Trade-offs Between Replication and Availability in Distributed Databases

Amitabh Shah
Keith Marzullo

TR 89-1065
December 1989

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

Trade-offs between Replication and Availability in Distributed Databases

Amitabh Shah and *Keith Marzullo*
Department of Computer Science
Upson Hall, Cornell University
Ithaca, New York 14853
{shah,marzullo}@cs.cornell.edu

August 22 1989
(Revised December 15 1989)

Abstract

Distributed databases are generally built on top of standard communication facilities such as leased phone lines. Often several applications, running in different databases, use the same network for their communication. But the applications that run in these databases have grown increasingly more complex and demanding in their availability requirements, often with temporal constraints (such as real-time databases). We argue in this paper that there is a need for dedicated communication networks designed for specific applications. To design such networks, we argue that it is necessary to analyze the data access patterns in the applications that run on the top of the networks. Such analysis would give a network designer an insight into where and how much to replicate the data in the system. Replication can increase availability of data, but too much replication can also hamper it. Thus, for a given application, there exists a right balance between replication and availability. Our goal is to find this balance and show how to design a network of the cheapest cost that achieves it. In this paper, we take the first step towards this design problem by precisely characterizing the trade-offs between replication and availability and suggest a network design strategy to exploit these trade-offs.

Trade-offs between Replication and Availability in Distributed Databases

1 Introduction

Availability, as a measure of performance, is an important criterion in the design of fault-tolerant database systems [BG86,BHG87]. Centralized databases show extreme behavior in presence of failures: they are either available or unavailable. Besides, they cannot deal effectively with data that is naturally distributed. Thus there is a need for *replicated* distributed databases. By replicating the data at non-local sites, the availability of the data at those sites is increased.

However, this benefit is not without costs. First, there is the physical cost of replicating data. Perhaps more costly is that the management of database applications becomes much more complex. This is because replicated databases have a strong consistency requirement that transactions executing in them be *one-copy serializable* [BG86]. The more a particular set of data is replicated, the more constraints there are on the transactions that read and write that data: the replica control protocol has to safeguard against the possibility that the same data could be inconsistently updated at different sites. The problem worsens if the sites and the underlying communication network are prone to failures that may lead to partitioning of the network into several groups. The transactions executing in one partition group may have to block if there is a possibility that transactions executing in another group may access the same data; blocking reduces availability.

Distributed databases are generally built on top of standard communication facilities such as leased phone lines. Often several applications, running in different databases, use the same phone lines for their communication. But the applications that run in these databases have increasingly grown more complex and demanding in their availability requirements, often with deadline constraints (real-time databases).

We argue in this paper that such real-world applications have an *a-priori* structure that can be exploited to design a dedicated communications network suited for that application. Doing so allows the deadline constraints of the application to be met, with a low interconnection cost. By designing a dedicated network, we mean specifying a pattern of interconnection. For example, this specification could include leasing of phone lines to connect only critical pairs of sites (as opposed to a fully connected system).

To design such networks, we argue that it is necessary to analyze the data access patterns in the database applications that run on the top of the network. Such analysis would give a network designer an insight into where, and how much, to replicate the data in the system. As mentioned earlier, availability of data is hampered by too little or too much replication. Thus, for a given application, there exists a right balance between the two. Our goal is to design a network of the cheapest cost that achieves this balance between replication and availability. In this paper, we take the first step towards this design problem by precisely characterizing the trade-offs between replication and availability and suggest a network design strategy to exploit these trade-offs.

In [COK86] Coan, Oki, and Kolodner explore the *quantification* of availability in distributed databases that are subject to partition failures. They derive an upper bound on availability, when the database is partitioned into two groups (a *simple partition*), assuming that the database satisfies certain simple constraints, viz., that the data is fully replicated and that the pattern of data access satisfies a *uniformity assumption*. They show that for a simple partition, the best availability that one can hope for occurs when all the read and update transactions successfully complete in one partition group (the majority group) and only the read transactions complete in the other. They mention that higher availability than that given by their upper bound may be achieved if uniformity of data access is not assumed, although they do not derive the upper bounds in these cases. We address precisely this question in this paper.

In fact, we address it along two directions. For the case of simple partitions, we derive upper bounds on availability based on two criteria: the *pattern of data access* and the *degree of replication* in the database. For the first criterion, we either assume that the data access pattern satisfies the uniformity assumption or that it satisfies a more realistic *dominant local access* assumption. For the second, we consider either full or partial replication. Our results show that for full replication, there are trade-offs involved between the two assumptions of data access—that update availability of a subset of data is increased at the expense of that of another subset, whereas, for read availability, exactly the opposite is true. For partial replication, the availability increases significantly with dominant local access. In fact, this case achieves the highest availability upper bound among the four possible cases.

The benefits of both partial replication and the assumption of dominant local access are even more apparent when the database is partitioned into more than two groups (a *multiple partition*). With full replication, in the worst case, only the read transactions may complete in the database. With partial replication, a substantial update availability can be achieved. Finally, our results lead us to the following observation: distributed database applications that customize the patterns of data access using knowledge of limited replication of data can achieve higher availability than those that do not. Thus, the architecture of the database can be customized to the requirement of the application to achieve higher availability. Based on the results in this paper, we have proposed a methodology for designing *Harary Networks* that provide the architectural support needed for higher availability [MS90].

