# Harary Networks: Architectural Support for Highly Available Distributed Databases

Amitabh Shah*
Keith Marzullo

TR 90-1070
May 1990

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

*Supported in part by a grant from Xerox Corporation.

# Harary Networks: Architectural Support for Highly Available Distributed Databases

(Preliminary Version)

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

January 18 1990
(Revised May 1 1990)

**Abstract**

A methodology, called *network designing by the desirable partition*, for designing underlying communication networks for distributed databases is proposed. It exploits the fact that in most real-life databases, the data is not fully replicated, and the transactions access pattern is highly local. The methodology consists of identifying a desirable partition of the sites based on the notion of *dependencies* between sites; the latter is defined by the replication of data and the transaction data access patterns. A hierarchical communication network, called a Harary Network, is then constructed for the identified partition. The notion of desirability takes into account the cost of connection, and thus provides the most desirable construction for a given cost. The method is probabilistic in the sense that in presence of failures, the probability of the occurrence of the desirable partition is higher than that of all other partitions; this results in very high *expected* availability. It is shown that for most intuitive formulations of the problem, finding the most desirable partition is *NP-Hard.* However, good and often optimal approximation algorithms exist for this problem. The methodology is particularly suited for designing communication support for real-time distributed databases.

CR Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**] Network Architecture and Design — *Distributed Networks*; *Network Topology*; C.2.4 [**Computer-Communication Networks**] Distributed Systems — *Distributed Databases*; C.3 [**Special-Purpose and Application-based Systems**] *Real-time systems*; C.4 [**Performance of Systems**] *Reliability, availability, and serviceability*

*Supported in part by a grant from Xerox Corporation.

Harary Networks: Architectural Support for Highly Available Database Systems

# 1 Introduction

*Availability*, as a measure of performance, is an important criterion in the design of fault-tolerant database systems [BHG87]. In this paper, we deal with the problem of designing an underlying communication network that helps maintain high data availability in a distributed database in spite of failures that lead to network partitions. We give a methodology for designing the underlying network that takes into account the transaction access patterns and the distribution of data in order to increase the chances of higher availability. This is done by identifying subsets within the set of sites that exhibit high interaction, and thus logically belong together.

In [COK86], Coan, Oki, and Kolodner explored the quantification of availability in such systems, and gave upper bounds on availability under simple assumptions (*full* replication of data, and *uniformity* of transaction access pattern) about the system. In [SM89], we extended the work of [COK86] to systems with more complex, and realistic assumptions and showed that in view of partitions, *partial* replication can achieve better availability than full replication can, if the set of transactions satisfies a *dominant local access* assumption (as opposed to the uniformity assumption). This assumption says that the access pattern of the transactions initiating at a site is highly skewed towards the local and neighboring data. With this assumption, which is usually satisfied by many real-life database systems, substantial availability can be achieved, even in the case of multiple partitions in the underlying network.

However, in certain pathological cases of partition due to failures, where the dominant local access assumption is not satisfied within every partition group, there may not be much gain in availability. The problem is that it is the partitioning that determines the data replication within each partition group, not vice versa. But, by the very nature of failures, it is not always possible that the system will partition such that the dominant local access assumption is satisfied within each group. However, it is possible to influence probabilistically the way the network will partition so that the chances of a good partition increase. In this paper, we investigate one such probabilistic solution.

Our approach is as follows. First, by examining the read and write patterns of a set of transactions, we identify a partition of the set of sites, called the *desirable partition*, such that each set in the partition exhibits a high level of transaction access activity within it, but little activity across the partition. Having found these sets, we define the connectivity of the network so that in case of failures, the chances that the network partitions according to the above partition increase. Specifically, in case of independent link failures,[1] the probability that the network partitions according to the desirable partition is more than the probability

[1] We consider only link failures here; the failure of a site also implies the failure of the links connecting that site, thus violating the independence of failure assumption.

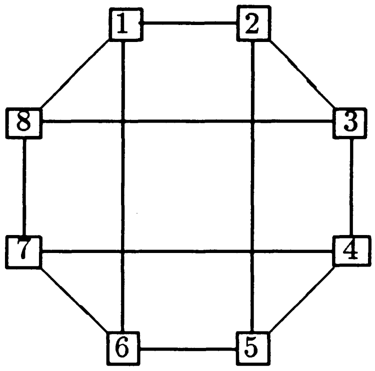


Figure 1: Uniform Connection

that any other partition occurs (with the same number of link failures). We call this approach *network design by the desirable partition*. We illustrate the flavor of this approach with the following example.

**Example:** Consider a database system consisting of 8 sites with 12 links allocated to connect them. One straightforward approach is to connect them uniformly so that each site has connectivity 3 (Figure 1).

Now consider the same system serving an intercontinental banking database with sites 1–4 in one continent and 5–8 in another. Since we expect fewer transactions across the continents than within, an appropriate configuration is as shown in Figure 2. Sites 1 and 8 could be the main offices in each continent and hence would serve as the gateways for other sites.

However, if the sites represent an airline reservation system, with sites 1 and 8 being two major hubs in the system, we expect a lot of (transaction) traffic between the two and hence

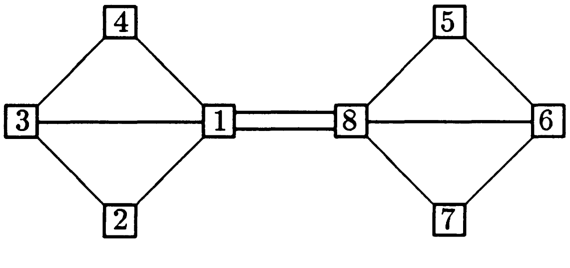


Figure 2: Banking Database

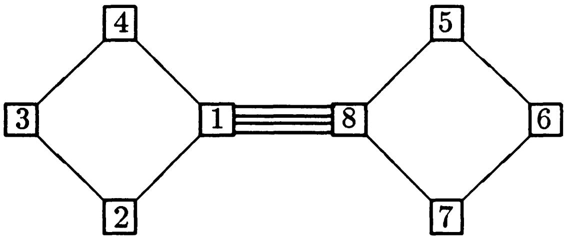


Figure 3: Airline Database

want them to be connected as much as possible. A possible connection scenario is shown in Figure 3.

