SIAM J. COMPUT.
Vol. 15, No. 1, February 1986

010

OPTIMAL TERMINATION PROTOCOLS FOR NETWORK PARTITIONING*

FRANCIS CHIN† AND K. V. S. RAMARAO‡

Abstract. We address the problem of maintaining the distributed database consistency in presence of failures while maximizing the database availability. Network Partitioning is a failure which partitions the distributed system into a number of parts, no part being able to communicate with any other. Formalizations of various notions in this context are developed and two measures for the performances of protocols in presence of a network partitioning are introduced. A general optimality theory is developed for two classes of protocols—centralized and decentralized. Optimal protocols are produced in all cases.

Key words. commit protocols, consistency, database availability, distributed databases, fault-tolerance, network partitioning, optimal protocals, transaction processing

1. Introduction. A database DB consists of a collection of *entities* $D=\{d_1, d_2, \cdots, d_m\}$ such that each d_i in D has a *value set* V_i associated to it, and a set R of relations r on $\times_{i=1}^{m} V_i$ which we call *consistency constraints* for DB. An *instance* of the database DB is an element from $\times_{i=1}^{m} V_i$. An instance $(v_1, v_2, \cdots, v_m)$ of DB is *consistent* if and only if $(v_1, v_2, \cdots, v_m)$ is in r for all r in R.

User programs map the set of database instances into itself. We are primarily interested in those programs that map the set of consistent database instances into itself. Executions of such programs are known as *transactions* and play the central role in database literature [10], [12]. Formally, a transaction t is an execution of a (user) program, $t: \times_{i=1}^{m} V_i \rightarrow \times_{i=1}^{m} V_i$ such that for any consistent instance I, $t(I)$ is also consistent. If only a sequence of transactions is allowed to operate on a consistent database instance, then the final instance is also guaranteed to be consistent. We require that all transactions are ensured that the instances they are going to operate on are consistent. This guarantees the database consistency, without any need for an explicit validation of the consistency constraints. Several mechanisms are available which guarantee that any transaction sees an initial consistent instance and can map the set of consistent DB instances into itself even if several transactions are being concurrently run. See [2], [3] for a comprehensive survey on this area.

Without loss of generality, assume that, if $O_1, O_2, \cdots, O_k$ is the sequence of operations in a transaction, then no subsequence $O_1, O_2, \cdots, O_p$, $1 \leqq p < k$, guarantees the resulting database instance to be consistent. At the implementation level, this would imply that, given a consistent instance I_1 of DB, a transaction t acting on I_1 would lead to a consistent instance I_2 if and only if either t does not modify I_1 or all modifications made by t are incorporated onto I_1. Such an implementation of a transaction is known as *atomic* implementation [12]. Thus, a transaction can legally be completed in only one of two possible modes—either it is *committed* in which case all its effects are incorporated into the database instance, or it is *aborted* in which case none of its effects are incorporated.

A *distributed system* is an undirected connected graph $G=(V, E)$. Each node represents a *site* consisting of a processor and possibly storage and other modules. Each edge represents a *communication link*. A database is said to be *distributed* if it physically resides at more than one site. That is, each site contains a subset of the database entities. We make no assumptions on the degree of replication—each entity

* Received by the editors November 29, 1982, and in revised form August 25, 1984.

† Department of Computer Studies, University of Hong Kong, Pokfirlam Road, Hong Kong.

‡ Department of Computer Science, University of Pittsburgh, Pittsburgh, Pennsylvania 15260.

d_i resides at a number of sites n_i, $1 \leqq n_i \leqq |V|$. A transaction is said to be distributed if it is physically run at more than one site. A site at which a distributed transaction is executed is known as a "participating site" for that transaction. It can be seen that a distributed transaction can be atomically implemented if and only if either all participating sites commit it or all of them abort it. Communication among the participating sites is required to guarantee this condition and such *protocols* are known as *commit protocols* [10]. Thus, an execution of a commit protocol is associated with each transaction. Without loss of generality, assume that each transaction t has all sites in the distributed system participating in its execution.

A number of possible failures could occur in a distributed system. "Processor malfunctioning" is one of the most widely studied failures in the literature. See [14] for an informal survey of this topic. "Clean" site failures, where the processor at a site simply stops in case of a fault, are also extensively studied [1], [4], [8], [10], [11], [15]. A third kind of failure, the one we are interested in, is "network partitioning" where the graph G gets partitioned into a number of connected subgraphs. Existing literature on this problem is rather sparse [6], [7], [17].

In practice, commit protocols are not expected to handle failures in the system. Thus, the commit protocol is simply frozen when a fault is detected and a new type of protocol, known as "termination protocol" (TP) is invoked to handle the exception. (Formal definitions of these protocols will be given in § 2.) An execution of the TP directs the termination of an incomplete transaction. We require that the TPs also guarantee the database consistency. A commit protocol P is said to be *nonblocking* [15] to a failure type F if and only if there is a TP associated with P such that any incomplete transaction in presence of an arbitrary instance of F can be consistently completed at all operational sites by that TP. P is blocking otherwise. Obviously, one would prefer nonblocking protocols since they enhance the *availability* of the database.

Our interest in the network partitioning problem stems from the fact that there exists no commit protocol nonblocking to network partitioning [17]. Contributions made in this paper are as follows:

1. Formalization of termination protocols for network partitioning which extracts all the available information into the formalism.
2. Study of the properties of TPs, introducing the notion of "nontrivial" TPs.
3. Characterization of commit protocols allowing nontrivial TPs, recognition of a canonical commit protocol.
4. Introduction of measures for the performance of TPs.
5. Development of an optimality theory for these measures and the design of optimal termination protocols.

Organization of this paper is as follows: Section 2 develops the necessary theoretical background for the study of TPs. Notion of nontrivial TPs is introduced and a characterization theorem is proved. Section 3 presents the optimality results and optimal protocols for a class of protocols known as decentralized protocols. Section 4 deals with the centralized protocols. The effectiveness of centralized and decentralized protocols is compared and it is proved that certain centralized protocols are superior. Section 5 discusses certain consequences of our results and concludes the paper.

