COMMUNICATION COST AND TIME OF COMMITMENT ALGORITHMS

by

A. Segall and O. Wolfson

Technical Report #458

July 1987

COMMUNICATION COST AND TIME OF COMMITMENT ALGORITHMS

Adrian Segall
Ouri Wolfson

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, 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 links 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. The execution time of our algorithm is the best one can practically obtain in the class of algorithms which have minimal communication cost.

---

Work of the second author was supported by Technion V.P.R. Fund - E.I.J.Bishop Research Fund. Work of the first author was supported by the Technion Fund for Promotion of Research.
1. INTRODUCTION

In a distributed database a transaction consists of several subtransactions, each running at a different site. The commitment problem (see [1]) 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.

The model considered in this paper is of a distributed database located at the nodes of a computer-communication network (see Fig. 1.1a). Each subtransaction of a given transaction runs at some network 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. We assume that failures of the computing environment components (everything except the transaction) are rare and detectable.

Consider first the communication cost of transaction commitment. Traditionally, the communication cost has been quantified in terms of the number of intersite messages, while disregarding the network topology, particularly, the distance in the network between participant sites. However, the actual communication load on the network is reflected by the number of messages between neighbors in the network referred to here as network messages.
For example, suppose that the participants in a transaction are the boxes in Fig. 1.1a. A possible commitment protocol execution, or instance, is that sites 2,3,4 send their 'yes' votes to site 1, then 1 makes the commit decision and sends it separately to sites 2,3,4. This takes 6 intersite messages. Another possibility is the "linear" scheme where 4 sends its vote to 1, then 1 sends its vote to 3, then 3 sends its vote to 2, then 2 commits and the commit messages travel from 2 to 3 to 1 to 4. This also takes six intersite messages. In fact, it has been shown in [DS1] that six intersite messages is optimal for four participating sites. However, note that the load that these instances place on the communication network is higher than the instance in which 4 sends a 'yes' vote to 3, and then 3 sends a 'yes' vote to 2, while 1 also sends a 'yes' vote to 2, then 2 commits and the commit messages travel from 2 to 3 to 4, and from 2 to 1. If we assume that the protocol messages travel through the shortest path in the network, then the first instance takes 20 network messages, the second takes 22, while the latter takes only 14.

The communication cost criterion 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 sender and receiver sites. In fact, this criterion can be easily generalized. In a network with dynamic routing, 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. For example, in SNA there are eight possible routes with different priorities, between each pair of network nodes ([A]). The communication cost of a message between the sites can be taken to be some weighted average of their lengths.

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 instance? In this paper we establish this cost to be twice the weight of a minimum spanning tree in the complete graph, representing the distance in the network between every pair of sites participating in the commitment protocol (see Fig. 1.1b for example). As explained at the beginning of Section 3, this result is not trivial because, as we shall show, a minimal communication cost instance does not necessarily have a single coordinator, namely a participant to which all other participants send their votes. In fact, the most difficult part of the proof consists of showing that there always exists a minimum communication cost instance with one coordinator. The next question arising: Is the necessary cost also sufficient? We answer this question positively, by presenting a simple distributed algorithm, TREE COMMIT, which achieves the necessary cost.
The other measure to be considered in comparing the performance of commit protocols is the \textit{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 operation implies in particular a message delay of one time unit, independently of the network location of the sender and receiver.

In our model we allow, more realistically, arbitrary subtransaction completion times as well as arbitrary network delays. Therefore, we dispose of unrealistic assumptions regarding synchronous operation of geographically dispersed processors\(^1\). Particularly, different participants may complete their subtransaction at different times, and intersite message delays may differ depending on sender and receiver. We show that the execution time of TREE-COMMIT is the best one can practically obtain in the class of algorithms that have minimal communication cost. Surprisingly, this minimal 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.

\(^1\) Solution to the consensus problem exists since faults are detectable, for example by means of time-out.
Transaction commitment is a variation of a fundamental problem in distributed systems, namely distributed consensus. [F] presents a survey of the subject, and [DS2] presents an interesting taxonomy of consensus problems. Almost all previous research concentrates on the effects of failures of the consensus problem. We, on the other hand, concentrate on performance issues. [MLO] also discussed performance issues of commit protocols, 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, at subtransaction completion time, there may not be any participant which knows the identity of all participants in the transaction. Second, [MLO] as the other previous works, has disregarded the network topology.