Put another way, in case of partition due to independent link failures, in case of the banking database, it is desirable that sites 1–4, and 5–8 remain connected together, while in case of the airline database, sites 1 and 8 remain connected together. In the example above, the probability that sites 1 and 8 are partitioned in Figure 2 is 1/66, whereas in Figure 3 this probability is 1/495, assuming that each link is equally likely to fail. □

We believe that this approach to designing the communication network is suitable for databases where the placement of the data and the access patterns are relatively (statistically) stable. By designing, we do not necessarily mean constructing a brand-new communication network; it can also mean making decisions as to how many and which phone lines to lease, for example. We believe that the approach is particularly suitable for designing communication support for *real-time* databases; the latter are characterized by non-migratory data (usually corresponding to the positions of sensors and actuators in the system) as well as highly regular data access patterns, for example, patterns generated by the same physical process that runs periodically.

The remainder of the paper is organized as follows. The model of the system defined in Section 2, as well as a notion of dependencies between sites is developed. In Section 3, we give an algorithm for allocating links based on natural fault-tolerance requirements of the system. In Section 4, we show that for very intuitive definitions of a desirable partition, the problem of finding the most desirable partition is *NP-Hard.* We discuss several approximation algorithms that are the best possible in terms of the approximations they achieve, as well as in their running time. We conclude with a discussion on the applications of Harary Networks in Section 5.

## 2 Model of the System

We consider a *distributed database system* consisting of a set $\mathcal{S}$, of $n$ sites, connected by a *communication network.* The set of *data 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 correctness criterion for the protocol execution is assumed to be *one-copy serializability* [BG86].

The links in the communication network can fail by *crashing* [Had84]. When the failed links recover, they are integrated back into the network. 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.

The transactions in the system are of two kinds: *read* and *update.* We say that a transaction *accesses* a data object if it either reads or updates that object. Let $T$ be the set of transactions presented to the system (over a long period of time), and let a subset $T_c$ actually complete, the rest accounted for by partition failures. Then the *availability* for this system is defined as the proportion of $T_c$ in $T$ [COK86], i.e.,

$$\textit{availability} \triangleq \frac{|T_c|}{|T|}. \tag{1}$$

## 2.1 Replication of Data

Each data object is associated with a set of sites, called the *primary sites* for that object. For a data object $d \in \mathcal{D}$, 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 $j$ (but not local to all the sites) is denoted as $D_j$ and the set of data local to a set of sites $\mathcal{P}_j$ (but not local to all the sites) is denoted as $\mathcal{D}_j$. We assume that a data item is replicated only at its primary sites. There could be data items whose primary sites are the whole set of sites. We call such data *global* data and denote them as $D_g$; by definition they are replicated at every site in the system. Thus, $\mathcal{D} = \cup_{j \in \mathcal{S}} D_j \cup D_g$.

Thus, there are three kinds of data in the system from the point of view of a site $j$: *local*, *non-local*, and *global.* The first set is that which not global, and $j$ is one of the primary sites for that set. The second set is that which is also not global, but $j$ is not one of the primary sites for that set. In other words, the non-local data for $j$ is the union of the set of data that is local to other sites. The notion of local and non-local data can be easily extended to a set of sites.

The transactions initiated at a site can now be characterized by the type of data they access; they can be

- local transactions—those that access only data local to the site,
- global transactions—those that access global data, perhaps also with local and non-local data,

- non-local transactions—those that access only non-local data, and
- mixed transactions—those that access local data along with some non-local data (but not global data).

The idea behind defining the global transactions as above is that for a global transaction initiating at the site, the site may need to communicate with all the other sites in the system, in particular with the sites primary to other non-global data that may also be accessed.

## 2.2 Locality of Access

Next, we want to quantify the notion of the *locality* of access in such a system. For a subset $X$ of $\mathcal{D}$, we denote by $T_X$ the set of non-global transactions that accesses data objects in $X$, and by $T^j$ the set of transactions that are initiated at site $j$. Thus, $T^j_X$ is the set of non-global transactions initiated at $j$, accessing data in $X$. The set of global transactions is denoted as $T_g$.

Define the *local load* $\lambda_j$ of a site $j$ as the proportion of the transactions initiated at site $j$ that access data local to $j$, among all the non-global transactions that access data local to $j$. Mathematically:

$$\lambda_j \triangleq \frac{|T^j_{D_j}|}{|T_{D_j}|}. \tag{2}$$

Note that the local load includes the transactions that access only local data and the transactions that access local data with other kinds of data (from the set of mixed transactions). Also, the transactions in $T_{D_j} \setminus T^j_{D_j}$ are the non-local accesses at $j$ from other sites.

