Efficient Monitoring and Querying of Distributed, Dynamic Data via Approximate Replication

Christopher Olston
Carnegie Mellon University

Jennifer Widom
Stanford University

Abstract

It is increasingly common for an application's data to reside at multiple disparate locations, while the application requires centralized access to its data. A simple solution is to replicate *all relevant data at a central point, forwarding updates from master copies to replicas without any special processing or filtering along the way. This scheme maintains up-to-date centralized data, but incurs signficant communication overhead when the data is highly dynamic, because the volume of updates is large. If communication resources are precious, communication can be reduced by prioritizing and filtering updates inside the network, at or near the data sources. When updates are dropped, the replicas become* approximate *rather than exact. Fortunately, many real-world applications involving distributed, dynamic data can tolerate approximate data values to some extent, so* approximate replication *is an important technique for balancing replica precision against the communication resources to achieve it.*

This paper studies the problem of making efficient use of communication resources in approximate replication environments. After motivating and formalizing the problem, high-level descriptions of several complementary solutions are provided. The details of these solutions are found in previous papers by the authors, which are referenced here. This paper is intended to serve primarily as an introduction to and roadmap for the authors' prior work on approximate replication, as well as providing a significant bibliography of related work.

1 Introduction

In distributed environments that collect or monitor data, useful data may be spread across multiple distributed nodes, but users or applications may wish to access that data from a single location. One of the most common ways to facilitate centralized access to distributed data is to maintain copies of data objects of interest at central locations using *replication*. In a typical replication environment, illustrated abstractly in Figure 1, a central *data repository* maintains copies, or *replicas* of data objects whose *master copies* are spread across multiple remote and distributed *data sources*. (In general there may be multiple data repositories, but to simplify exposition we focus on a single repository.) Replicas are kept synchronized to some degree with remote master copies using communication links between the central repository and each source. In this way, querying and monitoring of distributed data can be performed indirectly by accessing replicas in the central repository.

While querying and monitoring procedures tend to become simpler and more efficient when reduced to centralized data access tasks, a significant challenge remains: that of performing data replication efficiently and effectively. Ideally, replicas of data objects at the central repository are kept exactly consistent, or synchronized, with the remote master copies at all times (modulo unavoidable communication latencies, of course). However,

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

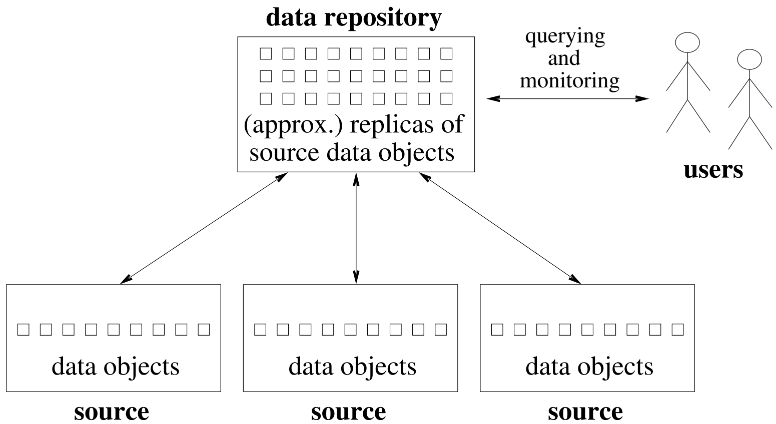


Figure 1: Abstract replication architecture.

propagating all master copy updates to remote replicas may be infeasible or prohibitively expensive: data collections may be large or frequently updated, and network or computational resources may be limited or only usable at a premium.

Situations where exact replica consistency is infeasible or only feasible at excessive expense can be found in many contexts. As one example, in video conferencing applications (*e.g.*, [6]), the viewer screen frame-buffer can be thought of as containing replicas of video data generated by remote cameras. Since streaming video data can be very large, it often becomes necessary to allow some staleness on parts of the screen. As another example, consider the important problem of monitoring computer networks in real-time to support traffic engineering, online billing, detection of malicious activity, etc. Effective monitoring often requires replicating at a central location a large number of measurements captured at remote and disparate points in the network. Attempting to maintain exact consistency can easily result in excessive burden being placed on the network infrastructure, thus defeating the purpose of monitoring [7], but fortunately network monitoring applications do not usually require exact consistency [17]. As a final example, consider the problem of indexing the World-Wide Web. Keeping an up-to-date Web index requires maintaining information about the latest version of every document. Currently, Web indexers are unable to maintain anything close to exact consistency due to an astronomical number of data sources and data that is constantly changing.

