# A probabilistic analysis of transactions success ratio in real-time databases

Mourad Kaddes, Majed Abdouli, Laurent Amanton, Bruno Sadeg, Alexandre Berred, Rafik Bouaziz

# A probabilistic analysis of transactions success ratio in real-time databases


## Mourad Kaddes* and Majed Abdouli

Department of Information System, FCIT,
University of Jeddah, Saudi Arabia and MIR@CL,
Pôle Technologique de Sfax BP 242, 3021, Sfax University, Tunisia
Email: mourad.kaddes@gmail.com Email: mabdouli@uj.edu.sa
*Corresponding author

## Laurent Amanton and Bruno Sadeg

LITIS,
University of Le Havre,
25 rue Philipe Lebon 76600, Le Havre, France
Email: laurent.Amanton@univ-lehavre.fr, Bruno.Sadeg@univ-lehavre.fr

## Alexandre Berred

LMAH,
University of Le Havre,
25 rue Philipe Lebon 76600, Le Havre, France
Email: alexandre.berred@univ-lehavre.fr

## Rafik Bouaziz

MIR@CL,
Pôle Technologique de Sfax BP 242, 3021, Sfax University, Tunisia
Email: rafik.bouaziz@usf.tn



**Abstract:** Nowadays, due to rapidly changing technologies, applications handling more data and providing real-time services are becoming more frequent. Real-time database systems are the most appropriate systems to manage these applications. In this paper, we study statistically the behaviour of real-time transactions under the generalised earliest deadline first scheduling policy (GEDF). GEDF is a new scheduling policy in which a priority is assigned to a transaction according to both its deadline and a parameter which expresses the importance of the transaction in the system. In this paper, we focus our study on the influence of transactions composition. Precisely, we study the influence of transaction distribution on the system performances and on approximation of transactions success ratio behaviour by a probability distribution. To this end, we have developed our RTDBS simulator and we have conducted intensive Monte-Carlo simulations.



**Keywords:** real-time databases system; RTDBS; transactions; schedule; generalised earliest deadline first; GEDF; stochastics; Monte-Carlo simulation.

## 1 Introduction

A real-time database systems (RTDBSs) can be considered as a combination of a traditional database system and a real-time system. RTDBSs have to satisfy both temporal consistency and logical consistency of the database, i.e., they must guarantee the transactions atomicity, consistency, isolation, durability *(ACID)* properties on one hand, and they must schedule the transactions in order to meet their individual deadlines, on the other hand (Han et al., 2014; Ramamritham et al., 2004).

In RTDBSs, the main issue is the transaction scheduling. In fact, different scheduling algorithms are proposed in the literature to schedule transaction in RTDBSs according to the type of knowledge used [see Han et al. (2016), Li et al. (2016), Han et al. (2012), Shanker et al. (2008)]. The most studies use *EDF* scheduling policy which is based on a priority assignment according to the deadlines.

In Kaddes et al. (2013) and Semghouni et al. (2007), have proposed a new scheduling protocol, generalised earliest deadline first (GEDF), in which transaction priority is assigned according to both deadlines and a parameter, called SPriority, which expresses the importance of transactions in order to address the weakness of EDF. In these works, authors did not take account different types of interactions with users. Hence, we propose to use different transactions size distributions and study their influence on the system performance. Moreover, we present a reasonable approximation of the success ratio of user transactions by probability density function under different size of transaction distributions.

To this purpose, we have conducted intensive Monte Carlo simulations on the RTDBS simulator we have developed. This simulator is based on components generally encountered in RTDBSs (Ramamritham et al., 2004; Kim et al., 1996).

The remainder of this paper is organised as follows. In first section, we briefly present *GEDF* policy and the simulator components and the metrics used. The second section presents the Monte Carlo simulation experiments and results. In last section, we conclude the paper and discuss some aspects of our future work.

## 2 System model and simulator

### *2.1 Simulator*

A transaction generator (TG) generates two types of transactions: user and update transactions. Each update transaction access to one sensor data. User transactions access at different data (real-time data and non-real-time data). The number of operations generated for each user transaction follows a specific distributions (Binomial, Poisson, geometric, uniform). Data accessed by the operations of the transaction are randomly generated and built according to the level of data conflicts (see following section). User transactions are submitted to the system following a Poisson process (see Figure 1) with an average rate $\lambda$, into the active queue. The *deadline controller (DC)* supervises the transactions' deadlines, and informs the *transaction scheduler (TS)* when a transaction misses its deadline in order to abort it. *Freshness manager (FM)* exploits the *absolute validity interval* (*avi*) to check the freshness of a data item before a user transaction accesses it and blocks all user transactions which read stale temporal data. Transactions data conflicts are resolved by the *concurrency controller (CC)* according to transactions priorities. *CC* informs *TS* in the following cases:

a when a transaction is finished (committed) and its results are validated in the database