The notion of local load is extended 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 $\Lambda_i$ 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 $\mathcal{P}_i$:

$$\Lambda_{\mathcal{P}_i} \triangleq \frac{|T^{\mathcal{P}_i}_{\mathcal{D}_i}|}{|T_{\mathcal{D}_i}|}, \quad \begin{array}{ll} \text{where} & T^{\mathcal{P}_i}_{\mathcal{D}_i} = \bigcup_{k\in\mathcal{P}_i}\bigcup_{j\in\mathcal{P}_i} T^k_{D_j}, \\ \text{and} & T_{\mathcal{D}_i} = \bigcup_{k\in\mathcal{S}}\bigcup_{j\in\mathcal{P}_i} T^k_{D_j}. \end{array} \tag{3}$$

Our particular interest in in the notion of *dominant local access*. This assumption says that the distribution of transactions is skewed towards accessing local and neighboring data rather than distant ones. Assumption of dominant local access implies that $\lambda_j \approx 1$ for every site $j$, and that there exists a partitioning $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k)$ of the set of sites $\mathcal{S}$, such that $\Lambda_i \approx 1$ for every set $\mathcal{P}_i$ in $\mathcal{P}$. Note that the local loads of sites need not be close to 1; in fact, in a partition satisfying dominant local access, the local load of every group will be more than the local loads of the sites in the group. The underlying assumption here is that

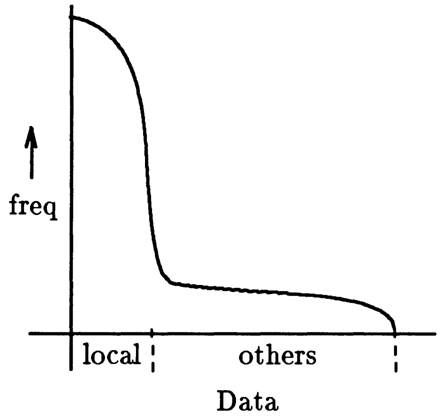


Figure 4: Locality of Data Access

for the partition satisfying dominant local access, there exists *mutual dependency* among the sites in every group.

Similarly, we define the *global load* of a site $j$, denoted as $\gamma_j$, to be the proportion of the number of global transactions initiating at $j$ among all the global transactions in the system:

$$\gamma_j \triangleq \frac{|T_g^j|}{|T_g|}. \tag{4}$$

As above, we can extend the definition of global load to a set of sites and denote it as $\Gamma_{\mathcal{P}_i}$.

By the assumption of dominant local access, it is implied that the set of transactions initiating at a site is dominated by the local transactions. Thus, we expect the frequency of data accesses at any site to be as shown in Figure 4.

## 2.3 Dependencies between Sites

Next, we want to define the notion of dependencies between sites. This idea is central to finding the desirable partition of sites. Roughly speaking, we would like to identify a clustering of the sites such that the sites within every cluster depend on each other much more than sites across the clusters. We formalize this idea as follows.

Define the *dependency* of a site $i$ on a site $j$, denoted as $dep(i,j)$, to be the proportion of the transactions initiating at $i$, accessing data at $j$ among all the transactions accessing data at $j$:

$$dep(i,j) \triangleq \frac{|T_{D_j}^i|}{|T_{D_j}|} + \gamma_i. \tag{5}$$

The value of $dep(i,j)$ will be in the range $[0,2]$, although, note that if a value $dep(i,j)$ is 2 then all the global transactions in the system must initiate at $i$ and hence the global load of every other site must be 0; thus all the other $dep(i,j)$ values can be at most 1. Finally, note that $dep(i,i)$ is simply $\lambda_i + \gamma_i$. Dependency captures the idea that $i$ needs to communicate with $j$ to complete these transactions.

This definition can be extended to sets of sites as:

$$dep(\mathcal{P}_i, \mathcal{P}_j) \triangleq \frac{|T_{\mathcal{D}_j}^{\mathcal{P}_i}|}{|T_{\mathcal{D}_j}|} + \Gamma_{\mathcal{P}_i}. \tag{6}$$

Again note that $dep(\mathcal{P}_i, \mathcal{P}_i)$ is $\Lambda_{\mathcal{P}_i} + \Gamma_{\mathcal{P}_i}$. We also refer to this as the *inter-dependency* of $\mathcal{P}_i$; it captures the idea of how much the sites in a set depend on each other. We want to use this to define the desirable partition, which, roughly speaking, is the partition of the set of sites such that the inter-dependency of each group in the partition is high, and there is little interaction across the groups.

Finally, we define the *dependency matrix* for the set of sites $\mathcal{S}$ to be an $n \times n$ matrix with the $(i,j)$th entry being $dep(i,j)$. This matrix can be determined from the description of the read and the write sets of each transaction. For most real-life applications, this matrix will be an almost symmetric matrix, reflecting the mutual dependency of the sites given by the dominant local access assumption. In practice, the matrix will be determined from a statistical description of the transaction access patterns, which we assume is known and is an input to the problem.

# 3 Harary Networks

Before we define the notion of desirability, we describe next the notion of a Harary Network. Clearly, it is undesirable that the network partition at all—the whole system, if connected, attains complete availability. Thus, the lowest probability of partitioning occurs when the underlying network is completely connected.[2] In presence of failures however, even this network can not achieve complete availability, although it will achieve a high *expected* availability, indeed higher expected availability than any other network design. There is a cost involved with this benefit—that of the number of links being $n(n-1)/2$ for a system with $n$ sites.

Thus, the idea is to incorporate the cost of connectivity while defining the most desirable partition, since in the absence of cost constraints the question of such a partition is moot. We would like to investigate a design of the network that uses far fewer links, and still attains *comparable* expected availability. There are several versions of this problem that are of interest. For example, given a fixed number or cost of links, design the network with the highest expected availability. Or, given a desirable level of expected availability, construct a network with as few links as possible. We consider the former version in this paper.

[2]We assume for now that there are no multiple links between the sites.

Let $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k)$ be a partition of the set of sites $\mathcal{S}$. We define the communication network by defining the connectivity of each group $\mathcal{P}_i$, and the connectivity of the *hyper-graph*—the one connecting all the groups together. We call such hierarchically connected network a ***Harary Network*** (HN).

