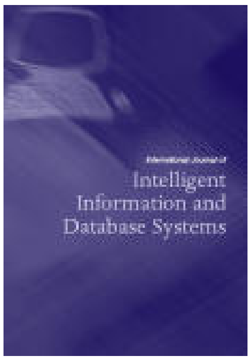


International Journal of Intelligent Information and Database Systems

ISSN online: 1751-5866 - ISSN print: 1751-5858
https://www.inderscience.com/ijiids

A semantics-based validation approach for enhancing QoS in distributed real-time DBMS

Fehima Achour, Emna Bouazizi, Wassim Jaziri

DOI: 10.1504/IJIIDS.2025.10068164

Article History:

Received:	25 December 2023
Last revised:	29 February 2024
Accepted:	28 March 2024
Published online:	23 December 2024

Copyright © 2025 Inderscience Enterprises Ltd.

A semantics-based validation approach for enhancing QoS in distributed real-time DBMS

Fehima Achour*

MIRACL Laboratory,
The Higher Institute of Computer Science and Multimedia,
P.B. 242, Sfax 3021, Tunisia
Email: fehima.achour@gmail.com
*Corresponding author

Emna Bouazizi

MIRACL Laboratory,
The Higher Institute of Computer Science and Multimedia,
P.B. 242, Sfax 3021, Tunisia
and
The College of Computer Science and Engineering,
University of Jeddah,
Jeddah, Saudi Arabia
Email: AALTAHER@uj.edu.sa

Wassim Jaziri

MIRACL Laboratory,
The Higher Institute of Computer Science and Multimedia,
P.B. 242, Sfax 3021, Tunisia
and
King Faisal University,
Al Ahsa, KSA
Email: jaziri.wassim@gmail.com

Abstract: Distributed real-time database management systems (DRTDBMS) are essential for critical applications such as financial trading and healthcare, where timely processing is vital. Quality of service (QoS) approaches, like the distributed feedback control scheduling (DFCS) architecture, are favoured for improving DRTDBMS performance. However, transactions scheduling remains a challenge in such systems, impacting user's experience. Recently, the advanced earliest deadline first based on transactions aggregation links and data semantic links (AEDF-TAL-DSL) protocol emerged as a scheduling solution in a centralised real-time context. This paper focuses on exploring the adaptability of the AEDF-TAL-DSL protocol to the DFCS architecture, offering a significant opportunity to optimise DRTDBMS performance. The novelty of our approach is to enable advanced transaction scheduling while considering transaction aggregation links and data semantic links aiming to

Copyright © 2025 Inderscience Enterprises Ltd.

maximise the number of satisfied transactions. We illustrate the contributions of our approach through simulation results, demonstrating its effectiveness to enhance the QoS in DRTDBMS.

Keywords: distributed real-time database management systems; DRTDBMS; quality of service; QoS; scheduling; user satisfaction; AEDF-TAL-DSL; transactions; simulation.

Reference to this paper should be made as follows: Achour, F., Bouazizi, E. and Jaziri, W. (2025) 'A semantics-based validation approach for enhancing QoS in distributed real-time DBMS', *Int. J. Intelligent Information and Database Systems*, Vol. 17, No. 1, pp.124–142.

Biographical notes: Fehima Achour is a PhD student in Computer Science at the University of Sfax, Tunisia. She received her Master's in Computer Science from the University of Monastir, Tunisia, in 2013. Her research interests include quality of service, real-time systems, database management systems and feedback control scheduling.

Emna Bouazizi is an Assistant Professor in Computer Science at the Jeddah University, KSA. She received her Master's and PhD in Computer Science from the University of Le Havre, France, respectively in 2003 and 2009. Her research interests include real-time applications, intelligent transportation, database management, data warehouses and ontologies.

Wassim Jaziri received his PhD in Computer Science from INSA-Rouen, France, in 2004. He worked as a Post-Doctorate Researcher at the CIRAD-Montpellier, France. In 2007, he joined as an Assistant Professor the Higher Institute of Computer Science and Multimedia, Sfax-Tunisia. In 2010, he received accreditation to supervise research. Currently, he is a Professor at King Faisal University, KSA, managing PhD and Master students. His research interests include geographic information systems, spatio-temporal databases, spatial decision aid, and more. He has participated in international projects and published books and papers in international journals and conferences. He reviewed for international journals and organised conferences.

