Why Channels Suck". A somewhat more technical article at the CNET on-line site entitled "Networks Strained By Push", described a study indicating that push technologies were using an inordinate portion of corporate network bandwidth. Finally, a *Byte* magazine article in August 1997 had the tag line: "Web push technology is exploding — even though there's no such thing.". The *Byte* article went on to explain (correctly) that current push technology is "really pull++".

[1] Many of these articles had titles such as "When Push Comes to Shove", "The Pull of Push", or "X Gets Pushy" (where X is some product or company). The observant reader will notice that we have resisted such temptations for this paper.

1.3 The Current Situation

Recently, the media turmoil over push has settled down and expectations for the technology (at least for the short term) have lowered to arguably more reasonable levels. Still, the commercial activity in the area is impressive. As of January 1998, a register of push technology vendors listed 49 companies with announced products (see David Strom's site at http://www.strom.com/imc/t4a.html). Many other companies who have not yet announced products are working on push-based solutions. The major web browser vendors, Netscape and Microsoft, have both incorporated push into their products.

A development indicating a degree of maturation of the field is Microsoft's proposal of the Channel Definition Format (CDF) standard to the World Wide Web Consortium (W3C). CDF is a language that web publishers can use to turn their content into "Channels" that can be exploited by push (or "pull++") technologies. CDF allows the specification of metadata about a website, including a searchable title and abstract and information about the structure and update schedule of the site. A number of the major push vendors such as PointCast, BackWeb, and AirMedia have expressed support for the proposed standard. Such a standard raises the potential for push technology to be more widely integrated into the fabric of the Internet.

1.4 Sorting it All Out

The wide range of opinions on the pros and cons of push technology is understandable, given the fact that it is a major departure from the way distributed information systems have traditionally been built. Adding to the noise, however, is a wide-spread confusion about the basic principles of push and where it fits in to the world of data delivery. In this short paper we argue that this confusion stems from two fundamental causes: First, *push is just one dimension of a larger design space of data delivery mechanisms*. We identify three dimensions for data delivery mechanisms (push vs. pull is one of them) and show how different choices along these dimensions interact. Second, *networked information systems can employ different data delivery options between different sets of information producers and consumers*. Thus, complex systems will likely contain mixtures of push and pull (along with the other options) at various points in the network. In such a situation, it is inappropriate to identify an entire system as being "push-based" or "pull-based".

In the following, we present an overview of our ideas on data dissemination in order to provide a framework for thinking about push technology in the larger context of networked information systems. Our intent is to clarify some of the issues surrounding push technology and to characterize the design space for data delivery in dissemination-based information systems and applications.

2 Fundamental Properties

In this section, we present an overview of data delivery, focusing on how the notion of data push fits in with the other dimensions of the design space for delivery mechanisms. We then describe why it is often inappropriate to refer to complex distributed systems as simply "push-based" or "pull-based". A more detailed discussion of these issues can be found in [Fran97].

2.1 Options for Data Delivery

Support for different styles of data delivery allows a distributed information system to be optimized for various server, client, network, data, and application properties. We have identified three main characteristics that can be used to compare data delivery mechanisms: (1) push vs. pull; (2) periodic vs. aperiodic; and (3) unicast vs. 1-to-N. While there are numerous other dimensions that should be considered, such as fault-tolerance, ordering guarantees, error properties, network topology, etc., we have found that these three characteristics provide a good initial basis for discussing many popular approaches. In particular, we argue that all three of these characteristics must be considered in order to make intelligent choices about delivery mechanisms for specific situations. Figure 1 shows these characteristics and how several common mechanisms relate to them.

2.1.1 Client Pull vs. Server Push

We first focus on push vs. pull. 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, as discussed in the introduction, push-based data delivery involves sending information to a client population in advance of any specific request. With push-based delivery, the server initiates the transfer.

2.1.2 Aperiodic vs. Periodic

Both push and pull can be performed in either an aperiodic or periodic fashion. Aperiodic delivery is *event-driven* — a data request (for pull) or transmission (for push) is triggered by an event such as a user action (for pull) or data update (for push). In contrast, periodic delivery is performed according to some pre-arranged schedule. This schedule may be fixed, or may be generated with some degree of randomness.[2] 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.

2.1.3 Unicast vs. 1-to-N

The third characteristic of data delivery mechanisms is whether they are based on unicast or 1-to-N communication. With unicast communication, data items are sent from a data source (e.g., a single server) to one other machine, while 1-to-N communication allows multiple machines to receive the data sent by a data source.[3]

Two types of 1-to-N data delivery can be distinguished: multicast and broadcast. With multicast, data is sent to a specific subset of clients who have indicated their interest in receiving the data. Since the recipients are known, given a two-way communications medium 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 possibly unbounded set of clients can listen.

2.2 Classification of Delivery Mechanisms

It is possible to classify many existing data delivery mechanisms using the characteristics described above. Such a classification is shown in Figure 1. We discuss several of the mechanisms below.