In the general case, the Harary Network is defined as follows: Let $|\mathcal{S}| = n$, and $|\mathcal{P}_i| = n_i, i = 1, \ldots, k$ (so $\sum_{i=1}^{k} n_i = n$). Define the *fault-tolerance* of this system to be a $(k+1)$-tuple $(t, t_1, t_2, \ldots, t_k)$ where $t < k$, and $t_i < n_i, i = 1, \ldots, k$. Each $t_i$ is a resiliency requirement of the partition group $\mathcal{P}_i$, viz., that in spite of any set of $t_i - 1$ link failures in $\mathcal{P}_i$, the set of sites $\mathcal{P}_i$ should remain connected. In this case, we say that $\mathcal{P}_i$ is *$t_i$-resilient.* The number $t$ is the resiliency requirement of the hyper-graph, i.e., the hyper-graph remains connected in spite of any set of $t - 1$ link failures.

From graph theory, we know that the necessary condition for an $n$-node graph to be $t$-resilient is that the graph have at least $\lceil nt/2 \rceil$ links [BM76]. In fact, these numbers are tight—Harary showed that one can construct graphs with $n$ nodes and exactly $\lceil nt/2 \rceil$ edges such that the graph is $t$-resilient [Har62]. There is a simple algorithm to define the connectivity of such graphs [BM76]; we denote such graphs as $\boldsymbol{H}_{n,t}$. Thus, to define the HN for the partition $(\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k)$ of $\mathcal{S}$, we construct $k$ graphs $\boldsymbol{H}_{n_i,t_i}$ $(i = 1, \ldots, k)$, and construct a hyper-graph $\boldsymbol{H}_{k,t}$ to connect these $k$ graphs. The number of links needed for this construction is $\sum_{i=1}^{k} \lceil n_i t_i/2 \rceil + \lceil kt/2 \rceil$.

Some special cases of a Harary Network are:

- A *tree* HN, in which a spanning tree is constructed for each partition group; a spanning tree is also constructed for the hyper-graph. The fault-tolerance of the tree HN is (1, 1, ..., 1) (i.e., no failures can be tolerated) and the number of links needed are $n - 1$ (for a system of $n$ sites).

- A *ring* HN, in which each partition group is arranged as a ring; the hyper-graph is also a ring in this case. The fault-tolerance of the ring HN is (2, 2, ..., 2) and the number of links needed are at least $n$ and at most $n + k$ (for a system of $n$ sites and $k$ groups).

- A *complete* HN, in which each partition group is a completely connected graph, and so is the hyper-graph. The fault-tolerance of the complete HN is $(k-1, n_1 - 1, \ldots, n_k - 1)$ and the number of links needed are $\frac{1}{2}[\sum_{i=1}^{k} \lceil n_i(n_i - 1) \rceil + \lceil k(k-1) \rceil]$ (for a system of $n$ sites and $k$ groups).

One can also construct *hybrid* HN's, where the connectivity of the hyper-graph is defined differently from those of the partition groups. Indeed, real-life networks would have different kinds of connectivity from one group to another. In general, the design would be application dependent, and also governed by other factors, such as the cost of connecting sites, the physical proximity of the sites and so on. For example, if all the sites within a group reside within a small geographical area such as a building, or a campus, they could be connected via a broadcast network, such as Ethernet. However, to get a handle on comparing link costs, we consider only point-to-point connections in this paper.

### 3.1 Cost of a Harary Network

Define the *cost* $C(\mathcal{P})$ of the partition $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k)$ to be

$$C(\mathcal{P}) \triangleq \sum_{i=1}^{k} C_g(\mathcal{P}_i) + C_h(\mathcal{P}), \tag{7}$$

where $C_g(\mathcal{P}_i)$ is the cost of the links needed to connect the partition group $\mathcal{P}_i$ according to its fault-tolerance requirements, and $C_h(\mathcal{P})$ is the cost of the connection in the hyper-graph for $\mathcal{P}$. Thus, the cost of the partition is the cost of connecting the sites into the specified Harary Network. Note that the cost of connection need not simply be the number of links required for the fault-tolerance requirement; it can also incorporate other factors like the length of the link, whether it is dedicated or shared, and so on.

From the point of view of the database applications, especially those with the dominant local access patterns, it makes sense to design Harary Networks that have a high level of connectivity within the groups, and sparse connectivity across the groups. This is in conformity with our objective of maximizing the probability of communication within the groups, at the expense of providing support for transactions that access data in other groups; dominant local access implies that there are very few of the latter. Thus, in this paper, we will assume that hybrid HN's are constructed, where the hyper-graph connectivity is much sparser than the group connectivity. In fact, we will assume that the cost of the hyper-graph connection is asymptotically vanishing in comparison to the cost of connecting the groups. The complete-ring hybrid HN is one such example, where the groups are completely connected, and the hyper-graph is a ring. Such a HN, with 5 groups and 32 sites is shown in Figure 5.

Thus, we would like the HN to have the property that the cost of merging two groups into one is more than the savings in the hyper-graph connection due to this merger. Another way of saying this is that the total cost of connection is monotonically increasing as the database sites are merged into fewer and fewer groups. None of the uniform HN's have this property. The tree HN requires the same number of links $(n-1)$, no matter how many groups. The ring HN requires the fewest number of links in the extremal cases—when there is exactly one group, or when there are $n$ groups—$n$ links in both the cases. (There are other configurations also where a ring HN needs exactly $n$ links.) In general, a ring HN with $k$ groups, $2 < k < n/3$ where each group has at least 3 sites, uses $n + k$ links, and thus has exactly the reverse of the desired property. The complete HN requires the most number of links at these two extreme cases ($n(n-1)/2$ links). It is easy to construct examples where the above property is violated in a complete HN.

A good rule of thumb in designing HN's that have this property is to choose $t$, the resiliency of the hyper-graph to be much smaller than each $t_i$, the resiliencies of the groups. This is in conformity with our requirement of the HN as well: we need to support much higher level of interaction within the groups than across the groups.