Informal discussions of performance of transaction commitment in the absence of failures also appear in [G], [C], [S2], mainly in the context of different two-phase-commitment schemes. One of the most popular is the central site scheme. A designated "protocol coordinator" polls all the 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 other sites. Another scheme is the decentralized. In it each participant sends its vote to all other participants. Based on the received messages each participant makes the 'commit' or 'abort' decision. Finally, we will mention the linear scheme, in which 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. [G] mentions that the linear (it is called nested there) scheme is most efficient in terms of the number of intersite 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 intersite messages required by any transaction commitment protocol, in the absence of failures, is $2(n-1)$. By contrast, note again that in the present work we consider network messages rather than intersite messages.

The rest of the paper is organized as follows. In Section 2 we present the model of a commit instance. In Section 3 we establish the necessary communication-cost for the transaction commitment problem, and in Section 4 we give a complete characterization of minimal communication cost 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-
In Section 7 a measure of comparison between minimal communication cost algorithms is proposed, and a complete classification of commitment algorithms in terms of performance is provided. In Section 8 we conclude, and discuss future work.

2. COMMIT INSTANCES

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 $G$ is connected, and that a transaction in a distributed database executes at a subset of participating sites, or participants, $P \subseteq V$. 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 commit/abort protocol is executed. We will mainly analyze the case in which each site votes to commit the transaction, for the following reason. In Section 5 we discuss the extension of the results to the 'abort' case, and show that it is less expensive (from the communication cost point of view) than the 'commit' case. Therefore 'commit' represents a worst-case scenario, which is also the more frequent one.

Intuitively, an instance of the commit protocol is represented by the temporal, and thus partial, order of events occurring at the participants. Formally, an instance of the commit protocol, 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 $I$. The 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 an intersite message (without an intervening receive). 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. Only the last event occurring at a site may represent the sending of zero messages, and only the first event occurring at a site may represent the receiving of 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.

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. An event such as $e$ is called a C-event, because when it occurs, its site already knows the commit decision. A message sent by a C-event is called a commit message. Each message which 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 ([DS1]). Note that an event which happens after a C-event is also a C-event. Note also that a C-event may represent the receive of a vote message. 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 $V_{ij}$ ($C_{ij}$) of a node indicates that this is the $j$'th V-event (C-event) occurring at participant $i$.

Next we make two remarks about the preceding definitions. The first concerns the subtransaction completion, which occurs at every participant. We assume that a participant may send messages only after its subtransaction completes. Consequently, the first event at each participant represents the corresponding subtransaction completion, in addition to the send and receive of messages. The second remark concerns the "collapsing" of several message-receives and -sends into 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. Furthermore, the only relevant fact about subtransaction completion is that it must occur before the first send of a message, and this is reflected in the commit requirement.

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 most order arcs. However, it should be remembered that any two consecutive events at a site are connected by an order arc.
3. COMMUNICATION COST OF INSTANCES

In this section we establish the minimal communication cost of a commit instance. The proof would have been quite straightforward, had we known that in a minimal communication cost instance all votes must be sent to one participant, the coordinator, which then broadcasts the commit decision. However, as we shall show in Section 4, this is not necessarily the case.

We start with some formal definitions. The 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 has a strictly positive cost. The communication cost of an instance $I$, denoted $\text{Cost}(I)$, is the total communication cost of $I$. Denote by $\Psi$ the set of commit instances for some set of participants $P$, and communication network $G$. In this section we establish the minimal communication cost of a commit instance in $\Psi$. An instance of such cost will be referred to as a minimal 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.

Lemma 1: If $I$ is a minimal 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.

Proof: Assume that there are two or more C-events at some participant. There must be an incoming message into the second C-event by definition of an instance. By omitting this message, a commit instance of strictly lower

---

![Diagram of two-phase-commit protocol](image)

(a) $V_{1,1} \rightarrow V_{2,1} \rightarrow C_{2,1} \rightarrow C_{3,1}

(b) $V_{1,1} \rightarrow C_{1,1}

(c) $V_{1,1} \rightarrow V_{2,1} \rightarrow C_{2,1} \rightarrow C_{3,1} \rightarrow C_{1,1}

Figure 2.1: Schemes of the two-phase-commit protocol (a) Central with participant 1 the coordinator; (b) Decentralized; (c) Linear.
communication cost can be obtained\(^1\). The reason 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. \(\square\)

The (single) C-event at every participant \(i\) will be denoted by \(C_i\). Assume that \(C_j\) is a root of \(C_j\) for some minimal communication cost instance \(I\). This means that in the execution \(I\), 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 a commit instance of minimal communication cost in \(\Psi\), which has exactly one coordinator.

