Cost Based Data Dissemination in Broadcast Networks with Disconnection

Bo Xu and Ouri Wolfson
Department of Electrical Engineering and Computer Science
University of Illinois at Chicago
{bxu,wolfson}@eecs.uic.edu

Sam Chamberlain
Army Research Laboratory
wildman@arl.mil

Abstract

We consider the problem of data dissemination in a broadcast network. In contrast to previously studied models, broadcasting is among peers, rather than client server. Such a model represents, for example, satellite communication among widely distributed nodes, sensor networks, and mobile ad-hoc networks. We introduce a cost model for data dissemination in peer to peer broadcast networks. The model quantifies the tradeoff between the inconsistency of the data, and its transmission cost; the transmission cost may be given in terms of dollars, energy, or bandwidth. Using the model we first determine the parameters for which eager (i.e. consistent) replication has a lower cost than lazy (i.e. inconsistent) replication. Then we introduce a lazy broadcast policy and compare it with several naive or traditional approaches to solving the problem.

1 Introduction

A mobile computing problem that has generated a significant amount of interest in the database community is data broadcasting (see for example [42]). The problem is how to organize the pages in a broadcast from a server to a large client population in the dissemination of public information (e.g. electronic news services, stock-price information, etc.). A strongly related problem is how to replicate (or cache) the broadcast data in the Mobile Units that receive the broadcast.

*This research was supported in part by Army Research Labs grant DAAL01-96-2-0003, NATO grant CRG-960648.

In this paper we study the problems of broadcasting and replication in a peer to peer rather than client server architecture. More precisely, we study the problem of dissemination, i.e. full replication at all the nodes in the system. This architecture is motivated by new types of emerging wireless broadcast networks such as Mobile Ad-hoc Networks (see [13])[1], sensor and "smart dust" networks (see [27, 28]), and satellite networks. These networks enable novel applications in which the nodes of a network collaborate to assemble a complete database. For instance, in the case of sensors that are parachuted or sprayed from an airplane, the database renders a global picture of an unknown terrain from local images collected by individual sensors. Or, the database consists of the current location of each member in a military unit (in a MANET case), or another meaningful database constructed from a set of widely distributed fragments.

We model such applications using a "master" replication environment (see [23]), in which each node i "owns" the master copy of a data item D_i, i.e. it generates all the updates to D_i. For example, D_i may be the latest in a sequence of images taken periodically by the node i of its local surroundings. Each new image updates D_i. Or, D_i may be the location of the node which is moving; D_i is updated when

[1] A Mobile Ad-hoc Network (MANET) is a system of mobile computers (or nodes) equipped with wireless broadcast transmitters and receivers which are used for communicating within the system. Such networks provide an attractive and inexpensive alternative to the cellular infrastructures when this infrastructure is unavailable (e.g. in remote and disaster areas), or inefficient, or too expensive to use (see [33]). Mobile Ad-hoc Networks are used to communicate among the nodes of a military unit, in rescue and disaster relief operations, in collaborative mobile data exchange (e.g the set of attendees at a conference), and other "micronetworking" technologies (see [32]).

the Global Positioning System (GPS) on board the node i indicates a current location that deviates from D_i by more than a prespecified threshold. The database of interest is D = $\{D_1,...,D_n\}$, where n is the number of nodes and also the number of items in the database.[2]

It is required that D is accessible from each node in the network,[3] thus each node stores a (possibly inconsistent) copy of D. [4] Our paper deals with various policies of broadcasting updates of the data items. In each broadcast a data item is associated with its version number, and a node that receives a broadcasted data item updates its local database if and only if the local version is older than the newly arrived version. In the broadcast policies there is a tradeoff between data consistency and communication cost. In satellite networks the communication cost is in terms of actual dollars the customer is charged by the network provider; in sensor networks, due to the small size of the battery, the communication cost is in terms of energy consumption for message transmission; and in MANET's the critical cost component is bandwidth (see [13]). Bandwidth for (secure) communication is an important and scarce resource, particularly in military applications (see [38, 39]).

Now let us discuss the broadcast policies. One obvious policy is the following: for each node i, when D_i is updated, node i broadcasts the new version of D_i to the other nodes in the network. We call this the Single-item Broadcast Dissemination (SBD) policy. In the networks and applications we discuss in this paper, nodes may be disconnected, turned off or out of battery. Thus the broadcast of D_i may not be received by all the nodes in the system. A natural way to deal with this problem is to rebroadcast an update to D_i until it is acknowledged by all the nodes, i.e. Reliable Broadcast Dissemination (RBD). Clearly, if the new version is not much different than the previous one and if the probability of reception is low (thus necessitating multiple broadcasts), then this increase in communication cost is not justified. An alternative option, which we adopt in SBD, is to broadcast each update once, and let copies diverge. Thus the delivery of updates is unreliable, and consequently the dissemination of D_i is "lazy" in the sense that the copy of D_i stored at a node may be inconsistent.

How can we quantify the tradeoff between the increase in consistency afforded by a reliable broadcast and its increase in communication cost? In order to answer this question we introduce the concept of *inconsistency-cost* of a data item. This concept, in turn, is quantified via the notion of the cost difference between two versions of a data item D_i. In other words, the inconsistency cost of using an older version v rather than the latest version w is the distance between the two versions. For example, if D_i represents a location, then the cost difference between two versions of D_i can be taken to be the distance between the two locations. If D_i is an image, an existing algorithm that quantifies the difference between two images can be used (see for example [6]). If D_i is the quantity-on-hand of a widget, then the difference between the two versions is the difference between the quantities. Now, in order to quantify the tradeoff between inconsistency and communication one has to answer the question: what amount of bandwidth/energy/dollars am I willing to spend in order to reduce the inconsistency cost on a data item by one unit? Using this model we establish the cost formulas for RBD and SBD, i.e reliable and unreliable broadcasting, and based on them formulas for selecting one of the two policies for a given set of system parameters.

For the cases when unreliable broadcast, particularly SBD, is more appropriate, consistency of the local databases can be enhanced by a policy that we call Full Broadcast Dissemination (FBD). In FBD, whenever D_i is updated, i broadcasts its local copy of the whole database D, called $D(i)$. In other words, i broadcasts D_i, as well as its local version of each one of the other data items in the database. When a node j receives this broadcast, j updates its version of D_i, and j also updates its local copy of each other item D_k, for which the version number in $D(i)$ is more recent. Thus these indirect broadcasts of D_k (to j via i) are "gossip" messages that increase the consistency of each local database. However, again, this comes at the price of an increase in communication cost due to the fact that each broadcast message is n times longer.

The SBD and FBD policies represent in some sense two

[2]In case D_i is the location of i, the database D is of interest in what are called Moving Objects Database (MOD) applications (see [29, 31, 17, 18, 37]). If D_i is the location of object i in a battlefield situation, then a typical query may be: retrieve the friendly helicopters that are in a given region. Other MOD applications involve emergency (fire, police) vehicles and local transportation systems (e.g. city bus system).

[3]For example, the location of the members of a platoon should be viewable by any member at any time.

[4]By inconsistency of D we mean that some data items may not contain the most recent version.

extreme solutions on a consistency-communication spectrum of lazy dissemination policies. SBD has minimum communication cost and minimum local database consistency, whereas FBD has maximum communication cost and maximum (under the imperfect circumstances) local database consistency.

In this paper we introduce and analyze the Adaptive Broadcast Dissemination (ABD) policy that optimizes the tradeoff between consistency and communication using a cost based approach. In the ABD policy, when node i receives an update to D_i it first determines whether the expected reduction in inconsistency justifies broadcasting a message. If so, then i "pads" the broadcast message that contains D_i with a set S of data items (that i does not own) from its local database, such as to optimize the total cost. One problem that we solve in this paper is how to determine the set S, i.e. how node i should select for each broadcast message which data items from the local database to piggyback on D_i. In order to do so, i estimates for each j and k the expected benefit (in terms of inconsistency reduction) to node k of including in the broadcast message its local version of D_j.

Let us now put this paper in the context of existing work on consistency in distributed systems. Our approach is new as far as we know. Although gossiping has been studied extensively in distributed systems and databases (see section 6), none of the existing works uses an inconsistency-communication tradeoff cost function in order to determine what gossip messages to send. Furthermore, in the emerging resource constrained environments (e.g. sensor networks, satellite communication, and MANET's) this tradeoff is crucial. Also our notion of consistency is appropriate for the types of novel applications discussed in this paper, and is different than the traditional notion of consistency in distributed systems discussed in the literature (e.g., [3, 14, 7, 19]. Specifically, in contrast to the traditional approaches, our notion of consistency does not mean consistency of different copies of a data item at different nodes, and it does not mean mutual consistency of different data items at a node. In this paper a copy of a data item at a node is consistent if it has the latest version of the data item. Otherwise it is inconsistent, and the inconsistency cost is the distance between the local copy and the latest version of the data item. Inconsistency of a local database is simply the sum of the inconsistencies of all data items. We employ gossiping to reduce inconsistency, not to ensure consistency as in using vector clocks ([14, 3]).

In this paper we provide a comparative analysis of dissemination policies. The analysis is probabilistic and experimental, and it achieves the following objectives. First, it gives a formula for the expected total cost of SBD and RBD, and a complete characterization of the parameters for which each policy has a cost lower than the other. Second, for ABD we prove cost optimality for the set of data items broadcast by a node i, for i's level of knowledge of the system state. Third, the analysis compares the three unreliable policies discussed above, namely SBD, FBD, and ABD, and a fourth traditional one called flooding (FLD)[5] [40]. ABD proved to consistently outperform the other two policies, often having a total cost (that includes the cost of inconsistency and the cost of communication) that is several times lower than that of the other policies.

In summary, the key contributions of this paper are as follows.

- Introduction of a cost model to quantify the tradeoff between consistency and communication.
- Analyzing the performance of eager and lazy dissemination via reliable and unreliable broadcasts respectively, obtaining cost formulas for each case and determining the data and communication parameters for which eager is superior to lazy, and vice versa.
- Developing and analyzing the Adaptive Broadcast Dissemination policy, and comparing it to the other lazy dissemination policies.

The rest of the paper is organized as follows. In section 2 we introduce the operational model and the cost model. In section 3 we analyze and compare reliable and unreliable broadcasting. In section 4 we describe the ABD policy, and in section 5 we analyze it. In section 6 we compare the unreliable broadcast policies by simulation. In section 7 we discuss relevant work, and in the last section we summarize the paper. In appendix A we provide the proofs of our theorems and lemmas. In appendix B we describe further experimental results.

[5] In flooding a node i broadcasts each new data item it receives either as a results of a local update of D_i, or from a broadcast message.

2 The Model

In subsection 2.1 we precisely define the overall operational model, and in subsection 2.2 we define the cost model.

