ELECTRONIC WORKSHOPS IN COMPUTING

Series edited by Professor C.J. van Rijsbergen

B Novikov, University of St Petersburg, Russia and J W Schmidt, University of Hamburg, Germany. (Eds)

Advances in Databases and Information Systems, Moscow 1996

Proceedings of the International Workshop on Advances in Databases and Information Systems (ADBIS '96). Moscow, 10-13 September 1996

Key Integrity for Cooperative Database Environments with Stationary and Mobile Hosts

Vadim V. Doubrovski

Published in collaboration with the
British Computer Society

©Copyright in this paper belongs to the author(s)

Key Integrity for Cooperative Database Environments with Stationary and Mobile Hosts

Vadim V. Doubrovski
Department of Computer Science
RMIT*University, Melbourne
GPO Box 2476V, Melbourne
Victoria 3001, Australia
E-mail: vadim@cs.rmit.edu.au

Abstract

ACID properties and serialisability have been extensively studied for transactions that are submitted concurrently in a multithread database environment. A number of protocols have been proposed to ensure ACID properties for centralised and distributed databases. These protocols aim to ensure consistency of update operations. However, they do not address key integrity problems that arise in a heterogeneous multidatabase environment when some of the hosts in the multidatabase have no constant communication links which is a frequently found situation for environments with mobile hosts. Transactions in such environments have longer life time which complicates transaction management and replica control protocols.

In this paper we propose to use semantic information from the conceptual schema of the MDBS to ensure key integrity. Transaction handling and reconciliation protocols for entities with natural and surrogate primary keys are proposed and the description of the delayed commit protocol is presented.

1 Introduction

Transaction management for centralised and distributed databases ensures that all transactions in a multithread database environment are provided with ACID transactional guarantees [8, 16]. ACID guarantees are achieved if the results of all reads and writes are consistent and do not breach data integrity. This is based on the idea that a schedule of read/write operations for concurrent transactions should be equivalent to some serial schedule, when transactions are scheduled one at a time, which is known as the property of serialisability [7]. This property and the means of ensuring thereof has been studied quite extensively.

Cooperative and federated database environments known as multidatabase systems (MDBS) [10, 15, 14, 20] impose new challenges upon the transaction management mechanism. An MDBS environment represents a loose integration of several database systems. In such an environment both global applications whose transactions span more than one participating databases at once, and local applications that issue transactions that affect only one underlying database, must be supported. It has been shown that ACID properties, and global serialisability and atomicity in particular, are self-guaranteed if all local database systems (LDBS) ensure commit-order serialisability [2, 17] and do participate in a global two-phase commit. For global transactions this assumes use of the nested transaction management mechanism.

However, scheduling that ensures strict serialisability, usually in practice causes severe performance problems for a regular MDBS. Indeed, the *closed nested transaction management mechanism* [11] does not allow subtransactions to commit before the whole global transaction reaches the last, so called *ready-to-commit* state [3]. The commit only can begin when the longest subtransaction from the series has acknowledged that it has finished and is ready to

*R. M. I. T. - Royal Melbourne Institute of Technology

commit. Therefore, all locks placed to ensure serialisability (even for the shortest transaction in the series) cannot be released until the longest subtransaction finishes.

Many authors have argued that serialisability might be too strong of a requirement in an MDBS environment. Thus, a concept of *quasi serialisability* has been introduced [5]. Another concept of *open nested transaction management mechanism* [1, 8] is seen as a means of increasing efficiency of handling concurrent transactions while providing sufficient guarantees for data consistency and integrity. This is achieved via reducing the probability of subtransactions conflicts by allowing a subtransaction to commit as soon as possible after it has completed all its read and write operations. *Semantic-based transaction management* [4] based on open transaction management mechanism is proposed in order to speed up the MDBS performance further more by eliminating pseudo-conflicts through exploiting the semantic properties of subtransactions to assess the possibilities of a conflict to appear.

1.1 MDBS with mobile hosts

Cooperative heterogeneous database environments with mobile hosts represent a class of federated databases with no constant on-line communication links between component databases. This assumes certain restrictions in accessing the component LDBSs.

There is less of a problem when a constant on-line link between such mobile hosts and stationary networked nodes exists as although the mobile nature of the nodes adds to the complexity of the problem and has its impact on the way such an environment operates, but still, it resembles a conventional MDBS, whereby each of the nodes most of the time has an on-line connection to the rest of the MDBS.