Finally, note that we have assumed that the resiliency $t$ of an $n$-site group is less than $n$. If we would like to allow $t \geq n$, we must allow multiple links between sites. For $t = n$, it can

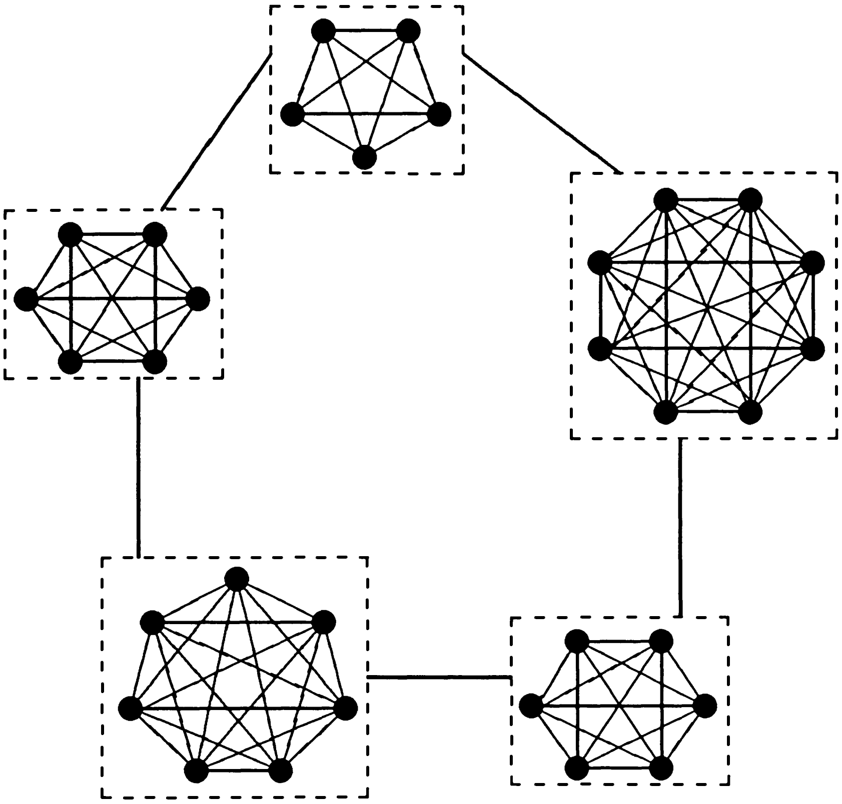

Figure 5: A Complete-Ring (Hybrid) Harary Network

be easily seen that the number of extra links required for an $n$-site group is $\lceil n/2 \rceil$. Similar results can be derived for other instances of HN's. In the next section, we show how to derive the most desirable partition of the database sites.

# 4 Defining Desirable Partitions

It is clear that the most desirable system state is that the whole set $\mathcal{S}$ remains connected, since in the case of no partitions, all the transactions in the system would complete and the availability would be 1. Clearly, this state is not always possible in view of unreliable links. Thus, we are interested in the case when the system can partition into at least two groups. In general, we would like to identify a partition $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k)$ of $\mathcal{S}$ with maximum *expected* availability. However, to find the expected availability, we need to find the mean probability that a transaction successfully completes. This computation, is a function of the

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

(a) (b)

Figure 6: Matrix Clustering

replica control protocol that is being used, while we need to formalize a notion of desirability that is independent of the protocol being used. In [SG90,Sha90], we show how to compute the expected availability for specific replica control protocols, using stochastic methods.

We suggest below several objective functions that are intuitive, and are independent of the protocol. However, it turns out that for these functions, indeed for a fairly general class of such functions, the optimization problem is *NP-Hard.* However, good and often provably optimal approximation algorithms exist for such problems. We will use these algorithms to find an approximation to the most desirable partition.

## 4.1 Clustering the Dependency Matrix

The first approach is based on the idea of clustering a matrix of data [MSW72,LR75]. The idea is that given a matrix $A$, where $a_{ij}$ measures the strength of the relationship between row $i$ and column $j$, the problem is to find a row permutation and a column permutation so that the resulting matrix obtained by applying these permutations on the original matrix shows the clarity of the relationship. For example, the matrix in Figure 6(a) is transformed to the one in Figure 6(b) by the the row permutation $((1,1)(2,3)(3,5)(4,2)(5,4))$ and the column permutation $((1,1)(2,3)(3,2)(4,4))$.

In our case, we have a very natural definition of the matrix to be clustered: the matrix of dependency relations, which is an input to the problem of finding the most desirable partition. The output of the approximate algorithm would determine a clustering of the set of sites that is the best possible. Let $k$ be the *order* of the clustering—the number of distinct clusters we can identify from the transformed matrix.

While the matrix in the example was a boolean one, the dependency matrix will have real values from the interval [0,2]. This gives us two options to formulate the problem. The first is to find the best possible clustering of the dependency matrix as it is. The second is to transform the matrix into a boolean one by defining a *threshold* value. An entry in the transformed matrix will be 1 if an only if the corresponding value in the dependency matrix was greater than the threshold value. The higher the threshold value, the weaker the strength of the relation will be, and thus, the clustering will be of a higher order. This option allows us to bring the cost of the partition into the picture. If a particular clustering results in a cost higher than that allowed, we can solve the problem again with a lower threshold value.

Note that there will be certain restrictions on the threshold value. Firstly, the threshold value can not be greater than the maximum value in the dependency matrix, else all the entries in the transformed matrix will be zero. Similarly, if the threshold value is smaller than the smallest value in the matrix, the transformed matrix will only have 1's, and the whole system will be just one cluster. Finally, the threshold value can not be higher than the minimum of the $n$ diagonal values, $dep(i,i)$'s, since a diagonal value being zero would imply that a site is not clustered with itself, an absurdity.