**Proof:** Let \(I\) be a minimal communication cost instance. We shall show that if \(I\) has two or more coordinators, then we can construct another commit instance \(I'\) of equal communication cost, but with one less coordinator. The construction is by substituting a commit message for a vote message between the same participants. Next we describe the selection of the vote message to be substituted for.

The C-events at the coordinators 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 \(C_j\) 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 V-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 V-event.

Suppose that \(V^o\) occurs at participant \(p_o\). Assume, without loss of generality, that \(C_{p_o}\) is not in the subtree rooted at \(C_i\). Denote the events on a path from \(V^o\) to \(C_i\), except the endpoints of the path, by \(V^1, \ldots, V^n\), (note that they must be V-events). Let \(V'\) be the last event on this path for which the following condition is true: \(V'\) occurs at a participant, \(p_r\), for which \(C_{p_r}\) is in a subtree different than the one rooted at \(C_i\). Since \(V^o\) satisfies the condition,

---

\(^1\) Note that after the omission the instance has to be adjusted if the omitted message was the only one exiting its send event, \(e\), and \(e\) is not the last event at its site; adjustment is by collapsing \(e\) and its consecutive event at a site. Two events \(e\) and \(f\) are collapsed by omitting the event \(f\).
there must be such $V'$. Denote the site at which $V'^{+1}$ occurs by $p_{r+1}$. By the definition of $V'$, the event of $C_{p_{r+1}}$ is in the subtree rooted at $C_i$. Now to obtain a commit instance of minimal communication cost with one less coordinator, perform the following transformation:

i) the arc $V'\rightarrow V'^{+1}$ is replaced by a message from $C_{p_i}$ 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

iii) if 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.

These modifications of $I$ are illustrated in Fig. 3.1. It is easy to see that these modifications result in a commit instance, $I'$. Also, the commit requirement is satisfied by $I'$ because by definition of $V'$, the removal of the arc $V''\rightarrow V'^{+1}$ 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 $C_{p_{\alpha}}$, which is in a different C-subgraph tree. Therefore $I'$ is a commit instance, and has the same coordinators as $I$, except participant $i$. Moreover, the transformation performed on $I$ does not alter the communication cost, hence $I'$ has minimal communication cost.

To summarize, starting with a minimal communication cost instance, $I$, with two or more coordinators, we obtained a minimal communication cost instance, $I'$, with one less coordinator. We can continue this procedure until a minimal communication cost commit instance with exactly one coordinator is obtained. □

Next we will obtain an additional lemma. In a minimal communication cost instance, $I$, having two or more coordinators, let $V^*_{p_{\alpha}}$ 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 the lemma that $I$ must have at least one boundary $V$-event. Assume that $C_{i}$ is a boundary $C$-event that is a successor of $V^*$, is not $C_{p_{\alpha}}$, and does not precede $C_{p_{\alpha}}$.

Such a boundary $C$-event will be called associated with boundary $V$-event $V^*_{p_{\alpha}}$.

Now let us return to the proof of Lemma 2. The fact that $I$ is a minimal communication cost instance, implies that the path $p$ from $V^*_{p_{\alpha}}$ to $C_{i}$ in $I$ is unique, and consists of a single message arc, $V^*_{p_{\alpha}} \rightarrow C_{i}$. The reason for this is as follows. If this is not the case, then there are more messages on $p$, or additional paths from $V^*$ to $C_{i}$. As established, one of the messages on $p$ is from $p_{\alpha}$ to $p_{r+1}$. Then, all messages on $p$ and the other paths, have been replaced in the proof of Lemma 2 by a message from $C_{p_{\alpha}}$ to $C_{p_{r+1}}$ to obtain the commit instance $I'$. This implies that $I'$ has a cost strictly lower than $I$, contradicting the fact that $I$ has minimal communication cost. Therefore, $V^* = V^{*'}$, $V^{*'} = C_{i}$, $p_{\alpha} = p_{i}$, $i = p_{r+1}$, and the transformation simply replaced a message $V_{p_{i}} \rightarrow C_{i}$ by a message $C_{p_{\alpha}} \rightarrow C_{i}$. We have therefore proved that:

Lemma 3: In any minimal 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, $V^*$, to any associated $C$-event, $C_{i}$, and it consists only of the message $V^* \rightarrow C_{i}$. □

For the proof of the next theorem we need the following definitions. Let $I$ be a commit instance, and $Z$ a directed graph having the set of participants $P$ as nodes. We say that the vote messages of $I$ correspond to $Z$, if the following condition holds. For every arc $i \rightarrow j$ of $Z$ there is a vote message of $I$ sent from $i$ to $j$, and vice versa; for every vote message of $I$ there is an arc of $Z$. Similarly we define the commit messages of a commit instance to
correspond to a directed graph.

