COMMUNICATION COST AND TIME OF COMMITMENT ALGORITHMS
(Revised Version of TR # 458)

by

O. Wolfson and A. Segal

Technical Report #551
May 1989
COMMUNICATION COST AND TIME OF COMMITMENT ALGORITHMS

Ouri Wolfson

Adrian Segall

Department of Computer Science
Technion - Israel Institute of Technology
Haifa 32000, Israel

ABSTRACT

We consider the communication protocol for transaction commitment in a distributed database. Specifically, we investigate the connection between the structure of communication among the participant sites, and the communication network topology. In order to do so, the cost of transaction commitment is defined as the number of network links that messages of the protocol must traverse (regardless of whether such link connects a participant, or a non-participant, site). We establish that the necessary cost for transaction commitment is twice the cost of a minimum spanning tree. Then we show that the necessary cost is also sufficient for commitment, by presenting a simple distributed algorithm called TREE-COMMIT. Next, we consider the communication-time of commitment algorithms and prove that the time of TREE-COMMIT is the best one can obtain, in the class of simple algorithms which have minimum communication cost.

1. This research was supported by the Foundation for Research in Electronics, Computers, and Communications, administered by the Israeli Academy of Sciences and Humanities.
2. This research was supported by the Technion Fund for the Promotion of Research.
1. INTRODUCTION

In a distributed database, a transaction consists of several subtransactions, each running at a different site that has a local database. When the subtransaction at a site completes, i.e. the corresponding process finishes, the local database manager knows whether it completed successfully or unsuccessfully\(^1\). The *atomic commitment* problem (see \([G]\)) for a local database manager, is to decide whether to make the changes to the local database permanent (commit the subtransaction), or invalidate them (abort the subtransaction). The generally accepted solution to the atomic commitment problem is to commit all the subtransactions, yielding a committed transaction, if all the completions are successful, and abort all of them otherwise, yielding an aborted transaction. To achieve atomic commitment, the local database managers execute a distributed protocol, exchanging 'yes' and 'no' votes. A 'yes' vote indicates the successful completion of a subtransaction, and a 'no' vote indicates an unsuccessful completion\(^2\).

The model considered in this paper is of a database distributed over a subset of the nodes in a computer-communication network. Each subtransaction of a given transaction runs at some node, called a *participant* site for that transaction. The collection of all participant sites for a given transaction may be a proper subset of all the network nodes. For example, in Fig. 1.1 we illustrate a transaction in a communication network. Squares represent participant sites, circles represent nonparticipant sites, and edges represent two-way communication links.

---

1. For example, a subtransaction may complete unsuccessfully as a result of deadlock.
2. Transactions are used in distributed systems other than databases, such as Argus and Camelot ([Sp]). The atomic commitment problem, as described, arises in such systems as well.
Consider first the communication cost of the transaction commitment protocol. Traditionally, the communication cost has been quantified in terms of the number of intersite messages. These are logical messages among participant sites. This cost measure disregards the physical network topology, particularly, the distance in the network between participant sites. But, the actual communication load on the network is reflected by the number of messages between neighbors in the network (participants or nonparticipants). These messages are referred to in this paper as network messages.

The following examples show how three commitment executions, or instances, may use the same number of intersite messages but require different numbers of network messages. Assume that the transaction occurs on the computer-communication network given in Fig. 1.1. A possible commitment protocol execution, based on the "central site" scheme, is illustrated in Fig. 1.2a: each one of the sites 2, 3, and 4 sends its 'yes' vote directly to site 1, which also votes 'yes', commits, and sends the commit decision separately to each of sites 2, 3, and 4 (each of which commits when it receives the message). This execution requires 6 intersite messages. Another possible execution, based on the "linear" protocol of Gray ([G]), is illustrated in Fig. 1.2b: 4 sends its 'yes' vote to 1, which then sends its and 4's 'yes' votes to 3, which then sends its and 1's and 4's 'yes' votes to 2; 2 then votes 'yes', commits, and sends the commit decision to 2, which sends it to 3, which sends it to 4. This instance also takes 6 intersite messages. In fact, from the lower bound result of Dwork and Skeen [DS1] for atomic commitment, we know that any four-participant atomic commitment execution requires at least six intersite messages. Note, however, that the load that these instances place on the communication network is higher than the following instance (see Fig. 1.2c): 4 sends a 'yes' vote to 3, which sends its and 4's 'yes' votes to 2; meanwhile, 1 also sends a 'yes' vote to 2, which, after receiving the two messages, also votes 'yes' and commits, after which the commit messages travel from 2 to 3 to 4 and from 2 to 1. If we assume that the intersite messages travel through the shortest path in the network, then the first instance takes 20 network messages, the second takes 22, and the latter takes only 14.

The communication cost measure in this paper is taken to be the total number of network messages dictated by the commit instance, assuming that the intersite messages propagate on the shortest path in the network between the sender and receiver sites. This measure can be easily generalized in a network with dynamic routing, in which different messages from the same sender to the same receiver, may propagate on different paths. In this case the communication cost of an intersite message can be the average of the number of network messages required, over all possible routing paths between the sites.
The following question immediately arises. Given a computer communication network, a subset of the sites of which are participants of a transaction, what is the necessary communication cost of atomic commitment, in any atomic commitment protocol? In this paper we establish this cost to be twice the weight of a minimum spanning tree in a complete graph; the graph represents the distance in the network between every pair of participant sites. For example, in Fig. 1.3 we show the distance graph of the participant sites and network of Fig. 1.1. As explained at the beginning of Section 3, this result is not trivial because, as we shall show, an atomic commitment execution of minimum communication cost, does not necessarily have a single coordinator (a participant to which all other participants send their votes). In fact, the most difficult part of the lower bound proof is showing that there always exists a minimum communication cost instance with one coordinator. This, and other proofs in this paper, are based on a novel model of a commitment protocol instance, introduced in this paper. Then, we show that the necessary cost is also sufficient for atomic commitment, by presenting a simple distributed protocol, TREE COMMIT, which achieves exactly the necessary cost.

The other measure to be considered in comparing the performance of commit protocols is the execution time. Execution time of an instance is defined as the interval of time starting when the first subtransaction completes, and ending when the last site commits its subtransaction. In this respect we also depart from traditional models ([DS1], [R]), which assume a synchronous communication network, and simultaneous completion of all subtransactions. The synchronous communication implies, in particular, a unit-time intersite message delay, independent of the network location of the sender and receiver.
Fig. 1.3

In our model we allow, more realistically, arbitrary subtransaction completion times as well as different network delays. Particularly, different participants may complete their subtransaction at different times, and intersite message delays may differ, depending on the sender and the receiver. Therefore, we dispose of unrealistic assumptions regarding synchronous operation of geographically dispersed processors. We show that the execution time of TREE-COMMIT is the best one can practically obtain in the class of simple algorithms that have minimum communication cost. Surprisingly, this minimum execution time is obtained by TREE-COMMIT for arbitrary subtransaction completion times and arbitrary network delays, even though these quantities are not known in advance, and are not used by the algorithm.

In our model we assume that each participant knows the identity of all the other participants, without sending or receiving additional messages. This holds for many transactions, such as one that simply updates a data item (e.g. employee record) that is replicated at several database sites. Additionally, it holds for every transaction in a fully replicated database. Moreover, we shall argue at the end of Section 4 that, in general, with limited knowledge of participants identity, minimum communication cost cannot be achieved.

Transaction commitment is a variation of a fundamental problem in distributed systems, namely distributed consensus. Fischer presents a survey of the subject ([F]), and Dwork and Skeen present an interesting taxonomy of consensus problems ([DS2]). Hadzilacos discusses the applicability of consensus results to the transaction
commitment problem ([H1]), particularly the types of failures that are meaningful. In short, almost all existing research concentrates on the effects of failures on achieving consensus in general, and sometimes transaction commitment in particular.

In contrast, in this paper we concentrate on performance issues. Mohan et. al. also discussed performance issues of commitment protocols ([MLO]), however our work differs from theirs in two respects. First, their commitment algorithms are more complicated mainly because of a more complicated transaction model, allowing for the fact that there may not be any participant that knows the identity of all the other participants in the transaction. Second, [MLO] as the other previous works, has disregarded the network topology.

