TRANSACTION COMMITMENT AT MINIMAL COMMUNICATION COST

(Preliminary Version)

by

A. Segall and O. Wolfson

Technical Report #441
December 1986
TRANSACTION COMMITMENT AT MINIMAL COMMUNICATION COST

Adrian Segall
Ouri Wolfson
The Technion - Israel Institute of Technology,

ABSTRACT

We consider the communication protocol for transaction commitment in a distributed database. Specifically, the connection between the structure of communication among the participating sites, and the communication network topology is investigated. In order to do so, the cost of transaction commitment is defined as the number of network hops that messages of the protocol must traverse. We establish the necessary cost for transaction commitment, and show that it is also sufficient. A simple distributed algorithm is presented to prove sufficiency. Our algorithm is also time-efficient, and in order to prove that we show that the timing of our algorithm is optimal within a natural class of commit-protocols.
1. INTRODUCTION

1.1 Background

In a distributed database a transaction consists of several subtransactions, each running at a different site. The commitment problem (see [G]) arises when each local database manager has to decide whether to make the changes to each local database permanent (i.e. commit the transaction), or not (i.e. abort). When the subtransaction at a site terminates the local database manager is aware of its vote - 'yes' or 'no'; it indicates whether the subtransaction completed successfully. The generally accepted solution to the problem is to commit the transaction if all subtransactions voted 'yes', and abort it otherwise.

Transaction commitment is a variation of a more fundamental problem in distributed systems, namely distributed consensus. This problem has been studied extensively in the literature. [F] presents a survey of the field, and [DS2] presents an interesting taxonomy of consensus problems. Almost all research on the subject concentrates on the effects of failures on the consensus problem. What types and number of failures allow a solution to the consensus problem (e.g. [FLP])? How consensus should be obtained, in the face of failures which do not preclude it (e.g. [CL])? Here we assume that failures of the computing environment components (everything except the transaction) are rare, detectable and handled separately by means outside the scope of this paper.

Informal discussions of transaction commitment in the absence of failures appears in [G], [CP], [S], mainly in the context of different two-phase-commit schemes. One of the most popular is the central site scheme. A designated "protocol coordinator" polls all sites which participate in running the transaction. In response, each participating site sends its vote to the coordinator. The coordinator makes the decision, and sends the 'commit' or 'abort' message to all other sites. Another scheme is the decentralized. In it each participating site sends its vote to all other participating sites. Based on the received messages each site makes the 'commit' or 'abort' decision. Finally, we will mention the linear scheme, in which all participating sites are sequentially ordered. Each site sends its vote to the next site in the sequence. The last site is the protocol coordinator, which reverses the flow direction, by sending the decision message to its predecessor in the sequence. [G] mentions that the linear (it is called nested there) scheme is most efficient in terms of the number of messages it requires. This number is 2(n-1) where n is the number of participating sites. In [DS1] it was formally proven that the number of messages required by any transaction commitment protocol in the absence of failures...
is $2(n-1)$.

1.2 The Problem

The problem with existing transaction commit schemes is that they do not take the communication network topology into consideration. For example, assume that the communication network consists of sites 1 through $n$ linked in a string. In other words, each site $i$ is connected by a bidirectional link to site $i+1$. Consider the linear two-phase-commit scheme executed in a database distributed over this network. It tells us nothing about the connection between the sequence of sites established in the protocol, and the topology of the network. Although it is intuitively clear that in order to reduce communication, the sequence of sites in the protocol should be 1 through $n$ (or $n$ through 1) the scheme allows any permutation as the sequence. Moreover, as far as the number of messages is concerned, it indeed does not matter! Any sequence will require $2(n-1)$ messages. This is strange. The reason for this anomaly is that messages which have to pass many hops to reach their destination, and messages which do so in one hop are treated equally, and counted as one message. We feel that the difference should be reflected in the communication-cost of the message, and one should attempt to minimize the total communication cost of messages sent by a commitment protocol.