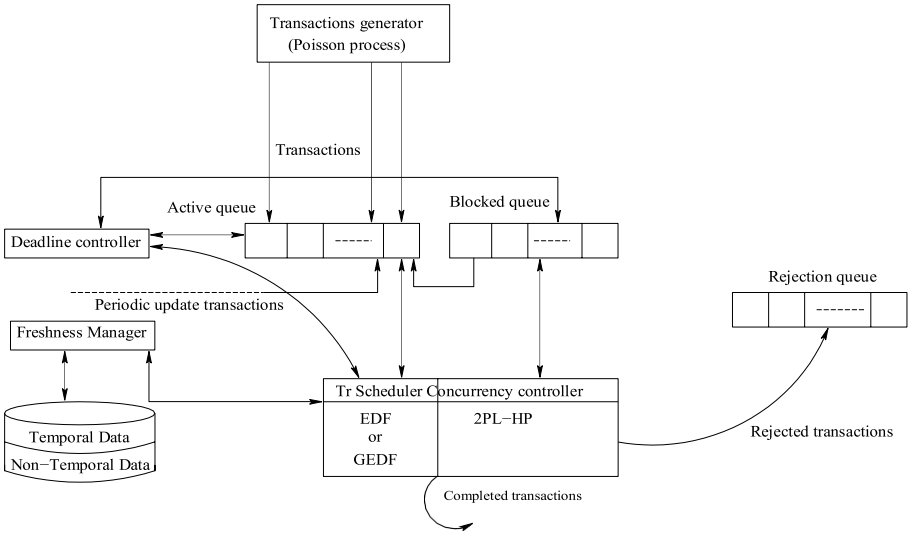
b when a transaction is blocked waiting for a conflict resolution

c when a transaction is restarted, following the commit of other transactions

d when a transaction is rejected because its restart is impossible, i.e., its best execution time (BET) is higher than its deadline minus the current time ($BET_T > DT - currenttime$)

e when a transaction is transferred from the blocked queue to the active queue, i.e., its data conflicts are resolved.

**Figure 1** Simulator architecture




## *2.2 Conflicts level*

In the database, some data are more important than others and they are frequently requested by user transactions resulting data conflicts. In order to reproduce this behaviour, a drawing probability is assigned to each data item in the following manner.

Let $r_1 < r_2 < \cdots < r_k < \cdots < r_n$, denote the ranking of the data items $D_1, D_2, \ldots, D_k, \ldots, D_n$ respectively.

The probability of drawing the data item $D_i$ is given by the following formula:

$$Prob_{D_i} = \frac{r_i}{R}$$

where $R = \sum_{i=1}^{n} r_i$, is the sum of all ranks. Thus, data with high probabilities will be more drawn than those with low probabilities. Data item $D_k$ is selected according to the above probabilities, i.e., a uniform random variable is generated U in $(0, 1)$ and select $D_k$ if $U \in (\sum_{i=1}^{k-1} Prob_{D_i}, \sum_{i=1}^{k} Prob_{D_i}]$, by convention $k = 1$ if $U \in (0, Prob_{D_1}]$. The drawing intervals of data items are resumed in Table 1.

**Table 1** Drawing intervals of data items

| Data | $D_1$ | $D_2$ | ... |
|---|---|---|---|
| Drawing interval | $[0, Prob_{D1}]$ | $]Prob_{D1}, Prob_{D1} + Prob_{D2}]$ | ... |
| Data | ... | $D_k$ ... | $D_n$ |
| Drawing interval | ... | $] \sum_{i=1}^{k-1} Prob_{D_i}, \sum_{i=1}^{k} Prob_{D_i}]$ ... | $] \sum_{i=1}^{n-1} Prob_{D_i}, 1]$ |

### *2.3 Transaction priority*

We consider only firm real-time transactions and we classify them into update and user transactions. Update transactions are periodic and only write temporal data which capture the continuously state changing environment. We assume that an update transaction is responsible for updating a single temporal data item in the system. Each temporal data item is updated following a *more-less* approach where the period of an update transaction is assigned to be more than half of the validity interval of the temporal data (Xiong et al., 2004). User transactions can read or write non-temporal data and only read temporal data. They arrive in the system according to a Poisson process with an average rate $\lambda$.

*GEDF* is a dynamic scheduling policy where transactions are processed in an order determined by their priorities, i.e., the next transaction to run is the transaction with the highest priority in the active queue. The priority is assigned according to both the deadline which expresses the criticality of time and the *SPriority* which expresses the importance of the transaction. We consider that the zero value of *Priority* ($Priority = 0$), corresponds to the highest priority in the system. A transaction $T$ is assigned a priority by the formula:

$$Priority(T) = (1 - a) \times Deadline(T) + a \times SPriority(T) \tag{2}$$

where:

- $0 \le a \le 1$, is the weight of *SPriority* in the priority formula (see Table 2).
- *SPriority* : system priority is a parameter related to each transaction. It expresses the degree of importance of the task(s) executed by a transaction and defines its rank among all the transactions in the system. Two weight functions are used according to the transaction class to assign the *SPriority* value and are described in what follows.

#### *2.3.1 Update transactions class*

The *SPriority* of an update transaction $T$ is computed according to the following formula:

$$SPriority_{update} = N \times \frac{Periode_T}{MaxPeriode} \tag{3}$$

where:

- $Periode_T$ is the period of an update transaction $T$.

- *MaxPeriode* is the longest period among the periods of update transactions.
- *N* is the value that divides the *SPriority* interval [0, $MaxValue$] according to transactions class, *i.e.* $SPriority_{update} \in ]0, N]$ and $SPriority_{user} \in ]N, MaxValue]$.

### *2.3.2 User transactions class*

The user transaction importance *SPriority* uses criteria based on both the transaction *'write set'* operations and the transaction *'read set'* operations. A user transaction *T* is assigned a *SPriority* value by the following formula:

$$SPriority_{user} = MaxValue - \gamma \times Weight_T - (1 - \gamma) \times DBA_{Value} \quad (4)$$

where

1 $Weight_T$ denotes the weight assigned to the current user transaction and is given by

$$Weight_T = (\sum_{i=1}^{n} Wread_{TD} + \sum_{j=1}^{m} Wwrite_{NTD} - \sum_{k=1}^{l} Wread_{NTD}) \quad (5)$$

where

a $Wread_{TD}$, $Wwrite_{NTD}$ and $Wread_{NTD}$ denote respectively the weight assigned to a *read operation* of a temporal data, the weight assigned to a *write operation* of a non-temporal data and the weight assigned to a *read operation* of a non-temporal data (see the transaction characteristics in Table 3).