Now, let us define the distance graph $D$ as the complete graph having the set of participants $P$ as its nodes, and the weight of each edge $(i,j)$ equals to the length of 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 $k \in P$, 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 $k$. The instance $I(k,T_1,T_2)$ is a commit instance that has a set of vote messages which correspond to $T_1'$, and commit messages which correspond to $T_2'$. The definition of $I(k,T_1,T_2)$ is illustrated in Fig. 3.2. 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 $P$ a set of participants. Then

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

**Proof:** Since the cost of an instance $I(k,T_1,T_2)$ for some minimum spanning tree $T_1$ is $2\cdot\text{CMST}(G,P)$, then clearly $2\cdot\text{CMST}(G,P) \geq \min_{I \in \mathcal{I}} \text{Cost}(I)$. To finish the proof observe that from Lemma 2, there exists an instance $I^*$ of minimal communication cost, with one coordinator, say $k$. Based on $I^*$ construct an undirected graph, $H$, defined

![Diagram](image-url)

Figure 3.2: Minimum spanning trees $T_1$ (a) and $T_2$ (b), and an instance on $T_1$ and $T_2$ coordinated at site 4 (c).
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 $j$ in $I^*$. Since in $I^*$ there is a path from the first event at every site to $C_T$, $H$ 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, $2\cdot CMST(G,P) \leq \min_{I \in \mathcal{I}} \text{Cost}(I)$. 

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 $\min_{I \in \mathcal{I}} \text{Cost}(I) = 2(n-1)$. 

4. CHARACTERIZATION OF MINIMAL COMMUNICATION COST INSTANCES

In this section we give a complete characterization of all possible instances of minimal communication cost. We determine that if an instance has a minimal communication cost, then each site sends at most one vote message, and receives at most one commit message. Also, in a minimal 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 minimal 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 minimal communication cost instance $I(k,T_1,T_2)$ on minimum spanning trees $T_1$ and $T_2$, coordinated at a site $k$, was defined in Section 3. 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 sites $m$ and $n$ are neighbors in both trees. A minimal communication cost instance on $T_1$ and $T_2$ coordinated at $m$ and $n$ is denoted $I(m,n,T_1,T_2)$ and 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 adding to it the arc $m \rightarrow n$. Denote by $T_2'$ 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 \( m \rightarrow n \). \( I(m,n,T_1, T_2) \) is a commit instance having the vote messages correspond to \( T_1' \) and the commit messages correspond to \( T_2' \). 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.

The definition is illustrated in Fig. 4.1. (order arcs are indicated in Fig. 4.1 only at the coordinators).

Theorem 2: Any minimal communication cost instance must be of the form \( I(k, T_1, T_2) \) or \( I(m, n, T_1, T_2) \) for some participants \( k, m, n \), and minimum spanning trees \( T_1, T_2 \) of the distance graph.

In order to prove Theorem 2 we shall show that a) any one-coordinator minimal communication cost instance must be of the form \( I(k, T_1, T_2) \); b) any two-coordinator minimal communication cost instance must be of the form \( I(i, j, T_1, T_2) \) and c) an instance with three or more coordinators cannot have minimal communication cost.

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

Proof: Consider a minimal communication cost instance \( I \) whose coordinator is \( k \). 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 \( (i,j) \) 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

---

**Figure 4.1:** The minimal communication cost instance on \( T_1 \) and \( T_2 \) of Figure 3.2, coordinated at sites 2 and 4.
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 valid minimal communication cost instance $I$ with two or more coordinators into another valid 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 minimal communication cost instances with two or more coordinators.

**Lemma 5**: A two-coordinator minimal communication cost instance $I$ must be of the form $I((i,j), T_1, T_2)$, for some $i,j$ 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 $I$. 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 $V^*$ denote a boundary $V$-event, as defined after the proof of Lemma 2. Since $i$ and $j$ are the only coordinators, $V^*$ precedes both $C_i$ and $C_j$. Let $C_i$ be the boundary $C$-event associated with $V^*$, and $p_o$ the participant at which $V^*$ occurs. Since there are only two boundary $C$-events, by definition of an "associated boundary $C$-event", $C_p$ 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 $V^*$ to $C_j$ is the message arc $V^* ightarrow C_i$. Consider the instance $I'$ obtained from $I$ by replacing $V^* ightarrow C_i$ by $C_p ightarrow C_i$ (as in the proof of Lemma 2). Clearly $I'$ is a commit instance. $I'$ has only one coordinator, $j$, and has the same cost as $I$, i.e. minimal 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 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 $V^* ightarrow C_i$, hence $V^*$ occurs at site $j$, or in other words $p_o = j$.