The question immediately arising is the following. Given a network and a subset of sites participating in the commitment of a transaction, what is the necessary communication-cost for a commitment protocol execution (or an instance)? In this paper we establish this cost to be twice the weight of a minimum spanning tree. It is the subtree of a complete graph, representing the distance in the network between every pair of sites participating in the commitment protocol. We obtain this result by modeling an instance as a set of message-send events, temporally ordered by the "happens-before" relation (in the sense introduced by Lamport in [L]). Each intersite message of the instance has a cost, representing the distance in the communication network between the sending and receiving sites.

The next question arising: Is the necessary cost also sufficient? We answer this question positively, by presenting a simple distributed algorithm which achieves the necessary cost. However, our algorithm does more: it is also communication-time efficient. In the modeling of the communication-time of an instance, our work also differs from others ([DS1], [R]). There, each message between two sites takes constant time, whereas here, the travel-time of a message may vary depending on the distance in the network between the sender and the
receiver. The other works have also implicitly assumed that all subtransactions complete simultaneously. Our model considers (more realistically, we feel) that different subtransactions complete at different times. We therefore define the communication-time of an instance, as the number of time-units from the completion of the first subtransaction, until the last site commits its subtransaction. Based on this definition, we proceed to prove that our algorithm executes an instance, \( I \), of minimal communication-time, in a class of minimal communication-cost instances. It is possible that \( I \) does not have the absolute optimal communication time among all instances of minimal communication cost. However, note that the absolute optimal communication time instance depends on the minimal subtransaction completion time, and in order to establish it a site needs all subtransaction completion times. This information usually is not available at a site when its subtransaction completes.

The rest of the paper is organized as follows. In Section 2 we present the system model. In Section 3 we establish the necessary communication-cost for the transaction commitment problem, and in Section 4 we present and discuss the distributed algorithm. In Section 5 we define the notion of communication-time, and prove communication-time minimality of our algorithm in a class of minimal cost instances. Section 6 discusses future work.

2. THE MODEL

Let \( V \) be a set of computer identifications and \( L \) a set of unordered pairs of \( V \), representing communication links. \( G = (V, L) \) is a communication network graph. We assume that a transaction in a distributed database executes at a subset of participating sites \( B \subseteq V \). When the transaction completes, the database management system executes a commitment protocol at the sites of \( B \), to decide whether to commit or to abort the transaction. The discussion is restricted to the case where no failures occur from the time the transaction begins, until it is committed or aborted at every site. We will analyze the case in which each site votes to commit the transaction. In Section 4 we will discuss the extension of the results to the 'abort' case. At this point we will mention that we concentrate on the 'commit' case, since as we show in section 4, it is more expensive (from the communication cost point of view) than the 'abort' case. Therefore the necessary cost for a 'commit' execution is necessary for the consensus problem.
Intuitively, an instance is represented by the temporal partial order of send-events in an execution of the commit protocol. By a send-event we mean the sending of a message to one or more sites, namely a multicast. Assume that a message from site $i$ to site $j$ has been sent during the protocol, by $r'$th send-event at site $i$. The message is represented by the ordered pair $(S_{ir'}, S_{jr'})$. We omit Receive events from the model, but we assume that the message had been received before $S_{jr'}$ and after the send-event immediately preceding it at site $j$. We further assume that a message sent by event $S$ contains (or implies) the votes sent by all events which "happen before" $S$ (in the sense of Lamport).