Informal discussions of the performance of transaction commitment protocols in the absence of failures, also appear in [BHG], [CP], [G], [Sk], mainly in the context of different two-phase-commit schemes. One of the most popular is the central site scheme (see Fig 1.2a); a participant designated as the "protocol coordinator" polls all the other participants. In response, each participant sends its vote to the coordinator. The coordinator makes the decision, and sends the 'commit' or 'abort' message to all the other sites. In the decentralized scheme each participant sends its vote to all the other participants. Based on the received messages each participant makes the 'commit' or 'abort' decision. This scheme does not minimize the number of intersite messages, or the communication cost. Finally, in the linear, or nested, scheme (Fig. 1.2b) all participants are sequentially ordered. Each participant sends its vote to the next one in the sequence. The last participant is the protocol coordinator, which reverses the flow direction, by sending the decision message to its predecessor in the sequence. Gray notes that the linear scheme is the most efficient possible in terms of the number of intersite messages it requires ([G]). This number is \(2(n-1)\), where \(n\) is the number of participating sites. Dwork and Skeen have formally proven that the number of intersite messages required by any transaction commitment protocol, in the absence of failures, is \(2(n-1)\) ([DS1]). By contrast, note again that in the present work we consider network messages, rather than intersite messages. In the special case in which all the network sites are also participants, our lower bound result for network messages matches the \(2(n-1)\) result for intersite messages.

The rest of the paper is organized as follows. In Section 2 we introduce our model of a commit instance. In Section 3 we establish the necessary communication cost for the transaction commitment problem, and in Section 4 we provide a complete characterization of the minimum communication-cost commit-instances. In Section 5 the TREE-COMMIT algorithm is presented. In Section 6 we define the notion of execution time, and prove time
properties of TREE-COMMIT. In Section 7 a measure of comparison between minimum communication cost algorithms is proposed, and a classification of commitment algorithms, in terms of performance, is provided. In Section 8 we conclude, and discuss future work.

2. COMMIT INSTANCES

In this section we provide some key definitions, particularly, we formally introduce our novel model of a commit instance. Let \( V \) be a set of processors, and \( L \) a set of unordered pairs of \( V \), representing two-way communication links between processors. The graph \( G = (V, L) \) is a communication network, or a network for short. We assume that \( G \) is connected, and that a transaction in a distributed database executes at a subset of participant sites \( P \subseteq V \). When the transaction completes, the database management system executes a commitment protocol at the sites of \( P \), to decide whether to commit or to abort the transaction. The discussion is restricted to the case where no failures occur while the commitment protocol is executed. We will mainly analyze the case in which each site votes to commit the transaction (and thus the transaction commits), for the following reason. In Section 5 we discuss the 'abort' case of the algorithm TREE-COMMIT, and show that it is less expensive (from the communication cost viewpoint), than the 'commit' case. Therefore, successful commitment represents a worst-case scenario, again from the communication cost viewpoint; it also represents the more likely case for transactions in most database systems.

Intuitively, the instance of a commitment protocol is represented by the temporal, and thus partial, order of events occurring at the participants. Formally, an instance is a directed acyclic graph, \( I = (E, A) \) (see Fig. 2.1a). \( E \) is a set of nodes, called events, and \( A \) is a set of arcs (i.e. directed edges). Every event occurs at some participant, and all events occurring at a participant are totally ordered in \( I \). Informally, each event represents zero or more consecutive receives of an intersite message at the site (without an intervening send), followed by zero or more consecutive sends of intersite messages (without an intervening receive). The first event occurring at a participant also represents the completion of the corresponding subtransaction. Every pair of consecutive events occurring at a participant are connected by an arc called an order arc (since it represents the order in which the two events occur at the participant). The other arcs of \( A \) are called messages. A message is an arc from an event called the send of the message, to an event called the receive of the message, which occurs at a different site from the send. Only the last event occurring at a site may send zero messages, and only the first event occurring at a site may receive zero messages. Thus, into every event, except possibly the first one occurring at each site, enters at least one message, and
from every event, except possibly the last one occurring at each site, exits at least one message. If there is a path in \( I \) between events \( a \) and \( b \), then we say that \( a \) happens before \( b \) (in the sense of Lamport [L]), and \( b \) happens after \( a \). We assume that a message sent at an event \( a \) contains all votes that happened before \( a \). An instance at three participants is illustrated in Fig. 2.1a.

Next, we comment on the representation of several message-receives and -sends by one event. For the purpose of this paper, the order of consecutive message-sends is irrelevant, as is the order of consecutive message-receives. For example, we do not distinguish between two "instances" in which the only difference is that at some participant the order of two consecutive receives is reversed. Only the relative order of blocks of receives and sends is relevant, since we assume that each message sent includes all, and only, the votes received before sending of the message, and the vote of the sender.

We assume that a participant may send messages only after its subtransaction completes. Consequently, the first event at each participant represents zero or more message-receives, followed by the corresponding subtransaction completion, followed by zero or more receives, followed by one or more message-sends.

A commit instance is an instance \( I \) that satisfies the following commit requirement: At each participant occurs at least one event, \( e \), which happens after the first event occurring at every other participant. Any event such as \( e \), that happens after some, and particularly the first, event at each other site, is called a \( C \)-event; when it occurs, its participant, say \( j \), already knows the commit decision. The reason for this is that \( j \) has received all the 'yes' votes from the other sites, and it knows that its own vote is 'yes'. The vote of some participant, say \( i \), propagates to \( j \) along the path from the first event at \( i \), to the \( C \)-event of \( j \). A message sent by a \( C \)-event is called a commit message. Each message that is not a commit message is a vote message. An event which is not a \( C \)-event is called a \( V \)-event.

The motivation for the commit requirement is that each participant must receive the vote of every other participant before knowing that it can commit ([DS1, H2]). The instances illustrated in Fig. 2.1 are commit instances. In our figures the \( V \)-events (\( C \)-events) are denoted by a subscripted \( V \) (\( C \)). The label \( V_{i,j} \) (\( C_{i,j} \)) of a node indicates that this is the \( j \)'th \( V \)-event (\( C \)-event) occurring at participant \( i \).

Notice that the commit requirement ensures that every commit instance is connected. Notice also that any event which happens after a \( C \)-event, must also be a \( C \)-event. Another observation is that a \( C \)-event may represent the receive of a vote message (for example \( C_{1,1} \) in Fig 2.1a).
The variations of the two-phase commit protocol mentioned in the introduction are illustrated in terms of our model in Fig. 2.1. In order to prevent cluttering the figures we omit the order arcs; however, remember that any two consecutive events at a site, are connected by an order arc. The central site scheme, with participant 1 as the coordinator, is illustrated in Fig. 2.1a. The decentralized scheme is illustrated in Fig. 2.1b, and the linear scheme in Fig. 2.1c.

3. MINIMUM COMMUNICATION COST OF COMMIT INSTANCES

In this section we establish the minimum communication cost of a commit instance. This would have been quite straightforward, had we known that in a minimum communication cost commit instance all votes must be sent to one participant, the coordinator, which then broadcasts the commit decision. This would have meant that the problem of atomic commitment at minimum communication cost can be decomposed into two problems, collect-at-minimum-cost and broadcast-at-minimum-cost, and these two problems can be solved independently. However, as we shall show in Section 4, not every minimum communication cost commit instance consists of a collect-to-one, followed by a broadcast. In this section we show that there always exists such an instance with minimum cost. We do not know this yet, nor do we know that in a minimum communication cost commit instance, each participant sends at most one vote message. It is entirely possible that by sending two (or more) vote messages, a processor can reduce the overall communication cost, by reducing the cost of commit messages. In other words, it is possible that

\[ V_{1,1} \rightarrow V_{2,1} \rightarrow V_{3,1} \rightarrow C_{1,1} \rightarrow C_{2,1} \rightarrow C_{3,1} \]

Figure 2.1:
a minimum communication cost instance uses more than \(2(n-1)\) intersite messages (the necessary and sufficient number of intersite messages).

We start with some formal definitions. The \textit{communication cost} of a message arc in an instance is taken to be the length of the shortest path between \(i\) and \(j\) in the communication network, and the communication cost of an order arc is taken to be zero. Note that any message-arc has a strictly positive cost. The communication cost of an instance \(I\), denoted \(\text{Cost}(I)\), is the total communication cost of all the arcs in \(I\). Denote by \(\Psi\) the set of commit instances for some set of participants, \(P\), in a communication network, \(G\). The set \(P\) determines \(\Psi\), however, the cost of each instance in \(\Psi\) depends on \(G\). In this section we establish the minimum communication cost of a commit instance in \(\Psi\). A commit instance of such cost will be referred to as a \textit{minimum communication cost} instance.

Given a commit instance \(I\), its \(C\)-subgraph, denoted \(C_I\), is defined as the subgraph of \(I\) induced by its \(C\)-events.

