Dissemination of Dynamic Data*

Pavan Deolasee Amol Katkar Ankur Panchbudhe Krithi Ramamritham Prashant Shenoy

Department of Computer Science and Engineering.
Indian Institute of Technology Bombay
Mumbai, India 400076
{*pavan,amolk,ankurp,krithi*}@cse.iitb.ernet.in

Department of Computer Science
University of Massachusetts
Amherst, MA 01003
{*krithi,shenoy*}@cs.umass.edu

An increasing fraction of the data disseminated through the the internet is time-varying (i.e., changes frequently). Examples of such data include sports information, news, and financial information such as stock prices. An important issue then is the maintenance of their *temporal coherency*. A user may be willing to receive sports and news information that may be out-of-sync by a few minutes with respect to the server, but may desire to have stronger coherency requirements for data items such as stock prices. We assume that a user specifies a temporal coherency requirement (tcr) for each item of interest. The value of tcr denotes the maximum permissible deviation of the presented value from the value at the server and thus constitutes the user-specified tolerance. We consider temporal coherency requirements specified in terms of the value of the object (maintaining temporal coherency specified in units of time is a simpler problem that requires less sophisticated techniques). As shown in Figure 1, let $S(t)$, $P(t)$ and $U(t)$ denote the value of the data item at the server, proxy cache and the user, respectively. Then, to maintain temporal coherency we should have $|U(t) - S(t)| \leq c$.

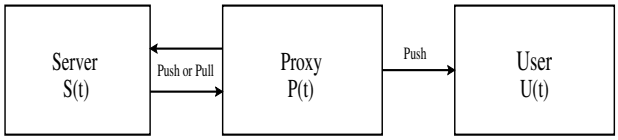

Figure 1: The Problem of Temporal Coherency

Push vs. Pull. In the case of servers that only respond to *pull*s, clients need to frequently *pull* the data based on the dynamics of the data and a user's coherency requirements. In contrast, servers that possess *push* capability maintain state information pertaining to clients and push only those changes that are of interest to a user. We have conducted extensive evaluation of the canonical push- and pull-based techniques using traces of real-world dynamic data. Our results show that (a) a pull-based approach does not offer high fidelity when the data changes rapidly or when the coherency requirements are stringent. Moreover, the pull-based approach imposes a large communication overhead (in terms of the number of messages exchanged) when the number of clients is large; (b) a push-based algorithm can offer high fidelity for rapidly changing data and/or stringent coherency requirements. However, it incurs a significant computational and state-space overhead resulting from a large number of open push connections. Moreover, the approach is less resilient to failures due to its stateful nature. These properties indicate that a push-based approach is suitable when a client requires its coherency requirements to be satisfied with a high fidelity, or when the communication overheads are the bottleneck. A pull-based approach is better suited to less frequently changing data or for less stringent coherency requirements, and when resilience to failures is important.

Adaptive Combinations of Push and Pull. Tcrs of clients may vary across clients and bandwidth availability may vary with time, so a static solution to the problem of disseminating dynamic, i.e., time-varying, data will not be responsive to client needs or load and bandwidth changes. We need an *intelligent* and *adaptive* approach that can be tuned according to the client requirements and conditions prevailing in the network or at the server/proxy. Moreover, the approach should not sacrifice the scalability of the server (under load) or reduce the resiliency of the system to failures. To this end, we demonstrate several techniques[1] that combine push and pull in an *intelligent* and *adaptive* manner while offering good resiliency and scalability. These include:

1. PaP, which simultaneously employs both push and pull to disseminate data, but has tunable parameters to determine the degree to which push and pull are used. Conceptually, pull is used, with the server allowed to push additional updates that are undetected by the pulls. By appropriate tuning, our algorithm can be made to behave as a push algorithm, a pull algorithm or a combination. Since both push and pull are simultaneously employed, albeit to different degrees, we refer to this algorithm as *Push-and-Pull (PaP)*.

2. PoP, which allows a server to adaptively choose between push- and pull-based dissemination for each connection. Moreover, the algorithm can switch each connection from push to pull and vice versa depending on the rate of change of data, the temporal coherency requirements and resource availability. Since the algorithm dynamically makes a choice of push or pull, we refer to it as *Push-or-Pull (PoP)*.

We demonstrate these algorithms using continuous queries over dynamic Web data.

*Work supported by National Science foundation grants IRI-9619588, CCR-9984030, CDA-9502639 and EIA-0080119, Tata Consultancy Services, IBM, MERL, EMC, IBM, Intel, and Sprint.

[1] P. Deolasee, A. Katkar, A. Panchbudhe, K. Ramamritham, and P. Shenoy, Adaptive Push-Pull: Dissemination of Dynamic Web Data, *10th International World Wide Web Conference*, May 2001.