2. Formal background. The following model of a distributed transaction is used throughout the paper: a transaction t is initiated at some site in the system and is decomposed into an appropriate number of *subtransactions* such that each site participating in t would run exactly one subtransaction. No assumption is made on how this is accomplished—a simple scheme is for the site where t is originated itself to compute this. Each participating site receives its subtransaction, runs it concurrently with others and makes a (local) decision of "commit" or "abort." All participating sites then cooperatively make a global "commit" or "abort" decision following a *commit rule.* Finally, they commit or abort the transaction according to the global decision. The only requirement on the commit rule is that the conditions for "commit" and "abort" global decisions partition the set of all possible combinations of local decisions. For instance, a simple commit rule is as follows: if there is a site whose local decision is "abort" then the global decision is "abort"; otherwise it is "commit." The "commit" and "abort" *actions,* once implemented, are *irreversible.* Here, observe that we are making a definite distinction between making a decision and implementing it. In theory, a decision is reversible; but, once it is implemented, it cannot be reversed.

In this model of distributed transaction, commit protocol comes into play in implementing the commit rule. Specifically, its task is to ensure that, a) a global decision according to a commit rule is made on the mode of completion, and b) the same decision is implemented by all sites. We assume that the processors do not exhibit any malicious behavior.

One simple commit protocol is to pool all local decisions at a specific site, let that site apply the commit rule and compute the global decision which can be sent to all other participating sites. Such protocols where a single site coordinates the task of completing a transaction are known as "centralized" protocols. At the other extreme, each site can independently contact all other sites. These are known as "decentralized" protocols. Though several intermediate schemes are possible, results obtained in these two cases can be easily extended to all other cases. In this paper, we study only the centralized and decentralized protocols.

We model the execution of a commit protocol by a collection of finite state automata (FSA), one associated with each participating site [17]. An FSA in a state s reads a set of *messages* from other sites, performs local computation (if any), sends a (possibly empty) set of messages to other sites, and changes its state. Each FSA satisfies the following conditions:

1. It is indeterministic.
2. Each transition is associated with a unique input.
3. The final states are partitioned into two classes—"abort states" Ab and "commit states" Com.
4. There are no transitions from a final state.
5. The state graph is acyclic.

Notation. For any FSA, we denote all nonfinal states that lead only to commit states by Pc, all nonfinal states that lead only to abort states by Pa, and all states that lead to both commit and abort states by N. When necessary, we use the identity of the site as a suffix to distinguish FSAs at different sites. For simplicity, we equate the FSA at a site to the site itself. Thus, we use "site i" to mean "the FSA at site i" when no confusion can arise.

DEFINITION 1. A protocol P represented by a collection of FSAs with the above properties is a *commit protocol* if and only if for every input (from the input alphabet of messages on local and global decisions), the FSAs all reach final states from the same class (either Ab or Com) in the absence of any failures.

2.1. Structure of the FSAs. State graphs of FSAs in a commit protocol are acyclic digraphs by definition. Now, we assume that they are in fact trees. (It is easy to see that any acyclic FSA with the above properties can be converted into a tree FSA, by introducing some dummy states if necessary.) Practical considerations introduce more structure into the FSAs. One such aspect is the "degree of synchronization" among the FSAs [17]:

DEFINITION 2. Let P be a protocol and $F_1, F_2, \cdots, F_n$ the FSA in P. A *global state* of P, $(s_1, s_2, \cdots, s_n)$, is an element of $S(F_1) \times S(F_2) \times \cdots \times S(F_n)$ where $S(F_i)$ is the set of states in F_i, $1 \leqq i \leqq n$, with the property that there is an input for P on which F_i are concurrently in states s_i, $1 \leqq i \leqq n$.

For any state s in an FSA, let $d(s)$ represent the number of state transitions required to transform the initial state into s. A protocol P is said to be "synchronized within k states" for $k \geqq 1$, if and only if $|d(s) - d(t)| \leqq k$ for all pairs of states s, t in two different FSAs such that there is a global state of P in which both s, t occur. During the execution of a commit protocol, each site sends certain messages out and waits for similar messages or responses to its own messages from certain other sites. Thus, the sites "progress" through the execution of the protocol at approximately the same pace. In fact, no site leads any other site by more than one state transition. Thus, all commit protocols being used are synchronized within one state. For instance, if a site has reached a final state, any other site is either in a final state or in a state adjacent to a final state. Commit protocols synchronized within one state can be justified by the fact that they are inexpensive while no generality is sacrificed in using them.

LEMMA 1. *Let P be a commit protocol synchronized within one state. Then, all its* FSA *are isomorphic.*

Proof. Let F_1, F_2 be two FSAs of P. For any path from the initial state to a final state of F_1, there corresponds a path in F_2 from its initial state to a final state. The lengths of these paths can differ at most by one, due to the synchronization. When they differ, a dummy state can be introduced into the appropriate FSA without changing its behavior. Since there is a unique input associated with each transition, there correspond two different paths in F_2 for any two different paths in F_1. Hence, being trees, the state graphs of F_1, F_2 are isomorphic. Q.E.D.

All FSAs in a protocol can be identical in which case we call it a *uniform protocol.* It is nonuniform otherwise. Observe that the decentralized commit protocols are uniform while the centralized are not.

2.2. Termination protocols. Given that a site i is in state s, consider the set of all possible states any other site j could be in at the same (global) instance. In a uniform commit protocol where the FSAs are synchronized within one state, such possible states are simply the states adjacent to s (assuming the same names for states in different FSAs).

DEFINITION 3. Let P be a commit protocol. The *concurrency set* of a state s at site i, denoted by $R(i, s)$, is the set of all ordered pairs (j, t) such that there is a global state $(s_1, s_2, \cdots, s_n)$ of P satisfying $s = s_i$ and $t = s_j$.

The following property of commit protocols is immediate:

PROPERTY 1. If $s \in \mathrm{Ab} \cup \mathrm{Pa}$, then there does not exist an ordered pair (j, t) in $R(i, s)$ for any $t \in \mathrm{Com} \cup \mathrm{Pc}$ and for any sites i, j, and conversely, if $s \in \mathrm{Com} \cup \mathrm{Pc}$, then there does not exist (j, t) in $R(i, s)$ for any $t \in \mathrm{Ab} \cup \mathrm{Pa}$.

