Modelling Concurrent Mobile Transactions Execution in Broadcasting Environments

Ahmad Al-Qerem[1] and Walter Hussak[2]
[1]Computer Science Department, Zarqa Private University, Jordan
[2]Department of Computer Science, Loughborough University, UK

Abstract: *Broadcast is an efficient and scalable method for resolving the bandwidth limitation in a wireless environment. There is a trade-off between clients' access time and throughput for update mobile transactions in on-demand data dissemination environments. Data scheduling at the fixed server can allow more transactions to commit while retaining the access time for each transaction. In this paper, we present a data scheduling scheme for both read only and update mobile transactions in pull-based broadcasting environments. Rather than consider access time, which is well studied elsewhere in [1, 2, 3], our concern is to examine the probability that a mobile transaction is able to avoid conflict and commit. Specifically, a set of formulas giving an analysis of this probability is examined. Furthermore, a report of a simulation study for validating these formulas is also provided.*

Keywords: *Wireless broadcast, data organization, mobile transactions.*

Received February 13, 2007; accepted June 8, 2007

1. Introduction

The rapid advances in computer software, computer hardware, and wireless network technologies have led to the widespread implementation of mobile computing. In this environment, users can retrieve information from wireless channels (with generally narrow bandwidth) anytime and anywhere. The problem of how to disseminate data efficiently to a large number of users in the mobile computing environment is challenging due to the necessity to consider time and energy efficiencies, given that mobile devices have limited energy capacities associated with their reliance on battery power. There are two general methods to disseminate data through wireless channels: (1) broadcast (push-based), which enables users to retrieve data by simply listening to a particular channel, and (2) on-demand (pull-based), in which users send requests to get data. Two important factors must be considered in a broadcast-based information system, access time [1, 8, 9, 12, 13, 14] and tuning time [4, 5, 7, 10]. The access time is the time elapsed from the moment a client device submits a query into the broadcast channel to the moment the desired data are acquired. This is the total time a client device must spend and is often used to evaluate the performance of the broadcast system. The tuning time is the time spent by the client listening to the broadcast channel. When the clients are listening to the data in the broadcast channel, the clients are in the active mode. Therefore, the tuning time is often used to evaluate the power consumption of the clients. The aim of our paper is to reduce the access time through intelligent organization of the broadcast data. Many approaches have been proposed to reduce the access time [1, 2, 6, 11]. They can be classified into two categories: uniform data broadcasting [6] and no uniform data broadcasting [1, 2, 11]. [1] proposed the concept of broadcast disk for no uniform data broadcasting. A single broadcast channel is used to broadcast data items in different frequencies according to their relative access rates. That is, popular data items are more frequently broadcast than unpopular ones. In [11], a scheduling method is proposed to make the broadcast by using a stochastic model. It considers the access frequencies of data objects and controls their delivery intervals. None of these studies considers the relationship between data objects when the mobile transaction contains more than one data object. Furthermore, all these approaches assume read only mobile transactions. In this paper we investigate an efficient scheduling method that take into account an update mobile transaction. Our scheduling algorithm works in an ad hoc manner to determine efficiently the best placement of data items that maximizes the number of committed transactions per broadcast cycle. The remainder of this paper is organized as follows. Section 2 formulates the data organization problem. Section 3 proposes our scheduling algorithm. In Section 4 a mathematical model is introduced along with its probability formulas. Section 5 contains the experimental results. Finally, conclusions are given in section 6.

2. Problem Formulation

Since transaction conflict is the main factor affecting throughput for transaction processing in wireless computing environments, our server analyzes the accessed data patterns of the mobile transactions and decides the ordering of the data items in the broadcast channel. Data items are broadcast in a way that minimizes both the access time and probability of abort for transactions, and so data objects with minimum conflict and a maximum number of requests are favoured. A good placement of data in a broadcast channel can help a large number of mobile transactions to be committed early and, as a result, increase the system throughput and decrease the response time. Let $T = \{T_1, T_2, \dots, T_n\}$ be a set of *n* mobile transactions, where each mobile transaction *Ti* requests a set of data objects either for write or read access. Transaction *Ti* is an update mobile transaction if it requests at least one data object for write access, otherwise the transaction is read only. Let $D = \{d_1, d_2, \dots, d_m\}$, denote the union of the data items accessed by transactions in *T*, that is,