Therefore, $i$ and $j$ are neighbors in $T_2$. Also, recall that $I'$ is obtained from $I$ by simply replacing $V_o ightarrow C_i$ by $C_j ightarrow C_i$, and that $I'$ has the form $I((j,T_1, T_2)$. Therefore the instance $I$ must have the following properties: i) exactly one vote message is sent by each participant, and therefore exactly one $V$-event occurs at each participant; ii) the vote messages correspond to the graph consisting of the oriented tree with sink $j$ obtained by directing the edges of
T_1, plus the arc j\rightarrow i; iii) the commit messages correspond to the graph consisting of the rooted tree with root j obtained by directing the edges of T_2, minus the arc j\rightarrow i.

Left to show is only that i is a neighbor of j in T_1, or in other words, that i sends its vote to j in I. Denote the single V-event at each participant r 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^i another boundary V-event, i.e. V^i\neq V^o. The associated C-event of V^i cannot be C_j , since otherwise as above, we can show that V^i=V_j, which in turn implies that V^i=V^o. Thus the associated C-event of V^i must be C_j, and as above V^i occurs at participant i, i.e. is actually V_i, and participant i sends its vote to participant j.

Lemma 6: There are no minimal communication cost instances with three or more coordinators.

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

Suppose that there exists a minimal 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_{p_o}\rightarrow C_i, we obtain as in Lemma 2 an instance I' of minimal 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., \notin 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.

The following is an immediate result of Theorem 2:

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

Obviously, as demonstrated in the introduction, the converse of corollary 2 is not true, i.e. minimal number of inter-site messages does not imply minimal communication cost.
The characterization in Theorem 2 helps us demonstrate that in general, minimal communication cost cannot be achieved with limited knowledge of participants' identity. For example, suppose that in Fig. 1.1a, 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 minimal spanning tree, such a scenario cannot achieve minimal communication cost.

5. THE PROPOSED TRANSACTION COMMITMENT ALGORITHM

In discussing commitment algorithms we assume, as in [DS2], that when completing its subtransaction each participant knows the identity of all the participants, and in addition we assume that it knows the network, \( G \). The analysis of section 3 suggests a very simple minimal 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 \( T \) with its edges directed towards \( k \); the commit messages correspond to \( T'' \), which is \( T \) with its edges directed away from \( k \). Specifically, the procedure executed by a participant, \( i \neq k \), is to wait until subtransaction completion and until receiving all vote messages represented by the arcs incoming into \( i \) in \( T' \), and then to send a 'yes' vote message (assuming the commit case) corresponding to the arc exiting \( i \). Participant \( k \) completes its subtransaction and receives votes from all neighbors in \( T \), and then commits. The behavior after the commit 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 minimal 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 minimal communication cost algorithm adapted from the PIF algorithm of [S1]. Each participant, after completing its subtransaction, constructs a minimum spanning tree, \( T \), of the distance
graph, common to all participants. Contrary to fixed-instance, a coordinator is not selected. Then 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 $T$. Clearly 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 minimal. A possible situation in the voting stage, i.e. before the coordinators are determined, is illustrated in Fig. 5.1a. Suppose that in the scenario illustrated participants 1, 2, and 5 have completed their subtransactions, and participants 3 and 4 have not. Note that participant 2 has not voted yet because it has not received the votes of two of its neighbors. Fig. 5.1b, 5.1c illustrate possible instances executed by TREE-COMMIT at the completion of the described voting stage situation. In the first case 3 completed its subtransaction, and its vote had reached 2 before the vote of 4 did so. In the second case, 3 completed its subtransaction after having received the vote of all participants.

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. In this modified scheme each participant of an established sequence, except its endpoints, votes as soon as it completes its subtransaction and receives the vote of one of its neighbors in the linear string. 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 abort case is handled by amending TREE-COMMIT by the following step: Site $i$ sends 'abort' messages when the first of the following two cases occurs: i) It had unsuccessfully terminated its subtransaction; then an 'abort' message is sent to each neighbor or ii) it had received the first among all 'abort' messages that are received at site $i$, from site $j$; then an 'abort' message is sent to each neighbor in $T$, except $j$. 
Figure 5.1: (a) An execution of TREE-COMMIT(T) in its voting stage. (b,c) Two possible instances at completion.