2.1 Operational model

The system consists of a set of n nodes that communicate by message broadcasting. Each node i $(1 < i < n)$ has a data item D_i associated with it. Node i is called D_i's owner. This data item may contain a single numeric value, or a complex data structure such as a motion plan, or an image of the local environment. Only i, and no other nodes, has the authorization to modify the state of D_i. A data item is updated at discrete time points. Each update creates a new version of the data item. In other words, the *kth version of* D_i, denoted $D_i(k)$, is generated by the kth update. We denote the latest version of D_i by $\overline{D_i}$. Furthermore, we use $v(D_i)$ to represent the version number of D_i, i.e. $v(D_i(k)) = k$. For two versions $D_i(k)$ and $D_i(k')$, we say that $D_i(k)$ is *newer than* $D_i(k')$ if $k > k'$, and $D_i(k)$ is *older than* $D_i(k')$ if $k < k'$.

An owner i periodically broadcasts its data item D_i to the rest of the system. Each such broadcast includes the version number of D_i. Since nodes may be disconnected, some broadcasts may be missed by some nodes, thus each node j has a version of each D_i which may be older than $\overline{D_i}$. The *local database* of node i at any given time is the set $< D^i_1, D^i_2, ..., D^i_n >$, where each D^i_j (for $1 \leq j \leq n$) is a version of D_j. Observe that since all the updates of D_i originate at i, then $D^i_i = \overline{D_i}$. Node i updates D^i_j $(j \neq i)$ in its local database when it receives a broadcast from j.

Nodes may be disconnected (e.g. shut down) and thus miss messages. Let p_i be the percentage of time a node i is connected. Then p_i is also the probability that i receives a message from any other node j. For example, if i is connected 60% of the time (i.e. $p_i = 0.6$), then a message from j is received by i with probability 0.6. We call p_i the *connection probability* of i.

2.2 Cost Model

In this subsection we introduce a cost function that quantifies the tradeoff between consistency and communication. The function has two purposes. First, to enable determining the items that will be included in each broadcast of the ABD policy, and second, to enable comparing the various policies.

Inconsistency cost

Assume that the distance between any two versions of a data item can be quantified. For example, in moving objects database (MOD) applications, the distance between two data item versions may be taken to be the Euclidean distance between the two locations. If D_i is an image, one of the many existing distance functions between images (e.g. the cross-correlation distance ([6])) can be used.

Formally, the *distance* between two versions $D_i(k)$ and $D_i(j)$, denoted $DIST(D_i(k), D_i(j))$, is a function whose domain is the nonnegative reals, and it has the property that the distance between two identical versions is 0. If the data item owned by each node consists of two or more types of logical objects, each with its own distance function, then the distance between the items should be taken to be the weighted averages of the pairwise distances.

We take the $DIST$ function to represent the cost, or the penalty, of using the older version rather than the newer one. More precisely, consider two consecutive updates on D_i, namely the kth update and the $(k+1)$st update. Assume that the kth update happened at time t_k and the $(k+1)$st update at time t_{k+1}. Intuitively, at time t_{k+1} each node j that did not receive the kth version $D_i(k)$ during the interval $[t_k, t_{k+1})$, pays a price which is equal to the distance between the latest version of D_i that j knows and $D_i(k)$. In other words, this price is the penalty that j pays for using an older version during the time in which j should have used $D_i(k)$. If j receives $D_i(k)$ sometime during the interval $[t_k, t_{k+1})$, then the price that j pays on D_i is zero. Formally, assume that at time t_{k+1} the latest version of D_i that j knows is v $(v \leq k)$. Then *j's inconsistency cost on version k of D_i* is $COST\_INCO_j(D_i(k)) = DIST(D_i(v), D_i(k))$.

The *inconsistency cost of the system on* $D_i(k)$ is $COST\_INCO(D_i(k)) = \sum_{1 \leq j \leq n} COST\_INCO_j(D_i(k))$.

The *total inconsistency cost of the system on D_i up to the mth update of D_i*, denoted $COST\_INCO(i, m)$, is $\sum_{1 \leq k < m} COST\_INCO(D_i(k))$.

The *total inconsistency cost for the system up to time t* is $COST\_INCO(t) = \sum_{1 \leq i \leq n} COST\_INCO(i, m_i)$, where m_i is the highest version number of D_i at time t.

Communication cost

The cost of a message depends on the length of the message. In particular, if there are m data items in a message, the cost of the message is $C_1 + m \cdot C_2$. [6]

C_1 is called the *message initiation cost* and C_2 is called the *message unit cost*. C_1 represents the cost of energy consumed by the CPU to prepare and send the message. C_2 represents the incremental cost of adding a data item to a message. The values of C_1 and C_2 are given in inconsistency cost units. They are determined based on the amount of resource that one is willing to spend in order to reduce the inconsistency cost on a version by one unit. For example, if $C_1 = C_2$ and one is willing to spend one message of one data item in order to reduce the inconsistency by at least 50, then $C_1 = C_2 = 1/100$.

The *total communication cost up to time t* is the sum of the costs of all the messages that have been broadcast from the beginning (time 0) until t.

System cost

The *system cost up to time t*, denoted $COST\_SYS(t)$, is the sum of the total inconsistency for the system up to t, and the total communication cost up to t. The system cost is the objective function optimized by the ABD policy. When comparing ABD with other broadcast policies, there are two additional costs, namely computation and storage, which will come into play. We will explain the inclusion of these costs in the model in section 6.

3 Reliable versus Unreliable Broadcasting

In this section we completely characterize the cases in which lazy dissemination by unreliable broadcasting outperforms eager dissemination by reliable broadcasting, and vice versa. Lazy dissemination is executed by the Single-item Broadcast Dissemination policy, in which each node i unreliably broadcasts each update it receives, when i receives it. Eager dissemination is executed by the Reliable Broadcast Dissemination (RBD) policy, in which each node i reliably broadcasts each update it receives, when i receives it; by reliable broadcast we mean that i retransmits the message until it is acknowledged by all the other nodes. Performance of the two policies is measured in terms of the system cost, as defined at the end of the previous section. We first derive the closed formulas for the system costs of SBD and RBD. Then, based on these formulas, we compare SBD and RBD.

3.1 Quantification of SBD and RBD Performance

In the following discussion, we assume that for each node i, the updates at i are generated by a Poisson process with intensity λ_i. Let $\lambda = \sum_{1 \le i \le n} \lambda_i$. The number of nodes in the system is n, the connection probability p_i for each node i, message initiation cost C_1, and the message unit cost C_2.

The following theorem gives the system cost of SBD up to a given point in time.

Theorem 1 The system cost of SBD up to time t (i.e. $COST\_SYS_{SBD}(t)$) is a random variable whose expected value is

$$E[COST\_SYS_{SBD}(t)] =$$
$$\lambda \cdot t \cdot (C_1 + C_2) + \sum_{1 \le i \le n} \sum_{m=1}^{\infty} (\frac{e^{-\lambda_i \cdot t} \cdot (\lambda_i \cdot t)^m}{m!} \cdot$$
$$\sum_{q=1}^{m-1} \sum_{1 \le j \le n \text{ and } j \ne i} ((1-p_j)^q \cdot DIST(D_i(0), D_i(q))$$
$$+ \sum_{k=1}^{q-1} p_j \cdot (1-p_j)^{q-k} \cdot DIST(D_i(k), D_i(q)))) \quad (1)$$

□

Now we analyze the system cost of the reliable broadcast dissemination (RBD) policy. First let us introduce a lemma which gives the expected number of times that a message is transmitted from node i (remember that in RBD a message is retransmitted until it is acknowledged by all the other nodes).

Lemma 1 Let R_i be the number of times that a message is transmitted. Then R_i is a random variable whose expected value is:

$$E[R_i] = \sum_{k=1}^{\infty} (k \cdot (\prod_{1 \le j \le n \text{ and } j \ne i} (1-(1-p_j)^k) - \prod_{1 \le j \le n \text{ and } j \ne i} (1-(1-p_j)^{k-1}))) \quad (2)$$

[6] Actually the cost of a message can be any non-decreasing function of the length of the message. In section 4.2 we will discuss how our approach can be extended to this more general case.

□

Theorem 2 The system cost of RBD up to time t (i.e. $COST\_SYS_{RBD}(t)$) is a random variable whose expected value is:

$$\begin{aligned} &E[COST\_SYS_{RBD}(t)] \\ &= (C_1 + C_2) \cdot t \cdot \sum_{i=1}^{n} (\lambda_i \cdot E[R_i]) \\ &\quad +(n-1) \cdot C_1 \cdot \lambda \cdot t \end{aligned} \tag{3}$$

(the value of $E[R_i]$ was derived in Lemma 1) □

3.2 Comparison of SBD and RBD

The objective of this subsection is to identify the situations in which SBD outperforms RBD, and vice versa.

Theorem 3 $E[COST\_SYS_{SBD}(t)] < E[COST\_SYS_{RBD}(t)]$ if and only if

$$C_1 > \frac{\sum_i E[COST\_INCO^i(t)] - (t \cdot \sum_i (\lambda_i \cdot E[R_i]) - \lambda \cdot t) \cdot C_2}{t \cdot \sum_i (\lambda_i \cdot E[R_i]) + \lambda \cdot t \cdot (n-2)} \tag{4}$$

(Recall that $E[COST\_INCO^i(t)]$ is the expected inconsistency cost of the system on D_i up to t).□

The meaning of Theorem 3 is visually expressed by Figure 1(a), where $K_1 = \frac{D}{A+B\cdot(n-2)}$ and $K_2 = \frac{D}{A-B}$. In Figure 1(a), inside and only inside the shadowed triangular area RBD is better than SBD. In other words, if the communication cost is relatively high, then it is better to use unreliable rather than reliable broadcasting. The intuition is that since in RBD each message may be transmitted more than once, as C_1 and C_2 increase, the system cost of RBD increases faster than that of SBD.

Observe that the characterization of Theorem 3 is dependent of the total cost of inconsistency of each data item (since K_1 and K_2 are dependent on these inconsistencies). In some cases, however, the difference between any two versions of D_i is a constant. For example, assume that if a node j does not have the latest version of D_i, then it pays a fixed cost (because, say, j makes an erroneous decision), regardless of the version of D_i that j actually has. In this case, the difference between any two arbitrary versions is a constant. Now we characterize when SBD is better than RBD in this case.

