# (m, k)-Firm Constraints and Derived Data Management for the QoS Enhancement in Distributed Real-Time DBMS

Malek Ben Salem[1], Emna Bouazizi[1,2], Claude Duvallet[3], and Rafik Bouaziz[1]
[1]Higher Institute of Computer Science and Multimedia, Sfax University, Tunisia
[2]College of Computer Science and Engineering, Jeddah University, Jeddah, Saudi Arabia
[3]Normandie Univ, UNIHAVRE, LITIS, 76600 Le Havre, France

**Abstract:** *Distributed Real-Time DBMS (DRTDBMS) is a collection of Real-Time DataBase Management Systems (RTDBMS) running on sites connected together via communication's networks for transaction processing. This system is characterized by the data distribution and the unpredictable transactions. In addition, in this system, the presence of several sites raises the problem of the unbalanced load between those nodes. In order to enhance the performance of DRTDBMS with taking into account those problems, Quality of Service (QoS) based approaches are the most appropriate. Distributed Feedback Scheduling Control Architecture (DFCSA) is proposed for managing the QoS in this system. In DRTDBMS, the results produced in time with less precision are sometimes preferable than exact results obtained in delay, inaccuracy can be tolerated. In order to take this assertion, in this paper, we extend the DFCSA by using the (m, k)-firm constraints, which take into account the imprecise results, using three data replication policies. The obtained architecture is called (m, k)-firm-User-DFCS. The second contribution consists of taking into account the real-time derived data on (m, k)-firm-User-DFCS architecture, always, using three data replication policies. The obtained architecture is called Derived Data Management (DDM)-(m, k)-firm-User-DFCS. Then, we are focusing on two ways of service optimization in DRTDBMS. We are interested in (1) the space optimization in which we propose to apply three replication data policies, and (2) the QoS optimization in which we propose to take into account the real-time derived data and (m, k)-firm constraints.*

**Keywords:** *Software DRTDBMS, QoS management, feedback control scheduling, (m, k)-firm constraints, derived data.*

*Received June 28, 2015; accepted January 4, 2017*

## 1. Introduction

Currently, many applications use distributed computing, real-time processing and require, in addition, the management of large amount of data like wireless sensor network [8], mobile computing technology [17] and dynamic environments [3]. Thus, Distributed Real-Time DBMS (DRTDBMS) are increasingly needed to satisfy these applications since they are designed to manage, in both, large volumes of distributed data and real-time constraints. DRTDBMS include a set of nodes connected via communication networks for transactions processing, where data and transactions are totally distributed.

DRTDBMS are greatly exposed to unpredictable workload, caused by user transactions arriving at varying frequencies, and to an unbalanced distribution of this workload between nodes. This leads to instability periods in which the system becomes overloaded. During overloading periods, there is a lack of DRTDBMS resources, so that transactions greatly miss their deadlines. Therefore, it is essential to keep the system in a stable state in order to guarantee a better QoS.

To control system's instability periods, approaches based on feedback control real-time scheduling theory are proposed [1, 14]. Further, Wei *et al*. [18] proposed an algorithm for Quality of Service (QoS) guarantees in DRTDBMS based on a Distributed Feedback Control Scheduling Architecture (DFCSA). Although this solution guarantees a better overall QoS, it still has several drawbacks.

The use of full data replication policy is a costly technique if the amount of data is high (cost of storage and cost of updating copies). This can be very costly in the case of a distributed Database Management Systems (DBMS) with a number of sites greater than eight. This shortfall can be remedied by using other replication policies in order to take into account the increase in the number of distributed RDBMS sites.

This architecture use a classical admission controller of transactions where the atomicity property of transactions is guaranteed, i.e., the transaction is full accepted with their operations or fully rejected. This controller contributes greatly of the balance of system state, but that is in detriment of the number of committed transactions before their deadline. This is due to the number of user transactions to be removed from the system increases during the overloading phases. In order to increasing the number of committed transactions in deadline, a solution may be proposed which relax the atomicity property of transactions.

The DFCS architecture don't take into account the real-time derived data management, to our knowledge. Only the real-time data are considered in this architecture. However, in real-time databases, derived real-time data are massively used to reduce the databases size and decrease the processing time of data that depends on other data. For this, a solution could be envisaged for better management of the real-time derived data while including it in a loop feedback control and in the distributed context.

In previous work, Hamdi *et al.* [11] have proposed a distributed feedback control scheduling architecture taking into account the real-time derived data using only the semi-total data replication policy [6]. Furthermore, in Ben Salem *et al*. [4] have proposed an architecture which apply the (m, k)-firm constraints for user transactions in distributed feedback loop using only the full data replication policy.

In this paper, our objective is to enhance QoS in DRTDBMS by maximizing the number of transactions which meet their deadlines, while maintaining a robust DRTDBMS behaviour facing instability periods. Our approach consists, in the first step, of applying the (m,k)-firm technique to user transactions in distributed feedback control scheduling using three data replication policies, the obtained architecture is called (m,k)-Firm-User-DFCS Architecture. The second step consists of taking into account the real-time derived data on the (m, k)-Firm-User-DFCS Architecture, the result architecture is called Derived Data Management (DDM)-(m, k)-Firm-User-DFCS Architecture. The purpose of our work is to propose architectures that provide efficient QoS adaptability and performance reliability even in the presence of unpredictable workload in DRTDBMS.

The remaining of the paper is organized as follows. In section 2, we present the related work which consists of the real-time database model, the performance metric, the existing architecture on which our work is based, and the notions of real-time derived data and (m, k)-firm constraints. Section 3 describes our proposed approaches for QoS guarantees in DRTDBMS where the first approach consists of applying (m, k)-firm constraints for user transactions on DFCS Architecture using three data replication policies, and the second approach consists of taking into account the real-time derived data on (m, k)-firm-User-DFCSA always using three data replication policies. Those proposed extended architectures are evaluated according to a set of simulation results in section 4. We conclude the paper, in section 5, by briefly discussing our work and by presenting our future work.

## 2. Related Work

In this section, we describe our distributed real-time database model by presenting data and transaction models, and defining the basic performance metric we consider. Similarly, we present the QoS management architecture proposed in [18], on which we based our work. We finish by giving an overview of the previous work in which the (m, k)-firm approach and derived real-time data are used for the QoS enhancement.

In our data model, we consider a replicated main memory database in which we have real-time and non real-time data [1]. Real-time data are sensor data from physical world, and are updated periodically to reflect accurately the real-world state. Each real-time data object has a validity interval beyond which it becomes useless, and a timestamp indicating the last observation of the real-world state. Non real-time data are those found in conventional databases and that do not change dynamically with time. The data replication technique increases the data availability at different sites. Then, it significantly helps transactions to meet their time requirements [18]. Data could be either fully [10] or semi-total [6] or partially replicated [16].