1 Introduction

Nowadays, distributed systems make the modern economy work, which is certainly useful. Notably, distributed database management systems (DBMS) are widely used in large organisations, both for internal business purposes and for internet services, including e-commerce websites and online multiplayer games (where millions of gamers from all over the world simultaneously connect to the servers hosted by the provider via an internet connection). Such uses have generated huge amounts of data that must be stored somewhere, quickly retrieved, and manipulated in many ways. Distributed databases offer unique features that cannot be found in the centralised ones. Not only they help solving today's data storage problems, but also they can change the way applications work on the internet and elsewhere.

Applications such as Facebook and Twitter are distributed and use distributed DBMS. In particular, educational institutions, banking sector, stock market, military organisations, etc. need to be managed by distributed DBMS whose manipulated transactions have explicit time constraints to be respected. In this case, we need to use distributed real-time DBMS (DRTDBMS). Within such systems, the main challenge is to maintain the consistency of the database while ensuring that transactions meet their deadlines. Such a goal is difficult to achieve in a centralised context and it is certainly even more in a distributed context. Indeed, new issues are raised with distributed DBMS such as communication delays. The algorithms applied to these systems, and in particular the scheduling algorithms, are intended to carry out the transaction before its deadline expires, whereas the throughput is not a major factor.

Currently, research efforts on scheduling continue to emerge to address the challenges of real-time environments. In Jiang et al. (2020), the authors proposed a real-time scheduling approach for parallel tasks with tight deadlines, particularly in the context of systems using multicore processors. However, Zhang et al. (2022) focused their study on distributed real-time scheduling of multiple services to meet dynamic and personalised demands in the field of cloud manufacturing. Additionally, in Zou et al. (2023), the authors designed a graphics processing unit incorporating a real-time scheduler specifically for parallel tasks with strict deadlines. From a different perspective, Salamun et al. (2023) explore the use of genetic programming to automatically synthesise scheduling heuristics in loosely constrained real-time systems. In the context of embedded real-time systems, Mahmood and Ahmad (2023) propose a semi-partitioned fixed-priority scheduling algorithm for uniform multicore processors.

The earliest deadline first (EDF) (Buttazzo, 2004) is one of the main real-time scheduling protocols and it remains a prominent choice in real-time systems due to its simplicity and effectiveness in meeting task deadlines. Its principle consists of assigning the highest priority to the transaction with the closest deadline. Recent studies continue to utilise EDF in various applications. For instance, Lin et al. (2024) employed EDF for improving the performance of real-time systems specifically by addressing the challenges posed by shared cache resources in modern multicore processors. Likewise, in Wu et al. (2024), the authors focused their study on investigating EDF for asymmetric multiprocessor platforms.

With the previously mentioned scheduling works, various criteria have been considered to determine the next task to execute. Most of these criteria are primarily focused on satisfying temporal constraints in a real-time context. However, there are certain important parameters that remain unexplored even though they could bring a new dimension to scheduling. In particular, it has been proposed the advanced earliest deadline first based on transactions aggregation links and data semantic links (AEDF-TAL-DSL), in Achour et al. (2021a), as a scheduling protocol in a centralised context of real-time DBMS. It is an EDF-based protocol which explores new parameters including the user choice as well as the semantic and aggregation links existing between the transactions. Given the promising results that proved this protocol in allowing the system to better adapt to the specific requirements of the user and satisfying the most of its requests, the effectiveness of such a scheduler in a distributed environment is worth to study. Our main goal in this paper is to focus on the applicability of the AEDF-TAL-DSL scheduler in DRTDBMS architecture based on a feedback control scheduling and which is considered as the basic model of quality of service (QoS) management in these systems. This architecture was proposed in Wei et al. (2003) and is known as the distributed feedback control scheduling (DFCS) architecture.