Let Q be the set of states in a uniform commit protocol P. Let J denote the power set of $V \times Q$. Let D be the maximal subset of J such that $S \in D$ if and only if

i) $(i, s) \in S$ and $(i, t) \in S$ implies $s = t$,

ii) (i, s), $(j, t) \in S$ implies $(j, t) \in R(i, s)$ and $(i, s) \in R(j, t)$.

It is not hard to extend the definition of D to nonuniform protocols. The importance of D is based on the following concern: Recall that a network partitioning partitions the graph G into a number of connected subgraphs such that there is no edge joining any two of these subgraphs. This could happen due to the deletion (failure) of certain nodes (sites) and/or edges (communication links). When such a partitioning is detected, the commit protocol is frozen and each site remains in a well defined state. The entity D defined above provides us with the formal representation of the status of each subgraph immediately after a partitioning is detected. Any set S in D corresponds to a physically realizable subgraph, embedding the status of a transaction being run with the commit protocol P. Notice that all the information available in the physical partitioning has been extracted into this formalism: the first condition represents the fact that no site can be in more than one state at the same time, while the second ensures that the concurrency relations of states under P are preserved. We call such S in D a *component* so that D represents the set of all physically realizable components of the given network under a given commit protocol. For S in D, let site $(S)=\{i|(i, s)\in S$ for some $s\in Q\}$ and state $(S)=\{s|(i, s)\in S$ for some $i\in V\}$. Extending the above definition of concurrency to different components, we say two components S, T are *concurrent* if and only if a) site $(S)\cap$ site $(T)=\emptyset$, and b) for $(i, s)\in S$ and $(j, t)\in T$, $(i, s)\in R(j, t)$ and $(j, t)\in R(i, s)$. The second condition can be restated simply as $S\cup T\in D$. In the following, we first describe the centralized version of the general termination protocol before formally defining it.

When a network partitioning is detected, sites in each component S "elect" a single site as a "coordinator" to execute the termination protocol (TP). (See [9] for some election protocols.) Coordinator collects the identities of the sites and their states, and applies the TP based on this information. Let x, y, z represent the actions to be taken by a TP. The coordinator delays the transaction until a network reconfiguration if the action is z. It directs the other sites to commit (abort) if the action is x (y). The primary requirement for a TP is that it consistently terminates a transaction in presence of faults. The situation is quite involved in case of network partitioning since there can exist a number of components in the system, each trying to complete the incomplete transactions, independent of all other components. In spite of these independent activities, each transaction's atomic implementation and consistency should be guaranteed. The following definition formulates the notion of a TP:

DEFINITION 4. Let P be a commit protocol. Let $f: D\rightarrow\{x, y, z\}$ be any function. We say f has *preservation property* if and only if, for any S in D,

i) state $(S)\cap \mathrm{Com}\neq\emptyset$ implies $f(S)=x$,

ii) state $(S)\cap \mathrm{Ab}\neq\emptyset$ implies $f(S)=y$.

f has *commit property* if and only if, for any two concurrent components S, T in D, $\{f(S), f(T)\}\neq\{x, y\}$.

f is a *termination protocol* (TP) if and only if f has both preservation and commit properties.

Observe that the commit and preservation properties are mutually consistent due to Property 1. Notice that we have considered three possible actions x, y, z for a TP, in contrast to only two actions for a commit protocol. This is due to the following previously known result:

THEOREM 1 [17]. *Let P be a commit protocol. Assume that arbitrary network partitionings are possible. Then, for any* TP *f of P, there exists a component S in D such that $f(S)\neq x$ and $f(S)\neq y$.*

Thus, there are occasions on which a component should wait until a reconfiguration. Given the above limitation, we want to optimize the performance of TPs.

The above definition of TPs does not specify how a TP is implemented and how reconfigurations are handled. We leave these details unspecified. We do not require any assumptions in this respect for the purposes of this paper. The interested reader is referred to [13].

The following property of TPs is immediate from the definition.

PROPERTY 2. Let P be a commit protocol synchronized within k states, for $k \geqq 1$. For any FSA, let $Pa_k = \{s \in \mathrm{Pa} \mid |d(a) - d(s)| \leqq k$ for some $a \in \mathrm{Ab}\}$, $\mathrm{Pc}_k = \{s \in \mathrm{Pc} \mid |d(c) - d(s)| \leqq k$ for some $c \in \mathrm{Com}\}$. Let f be any TP of P. Then, for any S in D,

i) state $(S) \subseteq (N \cup \mathrm{Pa}_k)$ implies $f(S) \neq x$,
ii) state $(S) \subseteq \mathrm{Pc}_k$ implies $f(S) \neq y$.

2.3. Measuring the performance of TPs. For a given commit protocol, we wish to find the TP which maximizes the number of incomplete transactions that can be completed, when used in conjunction with all possible network partitionings. This is because an incomplete transaction makes its resources unavailable to the other transactions. Since a transaction may be completed in one component and kept incomplete in another, the number of components where transactions are completed needs to be maximized. On the other hand, it is not wise to complete a transaction in a "small" component (containing a few sites) while leaving an incomplete transaction in a large component. Hence, the number of sites should also be considered in assessing the performance of a TP. Given a TP f, let CM $(f) = |\{S \in D \mid f(S) = z\}|$ and SM $(f) = \sum \{|S|$ where $S \in D$ and $f(S) = z\}$.

DEFINITION 5. Let P be a commit protocol. A TP f of P is *component optimal* in a class of TPs F if and only if CM $(f) \leqq$ CM (f') for all f' in F.

DEFINITION 6. A TP f of P is *site optimal* in F if and only if SM $(f) \leqq$ SM (f') for all f' in F.

2.4. Nontrivial termination protocols. As the first step towards an optimality theory, we ask: how many TPs does a commit protocol have? Clearly, any commit protocol has at least one TP defined as follows: for $S \in D$, states $(S) \cap \mathrm{Com} \neq \emptyset$ implies $f(S) = x$, state $(S) \cap \mathrm{Ab} \neq \emptyset$ implies $f(S) = y$, and $f(S) = z$ otherwise. This TP is trivial since it satisfies the commit property by doing nothing. We intend to find more effective, "nontrivial" TPs.

For simplicity, we restrict our attention to commit protocols synchronized within one state throughout the remainder of this paper. The results obtained can be extended without much difficulty.