For distributed real-time transactions, we consider firm deadline transactions [1], where if a transaction misses its deadline, it will be aborted and becomes useless for the system. Transactions are divided into update and user transactions according to the type of their accessed data items. Update transactions are executed periodically, in order to refresh real-time data objects. They update, likewise, real-time data replicas. We consider that each update transaction consists of one sub-transaction, having always a write operation. User transactions are a periodic. We consider that each one consists of a set of sub-transactions. Then, each sub-transaction is composed of a set of read operations on both real-time and non real-time data objects, and of write operations on only non real-time data objects.

In distributed real-time databases, the transactions may executed in local or in remote site [5, 6] according to the location of their required data items. It should note that when dealing with DRTDBMS models based on a load balancing technique, a transaction may have all of its required data at its local site, but it is then distributed in order to alleviate the overload situation of that site.

### 2.1. Performance Metric

The main performance metric, we consider in our model, is the Success Ratio (SR). It is a QoS parameter which measures the percentage of transactions that meet their deadlines. It is defined as follows:

$$SR = 100 \times \frac{\#timely}{\#timely + \#tardy} (\%) \quad (1)$$

where *#timely* and *#tardy* represent, respectively, the number of transactions that have met and missed their deadlines.

## 2.2. QoS Management in DRTDBMS

The QoS is increasingly important for evaluating the performance of a DRTDBMS, in which the system performance depends on the workload distribution. In Wei *et al*. [18] proposed an architecture using feedback-based global load balancers and local feedback controllers. This architecture, called Distributed Feedback Scheduling Control Architecture (DFCSA), on which we base our work, aims a load balancing between system nodes and an efficient management of transactions workload fluctuations. The general outline of the DFCSA is shown in Figure 1. In what follows, we give a brief description of its basic components.

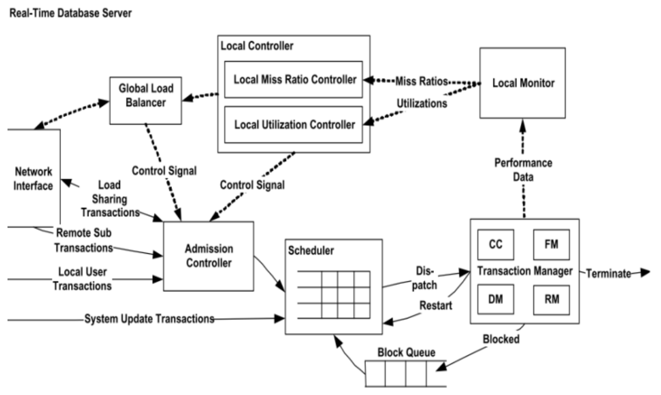

Figure 1. The DFCS architecture for QoS guarantees [18].

The admission controller is used to regulate the system workload in order to prevent its overloading, by referring to the estimated CPU utilization and the target utilization set point of the system. The scheduler is used to schedule transactions according to the Earliest Deadline First (EDF) protocol [2, 14].The transaction manager handles the transactions' execution. It consists of a Concurrency Controller (CC), a Freshness Manager (FM), a Data Manager (DM) and a Replica Manager (RM). The CC solves accessing data conflicts appearing between transactions using the Two Phase Locking High Priority (2PL-HP) [1] protocol. The FM checks the data freshness and blocks a *user* transaction if the accessed data item is stale. The DM has to update real-time data replicas, and the RM handles data replicas using a replication control protocol.

At each sampling period, the local monitor samples the system performance data, by referring to statistics about transactions' execution which it retrieves from the transaction manager. Measured values belong to the feedback control loop and are, then, reported to the local controller. The local controller includes the local utilization controller and the local miss ratio controller, which generate, respectively, the local miss ratio and the local utilization control signals, based on the received values and on the system reference parameters, according to which this controller sets the system target utilization to be considered at the next sampling period.

The global load balancer ensure the system load balancing, by exchanging the system performance data with others nodes, which is guaranteed by transferring transactions from highly overloaded nodes to less overloaded nodes. The amount of workload to be transferred is controlled by the Load Transferring Factor (LTF) of each node [18].

## 2.3. The (m, k)-Firm Constraints in Literature

The (m, k)-firm constraints was initially introduced for periodic tasks in real-time systems [15], in order to relax strict real-time constraints. It has also been adapted to transactions in real-time databases, aiming to decrease the number of missed deadlines [7].

In DRTDBMS, a (m, k)-firm real-time transactions model has been proposed in [13]. It consists of decomposing each user transaction $T_i$ into $k_i$ sub-transactions, among which $m_i$ are mandatory and the others are optional. Mandatory sub-transactions are distinguished from optional according to their weights, i.e., the $m_i$ mandatory sub-transactions are those having the highest weights.

Likewise, to manage the QoS in DRTDBMS, the (m, k)-firm constraints was used on distributed feedback control technique in [4]. In this work, Ben Salem et al. have proposed a new architecture based on the (m, k)-firm constraints for *user* transactions on distributed feedback control using only full data replication policy.

## 2.4. Real-Time Derived Data in Literature

In addition to the base real-time data, the DRTDBMS very often contains real-time derived data. This is data that indirectly reflects the state of the outside environment, and can be computed from the basic real-time data [9].

The real-time derived data use real-time data that have a validity interval in which its can be used. When, derived data is computed following more real-time data, this data become a real-time and it validity interval is the intersection of validity intervals of each used real-time data.

Three types of real-time derived data are defined according to their sources:

- Real-time derived data are calculated only from basic real-time data.
- Real-time derived data are calculated from basic real-time data and real-time derived data.
- Real-time derived data which is calculated only from basic real-time data and censorial data.

The challenge of this kind of data is the updating policies [9]. Indeed, whenever the source data is updated, then the real-time derived data must recomputed following one updating policy. Among these policies, we can distinguish: the periodic update policy, the triggered update policy, the on-demand update policy, the forced wait update policy, the forced delay update policy and the mixed update policy [9].

In DRTDBMS, the real-time derived data was taking into account on distributed feedback control technique in. In this work, Hamdi *et al*. [11] have extended the proposed architecture in [18] with taking into account the real-time derived data using only semi-total data replication policy [6]. The authors have compared between the most used of real-time derived data update methods. Based on obtained results, it shows that the mixed method gives the best performance than others. For that, in our present work, we use the mixed policy to manage the real-time derived data update in our proposed approach presented in the following section.

## 3. Extended Works for QoS Enhancement in DRTDBMS

Based on previous work proposed in literature for QoS improvement in DRTDBMS, we present, in this paper, our contribution which involves two extensions of the DFCS Architecture described above. The first extension of DFCS Architecture consists of applying the (m, k)-firm constraints on distributed user transactions in order to make those transactions meeting their deadlines. In the second extension, we propose to take into account the use of the real-time derived data on the first extension.