When there is no possibility to have each of such mobile hosts constantly connected to the rest of the MDBS because of various problems (e.g. higher costs of communication link that is not used completely, location outside of the area of coverage for mobile phones, etc), additional problems in maintaining MDBS integrity may arise. There is no longer the possibility to enforce complete integrity, controlling the execution of a distributed transaction. Certain design decisions, transaction and replica control protocols must be put in place in order to ensure MDBS integrity.

There are certain conflicting operational requirements that such an MDBS environment should satisfy. Among them are the following:

- The requirement for stationary and mobile hosts running LDBSs to preserve their autonomy.
- Support for distributed transactions as well as local transactions at each host.
- Provision of certain level of transparency for global transactions where the mobile nature of some hosts is hidden.
- Correct handling of off-line periods with no communication link for the rest of the MDBS as well as tolerance to network failures.
- Periodical broadcasting database updates for both mobile and stationary hosts.

There are some approaches and protocols proposed to handle transaction processing for an MDBS with mobile hosts [9, 23] that address specific features of mobile computing.

1.2 Key integrity in an MDB system

Serialisability as a means of enforcing ACID properties, and special transaction management protocols proposed for MDB systems, ensure data consistency when data is being accessed and updated by a number of concurrent transactions. Hence, the correctness of protocols has been the main theme of research concerned with transaction management for MDBSs.

However, when it comes to consideration of data integrity for heterogeneous MDB systems with no constant communication link, the integrity of primary keys, maintaining of referential integrity of primary key - foreign key links for databases supporting relational data model, becomes an important topic of research. Since transactions in such an environment can have longer life time because of the necessity to wait for periodical on-line connections that

are established from time to time, special *replica control* or *reconciliation* protocols need to underlie the transaction management mechanism. Clearly, in order to solve the key integrity problems, semantics found in conceptual and logical models of the MDBS must be used.

This paper makes the following contribution:

- It studies how the semantics extracted from the conceptual model can be used to enforce key integrity.
- It studies the types and semantics of entity keys from the data modelling point of view and explains how these affect the length of a transaction life and type of reconciliation protocol needed.
- It provides two models for reconciliation for entities with natural and surrogate primary keys.
- It introduces the *delayed commit protocol* to resolve the key integrity problem for entities with surrogate primary keys.

The structure of the paper is as follows. Section 2 discusses the class of MDBSs being the subject of enquiry for this paper, the model thereof and main assumptions. Section 3 is devoted to the types of update operations that a mobile host can initiate in an MDBS. Reconciliation protocol for entities with a natural primary key and delayed commit protocol are discussed in section 4. The last section of the paper represents conclusions.

2 The Model and Assumptions

This section sets out the framework for the study and describes the class of multidatabase systems being considered for the rest of the paper. We describe assumptions regarding the topology of the multidatabase system and logical schema thereof. For the sake of clarity, we assume that all LDB systems at each node have the same type of data model and this being the relational model. This slight simplification is still fairly close to the reality as the relational model can in most cases be considered as a reasonable common ground for generalisation of the global database schema for a heterogeneous MDBS.

Our assumptions and study encompass two levels of abstraction, namely - **conceptual level** where we consider modelling entity types (referred to as just *entities* further on in the paper) and entity instances, and **logical level** where relations and their tuples are considered, and how the tuples participate in transaction and reconciliation protocols.

We assume that the *primary site* technique that is based on the idea of *distinguished copy* [12, 13], is used, and it is used over particular type of network topology being the star topology (Fig. 1a). In such a case the central node is the primary site that maintains the primary copy of the MDBS. This assumption can be easily extended to the case of a network with non-mobile hosts (constantly on-line ones) replacing single central node (Fig. 1b).

Star topology for cooperative database environments is very popular when it comes to designing database systems for industry, whereby a headquarter controls operation of its branches, or when there is a central company and a number of agents that operate for the company. It is frequently found these days that such an agent or a regional host is mobile and has certain level of autonomy.

This assumption is very important for the study as while designing reconciliation protocols we shall assume that there is always at most one non constant communication link between a mobile host and the central site.