DEFINITION 7. Let P be a commit protocol (synchronized within one state) and f a TP of P. For any FSA of P, let $N_P = \{s \in N \mid |d(t) - d(s)| = 1$ for some $t \in \mathrm{Pc} \cup \mathrm{Com}\}$ and $P_N = \{s \in \mathrm{Pc} \mid |d(t) - d(s)| = 1$ for some $t \in N\}$. We say f is a *nontrivial* TP (NTP) if one of the following is true: i) There exists S in D such that coordinator $\notin$ site (S) (for centralized P), state $(S) \subseteq N_P$ and $f(S) \neq z$. ii) Let $P_N \neq \emptyset$ for some FSA. There exists S in D such that state $(S) \subseteq P_N$ and $f(S) \neq z$.

This definition extracts the cases where an intelligent behavior is demanded from the TP. Existence of NTPs depends directly on the commit protocol. For instance, consider the following decentralized version of the widely-used two-phase commit protocol [10], [11]:

Step 1. Each site sends its local decision to all other participating sites and receives the local decisions of all other participating sites.

Step 2. Using the commit rule, each site computes the global decision. The transaction is completed accordingly.

Figure 1 is the state diagram for these FSAs. Messages received are shown above the line and messages sent out below the line.

It is not hard to check that this protocol has no NTP. In fact, it can be shown that its only TP is the trivial one. Following is a characterization of the commit protocols having NTPs.

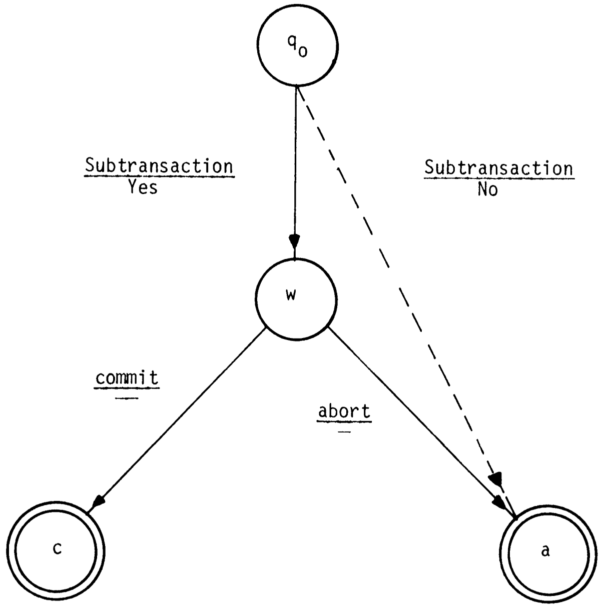

FIG. 1. *Two-phase commit protocol.*

THEOREM 2. *Let P be a commit protocol. Then, P has an* NTP *if and only if* $\text{Pc} \neq \emptyset$ *for an* FSA *of P.*

Proof. Sufficiency—Assume that $\text{Pc} \neq \emptyset$ for some FSA of P. Define $f: D \to \{x, y, z\}$ as follows:

for $S \in D$, state $(S) \cap (\text{Com} \cup \text{Pc}) \neq \emptyset \Leftrightarrow f(S) = x$,
state $(S) \cap (\text{Ab} \cup \text{Pa}) \neq \emptyset \Leftrightarrow f(S) = y$, and
$f(S) = z$ otherwise.

Let $S, T \in D$ such that $S \cap T = \emptyset$, $f(S) = x$ and $f(T) = y$. Then, state $(S) \cap (\text{Com} \cup \text{Pc}) \neq \emptyset$ and state $(T) \cap (\text{Ab} \cup \text{Pa}) \neq \emptyset$. But, this implies by Property 1 that $S \cup T \notin D$, proving that f is a TP. Since there is an FSA in which Pc is nonempty by hypothesis, P_N is nonempty for that FSA. Hence, f is an NTP.

Necessity—Assume that $\text{Pc} = \emptyset$ for all FSAs of P, and that there is an NTP f of P. Thus, there exists $S \in D$ such that coordinator $\notin$ site (S), state $(S) \subseteq N_P$ and $f(S) \neq z$. As in Property 2, it can be shown that $f(S) \neq x$. Thus, $f(S) = y$. Consider $S' \in D$ such that site $(S') \subset V$-site (S) and state $(S') \subseteq \text{Com}$. Then, $S \cup S' \in D$ by the hypothesis. Since f is a TP, $f(S') = x$ due to the preservation property. But, this implies that $(f(S), f(S')) = (y, x)$, a contradiction to the commit property of f. Q.E.D.

Consequently, any version of the two-phase commit protocol cannot have an NTP. In view of the above characterization, the simplest commit protocol with NTPs has at least one FSA with the state graph shown in Fig. 2.

Any other commit protocol with NTPs can be considered as an extension of this *canonical* commit protocol. This commit protocol is known in the literature as the "three-phase commit" [15]. For simplicity, we consider only the commit protocol with the above canonical state graph for a detailed study. This can be justified as follows: It is clear from the definition of NTPs that the only states that need special attention are the N_P and P_N states, i.e., "wait" and "commitable" states. Hence, even if a more general protocol is considered, all other states play no critical role in studying the NTPs. Thus, the TPs of the canonical commit protocol can be extended to TPs of more general protocols in a simple fashion.

