Increasing Availability in Replicated Distributed Database Systems

S.Daneshi

Member of academic board

Ministry of Culture and Higher Education

A.Abdollahzadeh-Barfoursh

Assistant professor

Computer Eng.Dept.

Amirkabir Technical University

M.R.Meybodi

Associate professor

Computer Eng.Dept.

Amirkabir Technical University

Abstract

A distributed database system consists of sites and communication links. Both of these two components are liable to failure. Communication link failure results in network partitioning. Network partitioning is a serious threat to the availability of replicated data. When new partitions are created, then the next question is which partition is executing user transactions. Each partition has to decide individually whether it is capable of performing operations on user transactions. We propose a new protocol which provides high availability. In the pessimistic approach only one partition is able to execute user transactions. Hence, our protocol should be able to choose the partition. Our approach uses a dynamic voting protocol but with different methods. Each site has information about its neighbour links, status of other sites and the connections between sites and links. We show that our protocol will provide more availability than other protocols. Our model has been simulated and compared with a simulation of other conventional protocols, and the results bear out the conclusions of this study.

Keywords: Distributed database systems, availability, consistency, partitioning, general dynamic.

1 Introduction

A *distributed system* is a set of sites connected by a *communication network*. A *site* or a *node* is an autonomous computer system with its own processing unit and a storage (volatile and stable) device. Sites can communicate with each other solely by sending messages through a communication network. Sites are responsible for processing information and the communication link transfers information between different sites. A *database* is a collection of data. A *distributed database* is a collection of data which is distributed over a number of sites through a communication network. Because the architecture of a distributed database system varies from one system to another, we have chosen as abstract view of a distributed database system.

Each site has a site-identifier number, where these numbers form a total order. When sites fail, their failure is *clean*, i.e., they stop running. Link failures result in *network partitioning* [4, 5], and cause sites to communicate with each other only if they are in the same partition, but not with the sites in other partitions.

After the creation of a new partition, the next question is which partition can execute user transactions. If there exists a majority partition, then this is the partition that performs operations. But if there is no majority partition, then it is desirable to increase the probability of having at least one partition capable of performing operations. The decision of whether a partition can perform transactions is the key issue for the availability of our protocol. Since different partitions cannot communicate with each other, each partition has to make the decision regarding the execution of transactions with the information available inside the partition. Hence, each partition should make the decision individually.

In this paper we propose our protocol for dynamic voting. Our protocol is different and unique in the way it makes its decision about the majority partition. Our protocol keeps track of the way different partitions merge or depart from each other, by keeping enough information at each site. When partitioning occurs, each partition can decide if it is capable of executing user transactions or not. The decision can become very complicated when several groups exist in a network and none of them is capable of performing operations. In this paper, we propose a new protocol that increases the probability that a partition can accept operations from users and hence increases data availability.

2 Dynamic Voting Protocol

In other dynamic voting protocols (Davcev and Burkhard [3], Davcev [2], Jajodia and Mutchler [7, 8, 6]), a partition is considered as a majority partition if it has a majority of the sites of its previous majority partition. By contrast, in our proposed scheme a decision is made about the partition that can execute transactions according to the configuration of the system. One-copy serialisability is obtained because only one partition is allowed to operate transactions.

2.1 Data Structures

Each site has information about the site's link connections with other sites. For example, in Figure 1, Site A, denoted as AM_A, is aware that site A has links with nodes

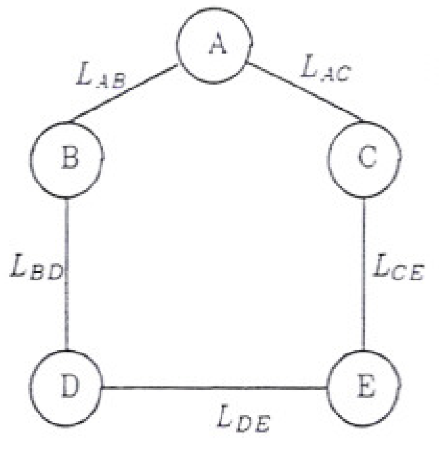

Figure 1: Network Connection

B and C. When AM_A finds out that the link between sites A and C has failed, it records this failure and passes this information to another node. In our protocol, we are only concerned that two sites can send messages to each other through a link. But how this message transmission is done does not concern us. Figure 1 is an example and was chosen to describe the protocol, because of its simplicity. Each link is shown by a three parameters:

(link name, link status, link version number).

