Using Experimental Measurements to Assess Dependable Adaptation Support Mechanisms for Timed Transactions[*]

Mônica Dixit
mdixit@di.fc.ul.pt
Univ. of Lisboa[†]

António Casimiro
casim@di.fc.ul.pt
Univ. of Lisboa

Nuno Laranjeiro
cnl@dei.uc.pt
Univ. of Coimbra[‡]

Marco Vieira
mvieira@dei.uc.pt
Univ. of Coimbra

Abstract

Recently, we proposed a framework that allows to analyze stochastic data, namely end-to-end measurements of distributed interactions, and hence characterize the temporal probabilistic behavior of underlying communication or transactional services, to gather fundamental information for adaptation and dependability purposes. In particular, we argue that this framework can be used to provide support for some classes of timed transactions and it allows to satisfy important dependability objectives.

This paper studies the framework in the scope of transactional environments, assessing the applicability of the proposed techniques for monitoring stochastic data, which is observed in these environments. We apply the framework using several sets of experimental field data that was collected with different kinds of transactions and in different load conditions. We compare the results with a baseline and conservative probabilistic approach and we show that the framework presents some benefits and may indeed be a useful tool to achieve more dependable timed transactions.

1. Introduction

One of the most important concepts in a database management system (DBMS) is the concept of transaction [7]. A transaction performs one or more commands within the database and the DBMS ensures that the database is always kept in a consistent state. In fact, typical DBMS transactions exhibit ACID (Atomicity, Consistency, Isolation, and Durability) properties, which are fundamental in distributed environments to deal with concurrency, failures and uncertainty on the order in which commands are executed.

However, regarding temporal or real-time properties, existing support by currently available DBMS is typically very reduced or not present at all. This kind of support could be interesting for many distributed transactional applications over the web, which have clear requirements for timeliness, or at least aim at securing some Quality of Service (QoS) levels to users. Of course, in order to add timeliness properties to DBMS, some additional low level functionalities are needed, and also it becomes necessary to at least monitor the QoS that is available from the underlying communication and computational infrastructure.

In the TACID (Timely ACID transactions in DBMS) project [15] we addressed the problem, identifying different kinds of timed transactions and studying the mechanisms that are needed to support them. Three transaction classes are considered in TACID: a) with no temporal requirements, that is, normal ACID transactions; b) with strict temporal requirements, meaning that the transaction has to be concluded within a specified time frame, otherwise the system must detect the timing failure and take some action upon it (corresponding to hard real-time constraints); c) with relaxed temporal requirements, meaning that a time frame is specified and expected to be met with a given probability, while the transaction will always execute (corresponding to soft real-time constraints). The results presented in this paper are essentially relevant for this last class, which requires the capability for monitoring the system and taking the adequate adaptation decisions.

A general problem of adaptive systems in probabilistic environments is how to characterize the actual state of the environment. More specifically, the problem is to determine a good probabilistic description for fundamental stochastic variables, like the message or packet transmission delay, the execution time of a certain function, or the whole time needed to execute a particular transaction. Then, having determined a probabilistic curve using available monitoring data, it is possible to make decisions on how to adapt to achieve some goal, for instance, which deadline to assume in order to ensure timely completion of a transaction.

In another work [1] we introduced a framework that is

[*]This work was partially supported by the FCT, through project POSC/EIA/61568/2004 (TACID), and by the EC, through project IST-FP6-STREP-26979 (HIDENETS).

[†]Faculdade de Ciências da Universidade de Lisboa. Bloco C6, Campo Grande, 1749-016 Lisboa, Portugal. Navigators Home Page: http://www.navigators.di.fc.ul.pt.

[‡]DEI/CISUC, Faculdade de Ciências e Tecnologia da Universidade de Coimbra, Polo 2 - Universidade de Coimbra, 3030-290 Coimbra, Portugal. Home Page: http://cisuc.dei.uc.pt.

aimed at addressing this problem and helping system architects in building adaptive and dependable systems in probabilistic environments. This framework makes some assumptions about the probabilistic nature and dynamics of the monitored data, which we believe are reasonable for a large range of environments and applications, therefore rendering the solutions applicable in all those cases.