*b* *n, m,* and *l* are respectively the numbers of $Read_{TD}$, $Write_{NTD}$, $Read_{NTD}$ operations in each user transaction.

2 $\gamma$ ]0, 1] is the rank assigned to the transaction weight in the *SPriority* formula (see Table 3).

3 $DBA_{Value}$ is a uniform random variable between 0 and ($MaxValue - N$), i.e., $Random(MaxValue - N)$.

4 $Maximum(\gamma \times Weight_T - (1 - \gamma) \times DBA_{Value}) \leq MaxValue - N$, because the user transactions *SPriority* belongs to ]*N*, $MaxValue$].

## 3 System performance metrics

To measure the system performances, we consider transaction success ratio as the main metric. The success ratio is given by:

$$SRatio = \frac{CommitT}{SubmittedT} \tag{6}$$

where *CommitT* indicates the number of transactions committed by their deadlines, and *SubmittedT* indicates all submitted transactions to the system in the sampling period. We divide this metric into two parts according to the class of transactions:

1 Success ratio of update transactions

$$SRatio_{Update} = \frac{CommitT_{Update}}{SubmittedT_{Update}} \tag{7}$$

This ratio indicates the number of update transactions committed by their deadline. It represents the consistency level of temporal data in the database.

2 Success ratio of user transactions

$$SRatio_{User} = \frac{CommitT_{User}}{SubmittedT_{User}} \tag{8}$$

This ratio indicates the number of user transactions committed by their deadlines.

## 4 Simulation results

We carried out Monte Carlo simulations that allow us to study the transactions' success ratio behaviour and the system quality of service. According to the system parameters given on Tables 2, 3 and 4, we repeat the experiment 1,000 times in each simulation in order to obtain a sample of 1,000 values for the performances.

**Table 2** Simulation parameters used for database characteristics

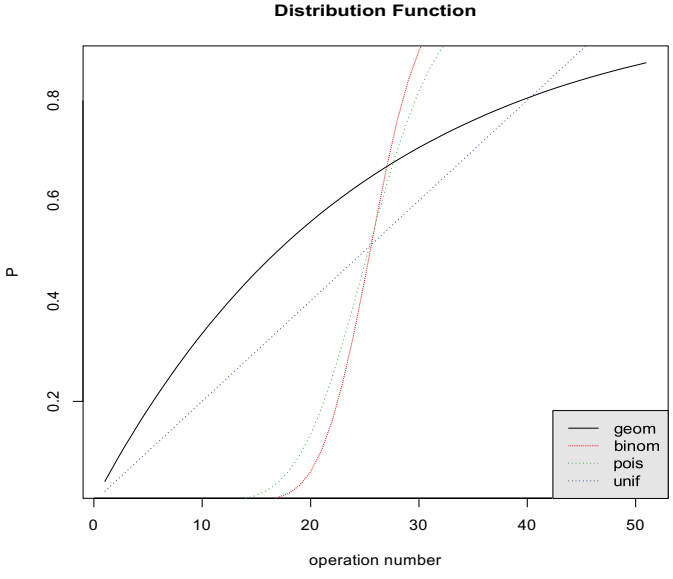
| *Database characteristics* | | |
|---|---|---|
| *Notation* | *Signification* | *Values* |
| $\lambda$ | User transaction arrival rate. | 0.1 to 2.3. |
| Time | Duration of one experiment. | 1,000 clock cycles. |
| DBSize | Number of data in the DB. | 300. |
| TD-size | Number of temporal data in the DB. | 15% × DBSize |
| Min_*avi*, Max_*avi* | Minimal and maximal *avi*. | Min_*avi*=5 clock cycles, Max_*avi*=100 clock cycles. |

### *4.1 Influence of transaction size distribution*

Applications can have different types of interactions with users. For example, many applications fixed the user request (fixed size transactions) but others applications authorise ad hoc request (variable size transaction). For this, we propose to analyse the influence of the distribution of user transactions size on performance of the system under *GEDF*.

The distribution function of user transactions size is depicted in Figure 2. It shows that, with geometric and uniform distributions, we obtain a large variation of user transactions size. Unlike with the binomial and Poisson distributions the size of transactions is homogeneous around the average. Figures 3 and 4 depicted graphically the influence of transaction size distribution on system performance under *EDF* and *GEDF* protocol respectively.

**Figure 2** User transaction size distribution (see online version for colours)

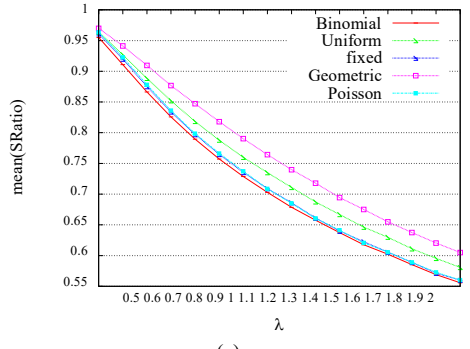



Figures 3 and 4 show clearly that transaction size distribution have less influence on system performance with GEDF. In fact, with GEDF, transaction priorities are computed using partially deadline and transaction importance. Therefore, transaction priorities are less affected by transaction distribution size. For example, when database size is 300, the difference between *SRatio* (respectively $SRatio_{user}$) obtained under various transaction size distributions, reaches 6% (respectively 15%) with *EDF* and decreases with *GEDF*: it reaches 4% (respectively 11%) when $a = 0.3$ for example. For a database size equal to 500, the difference between various $SRatio_{user}$ obtained under different distributions increases. It reaches 20% under *EDF* and 13% under $GEDF_{a=0.3}$.