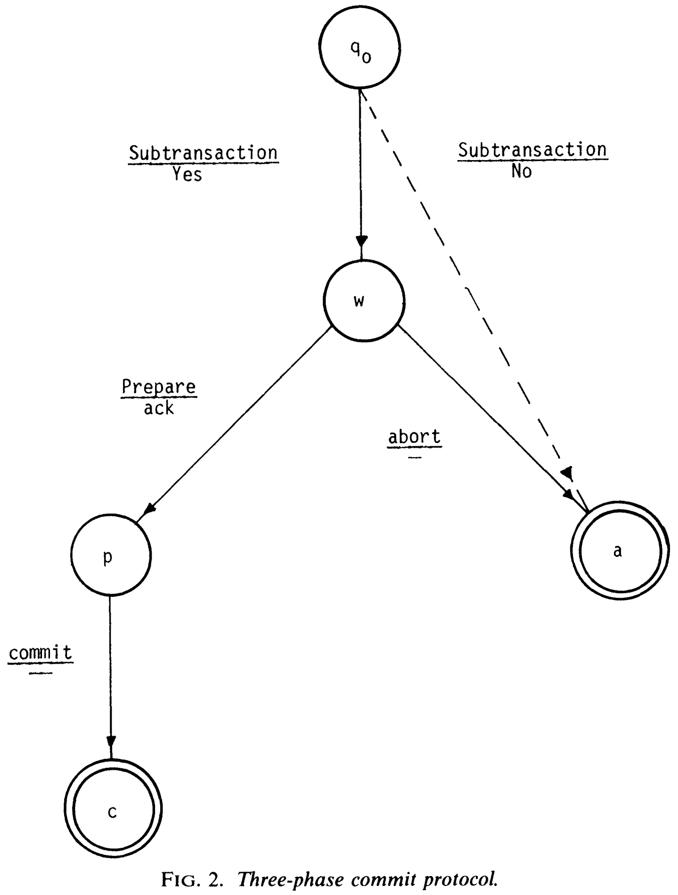

FIG. 2. *Three-phase commit protocol.*

We consider the decentralized and centralized versions of the above canonical commit protocol P: initial state is q, "wait" state is w, "commitable" state is p, "commit" state is c, and "abort" state is a.

3. Decentralized commit protocol.

DEFINITION 8. A version P of the canonical commit protocol is *decentralized* if and only if all FSAs are identical and each site communicates with all other sites.

TPs of the decentralized protocol are referred to as decentralized TPs (DTPs).

LEMMA 2. *Let f be a* DTP *of P. Let $S, T \in D$ such that* state $(S \cup T) \cap (\text{Com} \cup \text{Ab}) = \varnothing$. *Then, $f(S) = x$ and $f(T) = y$ implies that* site $(S) \cap$ site $(T) \neq \varnothing$.

Proof. Assume that the claim is false and that there exist S, T in D with the above properties. Then, state $(S) \not\subseteq N$ and state $(T) \not\subseteq \text{Pc}$ by Property 2. This, together with the assumption that site $(S) \cap$ site $(T) = \varnothing$ implies that $S \cup T \in D$, thus violating the commit property of f. Q.E.D.

From the definition of a TP, the above conditions are also sufficient. Hence, we have,

THEOREM 3. *Necessary and sufficient conditions that $f: D \rightarrow \{x, y, z\}$ is a* DTP *are,*

i) *f satisfies the preservation property,*

ii) state $(S \cup T) \cap (\text{Com} \cup \text{Ab}) = \varnothing$, *$f(S) = x$ and $f(T) = y$ implies that* site $(S) \cap$ site $(T) \neq \varnothing$.

Following are two simple but powerful properties of DTPs which play a critical role in obtaining the optimality results:

LEMMA 3. *Let $S \in D$ such that* state $(S) \subseteq N$. *Construct T such that* state $(T) \subseteq \mathrm{Pc}$ *and* site $(T) \subseteq V$-site (S). *Then, for any* DTP f *(of the canonical commit protocol P), either $f(S) = z$ or $f(T) = z$ or both.*

Proof. Follows immediately from Theorem 3.

LEMMA 4. *Let $S, T \in D$ such that* site $(S) =$ site (T), state $(S) \subseteq N$, state $(T) \subseteq \mathrm{Pc}$, $f(S) = y$ *and* $f(T) = x$. *Then, $f(R) = z$ for any R in D such that* site $(R) \subseteq V$-site (S).

Proof. $S \cup R \in D$ and $T \cup R \in D$ for any such R. Owing to the commit property of f, $f(R)$ can neither be x nor y. Q.E.D.

For an intuitive understanding of the above result and the results to follow, let us consider a simple tabular representation for the components. Consider for example the case of three sites 1, 2, 3. Each site represents a column and each row is a vector of states. It is sufficient to consider only the states w and p.

Each row in Table 1 can be interpreted as a component. For instance, row 1 represents the component $\{(3, w)\}$ while row 5 represents the component $\{(1, w), (3, w)\}$. If a TP f maps row (component) 1 to y, then it is easy to see that rows (components) 12, 8, 9 should be mapped to z. Lemma 3 above formalizes this fact. Similarly, if row 1 is mapped to y and row 7 is mapped to x by f, then the rows 2, 3, 6, 8, 9, 12, 17, 18 should all be mapped to z. This is formalized by Lemma 4.

TABLE 1

sites→		1	2	3		1	2	3
states↓	1.			*w*	10.		*p*	*p*
	2.		*w*		11.	*p*		*p*
	3.	*w*			12.	*p*	*p*	
	4.		*w*	*w*	13.		*w*	*p*
	5.	*w*		*w*	14.		*p*	*w*
	6.	*w*	*w*		15.	*w*		*p*
	7.			*p*	16.	*p*		*w*
	8.		*p*		17.	*w*	*p*	
	9.	*p*			18.	*p*	*w*	

We now give a lower bound on the component measure of the TPs, which can be easily appreciated from the above tabular representation.

THEOREM 4. *Let f be a* DTP *of P. Then,* CM $(f) \geqq 2^n - 2$ *where $n = |V|$.*

Proof. Let $S \in D$ such that state $(S) \subseteq N$. Let $T \in D$ such that site $(T) = V$-site (S) and state $(T) \subseteq \mathrm{Pc}$. Either $f(S) = z$ or $f(T) = z$ by Lemma 3. Clearly, for any S in D such that state $(S) \subseteq N$ and $|S| \leqq n - 1$, there is a T such that either $f(S) = z$ or $f(T) = z$. But, the total number of such S in D is $2^n - 2$. Thus, CM $(f) = |\{S \in D | f(S) = z\}| \geqq 2^n - 2$. Q.E.D.

Observe that Theorem 1 follows simply as a corollary of this lower bound result.

DEFINITION 9. A *quorum protocol f* of P, characterized by an ordered pair (d, e) of positive integers satisfying $d + e > n$, is a function $f: D \to \{x, y, z\}$ such that, a) f satisfies the preservation property, b) state $(S) \cap \mathrm{Pc} \neq \varnothing$ and $|S| \geqq d$ implies $f(S) = x$, c) (state $(S) \cap \mathrm{Pc} = \varnothing$ OR $|S| < \mathrm{d}$) AND (state $(S) \cap N \neq \varnothing$ and $|S| \geqq e$) implies $f(S) = y$, and d) $f(S) = z$ otherwise.