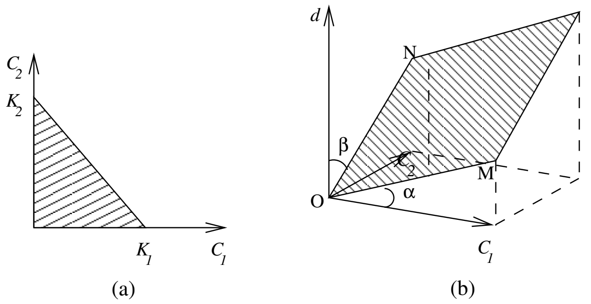


Figure 1. (a) RBD outperforms SBD inside and only inside the shadowed area; K_1 and K_2 depend on the inconsistency cost of each data item. (b) SBD outperforms RBD below and only below the shadowed plane

Theorem 4 Assume that for any node i, the difference between two arbitrary versions $D_i(k)$ and $D_i(j)$ is a constant d. Then the expected system cost of SBD up to t is:

$$\begin{aligned} &E[COST\_SYS_{SBD}(t)] = \lambda \cdot t \cdot (C_1 + C_2) \\ &+d \cdot \sum_{i=1}^{n} ((\lambda_i \cdot t + e^{-\lambda_i \cdot t} - 1) \cdot \sum_{1 \le j \le n}^{j \ne i} (1 - p_j)) \end{aligned} \tag{5}$$

□

Theorems 2 and 4 enable us to compare the performance of SBD and RBD for given C_1, C_2, and d. We identify the ranges of C_1, C_2, and d for which SBD outperforms RBD and vice versa. This is illustrated in Figure 1(b), where α is the angle between the line OM and the axis C_1, and β is the angle between the line ON and the axis d. $tg(\alpha) = \frac{A-B}{\sum_{i=1}^{n}((\lambda_i \cdot t + e^{-\lambda_i \cdot t} - 1) \cdot \sum_{1 \le j \le n \text{ and } j \ne i} (1-p_j))}$ and $ctg(\beta) = \frac{A+B\cdot(n-2)}{\sum_{i=1}^{n}((\lambda_i \cdot t + e^{-\lambda_i \cdot t} - 1) \cdot \sum_{1 \le j \le n \text{ and } j \ne i} (1-p_j))}$. Specifically, SBD is better than RBD if and only if the parameters C_1, C_2, and d denote a point which is below the shadowed plane of Figure 1(b). One of the implications of this result is that as a point on the (C_1 , C_2) plain moves farther away from the origin, SBD is better for a wider range of d's. This quantifies the intuition that SBD becomes the preferred policy as the communication cost increases.

4 The Adaptive Broadcast Dissemination Policy

In this section we describe the Adaptive Broadcast Dissemination policy. Intuitively, a node i executing the policy behaves as follows. When it receives an update to D_i, node i constructs a broadcast message by evaluating the benefit of including in the message each one of the data items in its local database. Specifically, the ABD policy executed by i consists of the following two steps.

(1) Benefit estimation: For each data item in the local database, estimate how much the inconsistency of the system could be reduced if that data item is included in the message.

(2) Message construction: Construct the message which is a subset of the local database so that the total estimated net benefit of the message is maximized (The net benefit is the difference between the inconsistency reduced by the message and the cost of the message). Observe that the set of data items to be broadcast may be empty. In other words, when D_i is updated, node i may estimate that the net benefit of broadcasting any data item is negative.

Each one of the above steps is executed by an algorithm which is described in one of the next two subsections.

4.1 Benefit Estimation

Intuitively, the benefit to the system of including a data item D_j in a message that node i broadcasts is in terms of inconsistency reduction. This reduction depends on the nodes that receive the broadcast, and on the latest version of D_j at each one of these nodes. Node i maintains data structures that enable it to estimate the latest version of D_j at each node. Then the benefit of including a data item D_j in a message that i broadcasts is simply the sum of the expected inconsistency reductions at all the nodes.

In computing the inconsistency reduction for a node k we attempt to be as accurate as possible, and we do so as follows. Node i maintains a "knowledge matrix" which stores in entry (k, j) the last version number of D_j that node i received from node k (this version is called $v(D_j^k)$), and the time when it was received. Additionally, i saves in the "real history" for each D_j all the versions of D_j that i has "heard" from other nodes, the times at which it has done so, and from which node they were received[7]. The reason for maintaining all this information is that now, in estimating which version of D_j node k has, node i can take into consideration two factors: (1) the last version of D_j that i received from k at time, say t, and (2) the fact that since time t node k may have received updates of D_j by "third party" messages that were transmitted after time t, and "heard" by both, k and i. Node i also saves with each version v of D_j that it "heard", the distance (i.e. the inconsistency caused by the version difference) between v and the last version of D_j that i knows; this difference is the parameter necessary in order to compute the inconsistency cost reduction that is obtained if node i broadcasts its latest version of D_j.

In subsection 4.1.1 we describe the data structures that are used by a node i in benefit estimation. In subsection 4.1.2 we present i's benefit estimation method.

4.1.1 Data Structures

(1) The Knowledge matrix: For each data item D_j ($j \neq i$), denote by $v(D_j^k)$ the latest version number of D_j that i received from k, and denote by $t(D_j^k)$ the last time when D_j^k was received at i. The *knowledge matrix* at node i is:

$$M_i = \begin{pmatrix} (t(D_1^1), v(D_1^1)) & (t(D_2^1), v(D_2^1)) & \dots & (t(D_n^1), v(D_n^1)) \\ (t(D_1^2), v(D_1^2)) & (t(D_2^2), v(D_2^2)) & \dots & (t(D_n^2), v(D_n^2)) \\ \vdots & \vdots & \ddots & \vdots \\ (t(D_1^n), v(D_1^n)) & (t(D_2^n), v(D_2^n)) & \dots & (t(D_n^n), v(D_n^n)) \end{pmatrix}$$

Node i updates the matrix whenever it receives a message. Specifically, when i receives a message from k that includes D_j, i updates the entry (k,j) of the matrix. In addition, if the version of D_j received is newer than the version in i's local database, then the newer version updates D_j in the local database.

(2) Version sequence: A version sequence records all the version numbers that i has ever known about a data item. Due to unreliability, it is possible that i has not received all the versions of a data item. In particular, the *version sequence of* D_j is $VS_j =< v_1, v_2, ..., v_h >$ where $v_1 < v_2 < ... < v_h$ are all the version numbers that i has ever known about D_j. For each $v \in VS_j$, i saves in the distance between $D_j(v)$ and $D_j(v_h)$. Figure 2(a) illustrates an

[7] There is a potential storage problem here, which we address, but we postpone the discussion for now

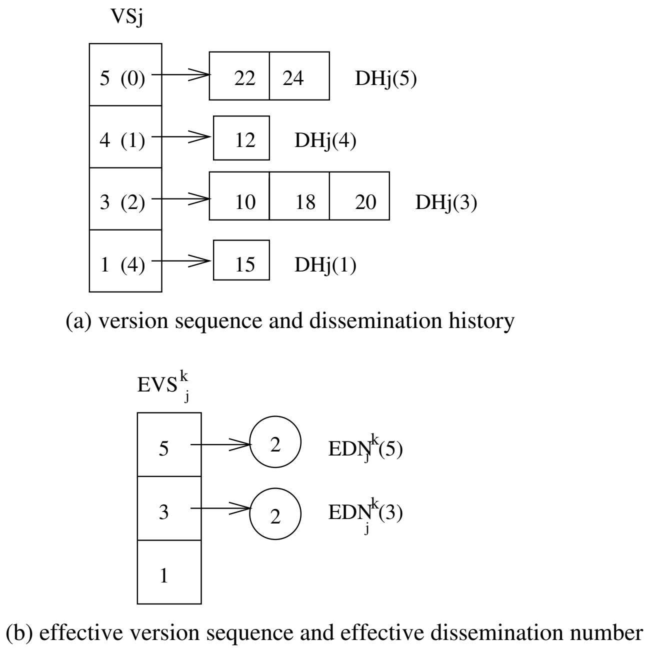


Figure 2. Data structures in benefit estimation

example of a dissemination history. The number in parenthesis besides a version number is the distance between that version of D_j and the last version of D_j which is 5. Thus, in this example $DIST(D_j(v), D_j(v')) = |v - v'|$.

(3) Dissemination history: For each version number v in each VS_j, i maintains a *dissemination history* $DH_j(v)$. This history records every time point at which i received $D_j(v)$ from a node. $DH_j(v)$ also contains every time point at which i broadcast $D_j(v)$. Figure 2(a) gives an example of a version sequence and its dissemination histories. Figure 2(a) shows that node i received version 1 of D_j at time 15, and it received version 3 at times 10,18 and 20.

Now we discuss how we limit the amount of storage used. Observe that the lengths of each version sequence VS_j and dissemination history $DH_j(v)$ increases unboundedly as i receives more broadcasts. This presents a storage problem. A straight-forward solution to this problem is to limit the length of each version sequence to α and the length of each dissemination history to β. We call this variant of the ABD policy ABD(α,β). The drawback of ABD(α,β) is that when the length of a dissemination history $DH_j(v)$ is smaller than β, since each dissemination history is limited to β, other dissemination histories can not make use of the free storage of $DH_j(v)$. A better solution, which we adopt in this paper, is to limit the sum of the lengths of each dissemination history in each version sequence. In particular, we use ABD-s to denote the ABD policy in which $\sum_{1\leq j\leq n}\sum_{v\in VS_j}|DH_j(v)|$ is limited to s [8]. s must be at least n.

4.1.2 The Benefit Estimation Method

When an update on D_i occurs, node i estimates the benefit of including its latest version of D_j in the broadcast message, for each D_j in the local database. Intuitively, i does so using the following procedure. For each node k compute the set of versions of D_j that k can have, i.e. the set of versions that were received at i after D_j^k was received. Assume that there are m such versions. Then, compute the set of broadcasts from which k could have learned each one of these versions. Based on this set compute the probabilities $q_1, q_2, ..., q_m$ that k has each one of the possible versions $v_1, v_2, ..., v_m$. Finally, compute the expected benefit to k as the sum $q_1 \cdot DIST(v(D_j), v_1) + q_2 \cdot DIST(v(D_j), v_2)$ +...+ $q_m \cdot DIST(v(D_j), v_m)$.

Formally, node i performs the benefit estimation in five steps:

(1) Construct an *effective version sequence* (EVS) of D_j^k which is a subsequence of VS_j:

$$\begin{aligned} EVS_j^k \;\; = \;\; & \{\, v|v \in VS_j \text{ and } v \geq v(D_j^k) \text{ and there exists} \\ & t \in DH_j(v) \text{ such that } t \geq t(D_j^k)\,\} \end{aligned} \quad (6)$$

Intuitively, EVS_j^k is the set of versions of D_j that k can have, as far as i knows. In other words, EVS_j^k contains each version v that satisfies the following two properties: (i) v is higher than or equal to the latest version of D_j that i has received from k (i.e. $v(D_j^k)$), and (ii) i has received at least one broadcast which includes $D_j(v)$, and that broadcast arrived later than D_j^k. For example, Figure 2(b) illustrates EVS_j^k for the example in Figure 2(a). We assume $t(D_j^k) = 15$ and $v(D_j^k) = 1$, i.e. the version of D_j^k is 1, and it was received at time 15. Notice that EVS_j^k is not necessarily a consecutive subsequence of VS_j^k. For example, version 4 is not in EVS_j^k because it was broadcast at time 12, i.e. before D_j^k. This means that k has not received

[8] $|A|$ denotes the size of the set A.

this broadcast, and thus, as far as i is concerned, 4 is not a possible current version number of D_j in k's local database.

(2) For each v in EVS_j^k that is higher than $v(D_j^k)$, count the *effective dissemination number* which is the size of the set $\{t|t \in DH_j(v)$ and $t > t(D_j^k)\}$, and denote this number $EDN_j^k(v)$. Intuitively, $EDN_j^k(v)$ is the number of broadcasts from which k could have learned $D_j(v)$, based on i's knowledge. Figure 2(b) illustrates each $EDN_j^k(v)$ which is derived from the example in Figure 2(a). Notice that $EDN_j^k(3) = 2$ because 10 was broadcast before D_j^k (which was broadcast at time 15), and thus k could not have received that broadcast (otherwise it would have broadcast a higher version number at time 15).