The DFCS architecture serves to control the local behaviour of each site in the system, while ensuring a certain load balance between the various sites. As a result, it meets the QoS requirements of DRTDBMS, notably by adopting a mechanism to combat system imbalance situations. Although this solution has demonstrated a better guarantee of overall QoS (Salem et al., 2019; Abid et al., 2021), it has never been implemented with a scheduling protocol other than the EDF protocol (Buttazzo, 2004), which is based only on transaction deadlines. However, other scheduling approaches such as the AEDF-TAL-DSL protocol contribute to improving QoS in a real-time context and are based on other parameters including deadlines. This is the context of our new contribution.

The rest of this paper is structured as follows. In Section 2, we present the related work, which includes the QoS management model as well as the semantics-based scheduling strategy on which we base our work. Section 3 is dedicated to the description of our proposed QoS approach, while Section 4 presents its simulation results, which are discussed in the same section to evaluate its performance. We conclude this paper in Section 5 by discussing our contribution and presenting our future work.

2 Related work

In this section, we present our distributed real-time database model by introducing data and transaction models, and specifying the fundamental performance metrics we consider. Likewise, we introduce the QoS management approaches on which is based our research work.

2.1 Data and transaction models

In our data model, we consider a replicated main memory database that incorporates both real-time and non-real-time data (Abbott and Garcia-Molina, 1992). Real-time data consist of sensor data from the physical world, which are regularly updated to accurately reflect the real-world state. Each real-time data object comes with a validity interval, beyond which it becomes obsolete, and a timestamp indicating the most recent observation of the real-world state. Non-real-time data refer to those typically found in traditional databases, which do not dynamically change over time. The data replication technique enhances data availability across various sites. Consequently, it plays a crucial role in helping transactions meet their time requirements (Wei et al., 2003). Data in our model are semi-totally replicated (Ben Salem et al., 2014).

For distributed real-time transactions, we focus on firm deadline transactions (Abbott and Garcia-Molina, 1992), wherein a transaction, upon missing its deadline, is aborted and then is useless for the system. Transactions are categorised into update and user transactions based on the type of data items they access. Update transactions are executed periodically to refresh real-time data objects, simultaneously updating replicas of real-time data. Each update transaction is considered to comprise a single sub-transaction, invariably involving a write operation. User transactions, on the other hand, are aperiodic. We assume that each user transaction consists of a set of sub-transactions. Consequently, each sub-transaction is constructed from a series of read operations on both real-time and non-real-time data objects, along with write operations restricted to non-real-time data objects.

In distributed real-time databases, transactions can be executed either locally or remotely, depending on the location of the required data items (Ben Salem et al., 2014). It is important to highlight that in DRTDBMS models employing load balancing techniques, a transaction might initially have all the necessary data available at its local site. However, it is possible to be distributed to address an overloaded situation at that site and ensure optimal system performance.

2.2 QoS management in DRTDBMS

The QoS assessment is becoming progressively more significant for the evaluation of DRTDBMS performance, where system efficiency relies on the distribution of workloads. In Wei et al. (2003), the authors introduced an architecture featuring feedback-based global load balancers and local feedback controllers. This architecture is known as the DFCS architecture. It serves as the foundation for our research work and is designed to achieve load balancing among system nodes and efficient management of fluctuations in transaction workloads.

The DFCS architecture involves an admission controller which plays a crucial role in governing the system workload to prevent overload. This is accomplished by considering the estimated CPU utilisation and the predetermined target utilisation set point for the system. Likewise, it includes a scheduler which is indispensable to organise transactions based on the EDF protocol (Buttazzo, 2004). This architecture contains also a transaction manager which is responsible for executing transactions, and which comprises a concurrency controller, a freshness manager, a data manager, and a replica manager. The concurrency controller resolves data access conflicts that arise between transactions through the use of the two-phase locking high priority (2PLHP) protocol (Bernstein et al., 1987). The freshness manager verifies the freshness of data and prevents a user transaction from proceeding if the accessed data item is outdated. The data manager is responsible for updating real-time data replicas, while the replica manager manages data replicas through a replication control protocol.

During each sampling period, the local monitor of the DFCS architecture samples system performance data by referencing statistics related to transactions' execution obtained from the transaction manager. The measured values form part of the feedback control loop and are subsequently delivered to a local controller. This local controller comprises a local utilisation controller and a local miss ratio controller, each generating the local miss ratio and local utilisation control signals, respectively. These signals are based on the received values and the system reference parameters. The controller then establishes the system's target utilisation to be taken into account in the next sampling period.