### 3.1. (m, k)-Firm Constraints for QoS Enhancement in DRTDBMS

Our first extended work of distributed feedback loop approach consists of applying the (m, k)-firm constraints for user transactions on distributed feedback-based architecture, using the three data replication types: the full replication, the partial replication [16] and the semi-total replication [6], for QoS enhancement in DRTDBMS. Compared to the work presented in [4], in this paper, we use three data replication policies. The proposed architecture was called(m, k)-Firm-User-DFCSA. It involves the admission control, the concurrency control and the commit process of transactions, by adapting their functioning to take into account the (m, k)-firm constraints of user transactions. This proposed architecture was validated by using only the full data replication policy.

Our challenge is to increase the number of transactions that meet their deadlines while maintaining a robust system's behaviour, face to unpredictable workloads induced by user transactions. The general outline of our approach architecture is shown in Figure 2. Compared to the conventional DFCSA, it is distinguished by the (m, k)-Firm-User admission controller and the (m, k)-Firm-User concurrency controller as shown in Figure 2, and by the (m, k)-Firm-User commit process even it does not appear in this figure [4].

Based on the (m, k)-firm constraints, we had proposed a model for user transactions. Each user transaction Ti (i in [1..n]) submitted to the system is decomposed into a set of sub-transactions. We denote by ki, the number of sub-transactions of Ti. Distinguishing mandatory sub-transactions from optional ones is based on the criticality of data required by transactions.

For example, in an air traffic control system and from a meteorological perspective, data like strength and trajectory of the wind are considered as more critical than data like the moisture rate in the air. Indeed, each sub-transaction Tij (j in [1..ki]) consists of a set of operations, each of which accesses a precise data item having a specific criticality. Thus, each operation accessing a critical data item is considered as critical, too. Accordingly, the mi mandatory sub-transactions of Ti are those having highest critical operations.

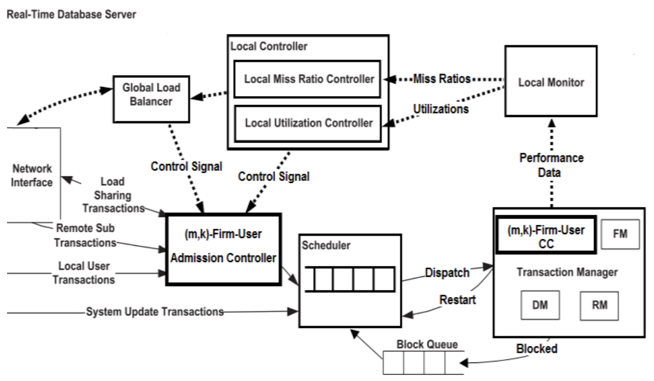

Figure 2. The (m,k)-Firm-User-DFCSA.

We consider that them $\frac{m}{k}$ ratio parameter should be fixed, in advance, by the Database Administrator (DBA), then, the value of $m_i$ is determined by Equation (2).

$$m_i = \frac{m}{k} \times k_i \tag{2}$$

For us*er* transactions admission, the admission controller in the conventional DFCSA operates in a binary way in order to respect the atomicity property of transactions. In our approach, we have proposed a (m, k)-Firm-User admission controller (cf. Figure 2), that takes into account the (m, k)-firm constraints defined for *user* transactions, aiming to relax the strict decision of the classical one of the DFCSA. The (m, k)-Firm-User admission controller tolerates the partial admission of a transaction, in which only its mandatory sub-transactions are admitted, if it is not possible to accept it in its entirety during overloading periods.

For transactions concurrency in the (m, k)-Firm-User-DFCSA, we have proposed a (m, k)-Firm-User concurrency controller (cf. Figure 2), that consider (m, k)-firm constraints of *user* transactions. With this concurrency controller, the admission of optional sub-transactions does not lead to overload the system when considering data access conflicts.

Therefore, for transaction's commit, we have proposed the(m, k)-Firm-User commit method, in which, an user transaction $T_i$ can commit if, at least, its $m_i$ mandatory sub-transactions commit. Furthermore, in this proposed commit method, and following the PROMPT protocol principle [12], we allow transactions to borrow non-committed data. This aims to reduce the blocking time of transactions, caused by the inaccessibility of data held by transactions waiting for committing, and give them more opportunities to meet their deadlines. By this way, we relaxed the isolation property of transactions, which imposes that the result of a transaction, that has not yet finished its execution, should be invisible for other transactions. However, the lending data process in our approach is controlled, so that it can only be performed by mandatory sub-transactions, considering that the optional ones can be ignored when committing a transaction.

In current work, we apply the (m, k)-Firm-User-DFCS Architecture under three data replication policies. In the first case, we apply the semi-total data replication policy and the partial data replication policy. In the second case, we proceed to a comparison between the three data replication policies (full, semi-total and partial).

### 3.2. Management of Real-Time Derived Data on (m, k)-Firm-User-DFCS Architecture

Hamdi *et al*. [11] have proposed a distributed feedback control scheduling architecture with taking into account the real-time derived data using only the semi-total data replication policy [6].

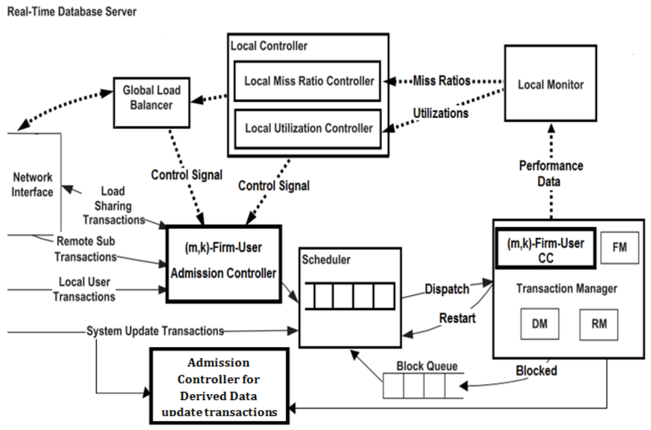

Figure 3. The DDM-(m,k)-firm-user-dfcs architecture.

In current work, we have proposed an extended architecture to manage the QoS in DRTDBMS based on (m, k)-firm-User-DFCS Architecture and taking into account of real-time derived data using three data replication policies. (cf. Figure 3). In fact, the (m, k)-firm-User-DFCS Architecture stabilizes the system during the phases of overload or under-utilization. So, it ensures the robustness of the DRTDBMS. But, it has one major drawback: it does not allow handling derived data. The objective of this extension is to keep the behavior of distributed real-time DBMS in a stable condition and reduce the number of transactions that miss their deadlines in order to improve the QoS provided to users with taking account the real-time derived data allowing the inaccuracy of the data and transactions in order to provide the requested QoS and to ensure a better use of available resources. We call this extension DDM-(m,k)-firm-User-DFCS Architecture. Two components have been modified on the obtained architecture. The first is the QoD manager which requires the information coming from other components to manage the admission of the transactions. That is why, as for the admission controller, a parameter coming from the feedback loop is transmitted to it Max Value Interval (MVI). The value of MVI is calculated according to the use of the system. It suggested the modification of the algorithm used to the level of the manager of quality of data in order to distribute the effort to provide on each of the components (controller of admission, controller of admission of the derived data, controller of precision) that have charged of the different types of present transactions in the system. The second is the Freshness Manager (FM). When a user transaction reaches an obsolete derived data, the FM blocks the transaction in a queue and sends an order to the controller of admission of the derived data re-computation transactions do not separate the transaction of update of the data in question.