It is easy to see that the threshold values have the following property: if two threshold values result in the same partitioning of the sites, then any value between those two values will also result in the same partitioning. This fact can be exploited to implement an algorithm finds the best clustering within the cost in at most $\log n$ applications of the matrix clustering procedure. Note that if the transactions in the system are mostly global, all the entries in the dependency matrix would be approximately the same and high. This would result in extreme clustering—either one cluster or $n$ clusters—based on the threshold values. For most threshold values, this would give one cluster of the sites which is in conformity with the requirement that the sites be able to communicate with each other for most transactions.

Let us see an objective function for the clustering problem proposed by McCormick *et al* [MSW72]. Let $A$ be the given matrix of dependencies (whether the original one, or the transformed boolean one). Let $\rho$ and $\sigma$ be permutations of the row and column indices, i.e., the $(i,j)$th entry of the transformed matrix is the $(\rho(i),\sigma(j))$ the entry of the original matrix. Create a dummy row 0 and a dummy column 0, each composed of 0's, and assume that $\rho(n+1)=\sigma(n+1)=\rho(0)=\sigma(0)=0$. Define the *desirability* of a pair of permutations $(\rho,\sigma)$, denoted $des(\rho,\sigma)$, to be

$$des(\rho,\sigma) \triangleq \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{\rho(i)\sigma(j)}\left(a_{\rho(i)\sigma(j-1)} + a_{\rho(i)\sigma(j+1)} + a_{\rho(i-1)\sigma(j)} + a_{\rho(i+1)\sigma(j)}\right). \tag{8}$$

In other words, $des(\rho,\sigma)$ is the sum of the products of the adjacent elements in the permuted matrix. Then the problem is to find a pair of permutations such that the permutation of the dependency metric has the highest desirability. For example, for the matrix in Figure 6(a), the desirability is 0, while for that in Figure 6(b), the desirability is 11.

It is known that the data clustering problem is *NP-Hard*, and it can be modeled as a *traveling salesperson problem* (TSP) [Len74,LLRS85]. In fact, Equation 8 reduces to

$$\sum_{i=0}^{n}\sum_{j=1}^{n} a_{\rho(i)j}a_{\rho(i+1)j} + \sum_{j=0}^{n}\sum_{i=1}^{n} a_{i\sigma(j)}a_{i\sigma(j+1)}, \tag{9}$$

and thus the problem of maximizing $des(\rho,\sigma)$ decomposes into two optimization problems $des(\rho)$ and $des(\sigma)$. Each of these problems can be modeled as a TSP problem, and thus the techniques for finding approximate solutions for TSP apply here.

In our case, since the dependency matrix is almost symmetric, it is easy to see that the row permutation will be usually the same as the column permutation. But, in this case, we still have one optimization problem to solve (which will be modeled as TSP), thus the complexity of the problem remains the same. There is a plethora of literature on approximate algorithms for TSP. See the treatise by Lawler *et al* [LLRS85] for a detailed exposition on the properties of TSP, as well as on approximations to it.

## 4.2 Clustering to Minimize Inter-Cluster Distance

Another approach to solve the desirable partition is based on the *clustering* problem [Gon85, HS86]. Formally the $k$-clustering problem is the problem of finding a partitioning of $n$ sites into $k$ groups such that some objective function is optimized. It comes in several versions:

1. Given a complete graph of $n$ points, with positive real weights associated with every edge in the graph, an integer $k$ and a real number $d$, does there exist a $k$-clustering of the $n$ points such that the maximum weight of an edge in any cluster (the diameter) is less than or equal to $d$?

2. Given $n$ points in a Euclidean space, a distance metric on that space, and an integer $k$, find a $k$-clustering that minimizes the maximum of the $k$ cluster diameters.

The first problem above is the decision problem (we want to find *if* there exists a clustering), while the second is an optimization problem (find the best clustering).

The reason for using the clustering problem is that besides being closely related to the problem at hand, approximate algorithms are available that are *best possible* in two senses: firstly, it is known that it is not possible to find polynomial time solutions that are better than some factor of the optimal solutions (unless $P = NP$), and there are algorithms that achieve this approximations. Secondly, there exists optimal algorithms to find such approximate clustering: Feder and Greene [FG88] show that any algorithm to find an approximate $k$-clustering of $n$ points takes at least $O(n \log k)$ time; they also demonstrate an algorithm that matches this lower bound.

We can use the approximate clustering algorithms once we define an appropriate distance metric. Below, we propose two such metrics that are intuitive for the application at hand. They are based on the idea that the distance between two sites should be inversely related to the dependency between them:

1. For $n$ sites $s_1, s_2, \ldots s_n$, consider an $n$ dimensional space. The coordinates of the site $s_i$ are defined as

$$c(s_i) \triangleq \left(\frac{1}{dep(s_i,s_1)}, \frac{1}{dep(s_i,s_2)}, \ldots, \frac{1}{dep(s_i,s_n)}\right). \tag{10}$$

The distance metric is the standard $L_2$ metric defined on this $n$-dimensional space. It is easy to see that this metric satisfies the property above.

2. Here the idea is to directly define the distance between two sites $s_i$ and $s_j$ as:

$$d(s_i, s_j) \triangleq \frac{1}{(dep(s_i, s_j) + dep(s_j, s_i))}. \tag{11}$$

Again, the higher the dependency between two sites, the closer they will be. Note that it is possible that the space defined by this distance metric is non-Euclidean; in particular, the triangle inequality for the distances between three points need not hold. However, we are interested in systems with the dominant local access assumption. This says that there exists a partitioning of the set of sites such the sites in each group are *mutually* dependent. Thus, if $dep(s_i, s_j)$ is high (low) then we also expect $dep(s_j, s_i)$ to be high (low). Moreover, if $dep(s_i, s_j)$ is high and $dep(s_i, s_k)$ is low (also high) then we expect $dep(s_k, s_j)$ also to be low (high).