In particular, the DFCS architecture is characterised by a global load balancer that guarantees the balancing of system loads through the exchange of system performance data with other nodes. This is achieved by transferring transactions from highly overloaded nodes to less overloaded ones. The load transferring factor (*LTF*) of each node controls the amount of workload to be transferred (Wei et al., 2003).

As we mentioned, the DFCS architecture relies on the EDF protocol for scheduling transactions. Nevertheless, there are alternative approaches that support QoS in a real-time context and which can be beneficial for our work. In the next section, our focus will be on presenting a scheduling protocol, the integration of which into the described architecture will be of great interest for QoS management in a distributed context.

2.3 The AEDF-TAL-DSL protocol

The AEDF-TAL-DSL is a semantics-based protocol for scheduling real-time transactions introduced in Achour et al. (2021a). The AEDF-TAL-DSL protocol is an enhancement of the EDF protocol, incorporating additional scheduling parameters into consideration. Initially, it considers the users choice within the system to define the most critical transaction according to their preferences. Each transaction is then assigned an indicator of its criticality level (Achour et al., 2016). Hence, this protocol enhances system user satisfaction by prioritising these transactions. Secondly, the AEDF-TAL-DSL protocol considers potential aggregation links between transactions as an additional scheduling parameter. These links are determined based on the type of operations within each transaction and the accessed data associated with each operation. Thirdly, this protocol incorporates an extra parameter into the transaction scheduling process by revealing the semantics of users' queries. This parameter involves the semantic relations that exist between these queries. These relations are derived from the chosen semantic resource which is the WordNet linguistic ontology (Miller et al., 1990; Bayoudhi et al., 2017, 2018; Jaziri et al., 2019).

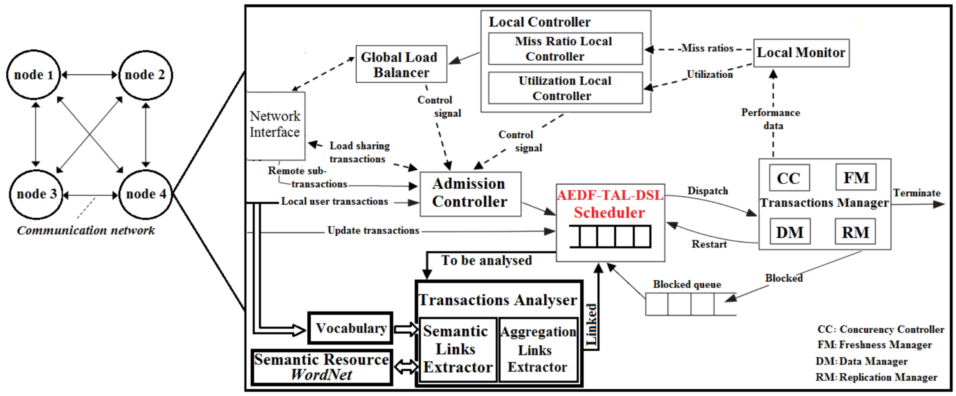
The AEDF-TAL-DSL protocol facilitates the identification of numerous links between the transactions to be scheduled. Enhancing the number of linked transactions contributes to improving the number of transactions that successfully complete their executions before their deadlines, thereby satisfying user requirements and ensuring better QoS. These results have been demonstrated in Achour et al. (2021a), where this protocol has been implemented in a feedback-control-based architecture for enhancing QoS in RTDBMSs (Achour et al., 2021a). Similarly, it has been validated within a QoS-based approach for ontology-based real-time DBMSs (Achour et al., 2021b). The promising results motivate the implementation of this protocol within the DFCS architecture to enhance QoS in DRTDBMS. In the following section, we provide more details about our proposed approach.