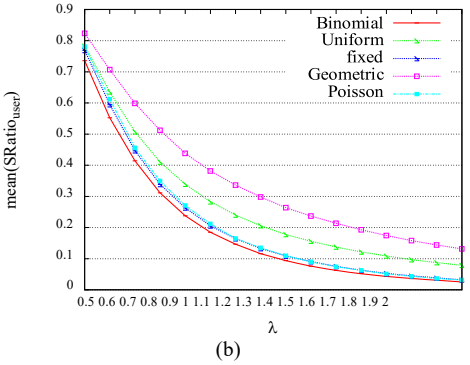
**Table 3** Simulation parameters used for transaction characteristics

| *Transaction characteristics* | | |
|---|---|---|
| *Notation* | *Definition* | *Values* |
| $\varphi$ | Probability to execute a 'read' or a'write' operation. | $\varphi(Read) = 2/3$, $\varphi(Write) = 1 - \varphi(Read) = 1/3$. |
| $Update_{size}$ | Operations in update transactions. | 1 write operation. |
| $D_{-UpT}$ | Deadline of update transaction (*more-less* approach). | $D_{-UpT} = \frac{1}{3} \times Avi.$ |
| $P_{-UpT}$ | Period of update transaction (*more-less* approach). | $P_{-UpT} = \frac{2}{3} \times Avi.$ |
| SPriority | Intervals of SPriority. | $SPriority_{Update} \in [0, 16]$ and $SPriority_{User} \in ]16, 80]$. |
| $\gamma$ | Initialisation of $\gamma$. | $\gamma = 0.8$. |
| $Wread_{TD}$ ($Wread_{NTD}$) | Reading weight of one temporal data (non-temporal data) | $Wread_{TD} = Wread_{NTD} = 1$. |
| $Wwrite_{NTD}$ | Writing weight of non-temporal data. | $Wwrite_{NTD} = 2$. |

**Figure 3** Comparison between different user transactions distributions with *EDF* protocol when $DBsize = 300$, (a) $Sratios$ (b) $Sratios_{user}$ (see online version for colours)

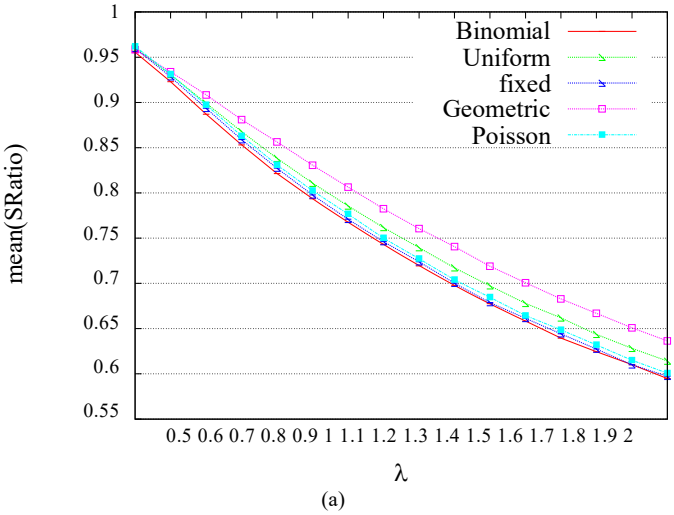



(a)

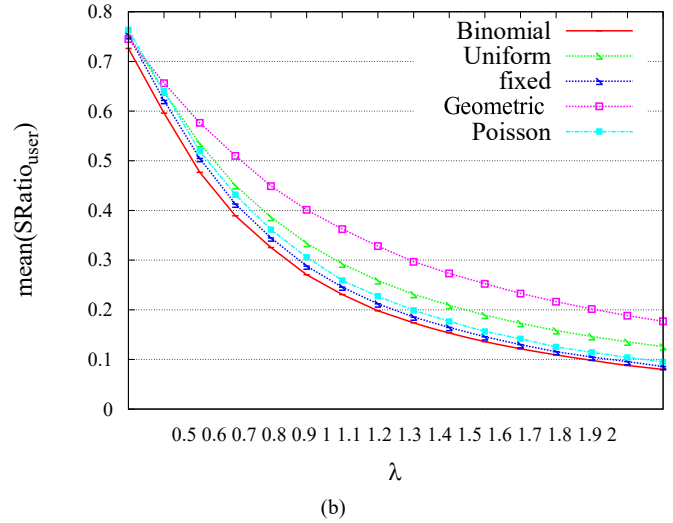



(b)

**Figure 4** Comparison between different user transactions distributions with GEDF$_{a=0.3}$ protocol when $DBsize = 300$, (a) $Sratios$ (b) $Sratios_{user}$ (see online version for colours)




(a)




(b)

With GEDF, update transactions have highest priorities, hence the success ratio of update transactions $SRatio_{update}$ is maximal, i.e., 100% for all system workload conditions. Moreover, update transactions performances are independent from user transactions, i.e., the user transactions number and size has no significant effect on the update transactions performances.