Formally, an instance of the commit protocol, $I$, is a directed acyclic graph (dag). For each site $i \in B$ the dag $I$ has $S_{i1} \rightarrow S_{i2} \rightarrow \cdots \rightarrow S_{ik}$ as a subgraph, where each $k_i$ is some integer $\geq 2$. The $S_{ij}$'s are the send-events occurring at site $i$. An arc $(S_{ij}, S_{ij+1})$ is called an intrasite arc and it represents the order in which the two events occur at site $i$. The other arcs of $I$ are intersite arcs. An intersite arc is an arc between send events occurring at different sites; it represents a message. $I$ satisfies the following correctness requirement: At each site $i \in B$ occurs a send-event preceded (in $I$) by at least one send-event from every site of $B$. A send-event which is preceded by at least one send-event from each site is called a C-send (send of a Commit message). An event which is not a C-send is called a V-send (send of Vote message). Note that an event which is succeeded by a C-send is also a C-send. Therefore, the send-events at a site can be also denoted $V_{i1}, \ldots, V_{im}, C_{l1}, \ldots, C_{lm}$ for $l_1, m \geq l$. The V-subgraph of an instance $I$ is the subgraph induced by the V-send events. Similarly, the C-subgraph of $I$ is defined. This model is sufficiently powerful to represent the variations of the two-phase commit protocol mentioned in the introduction. They are illustrated in Figure 1.

The communication cost of an arc $a \rightarrow b$ in an instance is zero if $a$ and $b$ occur at the same site. If they occur at different sites $i$ and $j$ of $B$, then the communication cost of the arc is the length of the shortest path in the network $G$ between $i$ and $j$. The communication cost of an instance $I$, denoted $\text{Cost}(I)$, is the total communication cost of messages in $I$. Denote by $\text{COMMIT}(G,B)$ the set of instances for the subset of participating sites $B$, in the network $G$. 

Technion - Computer Science Department - Technical Report CS0441 - 1986
Proof: Let $I$ be a $mcc$ instance. Based on Lemma 1 we can assume w.l.g. that for each site $i$, $I$ has exactly one coordinator. By definition of an instance, at least one of these arcs must have communication cost greater than zero. Such an arc can be omitted from $I$ to obtain a correct instance of lower cost; contradiction to cost-minimality of $I$. □

Assume that $C_{j1}$ is a root of the $C$-subgraph of some $mcc$ instance $I$. This means that in the execution of $I$, site $j$ knows all votes without receiving a commit message. In such case we say that site $j$ is a coordinator of the instance $I$.

Lemma 2: There exists an instance of minimal communication cost which has exactly one coordinator.

Proof: Let $I$ be a $mcc$ instance. Based on Lemma 1 we can assume w.l.g. that for each site $i$, $I$ has exactly...
one C-event, \( C_i \), from which all intersite arcs representing commit messages exit. Assume that \( I \) has two or more coordinators (otherwise the proof is completed). Then \( I \) has two or more C-events none of which is preceded by a C-event. Call these boundary C-events. Consider a V-send, \( V_0 \), which precedes two or more boundary C-events, but each one of its V-send successors precedes only one boundary C-event. There must be such. Assume that \( V_0 \) precedes \( C_i \) and \( C_j \), and that \( V_0 \) occurs at site \( k \). There is only one \( C_k \) event, and the C-subgraph of \( I \) is a forest, thus \( C_k \) cannot be in the tree rooted at \( C_i \) and in the tree rooted at \( C_j \). Assume w.l.g. that \( C_k \) is not in the subtree rooted at \( C_i \). Denote the events on a path from \( V_0 \) to \( C_i \) by \( V_1, \ldots, V_n \). Let \( V_r \) be the last event on this path for which the following is true: \( V_r \) occurs at a site \( d \) for which \( C_d \) is in a subtree different than the one rooted at \( C_i \). Since \( V_0 \) satisfies the condition, there must be such \( V_r \). Denote the site at which \( V_{r+1} \) occurs by \( b \). By definition of \( V_r \), the event \( C_b \) is in the subtree rooted by \( C_i \). Replace the arc \( V_r \rightarrow V_{r+1} \) by the arc \( C_d \rightarrow C_b \) of the same length. If \( b \neq i \), reverse the direction of the arcs from \( C_j \) to \( C_b \). These modifications are illustrated in Fig. 2.

