OVERHEAD OF LOCKING PROTOCOLS
IN A DISTRIBUTED DATABASE

by

Ouri Wolfson

Technical Report #386

November 1985
OVERHEAD OF LOCKING PROTOCOLS IN
A DISTRIBUTED DATABASE

Ouri Wolfson

The Technion - Israel Institute of Technology,
Computer Science Dept., Haifa 32000, Israel.

ABSTRACT

The main purpose of a locking protocol is to ensure correct interleaving of actions executed by different transactions. The locking protocol consists of a set of rules dictating how accessed entities should be locked and unlocked. As a result of obeying the rules the transactions incur an overhead, particularly in a distributed database. We propose three measures for evaluating this overhead, each most suitable to a different type of underlying communication network. Then we analyze and compare three protocols according to each measure: two-phase-locking ([EGLT]), two-phase-locking with a fixed order imposed on the database-entities (ensuring deadlock freedom), and the tree protocol ([SK1]). Our analysis is based on a graph theoretic model.
OVERHEAD OF LOCKING PROTOCOLS IN
A DISTRIBUTED DATABASE

Ouri Wolfson

The Technion - Israel Institute of Technology,
Computer Science Dept., Haifa 32000, Israel.

1. INTRODUCTION

Many concurrency control mechanisms for a distributed database appeared in the literature. Most of them are surveyed by Bernstein and Goodman in [BG1]. There, the algorithms are categorized into two main classes, according to the mechanism used: Locking and Timestamping. The surveyed locking-based algorithms employ the popular two-phase protocol. Two-phase-locking (2PL) states that in order to ensure correctness of concurrent executions every transaction has to obtain locks on all database entities it accesses before unlocking any entity. Other locking protocols have been proposed, and they allow more concurrency if the database is physically or logically structured. They are called structured protocols. The best known is the tree protocol (TP) proposed in [SK1]. It assumes that the database entities are structured as a rooted directed tree. The protocol provides the following rules for locking and unlocking entities. A transaction issues its first lock on an arbitrary entity. Subsequent locks can be issued when they request an entity which is a child in the database tree of some entity y, and the transaction holds a lock on y. Each entity can be locked only once. In addition to ensuring correctness the tree protocol is also deadlock free. 2PL can also ensure deadlock freedom by imposing a total order on the database entities, thus transforming 2PL into a structured protocol, and adding the
A transaction can issue a lock only when it requests an entity which succeeds in the total order all the entities the transaction holds a lock on [U].

Structured locking protocols have been studied extensively [FKS, KS1, KS2, KS3, S, SK2, Y]. However, little attention has been paid to specific issues pertaining to a distributed environment. Some theoretical and complexity issues have been addressed in [W], but the issue of overhead has been largely ignored thus far. For 2PL, an informal discussion of overhead appears in [BG2].

In this paper we formalize the notion of locking protocol overhead. Intuitively, it is the overhead incurred by the transactions in obeying the rules of the protocol. This overhead is negligible in a centralized database. In a distributed database this is not so, for the following reason. Distributed transactions execute in parallel at different computers of the network (sites), and the processes at different sites communicate by message passing; however, obeying the rules of the protocol inhibits parallelism and requires message passing. As a result, we propose three measures for evaluating locking protocol overhead.

The first one deals with the issue of parallel execution and measures the number of protocol steps that must be executed sequentially. Namely, we ask what is the longest sequence of Lock-Unlock steps that any transaction obeying the protocol must have? This measure is appropriate for example in a fast local-area network, where the completion time of a transaction (or "response time") is determined by the longest sequence of steps, regardless of the number of intersite messages required in order to execute it. The second measure quantifies the longest sequence of intersite messages that any transaction obeying the protocol must have. It is appropriate in a wide-area network, where this determines response-time. The third measure quantifies the total number of intersite messages the protocol requires. It determines performance in a network in which the total number of messages constitutes the bottleneck. After formally defining the
measures we obtain the overhead of 2PL by each measure. Then, assuming a fixed
distribution of the database entities, we obtain the overhead of TP for "best" and
"worst" tree structures, and the overhead of deadlock-free 2PL for "best" and
"worst" order on the database entities. On the way we demonstrate what these
best and worst structures are. The demonstration also indicates, as the other
side of the same coin, what are the best and worst ways to distribute the entities
of a fixed database structure.