In Kaddes et al. (2013), the authors have shown that, when system is not overloaded, *EDF* gives a better success ratio than *GEDF* . This situation is reversed when system becomes overloaded. Simulations conducted using different transaction size distributions have shown that there is no significant influence on the inflection point under different

user transaction size distributions. In fact, we obtain the inflection point when $\lambda = 0.6$ and $\lambda = 0.4$, with DBsize = 300 and DBsize = 500, respectively, i.e., $SRatio_{EDF} > SRatio_{GEDF}$ when $\lambda < 0.4$ (0.6) and $SRatio_{EDF} <= SRatio_{GEDF}$ $\lambda >= 0.4$ (0.6). Furthermore, when the workload increases, the optimal value of $a$ increases too. Figure 10 illustrates graphically the comparison.

### *4.2 Stochastic behaviour of the success ratio*

In order to give a complete description of the behaviour of performance of system, we propose in this section to give a reasonable approximation of the probability density function of user transaction success ratio. Moreover, we analyse the influence of user transaction size distribution on the frequency distribution of $SRatio_{user}$ in order to refine the studies of the behaviour of user success ratio under different scheduling policies (mainly $GEDF_{a=0.3}$, $GEDF_{a=0.5}$ and $EDF$), system load and database size. To this end, we follow theses steps:

We recall that a random variable $X$ follows a standard beta distribution with parameters $p$ and $q$ if its probability density is given by:

$$f(x, p, q) = \frac{x^{p-1}(1-x)^{q-1}}{\mathrm{B}(p, q)}, \quad x \in [0, 1], \tag{9}$$

where

$$\mathrm{B}(p, q) = \int_0^1 x^{p-1}(1-x)^{q-1} dx.$$

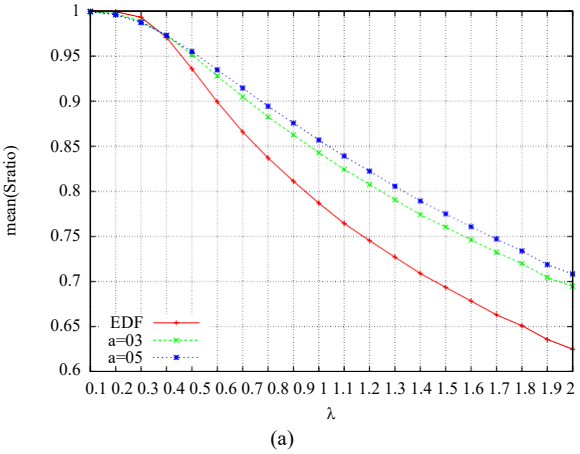
By using the moment method, we estimate the parameters $p$ and $q$ by:

$$\hat{p} = \bar{x}\left[\frac{\bar{x}(1-\bar{x})}{s^2} - 1\right] \qquad \hat{q} = (1-\bar{x})\left[\frac{\bar{x}(1-\bar{x})}{s^2} - 1\right]$$

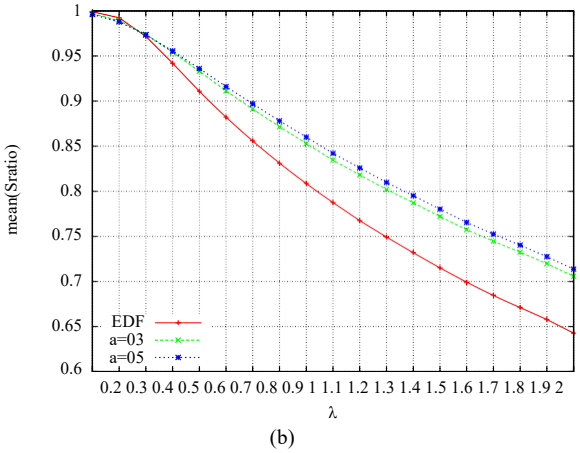
where $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ is the sample mean and $s^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$ is the sample variance.

Figures 5, 6 and 7 show some frequency distributions of the success ratio given by simulations and their approximation by beta density. Each histogram represents a sample of 1,000 values of success ratio. Figure 5 shows the histogram when using a fixed size transaction, i.e., 25 operations per transaction. Figures 6 and 7 show the histogram when the transaction size follows binomial and geometric distribution respectively. They summarise the results obtained when $DBsize = 300$, under differen t system workloads, i.e., with user transactions arrival rate $\lambda = 0.5$ (normal workload), $\lambda = 0.8$ (average workload) and $\lambda = 1.3$ (high workload).

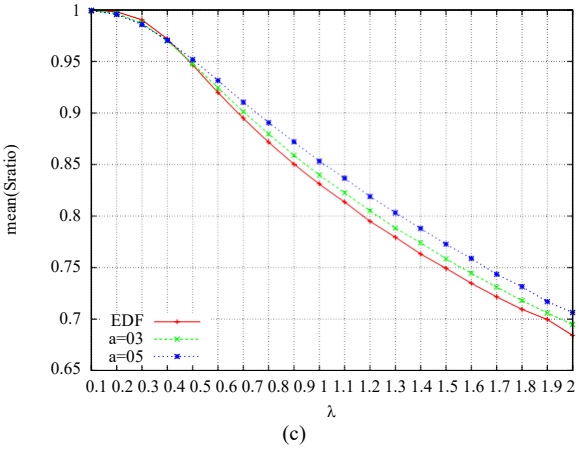
**Figure 5** $SRatio_{user}$ frequency distribution and its equivalent beta density when user transactions size is fixed, under $GEDF_{a=0.3}$, (a) $\lambda$=05 (b) $\lambda$=08 (c) $\lambda$=13 (see online version for colours)




(a)




(b)




(c)

**Figure 6** $SRatio_{user}$ frequency distribution and its equivalent beta density when user transactions size follow a binomial distribution, under $GEDF_{a=0.3}$, (a) $\lambda = 05$ (b) $\lambda = 08$ (c) $\lambda = 13$ (see online version for colours)