(3) For each v in EVS_j^k, compute η_v which, as we will prove, is the probability that the version number of D_j in k's local database is v. If $v = v(D_j^k)$,

$$\eta_v = \prod_{v' \in EVS_j^k \text{ and } v' > v} (1 - p_k)^{EDN_j^k(v')} \tag{7}$$

Otherwise,

$$\eta_v = (1-(1-p_k)^{EDN_j^k(v)}) \prod_{v' \in EVS_j^k \text{ and } v' > v} (1-p_k)^{EDN_j^k(v')} \tag{8}$$

(4) If the version number of D_j in k's local database is v, then the estimated benefit to k of including D_j^i in the broadcast message is taken to be the distance between $D_j(v)$ and D_j^i (i.e. $DIST(D_j(v), D_j^i)$). Denote this benefit $B(D_j^i, k, v)$.

(5) The estimated benefit to k of including D_j^k in the broadcast message is taken to be $p_k \cdot \sum_{v \in EVS_j^k}(\eta_v \cdot B(D_j^i, k, v))$. Denote this benefit by $B(D_j^i, k)$. Then the estimated benefit $B(D_j^i)$ of including D_j^i in the broadcast message is:

$$B(D_j^i) = \sum_{1 \le k \le n \text{ and } k \ne i,j} B(D_j^i, k) \tag{9}$$

4.2 Message Construction Step

The objective of this step is for node i to select a subset S of data items from the local database for inclusion in the broadcast message. The set S is chosen such that the expected net benefit of the message (i.e. the total expected inconsistency-reduction benefit minus the cost of the message) is maximized.

First, node i sorts the estimated benefits of the data items in descending order. Thus we have the benefit sequence $B(D_{k_1}^i) \ge B(D_{k_2}^i) \ge ... \ge B(D_{k_n}^i)$. Then i constructs the message as follows. If there is no number t between 1 and n such that the sum of the first t members in the sequence is bigger than $(C_1 + t \cdot C_2)$, then i will not broadcast a message.[9] Else, i finds the shortest prefix of the benefit sequence such that the sum of all the members in the prefix is greater than $(C_1 + m \cdot C_2)$, where m is the length of the prefix. i places the data items corresponding to the prefix in the broadcast message. Then i considers each member j that succeeds the prefix. If $B(D_j^i)$ is greater than or equal to C_2, then i puts D_j^i in the message. [10]

In section 5 we show that the procedure in this step broadcasts the subset S of data items whose net benefit is higher than that of any other subset.

This concludes the description of the ABD-s policy, which consists of the benefit estimation and message construction steps. It is easy to see that the time complexity of the policy is $O(n \cdot s)$.

5 Analysis of the ABD Algorithm

In this section we prove cost optimality of ABD based on the level of knowledge that node i has about the other nodes in the system. The following definitions are used in the analysis.

Definition 1 If at time t there is a broadcast from i which includes D_j, we say that a *dissemination of* D_j occurs at time t, and denote it $r_j(i, v, t)$ where v is the version number of D_j included in that broadcast. □

Definition 2 A *dissemination sequence of* D_j *at time* t is the sequence of all the disseminations of D_j that occurred from the beginning until time t:

$$RS_j(t) =< r_j(n_1, v_1, t_1), r_j(n_2, v_2, t_2), ...r_j(n_m, v_m, t_m) >$$

[9] remember that the cost of a message containing m data items is $(C_1 + m \cdot C_2)$.

[10] For the general case where the cost of a message is a non-decreasing function of the length of the message, i computes the net benefit of the first t members in the benefit sequence for each $1 \le t \le n$. If for all the values of t the net benefit is not greater than zero, then i will not broadcast a message. Else, i finds the t such that the net benefit is maximized and includes the first t data items the message.

where $t_1 < t_2 < ... < t_m \leq t$. □

Definition 3 Suppose k receives a message from i which includes D_j^i. Denote $\overline{D_j^k}$ the version of D_j in k's local database immediately before the broadcast. If the version of D_j^i is higher than the version of $\overline{D_j^k}$, then the *actual benefit to k of receiving* D_j^i, denoted $\overline{B}(D_j^i)$, is:

$$\overline{B}(D_j^i, k) = DIST(\overline{D_j^k}, \overline{D_j}) - DIST(D_j^i, \overline{D_j}) \quad (10)$$

Otherwise the actual benefit is 0. □

In other words, the actual benefit to k of receiving D_j^i is the reduction in the distance of D_j^k from $\overline{D_j}$. Observe that the actual benefit can be negative. For example, consider the case where D_j is a numeric value and $DIST(D(k), D(k')) = |D(k) - D(k')|$. If $\overline{D_j} = 300$, $D_j^i = 100$ and $\overline{D_j^k} = 200$, then $\overline{B}(D_j^i, k) = -100$.

Definition 4 The *actual benefit of dissemination* $r_j(i, v, t)$, denoted $\overline{B}(D_j^i)$, is the sum of the actual benefits to each node k that receives the message from i at t which included D_j^i. The *actual benefit of a broadcast message* is the sum of the actual benefits of each data item included in the message. □

Now we discuss two levels of knowledge of i about the other nodes in the system.

Definition 5 Node i is *absolutely reliable on D_j for node k by time t* if i has received all the broadcast messages which included D_j and were sent between $t(D_j^k)$ and t. i is *absolutely reliable on D_j by time t* if i is absolutely reliable on D_j for each node k by t. i is *absolutely reliable by time t* if i is absolutely reliable on each D_j by t. □

Definition 6 Node i is *strictly synchronized with D_j* at time t if at t D_j in i's local database is the latest version of D_j at t. i is *strictly synchronized at time t* if i is strictly synchronized with each D_j at t. □

Obviously, if i is strictly synchronized at time t, then i's local database is identical to the system state at t.

Observe that if each node j broadcasts D_j whenever an update on D_j occurs, then a node i which is absolutely reliable on D_j by time t is strictly synchronized with D_j at time t. However, in the ABD policy a node j may decide not to broadcast the new version of D_j, and thus i is not necessarily strictly synchronized with D_j even if i is absolutely reliable on D_j. On the other hand, i can be strictly synchronized even if it is not absolutely reliable. In other words, "absolutely reliable" and "strictly synchronized" are two independent properties.

Theorem 5 Let $RS_j(t)$ be a dissemination sequence of D_j in which the last dissemination is $r_j(i, v, t)$. The actual benefit of $r_j(i, v, t)$ (i.e. $\overline{B}(D_j^i)$) is a random variable. If i is absolutely reliable on D_j by t and strictly synchronized with D_j at t, then $B(D_j^i)$ given by the ABD policy(see Equality 9) is the expected value of $\overline{B}(D_j^i)$.

Proof idea: The proof of Theorem 5 is based on the following two lemmas.

Lemma 3 Let $RS_j(t)$ be a dissemination sequence of D_j in which the last dissemination is $r_j(i, v, t)$. If i is absolutely reliable on D_j by t, then for a node $k \neq i, j$, the version of D_j in k's local database at time t (i.e. $v(\overline{D_j^k})$) is a random variable. EVS_j^k gives the sample space of $v(\overline{D_j^k})$.

Lemma 4 For a node $k \neq i, j$, Equalities 7 and 8 give the probability that $v(\overline{D_j^k}) = v$. □

Now we devise a function which allows us to measure the cost efficiency of a broadcast.

Definition 7 The *actual net benefit* of a broadcast message is the difference between the actual benefit of the message and the cost of the message. Denote $\overline{NB}(M)$ the actual net benefit of broadcasting a set of data items M. □

Definition 8 A *broadcast sequence at time t* is the sequence of all the broadcasts in the system from the beginning (time 0) until time t:

$$BS(t) = < M(n_1, t_1), M(n_2, t_2), ..., M(n_m, t_m) > \quad (11)$$

where $M(n_l, t_l)$ is a message that is broadcast from n_l at time t_l, and $t_1 < t_2 < ... < t_m \leq t$. □

For a node which is both absolutely reliable by t and strictly synchronized at t, we have the following theorem concerning the optimality of the ABD policy.

Theorem 6 Let $BS(t)$ be a broadcast sequence in which the last broadcast is $M(i, t)$. The actual net benefit of broadcast $M(i, t)$ (i.e. $\overline{NB}(M(i, t))$) is a random variable. In particular, let $M = \{D_{k_1}^i, D_{k_2}^i, ..., D_{k_m}^i\}$ be the set of data items broadcast by the ABD policy at time t. If i is absolutely reliable by t and strictly synchronized at t, then:

(1) $E[\overline{NB}(M)] \geq 0$

(2) For any M' which is a subset of k's local database, $E[\overline{NB}(M')] \leq E[\overline{NB}(M)]$.

Proof idea: The proof of Theorem 6 is based on Theorem 5 and the following lemma.

Lemma 5 Let $\{D^i_{k_1}, D^i_{k_2}, ..., D^i_{k_m}\}$ be the message constructed by the message construction method, then

(1) The estimated benefit of broadcasting M is not lower than the cost of M.

(2) For any subset M' of i's local database, the estimated net benefit of broadcasting M' is not higher than that of broadcasting M. □

Theorem 6 shows that the message broadcast by the ABD policy is optimized because the expected net benefit of broadcasting any subset of i's local database is not higher than that of broadcasting this message. Granted, this theorem holds under the assumption of strict synchronization and absolute reliability, but i can base its decision only on the information it knows.

In some cases, Theorems 5 and 6 hold for a node which is not strictly synchronized.

Consider a data item D_i which is a single numeric value that monotonously increases as the version number of D_i increases. We call this a *monotonous data item.* Assume that the distance function is:

$$DIST(D_i(k), D_i(k')) = |D_i(k) - D_i(k')| \qquad (12)$$

We call this the *absolute distance function*.

For monotonous data items and absolute distance functions, Theorems 5 and 6 are true when i is absolutely reliable but not necessarily strictly synchronized at t. Thus we have the following two theorems.

Theorem 7 Let $RS_j(t)$ be a dissemination sequence where the last dissemination is $r_j(i, v, t)$. The actual benefit of $r_j(i, v, t)$ (i.e. $\overline{B}(D^i_j)$) is a random variable. For monotonous data items and absolute distance functions, if i is absolutely reliable on D_j by t, then $B(D^i_j)$ given by the ABD policy (see Equality 9) is the expected value of $\overline{B}(D^i_j)$.□

Theorem 8 Let $BS(t)$ be a broadcast sequence, where the last broadcast is $M(i, t)$. The actual net benefit of broadcast $M(i, t)$ (i.e. $\overline{NB}(M(i, t))$) is a random variable. In particular, let $M = \{D^i_{k_1}, D^i_{k_2}, ..., D^i_{k_m}\}$ be the message broadcast by the ABD policy at time t. For monotonous data items and absolute distance functions, if i is absolutely reliable by t, then:

(1) $E[\overline{NB}(M)] > 0$

(2) For any M' which is a subset of k's local database, $E[\overline{NB}(M')] \leq E[\overline{NB}(M)]$. □

6 Comparison of the Policies by Simulation

In this section we describe the experiments that we conducted in order to evaluate the three broadcast policies discussed in the previous sections, namely ABD, FBD, and SBD. To briefly recap, the policies behave as follows. In SBD a node i broadcasts the new value of D_i whenever it is updated. In FBD i broadcasts its whole local database whenever D_i is updated. In ABD i broadcasts a subset of the local database (as described in the previous section) whenever D_i is updated. We compared the above policies with traditional flooding (FLD), a conventional protocol for data dissemination [40]. In FLD, a node that receives or generates a new version of a data item, rebroadcasts a copy of the data item to all the other nodes. In contrast to ABD, FBD, and SBD, in the FLD policy a node i broadcasts the new version even if it is for a data item D_j other than D_i. In this case the new version must have been received from a message that provided a new value for D_j in i's local database. FLD is similar to SBD in the sense that each node broadcasts a single data item in each message. FLD is similar to FBD and ABD in the sense that a node i may broadcast a data item which is different than D_i.