In our model, we used one type of real-time derived data which are computed from conventional data and from basic real-time data. For updating real-time derived data, we use the mixed update policy (cf. section 2.4.).

## 4. Simulation Results of Proposed Approaches

In this section, we aim to evaluate the QoS performance provided by the proposed (m, k)-Firm-User-DFCSA and DDM-(m, k)-Firm-User-DFCSA according to a set of simulation experiments, where a set of parameters have been varied.

### 4.1. Simulation Settings

The main system parameter settings are given in Table 1. In the set of our experiments, we varied the number of system nodes and the data replication policy (full, semi-total and partial) which we applied to real-time data items of the database. Each simulation result represents the average of 10 simulations in order to obtain results confronting the reality. In our simulations, the generated real-time derived data are calculated from varied number of censor data and basic real-time data, and are updated using the mixed policy.

To evaluate the (m, k)-firm-User-DFCS Architecture, we have used three classes of DRTDBMS's size (8, 12, and 16). But, with taking account of real-time derived data on (m, k)-firm-User-DFCS Architecture, we use only the DRTDBMS's size with 16 nodes. Choosing the number of sites to 16 is without hypothesis. The transactions are scheduled according to the EDF algorithm. For resolving conflicts between transactions, we use the (m, k)-Firm-User concurrency controller which consider the (m, k)-firm constraints defined to user transactions. Distributed transactions are committed according to the (m, k)-Firm-User commit method. We note that arrival times of these transactions are generated according to the "Poisson" process using the lambda ($\lambda$) parameter. In our simulations, we varied the value of the m/k ratio in order to show the effect of either increasing or decreasing the number of mandatory sub-transactions in each user transaction. In our experiments results, we only consider user transactions.

Table 1. Simulation parameter values.

| Parameter | Significance | Value |
|---|---|---|
| **Duration** | Simulation time | 3000ms |
| **NbNonTRData** | Number of classical data items | 1000 |
| **NbTRData** | Number of real-time data items | 20 |
| **ValI** | Validity interval of real-time data items | [500ms, 1000ms] |
| **NbRTDData** | Number of real-time derived data items | 20 |
| **Dependance** | Dependence from other data | [1,5] |
| **MDE** | Maximum data error for real-time data | [2,5] |
| $\mathbf{T_{opRead}}$ | Read operation time | 1ms |
| $\mathbf{T_{opWrite}}$ | Write operation time | 2ms |
| $\lambda$ | Parameter of the "Poisson" distribution | 0.09 |
| **SF** | Slack factor of transactions | 10 |
| **RR** | Remote data ratio | 20 |
| $\mathbf{Nb_{SubTr}}$ | Number of sub-transactions per user transaction | [2, 4] |
| $\mathbf{Nb_{opRead}}$ | Number of read operations per sub-transaction | [0, 2] |
| $\mathbf{NB_{opWrite}}$ | Number of write operations per sub-transaction | 2 |
| **ratio** $\frac{m}{k}$ | Ratio for (m, k)-firm constraints | [0.5, 0.7] |

## 4.2. The (m, k)-Firm-User-DFCSA Simulation Results

In this section, we present and discuss the simulations results of (m, k)-Firm-User-DFCS Architecture under three classes of DRTDBMS's size: eight, twelve and sixteen nodes.

### 4.2.1. Simulation Results for Eight Nodes

The first set of experiments consists of simulating the behaviour of a DRTDBMS composed of 8 nodes according to the three data replication policies. By referring to Figures 4, 5, and 6, which represent, respectively, the transactions' success ratio using a full data replication, a semi-total data replication and a partial data replication in a system with 8 nodes, we find that the conventional DFCSA yields the worst performance, compared to the result provided by the (m, k)-Firm-User-DFCSA. As shown in these Figures, the number of successful transactions provided by the (m, k)-Firm-User-DFCSA is higher than the one provided by the conventional DFCSA either when the m/k ratio is equal to 0.5 or to 0.7. In addition, we can notice that the (m, k)-Firm-User-DFCSA with the ratio equal to 0.5 yields a better result than with the ratio equal to 0.7. Indeed, in the first case, more of optional sub-transactions are allowed to be rejected, giving then more opportunities for transactions to be executed before their deadlines. However, in the second case, the mandatory part of transactions is more important. Therefore, we can confirm that by minimizing the value of the ratio, the optional part of transactions becomes more important, which greatly reduce the transactions' miss ratio.

Let us compare now the (m, k)-Firm-User-DFCSA performances, when the ratio is equal to 0.5, according to the three data replication types in a DRTDBMS of 8 nodes. For this, we refer to the Figure 7 which shows that our approach using a full data replication provides at the beginning the best result. However, the limits of this type of replication reveal with the increase of the transactions number in the system compared to the partial data replication. This result is confirmed in the case when more the transactions number is high, more data requests are important, so that the cost of updating the data copies increases. We note also that the semi-total data replication gives the weakest performance, especially at the beginning. Indeed, the functioning of the semi-total replication is based on the execution history of transactions to determine the nodes that will be involved in the database replication. In other words, it starts without replicating data, and then according to transactions needs, in terms of data at different nodes, the execution history of transactions is dynamically updated and the replication process begins to take place. Therefore, the curve related to this replication policy, as shown in Figure 7, is more approximate by the end to the curves of the other replication types. The partial replication, providing the minimal cost of updating copies, is characterized by the best performance for a larger number of transactions. The Figure 8 corresponds to a comparison between the three types of data replication with a ratio fixed at 0.7. In this case, the number of mandatory sub-transactions is more important, which requires to satisfy more data access requests. For this reason the partial data replication gave the best result.

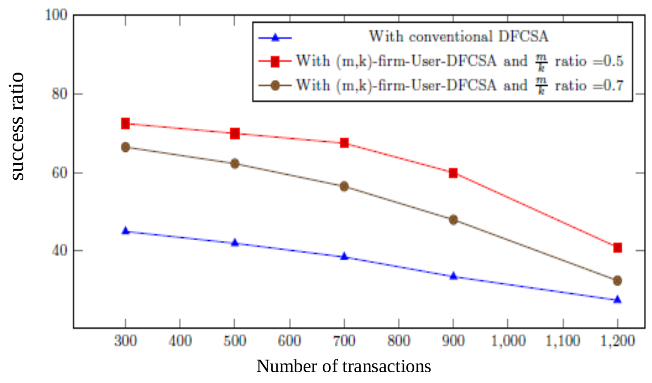

Figure 4. Simulation results using full replication and nb_sites = 8.