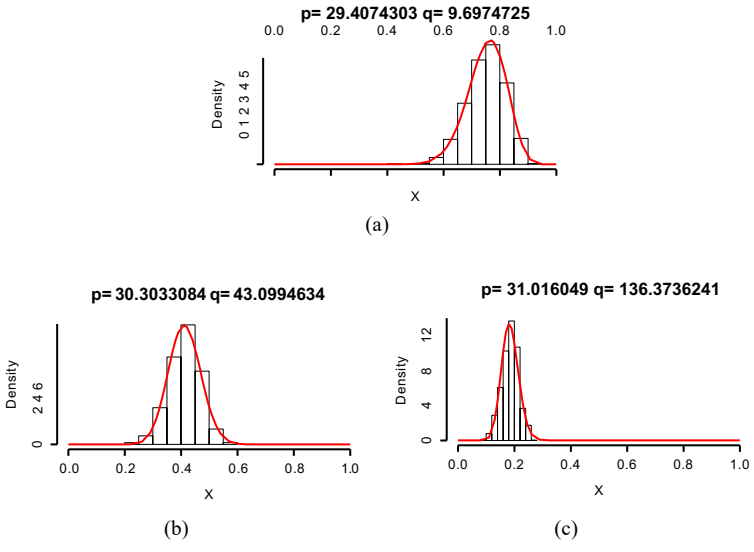



**Figure 7** $SRatio_{user}$ frequency distribution and its equivalent beta density when user transactions size follow a geometric distribution, under $GEDF_{a=0.3}$, (a) $\lambda = 05$ (b) $\lambda = 08$ (c) $\lambda = 13$ (see online version for colours)

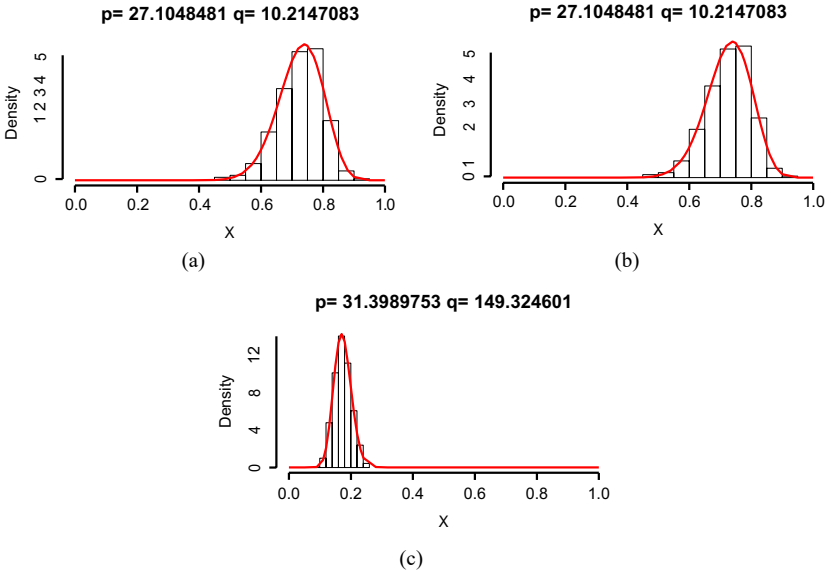
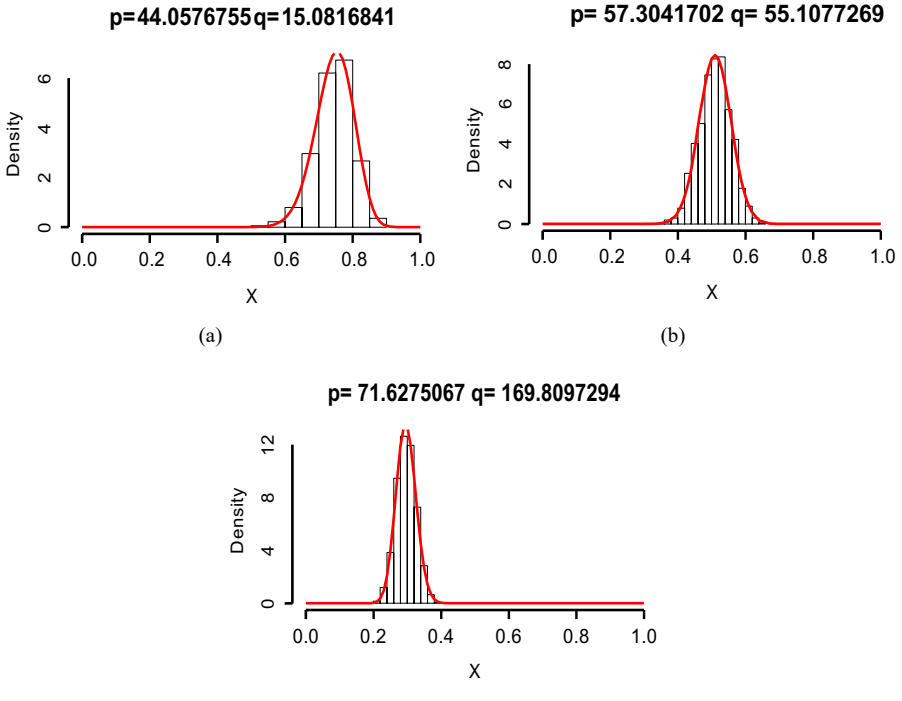