The infeasibility of exact replication in these environments and others is due in large part to the potentially high communication costs incurred. Communication cost tends to be of significant concern in many distributed environments, either because the bandwidth available on the network links is limited (relative to the size and update rate of the data collection), or because network resources can only be used at some premium. This premium for network usage may stem from the fact that increased congestion may cause service quality degradation for all applications that use the network. Alternatively, the premium may be manifest as a monetary cost, either in terms of direct payment to a service provider or as a loss of revenue due to an inability to sell consumed resources to others. As a result of communication resources being a valuable commodity, we have seen that in the applications described above (video conferencing, network monitoring, and Web indexing), maintaining replicas exactly synchronized with master copies cannot be achieved when data volumes or change rates are high relative to the cost or availability of bandwidth capacity.

1.1 Approximate Replication

In many applications such as the ones described above, exact consistency is not a requirement, and replicas that are not precisely synchronized with their master copies are still useful. For example, approximate readings from meteorological sensors often suffice when performing predictive modeling of weather conditions. In network security applications, “ball-park” estimates of current traffic levels can be used to detect potential denial-of-service attacks. Since exact data is often not a requirement in applications that rely on replication, it is common practice to use inexact replica consistency techniques such as periodic refreshing to conserve communication cost. We use the term *approximate replication* to refer collectively to all replication techniques that do not ensure exact consistency. Most existing approximate replication techniques for single-master environments can be classified into one of two broad categories based on the way they synchronize replicas[1]:

Periodic pushing: Sources propagate changes in master copies to the central data repository periodically, sometimes in large batches.

Periodic pulling: The central data repository accesses remote sources periodically to read master copies of data objects and update local replicas as necessary.

One motivation for performing periodic pushing is that one-way messages can be used in place of more expensive, round-trip ones. Also, sources can control the amount of their local resources devoted to replica synchronization. Periodic pulling, on the other hand, has the advantage that sources are not required to be active participants in the replica synchronization protocol. Instead, they need only respond to data read requests from the central repository, a standard operation in most environments. A principal feature shared by both these approaches is that the cost incurred for consumption of communication resources is bounded and controllable, which is not in general the case with exact synchronization methods. However, periodic pushing or pulling does not necessarily make good use of communication resources for the following two reasons:

1. Communication resources may be used wastefully while refreshing replicas of data objects whose master copy has undergone little or no change.

2. When the master copy of a data object undergoes a major change that could be propagated to the remote replica relatively cheaply, there may be a significant delay before the remote replica is refreshed to reflect the change.

1.2 Precision-Performance Tradeoff

We study the problem of making better use of communication resources in approximate replication environments than approaches based on periodic pulling or pushing. Our work begins with the observation that a fundamental tradeoff exists between the communication cost incurred while keeping data replicas synchronized and the degree of synchronization achieved. We refer to this characteristic property as the *precision-performance tradeoff*, illustrated in Figure 2, where *precision* is a measure of the degree of synchronization between a data object's master copy and a remote replica, and *performance* refers to how sparingly communication resources are used (*i.e.*, the inverse of communication cost). When data changes rapidly, good performance can only be achieved by sacrificing replica precision and, conversely, obtaining high precision tends to degrade performance.

Since there appears to be no way to circumvent the fundamental precision-performance tradeoff, we propose the following two tactics for designing synchronization algorithms for replication systems:

(a) Push the precision-performance curve as far away from the origin as possible for a given environment.
(b) Offer convenient mechanisms for controlling the point of system operation on the tradeoff curve.

Tactic (a) leads us to focus primarily on push-based approaches to replica synchronization, because they offer the opportunity for the best precision-performance curves, *i.e.*, more efficient use of communication resources,

[1]Depending on the environment, it may not be practical or possible for sources to communicate with each other, so we assume that such communication is not allowed and synchronization of replicas is performed directly between each source and the central repository. We do not study environments amenable to efficient intersource communication in this paper.