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

Proof: 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 $2CMST(G,P)$. 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 $CMST(G,P)$. 

Note that TREE-COMMIT uses only 'yes', 'commit', and 'abort' messages, therefore the message length can be restricted to two bits.
Our algorithm is based on the assumption that each participant knows the network topology and the identities of all participants at subtransaction completion time. Since failures are rare in the environment that we discuss, the network topology is stable, and therefore knowledge of this topology does not pose a problem in practice. Knowledge of the identities of all participants at subtransaction completion time, 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 practical problem with it.

6. EXECUTION TIME OF COMMIT INSTANCES

In this section we first define the term execution-time of an instance in an asynchronous network, and then show that for a minimal spanning tree $T$, TREE-COMMIT($T$) dynamically and distributively selects among all minimal communication cost commit instances on $T$, the one with minimal execution time. A minimal communication cost commit instance on $T$, i.e., an instance of the form $I(k, T, T)$ or $I(m, a, T, T)$, 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 asynchronous network behavior must be imposed. The only restriction we impose here is that the delay of a message between every pair of participants is a characteristic of the communication network, and thus any message, sent by any algorithm from $i$ to $j$, takes a fixed 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 communication link delays. 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 its received at $j$ ($t_{ij}$ may be different than $t_{ji}$). 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, and a participant $i$ does not send its first message before time $\tau_i$.

**Theorem 4:** Execution of TREE-COMMIT($T$) generates a $T$-instance of minimal execution time among all $T$-instances, for any given set of subtransaction completion times $\tau = \{\tau_1, \ldots, \tau_n\}$, and any given set of communication link delays $\tau = \{t_{ij} \mid i,j \in P\}$.

**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 $t_{ij}$. 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 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 $\tau_i, \tau'_i$ 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 $\tau'_i$ as the time when $k$ receives the last vote message. Observe that

$$\tau_i \leq \tau'_i \text{ if } j \text{ is not a single coordinator in } I'$$
$$\tau_i > \tau'_i \text{ if } j \text{ is a single coordinator in } I'.$$

We want to show that

$$\tau_i \leq \tau'_i \text{ for every participant } i.$$  

(2)

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

$$\tau_j = \min(\tau'_i : \tau_i > \tau'_i).$$

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 $\tau'_j$, namely

$$\tau'_j + t_{jr} \leq \tau'_j \forall r \in K$$

(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 $t_{jr} > 0$, we have

$$\tau'_{r} < \tau'_j \forall r \in K$$

(4)

and by definition of $j$, holds
From (6) we see that in $I$, each $r \in K$ sends its vote before receiving a vote from $j$, and consequently it must send its vote to $j$. From (1) 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. If $j$ is single coordinator in $I'$, then $K$ is the set of all its neighbors in $T$ and $j$ receives the vote of each one before $v_i'$. Therefore $j$ is a single coordinator in $I$ as well, contradicting the assumption that $I$ has two coordinators. On the other hand, 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 $t_j \leq v_j'$. Namely at time $v_j'$ or earlier TREE-COMMIT sends its vote in $I$, i.e. $v_j \leq v_j'$ contradicting the fact that $v_j > v_j'$.

This completes the proof that $v_i \leq v_i'$ for every participant $i$.

To complete the proof, 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 $\max(v_m' + d_m, v_n' + d_n)$. 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 algorithms. The primary comparison criterion is taken to be communication cost, therefore we restrict our attention to algorithms which achieve minimal 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 minimal 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 link delays, or delay invariant, and the ones which are not (thus the processors incur an extra overhead of monitoring and learning the communication link delays in the entire network). TREE-COMMIT belongs to the first class. 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 communication link delays between the participants, and any set of subtransaction completion times, the execution time of the instance of algorithm $A$ is not worse than the execution time for $B$. 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 $t=(t_{ij} \mid i,j \in P)$ is a set of communication link 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 minimal 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 minimal 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 $t$ will be called a fixed instance $T$-algorithm. If algorithm $A$ dominates algorithm $A'$, and in addition there exist $\tau_0$ and $t_0$ for which $X(A(\tau_0,t_0))<X(A'(\tau_0,t_0))$, then $A$ is said to strictly dominate $A'$.
Theorem 6: TREE-COMMIT(T) strictly dominates any minimum communication cost fixed instance T-algorithm.