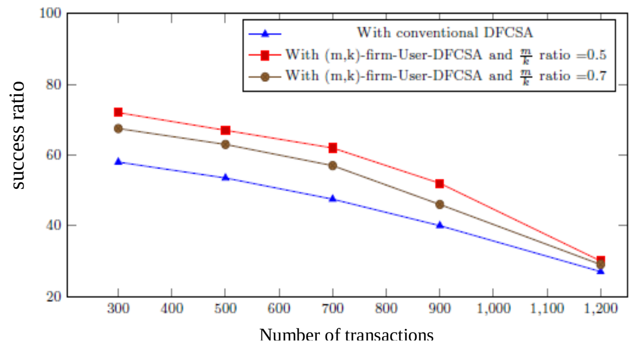

Figure 5. Simulation results using semi-total replication and nb_sites = 8.

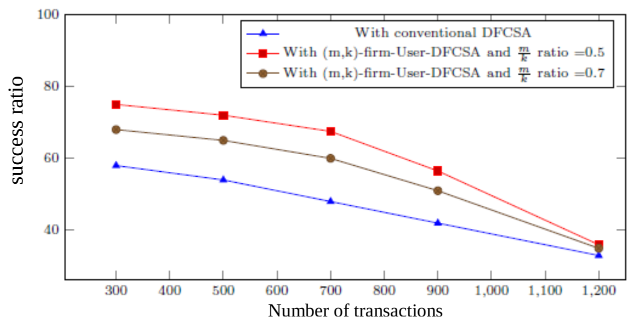

Figure 6. Simulation results using partial replication and nb_sites= 8.

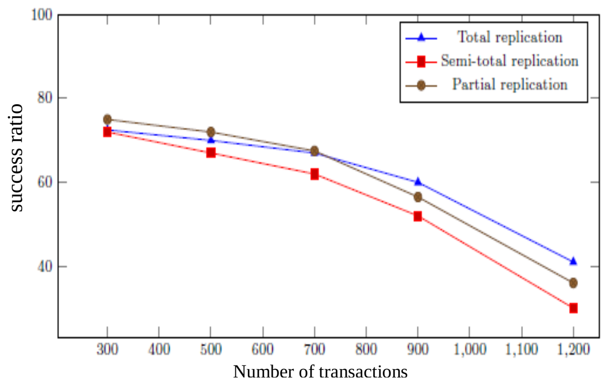

Figure 7. Three replication policies with *nb_sites* =8 and $\frac{m}{k}$ ratio=0.5.

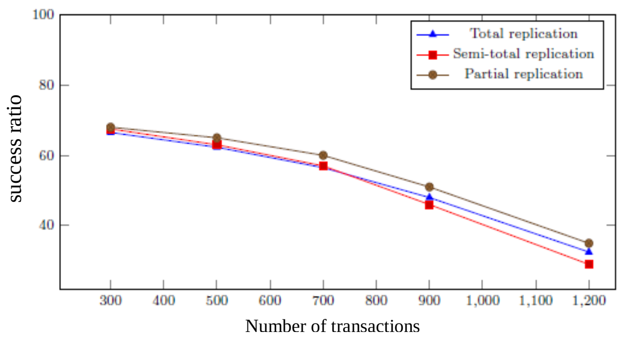

Figure 8. Three replication policies with *nb_sites* =8 and $\frac{m}{k}$ ratio=0.7.

### 4.2.2. Simulation Results for Twelve Nodes

The second set of experiments is to simulate the behaviour of a DRTDBMS composed of 12 nodes according to the three data replication policies. As shown in Figures 9, 10 and 11, the (m, k)-Firm-User-DFCSA keeps the best performance with respect to the conventional DFCSA, either with the ratio equal to 0.5 or to 0.7, although the improvement provided by the full data replication (cf. Figure 9) is slight due to the increase of the system size. Hence, using full data replication policy can't not improve the QoS which (m, k)-Firm-User-DFCSA must guarantee.

The comparison between the three replication policies, by referring to Figures 12 and 13, shows that the weakest performance is that of the full data replication, which becomes increasingly costly in terms of updating time of data copies when the system size increases. However, the semi-total and the partial data replication are more efficient when the number of copies increases in all system's nodes. As shown in Figure 12, the best result is that of the semi-total data replication policy, whose benefits increase with the increase of the system size, where the ratio is equal to 0.5. In contrast, the partial data replication policy provides a better result as shown in Figure 13, where the ratio is equal to 0.7.

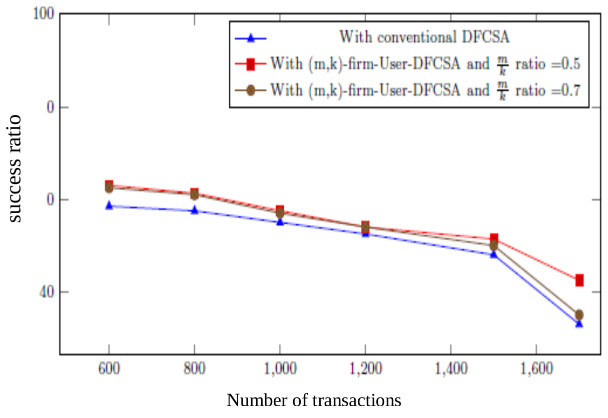

Figure 9. Simulation results using total replication and *nb_sites* = 12.

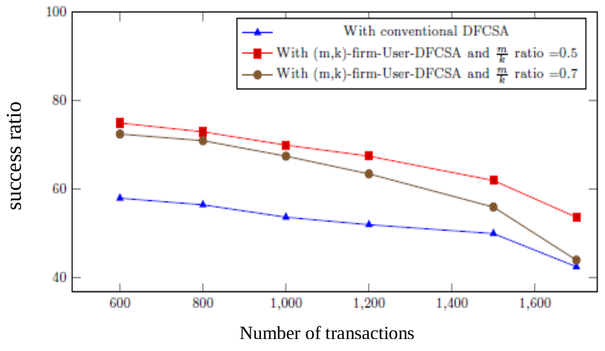

Figure 10. Simulation results using semi-total replication and *nb_sites*=12.

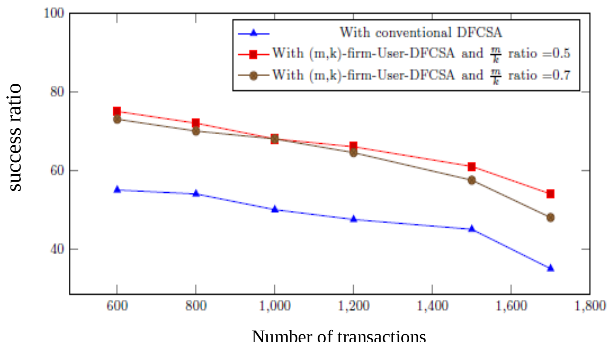

Figure 11. Simulation results using partial replication and *nb_sites*= 12.