These observations give us a relationship that closely resembles the triangle inequality; in fact, the stronger the nature of the dominant local access in a system, the better triangle inequality we will have. This also determines the strength of the approximation algorithm: most approximation algorithms for the $k$-clustering problem give solutions that are at most a factor of 2 from the optimal answers. The latter factor comes from a basic lemma that states that in a triangle formed by any three points, the longest side is at most twice the maximum of the remaining two sides. If the dominant local access assumption implies a better bound on the relative distances between the points in the graph, we can get better approximations to desirability.

Note that this approach is simply another variant of the matrix clustering problem: we can consider clustering the (symmetric) *distance* matrix with the strength of the $(i, j)$th entry being inversely related to its value.

Clearly, the problem of finding the most desirable partition would be expressed as the optimization problem described above, where we want to minimize the maximum of the $k$ cluster diameters. See [Gon85,FG88] for a description of the algorithms for this problem.

Unlike the matrix clustering approach, where the order of the cluster is determined by the algorithm, the clustering algorithms need the cluster order as an input. In this case also, however we can use binary search to determine the best clustering allowable within the cost. We start with $k = n/2$ and depending on whether the Harary Network for the cluster produced exceeds the cost or not, increase or decrease the cluster size. Again, within $O(\log n)$ iterations of the clustering algorithm, we can determine the best clustering within the cost.

To summarize the approaches above, we can either use the matrix clustering approach to cluster the dependency matrix to find the approximate desirable partition. We can use the threshold value approach to transform the dependency matrix to a boolean matrix first. In this case, we can find the clustering that best fits the cost constraints by a binary search on the threshold values. To use the $k$-clustering approach, we use a binary search on the cluster size and obtain the best approximation to the desirable partition within the allowable cost. To illustrate the above concepts, consider the following example.

| *site x* | $dep(x, s_1)$ | $dep(x, s_2)$ | $dep(x, s_3)$ | $dep(x, s_4)$ | $dep(x, s_5)$ | $dep(x, s_6)$ |
|---|---|---|---|---|---|---|
| $s_1$ | 0.47 | 0.37 | 0.39 | 0.20 | 0.18 | 0.36 |
| $s_2$ | 0.37 | 0.50 | 0.33 | 0.19 | 0.22 | 0.42 |
| $s_3$ | 0.35 | 0.37 | 0.48 | 0.23 | 0.21 | 0.40 |
| $s_4$ | 0.21 | 0.20 | 0.20 | 0.60 | 0.59 | 0.19 |
| $s_5$ | 0.22 | 0.21 | 0.22 | 0.55 | 0.58 | 0.19 |
| $s_6$ | 0.38 | 0.35 | 0.38 | 0.23 | 0.22 | 0.44 |

Figure 7: Example Dependencies

**Example:** The database consists of six sites: $\mathcal{S} = \{s_1, \ldots, s_6\}$. Assume that cost function is simply the number of link s needed to connect the sites, and the Harary Network to be constructed is a complete-ring hybrid network. There are 10 links available to connect the sites. The global loads of all the sites are assumed to be the same (0.167 each). The dependencies between the sites is given by the table in Figure 7.

A threshold value can range between 0.18 (minimum of the values) and 0.44 (minimum of the diagonal values). Any threshold value of 0.18 will give a transformed dependency matrix with all entries being 1, thus giving exactly one group in the Harary Network. Since the cost of this network would be 18, it is unacceptable. A value of 0.44 gives the clustering as $\mathcal{P}_1 = (\{s_1, s_2, s_3\}, \{s_6\}, \{s_4, s_5\})$; this results in a Harary Network of cost 7 which is allowable, but not the best one. Further binary search gives the best clustering as $\mathcal{P}_2 = (\{s_1, s_2, s_3, s_6\}, \{s_4, s_5\})$; this has a cost of 9 which is within the cost.

Thus, the best partition is $\mathcal{P}_2$ and we need 9 links to connect the sites in it. We allocate the remaining link to the second group in $\mathcal{P}_2$ since that will increase the expected availability of the system the most, as can be easily checked. The resulting Harary Network is as shown in Figure 8. □

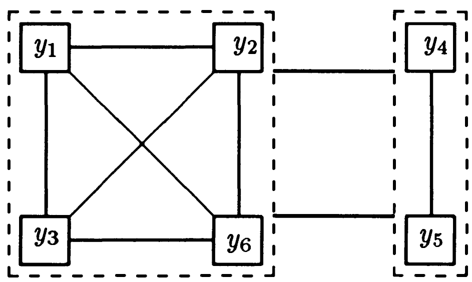


Figure 8: Harary Network for Example dependencies

### 4.3 Generalizing the dependency relations

So far, we have computed the dependencies between sites only using the number of transactions that initiate at the sites. We can easily generalize this to incorporate some of the complexities of real-life distributed database systems. For instance, the transactions in the system may have priorities; the availability of data to higher priority transactions may be more crucial than to lower priority transactions. Also, the length of the links between sites are not the same. Thus, the cost of communication between sites, hence the cost of non-local transactions, may be different. Finally, not all the sites in the system have the same load. Thus, for instance, the cost of establishing communication with a site that has heavy traffic, for example a network gateway, may be more than the cost of communicating with a site with low load.

All these factors can easily be incorporated in the above model of dependencies by considering a weighted sum of the transactions in Definition 5. The *weight* of a transaction would be a function from the set of transactions to, say, the set of natural numbers. The weight would account for the priority of the transactions, the cost of communication, as well of the nature of the sites involved in the communication. Then the desirable partition can be found using these weighted dependencies. The above is a very general model of dependency: it can incorporate complex interactions between the nature of transactions, and the nature of communication.

## 5 Discussion and Applications of Harary Networks

We have presented a practical methodology for designing the underlying communication network for distributed database systems. The method is geared towards providing very high availability in spite of partitioning of the network. This is done by ensuring that in case of independent failures, the network is more likely to split up into groups that are desirable as partition groups, since they exhibit high inter-dependency within each group but very low dependency across the groups. We believe that constructing highly available systems requires this kind of architectural support that is customized for the application at hand.