There are two types of transaction managers that run on different hosts of the MDBS. Each host runs its own *local transaction manager (LTM)* that is used either to access the LDBS autonomously or to accomplish subtransactions of a global transaction. Global transactions are handled by the *global transaction manager (GTM)* that can run on the primary host.

Generally, vertical fragmentation takes place between the central host and all or some of the regional hosts, and join is used for re-constructing full relations. In other words, let $A_i^c \in R^c$ ($i \in [1..n_c]$, where n_c is the degree of relation R^c) be attributes of a relation schema R^c being part of the local database schema for the central node, and $A_i^{m_j} \in R^{m_j}$ - those for the relation R within the j^{th} mobile host of the system. Then

$$\forall k \in [1..m] \Rightarrow A_k \in \{A_i^c\} \cap \{A_i^{m_j}\}$$

and $\{A_k\} = PK$ is the primary key of the relation.

Figure 1: Examples of network star topology with mobile hosts

Horizontal fragmentation is used to logically reflect on the distributed nature of the entity that is mapped to the relation R . That means, in turn, that corresponding copies of the relation R have either the same relation schemas or at least have compatible schemas for their primary keys. Outer union over all regional hosts is used to re-construct the regional part of the relation R :

$$r^m(R^m) = \bigcup_{i=1}^n r^{m_i}(R^{m_i}),$$

where $r(R)$ denotes the relation instance of R . No horizontal replication occurs between the regional nodes of the database, i.e.

$$\forall i, j \in [1..m] \Rightarrow r^{m_i}(R^{m_i}) \cap r^{m_j}(R^{m_j}) = \emptyset$$

We shall distinguish between two types of primary keys for entities whose corresponding relations are kept in the database. By *natural primary key (NPK)* we understand a set of attributes that (i) uniquely identifies an entity instance and (ii) retains its meaning beyond the context of the database, application, or data model thereof. Examples of an NPK can include such attributes as Social Security No, Driver's Licence No, Product ID, Part No, etc. From a strict data modelling point of view not all of them can be classified as natural primary keys as some of them are in fact former surrogate keys (see further) that have become natural keys with time. However, due to their relative stability they can be seen as completely natural. Clearly, it is possible that NPK can be either simple or composite. The existence of a natural primary key is the guarantee that there is a bijection between an entity instance (as a physical real world entity) and a tuple from the corresponding relation in the database.

Surrogate (artificial) primary key [6, 18, 19] is a key that has no explicit meaning and is frequently assigned a value by the transaction manager or the database engine. An example of an attribute that is not a natural key, could be Transaction Number for a bank transaction. This, in fact, is a surrogate key whose value does not have any explicit meaning beyond the scope of the corporate data model for the bank in question.

We shall assume that transactions are queued on the mobile host until the communication link is re-established and reconciliation begins.

3 Update Operations For An MDBS System With Mobile Hosts

A rigid approach to transaction management for an MDBS with features described in the previous section, dictates that in accordance with the *isolation* property, any update initiated from a mobile host should be first made invisible for the whole MDBS including the host of its origin. The transaction should be deemed committed and the result of the update made visible when the subtransaction that updates the primary copy successfully commits. Then the MDBS propagates the update through the other hosts according to its replica control protocol. This would mean that the result of the update is unavailable at least until the time of re-establishing a communication link, that may be totally impractical and may have serious impact on the performance of the MDBS. However, as we shall show further, immediate commit of the local subtransaction may create key integrity problems within the MDBS. Our aim is to devise such reconciliation protocols that would allow to relax the rigid requirements making the results of a transaction available as soon as possible, still avoiding, however, key integrity problems.

In order to reveal problems that update operations may introduce, and observe the impact they may have on the key integrity over the whole MDBS, let's consider different types of updates and the ways they affect reconciliation of an LDBS that resides on a mobile host, and the rest of the MDBS. First, assume that there is an entity RE within the conceptual schema of the cooperative database that is mapped to a relation R within the logical schema, that satisfies the criteria represented in the previous section. Then the following updates are possible:

- *modification* of the value for any attribute $A_i^{m_j} \in R^{m_j}$ of a tuple of R where $A_i^{m_j} \notin PK$. This might create no integrity problems if the attribute $A_i^{m_j}$ is also not part of any referential integrity constraints, and no pre-update triggers or RPC are associated with it. If these conditions are met, the corresponding local transaction can be executed straightaway while the rest of the global distributed transaction can be queued for submission to the GTM at a later stage when communication link is restored. The results of the local transaction can be committed immediately, thus, being made available for local use provided there are no constraints that are violated within the global part of the transaction. If there are those that are violated, compensating techniques [21, 22] should be used to roll the distributed transaction back.
- *deletion* of any tuple from the relation R . Again, this does not create integrity problems provided there are no referential integrity constraints held across the distributed system, whereby even at least one path that this type of constraint traverses, includes non-permanent communication link. In fact, it is clear that this requirement can be relaxed to the degree of having all such constraints that hold over this type of paths, specified with the cascade delete option. Again, the results of the local transaction can be committed and made available immediately, having the rest of the global transaction queued until connection between the mobile host and the rest of the system is restored.
- *insertion* of a new tuple into the relation R . This should not create problems for the entities with natural primary keys as there is the bijection between entity instances for the entity RE and tuples of the relation instance $r(R)$. However, a reconciliation for the relations that correspond to entities that have immediate and transitive relationships with RE may be required. Totally different situation occurs if RE has a surrogate primary key. The concept of a surrogate key in most cases requires that the system allocates a key value for the newly introduced tuple and this value should be unique. In a homogenous MDBS with on-line communication links it would be possible to solve the problem, having a composite primary key consisting, of the key of RE plus the site number. However, this cannot be achieved in a heterogenous cooperative MDBS. This creates integrity problem straightaway as a mobile node cannot immediately synchronise allocation of a unique primary key with the rest of the distributed system. The transaction cannot be regarded partially committed even locally until a unique primary key is assigned.
- *modification* of the value for any attribute $A_i^{m_j} \in R^{m_j}$ of a tuple where $A_i^{m_j} \in PK$. This either should be prohibited (ie automatically replaced by a deletion/insertion pair) or should be treated the same way as an insertion.

4 Reconciliation Protocols That Enforce Key Integrity

In this section we will concentrate on the insertion problems and show how the strict requirements can be relaxed to make a transaction locally committed before the communication link is restored and before the insertion is submitted to GTM. We also show how the semantics is extracted from the conceptual and logical model of the MDBS can be used in enforcing key integrity. For this purpose we assume that GTM as well as LTM for all mobile hosts have access to some kind of internal representation of the conceptual model (e.g. entity-relationship) for correct reasoning on primary key - foreign key links and propagation of the primary keys of any entity into composite primary keys for other entities connected to it via direct or transitive relationships. Naturally, each LTM can have its own version of the conceptual model, however, all the models must comply with the requirements presented in section 2.

4.1 Entities with natural primary keys

For the sake of simplicity we consider here only the first part of the reconciliation protocol, namely: submission of updates from a mobile host to the distributed transaction manager. When a communication link is established, LTM starts reconciliation using the following protocol.

Step 1. Let RE be an entity with the natural primary key $K = \{A_{1k}, A_{2k}, \dots, A_{nk}\}$. LTM for a mobile host isolates the relation instance $r(R)$ that corresponds to the conceptual level entity RE on the logical level, with the relation schema R , where $K \subset R$. It also searches for relationships between the entity RE and other entities of the conceptual model.

Not all such relationships need to be taken into consideration, however. The only relationships of interest are those through whose schemas the primary key K of R is propagated. Let SE be an entity on the other end of the relationship. Then, only the following relationships need to be considered:

- **1:1** relationships when the following holds for the cardinalities of the two entities: $car(RE) \leq car(SE)$, and perhaps the total participation constraint holds for RE in this relationship;
- **1:M** relationships where the entity RE participates at the '1' end of the relationship;
- all **M:N** relationships.

For all these cases the primary key of RE is normally propagated into the relation schema of SE and forms part of its primary key.

Step 2. LTM issues a query to GTM that runs at the other end of the communication link, to search for a tuple with the same value of the primary key K in the global relation RE . If the tuple does not exist then the reconciliation is simple and the tuple is added to the global relation instance $r(R)$ residing on the central site that maintains the primary copy, and GTM takes care of the fragmentation and replication issues for the rest of the MDBS. If the tuple does exist, then there is no need to insert the tuple again therefore avoiding key violation. The fact that the primary key K is a natural one, guarantees that the two tuples (from both ends of the link) correspond to one and the same entity instance. Reconciliation in this case consists of submitting tuples from all the other relations that were isolated on the previous step for insertion. These relations represent entities that have relationships with RE and their new tuples should be added to the corresponding relations within the primary copy.