In this section we first discuss the simulation method and then describe the simulation results.

6.1 Simulation Method

In this subsection we first describe the inconsistency cost functions used in our experiments. Then we discuss the extra storage and computation cost incurred by ABD compared to the other policies, and how this is taken into consideration in our experiments. Then we discuss the method we used in order to carry out each simulation run. Finally we describe, plot, and discuss the results of our simulations.

6.1.1 Inconsistency Cost Functions

First we discuss the two distance functions used for measuring inconsistency. One is version-based, namely the inconsistency between two versions $D_i(k)$ and $D_i(k')$ is the difference between the version numbers, i.e. $DIST(D_i(k), D_i(k')) = |k - k'|$. We call this the *version-based* distance function. The other distance function is value-based, namely the distance between two versions $D_i(k)$ and $D_i(k')$ is the difference between the values (we assume D_i contains a single numeric value), i.e. $DIST(D_i(k), D_i(k')) = |D_i(k) - D_i(k')|$. We call this the *value-based* distance function. Each time a data item is updated, we randomly select a real number between 0 and 100 as the value for the new version of that data item. This way the two distance functions represent two extreme patterns in which data changes over time: the version-based distance function represents a very regular pattern in which data is increased by one unit on each update, whereas the value-based distance function represents a very chaotic pattern in which data changes randomly. For space considerations the plots in our figures refer only to the version based distance function. However, the results for the value based function are qualitatively similar.

Parameter	Symbol	Value	
Number of nodes	n	20	
Maximum data item value		100	
Update intensity of node i	λ_i	Randomly selected from [0.00001,0.1] and fixed for all the simulations	
Policy		SBD, FBD, ABD, FLD	
Connection probability lower bound	*cplb*	0.1 - 1	
CPU factor	C_3	0.1	
Storage unit cost	C_4	0.0001	
Information cost model		Version-based	Value-based
Message initiation cost	C_1	1 - 20	1 - 1000
Message unit cost	C_2	0.01 - 10	0.01 - 500

Table 1: Parameter settings

6.1.2 Extra Resource Cost Incurred by the ABD policy

Observe that ABD performs additional computation for each message that it broadcasts (to determine the set of data items that will be broadcast) compared with the other policies. It also uses additional storage for the data structures that it maintains.

The additional computation incurred by ABD is modeled by the *CPU factor* denoted C_3. The CPU factor is a fraction added to the message initiation cost C_1 of ABD. For example, if in a simulation run $C_1 = 10$ and $C_3 = 0.2$, then ABD's message initiation cost is 12, rather than 10 for the other policies.

A node using the ABD policy also pays additional storage cost compared to the other policies. This depends on both, the size of the data structure it maintains and the length of time for which the data structure is kept. We use C_4 to denote the cost of a unit of storage occupied for one time unit and call it the *storage unit cost*. C_4 can be determined by the number of storage units that one is willing to maintain for one time unit, in order to reduce the inconsistency cost on a version by one unit. Formally, for ABD-s (recall that s is the size of data structures) the extra storage cost of i up to time t is $C_4 \cdot s \cdot t$.

Note that C_3 and C_4 are introduced for the sole purpose of comparing the system cost of ABD with that of the other policies. In contrast to the communication cost and the inconsistency cost, they have no impact on the execution of ABD. The reason for this is that the storage and extra computation expended by ABD is fixed, independently of the set of items that are actually included in the broadcast message.

6.1.3 Execution of a simulation run

Now we describe how we conduct each simulation run. We take the number of nodes in the system to be 20. Each simulation run uses one of the following five policies as the broadcast policy: SBD, FBD, FLD, ABD-400 and ABD-800. In ABD-400, for each data item D_j and each node k, node i keeps only the latest time point at which i received D_j from k (for $k \neq i$) or at which i broadcast D_j (for $k = i$). This way, the sum of the lengths of each dissemination history at node i is limited to 20x20x1 $= 400$. Similarly, ABD-800 limits the sum of the lengths of each dissemination history to 20x20x2 $= 800$ by keeping only the latest two time points at which i received D_j from a node k. We set up a connection probability lower bound called $cplb$. ($0 \leq cplb \leq 1$). For each node i, the connection probability p_i is randomly chosen from the interval $[cplb, 1]$. Intuitively, the p_i's increase as the $cplb$ parameter increases, and therefore the connection reliability increases. $cplb$ is a parameter of each simulation run. For each node i, the updates at i are generated by a Poisson process with intensity λ_i. This means that on average i generates λ_i updates per time unit. Each λ_i is randomly selected from the interval $[0.00001, 0.1]$. We select λ_i only once and keep it fixed for all the simulations. For each set of parameters we execute a simulation run for 10000 logical time units, which on average introduces 500 updates to each node. All the parameters and their value ranges are summarized in Table 1.

Each simulation run is executed as follows. In each time unit, updates are generated and all the nodes are processed in sequence. When a new update occurs on D_i, the total inconsistency on D_i is increased for each node that did not receive the previous update. Node i uses the policy of the simulation run to construct a message. i "broadcasts" the message and each other node j "receives" it with probability p_j and updates its local database accordingly. A message sent by a node in time slot i of a round is accounted for in the same round by all the nodes with higher slots. The nodes with lower slots will account for this message in the next round. The total resource cost is increased by the cost of this message.

6.2 Simulation Results

In this subsection we present the results of the comparison among the four policies SBD, FBD, FLD and ABD-800. We compare the policies in terms of their system cost, inconsistency cost, and resource cost. The resource cost of ABD-s is the sum of the communication cost (which includes the extra CPU cost) and the storage cost. The resource cost of SBD, FBD, FLD is simply the communication cost. The system cost is the sum of the inconsistency cost and the resource cost. In appendix B we discuss the comparison between ABD-800 and ABD-400; it quantifies the system cost reduction that results from storing an extra version for each data item.

The basic conclusion from these experiments is that for most parameter combinations of Table 1 ABD is superior to the other policies. Clearly, as the CPU and storage (of which ABD uses more than the other policies) unit costs increase, a crossover will occur. Thus, in appendix C we also describe experiments that quantify how the system cost of ABD changes as a result of storage and CPU unit costs, and for which such unit costs ABD becomes inferior to other policies.

Some of the simulation results are given below. We conducted many more simulation runs, but the results are omitted for space considerations. However, the omitted results confirm our basic conclusions.

Figure 3: System cost as a function of cplb (c1=1,c2=0.1)

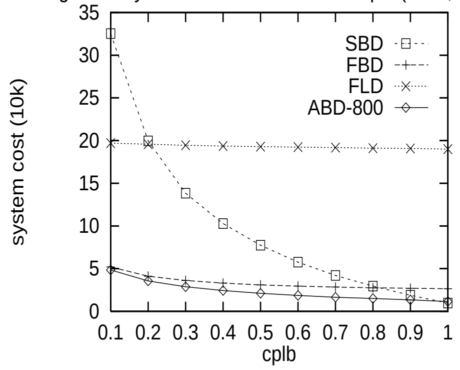
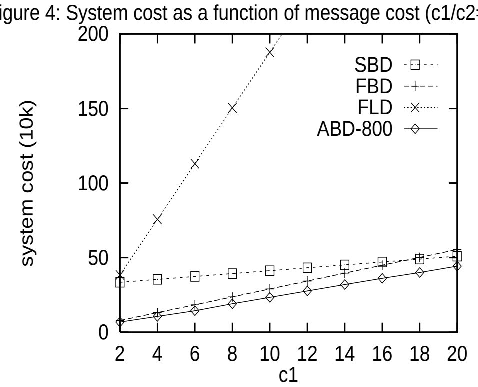


Figure 4: System cost as a function of message cost (c1/c2=10,cplb=0.1)



First we discuss the system cost as a function of the connection probability lower bound $cplb$ (Figure 3), and then we discuss the system cost as a function of the message initiation cost C_1 (Figure 4).

System cost as a function of cplb: Figure 3 plots the system costs of the four policies as a function of $cplb$ (ranging from 0.1 to 1), with the message initiation cost $C_1 = 10$, and the message unit cost $C_2 = 0.1$. We conducted similar experiments with C_1 ranging from 1 to 20 and C_2 from 0.01 to 10 (see Table 1).

Observe that SBD has the highest cost for a low $cplb$, but the difference between the system cost of SBD and ABD decreases as $cplb$ increases (Figure 3). The reason for this is that, clearly, since SBD broadcasts each update only once it pays a higher inconsistency cost as $cplb$ decreases.

System cost as a function of the message cost: Figure 4 plots the system costs of the four policies as a function of the message initiation cost $C1$ (ranging from 2 to 20). We did similar experiments with $cplb$ ranging from 0.1 to 1. The conclusion is that ABD has the lowest system cost and FLD has the highest one, with the gap increasing as the message cost increases.

7 Relevant Work

The problem of data dissemination in peer to peer broadcast networks has not been analyzed previously as far as we know. The data broadcasting problem studied in [5, 24, 43] is how to organize the broadcast and the cache in order to reduce the response time. The above works assume a centralized system with a single server and multiple clients communicating over a reliable network with large bandwidth. In contrast, in our environment these assumptions about the network do not always hold, and the environment is totally distributed and each node is both a client and a server.

Pagani et al. ([22]) proposed a reliable broadcast protocol which provides an exactly once message delivery semantics and tolerates host mobility and communication failures. Birman et al. ([4]) proposed three multicast protocols for transmitting a message reliably from a sender process to some set of destination processes. Unlike these works, we consider a "best effort" reliability model and allow copies to diverge.

Lazy replication by gossiping has been extensively investigated in the past (see for example [12, 2, 20]). Epidemic algorithms ([34, 36]) such as the one used in Grapevine ([35]) also propagate updates by gossiping. However, there are two major differences between our work and the existing works. First, none of these works considered the cost of communication; this cost is important in the types of novel applications considered in this paper. Second, we consider the tradeoff between communication and inconsistency, whereas the existing works do not. Alonso, Barbara, and Garcia-Molina ([1]) studied the tradeoff between the gains in query response time obtained from quasi-caching, and the cost of checking coherency conditions. However, they assumed point to point communication and a centralized (rather than a distributed) environment.