2. PRELIMINARY DEFINITIONS

2.1 Database

A database is a finite set of entities. The entities are the lockable units and we
are not concerned with their granularity; they can be blocks, records, files, etc. A
database is distributed if the entities are partitioned into pairwise disjoint sites.
Formally, with the set of entities is associated a resides-at (or distribution) func-
tion, assigning a value called site to each entity. A site is a computer-identification
and we assume that an entity cannot be stored in two computers. This implies
that our model is concerned with physical rather than logical entities (a logical
entity can have several physical replicas); in particular, locking is performed on
physical entities by the data-manager executing at the site ([BG1]).

2.2 Transactions

A transaction represents some possible execution of a set of programs. The
set is characterized by the fact that the programs cooperate although they may
run at different sites, and every execution of them as a unit, leaves a consistent
database if it starts with a consistent one. The formal definition of a transaction
(see below) ignores the read-write actions executed by the programs since in this
paper we are interested in the overhead incurred by the programs as a result of
obeying a locking protocol. Thus, we model a transaction by its locking protocol
steps. These steps are partially ordered and the partial order is imposed by two factors. First, the steps referencing entities residing at the same site are executed sequentially by the same computer. Second, a program of a transaction can wait for an intersite message from another program of the same transaction before executing the next step. This imposes an order between two steps of the same transaction executing at different sites.

Formally, a transaction $T$ is a directed acyclic graph $(\text{dag}) (S,A)$ such that the set of arcs $A$ is partitioned into two sets. A set of intrasite arcs $A_1$, and a set of intersite arcs $A_2$. $S$ is a set of Lock, Unlock steps referencing the database entities. The lock of an entity $x$, denoted $L_x$, is in $S$ if and only if its unlock, $U_x$, is in $S$. We say that $L_x$ and $U_x$ are executed at the residence site of $x$. If $(a,b) \in A_1$ then $a$ and $b$ reference entities residing at the same site. $A_1$ induces in $T$ a partial order in which all steps referencing entities residing at the same site are totally ordered. In this partial order $L_x$ precedes $U_x$ for any entity $x$ referenced in $T$. These restrictions on $A_1$ give in the case of one site the usual model of centralized transactions as sequences of steps. $A_2$ represents intersite messages. If $(a,b) \in A_2$ then $a$ and $b$ reference entities residing at different sites. A trivial observation which we use extensively in proving our results is the following: If in a transaction, a step executing at one site precedes a step executing at another site then every path between the two steps contains at least one intersite arc.

Our transaction definition is similar to the one introduced in [KP1] with the exception of the distinction between intersite and intrasite arcs. This distinction is crucial for the upcoming analysis.

2.3 Locking Protocols

A locking protocol is a set of transactions. Intuitively, the set consists of the transactions resulting from a possible execution of programs obeying some rules on the way locking and unlocking of entities should be carried out. Now we
formally define the locking protocols we consider in this paper. The first one is two-phase-locking. $2PL = \{ T \mid T$ is a transaction in which every lock precedes all the unlocks$\}$. (Format in Fig. 1(a).)

The second locking protocol we will consider is denoted by $DF2PL_p$ for some total order $p$ imposed on the database entities. $DF2PL_p = \{ T \mid T$ is in 2PL and: if $T$ locks entities $x$ and $y$, and $x$ precedes $y$ in $p$ then $L_x$ precedes $L_y$ in $T$}. (Fig. 1(b).)

For a database which is structured (logically or physically) as a rooted directed tree $t$ the tree policy, denoted $TP_t$, is defined as follows: $TP_t = \{ T \mid T$ is a transaction in which there exists a lock step which precedes all other steps of $T$, and any other lock step, $L_y$, is preceded by the lock of the father of $y$ in $t$, $f(y)$, and succeeded by the unlock of $f(y)\}$. (Fig. 2.)

For the sake of brevity, we omit the formal definitions of schedule, serializability and deadlock freedom (see [KP1], [W], [WY]). However we will observe that $2PL$, $DF2PL_p$, and $TP_t$ ensure serializability (serializability and deadlock freedom).

Also, the sets we defined are maximal in the following sense. If some transaction which is not in 2PL is added to the set 2PL, then there exists some non-serializable schedule (or history) of transactions in the newly created set. Similarly, if for some total order $p$ a transaction is added to $DF2PL_p$ then there exists some non-serializable schedule of transactions in the newly created set, or some partial schedule resulting in a deadlock. The same is true for the tree policy.
Figure 1 - Format of a transaction in 2PL (a) and in $DF2FL_p$ (b). Entities referenced by the steps are omitted.