We argue that this approach is suitable for designing distributed real-time systems, for two reasons. Firstly, there is a natural need for higher availability in such systems since the transactions executing in such systems usually have to adhere to time-critical deadlines. Secondly, by their design and requirements, such systems have transactions that are periodic in nature, with a predominant local access pattern. We are applying the results in this paper to the problem of designing highly available real-time systems.

The remainder of this paper is organized as follows. In section 2 the underlying model of the distributed database is defined, and the upper bounds on availability in the case of a simple partition are derived for different assumptions about data access and replication. In section 3 the question of multiple partitions is discussed, and the corresponding upper bound on availability is derived. In section 4 we conclude with some directions for future work. In particular, we discuss adapting replica control protocols to take into consideration the knowledge of limited replication and specific patterns of data access to achieve higher availability. We conclude with a discussion of the need for architectural support when designing systems that require high availability.

2 Replication vs. Availability

In this section we derive the upper bounds on availability for different assumptions about replication and patterns of data access. (For this section, we assume that the network is partitioned into exactly *two* groups. We deal with more than two partitions in the next section.) First we describe the model of the system and introduce our notation.

2.1 Model of the System

We consider a *distributed database system* consisting of n sites, numbered 1, 2, ..., n, connected by a *communication network*. The set of *data objects* (or *objects*) for the database is denoted as $\mathcal{D}$. The objects are replicated at various sites. The database is managed by a *replica control protocol* (or *protocol*) that runs at every site; the protocol manages actions at every site and the communication between sites.

The sites and the links within the communication network can fail by *crashing* [Had84]. When the failed components recover, they are integrated back into the network. As in [COK86], we say that a *partition* occurs when two functioning sites in the system cannot communicate for a significant interval of time. A *partition group* is a maximal set of sites that can communicate with each other [Dav84]. A partition is *simple* if the system partitions into exactly two groups, it is *multiple* otherwise.

We consider two different assumptions about the replication of data. We say that the database is *fully replicated* if the set $\mathcal{D}$ resides at all the sites in the system; it is *partially replicated* otherwise. In either case, with each data object is associated a set of sites, called the *primary sites* for that object. For a data object d, the set of its primary sites is denoted by $PS(d)$. The primary sites of a data object are intended to be the sites where the "source"—the physical entity modeled by the object—of d resides. We say that a set of data objects X is *local* to a site j, if $j \in PS(d)$ for every object d in X. We denote the set $\bigcup_{d \in X} PS(d)$ by $PS(X)$. The set of data local to a site j is denoted as D_j. Note that $\mathcal{D} = \bigcup_i D_i$, and for full replication, for every $i, D_i = \mathcal{D}$.

A *transaction* is a set of *ordered operations*; the operations are of two kinds: *read* and *update*.[1] We assume, without loss of generality, that a transaction reads and updates an object at most once and that if it reads and updates the same object, the read precedes the update.

Let T be the set of all the transactions that execute in the database. We say that a transaction *accesses* a data object if it either reads or updates that object. For every subset X of $\mathcal{D}$, we denote by T_X the set of transactions that accesses data objects in X. (In [COK86] this set is called *X-transactions.*) Note that $T = T_{\mathcal{D}}$. A set of transactions that accesses both a set X and a set Y is denoted as $T_{X,Y}$. For every set of transactions T_X, we denote by T_X^j, the subset of transactions that are initiated at site j. Every set of transactions T_X is a union of two disjoint sets, R_X and U_X, the set of read and update transactions that access X respectively.

Of the set of transactions, T, that is presented to the database, let T_c be the subset that actually completes. Then, as in [COK86], we define *availability* as the

[1]We do not consider *blind writes* here; we assume that a transaction reads an object before writing it.

proportion of T_c in T, i.e.,

$$availability \stackrel{\text{def}}{=} \frac{|T_s|}{|T|}.$$

Next we derive upper bounds for availability assuming first that the data access satisfies a uniformity assumption, and then assuming that it satisfies the dominant local assumption. For each case, we also consider the subcases where the database is either partially or fully replicated.

2.2 Availability with Uniform Data Access

Coan et al. define the *uniformity assumption* as follows: let the *load* L_j of a site j be defined to be the ratio of the number of transactions initiated at j to the total number of transactions in the system, i.e.,

$$L_j \stackrel{\text{def}}{=} \frac{|T_{\mathcal{D}}^j|}{|T_{\mathcal{D}}|}.$$

Then the uniformity assumption is defined to hold if every set of X-transactions is distributed over the sites according to their respective loads:

$$\forall j, \forall X \subseteq \mathcal{D}\ \frac{|T_X^j|}{|T_X|} = L_j.$$

As in Coan et al., we define a set S of sites to be a *majority* if $\sum_{j \in S} L_j > 1/2$. It is a *minority* otherwise. In case of a simple partition, we assume that the system partitions into a majority and a minority partition group. For a multiple partition, no group may be a majority one.

Before we derive the upper bounds on availability, we recall the result of [COK86], which is based on the following crucial observation: in a partitioned, fully replicated database, two transactions, one executing in each partition group, cannot update the same data object without violating the serializability constraints. Thus, the best we can hope for is that the data is read and updated in one partition group and only read in the other. To maximize availability, we would like all the read and updates transactions in the majority partition group, and the read transactions in the minority group, to complete, i.e.,

$$availability \leq \frac{1}{|T|}\Big[|R_{MAJ}| + |U_{MAJ}| + |R_{MIN}|\Big],$$

where R_{MAJ} and U_{MAJ}, respectively, are the read and update transactions initiated in the majority partition group and R_{MIN} are the read transactions initiated in the minority group. We now consider the two different assumptions about replication.