In this paper we consider the particular case of transactional systems, and the need for a middleware to support the class of TACID transactions with relaxed temporal requirements. Using several sets of experimental data that were collected using different transactions and under different load situations (with varying number of concurrent database clients), we applied the monitoring framework to assess its applicability to such a transactional environment. Based on the results we were able to verify the improvements that may be obtained when comparing our framework with a conservative probabilistic approach. Moreover, we observed some differences in terms of probabilistic behavior between the different kinds of transactions, allowing us to conclude about the situations in which the framework is more suitable. Finally, we also did some comparisons with a very simple ad hoc technique to support adaptation. This allowed us to formulate an interesting argument about the trade-off between using a more complex but also more dependable approach and a simple but not dependable approach.

The paper is organized as follows. In the next section we briefly introduce some related work that sets the context for the paper. Then, Section 3 explains what are timed transactions and their properties. The monitoring framework and the probabilistic techniques that we use are described in Section 4 and in Section 5 we present our evaluation based on the collected field data. Section 6 concludes the paper.

2. Related Work

Dealing with time and timeliness in the context of transactional systems is not a new issue. These problems have been identified and studied in the context of temporal [8, 9] and also real-time databases [10, 11, 14]. In the former, the idea is to have time-varying data stored in the databases, requiring time to be used as an additional criteria when executing transactions and querying data. This is still a very active area of research [13]. In real-time databases the idea is to ensure that transactions execute within some well-known deadlines. This necessarily requires special attention with how system support issues, for instance with how transactions are scheduled.

Measuring transactions execution time typically requires the instrumentation of both database clients and database servers. In fact, typical database environments do not include the mechanisms required to accurately measure transactions execution time, which can be done locally (on the client side or on the server side), or in a distributed way, considering that a transaction starts when the client application submits the first command and ends immediately after the server finishes the execution of the last command (and not when the client application receives the response).

In [6] it is described a transaction time measurement mechanism, which includes a time manager that runs on a server computer system, a time agent running on a client computer system, and a simple protocol for allowing them to communicate directly and efficiently. The Timely Computing Base (TCB) model [18] has been used in the TACID project to measure the duration of transactions both in a local and distributed way. The TCB is based on an improved round-trip technique that guarantees not only bounded, but also almost stable measurement errors [2].

Regarding the probabilistic characterization of stochastic variables (typically message transmission delays or round-trip time delays), there is considerable work addressing this problem and well-known approaches and mechanisms (see [19] for a nice overview). Among others, we can find approaches based on time-exponentially weighted moving histograms [12], on the Kolmogorov-Smirnov test [5] or the Mann-Kendall test. In our work we have used different approaches simultaneously, combining them in a single framework. Therefore, the chances that we achieve a better characterization of the actual state of the environment are higher. More details about the framework are provided in Section 4.

3. Timed Transactions

In the TACID project we have considered three classes of timed transactions. The class of relevance here is the one with relaxed temporal requirements, which may not always execute within some specified time bound, but whose probability of executing within the specified deadline is guaranteed. This implies that the assumed deadline may be adapted to follow probabilistic changes in the environment.

This kind of transactions, which involved adaptation of assumed bounds during the execution, is mostly suitable to what we call a *time-elastic class* of applications. These applications are typically able to decrease the transaction submission rate, increase the transactions deadline if possible, or postpone the execution of transactions to a latter time. Clearly, these applications can be based on transactions with relaxed temporal requirements. Examples of this class include databases that control mobile communication systems, where connection establishment can tolerate some delays (or may be refused) and billing transactions can be postponed, or continuous manufacturing processes such as chemical processes.