$$D = \bigcup_{1 \subseteq i \subseteq n} TDS(Ti) \quad (1)$$

where *TDS(Ti)* is the set of data items requested by transaction *Ti*. Let *RDS(Ti)* and *WDS(Ti)* denote the sets of read and write requests respectively, and assume that $\underline{WDS(Ti)} \subseteq RDS(Ti)$ which means that transaction *Ti* must read a data item before it can write to it (i.e., no blind writes). Further, assume that transaction *Ti* sends its writes to the database in finishing its execution. So each transaction *Ti* consist of *RI+1* steps - the first *RI* steps read data items from the broadcast channel and the last operation sends its update to the database server with a commit request. We consider the following Data Scheduling Problem (DSP): given a set of conflicting transactions $T = \{T_1, T_2, \dots, T_n\}$ arriving at the server during broadcast *i*, in the time interval [BC_l^i, BC_u^i], each with their read set *RDS(Ti)* and write set *WDS(Ti)*, the broadcast scheduling problem is to find an optimal broadcasting schedule which minimizes the total number of aborted transactions during broadcast cycle *i+1*, for given access time for the broadcasted data.

3. Scheduling Algorithm

Most existing approaches assume read only mobile transactions. In many applications, a mobile client might need to execute a transaction updating more than one data item. Under these circumstances, mobile transaction processing is highly affected by the data organization in the broadcast channel. To allow more transactions to commit, the data scheduling strategy can take the similarity between data accessed by different mobile transactions and the order in which each mobile transaction requests these data. The way a mobile client accesses a broadcast stream is illustrated in Figure 1, where the server broadcasts a set of data objects *{d1, d2, d3, d4, d5}* in one bcast, and a client accesses data items it needs in a sequential manner, i.e., *d1, d4*.

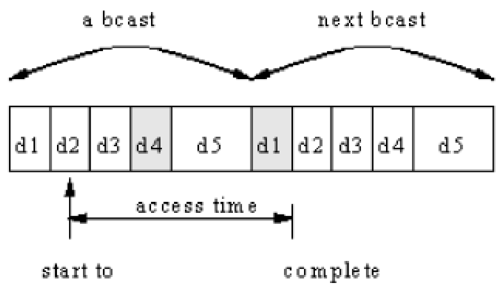

Figure 1. Data broadcasting.

Given a set of transactions $T = \{T_1, T_2, \dots, T_n\}$ whose union of accessed data items is the set $D = \{d_1, d_2, \dots, d_m\}$, with read and write sets for *Ti* *RDS(Ti)* and *WDS(Ti)* respectively, our scheduling algorithm for broadcasting data tries to optimize the following two factors: (1) the average access time for read only and update mobile transactions, and (2) the number of aborted transactions due to update made by one of the mobile transactions.

4. Mathematical Model

4.1. Transaction Conflict

Transactions are classified into equivalence classes. The transactions in one class process the same read set and write set. A transaction *Xj* in class *J* causes a transaction *Xi* in class *I* to be aborted if the update order of transaction *Xj* of an item in the union of WSI and RSI before *Xi* reads it, or *Xj* reads an item in WSI before *Xi* updates it. Therefore, a potential conflict between two transactions depends on the items they request and the partial order of operations in each transaction. For each class *I*, two sets of equivalence classes, *Gr(I)* and *Gw(I)*, are defined that give transactions which may conflict with those in class *I*. Let *Xi* be a transaction in class *I*. The set *Gr(I)* gives the classes of transactions whose update requests may abort some of *Xi's* read requests. The set *Gw(I)* gives the classes of transactions which may abort *Xi's* update request by reading some items in WSI. Below, is the algorithm that finds the sets *Gr(I)* and *Gw(I):*

A class *J* which is in both *Gw(l)* and in *Gr(i)* contains transactions which will read some items in *Xi's* write-set and update some items in the union of *Xi's* write set and read set suppose that a transaction *X* arrives during the course of the execution of transaction *XL*. If *X* comes neither from the classes which are neither in *Gr(I)* nor *in GW(I)*, It will never cause *Xi* to be aborted.