4.2 Entities with surrogate primary keys

When no natural key exists for the entity, quite popular solution is to use surrogate keys to ensure uniqueness of a tuple being inserted. Usually, the value of a surrogate key is assigned by the transaction manager or database engine and determined at the time of committing a transaction. As a rule, an MDBS provides certain means for that either via supporting special data types (e.g. autoincrement data type in Borland database engine - Paradox, Delphi) or via special mechanisms provided by the engine (e.g. sequences in Oracle DB engine).

In fact, it is frequently found that on the stage of mapping conceptual models to logical database design, the designer makes the decision to use surrogate keys even though a natural primary key exists. This is done to get rid of a

long sequence of attributes being the natural key for a relation, where those attributes are inherited from its ancestors through a long chain of **1:M** relationships.

Since many of DB engines are ACID compliant and ensure atomicity and isolation for serialisable schedules¹, there is no much of a problem to ensure correct assignment of a surrogate key for a centralised database environment. However, a transaction management mechanism for an MDBS environment that is just ACID compliant cannot ensure key integrity on its own. Hence we propose a delayed commit protocol for entities with surrogate primary keys, that addresses the following two issues:

- immediate local commit and queuing requests to the global manager to propagate the transaction throughout the rest of the MDBS for *modification* and *deletion* types of transactions;
- assigning locally unique values to the primary keys, immediate local commit and queuing requests to the global manager for *insertion* type of transactions with subsequent reconciliation of the values of the primary keys at a later stage.

Following is the strategy that the delayed commit protocol exercises (Fig. 2) for a transaction that is initiated between a mobile and the central host when an *insert* type transaction is being committed.

Step 1. The mobile host H_k is about to commit a record. LTM for this host generates a unique surrogate key for the new tuple and commits the subtransaction locally. A request for a global transaction is put in a queue to the global transaction manager. The local subtransaction is committed, therefore, the new record is now visible for LDBS, however, it is not considered fully globally committed as yet.

Step 2. The second stage of the protocol takes place when the mobile host and the central node of the MDBS initiate a communication session. The mobile host replicates all new records that were successfully committed locally since the last communication session between the two hosts. Each record, when replicated, is assigned a global primary key generated by the global transaction manager. The global transaction that is to propagate changes throughout the whole database is stored in the global transactions queue pending acknowledgment from the local host (see next steps). If this set of actions fails, a local roll back is initiated by the LTM on the mobile host.

Fig. 2. State transition diagram for the delayed commit protocol

Step 3. The next step begins when the central host initiates a new subtransaction on the mobile host H_k that changes the value of the primary key for every previously locally committed record. A separate subtransaction is initiated for every record. Not does the subtransaction encompass only the relation $r(R)$ where the new tuple was inserted in. LTM uses semantic information from the conceptual schema to update the value of $K \subset R$ for all relations that have direct and transitive relationships with R . If this part of the transaction fails, the whole global transaction rolls back to the state before step 2.

If the value of the primary key has been changed successfully, the mobile host acknowledges GTM that the subtransaction has been successfully completed locally. This triggers the global manager to start propagating changes for the rest of the distributed system.

5 Conclusion

We have studied key integrity problem for MDB systems with mobile hosts that do not maintain constant communication links. We have shown that there are two distinct types of integrity problems: the one for entities with natural primary keys and the one for entities with surrogate primary keys. It appears that due to the very nature of natural primary keys the key integrity problem reduces to a reconciliation problem for entities that have direct or transitive relationships with the entity in question. Maintaining integrity for entities with surrogate primary keys however, requires special protocol for insertion type transactions. We have presented the delayed commit protocol that preserves key integrity.

The main feature of the delayed commit protocol is that it uses semantic information from the conceptual model of the MDBS. It requires that this information is made available for LTM for correct propagating of the primary key within LDBS.

¹ There are also many light, mainly PC-based, commercial DB engines that do not comply with ACID properties and do not support the transaction mechanism altogether, but still it is possible to ensure atomicity and isolation using their locking mechanism through custom procedures.