The link between sites A and B is named L_{AB}. When this link is operational, its status is *active* and when this link fails, its status is changed to *fail.* The link version number starts at 1 at the time of initialisation of the link and it is incremented if the link fails and incremented again when it is repaired and is operational. Therefore if the status of a link is changed its version number is also incremented. For example, an entry for L_{AB} at the time of initialisation of the link is

$\langle L_{AB}$,a,1 $\rangle$.

a stands for active and f stands for failure. If the link fails then its status changes to failure and its version number is incremented to 2. In this situation the entry is

$\langle L_{AB}$,f,2 $\rangle$.

Each site has an entry for the links that are connected to its site. These entries are in an array called the *Link Status Array.* After initialisation of the site A the link status array has the form:

$\langle L_{AB}$,a,1$\rangle$, $\langle L_{AC}$,a,1$\rangle$.

and for site B this array has the form:

$\langle L_{AB}$,a,1 $\rangle$, $\langle L_{BD}$,a,1$\rangle$.

Later on, during reconfiguration of the network, information from other links is augmented with these entries.

Each site also has information about links that are connected to each site. This information is kept in an array called the *Link Site Array*. Each entry in this array consists of three parameters:

⟨link name, one end of the link, another end of the link⟩.

The entry ⟨ L_{AB},A,B⟩ means that link L_{AB} connects sites A and B and the entry ⟨ L_{BD},B,D ⟩ means link L_{BD} connects sites B and D. For Figure 1 this array is shown in Figure 2.

⟨ L_{AB},A,B ⟩	⟨ L_{AC},A,C ⟩	⟨ L_{BD},B,D⟩	⟨ L_{CE},C,E ⟩	⟨ L_{DE},D,E⟩

Figure 2: Link Site Array for Network Connection

This array is maintained in each site and is used by a partition in deciding if other sites are connected to each other to form a partition or if they are isolated.

Each site with a link status array and a link site array can determine what other sites in other partitions are doing. In other words, are they active with respect to executing transactions or are they inactive?

Each site has three parameters associated with it: its name, its status and its version number. Therefore, each entry has the form:

⟨name of the site, status of the site, version number of the site⟩.

The status of the site is either *active*, which means that the site is executing transactions or *inactive* which indicates that the site is not executing user transactions. The version number of the site is initialised to 1 and incremented when the site is changing its partition and is involved in executing transactions. These entries are kept in an array called the *Site Status Array* and this array is kept in each site. For the Figure 1 this array at the time of initialisation is shown in Figure 3.

⟨ A,a,1 ⟩	⟨ B,a,1 ⟩	⟨ C,a,1⟩	⟨ D,a,1 ⟩	⟨ E,a,1 ⟩

Figure 3: Site Status Array for Network Connection

It shows that all sites are active and since the system has just started to operate, each version number is 1. Each site has two more data structures. The first is its present majority partition in stable storage, or its previous majority partition if the site has failed. The second is a *majority flag*. If the site is in a partition which is executing transactions then this flag is set to the value "true", otherwise it is "false".

2.2 The Proposed Protocol

When a new partition is created and before execution of user transactions starts, a coordinator for this partition must be selected. The selection is made by choosing the site with the lowest site-identifier number in the partition. As will be seen the performance of this protocol is unaffected by the choice of a site as coordinator, and the lowest site-identifier number has been chosen merely for convenience of implementation. Once the selection has been made, the coordinator initiates the protocol within the partition and then executes the protocol. It is then able to decide whether or not user transactions are executable in the partition, and informs other sites within the partition accordingly. This protocol is called the *Dynamic Voting Protocol* and is described below:

The coordinator of each new partition, asks other sites in its partition to send their link status array, site status array and their majority partition to the coordinator. On receiving the replies, the coordinator, denoted as AM_C, continues the protocol:

1. AM_C updates its link status array as follows:
   - If for each link status array, one site has sent information then the information is accepted as the entry to bring the link status array up to date.
   - If more than one entry for a link is received then the entry with the highest version number is accepted as the entry for the up to date link status array.
2. AM_C constructs a schematic of each partition by looking at the up to date link status array and comparing it with the link site array as follows:
   - If all the links to a site have failed, then that site is running in isolation.
   - If all the links to a group of sites have failed, then that group constitutes a group partition.
3. AM_C compares the new partition with the majority partition sent from different sites. If the new partition has the last site's version number and has a majority of sites compared with its previous majority partition, or it has a majority partition with respect to the configuration of other partitions, then the new

partition is a majority partition.