2.2.1 Partial Replication with Uniform Access

First we derive the upper bound for the case when the data is partially replicated. Consider a partitioning of the network into two groups, $\mathcal{P}_1$ and $\mathcal{P}_2$. Let the data set $\mathcal{D}$ be partitioned into disjoint sets $\mathcal{D}_1$, $\mathcal{D}_2$ and $\mathcal{D}_{12}$, such that the set of data $\mathcal{D}_1$ resides completely in partition group $\mathcal{P}_1$, $\mathcal{D}_2$ completely in group $\mathcal{P}_2$, and the set $\mathcal{D}_{12}$ is replicated in both the groups (Figure 1(a)). Note that these sets need not be replicated at every site within the partition groups, only that they are accessible from within the group. (The partitioning of $\mathcal{D}$ is decided by the primary sites for the objects in $\mathcal{D}$: $PS(\mathcal{D}_1) \subseteq \mathcal{P}_1$, $PS(\mathcal{D}_2) \subseteq \mathcal{P}_2$ and $PS(\mathcal{D}_{12}) \cap \mathcal{P}_i \neq \emptyset$ for $i = 1, 2$.)

Let the set of transactions $T_{\mathcal{D}}$ be divided into disjoint sets $T_{\mathcal{D}_1}$, $T_{\mathcal{D}_2}$, $T_{\mathcal{D}_{12}}$, $T_{\mathcal{D}_{1,2}}$, $\ldots$, $T_{\mathcal{D}_{1,2,12}}$, where the first three terms denote the transactions that access only data sets $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_{12}$ respectively, $\ldots$, the last term denotes the transactions that access all the three sets together. Mathematically:

$$T_{\mathcal{D}} = T_{\mathcal{D}_1} \bigcup T_{\mathcal{D}_2} \bigcup T_{\mathcal{D}_{12}} \bigcup T_{\mathcal{D}_{1,2}} \bigcup T_{\mathcal{D}_{1,12}} \bigcup T_{\mathcal{D}_{2,12}} \bigcup T_{\mathcal{D}_{1,2,12}}.$$

Without loss of generality, we will assume that the sets of transactions $T_{\mathcal{D}_{1,2}}$, and $T_{\mathcal{D}_{1,2,12}}$ are empty. This is done mainly to avoid messy expressions of bounds on availability that are derived later in this section. We expect such transactions to be rare in practice. It is also easy to see how to decompose such transactions into transactions of the other kind—specifically of the kind $T_{\mathcal{D}_{12}}$, $T_{\mathcal{D}_{1,12}}$, and $T_{\mathcal{D}_{2,12}}$—without altering the consistency and correctness of the original transaction system. Here the underlying assumption is that $\mathcal{D}_{12}$ acts as a set of global variables through which local variables could be communicated. Thus, we will assume that the transactions were appropriately defined at the system design stage.

Let $\mathcal{P}_1$ be the majority partition group. Let the load of $\mathcal{P}_1$ be k ($k = \sum_{j \in \mathcal{P}_1} L_j$). Thus, $\mathcal{P}_2$ is a minority partition group, with load $1-k$. The assumption that $\mathcal{P}_1$ has a majority implies that $k > 1/2$. To compute availability in this scenario, we note that all the transactions accessing $\mathcal{D}_1$ and $\mathcal{D}_{1,12}$ in group $\mathcal{P}_1$ can successfully complete. By the uniformity assumption, there are $k|T_{\mathcal{D}_1}|$ and $k|T_{\mathcal{D}_{1,12}}|$ of these, respectively. Similarly for the transactions accessing $\mathcal{D}_2$ in $\mathcal{P}_2$, of which there are $(1-k)|T_{\mathcal{D}_2}|$. For transactions that access $\mathcal{D}_{12}$, we use the result of Coan et al.—the best availability for these transactions would be that the read and update transactions can complete