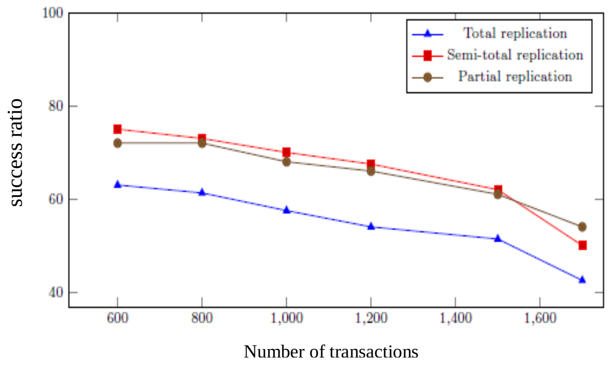

Figure 12. Three data replication with *nb_sites*=12and $\frac{m}{k}$ ratio=0.5.

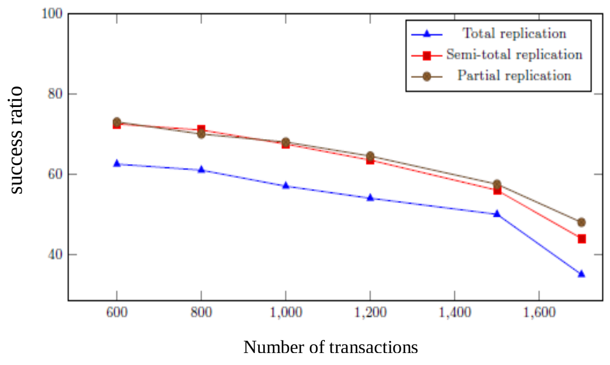

Figure 13. Three data replication with nb_sites=12and $\frac{m}{k}$ ratio=0.7.

### 4.2.3. Simulation Results for Sixteen Nodes

The last set of experiments consists of simulating the behaviour of a DRTDBMS composed of 16 nodes according to the three data replication policies. The simulation results displayed in Figures 14, 15 and 16 corresponding, respectively, to the full, semi-total and partial data replication policies in a system composed of 16 nodes, show that the (m, k)-Firm-User-DFCSA remains the most effective compared to the conventional DFCSA, and the best result is still that of the ratio equal to 0.5. As the obtained results with 12 nodes, the improvement provided by the full data replication (cf. Figure 14) is slight due to the increase of the system size where updating the copies of all data becomes costly in time which makes more transactions miss their deadline. The comparison of the three data replication types, illustrated by the Figures 17 and 18, confirms that the semi-total and the partial replication types are more beneficial, compared to the full data replication, and particularly when the system size becomes larger.

According to these results, we can say that the number of successful transactions increases by reducing the number of mandatory sub-transactions that is by reducing the value of the ratio. This allows more of transactions to be successfully executed, given that with the (m,k)-Firm-User concurrency controller, conflicts are resolved in favor of mandatory sub-transactions. In addition, with the (m,k)-Firm-User commit method, we have less requirements to validate a transaction, hence, it can be committed if at least its m mandatory sub-transactions are successfully executed. Then we can say that, with the (m,k)-Firm-User-DFCSA, transactions are executed under good conditions having more opportunity to be successfully terminated before their deadlines. This work is extended by taking account the real-time derived data that is described in the following section.

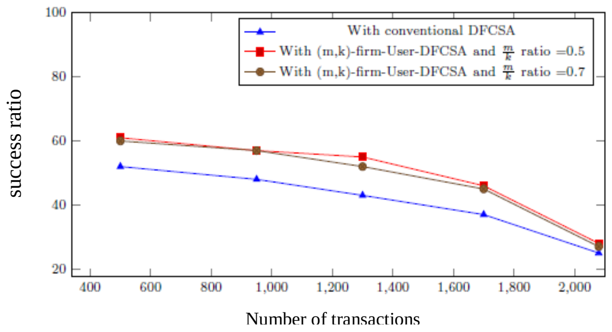

Figure 14. Simulation results using full replication and *nb_sites*= 16.

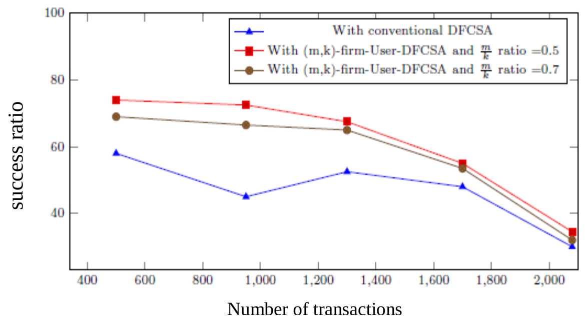

Figure 15. Results using semi-total replication and *nb_sites*=16.

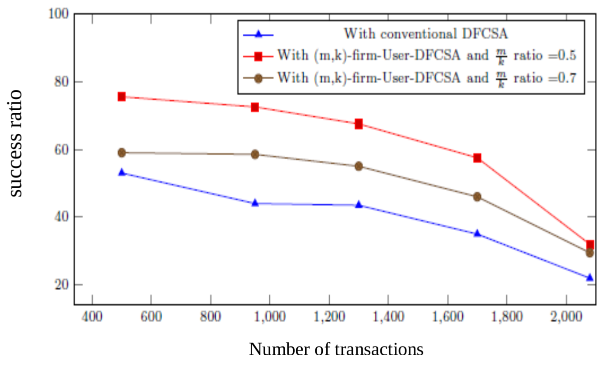

Figure 16. Results using partial replication and *nb_sites*=16.

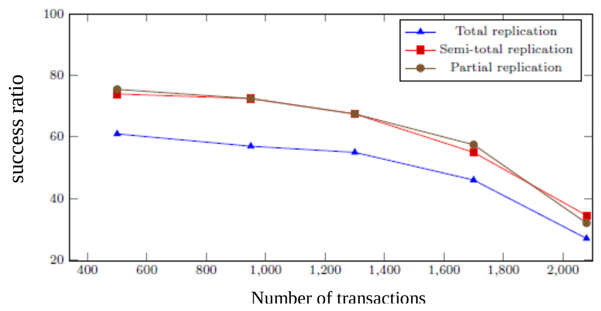

Figure 17. Three data replication with nb_sites=16 and $\frac{m}{k}$ ratio=0.5.