\textbf{Lemma 1:} If \(I\) is a minimum communication cost instance of \(\Psi\), then \(I\) has only one \(C\)-event at every site. Furthermore, its \(C\)-subgraph is a forest of rooted trees.

\textbf{Proof:} Assume that there are two or more \(C\)-events at some participant. Then, there must be at least one incoming message into the second \(C\)-event, by definition of a commit instance (in particular, that only the first event at a site may represent zero receives). By omitting this message, a commit instance of strictly lower communication cost can be obtained\(^1\). The reason that the message can be omitted, is that the commit requirement is satisfied for the participant, by the first \(C\)-event occurring at the site. Consequently, there is only one \(C\)-event at every site.

In order to show that \(C_I\) is a forest, it is now sufficient to show that there is no \(C\)-event having two incoming commit messages. But this fact is obvious; if there were such an event, then one of the incoming commit messages can be omitted, to obtain an instance of lower communication cost. \(\Box\)

The single \(C\)-event at every participant, \(i\), will be denoted by \(C_i\). Assume that \(C_j\) is a root of \(C_I\), for some minimum communication cost instance, \(I\). Informally, this means that, in the execution \(I\), site \(j\) knows all the votes, without receiving a commit message (that is, \(j\) knows all the votes without receiving a message from another site that knew all the votes). In such a case, we say that site \(j\) is a \textit{coordinator} of the instance \(I\).

\textbf{Lemma 2:} There exists a commit instance of minimum communication cost in \(\Psi\), which has exactly one coordinator.
Proof: Let \( I \) be a minimum communication cost instance. We shall show that if \( I \) has two or more coordinators, then we can transform it into another commit instance, \( I' \), of equal communication cost, but with one less coordinator. The transformation replaces a vote message from some site, \( i \), to some site, \( j \), by a commit message from \( j \) to \( i \). Since the costs of the two messages are equal, the costs of \( I \) and \( I' \) are equal. The effect of the transformation is to change the way in which one of the coordinators learns the votes of the other participants; that coordinator becomes a noncoordinator, and learns the votes of the others via a chain of message from one of the remaining coordinators.

We now describe the selection of the vote message which will be replaced. The C-events at the coordinators of \( I \) will be called boundary C-events. Consider a V-event, \( V^o \), which satisfies the following condition. It precedes at least two boundary C-events, say \( C_i \) and \( C_j \), and, if \( V^o \) has any V-event successors, then each one of them precedes only one boundary C-event (see Fig. 3.1). It is easy to see that every commit instance with two or more coordinators has a V-event that satisfies the condition. Simply start at the first event at some site, which by the commit requirement precedes every boundary C-event. Verify whether that event satisfies the above condition. If not, it means that one of its V-event successors precedes two or more boundary C-events. Repeat the verification at that successor, until a V-event with the following property is found: either it has no V-event successors, or, each one of its V-event successors precedes only one boundary C-event.

Suppose that \( V^o \) occurs at participant \( p_o \). By Lemma 1 we know the structure of the C-subgraph of \( I \), particularly that there is only one C-event at \( p_o, C_{p_o} \). Assume, without loss of generality, that \( C_{p_o} \) is not in the tree of \( C_i \) rooted at the event \( C_i \). Denote a path from \( V^o \) to \( C_i \), by \( V^o = W^o, W^1, \ldots, W^n, W^{n+1} = C_i \), where \( n \geq 0 \). Note that all events on the path, except the last one, are V-events. Let \( W^r \) be the last event on this path for which the following condition is true: \( W^r \) occurs at a participant, \( p_r \), for which \( C_{p_r} \) is in a tree different than the one rooted at \( C_i \). Since \( W^r \) satisfies the condition, there must be such \( W^r \). Denote by \( p_{r+1} \) the participant at which \( W^{r+1} \) occurs. By the definition of \( W^r \), the event \( C_{p_{r+1}} \) is in the tree rooted at \( C_i \). Now to obtain a commit instance of minimum communication cost with one less coordinator, perform the following transformation:

i) the arc \( W^r \rightarrow W^{r+1} \) is replaced by a message from \( C_{p_r} \) to \( C_{p_{r+1}} \).

ii) the direction of the arcs on the path from \( C_i \) to \( C_{p_{r+1}} \) is reversed, and 

 sage is the only one entering its receive event.
iii) if transformations i) and ii) result in any event that consists of message-receives only, then that event and the one following it at the same site are collapsed into one event.

Figures 3.1 and 3.2 provide two examples of the transformation. Fig. 3.1 illustrates the modifications performed on $I$, assuming that $W^0$ does have some V-event successors. Figures 3.2a and 3.2b illustrate an instance, $I$, before and after the transformation, respectively; $W^0$ is $V_2$, and it has no V-event successors.

It is easy to see that the transformation results in a commit instance; denote it $I'$. The commit requirement is satisfied by $I'$, because, by definition of $W^o$, the removal of the arc $W^r \rightarrow W^{r+1}$ 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 $C_{m}$, which is in a different C-subgraph tree. Therefore, $I'$ is a commit instance, and has the same coordinators as $I$, except participant $i$, that is a coordinator in $I$, but is not a coordinator in $I'$. Moreover, the transformation performed on $I$ does not alter the communication cost, hence $I'$ has minimum communication cost.
To summarize, starting with a minimum communication cost instance, $I$, with two or more coordinators, we obtained a minimum communication cost instance, $I'$, with one less coordinator. We can continue this procedure until a minimum communication cost commit instance with exactly one coordinator is obtained. □

Next, we will obtain an additional lemma; it will be used in section 4. In a minimum communication cost instance, $I$, having two or more coordinators, let $V^*, p_\mu, r$ and $C_i$ be as in the proof of Lemma 2. Namely, $V^*$ is a $V$-event that precedes two or more boundary $C$-events, but each one of its $V$-event successors, if any, precedes only one boundary $C$-event. Such a $V$-event will be called a boundary $V$-event. We have shown in the proof of Lemma 2 that $I$ must have at least one boundary $V$-event. Assume, as in the proof of Lemma 2, that $C_i$ is a boundary $C$-event that is a successor of $V^*$, is not $C_\mu$, and does not precede $C_\mu$. Such a boundary $C$-event will be called associated with boundary $V$-event $V^*$.

The fact that $I$ is a minimum communication cost instance, implies that the path in $I$, denoted $p$, from $V^*$ to $C_i$, is unique, and consists of a single message arc, $V^* \rightarrow C_i$. The reason for this is as follows. If this is not the case, then there are more messages on $p$, or there are additional paths from $V^*$ to $C_i$. As established, one of the messages on $p$ is from $p_r$ to $p_{r+1}$. Then, the transformation in the proof of Lemma 2 can be augmented by the elimination of all messages on $p$, and on the other paths from $V^*$ to $C_i$. The resulting graph is a commit instance that has a cost strictly lower than $I$. This contradicts the fact that $I$ has a minimum communication cost. Therefore, $V^* = W^r$, $W^{r+1} = C_i, p_o = p_r, i = p_{r+1}$, and the transformation in the proof of Lemma 2 simply replaces a message $V_{p_{t-1}} \rightarrow C_i$ by a message $C_{p_r} \rightarrow C_i$. We have therefore proved that:
Lemma 3: In any minimum communication cost instance of $\Psi$, with two or more coordinators, there is at least one boundary $V$-event. Additionally, there is a unique path from any boundary $V$-event, $V^o$, to any associated $C$-event, $C_i$, and it consists only of the message $V^o \rightarrow C_i$. □

Now, let us define the distance graph $D$ as the complete graph having the set of participants $P$ as its nodes. The weight of each edge $(i,j)$ equals the number of links on the shortest path between $i$ and $j$, in the communication network $G$. The cost of a minimum spanning tree in $D$ is denoted by $\text{CMST}(G,P)$.

Given $T_1$ and $T_2$, two not necessarily different, minimum spanning trees of $D$, we define an instance on $T_1$ and $T_2$ coordinated at some participant $k \in P$. It is denoted by $I(k,T_1,T_2)$, and specified as follows. Denote by $T'_1$, the directed graph obtained from $T_1$ by directing its edges such that from every node there is a path to $k$ (the graph obtained is called an oriented tree with sink $k$). Also, denote by $T'_2$, the directed graph obtained from $T_2$ directing its edges to form a rooted tree, with root $k$.

One obtains the instance $I(k,T_1,T_2)$ as a result of the following modifications:

1. relabel the nodes from $T'_1$; node $i$ is relabeled $V_i$.
2. relabel the nodes from $T'_2$; node $i$ is relabeled $C_i$.
3. node $V_k$ is omitted, and the arcs entering it are modified to enter $C_k$ instead.
4. add the order arcs $V_i \rightarrow C_i$ for each $i$, except the coordinator.