We claim that by these modifications a well defined instance, \( I' \), is obtained. Since the C-subgraph of \( I \) is a forest, we did not introduce any cycles in \( I' \). Also, the correctness requirement holds for \( I' \) for the following reason. By definition of \( V_0 \), the event \( V_r \) precedes in \( I \) only one boundary C-event, \( C_i \). Removal of the arc \( V_r \rightarrow V_{r+1} \) can only disconnect paths to the C-events in the C-subgraph tree rooted at \( C_i \). However, all those C-events are now preceded by \( C_b \). Therefore \( I' \) is a correct instance. By the transformation performed on \( I \) to obtain \( I' \), it is clear that \( \text{Cost}(I) = \text{Cost}(I') \). To summarize, starting with a mcc instance, \( I \), with two or more boundary C-events, we obtained an mcc instance, \( I' \), with one less boundary C-event. Therefore, there is an mcc instance with one boundary C-event.  

Denote by \( \text{MST}(G,B) \) a minimum spanning tree (mst) of the weighted graph \( D = (B,E) \), defined next. \( E = \{(i,j) | i,j \in B \} \), and the cost of the edge \( (i,j) \) is the shortest path in \( G \) between \( i \) and \( j \). \( D \) is the distance graph of the set \( B \) in \( G \). Denote the cost of \( \text{MST}(G,B) \) by \( \text{CMST}(G,B) \). For the proof of the main theorem, given below, we need the following definition. Given a mst of \( D \), \( T \), we define an instance on \( T \) coordinated at some site \( k \in B \). It is denoted \( I(T,k) \), and specified as follows. Consider \( T \) with its edges directed such that from every node there is a path to \( k \). Call it an oriented tree with sink \( k \). The V-subgraph of \( I(T,k) \) consists of this oriented tree with each node \( i \) of \( T \) replaced by \( V_i \). Its C-subgraph consists of \( T \) with all edges
Proof: Since the cost of an instance on some mst coordinated at a site is $2\cdot \text{CMST}(G,B)$, then clearly

Theorem 1: Let $G = (V,L)$ be a communication network. Then $2\cdot \text{CMST}(G,B) = \min_{I \in \text{COMM}(G,B)} \text{Cost}(I)$. 

Figure 2: Obtaining a minimal communication cost instance with one less boundary C-send.

...
8

The result of Dwork and Skeen ([DS1, Theorem 1]) is given by the following corollary from Theorem 1.

Corollary 1: If the number of participating sites is \( n \), and the distance between each pair of sites is one, then

\[
\min_{I \in \text{COMMIT}(G,B)} \text{Cost}(I) = 2(n-1).
\]

4. THE TRANSACTION COMMIT ALGORITHM

Note that the analysis of the previous section suggests the following mcc transaction-commit algorithm, which we call FIXED-INSTANCE. Each participating site starts out the protocol with the same instance on some mst \( T \), coordinated at some site \( k \). In the protocol, site \( i \) behaves as dictated by the common instance: it
waits until receiving all messages represented by the intersite arcs incoming into \( V_i \); then it sends the received votes, along with its own, in the message represented by the arc exiting \( V_i \). The behavior after the commit is similar.

This FIXED-COORDINATOR algorithm can be improved in a sense which will become clear in the next section. In the rest of this section we describe the superior algorithm, called TREE-COMMIT, prove it executes an instance of minimal communication cost, and discuss its extension to the abort case.