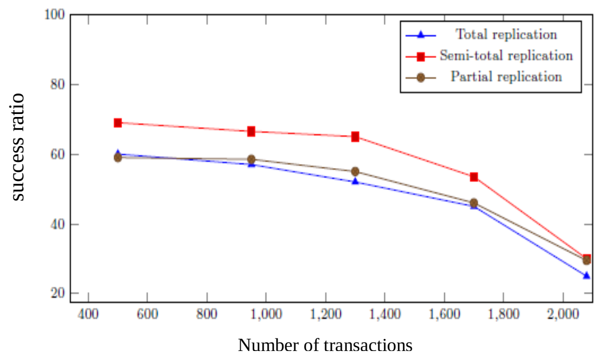

Figure 18. Three data replication with nb_sites=16 and $\frac{m}{k}$ ratio= 0.7.

## 4.3. The DDM-(m, k)-Firm-User-DFCSA Simu-Lation Results

In this section, we describe and discuss the obtained results by applying the set of experiments to evaluate the proposed DDM-(m, k)-Firm-User-DFCS Architecture. In this case, we use only one DRTDBMS's size which is sixteen nodes. The simulation results displayed in Figures 19, 20, 21, and 22 corresponding, respectively, to the full, semi-total and partial data replication in a system of 16 nodes, show that the DDM-(m, k)-Firm-User-DFCSA confirm the performances of (m, k)-firm-User-DFCS Architecture in case of using the real-time derived data, and the best result is still that of the ratio equal to 0.5 like as result without real-time derived data obtained before (cf. Section 4.2.).

Like in the proposed approach (m, k)-firm-User-DFCS Architecture, with taking into account of real-time derived data, the comparison of the three data replication types, illustrated by the Figure 22 and 23, confirms that the semi-total and the partial replication types are more beneficial, compared to the full data replication, and particularly when the system size becomes larger. In fact, this is explained by the fact that the mixed method update derived data takes into account the state of the system load. Thus this policy allows a large number of transactions executed in deadline, taking into account the state of the system through periods of overload and periods of underutilization.

According to these results, we confirm the performances of (m, k)-firm-User-DFCS Architecture in the case of taking an account the real-time derived data. Using calculated data in distributed real-time applications can decrease their performance but with using (m, k)-firm constraints, this performance is maintained and enhanced. Then we can say that, with the (m, k)-Firm-User-DFCSA and DDM-(m, k)-Firm-User-DFCSA, transactions are executed under good conditions having more opportunity to be successfully terminated before their deadlines. For that, we can say that our proposed approaches guarantee the QoS in DRTDBMS.

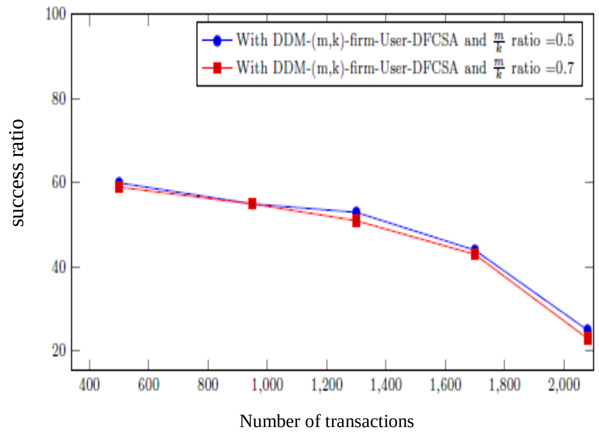

Figure 19. Simulation results using total replication and nb_sites=16.

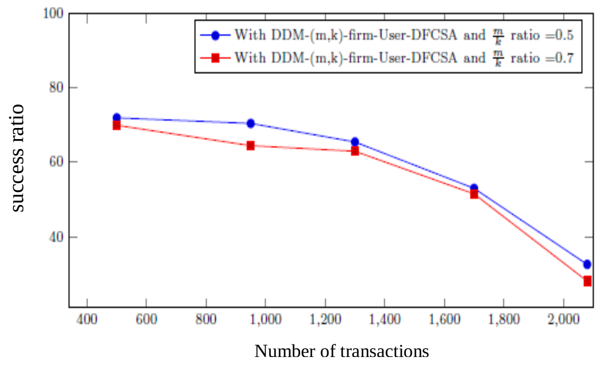

Figure 20. Simulation results using semi-total replication and nb_sites=16.

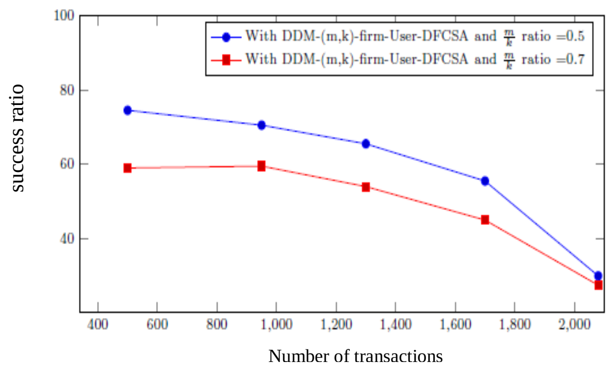

Figure 21. Simulation results using partial replication and nb_sites=16.

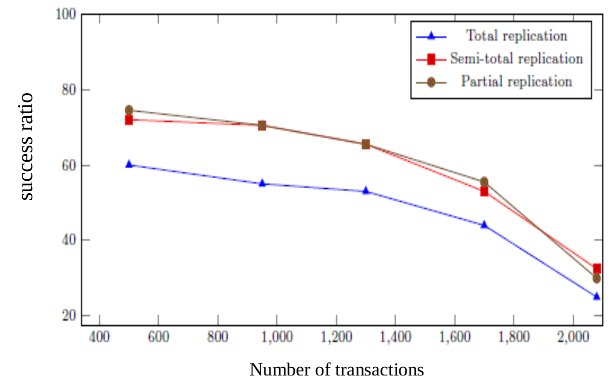

Figure 22. Three data replication with nb_sites=16 and $\frac{m}{k}$ ratio=0.5.

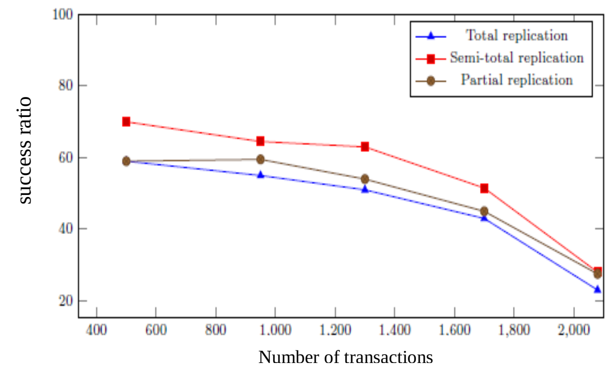

Figure 23. Three data replication with nb_sites=16 and $\frac{m}{k}$ ratio=0.7.

## 5. Conclusions and Future Work

In this paper, we have presented

1. The (m, k)-Firm-User-DFCSA using three data replication policies.
2. Its extension DDM-(m, k)-Firm-User-DFCSA for QoS improvement in DRTDBMS.