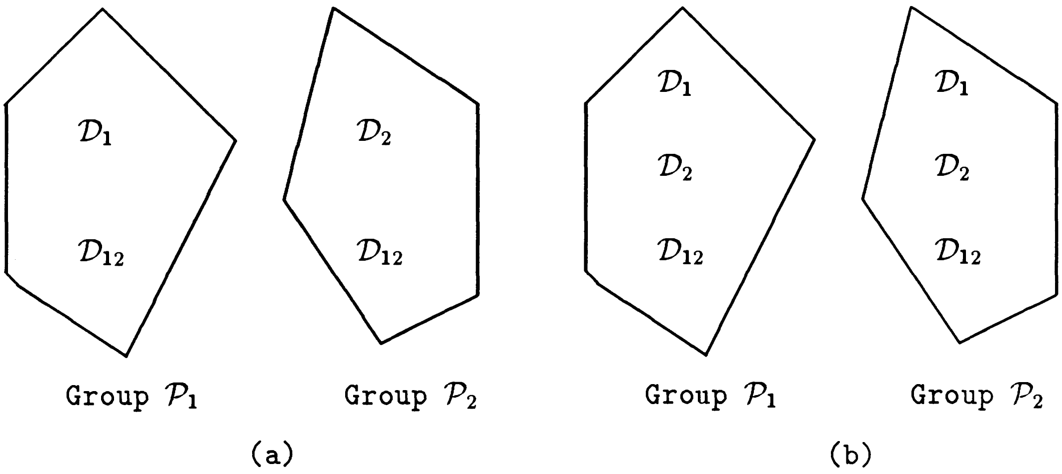

Figure 1: Simple Partition with (a) Partial Replication and (b) Full Replication

in the majority partition group ($k|T_{\mathcal{D}_{12}}|$ of these); only the read transactions ($(1-k)|R_{\mathcal{D}_{12}}|$ such) can complete in the minority group. For the set of data $\mathcal{D}_{2,12}$, only the read transaction can complete in the group $\mathcal{P}_2$ ($(1-k)|R_{\mathcal{D}_{2,12}}|$ of these). Thus, the upper bound on availability, $avail_{pu}$, is given by:

$$\begin{aligned} avail_{pu} \; &\leq \; \frac{1}{|T_{\mathcal{D}}|}\Big[k(|T_{\mathcal{D}_1}| + |T_{\mathcal{D}_{1,12}}| + |T_{\mathcal{D}_{12}}|) \\ &\qquad + (1-k)(|T_{\mathcal{D}_2}| + |R_{\mathcal{D}_{2,12}}| + |R_{\mathcal{D}_{12}}|)\Big]. \end{aligned} \tag{1}$$

2.2.2 Full Replication with Uniform Access

This is the case considered in [COK86]; we recapitulate their result in our terminology. Let $\mathcal{P}_1$ again be the majority partition group. We consider the partitioning as above except that now the data set $\mathcal{D}$ is replicated at every site (thus in every group) (Figure 1(b)). (Here the partitioning of $\mathcal{D}$ into $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_{12}$ is shown only to compare the availability with the results in other cases.)

Now the best availability we can hope for is that the read and update transactions in $\mathcal{P}_1$ complete but only the read transactions in $\mathcal{P}_2$ succeed. Thus, we get the following bound on availability, $avail_{fu}$:

$$\begin{aligned} avail_{fu} \;\le\; & \frac{1}{|T_{\mathcal{D}}|}\Big[k(|T_{\mathcal{D}_1}| + |T_{\mathcal{D}_{1,12}}| + |T_{\mathcal{D}_2}| + |T_{\mathcal{D}_{2,12}}| + |T_{\mathcal{D}_{12}}|) \\ & + (1-k)(|R_{\mathcal{D}_1}| + |R_{\mathcal{D}_{1,12}}| + |R_{\mathcal{D}_2}| + |R_{\mathcal{D}_{2,12}}| + |R_{\mathcal{D}_{12}}|)\Big]. \qquad (2) \end{aligned}$$

Note that the first term within the brackets corresponds to $|R_{MAJ}| + |U_{MAJ}|$ in Coan et al.'s terminology, while the second term corresponds to $|R_{MIN}|$.

2.3 Availability with Dominant Local Access

Now we make the assumption that the set of transactions does not satisfy uniformity of data access—that its distribution is skewed towards accessing local and neighboring data rather than distant ones. This assumption is more realistic than that of uniformity, as evidenced by several real-life systems. For instance, in a banking database, one would expect that the majority of transactions at the branches within a city would access the records of the local customers. A travel agency would mostly need to access the information about flights to and from the city it is located in. We call this the assumption of *Dominant Local Access.*

Before defining this assumption, we extend the notion of access as follows. Define the *local load* λ_j of a site j as the proportion of the transactions initiated at site j that access data at site j, among all the transactions that access data at site j. Mathematically:

$$\lambda_j \stackrel{\text{def}}{=} \frac{|T^j_{D_j}|}{|T_{D_j}|}.$$

We extend the notion of local load to a set of sites in a natural way. Let $\mathcal{P}_i$ be a set of sites, and let $\mathcal{D}_i$ be the subset of $\mathcal{D}$ that is local to $\mathcal{P}_i$. The local load Λ_i of $\mathcal{P}_i$ is defined as the proportion of transactions initiating in $\mathcal{P}_i$ that access data local to $\mathcal{P}_i$, among all the transactions that access data local to $\mathcal{P}_i$:

$$\Lambda_i \stackrel{\text{def}}{=} \frac{|T^{\mathcal{P}_i}_{\mathcal{D}_i}|}{|T_{\mathcal{D}_i}|}, \quad \begin{array}{lll} \text{where} & T^{\mathcal{P}_i}_{\mathcal{D}_i} = & \displaystyle\bigcup_{k\in\mathcal{P}_i}\bigcup_{j\in\mathcal{P}_i} T^k_{D_j}, \\ \text{and} & T_{\mathcal{D}_i} = & \displaystyle\bigcup_{k}\bigcup_{j\in\mathcal{P}_i} T^k_{D_j}. \end{array}$$

Then the assumption of dominant local access is said to be satisfied if the local loads, λ_j's and Λ_j's satisfy $1 \ge \lambda_j \gg 1/2$ and $1 \ge \Lambda_j \gg 1/2$ for every site j, and for every partition group $\mathcal{P}_i$.