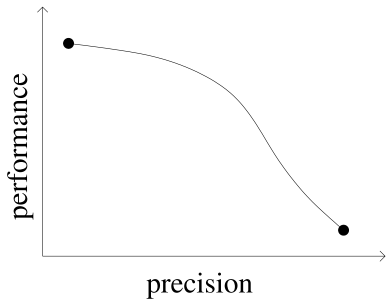
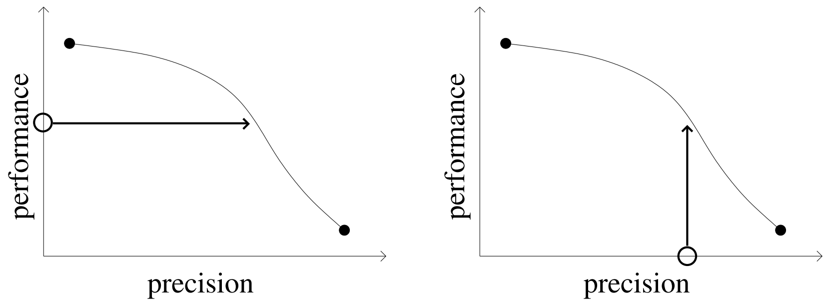


Figure 2: Precision-performance tradeoff.



Performance Fixed/Maximize Precision *Precision Fixed/Maximize Performance*

Figure 3: Strategies for managing the precision-performance tradeoff.

compared with pull-based approaches. (We verify this claim empirically in [14].) The reason for this advantage is that sources are better equipped than the central repository to make decisions about synchronization actions since they have access to the data itself and can obtain accurate precision measurements. Furthermore, our work uses metrics of replica precision that are based on content rather than solely on metadata. Metadata-based metrics have been used to drive synchronization policies in approximate replication environments (see Section 4 covering related work), and are usually intended to emulate an underlying metric based on content. For example, metadata metrics based on temporal staleness or number of unreported updates reflect an attempt to capture some notion of the degree of change to the content. If an appropriate content-based metric is used instead, precision can be measured directly, potentially leading to higher quality synchronization and precision-performance curves farther from the origin, as desired.

Tactic (b) leads us to study two ways to offer users or applications control over the position of system operation on the precision-performance tradeoff curve:

1. Users specify a minimum acceptable performance level (*i.e.*, fix a y-axis position in Figure 2), and the replication system attempts to maximize replica precision automatically while achieving the specified level of performance.

2. Users specify a minimum allowable precision level (*i.e.*, fix an x-axis position in Figure 2), and the replication system attempts to maximize performance while meeting the precision requirement provided.

In each of these complementary strategies, the user (or application) fixes the position of system operation along one dimension (precision or performance), and the system is expected to maximize the position along the other dimension. These converse strategies for establishing a point of system operation on the precision-performance curve are illustrated abstractly in Figure 3.

In our prior work [11, 12, 13, 14] we have studied the two strategies introduced above for controlling the operating point of a replication system in terms of precision and performance. In particular, we have proposed methods by which users or applications may constrain one dimension of the precision-performance space, and studied algorithms for maximizing the position along the other dimension. Our work forms the basis for offering a resource-efficient, user-controllable tradeoff between precision and performance while monitoring and querying distributed, dynamic data. In this paper we provide a high-level overview of this work (Sections 2 and 3), along with a summary of related work by others (Section 4).

2 Model, Assumptions, and Objectives

Recall that master copies of data objects are maintained at one or more distributed data sources (Figure 1). Consider a data object whose master source copy O undergoes updates over time. Let $R(O)$ represent the (possibly imprecise) replica of O at the central repository. Let $V(O,t)$ represent the value of O at time t. The value of O remains constant between updates. Let $V(R(O),t)$ represent the value of $R(O)$ at time t. Object O can be *refreshed* at time t_r, in which case a message is sent to the repository, and the replicated value is set to equal the current source value: $V(R(O),t_r) = V(O,t_r)$.

For simplicity, we assume that the communication latency between sources and the central repository is small enough to be neglected. However, the techniques we have developed can tolerate nonnegligible latencies, as discussed in [10]. We do assume there is adequate space at the central repository to store all replicas of interest. We also suppose that nodes are at all times connected to the network, and that the infrastructure is robust, *i.e.*, that node failures, network partitions, etc. are infrequent. Coping with failures or disconnections in the context of this work is not addressed in our work.

We now define the dual objectives of maximizing precision or performance in more detail.