Figure 2: Delayed commit protocol for entities with surrogate keys

References

- [1] Berri C., Bernstein P.A., Goodman N., "A Model for Concurrency in Nested Transaction Systems", *Journal of the ACM*, Vol 36, No. 2, 1989.
- [2] Breitbart, Y., Georgakopoulos, D., Rusinkiewicz, M., Silberschatz, A., "On Rigorous Transaction Scheduling", *IEEE Trans. on Softw. Eng.*, 17(1991), 9.
- [3] Breitbart, Y., Silberschatz, A., Thompson, G., "Reliable Transaction Management in a Multidatabase System". *Proc of ACM SIGMOD Conf.*, 1990, pp.215-224.
- [4] Deacon, A., Schek H.-J., Weikum G., "Semantic-based Multilevel Transaction Management In Federated Systems". *Proc. 10 Int. Conf. of Data Eng. (ICDE-94)*, Houston TX, 1994.
- [5] Du, W., Elmagarmid, A.K., "Quasi Serialisability: a Correctness Criterion for Global Concurrency Control in Interbase". *Proc. of the Intern. Conf. on VLDB*, 1989.
- [6] Elmasri, R., Navathe, S.B., *Fundamentals of Database Systems*, 2nd Edition, Benjamin/Cummings, 1994.

- [7] Gray, J., Lorie, R., Putzulo, G., "Granularity of Locks and Degree of Consistency in a Shared Data Base", in Nijssen, G. (ed) *Modelling in Data Base Management System*, North-Holland, 1976.
- [8] Gray, J., and Reuter, A. *Transaction Processing: Concepts and Techniques*, Morgan Kaufman, 1993.
- [9] Lai, S.J., Zaslavsky, A., Martin, G.M., "A Simulation Model of Adaptive Protocols in Distributed Computing Systems with Mobile Hosts", *28th Annual Simulation Symposium*, Phoenix, Arizona, 1995.
- [10] Larson, J.A., *Database Directions: from relational to distributed, multimedia, and object-oriented database system*. Prentice Hall, Upper Saddle River, NJ, 1995.
- [11] Mass, J.E.B., *Nested Transactions: An Approach to Reliable Distributed Computing*, MIT Press, 1985.
- [12] Menasce, D., Popek, G., Muntz, R. "A Locking Protocol for Resource Coordination in Distributed Databases", *Trans. on Datab. Syst.*, 5:2, June 1980.
- [13] Minoura, T., Weiderhold, G., "Resilient Extended True-Copy Token Scheme for a Distributed Database", *Trans on Softw. Eng.*, 8:3, May 1981.
- [14] Ozsu, M.T., Valduriez, P., "Distributed Data Management: Unsolved Problems and New Issues", in *Readings in Distributed Computing Systems*. IEEE Computer Society Press, 1994.
- [15] Ozsu, M.T., Valduriez, P., *Principles of Distributed Database Systems*, Prentice Hall, Englewood Cliffs, NJ, 1991.
- [16] Papadimitriou, C., *The Theory of Database Concurrency Control*, Computer Science Press, 1986.
- [17] Raz, Y., "The principle of Commitment Ordering", *VLDB Conference* 1992.
- [18] Reingruber, M.C., Gregory, W.W., *The Data Modeling Handbook: a Best-practice Approach to Building Quality Data Models*. John Wiley & Sons, Canada, 1994.
- [19] Simsion, G. *Data Modeling Essentials: analysis, design, and innovation*. Van Nostrand Reinholds, New York, 1994.
- [20] Sheth, A.P., Larson, J., "Federated Database Systems for Managing Distributed Heterogenous and Autonomous Databases", *Computing Surveys*, Vol.22, No 3, Sep 1990, pp. 183-236.
- [21] Schek, H.-J., Weikum, G., Ye, H., "Towards a Unified Theory of Concurrency Control and Recovery", *ACM PODS Conference*, 1993.
- [22] Weikum, G., Schek, H.-J., "Concepts and Applications of Multilevel Transactions and Open Nested Transactions", in A.K. Elmagarmid (Ed.), *Database Transaction Models for Advanced Applications*, Morgan Kaufmann, San Mateo, CA, 1992.
- [23] Yeo, L.H., Zaslavsky, A., "Submission of Transactions from Mobile Workstations in a Cooperative Multidatabase Processing Environment", In *Proc. of the 14th Int. Conf. on Distr. Comp. Systems*, Poznan, Poland. IEEE Computer Society Press, Los Alamos, California, 1994, 372- 379.