In these applications, the occurrence of timing failures does not compromise the correctness of the database application. Typically, what is important is to secure that the number of timing failures stays below a given bound, over an interval of mission. In other words, what is important to ensure is that a certain QoS is maintained, expressed in terms of a timing bound to be secured with a given (usually stable) coverage (probability that the timing failure does not occur). Therefore, provided that the application is aware of the actual coverage of the timing assumption, it may undertake one of the above-mentioned adaptation strategies, if necessary, when this coverage changes.

4. Evaluation Framework

The evaluation framework introduced in [1] is based on the assumption that the system behaves probabilistically, alternating periods during which the conditions of the environment remain fixed (*stable phases*), with periods during which the environment conditions change (*transient phases*). During stable phases, the statistical process that generates the data flow (e.g., the execution time of some transaction) is under control and then we can compute the corresponding distribution using an appropriate number of samples. On the contrary, if the environment conditions are changing, then the associated statistical process is actually varying, so no fixed distribution can describe its real behavior.

The scheme depicted in Figure 1 shows the architecture of the framework. It was modeled as a service that: (i) accepts the history size (i.e., the number of collected samples of the random variable under observation) and the required coverage as dependability related parameters. The coverage represents the probability that the assumed bounds are safe, i.e., the probability that no timing faults will occur; (ii) reads samples (e.g. measured execution times) as input, using them to fill up the history buffer that is used by the phase detection mechanisms and for the estimation of distribution parameters; and (iii) provides, as output, a time bound that should be used in order to achieve the specified coverage.

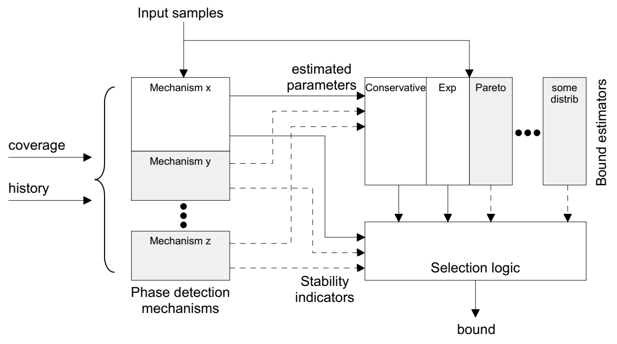

Figure 1. Schematic view of the framework for dependable adaptation.

The framework is composed by two goodness-of-fit (GoF) tests as phase detection mechanisms: the Kolmogorov-Smirnov (KS) test and the Anderson-Darling (AD) test. GoF tests are formal statistical procedures used to assess the underlying distribution of a data set. A stable period with distribution $\hat{D}$ is detected when some GoF tests establish the goodness of fit between the postulated distribution $\hat{D}$ and the evidence contained in the experimental observations [17].

Both AD and KS are distance tests based on the comparison of the cumulative distribution function (CDF) of the assumed distribution $\hat{D}$ and the empirical distribution function (EDF), which is a CDF built from the input samples. We are considering four distributions to be tested: exponential, shifted exponential, Pareto and Weibull. If the phase detection mechanisms do not identify a stable phase with none of these distributions, they assume that the environment is changing, i.e., a transient phase is detected.

In order to execute their statistical tests, the phase detection mechanisms need to estimate distributions parameters. Table 1 presents the parameters estimators, where $\{x_1, x_2, ..., x_n\}$ is the *ordered* history sample, $y_i = \ln(1 - F(x_i))$, and $F(x_i) = (i - 0.3)/(n + 0.4)$ (approximation of the median rankings).