Intuitively, $I(k,T_1,T_2)$ is a commit instance that has a set of vote messages which correspond to the arcs of $T'_1$, and commit messages which correspond to the arcs of $T'_2$. The definition of $I(k,T_1,T_2)$ is illustrated in Fig. 3.3. Given minimum spanning trees $T_1$ (a) and $T_2$ (b), the instance $I(4,T_1,T_2)$ is illustrated in (c). Clearly, the communication cost of $I(k,T_1,T_2)$ is $2 \cdot \text{CMST}(G,P)$.

Theorem 1: Let $G = (V,L)$ be a communication network, and let $P \subseteq V$ be a set of participants. Then

$$\min_{I \in \Psi} \text{Cost}(I) = 2 \cdot \text{CMST}(G,P).$$

Proof: Since the cost of an instance $I(k,T_1,T_1)$ for some minimum spanning tree $T_1$ is $2 \cdot \text{CMST}(G,P)$, then clearly $2 \cdot \text{CMST}(G,P) \geq \min_{I \in \Psi} \text{Cost}(I)$. To finish the proof observe that from Lemma 2, there exists a minimum communication cost instance, $I^*$, with one coordinator, say $k$. Based on $I^*$ construct an undirected graph, $H$, defined as follows. The nodes of $H$ are the participants in $P$, and edges of $H$ are $(i,j)$ there is a vote message from site $i$ to site
Since in $I^*$ there is a path from the first event at every site to $C_k$, $H$ must be connected. Therefore its cost is at least $CMST(G,P)$, which in turn implies that the communication cost of vote messages in $I^*$ is at least $CMST(G,P)$. Similarly we can show that the cost of the commit messages of $I^*$ is at least $CMST(G,P)$. Thus, $2 \cdot CMST(G,P) \leq \min_{I \in \mathcal{I}} \text{Cost}(I)$. $\square$

The result of Dwork and Skeen ([DS1, Theorem 1]) obtained for synchronous networks is extended to asynchronous networks by the following corollary of Theorem 1.

Corollary 1: Assume that the number of participants in a transaction is $n$. If the distance in the network between each pair of participants is one, then it holds that $\min_{I \in \mathcal{I}} \text{Cost}(I) = 2(n-1)$. $\square$

A similar extension of the result was obtained by Hadzilacos ([H2, Theorem 6]).

4. CHARACTERIZATION OF COMMIT INSTANCES WITH MINIMUM COMMUNICATION COST

In this section we give a complete characterization of all possible commit instances of minimum communication cost. We determine that if a commit instance has a minimum communication cost, then each site sends at most one vote message, and receives at most one commit message (a coordinator does not receive a commit message).

Also, in a minimum communication cost instance there are either one or two coordinators. If there are two coordinators, then each site sends exactly one vote message. If there is only one coordinator, then each site except the coordinator sends one vote message and receives one commit message; the coordinator does not send a vote message.
The messages of a minimum communication cost instance propagate "along edges" of minimum spanning trees of the distance graph. Specifically, there are two (not necessarily different) minimum spanning trees of the distance graph, such that the vote messages are only sent from a participant to its neighbor in one tree, and commit messages are only sent from a participant to its neighbors in the other. Moreover, if the instance has two coordinators, then the edge between the coordinators must exist in both trees.

A minimum communication cost instance on minimum spanning trees $T_1$ and $T_2$, coordinated at a site $k$, was defined in Section 3, and denoted $I(k, T_1, T_2)$. Similarly, we define next a minimum communication cost instance coordinated at two participants. Assume that $T_1$ and $T_2$ are two minimum spanning trees of the distance graph, such that participants $m$ and $n$ are neighbors in both trees. A minimum communication cost instance on $T_1$ and $T_2$ coordinated at $m$ and $n$, denoted $I(m, n, T_1, T_2)$, is defined as follows. Denote by $T_1'$ the graph obtained from $T_1$ by directing its edges to obtain an oriented tree with sink $m$, then omitting from it the arc $n \to m$. Denote by $T_2'$ the the graph obtained from $T_2$ by directing its edges to obtain a rooted tree with $m$ as a root, then omitting from it the arc $m \to n$.

One obtains the instance $I(m, n, T_1, T_2)$ as a result of the following modifications:

1. relabel the nodes from $T_1'$; node $i$ is relabeled $V_i$,
2. relabel the nodes from $T_2'$; node $i$ is relabeled $C_i$,
3. add the message arcs $V_m \to C_n$ and $V_n \to C_m$, and the order arcs $V_i \to C_i$ for each $i$.

Intuitively, $I(m, n, T_1, T_2)$ is a commit instance having vote messages that correspond to the arcs of $T_1'$, plus the arcs $m \to n$ and $n \to m$, and commit messages that correspond to the arcs of $T_2'$. In other words, in $I(m, n, T_1, T_2)$ each participant, including the coordinators, sends exactly one vote message. The vote of $n$ is received by $m$, and vice versa. Each participant, except the coordinators, receives exactly one commit message. Fig.4.1 demonstrates the definition. It illustrates a minimum communication cost instance on $T_1$ and $T_2$ of Fig. 3.3, coordinated at sites 2 and 4. Note that $I(m, n, T_1, T_2)$ is the same graph as $I(n, m, T_1, T_2)$.

Theorem 2 below shows that any minimum communication cost instance must have one or two coordinators, and be based on minimum spanning trees in the distance graph.

Theorem 2: Let $G(V, \mathcal{L})$ be a communication network, and let $P \subset V$ be a set of participants. Any minimum communication cost instance $I \in \Psi$, must be of the form $I(k, T_1, T_2)$ or $I(m, n, T_1, T_2)$, for some participants
Figure 4.1:

$k, m, n$, and minimum spanning trees $T_1$ and $T_2$ of the distance graph.

In order to prove Theorem 2 we shall show that: a) any one-coordinator, minimum communication cost instance, must be of the form $I(k, T_1, T_2)$; b) any two-coordinator, minimum communication cost instance, must be of the form $I(i, j, T_1, T_2)$; and c) a commit instance with three or more coordinators cannot have minimum communication cost. Fix $G$ and $P$ for the rest of this section.

Lemma 4: A one-coordinator minimum communication cost instance must be of the form $I(k, T_1, T_2)$ for some minimum spanning trees $T_1$ and $T_2$.

Proof: Consider a minimum communication cost instance, $I$, with one coordinator, $k$. Following a line of reasoning similar to one used in the proof of Theorem 1, it can be easily seen that the total cost of vote messages is exactly $CMST_G(P)$, and the total cost of commit messages is exactly $CMST_G(P)$. Consider the undirected graph, $H$, having as nodes the participants, and edges $\{(i, j)\mid$ there is a vote message from site $i$ to site $j$ in $I \}$. $H$ must be connected, its cost is $CMST_G(P)$, therefore it is a minimum spanning tree of the distance graph, $T_1$. Clearly, the vote messages of $I$ correspond to the oriented tree with sink $k$, obtained by directing the edges of $T_1$. Similarly we can show that the commit messages of $I$ correspond to the rooted tree obtained by directing the edges of a minimum spanning tree, $T_2$. □

In Lemma 2 we presented a procedure to modify any minimum communication cost instance, $I$, with two or more coordinators, into another commit instance $I'$ with one less coordinator, such that the communication costs of $I$ and $I'$ are identical. A similar procedure will be used here to characterize minimum communication cost instances.
Lemma 5: A two-coordinator minimum communication cost instance $I$ must be of the form $I(i,j,T_1,T_2)$, for some participants $i$ and $j$, and some minimum spanning trees, $T_1$ and $T_2$, both having the edge $(i,j)$.

Proof: Let $C_i$ and $C_j$ be the boundary $C$-events in $I$. By replacing one vote message by a commit message, as in Lemma 2, we obtain a one-coordinator instance of minimum communication cost. We then use the characteristics of such an instance given in Lemma 4, to show that $I$ is indeed of the form $I(i,j,T_1,T_2)$.

Let $V^o$ denote a boundary $V$-event, as defined after the proof of Lemma 2. Since $i$ and $j$ are the only coordinators, $V^o$ precedes both $C_i$ and $C_j$. Let $C_i$ be the boundary $C$-event associated with $V^o$, and $p_o$ the participant at which $V^o$ occurs. Since there are only two boundary $C$-events, by definition of an "associated boundary $C$-event", $C_{p_o}$ is in the tree of the C-subgraph which is rooted at $C_j$.