2.1 Maximizing Precision

For strategy (1) in Section 1.2, the goal is to maximize replica precision. Precision is quantified using a simple and general metric called *divergence*: The divergence between a source object O and its replicated copy $R(O)$ at time t is given by a numerical function $D(O,t)$. When a refresh occurs at time t_r, the divergence value is zero: $D(O,t_r) = 0$. Between refreshes, the divergence value may become greater than zero, and the exact divergence value depends on how the master source copy relates to the replica. There are many different ways to measure divergence that are appropriate in different settings; see [14] for more discussion.

For the purpose of measuring overall divergence in the central repository, we associate with each data object O a numeric weight W_O that can be determined using a variety of criteria such as importance or frequency of access. The objective of strategy (1) is to minimize the weighted sum of the time-averaged divergence of each object, $\sum_O [W_O \cdot \int_t D(O,t)\,dt]$, under constraints on communication resources.

Communication resources may be limited at a number of points. First, the capacity of the link connecting the central data repository to the rest of the network, the *repository-side* bandwidth, may be constrained. Second, the capacity of the link connecting each source to the rest of the network, the *source-side bandwidth*, may also be constrained and may vary among sources. Moreover, all bandwidth capacities may fluctuate over time if resource limitations are related to traffic generated by other applications. We assume a standard underlying network model where any messages for which there is not enough capacity become enqueued for later transmission.

2.2 Maximizing Performance

The objective of strategy (2) in Section 1.2 is to maximize performance by minimizing the total *communication cost* incurred during a period of time in which replica precision is constrained. Each message sent between object O's source and the central repository (such as a refresh message) incurs a numeric cost $C_O \geq 0$, and costs are additive.

Constraints on precision arise with respect to *queries*, which are submitted by users or applications in order to access data. We consider situations in which aggregation queries over numeric data objects are submitted to the central repository. The repository is to provide answers to queries in the form of numeric intervals $[L,H]$ that are guaranteed to contain the precise answer V that could be obtained by accessing current master source copies, *i.e.*, $L \leq V \leq H$. As discussed later, guaranteed answer intervals can be produced by establishing bounds on replica divergence. Each query submitted at the central repository specifies a *precision constraint* that specifies the maximum acceptable width $(H-L)$ for the answer interval. Two modes of querying are considered in our work. *One-time queries* request the answer a single time, and do not persist once the (approximate) answer has been produced. By contrast, *continuous queries* are ongoing requests for continually updated answers that meet the specified precision constraint at all times.

3 Overview of Our Work on Approximate Replication

Maximizing Precision. In [14] we tackle the problem of maximizing replica precision when communication performance is limited due to constraints placed by users, applications, or the network infrastructure. The first step is to provide a general definition of precision, based on replica divergence, that can be specialized to a variety of data domains. We then present a replica synchronization technique whereby sources prioritize data objects that need to be synchronized based on precision considerations, and push synchronization messages to the central repository in priority order. The rate at which each source sends synchronization messages is regulated adaptively to ensure that the overall message rate conforms to the performance constraints. We evaluate our technique empirically and compare its effectiveness with that of a prior pull-based approach.

Maximizing Performance. In [11] we tackle the inverse problem: maximizing communication performance while maintaining acceptable levels of data precision as specified by users or applications at the granularity of queries over groups of objects. We focus on continuous queries, or CQ's for short, and propose a technique for performing push-based replication that meets the precision constraints of all continuous queries at all times. Without violating any query-level precision constraints, our technique continually adjusts the precision of individual replicas to maximize the overall communication performance. Results are provided from several experiments evaluating the performance of our technique in a simulated environment. In addition, we describe a testbed network traffic monitoring system we built to track usage patterns and flag potential security hazards using continuous queries with precision constraints. Experiments in this real-world application verify the effectiveness of our CQ-based approximate replication technique in achieving low communication cost while guaranteeing precision for a workload of multiple continuous queries.

Answering Unexpected Queries. Providing guaranteed precision for a workload of continuous queries does not handle an important class of queries that arises frequently in practice: one-time, unanticipated queries. Users or applications interacting with the data repository may at any time desire to obtain a one-time result of a certain query, which includes a precision constraint and may be different from the continuous queries currently being evaluated, or the same as a current CQ but with a more stringent precision constraint. Due to the ad-hoc nature of one-time queries, when one is issued the data replicas in the repository may not be of sufficient precision to meet the query's precision constraint. To obtain a query answer of adequate precision it may be necessary to access master copies of a subset of the queried data objects by contacting remote sources, incurring additional performance penalties. In [13] we study the problem of maximizing performance in the presence of unanticipated one-time queries, and devise efficient algorithms for minimizing accesses to remote master copies.