Table 1. Equations for parameters estimation

Distribution	Parameters estimators
Exponential	$\hat{\lambda} = \frac{1}{\bar{x}}$
Pareto	$\hat{k} = x_{min}$ $\hat{\alpha} = \frac{n}{\sum_{i=1}^{n} \ln \frac{x_i}{\hat{k}}}$
Shifted Exponential	$\hat{\lambda} = -\frac{\sum_{i=1}^{n} x_i y_i - \frac{\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n}}{\sum_{i=1}^{n} x_i^2 - \frac{(\sum_{i=1}^{n} x_i)^2}{n}}$ $\hat{\gamma} = \frac{\bar{y} + \hat{\lambda}\bar{x}}{\hat{\lambda}}$
Weibull	$\hat{\gamma} = \frac{\sum_{i=1}^{n} x_i y_i - \frac{\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n}}{\sum_{i=1}^{n} x_i^2 - \frac{(\sum_{i=1}^{n} x_i)^2}{n}}$ $\hat{\alpha} = e^{-\frac{\bar{y} - \hat{\gamma}\bar{x}}{\hat{\gamma}}}$

The method of maximum likelihood estimation (MLE) is used to estimate exponential and Pareto parameters. This method selects as an estimation of a parameter θ the value for which the observed sample is most "likely" to occur [17]. However, for the Weibull and shifted exponential distributions the MLE method does not have a closed form solution: the equations must be simultaneously solved using iterative algorithms, which are very time-consuming. For this reason, the Weibull and shifted exponential parameters are estimated through linear regression. The implemented solution uses the method of least squares, which requires that a straight line be fit to a set of data points, minimizing the sum of the squares of the y-coordinate deviations from it [17].

Depending on the output of the phase detection mechanisms, one of the bounds calculated by the implemented bound estimators will be selected as the output of the framework. The current implementation has five bound estimators: the conservative one, which is based on the one-sided inequality of probability theory (using the estimated mean E(D) and variance V(D)), providing a pessimistic bound which holds for all probabilistic distributions; and estimators for the exponential, shifted exponential, Pareto and Weibull distributions, which are derived from the distributions CDF. The bound estimators are presented in Table 2.

Table 2. Bound estimators for coverage C

Estimator	Minimum time bound t
Conservative	$t = E(D) + \sqrt{\frac{V(D)}{1-C} - V(D)}$
Exponential	$t = \frac{1}{\lambda} \ln \frac{1}{1-C}$
Pareto	$t = \frac{k}{\sqrt[\alpha]{1-C}}$
Shifted Exponential	$t = \gamma + \frac{1}{\lambda} \ln \frac{1}{1-C}$
Weibull	$t = \alpha \sqrt[\gamma]{\ln \frac{1}{1-C}}$

Finally, the selection logic receives the mechanisms results and selects one of them as the output of the framework (see Figure 1). The selection is performed based on the bound estimators, and might have different levels of aggressiveness.

A complete description of the framework, including all the implementation details, complexity analysis and an extensible evaluation can be found in [4].

5. Evaluation of Field Data

5.1. Experimental set-up

In our experiments we used an implementation of the *TPC-App* performance benchmark [16], which is widely accepted as representative of real transactional environments. This benchmark simulates the activities of a business-to-business on-line distributor that heavily relies on database transactions to support on-line user ordering and browsing. It has support for multiple on-line business sessions and enables the execution of multiple transaction types that represent a wide variety of business functions. A subset of the transactions specified by *TPC-App* was used to demonstrate our framework: *changePayment*, *newCustomer*, *newProduct*, and *productDetail*.

The *TPC-App* benchmark includes three components: i) a remote business emulator (i.e., a multiple client emulator); ii) an application server that provides business functions; and iii) a database server. For sake of completeness, Table 3 describes the software and hardware used in our experiments. A Fast Ethernet network connected the three components.

Transactions execution times were measured from the application server point-of-view and under different client loads (1, 2, and 4 simultaneous client sessions). Different client loads were used so that we could test our framework with data from different transactional conditions.

Table 3. Systems used for the experiments

Node	Software	Hardware
App. Server	Windows Server 2003 R2 Enterprise x64 Edition SP2 & JBoss 4.2.2.GA	Dual Core Pentium 4 3Ghz 1.46GB RAM
Database Server	Windows Server 2003 R2 Enterprise x64 Edition SP2 & Oracle 10g	Quad Core Intel Xeon 5300 Series 7.84 GB RAM
Client	Windows XP Pro SP2	Dual Core Pentium 4, 3GHz, 1.96GB RAM

5.2. Evaluation goals

The objective of this work is to demonstrate the applicability of the previously proposed framework in transactional environments. In this section we show that our probabilistic assumption is realistic for transactional environments and demonstrate that the framework is able to characterize these environments and provide applications with dependable estimation of transactions execution times. The experimental data collected as described in Section 5.1 is used to fill up the history buffer provided to the framework and to compare the produced bounds with the real ones in order to evaluate the framework performance.