Each ordered pair of integers (d, e) represents a different quorum protocol. It can be verified using Theorem 3 that any quorum protocol is a DTP. Hereafter, we refer to quorum protocols as QTPs. See [16] for a discussion of quorum protocols used as commit protocols. The following result shows that QTPs exist in pairs in a strong sense:

LEMMA 5. *Let f be a* QTP *characterized by* (d, e). *Then, there exists another* QTP f' *such that* CM $(f) =$ CM (f') *and* SM $(f) =$ SM (f').

Proof. Consider the QTP f' characterized by (e, d). Observing that for any m, $1 \leqq m \leqq n$, there exist S, T in D such that $|S| = |T| = m$, state $(S) \subseteq N$ and state $(T) \subseteq \mathrm{Pc}$, it is easy to check that the claim is true. Q.E.D.

THEOREM 5. *There is a* QTP *g which is component optimal among all* DTPs *of P.*

Proof. Consider the QTP g characterized by $(1, n)$. Then, for any S in D, $g(S) = z$ if and only if state $(S) \subseteq N$. Thus, there are exactly $2^n - 2$ such components in total, corresponding to all proper subsets of V. (Recall that $|N| = 1$.) Q.E.D.

The "dual" QTP given by $(n, 1)$ is also optimal by Lemma 5. In fact, it is not hard to see that the QTPs given by $(2, n-1)$ and $(n\text{-}1, 2)$ are also component optimal. However, for $n > 2$, no other QTPs are component optimal. This can be checked from the following general formula for CM (f) when f is the QTP given by (d, e): assuming that $d \geqq e$,

$$\mathrm{CM}\,(f) = \sum_{r=1}^{e-1} 2^r \binom{n}{r} + \sum_{r=e}^{d-1} \binom{n}{r}.$$

On the other hand, not all component optimal DTPs are QTPs. For instance, consider a specific site i, map the component $\{(i, w)\}$ to z and T with site $(T) = V - \{i\}$ and state $(T) = \{p\}$ to x while all other components are mapped as for the QTP given by $(n, 1)$. This new DTP is component optimal but not a QTP.

3.1. Site optimal protocols. First, we notice that the component optimal DTPs obtained above may not be site optimal. To see this, let us consider the QTPs first. For the QTP f given by (d, e), SM (f) can be shown to be

$$\sum_{r=1}^{e-1} r \cdot 2^r \cdot \binom{n}{r} + \sum_{r=e}^{d-1} r \cdot \binom{n}{r}$$

assuming that $d \geqq e$. Let f' be the QTP given by $(n-1, 2)$ and f'' by $(n-2, 3)$. Then, it can be verified from the above formula that SM $(f'') <$ SM (f') for $n > 8$. In general, let k_0 be the smallest positive integer k such that $k \geqq (n-k)(2^{n-k} - 1)$. Denote by u the QTP characterized by $(k_0, n - k_0 + 1)$. Now, we prove that u is site optimal among the DTPs of P.

THEOREM 6. *u is site optimal among all* QTP*s of P.*

Proof. For any m, $\lceil n/2 \rceil \leqq m < n$, let f_m be the QTP given by $(m, n - m + 1)$. Then,

$$\begin{aligned} \mathrm{SM}\,(f_m) - \mathrm{SM}\,(f_{m+1}) &= (n-m)2^{n-m} \binom{n}{m} - n \binom{n}{m} \\ &= [(n-m)2^{n-m} - n] \binom{n}{m}. \end{aligned}$$

For $m < k_0$, $m < (n-m)2^{n-m} - n + m$, so that $(n-m)2^{n-m} - n > 0$. Thus, SM $(f_m) >$ SM (f_{m+1}). On the other hand, SM $(f_m) \leqq$ SM (f_{m+1}) for $m \geqq k_0$. Hence, SM (f_m) is minimum for $m = k_0$. Q.E.D.

THEOREM 7. *u is site optimal among all* DTP*s of P.*

Proof. We want to show that every DTP f has a QTP of better performance under the site measure. Application of the above theorem then proves our claim.

Case 1. Consider DTP f such that there are no S, T satisfying site $(S) =$ site (T), state $(S) = N$, state $(T) = \mathrm{Pc}$, $f(S) = y$ and $f(T) = x$. Thus, for any $M \subseteq V$, at least one of U or W where site $(U) =$ site $(W) = M$, state $(U) = \mathrm{Pc}$, state $(W) = N$, is mapped to z. Since there are subsets of V with size r, $1 \leqq r < n$, SM $(f) \geqq \sum_{r=1}^{n-1} r \cdot \binom{n}{r}$. Now, consider the QTP s characterized by $(n-1, 2)$. SM $(f) -$ SM $(s) = n^2 - 2n > 0$ for $n > 2$.

Case 2. Consider DTP f such that there exist S, T satisfying $M = \text{site}(S) = \text{site}(T)$, state $(S) = N$, state $(T) = \text{Pc}$, $f(S) = y$ and $f(T) = x$. Assume that $f(S') \neq z$ for all S' such that site $(S') =$ site (S) since it would not increase SM (f). Let S be the smallest component satisfying the above conditions. If $S \leqq \lfloor n/2 \rfloor$, then it is not hard to check that SM $(f) >$ SM (g). (Recall that g is the QTP given by $(n, 1)$.) Assume that $S > \lfloor n/2 \rfloor$. Let

$$L = \{M \subseteq V \mid \text{there exist } S, S' \text{ in } D, \text{ such that site}(S) = \text{site}(S') = M, \\ \text{state}(S) = N, \text{ state}(S') = \text{Pc}, f(S) = y \text{ and } f(S') = x\}.$$

If $|M'| > k_0$ for some $M' \subseteq V$ *not in* L, observe that SM (f) is not increased when M' is inserted into L (due to the definition of k_0 and Lemma 4). Thus, in general, all M' not in L such that $M' > k_0$ can be placed into L without increasing SM (f).

If $|M'| < k_0$ for some M' *in* L, then pick the smallest such M' and delete it from L. Again, by the definition of k_0, this cannot increase SM (f). Repeating this process, we can convert f into a QTP without increasing SM (f). Q.E.D.