TREE-COMMIT is a distributed mcc algorithm, which works as follows. Each participating site \( i \) starts out the protocol with a mst, \( T \), of the distance graph \( D \) (rather than a fixed instance). We suppose that each site knows which are all the participating sites, and what is the network topology, after completing its subtransaction. Therefore, \( T \) can be computed locally, before the protocol is started; assuming that all sites run the same algorithm, \( T \) is the same in all participating sites. Each site casts its vote as soon as possible after the completion of the subtransaction at its site, while ensuring that no more than two messages along a tree edge are necessary. This is accomplished if each site waits until receiving the votes from all its neighbors in \( T \), except one, before

---

Figure 4: An example of an instance in the voting stage, as executed by the TREE-COMMIT algorithm. Suppose that sites 1, 2, 3, 4, 5, 7, and 11 have completed the subtransaction, and the others have not. The figure indicates that sites 1, 2, 3, 4, 5, 11 have voted, and the others have not. Note that site 9 has not voted yet because it did not complete the subtransaction, and site 7 has not voted yet because it did not receive messages from all neighbors, except one.
voting. It implies that the leaves of $T$ vote immediately after completing the subtransaction. The votes travel from the leaves towards the "center" of $T$, where the coordinator(s) of the instance is (are) determined. An instance of TREE-COMMIT may have one (Fig. 3) or two (Fig. 6) coordinators. A possible situation in the voting stage (i.e. before a coordinator is determined) is illustrated in Fig. 4. In the commit stage, the commit message is simply propagated along the tree edges away from the coordinator(s).

Formally, in the algorithm TREE-COMMIT, each participating site executes the procedure TREE-COMMIT(T) of Fig. 5. Each step of the procedure specifies the "when and where" of vote and commit messages; when the message is sent, and to which sites.

**Theorem 2:** An instance of the algorithm TREE-COMMIT has minimal communication cost.

---

**TREE-COMMIT(T)**

1. Site $i$ sends a 'yes' vote (assuming that it had successfully completed the subtransaction), after having received 'yes' votes from all its neighbors in $T$, except one, $j$. Then the vote is sent to $j$.

2. Site $i$ sends a "commit" message in any of three cases:

   2.1 It had received 'yes' votes from all its neighbors in $T$, and after that, had successfully completed the subtransaction. Then, it sends the "commit" message to all neighbors (becoming the only protocol coordinator).

   2.2 It had received a 'yes' vote from neighbor $j$, after having sent a 'yes' vote to $j$. Then it sends the "commit" message to all neighbors, except $j$ (becoming one of the two coordinators - the other one is $j$).

   2.3 It had received the "commit" message from a neighbor. Then it sends the "commit" message to all other neighbors.

**Figure 5:** Algorithm TREE-COMMIT executes the above procedure at every site $i$ which participates in the transaction. The parameter $T$ is a minimum spanning subtree of the Distance-Graph, common to all sites.
Proof: We have to show that exactly two messages are transmitted along each edge of the tree $T$. If the instance executed has one coordinator, then it is clear that the algorithm sends (in opposite directions) one vote message and one commit message along each edge. If the instance has two coordinators, then this is the case for all edges of $T$, except one. This is the edge between the two coordinators. Along it no commit messages are sent; instead, two vote messages are sent in opposite directions. ☐

Note that dynamic determination of the coordinator can speed up the traditional linear two-phase-commit, without increasing communication cost or number of messages, regardless of the network topology. (We obviously use an intuitive notion of "speed-up"). In this modified scheme the sites are still numbered 1 through $n$, but each site, except the leaves, votes as soon as it completes its subtransaction and receives the vote of one of its neighbors.

Abort Case: This case is handled by amending TREE-COMMIT($T$) by the following step:

3. Site $i$ sends an 'abort' message after having received an 'abort' message, or after unsuccessfully terminating its subtransaction. It is sent to all neighbors from whom site $i$ received a 'yes' vote, and to all neighbors from which it did not receive a vote.

Theorem 3: The communication cost of an abort instance is at least $CMST(G,B)$ and at most $2CMST(G,B)$.

Proof: Omitted.