Moreover, we justify the use of our framework by comparing its results with two alternative methods: the first one is a general probabilistic (conservative) approach proposed in [3], which just assumes a probabilistic environment but not a specific probabilistic distribution (the framework uses this method to calculate dependable bounds in transient periods); and the second approach estimates transactions execution times based on the average of historical data.

5.3. Results and Discussion

We executed the framework for each measured data trace. The results presented in this section were obtained using the following parameters: history size $h = 50$[1], and required coverage $C = 98\%$[2]. Three metrics were used to evaluate the framework performance: (i) *Stable phases*, which is the percentage of the sample points that were detected as stable phases, This metric allows to validate our assumption regarding the probabilistic environment behavior; (ii) *Improvement of bounds*, which defines the improvement obtained with the framework in comparison with the conservative approach. The improvement is calculated based on the difference between the bounds produced by these two methods; and (iii)

[1]Preliminary tests using different history sizes shown that h=50 presented the best overall results in terms of achieved coverage and reduction of bounds for the collected traces.

[2]We consider acceptable that 2% of the soft real-time transactions are not completed within the specified time bound, specially because the lateness degree is normally very small, causing a low impact on the end users perception of the overall system performance.

Coverage, which represents the achieved coverage and proves that we reached dependable adaptation.

Each collected data trace is composed by the execution times of different transactions performed by one specific client. The framework was executed using these traces individually, and the average results are presented in Table 4.

Table 4. Results for $C = 98\%$ and $h = 50$

Client	Stable	Coverage	Improv.
1 of 1	82.7%	99.6%	21.9%
1 of 2	88.5%	99.2%	26.1%
2 of 2	89.0%	99.1%	25.9%
1 of 4	81.5%	99.1%	20.9%
2 of 4	80.9%	99.1%	20.5%
3 of 4	84.3%	98.8%	22.4%
4 of 4	84.0%	98.8%	23.0%

For all experiments, the framework detected stable phases in more than 80% of the sample points, which is a strong indicator that there are transactional systems in which our assumption about the probabilistic behavior of the environment is realistic. The correctness of the implemented methods is verified by the achieved coverage: the minimum desired coverage (98%) was secured in all executions. Comparing the (adaptive) time bounds produced by the framework with the (pessimistic) bounds produced by the conservative approach, the adaptive bounds were at least 20% lower than the pessimistic bounds. Since that the desired coverage was secured, these tighter bounds can be applied to dependably improve the detection time of timing faults.

As explained in Section 5.1, we measured the execution times of four different transactions. In order to analyze the impact of the transaction types on the framework results, we separated the collected data traces by transactions, i.e., each original trace generated four different traces, one for each transaction type. Table 5 shows the average results.

The *changePayment* transaction presented the best results: all sample points were characterized as stable periods, the desired coverage was secured and the average of the adaptive bounds were approximately 50% lower than the pessimistic bounds. The framework also achieved very good results for the *productDetail* transaction, detecting stable phases in more than 90% of the sample points and guaranteeing the expected coverage with the adaptive bounds.

The rates of stable phases detection were smaller for the *newCustomer* and *newProducts* transactions. In spite of that, the coverage was secured in all executions, suggesting that the probabilistic assumption holds, but the framework is not able to characterize the environment as stable. This indicates that the execution times actually follows some distribution (which can be a well-known or even a mixed distribution) which is not verified by the framework (current implementation tests the exponential, shifted exponential, Pareto and Weibull distributions). Given that there is less detection of stable phases, the average improvement of bounds is also smaller for these transactions.