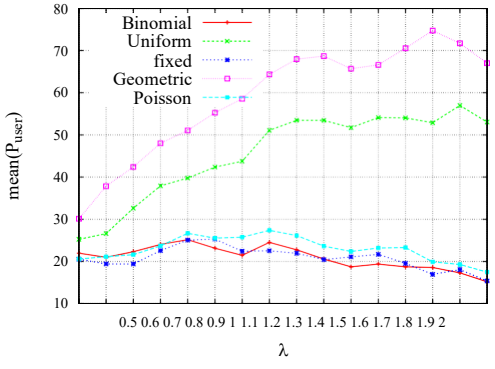
**Figure 8** Comparison of $\hat{p}_{user}$ under different user transactions size distribution, $DBsize = 300$, (a) comparison of $\hat{p}_{user}$ under $EDF$ using different distribution of user transactions (b) comparison of $\hat{p}_{user}$ under $GEDF_{a=0.3}$ using different distribution of user transactions (c) comparison of $\hat{p}_{user}$ user under $GEDF_{a=0.5}$ using different distribution of user transactions (see online version for colours)



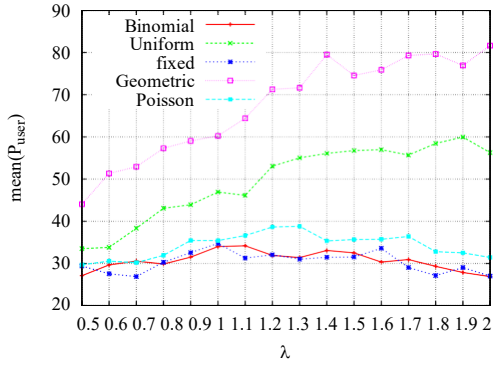

(c)

Since the histogram of $SRatio_{user}$ can be approximated by a beta density independently of the size distribution of user transaction, we have decided to analyse the influence of the size's distribution of user transactions by comparing the beta parameters ($\hat{p}_{user}$ and $\hat{q}_{user}$). Those parameters have been obtained by using the moment method, with various scheduling policies and various system workloads. Figures 9(a) and 9(b) present the evolution of $\hat{p}_{user}$ under $EDF$ and $GEDF_{a=0.3}$ when the system load increases. We observe that when the variation of transaction size is small (binomial or Poisson distributions), or null (fixed transaction size), the value of $\hat{p}_{user}$ is almost constant under different system workloads. Whereas, when the variation of user transactions size is important (geometric or uniform distribution), the $\hat{p}_{user}$ parameter grows up when the load of the system increases. We note the same behaviour of $\hat{p}_{user}$ under different scheduling policies. However, as shown in Figures 10(a) and 10(b ), the $\hat{q}_{user}$ curve appearance is not influenced by the user-transaction size distribution.

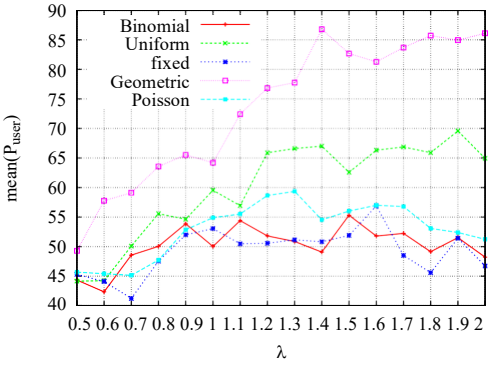
**Figure 9** Comparison of $\hat{q}_{user}$ under different user transactions size distribution, $DBsize = 300$, (a) comparison of $\hat{q}_{user}$ under $EDF$ using different distribution of user transactions (b) comparison of $\hat{q}_{user}$ under $GEDF_{a=0.3}$ using different distribution of user transactions (c) comparison of $\hat{q}_{user}$ user under $GEDF_{a=0.5}$ using different distribution of user transactions (see online version for colours)




(a)

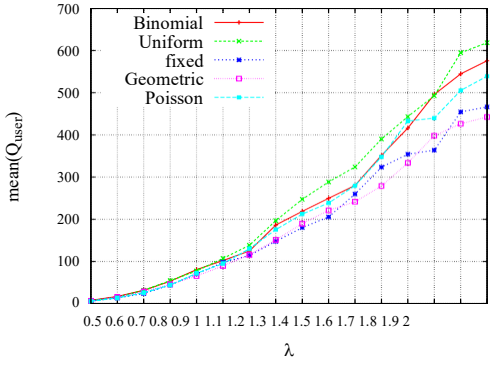



(b)

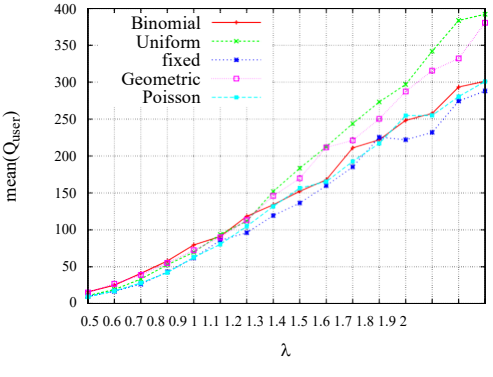



(c)