The following results are immediate from the above theorem and Lemma 5.

COROLLARY 1. *There are at most two* QTP*s which are site optimal among the* DTP*s and these are the only site optimal* DTP*s.*

COROLLARY 2. *For* $n \geqq 9$, *no component optimal* DTP *is site optimal and vice versa.*

These results are slightly disturbing since they show that the site and component optimalities are complementary. Ideally, one would like to have a TP that simultaneously optimizes the number of sites and components. But, we have just proved that it is not possible when the commit protocol used is decentralized. Furthermore, QTPs are not very desirable due to the fact that, for any QTP f, there exist partitionings in which *all components* (and hence all sites) wait under f. Thus, they cannot guarantee that at least one component in any partitioning can be allowed to complete the incomplete transaction.

We now consider the centralized commit protocol where the situation is rather pleasant: we produce a TP which is both component and site optimal. Furthermore, this TP guarantees the transaction completion in at least one component, as long as the coordinator site is operational.

4. Centralized commit protocol. We first present a slightly generalized definition of centralized protocols. Define a partial ordering on the states of an FSA as: $t \leqq s$ if and only if the distance of s to its nearest final state is no less than the distance of t to its nearest final state. A set of FSAs LS is said to be a *leading set* if and only if for any S in D, (i, s), $(j, t) \in S$ and $i \in \text{LS}$ implies $s \leqq t$.

DEFINITION 10. A version P of the canonical commit protocol is *centralized* if and only if it has a leading set of sites.

TPs of the centralized commit protocol are called centralized TPs (CTPs). Observe first that the class of DTPs considered in the previous section is a subclass of the CTPs when their domain is restricted to the realizable components for the centralized protocol. This is because the commit and preservation properties of a TP hold even if the TP is restricted over a proper subset of its domain. Secondly, we observe that there is a natural class of CTPs which is very interesting: the CTPs that explicitly exploit the existence of leading set. Notice that, when a leading site (a site whose FSA is in the leading set) is in state w ("wait"), all sites in the system are in states from $\{w, q\}$. We call the CTPs which use this fact *leading set* TPs (LTPs). Formally, DEFINITION 11. A CTP is an LTP if and only if, for all S in D, site $(S) \cap \mathrm{LS} \neq \emptyset$ implies $f(S) \neq z$ where LS is the leading set.

COROLLARY 3. *In absence of the simultaneous failure of all leading sites, for any transaction, there is at least one component in any partitioning that can successfully terminate that transaction when an* LTP *is used.*

In principle, it is possible to abort a transaction in a component as long as it is guaranteed that no other component commits it. But, due to practical considerations, this approach is not satisfactory. Typically, an aborted transaction is repeatedly tried until it is committed. Thus, a TP should try to commit a transaction if it is possible to do so. This is equivalent to saying that for any LTP f and S in D, $f(S) = x$ whenever state $(S) \cap (\mathrm{Com} \cup \mathrm{Pc}) \neq \emptyset$.

LEMMA 6. *Let f be an* LTP. *Then, $f(S) = z$ whenever* site $(S) \cap \mathrm{LS} = \emptyset$ *and* state $(S) \subseteq N$.

Proof. Notice that, if there are S', S'' such that $S \cup S' \in D$, $S \cup S'' \in D$, (site $(S') \cup$ site $(S'')) \cap$ site $(S) = \emptyset$, $f(S') = x$, and $f(S'') = y$, then $f(S) = z$. It is easy to see that one can produce such S', S'' when f is an LTP and site $(S) \cap \mathrm{LS} = \emptyset$. For instance, take site $(S') =$ site $(S'') = \mathrm{LS}$, state $(S'') = N$ and state $(S') = \mathrm{Pc}$. Q.E.D.

Based on this lemma, we construct a simple but effective CTP. But, first we generalize the notion of quorum protocols introduced in the previous section.

DEFINITION 12. A quorum TP is *weighted* if, in the definition of the QTP, each site is assigned a weight and the size of a set is replaced by its weight, i.e. sum of the weights of its elements. The condition $d + e > n$ is replaced by $d + e >$ sum of the weights of all nonleading sites.

Now, consider a weighted QTP defined as follows: assign a weight of 1 to all sites not in the leading set and assign a weight of n to each leading site. Set $d = 1$ and $e = n$. This CTP can be explicitly given as follows: Call it h.

1. state $(S) \cap (\mathrm{Com} \cup \mathrm{Pc}) \neq \emptyset$ implies $h(S) = x$,
2. state $(S) \cap (\mathrm{Com} \cup \mathrm{Pc}) = \emptyset$ and state $(S) \cap (\mathrm{Ab} \cup \{q\}) \neq \emptyset$ implies $h(S) = y$,
3. site $(S) \cap \mathrm{LS} \neq \emptyset$ and state $(S) \subseteq N$ implies $h(S) = y$,
4. $h(S) = z$ otherwise.

It can be easily seen that h is indeed an LTP. Following result shows that h is the "best" among the LTPs in a very strong sense:

LEMMA 7. *Let f be an* LTP. *Then, $f(S) = z$ whenever $h(S) = z$.*

Proof. $h(S) = z$ only if site $(S) \cap \mathrm{LS} = \emptyset$ and state $(S) \subseteq N$. $f(S) = z$ in this case, by Lemma 6. Q.E.D.

COROLLARY 4. *h is both component and site optimal among the* LTPs.

We now consider the decentralized TPs of the centralized commit protocol, which pay no attention to the existence of the leading set. Since these are only restrictions of the DTPs considered in § 3, we call them *restricted* DTPs (RDTPs).

LEMMA 8. *Let f be an* RDTP. *Then,* CM $(f) > 2^{n-|\mathrm{LS}|} - 2$.

Proof. Let $D' = \{S \in D \mid \text{site } (S) \cap \mathrm{LS} = \emptyset\}$. Then, as in the proof of Theorem 4, it can be shown that $|\{S \in D' \mid f(S) = z\}| \geqq 2^{n-|\mathrm{LS}|} - 2$. Since f is an RDTP, there exists $S \in D$ such that $f(S) = z$ and site $(S) \cap \mathrm{LS} \neq \emptyset$. Q.E.D.

THEOREM 8. *Let f be an* RDTP. *Then,* CM $(f) >$ CM (h).