Figure 2 - A database structure $t$ (a) and a transaction in $TP_t$ (b). Entities $v, x, z, x, z$ reside at different sites.

3. THREE OVERHEAD MEASURES

In this section we formally define the overhead measures, and discuss them.

First we define the measures for a transaction, then we generalize the definitions to a locking protocol, and show how to take the database structure into account.

For a transaction $T$ the execution length, denoted $El(T)$, is the number of vertices on the longest path in $T$. The message path of $T$, $Mp(T)$, is the maximum number of intersite arcs (messages) on a path in $T$. The total number of messages in $T$, $Nm(T)$, is the number of intersite arcs in $T$.

As mentioned, the transaction represents an execution of programs. For an execution the value of each measure is dependent of three factors: 1) the number of
entities referenced and their distribution among sites; 2) the semantics of the programs; 3) the locking protocol. In order to isolate the locking protocol overhead we fix a set of entities, and consider the minimum value for all transactions of the protocol referencing it.

Formally, for a protocol \( P \) and a set of entities \( X \) we denote by \( P(X) \) the set of transactions which belong to \( P \) and reference exactly the set of entities \( X \). We define the \textit{execution-length-overhead} of protocol \( P \) on set of entities \( X \), \( \text{EL} \cdot P(X) \), to be \( \min_{T \in P(X)} \{ \text{EL}(T) \} \). The intuitive meaning is that any set of programs which obeys the rules of the protocol and accesses exactly \( X \) during an execution, will have at least \( \text{EL} \cdot P(X) \) Lock-Unlock steps to execute sequentially. Different program semantics and different input parameters may produce transactions of \( P(X) \) with longer critical paths, but \( \text{EL} \cdot P(X) \) is the longest path required by the protocol, thus no transaction in \( P(X) \) can have a shorter longest path. Similarly, we define \( \text{Mp} \cdot P(X) = \min_{T \in P(X)} \{ \text{Mp}(T) \} \) and \( \text{Nm} \cdot P(X) = \min_{T \in P(X)} \{ \text{Nm}(T) \} \). They are the message-path-overhead and message-number-overhead respectively.

Now consider the protocols we defined. For 2PL the previous definitions suffice, but the structured protocols constitute of different sets for different structures; for example, if \( t_1 \neq t_2 \) then \( TP_{t_1} \neq TP_{t_2} \). Therefore, given a fixed distribution of the database entities, each possible structure defines a value for each measure (although different structures which are identical when restricted to \( X \) have the same value on \( P(X) \)). In other words, a structured protocol has a set of values for each measure. We are interested in the minimal and maximal value in each set, and based on these values we define best and worst structures. For example, \( \max_i \text{Nm}.TP_i(X) \) is the maximal number of intersite messages required by the protocol regardless of the particular tree structure; i.e., there is no database tree for which \textit{every} transaction referencing \( X \) and obeying the protocol needs more messages. We call a tree \( t \) for which this maximum is obtained a \textit{worst}
tree by the $Nm$ measure. Similarly we define a worst tree by the other two measures. A worst sequence of database entities for DF2PL by a measure $M$ is a sequence $p$ for which $\max_p M.DF2PL_p(X)$ is obtained.

Additionally, $\min_t Nm.\tau p_t(X)$ is the minimal number of messages required by the protocol regardless of the tree structure. A tree $t$ for which this minimum is obtained is a best, or optimal, tree by the $Nm$ measure. Similarly a best tree is defined by the other two measures, and a best sequence is defined for DF2PL by the three measures.

In the course of proving our results in the next section, we demonstrate for DF2PL a structure which is best by all three measures, and one which is worst by all three measures. We do the same for the tree protocol, with one exception. Our worst structure by the $El$ measure is different from the worst structure by the other measures. It can be demonstrated by an example that these structures are different in the general case.