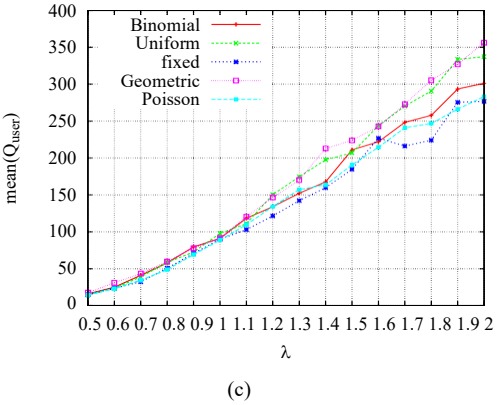
**Figure 10** Success ratio according different transactions size distributions, (a) success ratio when transactions size is fixed and $DBsize = 500$ (b) success ratio when transactions size follows a uniform distribution and $DBsize = 500$ (c) success ratio when transactions size follows a binomial distribution and $DBsize = 500$ (d) success ratio when transactions size follows a Poisson distribution and $DBsize = 500$ (e) success ratio when transactions size follows a geometric distribution and $DBsize = 500$ (see online version for colours)




(a)




(b)




(c)

**Table 4** Simulation parameters used for system characteristics

| *System characteristics* | | |
|---|---|---|
| *Notation* | *Definition* | *Values* |
| Quantum | Execution capacity in one clock cycle. | 20 Tasks/clock cycle |
| Task | Atomic action. | one Read or Write operation. |
| ReadTime | Consumption of a read operation. | 1 quantum unit. |
| WriteTime | Consumption of a write operation. | 2 quantum units. |
| $a$ | Initialisation of the parameter $a$ in formula (2) when using *GEDF*. | $a$ = 0, 0.3, 0.5. |
| SP | Scheduling policy. | '*EDF*', '*GEDF*'. |
| CC | Concurrency control protocol. | '2PL-HP'. |

**Table 5** Estimation et results of K.S. test using different user transaction size distribution

| *Distribution* | $\lambda$ | *Estimation* | | *Results of k.S. test* | |
|---|---|---|---|---|---|
| | | $\hat{p}_{user}$ | $\hat{q}_{user}$ | *distance* | *p-value* |
| Fixed size | 0.5 | 29.40743 | 9.697472 | 0.0183 | 0.8901 |
| | 0.8 | 32.57398 | 61.95368 | 0.0209 | 0.771 |
| | 1.3 | 32.07181 | 119.2753 | 0.0245 | 0.8519 |
| Binomial | 0.5 | 27.10485 | 10.21471 | 0.0202 | 0.8069 |
| | 0.8 | 31.51974 | 65.38273 | 0.0219 | 0.7221 |
| | 1.3 | 31.39898 | 149.3246 | 0.0202 | 0.8057 |
| Geometric | 0.5 | 44.05768 | 15.08168 | 0.0239 | 0.6128 |
| | 0.9 | 59.03434 | 72.4644 | 0.0176 | 0.9138 |
| | 1.3 | 71.6275 | 169.8097 | 0.0158 | 0.963 |

## 5 Conclusions

The analysis of the system performances we have conducted in this paper allows us to show that *GEDF* gives best performance when the system come overload independently of user-application interaction type *i.e.* independently of size distribution of user transactions. We have show also that *GEDF* is less affected by user transaction size distribution, i.e., with *GEDF* the performance of system is less sensible to the type of interaction with users. Furthermore, we have seen that the frequency distribution of the success ratio of the user transaction can be approximated by a beta distribution. In the future work, we plan to adapt the *GEDF* protocol to extended transactions models such as nested transaction model and analyse it's influence on the system performances.

## References


Han, S., Lam, K-Y., Wang, J., Son, S.H. and Mok, A.K. (2012) 'Adaptive co-scheduling for periodic application and update transactions in real-time database systems', *J. Syst. Softw.*, Vol. 85, No. 8, pp.1729–1743.

Han, S., Chen, D., Xiong, M., Lam, K.Y., Mo, A.K. and Ramamritham, K. (2014) 'Schedulability analysis of deferrable scheduling algorithms for maintaining real-time data freshness', *IEEE Transactions on Computers*, Vol. 63, No. 4, pp.979–994.

Han, S., Lam, K-Y., Chen, D., Xiong, M., Wang, J., Ramamritham, K. and Mok, A.K. (2016) 'Online mode switch algorithms for maintaining data freshness in dynamic cyber-physical systems', *IEEE Transactions on Knowledge and Data Engineering*, Vol. 28, No. 3, pp.756–769.

Kaddes, M., Amanton, L., Sadeg, B., Berred, A. and Abdouli, M. (2013) 'Enhancement of generalized earliest deadline first policy', in *Procedding of ICEIS 2013*, Vol. 1, pp.183–190.

Kim, Y-K. and Son, S.H. (1996) 'Supporting predictability in real-time database systems', *Proceedings of the 2nd IEEE Real-Time Technology and Applications Symposium (RTAS '96)*, pp.38–46.

Li, J., Chen, J-J., Xiong, M., Li, G. and Wei, W. (2016) 'Temporal consistency maintenance upon partitioned Multiprocessor platforms', *IEEE Transactions on Computers*, Vol. 65, No. 5, pp.1632–1645.

Ramamritham, K., Son Sang, H. and Dipippo, L.C. (2004) 'Real-time databases and data services', *Real-Time Syst.*, Vol. 23, Nos. 2–3, pp.179–215.

Semghouni, S., Amanton, L., Sadeg, B. and Berred, A. (2007) 'On new scheduling policy for the improvement of firm RTDBSs performances', *Data Knowl. Eng.*, Vol. 63, No. 2, pp.414–432.

Shanker, U., Misra, M. and Sarje, A.K. (2008) 'Distributed real-time database systems: background and literature review', *Distrib. Parallel Databases*, Vol. 23, No. 2, pp.127–149.

Xiong, M. and Ramamritham, K. (2004) 'Deriving deadlines and periods for real-time update transactions', *IEEE Transactions on Computers*, Vol. 53, No. 5, pp.567–583.