5. COMMUNICATION TIME COMPLEXITY

In this paper we are primarily interested in minimal communication cost. We have shown two such commit algorithms: FIXED-INSTANCE and TREE-COMMIT. TREE-COMMIT is superior in terms of our secondary criterion, namely communication-time. In this section we first define this new term for an arbitrary instance. Then we show in theorem 4, that TREE-COMMIT($T$) executes a $T$-instance of minimal communication-time, among all $T$-instances. (For a mst $T$, the $T$-instances are the mcc instances in which communication is confined to take place between neighbors in $T$.) On the other hand, FIXED-INSTANCE executes some $T$-instance, which is not necessarily the best in terms of communication-time. Finding the best $T$-instance locally, at each participating site, is impossible because it depends on all subtransaction completion times (see definition of communication-time below); this information is generally not available when FIXED-INSTANCE
starts out the protocol.

We start with some definitions. Intuitively, the communication-time of an instance is the number of time units required from the time the first subtransaction completes, until the last site commits. A time unit is the time required for a message to travel one communication-network hop. The communication-time of an instance $I$ is defined w.r.t. a set $\tau = \{\tau_1, \ldots, \tau_n\}$ of subtransaction relative completion times, or completion times for short. Each $\tau_i$ is a nonnegative integer, representing how many time units after the first, the subtransaction at site $i$ had completed; some $\tau_j = 0$ indicating that the subtransaction at site $j$ completed first. The definition of the instance-time is based on transforming an instance into a PERT-digraph (see [E]). (We omitted the "termination vertex" to simplify the notation.) The reason for this transformation is that an instance consists of a set of activities, each of which takes time, and some of the activities go on in parallel. Formally, the communication-time of an instance $I$ w.r.t. $\tau$ is defined based on a dag, denoted $PJ$. It is built by augmenting $I$ with a source node $s$, and an arc exiting from $s$ to the first $V$-send of every site. The arcs in $PJ$ have a length, and the communication-time is the longest path in $PJ$. An arc of $PJ$ which is also in $I$, has length equal to its communication cost. An arc from $s$ to the first $V$-send at site $i$ has length $\tau_i$.

Next we define a $T$-instance for a given mst $T$ of the Distance-Graph. An instance on $T$ coordinated at some site is a $T\text{-}instance$. A mcc instance on $T$ coordinated at a pair of neighbors in $T$, as illustrated in Fig. 6, is also a $T\text{-}instance$.

Note that w.r.t. a given set of completion times, different instances may have different communication-times. For example, consider the $T$-instances in Figures 3(b) and 6(b). Assume that all completion times are zero, and the length of each edge of $T$ (in Figures 3(a) and 6(a)) has a cost of one. Then the communication-time of the instance in Fig. 3(b) is 4 (the critical path is $V_1V_2V_4C_4C_2C_1$ with the arc $V_4\to C_4$ of length zero), and the communication-time of the instance in Fig. 6(b) is 3 (the critical path is $V_1V_2C_4C_2$).

Theorem 4: For any fixed set of completion times a committing instance of TREE-COMMIT($T$) is a $T$-instance of minimal communication time, among all $T$-instances.

Proof: Proof of Theorem 4: Denote by $I^*$ the instance executed by TREE-COMMIT($T$). The fact that $I^*$ is a $T$-instance is obvious. The fact that $I^*$ has minimal communication-time will become clear, once we identify it. $I^*$ is dependent on the set of completion times, denoted $\tau = \{\tau_1, \ldots, \tau_n\}$, and this is also the reason it is optimal for $\tau$. For its identification we need he following preliminaries.
Denote by $T_s$ the graph obtained from the tree $T$ by augmenting it with the node $s$, and with edges $(s,i)$ for each node $i$ in $T$. The length of each edge $(s,i)$ is $\tau_i$. For each node $i$ denote by $p_i$ some longest path from $s$ to $i$ in $T_s$. Label a node $i$ of $T$ by $b(i)$, defined as follows: If $p_i$ consists of the edge $(s,i)$, then $b(i) = \tau_i$; else $b(i)$ is the length of the longest path in $T_s$ from $s$ to $i$, which does not go through the last edge (i.e. the one incident upon $i$) in $p_i$. For some event $Y$ in $PJ$ denote by $l(Y)$ the length of the longest path from $s$ to $Y$ in $PJ$. It is clear from step 1 of TREE-COMMIT that $I^*$ is an instance for which the following holds: in $PJ^*$, $l(V_i) = b(i)$ for each $i \in B$.