Aperiodic Pull - Traditional request/response mechanisms use aperiodic pull over a unicast connection. If instead, a 1-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 (e.g, see [Acha97, Akso98]).

Periodic Pull - In some applications, such as remote sensing, a system may periodically send requests to other sites to obtain status information or to detect changed values. If the information is returned over a 1-to-N link, then as with request/response, other clients can snoop to obtain data items as they go by. Most existing Web or Internet-based "push" systems are actually implemented using Periodic Pull between the client machines and the data source(s).

Aperiodic Push - Publish/subscribe protocols are becoming a popular way to disseminate information in a network [Oki93, Yan95, Glan96]. In a publish/subscribe system, users provide information (sometimes in the form of a

[2] For the purposes of this discussion, we do not distinguish between fixed and randomized schedules. Such a distinction is important in certain applications. For example, algorithms for conserving energy in mobile environments proposed by Imielinski et al. [Imie94] depend on a strict schedule to allow mobile clients to "doze" during periods when no data of interest to them will be broadcast.

[3] Some systems attempt to implement a 1-to-N style of data delivery using unicast (i.e., by sending identical, individual messages to multiple clients). As discussed in Section 3, this type of pseudo-broadcast can result in tremendous bandwidth and server overload problems. For this reason, we classify such systems as "unicast-based" in our taxonomy.

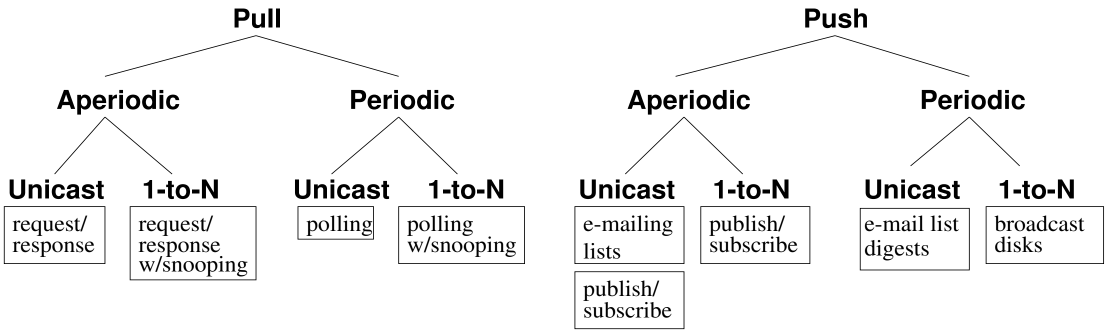

Figure 1: Data Delivery Options

profile) indicating the types of information they wish to receive. Publish/subscribe is push-based; data flow is initiated by the data sources, and is aperiodic, as there is no predefined schedule for sending data. Publish/subscribe protocols are inherently 1-to-N in nature, but due to limitations in current Internet technology, they are often implemented using individual unicast messages to multiple clients. Examples of such systems include Internet e-mail lists and some existing "push" systems on the Internet. True 1-to-N delivery is possible through technologies such as IP-Multicast, but such solutions are typically limited to individual Intranets or Local Area Networks.

Periodic Push - Periodic push has been used for data dissemination in many systems. An example of Periodic Push using unicast is Internet mailing lists that send out "digests" on a regular schedule. For example, the Majordomo system allows a list manager to set up a schedule (e.g., weekly) for sending digests. Such digests allow users to follow a mailing list without being continually interrupted by individual messages. There have also been many systems that use Periodic Push over a broadcast or multicast link. These include TeleText [Amma85, Wong88], DataCycle [Herm87], Broadcast Disks [Acha95a, Acha95b] and mobile databases [Imie94].

2.3 End-to-End Considerations

The second source of confusion about push technology is the fact that networked information systems typically contain many interconnected nodes. These nodes may be (logically) organized in various structures, and different data delivery mechanisms may be used between different sets of nodes. Given the potential heterogeneity of delivery mechanisms in a complex system, it is often not appropriate to describe the entire end-to-end (i.e., data source to consumer) system as "push-based" or "pull-based".

In general, a distributed information system can be though of as having 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*, (or agents, mediators, etc.) 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.

While the previous discussion has focused primarily on different modes of data delivery, the brokers provide the glue that binds these modes together. In many cases, the expected usage patterns of the brokers can drive the selection of which mode of delivery to use. For example, a broker that typically is very heavily loaded with requests could be an excellent candidate for a push-based delivery mechanism to its clients.

As we move upstream in the data delivery chain, brokers look like data sources to their clients. Receivers of information cannot detect the details of interconnections any further upstream than their immediate predecessor. This principle of *network transparency* allows data delivery mechanisms to change without having global impact. Suppose that node B is pulling data values from node A on demand. Further, suppose that node C is listening to a periodic broadcast from node B which includes values that B has pulled from A. Node C will not have to change it's data gathering strategy if A begins to push values to B. Changes in links are of interest only to the nodes that are directly involved. Likewise, this transparency allows the "appearance" of the data delivery at any node to differ from the way the data is actually delivered earlier in the network. This ability to change the appearance of data delivery is at the root of much of the confusion surrounding push technology.