4. RESULTS

In this section we first concisely state all our results (TABLE 1 below) then we prove each one. The entries in the results table are numbered 1-15. Each entry corresponds to an overhead-measure of a locking protocol and provides a value w.r.t. a fixed set of entities $X$. The value is provided as a function of the distribution parameters of $X$. The distribution parameters of $X$ are $k_1, \ldots, k_s$ for $s > 1$, meaning that $k_1$ entities of $X$ reside at site one, $k_2$ reside at site two, etc.
Next we will prove our results by proving each entry in Table 1. The proof of an entry \( i \) corresponding to protocol \( P \) and measure \( M \) consists of two parts: existence and bound. The existence proof for 2PL demonstrates a transaction of \( 2PL(X) \) for which the value of \( M \) is the one given in the entry. For the structured protocols it demonstrates a structure \( st \) and a transaction in the protocol structured by \( st \).

The bound proof for 2PL shows that the value of \( M \) for any transaction in \( 2PL(X) \) is at least the one given in the entry. If \( P \) is a structured protocol and the entry corresponds to \( \text{min} \ M_P \), then the proof shows that the value of \( M \) for any transaction in \( P(X) \) is at least the one given in the entry, regardless of the structure on \( X \).

If the entry corresponds to \( \text{max} \ M_P \), then the proof shows two things. First, that there does not exist a transaction \( T \) in the protocol structured by \( st \) of the existence proof, such that \( M(T) \) is lower than the value in the entry. Second, that
for any structure \( d \) there exists a transaction \( T \) in the protocol structured by \( d \), such that \( M(T) \) is the value given in the entry or less. In other words, we show that there exists a structure for which the protocol requires that the value of the measure be at least the one given in the entry; and the value in the entry is sufficient for a transaction to obey the protocol, regardless of the structure. For example, for entry 15, this proves the following. There exists a tree structure which necessitates \( A+s-1 \) intersite messages of any transaction which references the set of entities \( X \) and obeys the tree protocol. Also, \( A+s-1 \) intersite messages suffice for a transaction referencing the set \( X \) to obey the protocol regardless of the particular tree structure of the database entities.

We prove the entries sequentially by their number and start with some notation. Let \( T^0 \) be some transaction in \( 2PL(X) \) which has an intersite arc from the last lock at every site to the first unlock at each other site, and no other intersite arcs (Fig. 1(a)). Note that there is a whole subset of \( 2PL(X) \) with the specified intersite arcs, one for each order of locks and unlocks at each site (that is the reason we defined \( T^0 \) as "some" transaction in \( 2PL(X) \)).

**Proof of Entry Number 1. (Existence)** The transaction is \( T^0 \).

(Bound): By definition, any transaction referencing \( X \) has a path of length \( 2\max k_i \).

**Proof of Entry Number 2: (Existence)** The transaction is \( T^0 \).

(Bound). Any transaction of \( 2PL(X) \) has at least one intersite arc. []

Let \( p^+ \) be some total order of the database entities in which every entity at site \( i \) precede all entities at site \( i+1 \) for \( i = 1, \ldots, s-1 \). We will show later that \( p^+ \) is a best structure for \( DF2PL_i \); at this point we need it for the following definition. Let \( T^+ \) be some transaction in \( DF2PL_{p^+}(X) \) which has exactly the following intersite arcs:
(1) from the last lock at site \( i \) to the first lock at site \( i+1 \) for \( i = 1, \ldots, s-1 \);

(2) from the last lock at site \( s \) to the first unlock at site \( i \) for \( i = 1, \ldots, s-1 \).

\( T^+ \) has the format of Fig. 1(b). Clearly, there exists at least one such transaction in \( DF2PL_p^*(X) \). There can be many transactions of \( DF2PL_p^*(X) \) with the specified intersite arcs, each unlocking entities in a different partial order.

**Proof of Entry Number 3:** *(Existence)* The transaction is \( T^+ \).

*(Bound)*. Assume that \( T \in 2PL(X) \). Let \( T \) have \( l \) unlock steps of which none succeeds another unlock step; and \( m \) lock steps of which none precedes another lock step. Clearly \( s \geq l, m \geq 1 \). Let \( r_1, \ldots, r_s \) be the first unlock steps executed at sites \( 1, \ldots, s \) respectively. \( s-l \) of these steps are preceded by an unlock step executing at a different site. Thus, the number of intersite arcs between two unlocks in \( T \) is at least \( s-l \). Similarly, it can be shown that the number of intersite arcs between two locks is at least \( s-m \). Since all locks precede all unlocks there are at least \( l+m-min(l,m) \) intersite arcs from locks to unlocks. Thus the total number of intersite arcs is at least \( 2s-l-m+l\cdot m-min(l,m) \). It is easy to see that 

\[
\sum_{i} k_i \geq 2s-l-m+max_{i} k_i \]

therefore \( T \) has at least \( 2s-2 \) intersite arcs. 