Proof: The fact that TREE-COMMIT(T) dominates any T-algorithm is an immediate consequence of Theorem 4. In order to show strict dominance we have to find for any fixed instance T-algorithm, A, sets τ and t for which TREE-COMMIT(T) (τ,t) has execution time strictly less than A. In order to identify such sets, consider the participant i in A, which is not a coordinator. Denote by R some very large integer. Let τ_i=2R, and t_{ij}=R for any neighbor j of i in T. Denote the number of participants by n. All other subtransaction completion times of τ are zero, and each other link delay of t is R/n. With these τ and t, 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 (τ,t) will be at least 4R. The reason is that since participant i is not a coordinator, the coordinator can only send its first message at 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: if for some τ and sets of link delays t and t' that are equal except possibly on the delay between i and j, a message is sent from i to j in A (τ,t), then such a message is also sent in A (τ,t'). In other words, an algorithm is delay invariant if it behaves as follows: if for some τ and t site i sends a message to site j, then it will do the same if the link delay t_{ij} changes, provided that everything else stays the same. Algorithms which do not monitor link 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 on adjacent links, and behave differently for different delays. Obviously this puts additional load, and complicates the algorithm.

Theorem 7: There is no delay-invariant algorithm which strictly dominates TREE-COMMIT.

Proof: Assume that A is a delay-invariant algorithm which strictly dominates TREE-COMMIT(T) for some minimum spanning tree T. By Theorem 4, algorithm A cannot execute only T-instances. Thus, for some sets τ and t, A (τ,t) sends a message from i to j, where i and j are not neighbors in T. Consider t' in which the delays are the same as in τ, except that t'_{ij} is a very large number (for example, the sum of the other delays in t and the completion times). Since A is delay-invariant, A (τ,t') sends a message from i to j, and therefore X(A (τ,t')) > X (TREE-COMMIT (T) (τ,t')) contradicting the fact that A strictly dominates TREE-COMMIT(T).
Algorithms which are not delay-invariant can strictly dominate TREE-COMMIT. For example, consider the algorithm E(xtended)TREE-COMMIT which 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 property that has a shortest (among all trees) longest path in terms of link delays from i. If the commit messages are propagated along the edges of T', earlier execution time is possible. It is intuitive, and it can be proven rigorously, that this ETREE-COMMIT strictly dominates TREE-COMMIT. In conclusion, Figure 7.1 illustrates the classification of the minimum communication cost algorithms with respect to a minimum spanning tree T. TREE-COMMIT(T), a T-algorithm, dominates all T-algorithms, and it strictly dominates all T-instance algorithms. There is no delay invariant algorithm which strictly dominates TREE-COMMIT, but there is a delay variant algorithm, ETREE-COMMIT which does so.

There are three disadvantages to ETREE-COMMIT, constituting the reason it is not our proposed algorithm. First, the processors incur the extra overhead of monitoring link delays. The second is that it is an NP-complete problem to find a minimum spanning tree with shortest-longest delay ([1]). The third is that '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. In fact it can be shown that it is a hard problem for an algorithm to find a "best"

<table>
<thead>
<tr>
<th>delay invariant algorithms</th>
<th>delay variant algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>T-algorithms</td>
<td></td>
</tr>
<tr>
<td>fixed instance algorithms</td>
<td></td>
</tr>
<tr>
<td>TRE-COMMIT</td>
<td></td>
</tr>
<tr>
<td>ETREE-COMMIT</td>
<td></td>
</tr>
</tbody>
</table>

Figure 7.1: Minimum communication cost algorithms with respect to a minimum spanning tree, T.
Theorem 8: The following problem, MINIMAL COST AND TIME INSTANCE (MCTI) is NP-complete. The input consists of a network graph \( G = (V, L) \), a set of participants \( P \subseteq V \), a set of subtransaction completion times, \( \tau \), a set of intersite link delays, \( \tau \), and an integer \( K \). The question is whether there exists a minimal communication cost instance with execution time \( \leq K \) with respect to \( \tau \) and \( \tau \)?

Proof: The problem is obviously in NP. A nondeterministic algorithm for it need guess an instance, and check to see whether it has minimal cost, and execution time \( \leq K \). Next, we will transform the MRMST problem, defined next and shown NP-complete in [1], to the MCTI problem.

MINIMUM RADIUS MINIMUM SPANNING TREE (MRMST):

**Input:** A graph \( H = (V, E) \), a vertex \( r \in V \) (the root), a weight function, \( w \), defined on \( E \) and satisfying the triangle inequality, and an integer \( k \).

**Question:** Does \( H \) have a minimum spanning tree, \( T \), such that the weight of the path from \( r \) to any node is less than, or equal to, \( k \).