Input (all transaction requests during [BC_l^i , BC_u^i])
Output (data items organization for broadcast cycle i+1)
1. Construct the data -to- data matrix D_i as follow

$$D_i = \begin{array}{c|cccc} & d_1 & d_2 & \cdots & d_n \\ d_1 & cf_{11} & cf_{12} & \cdots & cf_{1n} \\ d_2 & cf_{21} & cf_{22} & \cdots & cf_{2n} \\ . & . & & & \\ . & . & & & \\ d_n & cf_{n1} & cf_{n2} & \cdots & cf_{nn} \end{array} \quad (1)$$

where co-freq (d_i ,d_j) represents the occurrences frequency of data items d_i, d_j in all mobile transactions during broadcast cycle i (i.e.,[BC_l^i , BC_u^i])

$$cf_{ij} = \begin{cases} co\text{-}freq(d_i, d_j) & if \;\; i \neq j \\ 0 & if \;\; i = j \end{cases} \quad (2)$$

2. Construct transaction- to- data matrix TD_i as follows:

$$TD_i = \begin{array}{c|cccc} & T_1 & T_2 & \cdots & T_m \\ d_1 & td_{11} & td_{12} & \cdots & td_{1m} \\ d_2 & td_{21} & td_{22} & \cdots & td_{2m} \\ . & . & . & \cdots & . \\ . & . & . & \cdots & . \\ . & . & . & \cdots & . \\ d_n & td_{n1} & . & .. & td_{nm} \end{array}$$

(3)

where

$$td_{ij} = \begin{cases} \dfrac{pos(d_i, T_j)}{LT_j} & if \; d_i \in WDS(T_j) \\ 0 & otherwise \end{cases} \quad (4)$$

and $pos(d_i, T_j)$ represents the position of the operation that accesses d_i in transaction T_j with respect to the partial order for all operations of transaction T_j and LT_j is the cardinality of transaction T_j (i.e., the number of operations in T_j).

3. Find $\varphi = (TD_i)^T * D_i$ (5)

4. Map the data item with the highest value of its column to the first position and the second highest to the second and so on. If two data items have the same value then use the row sum (i.e., the weight for the transaction) to determine the order of those data items that have same value.

Figure 2. DSP computational algorithm.

4.2. Probabilistic Formulas

This section investigates the performance of our scheduling algorithm in terms of the probability that a transaction is able to avoid conflict and commit. A conflict between two transactions results from a bad scheduling of their requests for a common set of items. The schedule of a transaction's requests can be determined by the co-frequency of the data items (*d1, d2,…dn*) in each transaction and the access order *Q* of these data items in each transaction, that is, for each transaction *Xi* in class *I*, the partial order of its operations and the updates made by conflicting transactions between any two consecutive data accesses. Since a database server broadcasts data to a large number of mobile clients at the same time, it is assumed that the arrivals of transactions in each class *I* determined by their first access of the broadcast channel happen according to a poisson process. Specifically, each equivalence class is viewed as a poisson stream of transactions.

```
For each class i, find the sets Gr(I) and Gw(I):
 Gr(I) := [ ] ; // Gr(I) is the set of equivalence classes each of which
 contains transactions whose update requests may abort some of Xi's read
 requests.
 Gw(I):= [ ]; // Gw(I) is the set of equivalence classes each of which
 contains transactions whose read requests may abort Xi's update request.
For each class J Do
 If (WSJ ⋂ (WSI ⋃ + RSI)) ≠ [ ] Then Gr (I):= Gr (I) + [J];
 If ((WSJ ⋃ RSJ) ⋂ WSI) ≠ [ ] Then Gw (I):= Gw (I) + [J];
End;
```

Figure 3. Computational algorithm 1.

The life-time of a transaction consists primarily of the access time for all data items needed by the transaction. The time between two consecutive accesses of a transaction is therefore based on the data organization of the broadcast channel. For our performance study the following term will be used:

- *D* is the time between two consecutive accesses of a transaction. It is the sum of access time of the two accessed data items; Since a transaction in class *I* reads k items from the broadcast channel in a sequential manner and sends $q \leq k$ updated items to the database server, it takes $\sum_{i=1}^{k} AC(d_i) + bq$ time to commit, where *bq* is the time needed to send the update to the database server. For the investigation of the probability that a transaction *Xi* in class *I* will commit, the following terms are defined:

- *Pi*: the probability that the transaction *Xi* will commit.
- *Qi*: an integer ranging from 1 to *RI!* inclusively. It is used to denote a specific order in which the read requests are made by the transaction *Xi* from the broadcast channel

- *c(Qi, d)* : the item accessed by the transaction *Xi* in its *d-th* operation, given that it accesses the items in the specified order *Qi*.
- *Pi (Qi, d)*: the probability that the transaction *Xi* is successful at its *d-th* request, given that it read the items in the specified order *Qi*.