We first show that, in fact, $p_o$ is the coordinator $j$. By Lemma 3 observe that the path from $V^o$ to $C_i$ is the message arc $V^o \rightarrow C_i$. Consider the commit instance $I'$ obtained from $I$ by replacing $V^o \rightarrow C_i$ by $C_{p_o} \rightarrow C_i$ (as in the proof of Lemma 2). $I'$ has only one coordinator, $j$, and has the same cost as $I$, i.e. minimum communication cost. Therefore, by Lemma 4, the instance $I'$ must be of the form $I(j,T_1,T_2)$ for some minimum spanning trees $T_1,T_2$. In particular, in the instance $I'$, the coordinator $j$ does not send a vote message, whereas in $I$, site $j$ must send a vote message, because there must be a path from the first event at $j$ to $C_i$. However, the only vote message deleted in $I$ to obtain $I'$ is $V^o \rightarrow C_i$; hence $V^o$ occurs at site $j$, or, in other words, $p_o=j$.

Because of the new arc, $C_{p_o} \rightarrow C_i$, $i$ and $j$ are neighbors in $T_2$. We shall show that $i$ is a neighbor of $j$ in $T_1$ as well, or, in other words, that $i$ sends its vote to $j$ in $I$. Denote the single $V$-event at each participant $r$ in $I$ by $V_r$. We have already shown that if $V^o$ has $C_i$ as an associated $C$-event, then $V^o$ is actually $V_j$. If there is no other boundary $V$-event in $I$, then all $V$-events must precede $V_j$, and therefore $V_j$ should be a $C$- rather than a $V$-event. Consequently $V_j$ cannot be the only boundary $V$-event in $I$. Denote by $V^1$ another boundary $V$-event, i.e. $V^1 \neq V^o$. The associated $C$-event of $V^1$ cannot be $C_i$, since otherwise as above, we can show that $V^1 = V_j$, which in turn implies that $V^1 = V^o$. Thus, the associated $C$-event of $V^1$ must be $C_j$, and, as above, $V^1$ occurs at participant $i$; that is, $V^1$ is actually $V_i$, so participant $i$ sends its vote to participant $j$.

Now, recall that $I'$ is obtained from $I$ simply by replacing $V_o \rightarrow C_i$ by $C_j \rightarrow C_i$, and that $I'$ has the form $I(j,T_1,T_2)$. Therefore, in the instance $I$, exactly one vote message is sent by each participant, and $I$ is of the form $I(i,j,T_1,T_2)$. Therefore, by Lemma 4, $I$ is indeed of the form $I(i,j,T_1,T_2)$.
Lemma 6: There are no minimum communication cost instances with three or more coordinators.

Proof: From the proof of Lemma 2, we know that for every minimum communication cost instance with two or more coordinators, there exists a minimum communication cost instance with one less coordinator. Consequently, it is sufficient to show that there is no minimum communication cost instance with three coordinators.

Suppose that there exists a minimum communication cost instance, \( I \), with three coordinators \( i, j, m \). Let \( V^o \) be a boundary V-event, and \( C_i \) an associated boundary C-event. Assume that \( V^o \) occurs at participant \( p_o \). By replacing \( V^o \rightarrow C_i \) with \( C^o_{p_o} \rightarrow C_i \), we obtain as in Lemma 2 an instance \( I' \) of minimum communication cost with two coordinators \( j \) and \( m \). By Lemma 5, instance \( I' \) must have the form \( I(j,m,T_1,T_2) \) for some minimum spanning trees \( T_1 \) and \( T_2 \). Event \( V^o \) occurs at exactly one participant, therefore it cannot occur at both coordinators. Assume without loss of generality that it does not occur at coordinator \( j \), i.e. \( p_o \neq j \). Then the vote messages exiting V-events occurring at participant \( j \) are the same in \( I(j,m,T_1,T_2) \) and \( I \). However, by definition, the only such vote message in \( I(j,m,T_1,T_2) \) is \( V_j \rightarrow C_m \). Therefore, in \( I \), the only V-event occurring at site \( j \) does not precede boundary C-event \( C_i \). Thus \( I \) does not satisfy the commit requirement, contradicting the fact that \( I \) is a commit instance.

We have completed the proof of Theorem 2; as an immediate corollary we conclude:

Corollary 2: If the communication cost of some commit instance is minimum, then it has a minimum number of intersite messages, i.e., \( 2(n-1) \).

Corollary 2 holds because the instance is composed of two minimum spanning trees, each of communication cost \( n-1 \).

Obviously, as demonstrated in the introduction, the converse of corollary 2 is not true, i.e. minimum number of intersite messages does not imply minimum communication cost.

The characterization in Theorem 2 helps us demonstrate that in general, minimum communication cost cannot be achieved with limited knowledge of participants' identity. For example, suppose that in Fig. 1.1, participants 2,3,4 know only about 1, and 1 knows only of participant 4. Then a message must be sent between 1 and 4, and since the edge 4-1 in the distance graph does not belong to any minimum spanning tree, such a scenario cannot achieve minimum communication cost.
5. THE PROPOSED TRANSACTION COMMITMENT ALGORITHM

In discussing commitment algorithms we assume, as in other works (e.g. [DS2]), that each participant knows the identity of all the participants, and in addition we assume that it knows the communication network, G. The analysis of section 3 suggests a very simple minimum communication cost commitment algorithm, which we will call fixed-coordinator. It proceeds as follows. After completing its subtransaction each participant constructs some minimum spanning tree, T, of the distance graph, and selects a participant, k, designated as the coordinator. T and k are assumed to be identical at all participants. This will be the case if the procedure which constructs T and selects k, is identical at all sites. The algorithm fixed-coordinator, executes an instance with the vote messages corresponding to T (which is the oriented tree with sink k, obtained by directing the edges of T towards k); the commit messages correspond to T' (which is the rooted tree obtained from T by directing its edges away from k). Specifically, in an execution in which all the participants commit, a participant i ≠ k will wait until subtransaction completion and receipt of all vote messages represented by the arcs incoming into i in T', and then it will send a 'yes' vote message corresponding to the arc exiting i. Participant k completes its subtransaction and receives votes from all neighbors in T, and then commits. The propagation of the commit decision is similar, in the opposite direction.

Although the fixed-coordinator algorithm achieves minimum communication cost, its execution time is worse than the execution time of the dynamic coordinator algorithm, which we describe next. Furthermore, the dynamic coordinator algorithm also achieves minimum communication cost, and is as simple as as the fixed coordinator algorithm. The dynamic coordinator algorithm will be referred to as TREE-COMMIT. In the next section we prove that the execution time of TREE-COMMIT is better than the execution time of fixed-coordinator.

TREE-COMMIT is a distributed, minimum communication cost algorithm, adapted from the PIF (Propagation of Information with Feedback) algorithm of [Se]. Each participant constructs the same minimum spanning tree, T, of the distance graph. In contrast to the fixed-coordinator algorithm, in TREE-COMMIT, a coordinator is not selected when the tree is constructed. The procedure performed by each site i is as follows. After subtransaction completion, it waits until receiving the votes from all its neighbors in T except one, say j, before voting; then it sends its vote to j. If i receives votes from all neighbors in T before it completes its subtransaction, then i commits and becomes the single coordinator. If i receives a vote message from j after having sent its vote to j, then it commits becoming one of two coordinators (j is the other one). If i receives a commit message from j, then it sends
commit messages to all its neighbors in $T$, except $j$. Therefore, the votes travel from the leaves of $T$ inwards, where one or two coordinators are established. In the commit stage, the commit message is simply propagated along the tree edges, away from the coordinator(s). We will denote by TREE-COMMIT($T$) the algorithm which uses the tree $T$. The reader should convince herself/himself that TREE-COMMIT($T$) generates an instance of the form $I(k, T, T)$ or $I(m, n, T, T)$ for some coordinators $m, n, k$, therefore its communication cost is minimum. A possible situation in the voting stage, i.e. before the coordinators are determined, is illustrated in Fig. 5.1a. In the scenario illustrated, we suppose that participants 1, 2, and 5 have completed their subtransactions, and participants 3 and 4 have not. Participant 2 has not voted yet because it has not received the votes of two of its neighbors. Fig. 5.1b, 5.1c illustrate two possible instances executed by TREE-COMMIT, at the completion of the voting stage situation described above. In the first case, 3 completed its subtransaction, and its vote had reached 2 before the vote of 4 did so; furthermore, the votes of 2 and 4 crossed, so they both became coordinators. In the second case, 3 completed its subtransaction after having received the vote of all participants, so 3 became the sole coordinator.
Note that a dynamic determination of the coordinator(s) can speed up the traditional linear two-phase-commit, without increasing communication cost or number of messages, regardless of the network topology. In this modified scheme, the participants are ordered in a predetermined sequence. The first participant votes as soon as it completes its subtransaction, sending the vote to its only neighbor; the same for the last participant in the sequence. Each other participant votes as soon as it completes its subtransaction, and receives the vote of one of its neighbors in the sequence. At that point it sends its vote to the other neighbor. The resulting instance will have one or two coordinators, and the coordinator may be any node (or pair of neighbors), in the sequence of participants.