Note that a transaction of \( 2PL(X) \) with the same format obtains the value for the \( E_l \) and \( M_p \) measures. A different-format transaction obtains the value for the \( N_m \) measure. This indicates the following tradeoff for \( 2PL \). Programs can reduce the total number of messages if they increase the longest sequence of messages, and the longest sequence of steps (and vice versa).

**Proof of Entry Number 4:** *(Existence)* The structure is \( p^* \) and the transaction is \( T^+ \).

*(Bound)*. Regardless of the total order of database entities, \( p \), if \( T \in DF2PL_p(X) \) then \( T \) has \( \sum_{i} k_i \) totally ordered locks which precede \( max_{i} k_i \) totally ordered unlocks. Thus \( E_l(T) \geq \sum_{i} k_i + max_{i} k_i \).
Proof of Entry Number 5: (Existence) The structure is \( p^+ \) and the transaction is \( T^+ \).

(Bound): Let \( p \) be some total order of the database entities and \( T \in DF2PL_p(X) \). Assume w.l.g. that the first lock that \( T \) executes is at site 1. Denote by \( d_i \) the first lock executed by \( T \) at site \( i \) for \( i = 2, ..., s \). Since all locks are totally ordered each \( d_i \) has an intersite arc incoming from the lock which immediately precedes it in \( T \). Denote this intersite arc entering \( d_i \) by \( l_i \). Also, denote by \( m \) a site different than the site at which \( T \) executes its last lock. There is an intersite arc, \( l_{s+1} \) entering the first unlock at site \( m \). \( l_2, ..., l_s, l_{s+1} \) are on a path. Thus \( Mp(T) \geq s \). []

Proof of Entry Number 6: (Existence) The structure is \( p^+ \) and the transaction is \( T^+ \).

(Bound): If \( p \) is a total order of database entities and \( T \in DF2PL_p(X) \) then \( T \in 2PL(X) \) thus the proof follows from entry number 3. []

Before proceeding we need the following preliminaries. For the set of database entities \( X \), a sequence on \( X \) is a total ordering of the entities of \( X \). A site alternation of a sequence on \( X \) is a pair of adjacent entities which reside at different sites. We are interested in a sequence on \( X \) which has the maximum number of site alternations, and will prove that algorithm MAXALTER (Fig. 3) builds such a sequence. Given \( X \), we denote the output-sequence of the algorithm by \( p^-(X) \). We will show that it is a worst sequence for \( DF2PL(X) \) by all measures. It is also a worst tree for \( TP(X) \) by the \( Mp \) and \( Nmv \) measures.

The algorithm MAXALTER assumes w.l.g. that the distribution parameters of \( X \) are \( k_1, ..., k_s \). It builds \( p^-(X) \) sequentially by appending one entity at a time at the end of \( p^-(X) \), and removing it from \( X \). It is done at steps 3, 5 or 8 of MAXALTER. Note that MAXALTER is nondeterministic in the sense that at these steps we just append an entity from some site, without specifying which one. However, this is immaterial, because for our purposes only the residence site of the entity matters. Intuitively, MAXALTER works as follows. While possible, a site alternation is created each time an entity is appended. The attempt is to leave \( X \) with an
equal number of entities at sites 1 and 2 (loop at steps 1-4). If successful, the
second loop (steps 6-10) is performed. Otherwise, it means that \( k_1 > \sum_{i=2}^{s} k_i \) and the
algorithm ends after performing step 5. Only at step 5 an entity can be appended
without creating a new site alteration of \( p^{-1}(X) \).

MAXALTER(X);
1. Do while \( X \) has some entities at site 2;
2. If \( X \) has the same number of entities at sites 1 and 2 then go to 6;
3. Append at the end of \( p^{-1}(X) \) (and remove from \( X \)) an entity from site 1,
then an entity from the highest site at which \( X \) has entities;
4. end;
5. Append at the end of \( p^{-1}(X) \) all entities left at site 1 in an
arbitrary order; quit;
6. Do while \( X \) is not empty;
7. For \( i=1 \) to \( s \) do;
8. If \( X \) has some entities at site \( i \) then append at the end
of \( p^{-1}(X) \) (and remove from \( X \)) an entity from site \( i \);
9. end;
10. end;

Figure 3 - Algorithm MAXALTER totally orders a set of input entities \( X \) into a
sequence \( p^{-1}(X) \). \( k_i \) entities of \( X \) reside at site \( i \) for \( 1 \leq i \leq s \), and we as-
sume \( k_1 \geq \ldots \geq k_s \).