4. If a partition is a majority partition, then AM_C updates its site status array as follows:

   - Changes the status of each site to active, then increments the site version number for each site.

   - Sets the majority flag to true.

5. If a partition does not have a majority partition, then AM_C updates its site status array as follows:

   Changes the status of each site to inactive and leave the site version number of each site unchanged.

6. Sends tne updated link status array, updated site status array, majority partition and majority flag to each participant and starts accepting user transactions, if it is a majority partition.

7. Participants update their link status array, site status array, majority partition and majority flag. If the majority flag is true, then they start to accept user transactions.

If a break down in the system occurs while our dynamic voting protocol is executing, the process is aborted and automatically restarted from the beginning. Clearly, this selection process involves an additional overhead, but the experimental results (described below in Section 3) show that this overhead is fully justifiable.

3 Simulation Studies

In order to assess the practicality of the model, a detailed and extensive simulation has been carried out, using the simulation package Block Oriented Network Simulator (BONeS) [1]. The simulation is described in detail in [1]. It covers a range of parameters including the number of sites, probability of node failure, and node down time. For comparison purposes the simulation was carried out with the protocol described in this paper, together with two widely used conventional protocols. The complete discussion of the results of this simulation is given in [1]. They show that the protocol described in this paper gives increased availability when compared with the conventional protocols and Figure 4 gives a typical example of the results obtained for the simulation. In the figure the lower curve represents the availability for other conventional protocols used in this study (they are in fact virtually indistinguishable from each other) while the upper curve gives the corresponding results for the protocol described in this paper. In the figure, the y-axis represents the number of times the search for a majority of partition was successful; for a range of probabilities of node failure. The reference cited in [1] contains a large number of such results and they confirm the advantages of the protocol described in this paper.

[1]Block Oriented Network Simulator and BONeS are trademarks of Comdisco Systems, Inc.