Because data broadcast in the downlink channel is accessible by all transactions, any read operation of a transaction is never rejected, thus

$$Pi\ (Qi,\ 1) = 1 \qquad (6)$$

where *Pi(Qi, RI+1)*: is the probability that the transaction *Xi* will commit, given that it reads the items in the specified order *Qi*. A transaction which starts after *Xi* in class *i*, has *Ri*! different ways to read the *RI* items in the union of its read set and write set. Using the assumption that all the *RI!* Sequences of read requests are equally likely, we have

$$Pi = \sum_{Q_i=1}^{RI!} \frac{P_i(Q_i, RI+1)}{RI!}$$
(7)

Let *Qi* be fixed and *c(Qi, l), c(Qi.2). . . . . ,c(Qi.RI)* be the items listed in the order in which they are read by the transaction *Xi*. If the transaction *Xi* succeeds at its *d-th* read request and *d*$\leq$*RI*, it reads each of the first *j* items before any transaction updates it. Suppose that a transaction *Xj* starts before the transaction *Xi* makes its *d-th* read request and that *J* is in *Gr(I)*. For the investigation of the conflict between the transactions *Xi* and *Xj*, we find among the sequence of the items *c (Qi, l), c(Qi,2), . . ., c(Qi, d)* an item whose position *e (Qi.J, d)* in the sequence is defined as follows:

- *e(Qi, J.d) <= d*,
- *c(Qi, e (Qi,J,d))* is in WSJ, and
- *e(Qi, J.d)* is the largest integer that satisfies the above two conditions.

If *(e(Qi,J,d)-l-RJ-*$|WSj|$*)>0*, and the transaction *Xi* is able to avoid the conflict and succeeds at its *d-th* read request, no transactions in class *J* start in the time interval *(st(Xi), st(Xi)+(e(Qi,J,d)-1-RJ-*$|WSj|$*)*D)* and commit. Considering all the classes in *Gr(I)*, we obtain

Pi (Qi, d)=

$$\prod_{J..inGr(I)and(RJ+|WSJ|<e(Qi,J,d)-1)|} e^{Pj*(e(Qi,J,d)-1-RJ-|W}$$
(8)

where *1<d<RI* and *st(Xi)* is the start time for transaction *Xi*.

A transaction *Xj* in one of the classes in *Gw(I)* may cause *Xi* to be aborted at its update request by reading an item in the write set WSI. Let *Qj* be fixed and *c(Qj, l), c(Qj,2), ... , c(Qj, RJ)*, be the sequence of items read by the transaction *Xj* in that order. In the above sequence of items the position *g(1, Qj)* of the first item that is in the write-set WSI is determined by the following conditions:

- *g(i, Qj) = RJ*,
- *c(Qj. g(i,Qj))* is in WSI, and
- *g(i,Qj)* is the smallest integer that satisfies the above two conditions.

If the transaction *Xi* succeeds at its *(RI+l)th* update request, it succeeds at each of its read requests, and no transactions in any class *J* of the set *Gw(I)* start in the time interval *(st(Xl). st(XI)+(RI+;WSI-g(i,QJ)+l)*D)* and read successfully their respective *c(Qj.g(i,Qj))*. The statement that no transactions in any class of the set *Gw(I)* start and read any item in the write set WSI since *Xi's* start, implies that no transactions in the intersection of *GW (I)* and *Gr (i)* are able to commit and conflict with any *Xi's* read request. Considering all the classes in the sets *Gw (I)* and *Gr (l)*, we obtain

Pi (Qi, RI+1) =

$$\left(\prod_{J..in(Gw(I)andQj=1and(g(i,Qj)-1)<RI+|WSI|)} e^{Pj(Qj,g(i,Qj))*(RI+|WSI|}\right.$$
$$*$$
$$\left(\prod_{J..in(Gr(I)-Gw(I)and..(RJ+|WSJ|<e(Qi,J,RI)-1)} e^{Pj*e(Qj,J,RI)-1-RJ-}\right.$$
(9)

The above equality is due to the assumption that the probability *Pj* for class *J* in *(Gr(I)-Gw(I))* is not significantly affected by the condition that no transactions in any class of the set *Gw(I)* start. The equations 1 to 4 are true for any class *I* and any operation order *Qi* = 1, 2,…, *RI*!