2.3.1 Partial Replication with Dominant Local Access

We again consider the partition scenario of section 2.2.1 (Figure 1(a)). Now the sets of data $\mathcal{D}_1$, and $\mathcal{D}_{1,12}$ in the partition group $\mathcal{P}_1$, and $\mathcal{D}_2$, and $\mathcal{D}_{2,12}$ in the group $\mathcal{P}_2$ are accessed according to the dominant local access assumption. Since $\mathcal{D}_{12}$ is replicated at all sites, we assume that the transactions that access $\mathcal{D}_{12}$ in $\mathcal{P}_1$ and $\mathcal{P}_2$ satisfy the uniformity assumption. Note that the update transactions on the set $\mathcal{D}_{2,12}$ can not succeed in $\mathcal{P}_2$ in the worst case, since the set $\mathcal{D}_{12}$ is shared between both the groups. Thus, the upper bound on availability, $avail_{pl}$, is given by

$$\begin{aligned} avail_{pl} \;\leq\; & \frac{1}{|T_{\mathcal{D}}|}\Big[\Lambda_1(|T_{\mathcal{D}_1}| + |T_{\mathcal{D}_{1,12}}|) + \Lambda_2(|T_{\mathcal{D}_2}| + |R_{\mathcal{D}_{2,12}}|) \\ & + k|T_{\mathcal{D}_{12}}| + (1-k)|R_{\mathcal{D}_{12}}|\Big]. \end{aligned} \tag{3}$$

2.3.2 Full Replication with Dominant Local Access

We now consider the partition scenario of section 2.2.2 (Figure 1(b)). Again, we assume that the set $\mathcal{D}_{12}$ is accessed uniformly in both $\mathcal{P}_1$ and $\mathcal{P}_2$. Since the data is fully replicated, we let only the read transactions succeed in the minority partition group, whereas all transactions succeed in the majority group. However, due to the dominant local assumption, the number of transactions that accesses $\mathcal{D}_2$ and $\mathcal{D}_{2,12}$ in $\mathcal{P}_1$ is very small compared to those in $\mathcal{P}_2$. Thus, the upper bound on availability, $avail_{fl}$, is given by

$$\begin{aligned} avail_{fl} \;\leq\; & \frac{1}{|T_{\mathcal{D}}|}\Big[\Lambda_1(|T_{\mathcal{D}_1}| + |T_{\mathcal{D}_{1,12}}|) + (1-\Lambda_1)(|R_{\mathcal{D}_1}| + |R_{\mathcal{D}_{1,12}}|) \\ & + (1-\Lambda_2)(|T_{\mathcal{D}_2}| + |T_{\mathcal{D}_{2,12}}|) + \Lambda_2(|R_{\mathcal{D}_2}| + |R_{\mathcal{D}_{2,12}}|) \\ & + k|T_{\mathcal{D}_{12}}| + (1-k)|R_{\mathcal{D}_{12}}|\Big]. \end{aligned} \tag{4}$$

2.4 Replication — Availability Trade-offs

Note that in the comparison of availability upper bounds above, the availability for the transactions that access set $\mathcal{D}_{12}$ is the same in all four cases. This is to be expected since we have assumed uniform access to $\mathcal{D}_{12}$ in all the cases. Thus, we shall only compare availability for transactions that access either the sets $\mathcal{D}_1$ and $\mathcal{D}_2$ alone, or along with $\mathcal{D}_{12}$.

For the uniformity assumption, the worst partition scenario occurs when both the partition groups are of about the same size ($k \approx 1/2$). This is when $avail_{fu}$ (which is also the *availability* in [COK86]) has the *smallest* upper bound. Recall that even in this worst case scenario, the local loads of each partition group are still $\gg 1/2$ when dominant local access is assumed.

From inequalities 1 and 2, it is clear that full replication has a better upper bound on availability than partial replication when the data access is uniform. Comparing inequalities 2 and 4 we find a trade-off between transactions accessing $\mathcal{D}_1$ and $\mathcal{D}_2$; the difference in the upper bounds for $avail_{fl}$ and $avail_{fu}$ is:

$$\begin{aligned}\frac{1}{|T_{\mathcal{D}}|}\Big[&(\Lambda_1(|T_{\mathcal{D}_1}|+|T_{\mathcal{D}_{1,12}}|)+(1-\Lambda_1)(|R_{\mathcal{D}_1}|+|R_{\mathcal{D}_{1,12}}|)\\ &+(1-\Lambda_2)(|T_{\mathcal{D}_2}|+|T_{\mathcal{D}_{2,12}}|)+(\Lambda_2)(|R_{\mathcal{D}_2}|+|R_{\mathcal{D}_{2,12}}|))\\ &-(k(|T_{\mathcal{D}_1}|+|T_{\mathcal{D}_{1,12}}|)+(1-k)(|R_{\mathcal{D}_1}|+|R_{\mathcal{D}_{1,12}}|)\\ &+k(|T_{\mathcal{D}_2}|+|T_{\mathcal{D}_{2,12}}|)+(1-k)(|R_{\mathcal{D}_2}|+|R_{\mathcal{D}_{2,12}}|))\Big],\end{aligned}$$

that is,

$$\frac{1}{|T_{\mathcal{D}}|}\Big[(\Lambda_1-k)(|U_{\mathcal{D}_1}|+|U_{\mathcal{D}_{1,12}}|)-(\Lambda_2+k-1)(|U_{\mathcal{D}_2}|+|U_{\mathcal{D}_{2,12}}|)\Big].$$

(In the above, we use the fact that $|T_X| = |R_X| + |U_X|$ for any set X.) This suggests that there is a trade-off between the update transactions for $\mathcal{D}_1$ and $\mathcal{D}_{1,12}$, and those for $\mathcal{D}_2$ and $\mathcal{D}_{2,12}$, viz., that the dominant local access assumption allows more updates on the former (i.e., on data whose primary sites are in the majority partition group), and fewer updates on the latter (with primary sites in the minority group); the uniformity assumption does the opposite.