Achieving global consistency by gossip messages is also used in distributed systems research (see [3, 14, 7, 19]. A typical mechanism is vector clocks, used for example by Birman ([3]) in order to implement causal delivery. As explained in the introduction, the main difference between these works and the present paper is the consistency model. This leads to other differences. First, the communication cost is not considered in vector clock works. Second, in such works a node is not selective about what clocks are piggybacked in a message (similarly to the FBR policy, all clocks are piggybacked). Third, the piggybacked information is meta data (timestamps), whereas in our present work the piggybacked information are data items.

A recent work similar to ours is TRAPP (see [15]). The similarity is in the objective of quantifying the tradeoff between consistency and performance. However, the main differences are in the basic assumptions. First, the TRAPP system deals with numeric data in traditional relational databases. Second, it quantifies the tradeoff for aggregation queries. Actually, probably the most fundamental difference is that it deals with the problem of answering a particular instantaneous query, whereas we deal with database consistency. Specifically, we want the consistency of the whole database to be maximized for as long as possible. In other words, we maximize consistency in response to continuous queries that retrieve the whole database.

Another research area to which this paper is related is disconnection management in mobile environments (see for example [11, 8, 9, 10]). However, these works assume *planned disconnection*, i.e. a node always informs the system of its intention to disconnect or reconnect. In other words, at any point in time the system is aware of which nodes are connected and which ones are not. Planned disconnection requires that a node has complete control of when to connect and when to disconnect. But in practice this is not always possible. A node may run out of battery or drive under a bridge and lose connection unexpectedly. Furthermore, planned disconnection is not realistic in peer-to-peer networks, since a node i may miss the disconnection notice from another node j if i itself is disconnected.

Finally, let us discuss a large body of important work dealing with replication, consistency, and broadcasting (see for example [44, 45]). These works are concerned with transactional properties and attaining serializability, i.e. perfect consistency, at minimum cost. In contrast, in this paper we consider applications where inconsistency can be tolerated and transactional properties are not strictly required. However, a framework in which each update is a transaction can be easily incorporated in our model.

8 Conclusion

In this paper we studied data dissemination in peer to peer broadcast networks. The problem is introduced by novel applications and networks such as Mobile Ad-hoc Networks, sensor and "smart dust" networks, and satellite networks. Each node i "owns" the master copy of a data item D_i, i.e. it generates all the updates to D_i (see [23]). Each update is broadcast to the other nodes. The database of interest is $D = \{D_1,...,D_n\}$, where n is the number of nodes. A version of this database is stored at each node.

We introduced a cost model for quantifying the tradeoff between inconsistency and communication. The inconsistency cost is captured via the notion of the distance between two versions of a data item D_i. The communication cost is captured via the notion of a message cost which is proportional to the length of the message. Then we used the model to first compare two data broadcast policies: (1) eager dissemination by Reliable Broadcast (RBD) which keeps the databases at each node consistent, and (2) lazy dissemination by Single-item Broadcast Dissemination (SBD) which allows inconsistency, but incurs a lower communication cost due to unreliable broadcast. We completely characterized the parameters for which SBD is superior to RBD, and vice versa. Intuitively, lazy dissemination incurs a lower total cost in low connectivity environments, and in environments in which the communication cost is high. This is not surprising, but the contribution of the analysis is in the quantifiable characterization.

Then we used our cost model for exploring lazy dissemination alternatives to SBD. In particular, we introduced the Adaptive Broadcast Dissemination (ABD) policy. In this policy, when a node i receives an update to D_i, it first estimates the expected benefit to the system of including in the broadcast message each data item D_j in i's local database. Then i constructs and broadcasts a message which consists of a subset of i's local database. The subset is chosen such that the "net" benefit of the message is maximized, i.e. the inconsistency-cost-reduction minus the message-cost is maximized. We showed that ABD is optimal for the level of knowledge that each node has about the distributed system. Optimality is in the sense that if ABD were to broadcast a different subset of data items, then the expected cost would be higher.

We compared the ABD policy with three other naive lazy dissemination policies, SBD, Full Broadcast Dissemination (FBD), and flooding (FLD). The first two policies broadcast a message only when the master copy of a data item i is updated. In SBD node i broadcasts only the updated data item, whereas in FBD i broadcasts the entire local database. The FLD policy on the other hand, broadcasts a message whenever some node i receives a new version of a data item; the new version may be of D_i, or of another data-item (in this latter case the update must have been received by a broadcast message from another node). We compared by simulation the four policies for a large number of parameters combinations. If the cost of the extra computation and storage used by ABD is reasonably small (see Table 1), then ABD consistently outperforms the other two policies, often having a total cost that is several times lower than that of the other policies. Otherwise, appendix B characterizes the cases where ABD becomes inferior.

References

[1] R. Alonso, D. Barbara, and H. Garcia-Molina, *Data Caching Issues in an Information Retrieval System*, ACM Transactions on Database Systems, Vol. 15, No. 3, Sept. 1990.

[2] R. Ladin, B. Liskov, S. Ghemawat, *Providing High Availability Using Lazy Replication*, ACM Transactions on Computer Systems, Vol. 10, No. 4, November 1992.

[3] K. Birman, A. Schiper, P. Stephenson, *Lightweight Causal and Atomic Group Multicast*, ACM Transactions on Computer Systems, Vol. 9, No. 3, August 1991.

[4] K. Birman, T. A. Joseph, *Reliable Communication in the Presence of Failures*, ACM Transactions on Computer Systems, Vol. 5, No. 1, Feb. 1987.

[5] T. Imielinski, B. Badrinath, *Mobile wireless computing: challenges in data management*, CACM, 37(10), October, 1994.

[6] L. G. Brown, *A Survey of Image Registration Techniques*, ACM Computing Surveys, 24(4):325-376, December 1992.

[7] C. Fidge, *Timestamps in message-passing systems that preserve the partial ordering*, in Proceedings of the 11th Australian Computer Science Conference, 1988.

R. Ladin, B. Liskov, S. Ghemawat, *Providing High Availability Using Lazy Replication*, ACM Transactions on Computer Systems, Vol. 10, No. 4, November 1992.

[8] J. Holliday, D. Agrawal, A. E. Abbadi, *Planned disconnections for mobile databases*, Proceedings of the 11th International IEEE Workshop on Database and Expert Systems (DEXA 2000).

[9] J. Holliday, D. Agrawal, A. E. Abbadi, *Exploiting planed disconnections in mobile environments*, Proccedings of the 10th IEEE Workshop on Research Issues in Data Engineering (RIDE2000), pp 25-29, Feb. 2000.

[10] R. Kravets, P. Krishnan, *Power management techniques for mobile communications*, MOBICOM'98, Dallas , TX, 1998.

[11] P. Keleher, *Decentralized replicated-object protocols*, Proceedings of the 18th ACM Symposium on Principles of Distributed Computing, Apr. 1999.

[12] B. Liskov, R. Scheifler, E. Walker, and W. Weihl, *Orphan detection (extended abstract)*, Proceedings of the 17th International Symposium on Fault-Tolerant Computing, July 1987.

[13] J. P. Macker and M. S. Corson, *Mobile Ad Hoc Networking and the IETF*, Mobile Computing and Communications Review, Vol. 2, No. 1, January 1998.

[14] F. Mattern, Virtual time and global states of distributed systems, in the Proceedings of the International Workshop on Parallel and Distributed Algorithms, North-Holland, 1989.

[15] C. Olston, J. Widom, *Offering a precision-performance tradeoff for aggregation queries over replicated data*, http://www-db.stanford.edu/pub/papers/trapp-ag.ps.

[16] W. Stallings, *Data and Computer Communications*, 5th ed., Prentice Hall, 1997.

[17] G. Kollios, D. Gunopulos and V. J. Tsotras, *On indexing mobile objects*, in Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, 1999, Philadelphia, PA.

[18] G. Kollios, D. Gunopulos, and V. J. Tsotras, *Nearest Neighbor Queries in a Mobile Environment*, Workshop on Spatio-Temporal Database Management, 1999, Edinburgh, Scotland.

[19] A. Schiper, J. Eggli, and A. Sandoz, *A new algorithm to implement causal ordering*, in the Proceedings of the 3rd International Workshop on Distributed Algorithms, Lecture Notes on Computer Science 392, Springer-Verlag, New York, 1989.

[20] D. B. Terry, K. Petersen, M. J. Spreitzer and M. M. Theimer, *The Case for Non-transparent Replication: Examples from Bayou*, Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, Vol. 21, No.4, 1998.

[21] M. J. Spreitzer, M. M. Theimer, K. Petersen, A. J. Demers and D. B. Terry, *Dealing with Server Corruption in Weakly Consistent, Replicated Data Systems*, Proc. ACM MOBICOM'97, pp. 234-240, Budapest, Hungary, 1997.

[22] E. Pagani and G. P. Rossi, *Reliable Broadcast in Mobile Multihop Packet Networks*, Proc. ACM MOBICOM'97, pp. 34-42, Budapest, Hungary, 1997.

[23] J. Gray, P. Helland, P. O'Neil, D. Shasha, *The dangers of replication and a solution*, Proc. ACM SIGMOD 96, pp. 173-182, Montreal, Canada, 1996.

[24] S. Jiang, N. H. Vaidya, *Scheduling data broadcast to "impatient" users*, Proceedings of ACM International Workshop on Data Engineering for Wireless and Mobile Access, Seattle, Washington, August 1999.

[25] M. Altinel, D. Aksoy, T. Baby, and M. Franklin, *DBIS-Toolkit: adaptable middleware for large*

scale data delivery, in Proceedings of the 1999 ACM SIGMOD, Philadelphia, PA, 1999.

[26] G. Grimmett, D. Welsh, *Probability: an introduction*, Clarendon Press, 1986.

[27] J. M. Kahn, R. H. Katz and K. S. J. Pister, *Next century challenges: mobile networking for "Smart Dust"*, Proceedings of the fifth ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM99), Seattle, WA, August, 1999.

[28] W. R. Heinzelman, J. Kulik and H. Balakrishnan, *Adaptive protocols for information dissemination in wireless sensor networks*, Proceedings of the fifth ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM99), Seattle, WA, August, 1999.

[29] P. Sistla, O. Wolfson, S. Chamberlain, S. Dao, *Modeling and Querying Moving Objects*, Proceedings of the Thirteenth International Conference on Data Engineering (ICDE13), Birmingham, UK, Apr. 1997.

[30] O. Wolfson, S. Chamberlain, S. Dao, L. Jiang, G. Mendez, *Cost and Imprecision in Modeling the Position of Moving Objects* to appear, Proceedings of the Fourteenth International Conference on Data Engineering (ICDE14), 1998

[31] O. Wolfson, B. Xu, S. Chamberlain, L. Jiang, Moving Objects Databases: Issues and Solutions, Proceedings of the 10th International Conference on Scientific and Statistical Database Management (SSDBM98), Capri, Italy, July 1-3, 1998, pp. 111-122.

[32] F. Bennett, D. Clarke, J. Evans, A. Hopper, A. Jones, and D. Leask, *Piconet: Embedded Mobile Networking*, IEEE Personal Communications, 4(5), October 1997.