4.3. Computational Algorithm

The equations 1 through 4 represent a set of non-linear equations expressed in terms of the probabilities *Pi* and *Pi(Qi,d)* for any class *I* and any *Qi = 1, 2, . . . , Rl*+1. The solution of these equations depends on the fact that a short transaction that takes a short time to commit, experiences a conflict with a small number of transactions, and its probabilistic formula involves fewer undetermined probabilities than those of a long transaction. The computational algorithm to solve these equations begins with the class of transactions which take the shortest time to commit. The probabilities for the transactions in this class can be obtained straightforwardly. Consequently, the number of classes of transactions whose probabilities are to be determined is reduced by one. The computational algorithm then repeats with the remaining classes of transactions. The mathematical analysis of the performance of our scheduling algorithm is complete with computational algorithm 2.

5. Simulation Experiments

Our model specifies a set of broadcasted data items and different classes of transactions. A class of transactions is further specified by a write set, a read set and order for which transactions in that class access their data items. We model the broadcast channel as a server with a fixed rate of broadcast and assume that the broadcast channel is error-free. A data object delivered on the broadcast can be received simultaneously by all of the clients that are waiting for it at that time. To justify the probabilistic formula which the model is based on, Table 1 shows a comparison of simulation and mathematical analysis results for the probabilities that a transaction avoids conflict and commits under different parameter settings. The order in which each transaction accesses its data items in a certain class is simulated by generating all the permutations of data accesses in that class and then assigning to each transaction one of these permutations randomly. Two experiments are conducted: one with repletion which allows more than one transaction in the same class to have the same order and the second gives each transaction in the same class a unique order. The remaining parameters represent the number of classes used and the number of transactions in each class. The number of operations for all classes is assumed to be fixed and equal to 8. The number of write operations is also assumed to be fixed and equal to 3 per transaction.

```
Input :( a set of classes where the probabilities for transactions in all these
classes have not been obtained yet - we denote it by Undetermined set)
Output (a set of classes, where the probabilities for transactions in all
these classes have been obtained already. We denote it by Determined set)
Undetermined: = [all the classes];
Determined: = []
While Undetermined <> [] Do
   Find a class I, among the classes in
         Undetermined, which has the smallest RI +WSI
   m: = RI + $|WSI|$ ;
    For each class j in Undetermined Do
        For ( Qj = 1, 2,..., RJ! ) Do
            For every d, d<=RJ AND d<=m AND Pj
                  (Qj, d) is Undetermined Do
               Compute Pj (Qj, d) by using Formula 3,
               // in Formula 3; this probability is only
               related to the probabilities for the classes
                in //Determined
            End ;
        End ;
   End ;
For Qi = 1, 2, . . ., RI! Do
      Compute Pi (Qi, Rl+l) by using Formula 4;
      // In Formula 4, this probability is only related
     to the Determined probabilities.
 End ;
Compute Pi by using Formula 2;
Undetermined: = Undetermined - [I];
Determined: = Determined + [I];
End ;
```

Figure 4. Computational algorithm 2.

Table 1. Simulation versus mathematical values.

Number of Classes	Number of Transaction Per Class	Order Repetition	Probability of Commit (Mathematical)	Probability of Commit (Simulation)
20	5	Yes	0.9031	0.8997
40	5	Yes	0.9171	0.9121
20	5	No	0.9331	0.9297
40	5	No	0.9471	0.9421

6. Conclusion

The construction of a mathematical model and an algorithm for a scheduling method has been presented. The model provides insight into aspects of transaction processing in broadcast environments taking into consideration the access order for the data items being broadcasted to large numbers of users that may have a common data access, and the scheduling of these data items to avoid conflict as much as possible for these transactions. The development is unique in its use of sets of equivalence classes as the basis for the study of transaction conflict. A simulation study was designed to test the model, i.e., to justify the underlying assumptions of the model and the probabilistic formulas on which the model is based.