Figure 2 shows a simple example of the importance of considering multiple network components and the impact of transparency. The figure shows how data delivery is performed in the initial versions of PointCast. To the user sitting at the screen, the system appears to be "push-based"; data flows across the screen without any user intervention. Due to current limitations of the Internet, however, that data is actually brought over to the client machine using a stream of periodic pull requests, delivered in a unicast fashion. Thus, the implementation of PointCast 1.0 between the client and the PointCast server is actually the exact opposite of the view that is presented to the user *in all three dimensions* of the hierarchy of Figure 1. This situation is not unique to PointCast;

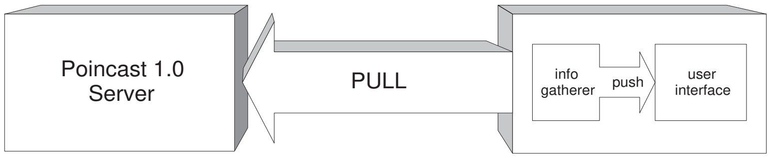

Figure 2: Pointcast 1.0

in fact, it is true for virtually all of the Internet-based push solutions, and stems from the fact that current IP and HTTP protocols do not adequately support push or 1-to-N communication.

3 Reexamining Current Push Technology

The previous section identified several of the sources of confusion in the current discussions and debate regarding push technology. In particular, the confusion stems from the mismatch between the user's perception and the actual data delivery mechanisms used by the system. Furthermore, this mismatch is also at the root of many of the performance concerns (particularly bandwidth overload) associated with current push technology. The impact of the mismatch on performance can be summarized as follows:

Pull instead of push - Current webcasting solutions typically use data pull to obtain information from data sources. This choice is due to limitations of the HTTP protocol, which is primarily pull-based. As stated previously, replacing push with pull requires that the pull be done in a *polling* manner. Polling can be quite resource intensive because it generates many requests. These requests consume client, server, and network resources. The problems are exacerbated if all clients poll individually, which could result in servers becoming overloaded due to the high volume of requests.

Periodic instead of aperiodic - Polling is typically done in a periodic manner that is independent of the events (e.g., data modifications) that would require data to be transfered. This independence results in a granularity problem: if polling is done too frequently, then the overhead can become substantial; if it is done too infrequently, then clients may unknowingly be accessing stale data.

Unicast instead of 1-to-N - In the absence of a true broadcast or multicast facility, systems that require 1-to-N behavior must implement it using multiple identical messages, one for each intended recipient. The potential bandwidth problems of such an approach are obvious. If *n* clients are interested in the same data item, then that same item must be sent over the network *n* times.

Fortunately, the concept of Network Transparency can be used to ameliorate this situation. One solution involves placing a local server inside an organization's firewall. All the clients interact with the local server in the way that is most appropriate for the local network and system configuration. The local server can then perform polling of the remote data source on behalf of the entire organization, which reduces Internet traffic. Likewise, the data source needs only to send a single copy of each data item to the local server, which can then distribute it to all the clients it represents. The local server can then multicast the data to its clients, if such capability exists.

4 Conclusions

In summary, push is currently a hot topic, but it is essential that it be placed in the proper context. Push is one choice (among many) for data delivery in distributed information systems. Push is not, for example, the same as broadcast. In fact, many existing push-based products are based on periodic pull over unicast connections. In our work on data dissemination, we have advocated a new look at the construction of distributed information systems that allows a seamless integration of all data delivery mechanisms including, but not limited to the various forms of push. We believe that this is a fertile area of work for the database community since the use of careful data management techniques in this context can have a significant impact on overall system performance and usability.

References

[Acha95a] S. Acharya, R. Alonso, M. Franklin, S. Zdonik, "Broadcast Disks: Data Management for Asymmetric Communication Environments", *ACM SIGMOD Conf.*, San Jose, May, 1995.

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

[Acha97] S. Acharya, M. Franklin, S. Zdonik, "Balancing Push and Pull for Data Broadcast", *ACM SIGMOD Conf.*, 1997.

[Akso98] D. Aksoy, M. Franklin, "Scheduling for Large-Scale On-Demand Data Broadcasting", *Proc. IEEE INFOCOM Conf.*, San Francisco, March, 1998.

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

[Fran97] M. Franklin, S. Zdonik, "A Framework for Scalable Dissemination-Based Information Systems" *ACM OOPSLA Conf.*, Atlanta, October, 1997.

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

[Glan96] D. Glance, "Multicast Support for Data Dissemination in OrbixTalk", *IEEE Data Engineering Bulletin*, 19(3), Sept., 1996.

[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", *ACM SIGMOD Conf.*, 1994.

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

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

[Yan95] T. Yan, H. Garcia-Molina, "SIFT – A Tool for Wide-area Information Dissemination", *Proc. 1995 USENIX Technical Conference*, 1995.