Lemma 1: For a set of entities \( X \) with distribution parameters \( k_1 \geq \ldots \geq k_s \) the
number of site-alternations of \( p^{-1}(X) \) is \( \min\{2(\sum_{i} k_i - \max k_i), \sum_{i} k_i - 1\} \).

Proof: In algorithm MAXALTER each execution of step 3 creates two site alterna-
tions (except the first one, of course, which creates only one). Assume that
\( k_1 > \sum_{i=2}^{s} k_i \), or in other words \( 2(\sum_{i} k_i - \max k_i) \leq \sum_{i} k_i - 1 \). Then step 3 will be executed
\( \sum_{i=2}^{s} k_i \) times and the goto at step 2 will never be performed. The algorithm will end
at step 5, which will create one more site alternation. Thus the total number of site alternations of $p^{-}(X)$ will be $2(\sum k_i - \max k_i)$.

Assume now that $k_1 \leq \sum_{i=2}^{n} k_i$. Then entities are appended to $p^{-}(X)$ only at steps 3 and 8 and at these steps the appendix of each entity creates one site alternation. Thus the total number of site alternations in $p^{-}(X)$ is equal to the number of entities in $X$, minus 1. []

**Lemma 2:** For a set of entities $X$ with distribution parameters $k_{i_{1}} \geq \ldots \geq k_{2}$, $p^{-}(X)$ has the maximum number of site alternations among all sequences on $X$.

**Proof:** Obviously a sequence $g$ on $X$ has at most $\sum k_i - 1$ site alternations. Assume now that $k_1 > \sum_{i=2}^{n} k_i$. Then at most $\sum_{i=2}^{n} k_i$ entities residing at site 1 can be immediately preceded in $g$ by an entity which does not reside at site 1. Thus the total number of entities which can be immediately preceded by an entity residing at a different site is at most $2\sum_{i=2}^{n} k_i$. []

Given a total order $p$ of the database entities, denote by $p(X)$ the subsequence of $p$ consisting of the entities of $X$. Also, denote by $T(p)$ some transaction in $DF2PL_p(X)$ which has exactly the following intersite arcs:

1) From $Lx$ to $Ly$ if $x, y$ is a site alternation in $p(X)$.

2) From the last lock executed by $T(p)$ to each first unlock it executes at a different site (than the last lock).

Again, $T(p)$ has the format of Fig. 1(b). Clearly, there exists at least one such transaction in $DF2PL_p(X)$.

**Observation 1:** $Mp(T(p))$ is equal to the number of site alternations of $p(X)$ plus one.

**Observation 2:** For any transaction in $DF2PL_p(X)$ the $Mp$ measure is at least
Denote by $p^-$ some total order of the database entities which has $p^-(X)$ as a subsequence.

**Proof of Entry Number 7: (Existence)** The structure is $p^-$ and the transaction is $T(p^-)$.

**(Bound):** By entry number 4, there is no transaction in $DF2PL_p(X)$ with a lower $EI$ value. For any total order of database entities $p$ and any $T \in DF2PL_p(X)$ in which there are no intersite arcs between two unlocks $EI(T) = \sum_{i} k_i + \max k_i$. []

**Proof of Entry Number 8: (Existence)** The structure is $p^-$ and the transaction is $T(p^-)$. The proof follows from Lemma 1 and Observation 1.

**(Bound):** The proof follows from Lemma 2 and Observations 1 and 2. []

**Observation 3:** $Nm(T(p))$ is equal to the number of site alternations of $p(X)$ plus $s-1$.

**Observation 4:** For any transaction in $DF2PL_p(X)$ the $Nm$ measure is at least $Nm(T(p))$.

**Proof of Entry Number 9: (Existence)** The structure is $p^-$ and the transaction is $T(p^-)$. The proof follows from Lemma 1 and Observation 3.