One can argue why we decided to use such elaborated statistical methods instead of some basic mathematical method, for instance, based on the simple average of the last recently measured execution times. We implemented a very simple method which estimates transactions execution times as the average of the history sample multiplied by a given safety factor. The safety factor is essential in order to guarantee the expected coverage. We performed a set of experiments to evaluate this approach. Table 6 presents the smallest possible safety factor for each transaction type, the achieved coverage when using this factor and the improvement obtained comparing the produced bounds to the adaptive bounds calculated by our framework.

Although this simple approach can produce better bounds than our framework, it has two main disadvantages. The first one is that the safety factor must be carefully adjusted to allow the method to produce adequate bounds while securing the desired coverage. In our experiments, we verified that this factor depends on the transaction type and on the number of clients in the system. Moreover, in a few cases the safety factor varies among different clients. We verified this situation for the *productDetail* transaction having four clients: the best safety factor was 5.0 for one client, 5.5 for another one and 3.5 for the remaining two clients. We decided to use the highest factor for dependability purposes, and the final result was that the bounds produced using this simple method were in average higher than the adaptive bounds.

The second and more serious disadvantage of this approach is that **it is not dependable**, in the sense that we cannot specify the necessary conditions to correctly apply it. On the other hand, our work is concerned with dependability objectives: we defined a set of reasonable assumptions and we guarantee that our solution is dependable as long as these assumptions hold.

The presented results suggest that it is possible to achieve dependable adaptation of time bounds in transactional systems, using well-established probabilistic mechanisms. Obviously, the relevance of the obtained improvements depends on the cost of using such mechanisms. In [4] we performed a complexity analysis which shown that our solution presents satisfactory performance and execution times.

6. Conclusion

In this paper we studied and discussed the applicability in transactional environments of a monitoring framework that was designed for monitoring stochastic data. Using traces collected from an experimental setup in which typical transactional activities take place, we were able to raise awareness about the expectable coverage of assuming a specific time bound for the duration of each kind of transaction, and compare these results with two baseline approaches, one in which

Table 5. Results for different transactions ($h = 50$)

Transaction	1 client			2 clients			4 clients		
	Stable	Coverage	Improv.	Stable	Coverage	Improv.	Stable	Coverage	Improv.
changePayment	100.0%	99.4%	53.2%	100.0%	99.5%	48.0%	100.0%	99.2%	44.7%
newCustomer	33.4%	99.3%	20.3%	37.9%	99.4%	17.8%	51.2%	98.8%	16.2%
newProducts	7.2%	98.7%	2.5%	23.8%	98.6%	6.7%	40.1%	98.2%	10.2%
productDetail	97.9%	99.3%	39.9%	99.4%	99.1%	38.8%	90.1%	99.3%	26.2%

Table 6. Results using simple average $\times$ safety factor ($h = 50$)

Transaction	1 client			2 clients			4 clients		
	Factor	Coverage	Improv.	Factor	Coverage	Improv.	Factor	Coverage	Improv.
changePayment	5.0	98.8%	10.5%	4.5	98.5%	19.4%	5.0	98.4%	10.5%
newCustomer	2.5	98.1%	50.4%	2.5	98.3%	45.1%	3.0	98.2%	33.0%
newProducts	2.0	98.2%	43.7%	2.5	98.4%	38.8%	2.5	98.2%	46.1%
productDetail	3.0	99.2%	5.2%	3.0	98.6%	12.4%	5.5	99.0%	-2.4%
All	2.5	98.2%	47.9%	3.5	98.7%	26.9%	4.0	98.2%	31.4%

a pessimistic probabilistic approach is used to determine time bounds, and another in which a straightforward but ad hoc and not mathematical approach is used.

The presented results are interesting in the sense that they show the possibility of using a dependable approach to select temporal bounds in the implementation of timed transactions. They also provide quantitative data to better evaluate the trade-off between using a dependable and more complex approach and a non dependable but simpler approach.