Proof.

$$\mathrm{CM}\,(h) = \sum_{k=1}^{n-|\mathrm{LS}|} \binom{n-|\mathrm{LS}|}{k} = 2^{n-|\mathrm{LS}|} - 2 < \mathrm{CM}\,(f). \qquad \text{Q.E.D.}$$

THEOREM 9. *Let f be an* RDTP. *Then,* SM $(f) >$ SM (h) *for sufficiently large n.*

Proof. The DTP u is site optimal among all DTPs by Theorem 7. It can be proved that u, when restricted to the centralized commit protocol, remains site optimal among RDTPs. Thus, it is sufficient to show that SM $(u) >$ SM (h) in the restricted domain. The following inequality holds for the restricted u:

$$\begin{aligned}\mathrm{SM}\,(u) &\geqq \sum_{k=1}^{n-k_0} k\cdot 2^{k-|\mathrm{LS}|}\binom{n}{k}+\sum_{k=n-k_0+1}^{k_0-1} k\cdot\binom{n}{k}\\ &=\sum_{k=1}^{n-1} k\cdot\binom{n}{k}+\sum_{k=1}^{n-k_0} k\cdot(2^{k-|\mathrm{LS}|}-1)\binom{n}{k}-\sum_{k=1}^{n-k_0}(k+k_0-1)\binom{n}{k+k_0-1}.\end{aligned}$$

On the other hand,

$$\mathrm{SM}\,(h)=\sum_{k=1}^{n-|\mathrm{LS}|} k\cdot\binom{n-|\mathrm{LS}|}{k}\leqq\sum_{k=1}^{n-1} k\binom{n-1}{k}.$$

Thus, SM $(h) <$ SM (f) for sufficiently large n. Q.E.D.

5. Conclusion. Network partitioning is harder to deal with than many other failure problems in a distributed environment. Though its occurrence is not very frequent, it cannot simply be ignored in certain applications. But, none of the existing systems have tried to consider the network partitioning problem. Being termed as a "catastrophic situation," it is manually handled [11].

In this paper, we have formalized various aspects of network partitioning problems and designed effective protocols to be used in the presence of this failure. Certain practical criteria are used for measuring the performance of termination protocols. Both decentralized and centralized versions of commit protocols are studied and optimal termination protocols are presented. For the centralized case, a single TP is shown to be optimal under both component and site measures. In the absence of coordinator failures, this optimal protocol guarantees continued transaction processing in at least one component in any partitioning. Quorum protocols are shown to be optimal in the decentralized case. This is not greatly satisfying since one cannot guarantee continued processing in any component in this case. Thus, it is desirable that a centralized TP be used when the coordinator sites are reasonably failure-free. This in turn would imply that a centralized commit protocol be employed during the normal (failure-free) operations of the system. Hence, our results can also be interpreted as lending support to the usage of centralized protocols—not only because they are less expensive (in terms of messages) than the decentralized ones, but also due to their effectiveness in handling network partitionings.

From our bounds on the component and site measures, it is not hard to see that the fraction of waiting components/sites in the event of an arbitrary network partitioning is very small when the optimal TPs are used.

We have also proved that any commit protocol should have a "committable" state to be of use in presence of partitioning. This tends to increase the cost of the normal operation. But, one can fine-tune the protocol by running FSAs with committable states only at certain critical sites.

The measures we have used involve no statistical information. They are somewhat simplified approximations of the actual system availability. Generalizations of these measures, which utilize the statistical information on partitionings and the states of the commit protocol are considered elsewhere [5].

REFERENCES

[1] P. ALSBERG AND J. DAY, *A principle for resilient sharing of distributed resources*, Proc. 2nd International Conference on Software Engineering, October 1976, pp. 562-570.

[2] P. A. BERNSTEIN AND N. GOODMAN, *Concurrency control in distributed database systems*, Comput. Surveys, 12 (1981), pp. 185-221.

[3] P. A. BERNSTEIN AND D. W. SHIPMAN, *A formal model of concurrency control mechanisms for database systems*, Proc. 3rd Berkeley Workshop on Distributed Data Management and Computer Networks, August 1978.

[4] F. CHIN AND K. V. S. RAMARAO, *An information-based model for failure handling in Distributed databases*, IEEE Trans. Software Engrg., to appear.

[5] ———, *Maximization of database availability in presence of network partitioning*, submitted to J. Assoc. Comput. Mach.

[6] E. C. COOPER, *Analysis of distributed commit protocols*, Proc. ACM SIGMOD, 1982, pp. 175-183.

[7] S. DAVIDSON AND H. GARCIA-MOLINA, *Protocols for partitioned distributed database systems*, Proc. IEEE Symposium on Reliability in Distributed Software and Database Systems, June 1981.

[8] H. GARCIA-MOLINA, *Reliability issues for completely replicated distributed databases*, Proc. IEEE COMPCON, Fall 1980, pp. 442-449.

[9] ———, *Elections in a distributed computing system*, IEEE Trans. Comput., C-31 (1983), pp. 393-481.

[10] J. N. GRAY, *Notes on database operating systems, operating systems: an advanced course*, Lecture Notes in Computer Science 60, Springer-Verlag, New York, pp. 393-481.

[11] M. HAMMER AND D. SHIPMAN, *Reliability mechanisms for* SDD-1, ACM Trans. Database Systems, 5 (1980), pp. 431-466.

[12] B. LAMPSON, *Atomic transactions, distributed systems architecture and implementation: an advanced course*, Lecture Notes in Computer Science 100, Springer-Verlag, New York, Chapter 11.

[13] K. V. S. RAMARAO, *On the completion of distributed transactions while recovering from a network partitioning*, Proc. 1984 Princeton Conf. on Information Sciences and Systems.

[14] H. R. STRONG AND D. DOLEV, *Byzantine agreement*, Proc. IEEE COMPCON, Spring 1983, pp. 77-82.

[15] D. SKEEN, *Nonblocking commit protocols*, Proc. ACM SIGMOD, 1981, pp. 133-147.

[16] ———, *A quorum-based commit protocol*, Computer Science TR 82-483, Cornell University, Ithaca, NY, 1983.

[17] D. SKEEN AND M. STONEBRAKER, *A formal model of crash recovery in a distributed system*, IEEE Trans. Software Engrg. TSE-83.