**(Bound):** The proof follows from Lemma 2 and Observations 3 and 4. []

For the next proofs we assume again that $k_1 \geq \ldots \geq k_s$ and we structure the set of database entities $X$ as a directed tree $t$ as follows. Arbitrarily structure the entities at site $i$, as a directed tree rooted at $r_i$ for $i=1,\ldots,s$. Build $t$ by making $r_2,\ldots,r_s$ the sons of $r_1$. $t$ is illustrated in Fig. 4. Denote by $t^+$ a directed tree structure of the database entities, of which $t$ is a subtree. We will show that $t^+$ is a best structure for $TP_t(X)$ by all measures. Let $F$ be some transaction of $TP_t(X)$ which has exactly the following intersite arcs: from $lr_1$ to $lr_i$ and from $lr_i$ to $ur_1$.
for \( i = 2, \ldots, s \). \( F \) is illustrated in Fig. 5.

![Diagram](image)

**Figure 4:** Optimal structuring of a set of entities into a directed tree.

![Diagram](image)

**Figure 5:** Optimal transaction obeying the tree protocol, by the \( El \) and \( Mp \) measures.

**Proof of Entry Number 10:** First we will clarify that the precise value for the entry is: \( 2\text{max}_k + 1 \) if \( X \) has \( \text{max}_k \) entities at two or more sites and \( 2\text{max}_k \) otherwise. The imprecision in the results-table is made for the sake of conciseness.

**Existence:** The tree is \( t^* \) and the transaction is \( F \).

**Bound:** Trivial if \( X \) has \( \text{max}_k \) entities at one site only. Otherwise, let \( T \in TP_t(X) \) for some tree \( t \). There exists a site at which \( T \) executes \( 2\text{max}_k \) steps at which the root of the tree referenced by \( T \) does not reside. The first lock executed by \( T \) at that site is preceded by another step, thus \( El(T) \geq 2\text{max}_k + 1 \).
Proof of Entry Number 11: (Existence) The tree is $t^+$ and the transaction is $F$.

(Bound): Let $T \in TP_t(X)$ for some tree $t$. Consider $Ly$, the first lock executed by $T$ at site at which the root of the subtree referenced by $T$ does not reside. Since the father of $y$ resides at a different site there exists an intersite arc entering $Ly$ and an intersite arc exiting $Ly$ or a step succeeding $Ly$. These two arcs are on a path. []

Let $G$ be some transaction of $TP_{t^+}(X)$ which has exactly the following intersite arcs: from $Lr_1$ to $Lr_{i+1}$ for $i=1,...,s-1$ and from $Lr_s$ to $Ur_1$ (Fig. 6).

![Figure 6: Optimal transaction obeying the tree protocol, by the Nm measure.](image)

Proof of Entry Number 12: (Existence) The tree is $t^+$ and the transaction is $G$.

(Bound): Consider some transaction $T \in TP_t(X)$ for some tree $t$. The unlock of the root of the subtree on $X$ (or a predecessor executing at the same site) must have an entering intersite arc. The first lock at any other site must also have an entering intersite arc. Thus $T$ has at least $s$ intersite arcs. []

Proof of Entry Number 13: (Existence) Assume w.l.g. that $\max_i$ entities of $X$ reside at site $s$. The structure is some total order in which every entity at site $i$ precedes all entities at site $i+1$, for $i = 1,...,s-1$. Denote it by $t^-$. We show that it is worst by the $El$ measure. The transaction from $TP_{t^-}(X)$ is one with the format of Fig. 1(b) (formally, $T(t^-)$).

(Bound): If $T \in TP_{t^-}(X)$ then the first lock executed by $T$ at site $s$ precedes
2\max_{i}k_i - 1 \text{ totally ordered steps, and is preceded by } \sum_{i}k_i - \max_{i}k_i \text{ totally ordered lock steps, thus } E_l(T) = \sum_{i}k_i + \max_{i}k_i. \text{ For an arbitrary tree } t, \text{ there exists a transaction with the format of Fig. 1(b) in } TP_t(X); \text{ its } E_l \text{ measure is } \sum_{i}k_i + \max_{i}k_i. \]
5. DISCUSSION

In this paper we introduced three measures of the overhead imposed by a locking protocol on transactions in a distributed database. The first one quantifies the longest sequence of necessary protocol steps $(E_1)$; the second, the longest sequence of necessary intersite messages $(M_0)$; and the third, the total number of necessary intersite messages $(N_0)$. Then we analyzed three important protocols by each measure: two-phase-locking (2PL), and the structured deadlock-free-two-phase-locking (DF2PL) and the tree-protocol (TP). The analysis provides for each pair (locking-protocol, measure) a lower bound on the overhead, for any transaction obeying the protocol. For the structured protocols there exists a range of such lower bounds, depending on the particular structure imposed on the database entities. The upper and lower bounds of this range are presented. All bounds are tight.