The transaction-abort case is handled by TREE-COMMIT as follows: Participant $i$ sends 'abort' messages when the first of the following two cases occurs: i) $i$ unsuccessfully completed its subtransaction, in which case $i$ sends an 'abort' (or a 'no' vote) message to each neighbor in $T$; ii) $i$ receives the first 'abort' message from a participant, say $k$, in which case $i$ sends an 'abort' message to each neighbor, except $k$.

The complete TREE-COMMIT algorithm, as executed by a participant, $i$, is given in Fig. 5.2. Note that TREE-COMMIT uses only 'yes', 'commit', and 'abort' messages, therefore the message length can be restricted to two bits.

We have not formally defined an 'abort' instance of an arbitrary commitment protocol. However, given the complete specification of TREE-COMMIT, we can establish the lower and upper bounds for the communication cost of an execution of TREE-COMMIT that results in the abort of the transaction.

Theorem 3: The communication cost of an abort instance executed by TREE-COMMIT is at least $CMST(G,P)$ and at most $2\cdot CMST(G,P)$.

Proof: Clearly, commit messages are sent only if all subtransactions completed successfully, therefore in an abort instance no commit messages are sent. Assume that the minimum spanning tree used in an abort instance is $T$. Each message in the instance is sent between two neighbors in $T$, therefore let us consider the edges of $T$. For each edge $(i,j)$ there is either exactly one abort message from $i$ to $j$, or exactly one 'yes' message from $i$ to $j$, but not both. Similarly from $j$ to $i$. Hence the total cost of messages is at most $2\cdot CMST(G,P)$. Additionally, note that for each edge $(i,j)$ of $T$ there is an 'abort' message from $i$ to $j$, or from $j$ to $i$. Hence the total communication cost of the instance is at least $CMST(G,P)$. □
TREE-COMMIT(G,P) /* procedure executed by participant I */

1. Construct a minimum tree, T, of the distance graph of P in G.

2. Wait until subtransaction completion, or receipt of an 'abort' message. /* vote messages are being received, but they do not wake-up this process; these messages are being considered in steps 5,6 */

3. if subtransaction completion then do;
   4. if successful completion then do;
   5. if received 'yes' votes from all neighbors in T then send 'commit' messages to all neighbors, and quit. /* I is a single coordinator */
   6. otherwise wait until receiving the first 'abort' message, or until receiving a 'yes' vote from all neighbors, except one.
   7. if 'abort' message from a neighbor, say k, then send 'abort' messages to all neighbors, except k, and quit.
   8. if 'yes' votes from all neighbors, except one, say j, then send a 'yes' vote to j.
   9. end.
  10. otherwise (unsuccessful completion) send 'abort' messages to all neighbors, and quit.
  11. end.
  12. otherwise ('abort' message from a neighbor, say k) send 'abort' messages to all neighbors, except k, and quit.
  13. Wait until receiving a message. /* the only way to get here is from step 8, and the message must have been received from j */
  14. if 'abort' message then send 'abort' messages to all neighbors, except j, and quit.
  15. otherwise ('commit' or 'yes'-vote) send 'commit' messages to all neighbors, except j, and quit. /* in the 'yes'-vote case, I and j are coordinators */

Figure 5.2

It is easy to see that the bounds of Theorem 3 are tight. If all the subtransactions complete unsuccessfully at exactly the same time, then two 'abort' messages are sent along each edge of T, one in each direction, and the communication cost is $2 \cdot CMST(G,P)$. If, on the other hand, some participant completes its subtransaction unsuccessfully, and after this, the 'abort' messages reach each participant before it completed its subtransaction, then one 'abort' message is sent along each edge of T, and the communication cost is $CMST(G,P)$. 
The algorithm TREE-COMMIT is based on the assumptions that the network topology is stable and each participant knows it, and that each participant also knows the identities of all the other participants. Thus each participant can independently construct the common tree. Knowledge of the identities of all participants is always available in a fully replicated database. In a partially replicated database the assumption may not always be practical. However, for many update transactions (e.g. add 10,000 dollars to an account which is replicated at four sites) there is no problem with it. Each participant usually stores a directory, indicating where each data-item is replicated.

6. EXECUTION TIME OF COMMIT INSTANCES

In this section, we first define the term execution-time of a commit instance, and then we show that for a minimum spanning tree, \( T \), TREE-COMMIT(\( T \)) dynamically and distributively selects among all minimum communication cost commit instances on \( T \), the one with minimum execution time. A minimum communication cost commit instance on \( T \) i.e., an instance of the form \( I(k,T,T) \) or \( I(m,n,T,T) \), will be called for short a \( T \)-instance.

Generally, time comparison of instances in a totally asynchronous network is impossible, because each message can have an arbitrarily long delay. Therefore some restrictions on the network behavior must be imposed. The only restriction we impose here is that the delay of a message between every pair of participants is fixed, at least for the duration of any commitment-algorithm execution. Thus, any message from \( i \) to \( j \), sent by any algorithm, takes a fixed and finite amount of time to arrive, say \( t_{ij} \).

The execution-time of an instance \( I \) is defined with respect to a set \( \tau = \{ \tau_i \mid i \in P \} \) of subtransaction completion times, and with respect to a set \( t = \{ t_{ij} \mid i,j \in P \} \) of intersite communication delays (or intersite delays for short). Each delay, \( t_{ij} \), is a positive real number, and each \( \tau_i \) is a nonnegative real number. The delay \( t_{ij} \) is the time from the sending of the message at \( i \), until it is received at \( j \). \( t_{ij} \) may be different than \( t_{jk} \). We assume that internal processing at each participant takes zero time, since in most networks processing is negligible compared to message propagation delay. This means that the sending of a message exiting an event, say \( V_i \), happens at the same time as the last receive of a message entering \( V_i \), provided that the subtransaction at participant \( i \) has completed. Otherwise, the sending of the messages happens at time \( \tau_i \), the subtransaction completion time. The beginning of the instance is at the smallest \( \tau_i \), assumed to be 0. The execution time of the instance \( I \), with respect to \( \tau \) and \( t \), is defined as the time when the last site commits. In other words, the execution time of an instance is the time the instance takes, assuming that a message between every pair of participants \( i \) and \( j \) takes \( t_{ij} \) time units, a participant
i does not send its first message before time $\tau_i$, and internal processing takes zero time.

Theorem 4: Given $G$, $P$, and $T$ a minimum spanning tree of the distance graph of $P$ in $G$, the execution of TREE-COMMIT($T$) generates a $T$-instance of minimum execution time among all $T$-instances, for any given set of sub-transaction completion times $\tau = \{\tau_1, \ldots, \tau_n\}$, and any given set of intersite delays $\tau = \{\tau_{ij} \mid i, j \in P\}$.

Proof: Assume first that the instance generated by TREE-COMMIT, denoted $I$, has one coordinator, $k$. Denote by $d_r$ the longest path in $T$ from a node $r$, when the length of each edge $(i, j)$ is $\tau_{ij}$. Observe that the execution time of $I$ is $\tau_k + d_k$. Let $I'$ be any other $T$-instance. In $I'$ there must be a message path going through the edges of $T$, from $V_k$ (or $C_k$ if $k$ is a single coordinator in $I'$) to every other $C_i$. Also, in $I'$, the message exiting $V_k$ (or $C_k$) cannot be sent before time $\tau_k$. Therefore, in this case, the execution time of $I$ is not higher than the execution time of $I'$.