Managing Precision for One-Time Queries. A significant factor determining the cost to evaluate one-time queries with precision constraints at a central data repository is the precision of data replicas maintained in the repository. In [12] we study the problem of deciding what precision levels to use when replicating data objects not involved in continuous queries but subject to intermittent accesses by one-time queries. Interestingly, this problem generalizes a previously studied problem of deciding whether or not to perform exact replication of individual data objects. We propose an adaptive algorithm for setting replica precision with the goal of maximizing overall communication performance given a workload of one-time queries. In an empirical study we compare our algorithm with a prior algorithm that addresses the less general exact replication problem, and we show that our algorithm subsumes it in performance.

4 Overview of Related Work

We now cover previous work by others that is broadly related to our own. Other work not covered here is related to fairly specific aspects of our work; see [10] for additional coverage.

A number of replication strategies have been proposed based on abandoning strict transactional replication protocols that guarantee one-copy serializability, and performing asynchronous propagation of all database updates in a nontransactional fashion [5, 15], in order to reduce response time and improve availability. These approaches alleviate many of the problems associated with transactional protocols, but they do not focus on reducing communication cost except for some batching effects since all updates are propagated eventually.

The focus of our work is on conserving communication cost in nontransactional environments by performing approximate replication. The need for approximate replication is perhaps most obvious in the distributed World Wide Web environment. Partly this need arises because exact consistency is virtually impossible in the presence of the high degree of autonomy featured on the Web, but also because the volume of data is vast and aggregate data change rates are astronomical. On the Web, two forms of approximate replication are currently in heavy use: Web crawling and Web caching. Mechanisms for synchronizing document replicas in Web repositories via *incremental Web crawling*, *e.g.*, [18], are typically designed to work within a fixed communication budget, and generally aim to minimize some variant of the *temporal staleness* metric: the average amount of time during which Web document replicas in the repository differ in some way from the remote master copy. In Web caching environments, synchronization of a set of documents selected for caching is typically driven by constraints on the temporal staleness of cached replicas (the constraints are commonly referred to as time-to-live (TTL) restrictions), and the goal is to minimize communication, *e.g.*, [3].

Incremental Web crawling can be thought of as an instance of the *Performance Fixed/Maximize Precision* scenario, while Web caching represents an instance of the inverse *Precision Fixed/Maximize Performance* scenario. Due to the high degree of autonomy present in the Web environment, solutions to these problems almost always employ pull-oriented techniques for replica synchronization. In addition, owing to the wide variety of content found on the Web, synchronization techniques usually optimize for temporal staleness, a simple precision metric based solely on metadata. (Other metadata-based precision metrics have also been proposed, such as the number of updates not reflected in the remote replica, *e.g.*, [8].)

For replication environments in which greater cooperation among nodes is possible and more is known about the nature of the data and needs of the users, push-oriented synchronization based on richer content-based precision metrics tends to lead to more desirable results, *i.e.*, higher quality synchronization at lower cost, as discussed in Section 1.2. The CU-SeeMe video conferencing project [6] represents an interesting instance of a push-oriented synchronization approach using direct, content-based precision metrics, which focuses on the *Performance Fixed/Maximize Precision* scenario. In CU-SeeMe, refreshes to different regions of remotely replicated images are delayed and reordered at sources based on application-specific precision metrics that take into account pixel color differences. Another domain-specific approach has been proposed for moving object tracking [19], which focuses on the *Precision Fixed/Maximize Performance* scenario, or alternatively aims at maximizing an overall "information cost" metric that combines precision and performance.

Our goal is to establish generic push-oriented approximate replication strategies that exploit and expose the fundamental precision-performance tradeoff common to all environments, in a manner suitable to a wide variety of applications that rely on replication. Some initial steps toward this goal have been made by others in previous work. To our knowledge, the first proposal on this topic was by Alonso et al. [1], and recently others have extended that work, *e.g.*, [9, 16, 20, 21] ([21] studies pull-oriented techniques). All of this work falls into the *Precision Fixed/Maximize Performance* category, with precision constraints specified at the granularity of individual objects. One portion of our work [14] focuses on the inverse problem of *Performance Fixed/Maximize Precision*, which to our knowledge has not been studied in a general, application-independent setting with flexible precision metrics.