[33] Z. J. Haas, *Panel Report on Ad Hoc Networks - MILCOM'97*, Mobile Computing and Communications Review, Vol. 2, No. 1, January 1998.

[34] A. Demers, D. Greene, etc., *Epidemic algorithms for replicated database maintenance*, Operating Systems Review, vol. 22, No. 1, pp. 8-32, Jan. 1988.

[35] M. D. Schroeder, A. D. Birrell, R. M. Needham, *Experience with Grapevine: the growth of a distributed system*, ACM Transactions on Computer Systems, vol. 2, No. 1, pp. 3-23, Feb. 1984.

[36] R. Golding, *A weak-consistency architecture for distributed information services*, Computing Systems, vol. 5, No. 4, 1992. Usenix Association.

[37] P. K. Agarwal, L. Arge, J. Erickson, *Indexing moving points*, to appear in ACM PODS'2000.

[38] S. Chamberlain, *Automated Information Distribution in Bandwidth-Constrained Environments* MILCOM-94 conference, 1994.

[39] S. Chamberlain, *Model-Based Battle Command: A Paradigm Whose Time Has Come,* 1995 Symposium on C2 Research & Technology, NDU, June 1995

[40] A. S. Tanenbaum, *Computer networks*, Prentice Hall, 1996.

[41] D. Agrawal, G. Alonso, A. El Abbadi, and I. Stanoi, *Exploiting atomic broadcast in replicated databases*, in Proceedings of the International Conference on EuroPar'97, 1997.

[42] S. Acharya, M. Franklin, S. Zdonik, *Balancing push and pull for data broadcast*, Proc. ACM SIGMOD 97, pp. 183-194, Tucson, Arizona, 1997.

[43] S. Acharya, M. Franklin, and S. Zdonik, *Prefetching from a broadcast disk*, in 12th International Conference on Data Engineering, Feb. 1996.

[44] Y. Breitbart and H. F. Korth, *Replication and consistency in a distributed environment*, Journal of Computer and System Sciences, Vol 59, No. 1, Aug. 1999.

[45] Y. Breitbart, R. Komondoor, R. Rastogi, S. Seshadri, *Update propagation protocols for replicated databases*, in Proceedings of ACM SIGMOD'99, Philadelphia, PA, June 1999.

Appendix A: Proofs of the lemmas and theorems

Proof of Theorem 1: Before we derive the system cost of SBD, let us introduce a lemma which gives the inconsistency cost of SBD up to a given point in time.

Lemma 2 The inconsistency cost of SBD on D_i up to time t (denoted $COST\_INCO^i(t)$) is a random variable whose expected value is

$$
\begin{aligned}
&E[COST\_INCO^i(t)] \\
&= \sum_{m=1}^{\infty} \frac{e^{-\lambda_i \cdot t} \cdot (\lambda_i \cdot t)^m}{m!} \cdot \sum_{1 \le q \le m-1} \\
&\quad \sum_{1 \le j \le n \text{ and } j \ne i} ((1-p_j)^q \cdot DIST(D_i(0), D_i(q)) \\
&\quad + \sum_{k=1}^{q-1} p_j \cdot (1-p_j)^{q-k} \cdot DIST(D_i(k), D_i(q))) \qquad (13)
\end{aligned}
$$

Proof of Lemma 2: In the above equation, $\frac{e^{-\lambda_i \cdot t} \cdot (\lambda_i \cdot t)^m}{m!}$ is the probability that exactly m updates have occurred on D_i up to t by a Poisson process with intensity λ_i [26]. Now compute the expected inconsistency cost of the system up to the mth update of D_i. Consider the expected inconsistency cost of each node j ($j \neq i$) on each version $D_i(q)$, for $q < m$. There are three cases in terms of the highest version of D_i in j's local database before the $(q+1)$st update of D_i.

(1) j did not receive any one of the first q updates from i before the $(q+1)$st update of D_i. The probability of this case is $(1-p_j)^q$, and the inconsistency cost of j on $D_i(q)$ in this case is $DIST(D_i(0), D_i(q))$.

(2) j received $D_i(k)$ ($k < q$) but did not receive any version higher than $D_i(k)$ before the $(q+1)$st update. The probability of this case is $p_j \cdot (1-p_j)^{q-k}$, and the inconsistency cost of j on $D_i(q)$ in this case is $DIST(D_i(k), D_i(q))$.

(3) j received $D_i(q)$. The probability of this case is p_j and the inconsistency cost of j on $D_i(q)$ in this case is 0.

Thus $(1-p_j)^q \cdot DIST(D_i(0), D_i(q)) + \sum_{k=1}^{q-1} p_j \cdot (1-p_j)^{q-k} \cdot DIST(D_i(k), D_i(q))$ is the expected inconsistency cost of j on $D_i(q)$. Summing up for j and q, we get the expected inconsistency cost of the system up to the mth update of D_i.□

Now we prove Theorem 1. The expected message cost up to time t is $\lambda \cdot t \cdot (C_1 + C_2)$. The expected inconsistency cost up to t is the sum of the expected inconsistency cost on each D_i up to t, which was derived in Lemma 2 .□

Proof sketch of Lemma 1: By definition of the expected value, each term in the sum is k multiplied by the probability that the number of transmissions of this message is exactly k. This probability is the difference between two multiplications. The first one is the probability that all the nodes acknowledged at least one of the first k transmissions of the message, and the second multiplication is the probability that all the nodes acknowledged at least one of the first $k-1$ transmissions of the message. The difference between these two probabilities is the probability that exactly k transmissions of the message are necessary in order for all the other nodes to receive i's message.□

Proof of Theorem 2: The system cost of RBD is the sum of two costs. The first one is the cost of message transmissions, which is $C_1 + C_2$ times the expected number of messages transmitted by each node i up to time t. The second one is the cost of acknowledgements. Observe that for each update there are exactly $n-1$ acknowledgements which are $n-1$ messages of cost C_1 each.□

Proof of Theorem 3: Denote

$A = t \cdot \sum_{i=1}^{n} (\lambda_i \cdot E[R_i])$

$B = \lambda \cdot t$

$D = \sum_{i=1}^{n} E[COST\_INCO^i(t)]$

Then the difference between the expected costs of the two policies is:

$$
\begin{aligned}
&E[COST\_SYS_{RBD}(t)] - E[COST\_SYS_{SBD}(t)] \\
&= A \cdot (C_1 + C_2) + B \cdot (n-1) \cdot C_2 - (B \cdot (C_1 + C_2) + D) \\
&= (A + B \cdot (n-1) - B) \cdot C_1 + (A - B) \cdot C_2 - D
\end{aligned}
$$

Note that if the connectivity is perfect, i.e. each p_j is 1, then RBD is identical to SBD because in this case both RBD and SBD broadcast each message exactly once. Therefore, we will assume that at least one p_j is smaller than 1. Observe that in this case, for each node i ($i \neq j$) $R_i > 1$. Therefore $A > B$. By a straightforward mathematical manipulation it can be seen that $E[COST\_SYS_{RBD}(t)] - E[COST\_SYS_{SBD}(t)] > 0$ if and only if inequality 4 holds.□

Proof sketch of Theorem 4: The proof is straightforward from the following three observations:

1. $(1-p_j)^q + \sum_{1 \le k \le q} (p_j \cdot (1-p_j)^{q-k}) = 1$
2. $\sum_{m=1}^{\infty} (\frac{e^{-\lambda_i \cdot t} \cdot (\lambda_i \cdot t)^m}{m!} \cdot m) = \lambda_i \cdot t$

3. $\sum_{m=1}^{\infty}(\frac{e^{-\lambda_i \cdot t}\cdot(\lambda_i \cdot t)^m}{m!}) = 1 - e^{-\lambda_i \cdot t}$ □

Proof of Lemma 3: Observe first that because i has received every broadcast which included D_j since i received D_j^k from k, any version number which is not in VS_j is not a possible version number of $\overline{D_j^k}$. Now we explain that each version number v in $VS_j - EVS_j^k$ is not a possible version number of $\overline{D_j^k}$ and therefore is not a point in the sample space of $v(\overline{D_j^k})$.

Since $v \in VS_j - EVS_j^k$, v satisfies at least one of the following two properties:

(i) v is lower than $v(D_j^k)$.

(ii) All the broadcasts of $D_j(v)$ arrived before D_j^k.

If v satisfies property (i), then v is not a possible version number of $\overline{D_j^k}$ because i received D_j^k from k and therefore $v(D_j^k)$ is the lowest possible version number of $\overline{D_j^k}$. Now consider a version v which satisfies property (ii). Notice that the transmission time interval of any broadcast of $D_j(v)$ can not overlap with that of the broadcast which included D_j^k. If k had received any broadcast of $D_j(v)$, then k should have received it before k broadcast D_j^k. In that case, k would not broadcast D_j^k which is older than $D_j(v)$. But k broadcast D_j^k. This indicates that k did not receive any broadcast which included $D_j(v)$. Therefore, v is not a possible version number of $\overline{D_j^k}$.

Now we explain that each version number v in EVS_j^k is a possible version number of $\overline{D_j^k}$ and therefore is a point of the sample space of $v(\overline{D_j^k})$.

Consider a version number v which satisfies the following two properties:

(i) v is higher than the latest version of D_j that i has received from k (i.e. $v(D_j^k)$).

(ii) i has received at least one broadcast which included $D_j(v)$ and that broadcast arrived after D_j^k.

For any broadcast which included $D_j(v)$ and arrived after D_j^k, that broadcast was broadcast after k broadcast D_j^k. If k received that broadcast, since v is higher than $v(D_j^k)$, k would update D_j^k with $D_j(v)$. Therefore v is a possible version number of $\overline{D_j^k}$.

D_j^k is also a possible version number of $\overline{D_j^k}$. In summary, each version number v in EVS_j^k is a possible version number of $\overline{D_j^k}$ and therefore is a point in the sample space of $v(\overline{D_j^k})$. □

Proof of Lemma 4: Prove in two cases:

(1) $v = v(D_j^k)$

Consider each element v' in EVS_j^k that is higher than $v(D_j^k)$. Observe that $EDN_j^k(v')$ is the number of broadcasts from which k had chance to know $D_j(v')$. The probability that $v(\overline{D_j^k}) = v(D_j^k)$ is the probability that k did not receive any broadcast which included any $D_j(v')$, i.e.

$$\begin{aligned} P\{v(\overline{D_j^k}) = v\} &= \prod_{v' \in EVS_j^k}^{v'>v} (1-p_k)^{EDN_j^k(v)} \\ &= \eta_v \end{aligned} \tag{14}$$

(2) $v \neq v(D_j^k)$

The probability that k received at least one of the broadcasts that included $D_j(v)$ is $1-(1-p_k)^{EDN_j^k(v)}$. The probability that k did not receive any broadcast which included any $D_j(v')$ ($v' \in EVS_j^k$ and $v' > v$) is $\prod_{v' \in EVS_j^k \text{ and } v'>v}(1-p_k)^{EDN_j^k(v')}$. Thus the probability that the version of $\overline{D_j^k}$ is v is:

$$\begin{aligned} & P\{v(\overline{D_j^k}) = v\} \\ &= (1-(1-p_k)^{EDN_j^k(v)}) \prod_{v' \in EVS_j^k}^{v'>v} (1-p_k)^{EDN_j^k(v')} \\ &= \eta_v \end{aligned} \tag{15}$$

□

Proof of Theorem 5: Assume $v(\overline{D_j^k}) = v$ ($v \in EVS_j^k$). If k receives the broadcast, $\overline{B}(D_j^i, k) = DIST(D_j(v), \overline{D_j}) - DIST(D_j^i, \overline{D_j})$; otherwise $\overline{B}(D_j^i, k) = 0$. The expected value of $\overline{B}(D_j^i, k)$ on the condition that $v(\overline{D_j^k}) = v$ is:

$$\begin{aligned} & E[\overline{B}(D_j^i, k) | v(\overline{D_j^k}) = v] \\ &= p_k \cdot (DIST(D_j(v), \overline{D_j}) - DIST(D_j^i, \overline{D_j})) \quad (16) \\ &= p_k \cdot DIST(D_j(v), D_j^i) \\ &= p_k \cdot B(D_j^i, k, v) \quad (17) \end{aligned}$$

where we have used the assumption that $D_j^i = \overline{D_j}$ and therefore $DIST(D_j^i, \overline{D_j}) = 0$.