Denote by $d(i)$ the longest path in $T$ from $i$. Given an instance $I$, denote by $cp(PJ)$ the longest (critical) path in $PJ$. If $I^*$ has one sink of the $V$-subgraph, $V_r$, then $cp(PJ^*) = l(V_r) + d(r) = b(r) + d(r)$. If it has two sinks of the $V$-subgraph, $V_q$ and $V_r$, then $cp(PJ^*) = \max\{b(r) + d(r), b(q) + d(q)\}$. In any case,

$$cp(PJ^*) \leq \max_{i \in T}\{b(i) + d(i)\}$$

Now consider $PJ$ for some arbitrary $T$-instance, $I$. Let $i$ be an arbitrary participating site. Then to each arc incident upon $V_i$ in $PJ$ corresponds an edge incident upon $i$ in $T_s$, and vice versa. Additionally, $V_i$ in $PJ$ has at most one incident arc directed from it to a neighbor. The other incident arcs are directed towards $V_i$. Now consider some path $p$ from $s$ to $i$ in $T$. There is a path of corresponding arcs (i.e. of the same length) in $PJ$, unless $p$ goes through the edge corresponding to the arc directed away from $V_i$. Therefore, by the way we
defined \( b(i) \), it is a lower bound on \( l(V_i) \) in \( P_J \). Also, in \( P_J \) there is a path from \( V_i \) to each \( C_j \), which consists of arcs corresponding to the edges of \( T \). Thus, \( cp(P_J) \geq b(i) + d(i) \). But recall that \( i \) was an arbitrary participating site. Therefore, \( cp(P_J) \geq \max_{i \in ?} \{ b(i) + d(i) \} \), which by inequality (1) implies that \( cp(P_J) \geq cp(P_J^*) \).

\[ \square \]

Note that TREE-COMMIT does not guarantee an instance of minimal communication time among all \( mcc \) instances, because it is confined to a prespecified tree. However, determining such an instance at a site can be done only when the site knows all subtransaction completion times, and passing this information would require extra communication, which in turn would violate our primary goal, namely minimal communication cost.

6. FUTURE WORK

We conjecture that the communication-time of TREE-COMMIT is in some sense optimal, if minimal communication cost is desired. The reason is as follows. At some point in time after completing its subtransaction, a participating site \( i \) sends its vote to another participating site \( j \). Site \( i \) must have a set of candidates that may receive \( i \)'s vote, and \( j \) is chosen to optimize communication time, while it is guaranteed that minimal communication cost does not have to be exceeded (for TREE-COMMIT(T) the candidates are all neighbors of \( i \) in \( T \)). Denote each such candidate \( k \) by an arc \((i,k)\), and consider the "voting" digraph, i.e. the graph consisting of the union of all candidate arcs in all sites (for TREE-COMMIT(T) the voting digraph is \( T \), when each edge of \( T \) represents two directed arcs). We feel that for any distributed commit algorithm, if the voting graph has a simple directed cycle of three or more sites, then minimal communication cost cannot be achieved. Note that a minimum spanning tree of the distance graph is a voting graph in which there is no such cycle; all simple cycles consist of two sites. Moreover, an undirected tree is a most general voting graph with this property, and the conjecture stems from this fact.

Once a coordinator \( r \) receives all votes, it propagates the commit message along the edges of a "commit" graph, which must be a tree rooted at \( r \). This commit tree can theoretically differ from the voting graph, in order for the coordinator to be its "center". However, finding such a commit tree would involve solving an NP-complete problem.
ACKNOWLEDGEMENT

We wish to thank Alon Itai for very helpful discussions and insightful remarks on earlier versions of the manuscript.

REFERENCES