3 Distributed feedback control scheduling with AEDF-TAL-DSL: the DFCS-AEDF-TAL-DSL architecture

The novelty of our contribution consists in enhancing the performance of the AEDF-TAL-DSL scheduler (Achour et al., 2021a), by taking advantage of the benefits of distributed DBMS in terms of reliability, data availability, speed of response to user requests, etc. The encouraging results recorded by this scheduler in a centralised context, according to the respect of the temporal constraints of the transactions, encourage us to validate it within the framework of a distributed real-time DBMS. And thus, at the same time, we will contribute to the improvement of the QoS in these systems. To do this, we opt for the DFCS architecture, considered as the basic QoS management approach in DRTDBMS in order to apply the AEDF-TAL-DSL protocol. The DFCS architecture proves to meet the requirements of the DRTDBMS, in terms of QoS, in particular by adopting a mechanism making it possible to fight against the situations of imbalance of these systems (Wei et al., 2003).

Our approach consists in extending the DFCS architecture by implementing the AEDF-TAL-DSL scheduler for real-time distributed transactions. Being based on new parameters, the implementation of this scheduler in such an architecture requires the addition of new components relating to these parameters on each site of the DRTDBMS. Our new extension is a distributed semantic-based feedback control scheduling architecture. Figure 1 displays the new architecture that we call distributed feedback control scheduling with AEDF-TAL-DSL and we denote by the DFCS-AEDF-TAL-DSL architecture. The main components of this architecture will be described in the following.

Figure 1 A DRTDBMS architecture using the AEDF-TAL-DSL scheduler (see online version for colours)



We begin by describing the components that mark the DFCS-AEDF-TAL-DSL architecture. On each site of the system, the admitted transactions will be executed in the order and at the time defined by the AEDF-TAL-DSL scheduler. Indeed, this scheduling protocol is characterised by taking into account aggregation links between transactions, including semantic links that exist between user requests submitted to the system. Moreover, it promotes the satisfaction of system users by favouring the execution of transactions that are critical to them. As a result, each site in the system is provided with a transaction analyser responsible for analysing the user transactions to be scheduled, with the aim of extracting the different types of links existing between them. The analysed transactions involve local transactions as well as distributed transactions. The semantic links are determined using the semantic link extractor (see Figure 1), whose role is to search the user requests. The input vocabulary, as shown in Figure 1, comprises terms collected from user queries and serves as the raw material for the extraction process. This vocabulary includes a diverse range of terms and phrases reflecting the varied needs and intents of the system users. These terms may cover actions, queries, commands, and preferences expressed in different linguistic styles and levels of specificity. Such a task requires the use of a semantic resource that we choose to be the linguistic ontology *WordNet* (Miller et al., 1990). WordNet groups words into sets of synonyms (synsets) and captures semantic relationships between them. It serves as a valuable resource for natural language processing tasks, aiding in word sense disambiguation, information retrieval, and other language-related applications. Developed at Princeton University, WordNet's structure provides a rich semantic network of words, making it widely used in the field of computational linguistics. *WordNet* analyses the semantic relationships embedded within the input vocabulary.

As shown in Figure 1, the transaction analyser also includes an aggregation link extractor ensuring the determination of aggregation links appearing between transactions. This will be on the basis of the operations that compose each transaction as well as the data items it accesses. As soon as the different types of links are ready, the AEDF-TAL-DSL scheduler proceeds with them in order to set the transaction priorities. It should be noted that in the case of distributed transactions, a sub-transaction inherits the criticality level of its parent transaction, to be considered when assigning priorities by the AEDF-TAL-DSL protocol. System update transactions will have a higher priority with our scheduler since they update real-time data needed for user transactions.

We focus now on describing the validation process to which the transactions scheduled by the AEDF-TAL-DSL protocol are subject in our architecture. We are particularly interested in the cases of the validation of transactions which admit other linked transaction(s) and/or sub-transaction(s) that are not yet validated. We denote by T_i the transaction in process of validating its execution. In this context, T_i can be either a local transaction or a sub-transaction of a distributed transaction. At this account, local transactions similar to T_i commit immediately. However, sub-transactions similar to T_i commit while their parent transactions do not until all their sub-transactions commit. We note that the principle of validation of distributed transactions follows the PROMPT protocol (Haritsa et al., 2000).

The execution of each transaction will take place at the transaction manager according to the order and at the time defined by the scheduler AEDF-TAL-DSL. To do this, this manager has other complementary modules, as illustrated in Figure 1. These different modules keep the same operation in our architecture as in the basic DFCS architecture (cf. Subsection 2.2).