Assume now that $I$ has two coordinators. We shall show that in $I$, every node sends its vote at the earliest possible time, among all $T$-instances. Let $I'$ be another $T$-instance, and let $v_i, v_i'$ be the time when participant $i$ sends its vote message in $I$ and $I'$, respectively. If $I'$ has a single coordinator, $k$, then $k$ does not send a vote message in $I'$, and we define $v_k'$ to be the time at which $k$ receives the last vote message. Observe that:

$$\begin{align*}
\tau_j &\leq v_j' \text{ if } j \text{ is not a single coordinator in } I' \\
\tau_j &\geq v_j' \text{ if } j \text{ is a single coordinator in } I'.
\end{align*}$$

We want to show that:

$$v_i \leq v_i' \text{ for every participant } i. \quad (1)$$

Suppose the contrary, and let $j$ be the participant such that:

$$v_j' = \min \{v_i' : v_i > v_j'\}$$

Since $I'$ is a $T$-instance, i.e. is of the form $I(k,T,T)$ or $I(k,m,T,T)$, there exists a set $K$ of neighbors of $j$, from which $j$ receives votes in $I'$, before, or at time $v_j'$. Namely,

$$v_r + t_{ij} \leq v_j' \quad \forall r \in K \quad (3)$$

If $j$ is a single coordinator in $I'$, then $K$ is the set of all neighbors of $j$ in $T$. Otherwise $K$ is the set of all neighbors in $T$, except one.

Since $t_{ij} > 0$, we obtain from (3) that:

$$v_r < v_j' \quad \forall r \in K \quad (4)$$

and by definition of $v_j'$, as being the smallest $v'$ such that $v_j > v_j'$. 

This implies that:

\[ v'_r \geq v_r \ \forall r \in K. \]  

(5)

and moreover,

\[ v_r \leq v'_r < v'_j < v_j \ \forall r \in K. \]  

(6)

From (6) we see that, in \( I \), each \( r \in K \) sends its vote before receiving a vote from \( j \), and consequently, by the specification of TREE-COMMIT, it must send its vote to \( j \). From (7) we deduce that, in \( I \), the vote from each \( r \) arrives at \( j \), before \( j \) sends its vote.

Now, we derive a contradiction in each one of two cases.

First, if \( j \) is a single coordinator in \( I' \), then \( K \) is the set of all its neighbors in \( T \). Thus (7) implies that, in \( I \), \( j \) receives the vote of each neighbor before \( v_j \), namely before voting. But then, \( j \) should be a single coordinator in \( I \) as well, contradicting the assumption that \( I \) has two coordinators.

Second, if \( j \) is not a single coordinator in \( I' \), then, from (7), by time \( v'_j \) it receives in \( I \) the votes of all neighbors in \( T \) except one, and from (1) it holds that \( v'_j \leq v'_j \). But this means that in TREE-COMMIT \( j \) sends its vote at time \( v'_j \), or earlier. Namely, \( v_j \leq v'_j \), contradicting the fact that \( v_j \geq v'_j \). This completes the proof of (2).

To complete the proof of the theorem, denote the two coordinators of \( I \) by \( m \) and \( n \). Then, the execution time of \( I \) is \( \max(v_m + d_m, v_n + d_n) \). Observe that the execution time of \( I' \) is at least \( \max(v'_m + d'_m, v'_n + d'_n) \), since the instance cannot complete before the votes of \( m \) and \( n \) reach every participant. Therefore, the execution time of \( I' \) is at least as long as the execution time of \( I \). □

7. A TAXONOMY OF COMMITMENT ALGORITHMS

In this section we examine the performance of TREE-COMMIT, compared to other atomic commitment algorithms. The primary comparison criterion is taken to be the communication cost, and therefore we restrict our attention to algorithms that achieve minimum communication cost, and compare their execution time. Based on theorems 2 and 4, it is straightforward to show that TREE-COMMIT(T) has optimal execution time among all minimum communication cost commitment algorithms in which communication is confined to take place between neighbors in \( T \). In order to extend the comparison to algorithms which are not confined to a specific tree we distinguish between two classes of algorithms: the ones which are oblivious of intersite delays, or are delay invariant, and
the ones which are not (thus the processors incur an extra overhead of monitoring and learning the intersite delays in the entire network). TREE-COMMIT belongs to the first class (note that it uses a minimum spanning tree based on network path-lengths, not intersite delays). We show that in the first class there is no algorithm which is superior to TREE-COMMIT(T), for any tree T. In the second class such algorithms exist, however, we show that even if an algorithm knew initially all subtransaction completion times, and message delays (which is unlikely), it would face an NP-complete problem in trying to minimize communication cost and time.

Next, a "dominates" relationship among commitment algorithms is defined. Intuitively, algorithm A dominates algorithm B if its communication cost is lower; if A and B have equal communication cost then A dominates B if for any set of intersite delays between the participants, and any set of subtransaction completion times, the execution time of algorithm A, is not worse than the execution time of B. Formally, a deterministic, distributed commitment algorithm, A, or an algorithm for short, is a function which maps each quadruple \((G,P,\tau,t)\) to an instance denoted \(A(G,P,\tau,t)\). \(G\) is the communication network, \(P\) is the set of participants, \(\tau=\{\tau_1,\ldots,\tau_n\}\) is a set of subtransaction completion times, and \(t=\{t_{ij} | i,j \in P\}\) is a set of intersite delays. We take \(G\) and \(P\) to be arbitrary, but fixed. Thus, to simplify notation, we denote \(A(G,P,\tau,t)\) by \(A(\tau,t)\). Also, since communication cost is our primary criterion, we will only consider minimum communication cost algorithms, i.e., algorithms which execute a minimum communication cost instance, for any pair of sets, \(\tau\) and \(t\). For a given minimum spanning tree \(T\), an algorithm, \(A\), in which messages are sent only between neighbors in \(T\), will be called a \(T\)-algorithm. Denote by \(X(I,\tau,t)\) the execution time of an instance \(I\), for \(\tau\) and \(t\). For an algorithm \(A\), instead of \(X(A(\tau,t)),\tau,t)\) we shall use the shorthand \(X(A(\tau,t))\) to denote the execution time of the instance \(A(\tau,t)\). An algorithm \(A\) is said to dominate algorithm \(A'\) if \(X(A(\tau,t)) \leq X(A'(\tau,t))\), for all \(\tau\) and \(t\).

Theorem 5: Let \(T\) be some minimum spanning tree of the Distance Graph. Then TREE-COMMIT(T) dominates any minimum communication cost \(T\)-algorithm.

Proof: Consider any minimum communication cost \(T\)-algorithm, \(A\). Since \(A(\tau,t)\) is a minimum communication cost \(T\)-instance for any \(\tau\) and \(t\), Theorem 2 implies that \(A(\tau,t)\) is of the form \(I(k,T,T)\) or \(I(m,n,T,T)\), for some participants \(k,m,n\). By Theorem 4, the execution time of TREE-COMMIT(T) \((\tau,t)\) is not bigger than of any instance of this form, with respect to \(\tau\) and \(t\). Thus, TREE-COMMIT(T) dominates \(A\).
A $T$-algorithm in which some participant, say $k$, is the coordinator for any $\tau$ and $\ell$, is called a fixed coordinator $T$-algorithm. If an algorithm, $A$, dominates another algorithm, $A'$, and in addition, there exist $\tau_0$ and $\ell_0$ for which $X(A(\tau_0,\ell_0)) < X(A'(\tau_0,\ell_0))$, then $A$ is said to strictly dominate $A'$.

Theorem 6: Let $T$ be some minimum spanning tree of the Distance Graph. TREE-COMMIT($T$) strictly dominates any minimum communication cost fixed coordinator $T$-algorithm.

Proof: Let $A$ be some fixed coordinator $T$-algorithm. The fact that TREE-COMMIT($T$) dominates $A$, is an immediate consequence of Theorem 5. In order to show strict dominance, we have to find for $A$ sets, $\tau$ and $\ell$, for which TREE-COMMIT($T$) $(\tau,\ell)$ has an execution time that is strictly less than $A$. In order to identify such sets, consider some participant, say $i$, that is not a coordinator in $A$. Denote by $R$ some very large integer. Let $\tau_i=2R$, and $\ell_j=R$ for any neighbor $j$ of $i$ in $T$. Denote the number of participants by $n$. All the other subtransaction completion times of $\tau$ are zero, and each other intersite delay of $\ell$ is $R/n$. With these $\tau$ and $\ell$, TREE-COMMIT($T$) will select $i$ as a single coordinator, its commit will occur at time $2R$ and the execution time of the instance will be less than $4R$. On the other hand the execution time of $A(\tau,\ell)$ will be at least $4R$. The reason is that, since participant $i$ is not a coordinator, the coordinator can send its first message no earlier than time $3R$, and the commit message cannot reach $i$ before time $4R$.

An algorithm, $A$, is delay invariant if for each pair of participants, $i$ and $j$, the following condition holds. Consider some $\tau$, and pair of intersite delays, $\tau$ and $\tau'$, in which all the delays are equal, except possibly for the delay from $i$ to $j$. If a message is sent from $i$ to $j$ in $A(\tau,\ell)$, then such a message is also sent in $A(\tau,\ell')$.