References

[1] Acharya S., Alonso R., Franklin M., and Zdonik S., "Broadcast Disk: Data Management for Asymmetric Communication Environments," *in Proceedings of the ACM SIGMOD Conference*, pp. 199-210, 1995.

[2] Acharya S., Franklin M., and Zdonik S., "Disseminating Updates on Broadcast Disks," *in Proceedings of the Very Large Data Bases Conference*, pp. 354-365, 1996.

[3] Chang Y. and Hsieh W., "An Efficient Scheduling Method for Query-Set-Based Broadcasting in Mobile Environment," *in Proceedings of the IEEE International Conference on Distributed Computing Systems Workshops*, pp. 478-483, 2004.

[4] Chen M., Wu K., and Yu P., "Optimizing Index Allocation for Sequential Data Broadcasting in Wireless Mobile Computing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 1, pp. 161-173, 2003.

[5] Chung Y. and Kim M., "An Index Replication Scheme for Wireless Data Broadcasting," *Journal of Systems and Software*, vol. 51, no. 3, pp. 191-199, 2000.

[6] Chung Y. and Kim M., "Effective Data Placement for Wireless Broadcast," *Distributed and Parallel Databases*, vol. 9, no. 2, pp. 133-150, 2001.

[7] Imielinski T., Viswanathan S., and Badrinath B., "Energy Efficient Indexing on Air," *in Special Interest Group on Management of Data (SIGMOD) ACM*, pp. 25-36, 1994.

[8] Lam K., Chan E., and Yuen J., "Approaches for Broadcasting Temporal Data in Mobile Computing Systems," *The Journal of Systems and Software*, vol. 51, no. 3, pp. 175-189, 2000.
[9] Lee G., Lo S., and Chen A., "Data Allocation on Wireless Broadcast Channels for Efficient Query Processing," *IEEE Transactions on Computers*, vol. 51, no. 10, pp. 1237-12525, 2002.
[10] Lo S. and Chen A., Optimal Index and Data Allocation in Multiple Broadcast Channels, Data Engineering, 2000.
[11] Su C., Tassiulas L., and Tsotras V., "Broadcast Scheduling for Information Distribution," *Wireless Networks*, vol. 5, no. 2, pp. 137-147, 1998.
[12] Tan K. and Yu J., "Generating Broadcast Programs that Support Range Queries," *IEEE Transactions on Knowledge and Data Engineering*, vol. 10, no. 4, pp. 668-672, 1998.
[13] Vaidya N. and Hameed S., "Scheduling Data Broadcasting in Asymmetric Communication Environments," *Kluwer Academic Publishers*, Dordrecht, vol. 5, no. 3, pp. 171-182, 1999.
[14] Yee W., Student Member, Navathe S., Omiecinski E., and Jermaine C., "Efficient Data Allocation over Multiple Channels at Broadcast Servers," *IEEE Transactions on Computers*, vol. 51, no. 10, pp. 1231-1236, 2002.

Ahmad Al-Qerem graduated in applied mathematics, obtaining a BSc in 1997 from JUST University and a Masters in computer science from Jordan University in 2002. After that he was appointed a full-time lecturer in the department of Computer Science at Zarqa Private University and also a part-time lecturer for the Arab Open University. He has also held a post in the Ministry of Labour. Currently, he is a PhD student at Loughborough University, UK. He is interested in concurrency control for mobile computing environments, and particularly transaction processing. He has published several papers in various areas of computer science.

Walter Hussak graduated in mathematics, obtaining a BSc in 1979 and a PhD in 1983 from Sheffield University. Later he obtained an MSc in systems design from Manchester University, awarded in 1987. He joined Manchester University and worked for Professor Brian Warboys as a research associate on the Alvey Flagship Parallel System Project and later the ESPRIT II European Declarative (Parallel) System (EDS) project which was collaborative with industrial partners ICL, Bull and Siemens and was an industrial-scale system to run relational database systems and declarative (functional and logic) languages efficiently. The success of the EDS system was indicated by a subsequent commercial derivative, the ICL GOLDRUSH system. He was appointed to his first university full academic post as a lecturer in computer science at Loughborough University in 1991. He has published several papers at international conferences and in journals, on formal methods and database concurrency. He is currently a member of the networks, control and complex systems research group at Loughborough University and is interested in formal methods and mathematical aspects of database concurrency.