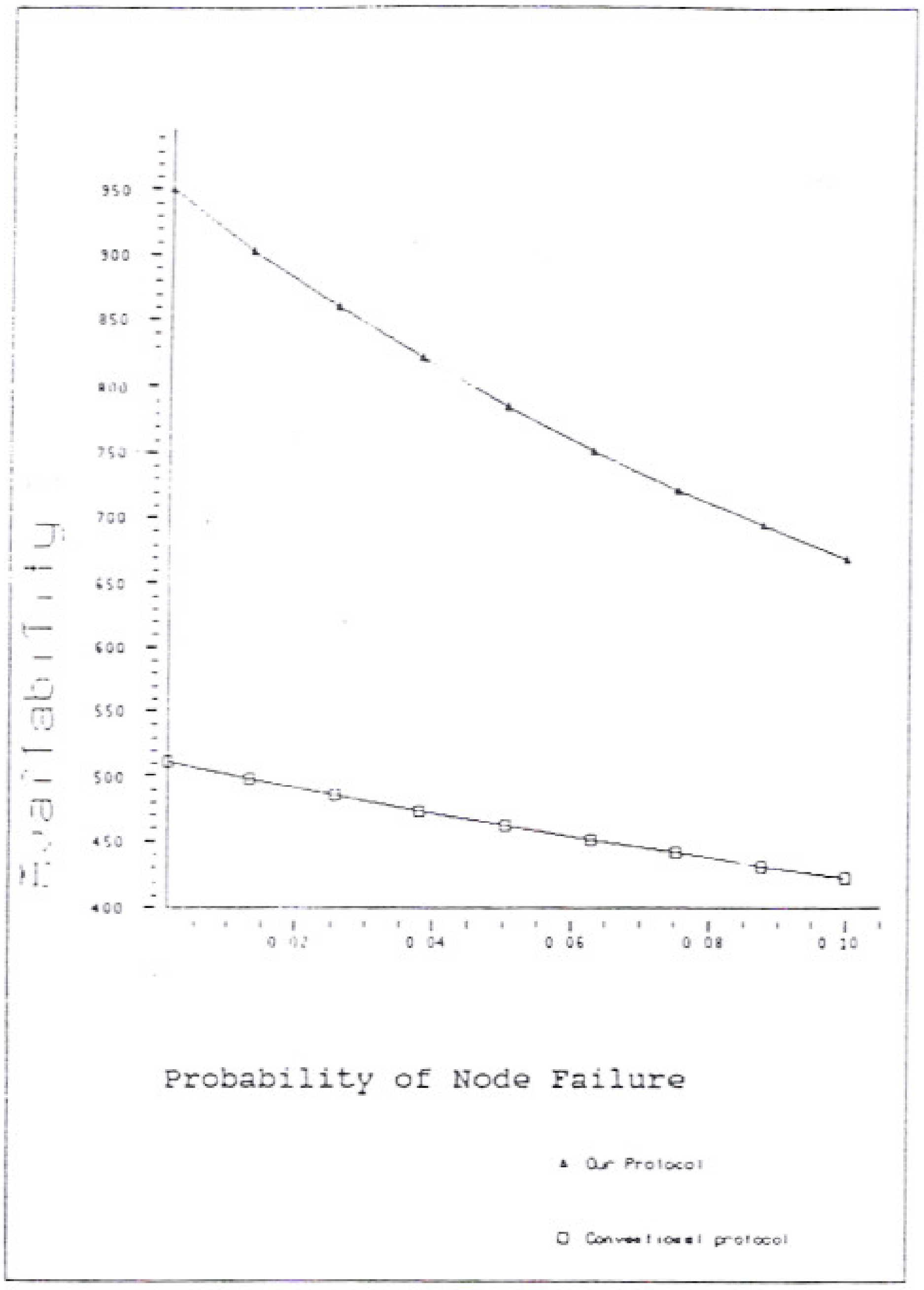

Figure 4: Availability vs. Probability of Node Failure

4 Conclusions

In this paper, the aim has been to develop a protocol for determining the majority partition after partitioning has occurred in a distributed database system. In contrast to other approaches, we use information regarding the configuration of the network in this selection process. The protocol to effect this selection has been described in detail, together with the necessary data structures. The protocol shown illustrate how the protocol works and demonstrate that the protocol increases data availability above that which is possible with previous protocols. The increased housekeeping cost incurred by keeping information about other sites is justifiable because the execution of user transactions can be continued despite failures of one or more sites, and without having a majority of sites in the partition. In order to test the conclusions of this study, a simulation has been carried out, and the results confirm our claim that our protocol provides better availability.

Acknowledgement

The authors wish to thank the referee for his helpful comments on the early version of this work.

References

[1] S. Daneshi. *'Availability and Consistency Issues in Replicated Distributed Database Systems'.* PhD thesis, University of Bristol, 1992.

[2] D. Davcev. 'A Dynamic Voting Scheme in Distributed Systems'. *IEEE Transactions on Software Engineering*, 15(1):93–97, 1989.

[3] D. Davcev and W. Burkhard. 'Consistency and Recovery Control for Replicated Files'. In *Proc. of 10th ACM Symposium on Operating Systems Principles*, pages 87–96, Oycas, Island, 1985.

[4] S. Davidson. 'Optimism and Consistency in Partitioned Distributed Database Systems'. *ACM Transactions on Database Systems*, 9(3):456–481, 1984.

[5] S. Davidson, H. Garcia-Molina, and D. Skeen. 'Consistency in Partitioned Networks'. *ACM Computing Surveys*, 17(3):341–370, 1985.

[6] S. Jajodia and D. Mutchler. 'Dynamic Algorithms for Maintaining the Consistency of a Replicated Database'. *ACM Transactions on Database Systems*, 15(2):230–280, 1990.

[7] S. Jajodia and D. Mutchler. 'Dynamic Voting'. In *ACM SIGMUD International Conference on Data Management*, pages 227–238, 1987.

[8] S. Jajodia and D. Mutchler. 'Integrating Static And Dynamic Voting Protocols to Enhance File Availability'. In *Proc. of 4th International Conference on Data Engineering*, pages 143–153, 1988.