We have considered only link failures in this paper; as mentioned in section 1, a site failure also implied the failure of the links connected to that site. However, note that if we are interested in just the site failures, our results are still applicable: the Harary graphs constructed in the previous section are also $t$-resilient for $t$ independent site failures. This is because for any graph, the vertex connectivity is less than or equal to the edge connectivity.

Harary Networks, with their definition of data replication, provide a very general model of a distributed database. By choosing the parameters appropriately, one can model a broad spectrum of system architectures and replication schemes. Below we list some of the applications where the notion of Harary Network is particularly suitable.

## 5.1 Real-time Databases

As we mentioned in the introduction, we believe that this method is suited for designing underlying networks for real-time databases. However, to analyze the expected availability, we have to formalize the notion of availability for such systems; in particular, incorporate the transaction deadlines and priorities. We are currently exploring such a definition of real-time availability.

We have also developed a stochastic model to analyze the dynamic behavior of a distributed database [SG90, Sha90]. This is a very general model that takes as input the information about replication, transaction access patterns, transaction arrival and service rates, and the rates of site and link failures, and predicts the system performance for two key parameters—availability and response time degradation. We are currently extending that model to incorporate real-time deadline and priorities as well.

## 5.2 Group Paradigm for Concurrency Control

In [ET89] El Abbadi and Toueg propose the *group paradigm* protocol for the management of replicated data. Their protocol consists of a division of the set of transactions into a number of groups. The concurrency control is achieved by first achieving serializability within each group, and then providing serializability across the groups. The result is a highly robust and flexible protocol for replica management. In fact, they show that several well-known protocols for managing replicated data are instances of the group paradigm, with suitable values of the group parameters.

Harary Networks appear to give a very natural definition of the transaction groups in their paradigm: the transactions initiating in a partition group form a transaction group. With the knowledge of data replication within the groups and across the groups, the concurrency control protocols can be customized for improved availability and response times.

## 5.3 Optimistic Concurrency Control

We also believe that Harary Networks make excellent candidates for the optimistic concurrency control protocol proposed by Davidson [Dav84]. This protocol, as the name suggests, starts with the assumption that inconsistent updates across a partition in a distributed database are relatively rare. Thus the protocol, instead of providing absolute correctness of transaction execution (via serializability) provides a rollback mechanism to undo inconsistent updates. The advantage is that for most of the transactions, the response time is good. Harary Networks provide an additional degree of optimism for database operations, since the chances of a group partitioning are much less than two groups partitioning from each other. Since each group exhibits very high level of access within it, by using optimistic concurrency control, the response times of transactions can be further improved.

Finally, Harary Networks provide a good model for database applications that exploit the knowledge of replication. One such example is that of Epidemic Algorithms [DGH+87] for maintaining replicated data, implemented at Xerox. Some of the algorithms used there can benefit from the knowledge of data replication and connectivity of a Harary Network.


## Acknowledgement

Jacob Aizikowitz and Pat Stephenson discussed the problem with us and provided useful suggestions. We also thank David Shmoys for a discussion on the clustering algorithms.


## References


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

[BHG87] Philip A. Bernstein, Vassos Hadzilacos, and Nathan Goodman. *Concurrency Control and Recovery in Database Systems.* Addison-Wesley, 1987.

[BM76] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications.* North Holland, 1976.

[COK86] Brian A. Coan, Brian M. Oki, and Elliott 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] Susan B. Davidson. Optimism and consistency in partitioned distributed database systems. *ACM Transactions on Database Systems*, 9(3):456–481, September 1984.

[DGH+87] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the Sixth ACM Symposium on Principles of Distributed Computing*, pages 1–12, Vancouver, British Columbia, August 1987. ACM SIGOPS-SIGACT.

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

[FG88] Tomás Feder and Daniel Greene. Optimal algorithms for approximate clustering. In *Proceedings of the Twentieth ACM Symposium on Theory of Computing*, Chicago, Illinois, May 1988. ACM SIGACT.

[Gon85] Téofilo Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[Had84] Vassos Hadzilacos. *Issues of Fault Tolerance in Concurrent Computations.* PhD thesis, Harvard University, June 1984. Department of Computer Science Technical Report 11-84.

[Har62] Frank Harary. The maximum connectivity of a graph. In *Proceedings of the National Academy of Sciences*, volume 48, pages 1142–46, 1962.

[HS86] D. S. Hochbaum and D. B. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *Journal of the ACM*, 33(3):533–550, 1986.

[Len74] J.K. Lenstra. Clustering a data array and the Traveling Salesman Problem. *Operations Research*, 22:413–414, 1974.

[LLRS85] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, editors. *The Traveling Salesman Problem: A guided tour of combinatorial optimization.* John Wiley and Sons, Chichester, 1985.

[LR75] J. K. Lenstra and A. H. G. Rinnooy Kan. Some simple applications of the Traveling Salesman Problem. *Operations Research Quarterly*, 26:717–733, 1975.

[MSW72] W.T. McCormick Jr., P. J. Schweitzer, and T. W. White. Problem decomposition and data reorganization by a clustering technique. *Operations Research*, 20:993–1009, 1972.

[SG90] Amitabh Shah and Dipak Ghosal. A stochastic analysis of the performance of distributed databases with site and link failures. Technical Report TR 90-1072, Department of Computer Science, Cornell University, January 1990.

[Sha90] Amitabh Shah. *Exploiting Trade-offs in the Design of Fault-tolerant Distributed Databases.* PhD thesis, Department of Computer Science, Cornell University, August 1990. In preparation.

[SM89] Amitabh Shah and Keith Marzullo. Trade-offs between replication and availability in distributed databases. Technical Report TR 89-1065, Department of Computer Science, Cornell University, December 1989.