Hence

$$\begin{aligned} & E[\overline{B}(D_j^i, k)] \\ &= \sum_{v \in EVS_j^k} E[\overline{B}(D_j^i, k) | v(\overline{D_j^k}) = v] \cdot P\{v(\overline{D_j^k}) = v\} \\ &= p_k \cdot \sum_{v \in EVS_j^k} (\eta_v \cdot B(D_j^i, k, v)) \\ &= B(D_j^i, k) \end{aligned}$$

Since $\overline{B}(D^i_j) = \sum_{k\neq i,j} \overline{B}(D^i_j, k)$,

$$\begin{aligned} E[\overline{B}(D^i_j)] &= \sum_{k\neq i,j} E[\overline{B}(D^i_j, k)] \\ &= \sum_{k\neq i,j} B(D^i_j, k) \\ &= B(D^i_j) \end{aligned}$$

□

Proof of Lemma 5: The first property is straight forward from the method.

Now consider the second property. Without loss of generality, assume that $B(D^i_{k_1}) \geq B(D^i_{k_2}) \geq ... \geq B(D^i_{k_m})$ and $B(D^i_{k'_1}) \geq B(D^i_{k'_2}) \geq ... \geq B(D^i_{k'_{m'}})$. We prove in two cases:

(1) $m \geq m'$

Observe that for $1 \leq h \leq m$, $B(D^i_{k_h}) \geq C_2$, and for $1 \leq h \leq m'$, $B(D^i_{k'_h}) \leq B(D^i_{k_h})$. Hence,

$$\begin{aligned} &\sum_{1\leq h\leq m'} B(D^i_{k'_h}) - (C_1 + m' \cdot C_2) \\ &\quad\leq \sum_{1\leq h\leq m'} B(D^i_{k_h}) - (C_1 + m' \cdot C_2) \\ &\quad\leq \sum_{1\leq h\leq m} B(D^i_{k_h}) - (C_1 + m \cdot C_2) \end{aligned}$$

(2) $m < m'$

According to the method, for $1 \leq h \leq m$, $B(D^i_{k'_h}) \leq B(D^i_{k_h})$. For $m < h \leq m'$, $B(D^i_{k'_h}) < C_2$ and therefore $\sum_{m<h\leq m'} B(D^i_{k'_h}) - (m' - m) \cdot C_2 < 0$. Thus we have

$$\begin{aligned} &(\sum_{h=1}^{m'} B(D^i_{k'_h}) - (C_1 + m' \cdot C_2)) \\ &\quad -(\sum_{h=1}^{m} B(D^i_{k_h}) - (C_1 + m \cdot C_2)) \\ &\quad = (\sum_{h=1}^{m'} B(D^i_{k'_h}) - \sum_{h=1}^{m} B(D^i_{k_h})) - (m' - m) \cdot C_2 \\ &\quad \leq (\sum_{h=1}^{m} B(D^i_{k_h}) + \sum_{h=m+1}^{m'} B(D^i_{k'_h}) \\ &\quad -\sum_{h=1}^{m} B(D^i_{k_h})) - (m' - m) \cdot C_2 \\ &\quad = \sum_{m<h\leq m'} B(D^i_{k'_h}) - (m' - m) \cdot C_2 \\ &\quad < 0 \end{aligned}$$

□

Proof of Theorem 6: According to Theorem 5 and Lemma 5, we have:

$$\begin{aligned} E[\overline{NB}(M)] &= E[\sum_{1\leq h\leq m} \overline{B}(D^i_{k_h}) - (C_1 + m \cdot C_2)] \\ &= \sum_{1\leq h\leq m} E[\overline{B}(D^i_{k_h})] - (C_1 + m \cdot C_2) \\ &= \sum_{1\leq h\leq m} B(D^i_{k_h}) - (C_1 + m \cdot C_2) \\ &\geq 0 \end{aligned}$$

and

$$\begin{aligned} E[\overline{NB}(M')] &= E[\sum_{1\leq h\leq m'} \overline{B}(D^i_{k'_h}) - (C_1 + m' \cdot C_2)] \\ &= \sum_{1\leq h\leq m'} E[\overline{B}(D^i_{k'_h})] - (C_1 + m' \cdot C_2) \\ &= \sum_{1\leq h\leq m'} B(D^i_{k'_h}) - (C_1 + m' \cdot C_2) \\ &\leq \sum_{1\leq h\leq m} B(D^i_{k_h}) - (C_1 + m \cdot C_2) \\ &= \sum_{1\leq h\leq m} E[\overline{B}(D^i_{k_h})] - (C_1 + m \cdot C_2) \\ &= E[\sum_{1\leq h\leq m} \overline{B}(D^i_{k_h}) - (C_1 + m \cdot C_2)] \\ &= E[\overline{NB}(M)] \end{aligned}$$

□

Proof of Theorem 7: Observe first that for monotonous data items and absolute distance functions, the actual benefit to k of receiving D^i_j is the difference between the version of $\overline{D^k_j}$ and D^i_j. To see this, notice that in Equality 10, $v(\overline{D^k_j}) < v(D^i_j) \leq v(\overline{D_j})$. Hence,

$$\begin{aligned} \overline{B}(D^i_j, k) &= DIST(\overline{D^k_j}, \overline{D_j}) - DIST(D^i_j, \overline{D_j}) \\ &= |\overline{D^k_j} - \overline{D_j}| - |D^i_j - \overline{D_j}| \\ &= \overline{D_j} - \overline{D^k_j} - \overline{D_j} + D^i_j \\ &= DIST(\overline{D^k_j}, D^i_j) \end{aligned}$$

Now consider Equality 16. We have Equality 17 without the assumption that $D^i_j = \overline{D_j}$. The rest of the proof is the same as that of Theorem 5. □

Appendix B: Further experimental analysis of ABD

Impact of the extra resource cost

Clearly the system cost of ABD increases as the CPU factor C_3 and the storage unit cost C_4 increase. Specifically, suppose one is given the values for all the parameters, except C_3 and C_4. Then, for any one of the other protocols there exist values of C_3 and C_4, such that above these values ABD has higher system cost than that protocol. This is visually illustrated by Figures A-1 and A-2.

Figure A-1: System cost as a function of the CPU factor (c1=2,c2=0.2,c4=0.0001,cplb=0.3)

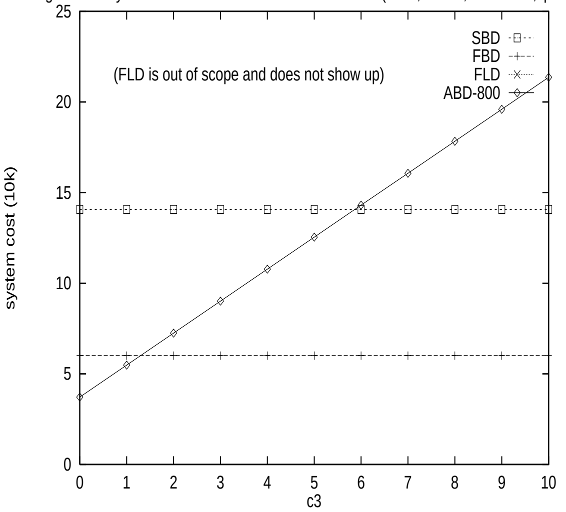
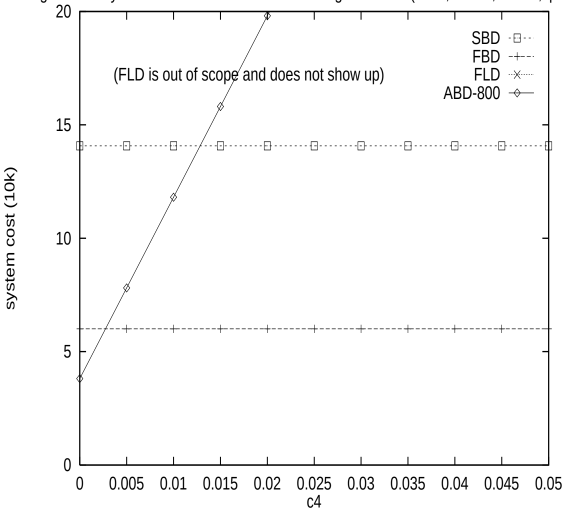


Figure A-2: System cost as a function of the storage unit cost (c1=2,c2=0.2,c3=0.1,cplb=0.3)



Comparisons between ABD-800 and ABD-400

Figure A-3 plots the system cost as a function of the message initiation cost C_1 with $cplb = 0.5$ in the military scenario. Figure A-4 plots the system cost as a function of cplb with $C_1 = 10$ and $C_2 = 1$. These experiments indicate that ABD-800 is always superior to ABD-400, with the cost of ABD-400 being up to 30% higher. This quantifies the benefit of using the extra storage to keep an additional version of each data item.

Figure A-3: System cost as function of message cost (c1/c2=2,cplb=0.5)

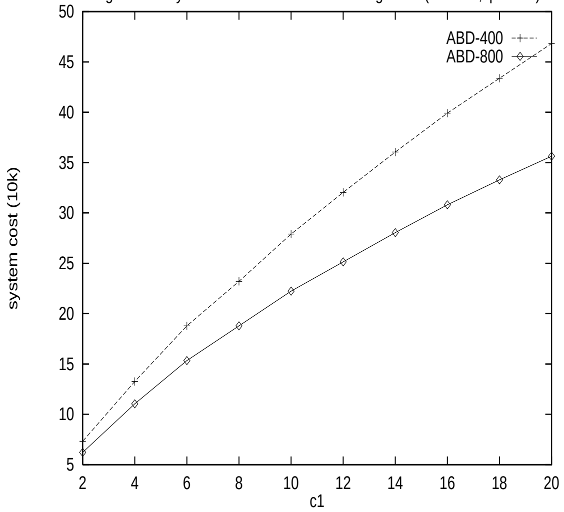


Figure A-4: System cost as function of cplb (c1=10,c2=1)