5.1 Tradeoffs

The measures we provide quantify some intuitive tradeoffs. For example, it is intuitive that the more "spread-out" the set of entities referenced by a transaction, the better its parallelism (i.e., lower overhead by the $E_1$ measure); however this increases the number of required intersite messages (i.e., higher overhead by the $N_0$ measure). The results point out some tradeoffs which are not so obvious. For example, assume that a set of programs implement transactions obeying the 2PL protocol. The programs can be designed to decrease the total number of intersite messages at the expense of an increased number of sequential intersite messages, and a lower transaction parallelism (and vice versa). Specifically, if locking of the entities is done in parallel at the different sites, then the total number of intersite messages required increases. On the other hand, if the programs are designed to obtain all locks at one site before starting to issue lock requests at another site, then the total number of intersite messages required
decreases. Furthermore, the programs can be designed to dynamically adjust to changing conditions by decreasing overhead according to one measure, while increasing it according to another. Since the different measures are appropriate for different environments, the strategy may depend on what the bottleneck in overall system-performance is at a given moment, and the priority of the task at hand. Similar tradeoffs exist for programs implementing transactions which obey the tree protocol.

5.2 Comparison of Protocols

2PL is at least as good as the structured protocols with the optimal structure, by the $E_l$ and $M_p$ measures, but TP is better by the $N_m$ measure. For worst structures DF2PL and TP are identical, and cause a greater overhead than 2PL, by all three measures. The difference in overhead between 2PL and the structured protocols quantifies the cost of deadlock-freedom. For optimal structures, TP is better than DF2PL by all three measures. The difference in overhead between TP and DF2PL represents the cost of the ability to reference any subset of entities under DF2PL (TP requires that the set of entities referenced by a transaction be a subtree). It is also interesting to observe that for DF2PL with the optimal total order of entities, and for 2PL, the $N_m$ measures are equal; thus, deadlock freedom comes at no additional cost in this case.

5.3 Database Structuring

Assuming a fixed entity-distribution and no a priori information concerning access patterns of transactions, the analysis indicates the following. The least overhead is caused by the tree protocol if at each site resides a whole subtree, with the entities at different sites incomparable in the partial order defined by the tree. Paths of the tree which have site alternations (i.e., consecutive entities residing at different sites) increase the overhead by the $M_p$ and $N_m$ measures. Paths in which a sequence at one site precedes a sequence at another site
increase the overhead by the $E_l$ measure, and the worst database structure is a path in which all entities at one site precede all entities at another site. The worst database structure by the $M_p$ and $N_m$ measures is a path with a maximum number of site alternations. It is a worst sequence for DF2PL as well. However, transactions obeying DF2PL can reference any subset of database entities thus a bad structure is not as severe as for the tree protocol. Namely, even for a worst structure, there exist transactions obeying DF2PL which reference a subsequence of entities with no site alternations. The structure has no effect on DF2PL by the $E_l$ measure, and by the other measures an optimal sequence is one in which all entities at one site precede all entities at another site.

A similar analysis can be carried out when the structure is fixed and the entity distribution varies.

5.4 What Factors Determine Overhead?

Protocol-overhead by the $E_l$ measure is a function of the maximum number of entities referenced by the transaction at a site. Depending on the structure, it may also be determined by the total number of referenced entities. Overhead by the $M_p$ measure is fixed (optimal case), or a function of the number of referenced sites, or a function of the number of referenced entities (worst case). Overhead by the $N_m$ measure is a function of the number of referenced sites or a function of the number of referenced entities.

ACKNOWLEDGEMENT

I wish to thank Ephraim Korach for helping establish entries 3 and 8 in the results-table and Shimon Even for proposing the execution-length measure.
REFERENCES


[KS3] Z. Kedem and A. Silberschatz, "Locking Protocols: From Exclusive to...