Given a MRMST instance we construct a MCTI instance as follows. Assume that \( m \) is the total weight of a minimum spanning tree of \( H \). The network graph \( G \), consists of \( H \) with each edge \((i, j)\) replaced by a path of length \( w(i, j) \). The path has \( i \) and \( j \) as endpoints, and if \( w(i, j) > 1 \), then new intermediate nodes are added, to extend the length. For each edge different intermediate nodes are used. Additionally, a path \( p \) of new nodes, \( z, z_2, \ldots, z_m+1 \), is added. The \( z_i \)'s are different than the previously defined intermediate nodes. The path \( p \) is connected by an edge to \( r \). Denote \( z_{m+1} \) by \( z \).

The set of participants, \( P \), consists of \( z \) and the nodes of the network which are also in \( H \), the link delay between any pair of participating sites is equal to the distance between the two sites, and the completion time of each subtransaction is zero, except for the completion time of site \( z \), which is \( 2m+1 \). Finally, \( K = 3m+2+k \). This completes the construction of the MCTI instance. Now we will show that the MRMST instance has a solution if and only if the MCTI instance has a solution.

(\( \ast \)) Let the commit instance \( I \) be a solution to the MCTI problem. We assume that \( k \leq m \), otherwise the MRMST problem is trivial. Clearly, any minimum spanning tree of the distance graph of the network \( G \), consists of a minimum spanning tree of \( H \), connected to the additional node \( z \). The connection of \( z \) to the minimum spanning
tree of $H$ is by the edge $r-z$ of length $m+1$. Therefore $z$ is a leaf in the minimum spanning tree of the distance graph.

Assume that $z$ is not a coordinator of $I$. Consider the execution time of $I$. Participant $z$ can send its vote message at time $2m+1$, or later, because of its subtransation completion time. This vote message reaches the coordinator at time $3m+2$ or later, because the message has to travel on the slow link from $z$ to $r$. The commit message has to travel to $z$ along the same slow link, thus it cannot reach $z$ at an earlier time than $4m+3$. and the commit message reaches $z$. But $4m+3 > K$. This implies that the MCTI problem does not have a solution, contradiction. Therefore we conclude that $z$ is a coordinator. Its vote (or commit if it is a single coordinator) message must be sent to $r$, because $z$ is a leaf connected to $r$ in any minimum spanning tree. The message cannot be sent before time $2m+1$. Therefore, $r$ sends its commit messages at time which is at least $3m+2$. The commit messages propagate along the edges of a minimum spanning tree of $H$ to all participants of $H$. Since the execution time of $I$ is $3m+2+k$ or less, there is a minimum spanning tree of $H$ with a maximal path length, from $r$, which is not higher than $k$.

(only if) Assume that $t$ is a solution to the MRMST problem. Denote by $T$ a minimum spanning tree of the distance graph consisting of $z$ connected by $z-r$ to $t$. It is easy to see that $I(z,T,t)$, i.e. the instance built on $T$ with the single coordinator $z$, is a solution to the MCTI problem.

8. DISCUSSION

Conclusion

In this paper we established the minimum communication cost of commitment, characterized minimal communication cost instances, proposed the TREE-COMMIT algorithm, and 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 algorithm which adjusts to different communication link delays and subtransation completion times, and therefore is time-efficient. 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 minimal communication cost, and compared their execution time. The comparison indicates the need to distinguish between two types of algorithms: the ones which use advance information on communication link delays, and the ones which do not. There is no algorithm which is superior to TREE-COMMIT in the latter class, but there are such in the former.
However, the former class of algorithms has several disadvantages, which we discussed qualitatively. They concern algorithm computational complexity, the need to monitor link delays, and the length of control messages. Because of these drawbacks we feel that in practice it is hard to argue for an algorithm which uses advance information on link delays. Furthermore, we conjecture that without advance information on subtransaction completion times, which is usually not available, knowledge of link delays cannot improve significantly the execution time of TREE-COMMIT.

Future Work

The results of this paper raise questions about several variations of the model. First, it is interesting to determine what is the minimal communication cost of a transaction, in which at subtransaction completion time only some of the participants know which are all other participants. This is the case if the site which makes the access plan initiates 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. Second, the minimal communication cost of nonblocking commit protocols, such as three phase commit, should be determined. We conjecture that for asynchronous networks, the minimal communication cost is three times the weight of the minimum spanning tree (of the distance graph). Contrast this with synchronous networks, in which the message complexity of blocking and nonblocking commitment is identical (DS1). Third, we suggest examining minimal 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 good polynomial approximations exist for it. 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.
REFERENCES