Comparing inequality 3 with inequalities 2 and 4, we find that partial replication with the assumption of dominant local access is superior to full replication with *either* assumption of data access: the difference in the upper bounds for $avail_{pl}$ and $avail_{fu}$ is

$$\begin{aligned}\frac{1}{|T_{\mathcal{D}}|}\Big[&(\Lambda_1-k)(|T_{\mathcal{D}_1}|+|T_{\mathcal{D}_{1,12}}|)+(\Lambda_2-k)(|T_{\mathcal{D}_2}|+|T_{\mathcal{D}_{2,12}}|)\\ &-(1-k)(|R_{\mathcal{D}_1}|+|R_{\mathcal{D}_2}|+|R_{\mathcal{D}_{1,12}}|+|R_{\mathcal{D}_{2,12}}|)\Big],\end{aligned}$$

which we expect to be positive unless the sets of transactions $T_{\mathcal{D}_1}$, $T_{\mathcal{D}_2}$, $T_{\mathcal{D}_{1,12}}$, and

$T_{\mathcal{D}_{2,12}}$ consist mostly of read operations—not a likely case.[2] The difference in the upper bounds for $avail_{pl}$ and $avail_{fl}$ is

$$\frac{1}{|T_{\mathcal{D}}|}\Big[(2\Lambda_2 - 1)|U_{\mathcal{D}_2}| - (1-\Lambda_1)(|R_{\mathcal{D}_1}| + |R_{\mathcal{D}_{1,12}}|) - (1-\Lambda_2)(|U_{\mathcal{D}_{2,12}}| + |R_{\mathcal{D}_2}|)\Big].$$

In this case, a substantial amount of update availability for $\mathcal{D}_2$ is gained at the expense of insignificant read availability for $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_{1,12}$, and update availability for $\mathcal{D}_{2,12}$. This is since we expect the local loads, the Λ_i's, to be ≈ 1.

To summarize the results above, partial replication with the dominant local access assumption has the highest upper bound on availability among the four possible cases. With full replication, there is a trade-off between two sets of update transactions, according to the access assumption. Partial replication with the uniformity assumption has the least upper bound on availability among the four cases.

3 Multiple Partitions

In this section, we consider the case when there are more than two groups as a result of network failures. When the data is fully replicated, the availability can be significantly reduced in this case: it is possible that none of the groups have a majority (in the worst case). In this case, the read transactions can complete in all the groups, but no update transactions can complete in any group, which is not very useful. This is where partial replication would be particularly attractive, especially with the assumption of dominant local access.

Consider the following scenario. Let the network be partitioned into groups $\mathcal{P}_1$, $\mathcal{P}_2$, ..., $\mathcal{P}_k$. Let the set of data, $\mathcal{D}$, be divided into a number of disjoint sets, as shown in Figure 2. For each partition group $\mathcal{P}_i$, the set $\mathcal{D}_i$ resides completely within that group, the set $\mathcal{D}_{ij}$ is replicated at groups $\mathcal{P}_i$ and $\mathcal{P}_j$, the set $\mathcal{D}_{ijk}$ resides in $\mathcal{P}_i$, $\mathcal{P}_j$ and $\mathcal{P}_k$, ..., and the set $\mathcal{D}_{12\ldots k}$ is replicated in all the k groups. (Note that the data replication abstraction described above is a very general one; it subsumes both the *complete sharing of memory* and *no sharing of memory* (*message passing*) abstractions discussed in the literature.)

By reasoning as in the previous section, we will assume without loss of generality, that besides the transactions that access the above sets, the only other kinds of

[2] If most of the transactions are read-only, then there will be fewer serializability requirements; network partitioning can not substantially decrease availability in that case.

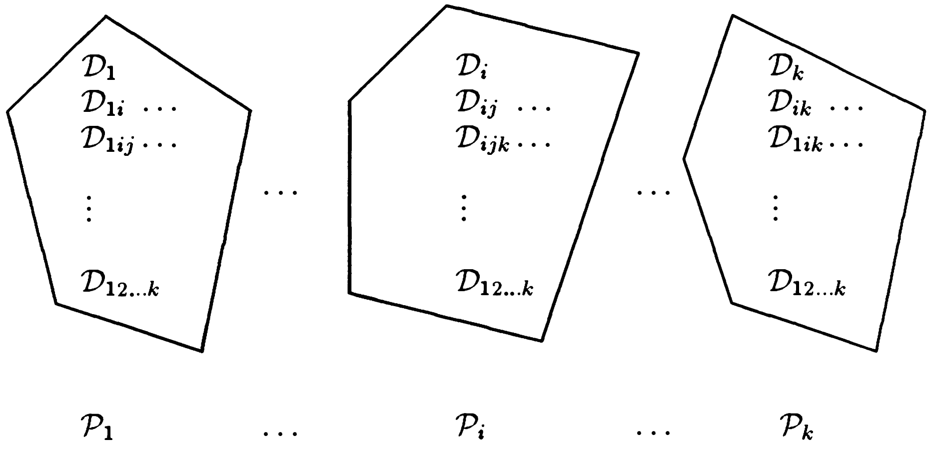

Figure 2: Multiple Partition with Partial Replication

transactions that we need to consider are those that access one of those sets and $\mathcal{D}_{12\ldots k}$ simultaneously. However, for the latter sets, as for the set $\mathcal{D}_{12\ldots k}$, only read transactions can succeed since none of the partition groups may have the majority for $\mathcal{D}_{12\ldots k}$ in the worst case. Thus, we get the following upper bound on the availability, $avail_{mult}$:

$$\begin{aligned} avail_{mult} \leq \; & \frac{1}{|T_{\mathcal{D}}|}\Big[\sum_i \Lambda_i |T_{\mathcal{D}_i}| + \sum_{ij}[\Lambda_{maj(i,j)}(|T_{\mathcal{D}_{ij}}| + |R_{\mathcal{D}_{ij,12\ldots k}}|) \\ & + \Lambda_{min(i,j)}(|R_{\mathcal{D}_{ij}}| + |R_{\mathcal{D}_{ij,12\ldots k}}|)] \\ & + \sum_{ijk}(|R_{\mathcal{D}_{ijk}}| + |R_{\mathcal{D}_{ijk,12\ldots k}}|) + \ldots + |R_{\mathcal{D}_{12\ldots k}}|\Big], \end{aligned} \tag{5}$$

where $\Lambda_{maj(i,j)}$ is Λ_i or Λ_j according to whether $\mathcal{P}_i$ or $\mathcal{P}_j$ has the majority for $\mathcal{D}_{ij}$; $\Lambda_{min(i,j)}$ is defined similarly. For all the data sets that are shared between three or more partition groups, only the read transactions can complete since none of the sharing groups may have a majority. But, assuming dominant local data access as in the previous section, we note that the terms in the first sum dominate others in the expression. Thus, not just all the read transactions in the database complete, as was the case with full replication, but also all the update transactions accessing the sets of data $\mathcal{D}_i$ (local to group $\mathcal{P}_i$) can complete. Substantial update availability is also achieved for the sets of data, $\mathcal{D}_{ij}$, that are shared between exactly two partition groups. (Note that even if uniformity was assumed, partial replication would still achieve higher availability than full replication.)

4 Discussion

The results in the previous sections suggest that the standard replicated database theory can be augmented to take advantage of the knowledge that data is not replicated at every site in the system and that the transaction mix at every site is skewed towards accessing local and neighboring data rather than non-local or distant data. We would like to pursue this problem from two directions, viz., that of adapting the replica control protocols to use the above results, and that of customizing the underlying network architecture to the specific database application at hand.

4.1 Adapting Replica Control Protocols

Various replica management schemes have been proposed to manage transactions in replicated databases in spite of partition failures [Her86, ET89]. The underlying notion in these schemes is that of the *view* of a site, consisting of the sites that it can communicate with (and thus, find out what partition group it belongs to). We suggest that the views be augmented to also include the knowledge of data replication and the transaction mix at these sites. For instance, if it is known in a view that all the primary sites for a data item are in the partition group corresponding to the view, transactions accessing that data item can be committed and new values installed, without blocking for transactions in other partition groups that may access the same data, thus increasing availability.

4.2 Need for Architectural Support

Most distributed database applications use available networks, such as leased phone lines for communication support. We argue that often it is necessary to build customized networks for enhanced availability. This makes sense when most of the data in the system is relatively stable, i.e. non-migratory. Most real-life applications exhibit such stability. Real-time database systems are particularly so, since it is the location of the sensors, or actuators that determine the source of data. These locations are usually given as an input specification to the design of the system, and are impermutable.

In such systems, it is important to exploit the system structure. We saw in the previous sections that for partitions that divide the set of data objects in certain ways, the availability can be better than that achieved by full replication of data. But it is not necessary that in case of independent component failures, the network will partition in this way. However, it may be possible to influence the network partitioning by defining its connectivity suitably. This suggests that one could also consider solving the following network design problem to increase availability, or to be more precise, to increase *expected* availability.

4.2.1 Design by Desirable Partition

The network consists of n sites, and there are m (physical) links available, each of which can connect two sites. Each site or link has some probability of failure (e.g. the *mean time between failures* is uniformly distributed). Assuming that the distribution of the data objects and the transactions that access them is known, the problem then is to connect the n sites using m links so that the upper bounds on availability (as given in sections 2 and 3) are achieved with maximum probability, in spite of k component failures. (It seems that this problem would be computationally hard: the related problem of *network reliability*—to determine with at least a given rational probability, whether a given subset of nodes remains connected in spite of independent link failures (also with rational probabilities)—is known to be *NP*-hard [GJ79]. In [MS90], we investigate a variant of this problem that is more tractable and practical.)

Thus, if certain sites hold data that are "logically" connected, we would like to maximize the probability of them being in the same partition, to increase availability for the transactions that access this data. Thus, we could allocate extra links to that set of sites that are logically connected (via the data objects that reside in them) at the expense of fewer links between those that are not so connected. We call this the problem of *network design by desirable partition.* The flavor of this problem is illustrated by the following example.