The first work consists of applying the (m, k)-firm constraints in distributed control scheduling loop under three data replication policies. In the second work, we have take into account the real-time derived data in the first proposed architecture using a mixed policy to updating real-time derived data. The obtained experimental results confirmed the benefits of the proposed approaches on increasing the number of distributed real-time transactions which meet their deadlines, even in the presence of unpredictable workload, and specially with semi-total and partial data replication policies compared to full data replication policy when the system size increase.

We plan to extend this work in several ways. In the first way, we propose to apply the (m, k)-firm constraints on update transactions. In another way, we plan to extend the proposed algorithms, for QoS improvement, so that it scales to large DRTDBMS because those algorithms are only effective for small to medium sized systems.

## References

[1] Abbott R. and Garcia-Molina H., "Scheduling Real-Time Transactions: A Performance Evaluation," *ACM Transactions on Database Systems*, vol. 17, no. 3 pp. 513-560, 1992.

[2] Achour F., Bouazizi E., and Jaziri W., "Scheduling Approach for Enhancing Quality of Service in Real-Time DBMS," *in Proceedings of 12th International Baltic Conference on Databases and Information Systems*, Riga, pp. 126-135, 2016.

[3] Almadani B., Abudalfa S., and Yang S., "QoS Adaptation for Publish/Subscribe Middleware in Real-Time Dynamic Environments," *The International Arab Journal of Information Technology*, vol. 14, no. 2, pp. 230-238, 2017.

[4] Ben Salem M., Achour F., Bouazizi E., Bouaziz, R., and Duvallet C., "Applicability of the (m, k)-firm Approach for the QoS Enhancement in Distributed RTDBMS," *in Proceedings of International Conference on Algorithms and Architectures for Parallel Processing*, Vietri sul Mare, pp. 166-175, 2013.

[5] Ben Salem M., Bouazizi E., and Bouaziz R., "Multi-Versions Data and Epsilon-Serializability for QoS Enhancement in Distributed RTDBMS," *in Proceedings of 12th IEE/ACS International Conference on Computer Systems and Applications*, Marrakech, pp. 1-6, 2015.

[6] Ben Salem M., Bouazizi E., Bouaziz R., and Duvallet C., "Quality of Service Management in Distributed Feedback Control Scheduling Architecture using Different Replication Policies," *in Proceedings of International Conference on Foundations of Computer Science and Technology*, Switzerland, pp. 75-87, 2014.

[7] Bouazizi E. and Duvallet C., "Utilisation des Contraintes (m, k)-Firm Pour la Gestion de la QdS Dans les SGBD Temps Réel," *in Proceedings of INFORSID Conference*, Lille, pp. 95-110, 2011.

[8] Diallo O., Rodrigues J., and Sene M., "Real-Time Data Management on Wireless Sensor Networks: A Survey," *Journal of Network and Computer Applications*, vol. 35, pp. 1013-1021, 2000.

[9] Duvallet C., "Prise en Compte des Données Dérivées dans les SGBD Temps Réel," *in Proceedings of the 7th Internationnal Conference on GEI*, Monastir, pp. 165-174, 2007.

[10] Haj Said S., Sadeg B., Ayeb B., and Amanton L., "The DLR-ORECOP Real-time Replication Control Protocol," *in Proceedings of 12th IEEE International Conference on Emerging Technologies and Factory Automation*, Palma de Mallorca, pp. 1-8, 2009.

[11] Hamdi S., Ben Salem M., Bouazizi E., and Bouaziz R., "Management of the Real-Time Derived Data in a Distributed Real-Time DBMS using the Semi-Total Replication Data," *in Proceedings of ACS International Conference on Computer Systems and Applications*, Ifrane, pp. 1-4, 2013.

[12] Haritsa J., Ramamritham K., and Gupta R., "The PROMPT Real-Time Commit Protocol," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 2, pp. 160-181, 2000.

[13] Haubert J., Sadeg B., and Amanton L., "(m, k)-Firm Real-Time Distributed Transactions," *in Proceedings of 16th WIP Euromicro Conference on Real-Time Systems*, Catania, pp. 61-65, 2004.

[14] Liu C. and Layland J., "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46-61, 1973.

[15] Ramanathan P. and Hamdaoui M., "A Dynamic Priority Assignment Technique for Streams with (m, k)-firm Deadlines," *IEEE Transactions on Computers*, vol. 44, no. 12, pp. 1443-1451, 1995.

[16] Serrano D., Patino-Martinez M., Jimenez-Peris, R., and Kemme B., "Boosting Database Replication Scalability Through Partial Replication and 1-Copy-Snapshot-Isolation," *in Proceedings of 13th Pacific Rim International Symposium on Dependable Computing*, Melbourne, pp. 290-297, 2007.

[17] Swaroop V. and Shanker U., "Mobile Distributed Real Time Database Systems: A Research Challenges," *in Proceedings of International Conference on Computer and Communication Technology*, Allahabad, pp. 421-424, 2007.

[18] Wei Y., Son S., Stankovic J., and Kang K., "QoS Management in Replicated Real Time Databases," *in Proceedings of 24th IEEE International Real-Time Systems Symposium. IEEE Computer Society*, Cancun, pp. 86-97, 2003.

**Malek Ben Salem** obtained his Bachelor's degree in Computer Science from Sfax University. Then, he obtained his Master's degree in Real-Time Computing Science from Sousse University and pursuing Ph.D in Computer Science in Sfax University. Currently, he is a Principal Professor at Monastir University. His current research interests are quality of service, distributed computing and real-time systems.

**Emna Bouazizi** is an Assistant Professor at the University of Monastir, Tunisia since 2012. She has defended her PhD in April 2009 on Quality of Service Management in Real-Time Databases. Her main topics of research are Real-Time Databases, Ontology specification, Geographic Information Systems and Data Warehouse.

**Claude Duvallet** is an Associate Professor in Computer Science at Le Havre Normandy University. He received his PhD in 2001. He is a member of the Intelligent Transport Systems team in the laboratory LITIS. His current research interests include real-time database systems, wireless sensor networks, vehicular ad hoc network and distributed multimedia system. He is also working on port logistics and marine traffic. He participates at many research projects on the topics of intelligent transport systems.

**Rafik Bouaziz** is Professor Emeritus on computer science at the Faculty of Economic Sciences and Management of Sfax University, Tunisia. He was the president of this University during August 2014 – December 2017, and the director of its doctoral school of economy, management and computer science during December 2011 – July 2014. His PhD has dealt with temporal data management and historical record of data in Information Systems. The subject of his accreditation to supervise research was "A contribution for the control of versioning of data and schema in advanced information systems". Currently, his main research topics of interest are temporal databases, real-time databases, information systems engineering, ontologies, data warehousing and workflows. Between 1979 and 1986, he was a consulting Engineer in the organization and computer science and a head of the department of computer science at CEGOS-TUNISIA.