Intuitively, an algorithm is delay invariant if it behaves as follows: if for some $\tau$ and $\ell$ participant $i$ sends a message to participant $j$, then it will do the same if the intersite delay $\ell_j$ changes, provided that everything else stays the same. Algorithms which do not monitor intersite delays, obviously exhibit this type of behavior. In order for an algorithm not to be delay invariant, or be delay variant, participants must send messages to monitor delays, and behave differently for different delays. Obviously this puts additional load, and complicates the algorithm.

Theorem 7: Let $T$ be some minimum spanning tree of the Distance Graph. There is no delay-invariant algorithm which strictly dominates TREE-COMMIT($T$).

Proof: Assume that $A$ is a delay-invariant algorithm which strictly dominates TREE-COMMIT($T$). By Theorem 4,
Algorithm $A'$ cannot execute only $T$-instances. Thus, for some sets $\tau$ and $\iota$, $A(\tau,\iota)$ sends a message from $i$ to $j$, where $i$ and $j$ are not neighbors in $T$. Consider $\iota'$ in which the delays are the same as in $\iota$, except that $\ell_{ij}$ is a very large number (for example, the sum of the other delays in $\iota$ and the completion times). Since $A$ is delay-invariant, $A(\tau,\iota')$ sends a message from $i$ to $j$, and therefore $X(A(\tau,\iota')) > X(TREE-COMMIT(T)(\tau,\iota'))$. This contradicts the fact that $A$ strictly dominates $TREE-COMMIT(T)$.

Algorithms that are not delay-invariant can strictly dominate $TREE-COMMIT$. For example, consider the algorithm $E$(xtended)$TREE-COMMIT$, that we describe informally below. It behaves like $TREE-COMMIT(T)$ in the voting stage. If participant $i$ is a single coordinator, instead of propagating the commit message along the edges of $T$, it finds a new minimum spanning tree, $T'$, with the following property. Consider the set $S$ of minimum spanning trees of the distance graph. For each tree in this set assign the length of each edge $(k,j)$ to be the intersite delay, $t_{kj}$. The tree $T'$ is the member of $S$ that has the shortest (among all trees in $S$) longest path from $i$.

If the commit messages are propagated along the edges of $T'$ instead of $T$, then a shorter execution time is possible. It is intuitive, and it can be proven rigorously, that $ETREE-COMMIT$ strictly dominates $TREE-COMMIT$.

In conclusion, Fig. 7.1 illustrates the different classes of minimum communication cost commitment algorithms, for a fixed network, $G$, and set of participants, $P$. The class of delay-variant algorithms is disjoint from the class of delay invariant algorithms. For any minimum spanning tree, $T$, the class of $T$-algorithms intersects both classes above. $TREE-COMMIT(T)$ is a delay invariant $T$-algorithm, that dominates all $T$-algorithms. Fixed coordinator $T$-algorithms are delay invariant algorithms, that are strictly dominated by $TREE-COMMIT(T)$. There is no delay invariant algorithm that strictly dominates $TREE-COMMIT(T)$, but there is a delay variant algorithm, $ETREE-COMMIT$, that does so.

There are three disadvantages to $ETREE-COMMIT$, constituting the reason that it is not our proposed algorithm. First, the processors incur the extra overhead of monitoring intersite delays, and transmitting them to the other processors. Second, it is an NP-complete problem to find a minimum spanning tree with shortest-longest delay ([II]). Third, 'commit' messages must be longer, and include routing information, since a node that receives a commit message must be informed by the coordinator how to propagate it (the minimum spanning tree used by the participants in voting is no longer to be used). In fact, it can be shown that it is a hard problem for an algorithm to find a "best" communication-cost and time instance, even if all subtransaction completion times and intersite delays
are known a priori. Specifically, the following problem, MINIMAL COST AND TIME INSTANCE (MCTI), is proved in [SW] to be NP-complete. The input consists of a network graph $G=(V,E)$, a set of participants $P \subseteq V$, a set of subtransaction completion times, $\tau$, a set of intersite delays $t$, and an integer $K$. The question is whether there exists a minimum communication cost instance with execution time $\leq K$, with respect to $\tau$ and $t$.

8. DISCUSSION

8.1 Conclusion

In this paper we established the minimum communication cost of transaction atomic commitment, we characterized the minimum communication cost instances, we proposed the TREE-COMMIT algorithm, and we examined alternative commitment algorithms. The minimum communication cost of commitment is twice the weight of the minimum spanning tree of the distance graph between the participants in the network. TREE-COMMIT is a minimum communication cost, time efficient, algorithm, that adjusts automatically, i.e. without incurring any extra overhead, to different communication intersite delays and subtransaction completion times.

Can TREE-COMMIT be improved upon? In order to answer the question a taxonomy of commitment algorithms was presented. We were interested only in algorithms which always achieve minimum communication cost,
and compared their execution time. The comparison indicates the need to distinguish between two types of algorithms: the delay-variant, and the delay-invariant algorithms. Algorithms in the former class use advance information on the intersite delays in the network. Algorithms in the latter class do not. TREE-COMMIT is delay-invariant, and we showed that no delay-invariant algorithm is superior to TREE-COMMIT. Some delay-variant algorithms are superior to TREE-COMMIT in terms of execution time. However, such algorithms have several disadvantages, which we discussed qualitatively. They concern algorithm computational complexity, the need to monitor and disseminate intersite delay information, and the length of control messages. Because of these drawbacks, we feel that, in practice, it is hard to argue for an algorithm that uses advance information on intersite delays. Furthermore, we conjecture that the knowledge of intersite delays cannot improve significantly the execution time of TREE-COMMIT, without advance information on subtransaction completion times. This means that participants would have to know a priori when other subtransactions complete, in order to significantly improve execution time of atomic commitment algorithms.

Finally, let us mention that the results in this paper also apply to the following variant of the "gossip" problem (gossip is surveyed by Hadetniemi et. al. in [HHL]). There is a set $A$ of individuals, each of which knows a unique piece of information and must transmit it to every other person. In addition, there is a set, $C$, consisting of a cost, denoted $c_{ij}$, for each pair $i < j \in A$. $c_{ij}$ is the cost of a message from $i$ to $j$, and the cost of a message from $j$ to $i$ (i.e., these costs are equal). The problem is to produce a partial order of ordered pairs $(i, j), \quad i, j \in A$; each pair represents a message in which $i$ sends to $j$ all the information that it knows at the time. At the end, each individual must know everything, and the partial order must have minimum total cost.

The results of this paper indicate that the lower bound on the total cost is twice the minimum cost tree spanning the complete graph on $C$, and that TREE-COMMIT solves the problem. Also, our time analysis and taxonomy of algorithms, carry over to this variant of the gossip problem.

8.2 Future Work

The results of this paper raise questions about several variations of the model. First, it is interesting to determine what is the minimum communication cost of a transaction, in which only some of the participants know which are all the other participants. This is the case if the host site of the transaction, which makes the access plan, initiates

---

1. The case of a cost associated with each pair (as opposed to a one-unit cost for all pairs) was considered, but not solved, by Cot (IC).
some subtransactions before knowing all participants. For example, it is possible that for a transaction executed at participants 1,2,3,4, only sites 2 and 4 know what the whole set of participants is. Such limited knowledge of participants' identity can be modeled in our framework, by limiting commit instances to have only vote messages from one participant to another, such that the former knows about the latter.

Second, the minimum communication cost of nonblocking commit protocols, such as three phase commit, should be determined. We conjecture that the minimum communication cost is three times the weight of the minimum spanning tree of the distance graph. Contrast this with the result of Dwork and Skeen in which the inter-site message complexity of blocking and nonblocking commitment protocols is identical ([DS1]).

Third, we suggest examining minimum communication cost for networks in which nonparticipating processors can "help" in the commitment. This leads to a variation of the steiner tree problem, which is NP-complete, but for which good polynomial approximations exist. For example, consider a star network with processors 1,2,3,4, in which processor 4 is the center. Let the participants in the commitment protocol be 1,2,3. If processor 4 does not simply relay messages, but also examines their content, it can reduce communication cost as follows. When receiving a 'yes' vote going from participant 1 to 2, it can hold it until receiving the vote from 3 to 2, and then transmitting one message, of cost one, to participant 2. This way the cost of commitment can be reduced from 8 to 6.

ACKNOWLEDGEMENT

We wish to thank Alon Itai for very helpful discussions and insightful remarks. We also thank the anonymous referees for detailed and thorough reports, that helped us improve this paper significantly.

REFERENCES


2. One-way messages, as opposed to two-way sessions, were considered by Harary and Schwenk ([HS]), and by Golumbic ([Go]).