References

[1] A. Casimiro, P. Lollini, M. Dixit, A. Bondavalli, and P. Veríssimo. A Framework for Dependable QoS Adaptation in Probabilistic Environments. In *23rd ACM Symp. on Applied Computing, Dependable and Adaptive Distributed Systems Track*, pages 2192–2196, 2008.

[2] A. Casimiro, P. Martins, P. Veríssimo, and L. Rodrigues. Distributed Durations with Stable Errors. In *Procs. of the 22nd IEEE Real-Time Systems Symp.*, pages 310–319, 2001.

[3] A. Casimiro and P. Verissimo. Using the Timely Computing Base for Dependable QoS Adaptation. In *Procs. of the 20th IEEE Symp. on Reliable Distributed Systems*, pages 208–217, 2001.

[4] M. Dixit, A. Casimiro, P. Lollini, A. Bondavalli, and P. Verissimo. A Probabilistic Framework for Automatic and Dependable Adaptation in Dynamic Environments. *Submitted to IEEE Trans. on Computers, Special Issue on Autonomic Network Computing*, 2008.

[5] T. Elteto and S. Molnar. On the Distribution of Round-Trip Delays in TCP/IP Networks. In *Conf. on Local Computer Networks*, pages 172–181, 1999.

[6] R. F. Forman, J. M. Pechacek, and W. H. Schwane. US Patent 6178449 - Apparatus and Method for Measuring Transaction Time in a Computer System, Jan. 2001.

[7] J. Gray and A. Reuter. *Transaction Processing: Concepts and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Jim Gray, 1993.

[8] C. S. Jensen and L. Mark. Queries on Change in an Extended Relational Model. *IEEE Trans. on Knowl. and Data Eng.*, 4(2):192–200, 1992.

[9] S.-K. Kim and S. Chakravarthy. Temporal Databases with Two-Dimensional Time: Modeling and Implementation of Multihistory. *Inf. Sci. Inf. Comput. Sci.*, 80(1-2):43–89, 1994.

[10] Y.-K. Kim and S. H. Son. An Approach Towards Predictable Real-Time Transaction Processing. In *Procs. of the 5th Euromicro Workshop on Real-Time Systems*, pages 70–75, Jun 1993.

[11] K.-Y. Lam and T.-W. Kuo. *Real-Time Database Systems: Architecture and Techniques*. Kluwer Academic Publishers, USA, 2001.

[12] M. Menth, J. Milbrandt, and J. Junker. Time-Exponentially Weighted Moving Histograms (TEWMH) for Application in Adaptive Systems. In *Procs. of the Global Telecommunications Conference*, pages 1–6, Nov. 2006.

[13] D. Papadias, D. Zhang, and G. Kollios, editors. *Advances in Spatial and Temporal Databases, 10th Int. Symp., SSTD 2007, Boston, MA, USA, July 16-18, 2007, Procs.*, volume 4605 of *Lecture Notes in Computer Science*. Springer, 2007.

[14] K. Ramamritham. Real-time Databases. *International Journal of Distributed and Parallel Databases*, 1:199–226, 1993.

[15] TACID (Timely ACID Transactions in DBMS) web page. http://gbd.dei.uc.pt/view_project.php?id_p=55.

[16] Transaction Processing Performance Council. TPC BenchmarkTM App (Application Server) Standard Specification, Version 1.3, 2008. Web page: http://www.tpc.org/tpc_app.

[17] K. S. Trivedi. *Probability and Statistics with Reliability, Queuing and Computer Science Applications*. John Wiley and Sons, 2002.

[18] P. Verissimo and A. Casimiro. The Timely Computing Base Model and Architecture. *IEEE Transactions on Computers*, 51(8):916–930, 2002.

[19] M. Yang, X. R. Li, H. Chen, and N. S. V. Rao. Predicting Internet End-to-End Delay: An Overview. In *Proc. of the 36th IEEE Southeastern Symposium on Systems Theory*, pages 210–214, Mar. 2004.