The portion of our work that addresses the *Precision Fixed/Maximize Performance* problem departs significantly from previous work by considering precision constraints at the granularity of entire queries rather than at the granularity of individual replicated objects. The rationale for this choice is twofold. First, we sought to align the granularity of precision constraint specification with the granularity of data access. (Since queries may be posed over individual data objects, our mechanism generalizes the previous approach.) Second, precision constraints at the per-query granularity leave open the possibility for optimizing performance by adjusting the allocation of precision requirements across individual objects involved in large queries. Indeed, much of our work [11, 12] focuses on realizing such optimizations using adaptive precision-setting techniques, which we show to enable significant improvements in synchronization efficiency. (The application of adaptive precision-setting techniques to hierarchical replication topologies has recently been studied in [4], with a particular focus on sensor network environments.)

Our work is also unique in focusing on user control over query answer precision, which may lead to unpredictable precision requirements. Specifically, to our knowledge it is the first to consider the problem of efficiently evaluating one-time queries with user-specified precision constraints that may exceed the precision of current replicas, thereby requiring access to some exact source copies [13]. (A version of this problem using a probabilistic model of precision was studied subsequently in [2].)

Acknowledgements. We thank the following individuals for their valuable input and assistance with this work: Michael Franklin, Hector Garcia-Molina, Jing Jiang, and Boon Thau Loo.

References

[1] R. Alonso, D. Barbara, H. Garcia-Molina, and S. Abad. Quasi-copies: Efficient data sharing for information retrieval systems. In *Proc. EDBT*, 1988.

[2] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *Proc. ACM SIGMOD*, 2003.

[3] E. Cohen and H. Kaplan. Refreshment policies for web content caches. In *Proc. IEEE INFOCOM*, 2001.

[4] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Hierarchical in-network data aggregatio with quality guarantees. In *Proc. EDBT*, 2004.

[5] L. Do, P. Ram, and P. Drew. The need for distributed asynchronous transactions. In *Proc. ACM SIGMOD*, 1999.

[6] T. Dorcey. CU-SeeMe desktop videoconferencing software. *Connexions*, 9(3), 1995.

[7] A. Householder, A. Manion, L. Pesante, and G. Weaver. Managing the threat of denial-of-service attacks. Technical report, CMU Software Engineering Institute CERT Coordination Center, Oct. 2001. `http://www.cert.org/archive/pdf/Managing_DoS.pdf`.

[8] Y. Huang, R. Sloan, and O. Wolfson. Divergence caching in client-server architectures. In *Proc. PDIS*, 1994.

[9] A. Jain, E. Y. Chang, and Y.-F. Wang. Adaptive stream resource management using kalman filters. In *Proc. ACM SIGMOD*, 2004.

[10] C. Olston. Approximate replication. Doctoral dissertation, Stanford University, 2003.

[11] C. Olston, J. Jiang, and J. Widom. Adaptive filters for continuous queries over distributed data streams. In *Proc. ACM SIGMOD*, 2003.

[12] C. Olston, B. T. Loo, and J. Widom. Adaptive precision setting for cached approximate values. In *Proc. ACM SIGMOD*, 2001.

[13] C. Olston and J. Widom. Offering a precision-performance tradeoff for aggregation queries over replicated data. In *Proc. VLDB*, 2000.

[14] C. Olston and J. Widom. Best-effort cache synchronization with source cooperation. In *Proc. ACM SIGMOD*, 2002.

[15] Y. Saito and M. Shapiro. Replication: Optimistic approaches. Technical report, Hewlett-Packard Labs, 2002. HPL-2002-33.

[16] S. Shah, S. Dharmarajan, and K. Ramamritham. An efficient and resilient approach to filtering and disseminating streaming data. In *Proc. VLDB*, 2003.

[17] R. van Renesse and K. Birman. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. Technical report, Cornell University, 2001.

[18] J. L. Wolf, M. S. Squillante, P. S. Yu, J. Sethuraman, and L. Ozsen. Optimal crawling strategies for web search engines. In *Proc. WWW*, 2002.

[19] O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang. Moving objects databases: Issues and solutions. In *Proc. SSDBM*, 1998.

[20] H. Yu and A. Vahdat. Design and evaluation of a continuous consistency model for replicated services. In *Proc. OSDI*, 2000.

[21] S. Zhu and C. V. Ravishankar. Stochastic consistency, and scalable pull-based caching for erratic data stream sources. In *Proc. VLDB*, 2004.