Example: Figure 3(a) represents the network for an airline database system, consisting of 8 sites and 12 links. Sites A and B are two major hubs within the airline system. Since we expect a lot of traffic between these two hubs, we would like to ensure that they remain connected as much as possible; thus, two links are allocated between them. Assume, for simplicity, that the sites in this network do not fail, only

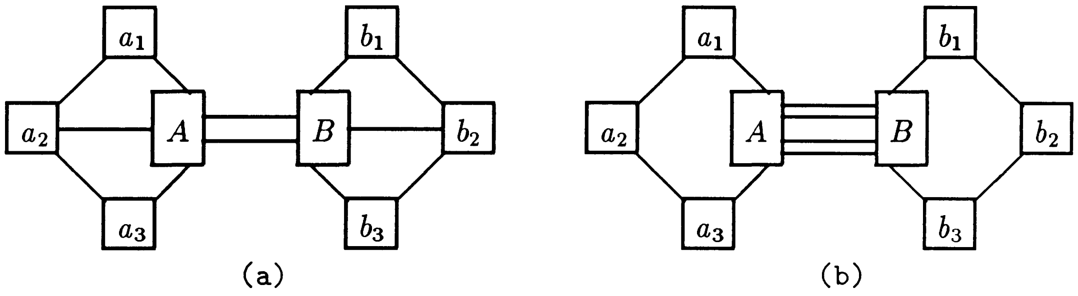

Figure 3: A Communication Network

links do. In this system, it takes these two link failures to disconnect the two hubs from each other. If all link failures are equally likely within some interval of time, this happens with probability 1/66. If the network was reconfigured as shown in Figure 3(b), then it would take four link failures to disconnect the hubs, with corresponding probability 1/495. Of course, this is at the expense of the connectivity of sites a_2 and b_2 in the network, increasing the chances that they could be disconnected from the system. Thus, availability of more critical data is increased at the expense of less critical data.

On the other hand, if the same network was designed for an inter-continental banking database, with the four sites on the left on one continent, and the four on the right on another, we might prefer the first design over the second: it might be more desirable to let the partition group within each continent operate autonomously than suffer a loss of availability within it. This would be at the expense of availability to the transactions that access data in both the continents simultaneously. ■

In the example, we have considered a very simple model of the link costs. In general, the cost of a link, and thus the cost of communication, depends on the distance between the sites it connects, as well as on various other factors. For example, the nature of the transactions that run in the system has to be accounted for: not all of them have the same importance, or priority. Thus, the quantification of availability has to incorporate such factors. In [MS90] we propose a framework of computing *dependencies* between sets of sites in a distributed database that accounts for these factors. Based on this framework, we propose the model of *Harary Networks* to design the connectivity of the network and give an algorithm to construct a Harary Network for specific database applications.

5 Conclusions

We have presented an analysis of transaction availability in a distributed database that is prone to site and link failures. We derived upper bounds on availability in such a system under an arbitrary locality of access. In particular, we applied the results to dominant local access, a common assumption of most real-life distributed databases. We showed that in case of partial replication and dominant local access, a substantial gain in availability can be made despite the case of network partitioning where none of the partition groups has a majority.

Based on the above results, we propose a communication network design strategy that maximizes the expected availability in such a system [MS90]. We have also solved a stochastic model of a hierarchically constructed distributed database; the model is parametrized by the locality of access—specifically by the arrival rates of transactions in the system [GS90]. Our simulation studies with this stochastic model show that indeed a substantial gain in availability is made with the assumption of dominant local access within the partition groups.

In conclusion, the problem of designing the "right" architecture depends on the application. The knowledge of availability requirements and the logical relation between the data that reside at various sites can help in choosing this architecture. We are currently in the process of applying our design strategy to a real-life distributed database system with time-constrained transactions.

Acknowledgments

We would like to thank Jacob Aizikowitz, Brian Coan, and Susan Davidson for carefully reading an earlier draft of the paper, and suggesting several improvements in its presentation.

References

[BG86] P. A. Bernstein and N. Goodman. Serializability theory for replicated databases. *Journal of Computer and System Sciences*, 31(3):355–374, December 1986.

[BHG87] P. A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley, 1987.

[COK86] B. A. Coan, B. M. Oki, and E. K. Kolodner. Limitations on database availability when networks partition. In *Proceedings of the Fifth ACM Symposium on Principles of Distributed Computing*, pages 187–194, Calgary, Alberta, August 1986. ACM SIGOPS-SIGACT.

[Dav84] S. B. Davidson. Optimism and consistency in partitioned distributed database systems. *ACM Transactions on Database Systems*, 9(3):456–481, September 1984.

[ET89] A. El Abbadi and S. Toueg. Maintaining availability in partitioned replicated databases. *ACM Transactions on Database Systems*, 14(2), June 1989.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, 1979.

[GS90] D. Ghosal and A. Shah. A stochastic analysis of the performance of distributed databases with site and link failures. Technical Report TR 90-1071, Department of Computer Science, Cornell University, January 1990.

[Had84] V. Hadzilacos. *Issues of Fault Tolerance in Concurrent Computations.* Ph.D. dissertation, Harvard University, June 1984. Department of Computer Science Technical Report 11-84.

[Her86] M. Herlihy. A quorum consensus replication method for abstract data types. *ACM Transactions on Computer Systems*, 4(1):32–53, February 1986.

[MS90] K. Marzullo and A. Shah. Harary Networks: Connectivity for highly available real-time distributed databases. Technical Report TR 90-1070, Department of Computer Science, Cornell University, January 1990.