In the following section, we are interested in evaluating the performance of the DFCS-AEDF-TAL-DSL architecture in order to highlight its contribution to the improvement of QoS in DRTDBMS.

4 Evaluation of the DFCS-AEDF-TAL-DSL architecture performance

4.1 Description of the DFCS-AEDF-TAL-DSL simulator

In order to evaluate the performance of our approach and validate the resulting benefit, in terms of maximising the number of transactions that meet their deadlines as well as the number of satisfied users, we rely on a RTDBMS simulator. This simulator, called DFCS-Simulator, was developed as part of the thesis work of Ben Salem (2017), where the aim is to simulate the operation of the various components of the DFCS architecture. In order to validate ours, we make modifications to the DFCS-Simulator by adding the modules are necessary for the proper functioning of our scheduler AEDF-TAL-DSL in the new architecture. In its new reform, we note this simulator by DFCS-AEDF-TAL-DSL-Simulator whose architecture is illustrated by Figure 2. We give below an overview of the functions of the different modules managed by each site of our system and which are as follows:

- *Database:* The database is made up of a set of both real-time and conventional data, the generation of which is done randomly. Access to the various data items takes place at the transaction execution module.
- *Data manager:* This module takes care of updating the different copies of replicated data, in order to ensure as much as possible their freshness, and thus increase the availability of data for transactions.
- *Update transactions generator:* For each real-time data item in the database, it is responsible for generating an update transaction invoked periodically, in order to maintain the freshness of the corresponding data item. These transactions are generated according to a universal method assigning a random period, so that the deadline of the transaction can be greater than, less than or equal to the validity period of the data to be refreshed.
- *Precision controller:* This module is responsible for discarding update transactions when the data to be refreshed is sufficiently representative of the real world, taking into account the parameters data error (*DE*) and maximum DE (*MDE*). A transaction is thus discarded once $DE \leq MDE$. The adjustment of the value of *MDE* is done according to the current state of the system's workload.
- *User transactions generator:* It generates user transactions randomly according to the Poisson distribution, in order to materialise the unpredictability in the arrival of this type of transaction on each site of the system.
- *Admission controller:* Any generated user transaction, with an arrival date greater than or equal to the current time, is subjected to this module which will decide its acceptance or rejection. The entry of transactions is adjusted based on the workload of the respective site, while ensuring that the accepted ones can meet their deadlines.
- *Global load balancer:* It is responsible for ensuring a certain load balance between different sites of the system, so that the overload of one site does not burden the operation of others. This is achieved by transferring transactions from the overloaded site to those that are not. The amount of transactions to be transferred during the next sampling period is determined based on the *LTF* of each site. The transactions thus transferred are submitted to the admission controllers of the target sites.
- *Local controller:* Being composed of a local deadline controller and a local processor utilisation controller, this module utilises system performance data sent by the monitor to provide the global load balancer and the admission controller with sufficient information about the system's state. These data include the percentage of missed transaction deadlines deduced from the completion of certain transactions during the current sampling period, as well as the processor utilisation rate calculated based on transaction execution durations during this same period. It is according to these values and the reference parameters entered by the users that this module establishes the processor utilisation value to be considered at the next sampling period. Based on this value, the global load balancer of each site adjusts its *LTF* in order to decide whether it is necessary to transfer transactions

or not, and if so how much to transfer. Similarly, the admission controller will decide whether to accept more or less user transactions in the system.

- *Monitor:* It is a key module in the simulator. It is used to scrutinise the state of the system (stable, underused or overloaded) by referring to the execution results of transactions. Subsequently, it delivers the collected information to the local controller. This is how the feedback mechanism is established to take advantage of it when filtering incoming user transactions to the system.
- *User choice list:* It is an essential list for the scheduler in our simulator. In this list, the user of the system designates the transactions which are the most important to him by assigning a well-determined level of criticality to each one. The most important transaction for a user will have the highest criticality level value.
- *Aggregation links analyser:* It is a module that distinguishes the DFCS-AEDF-TAL-DSL-Simulator (cf. Figure 2) and which is complementary to the scheduling module. Its role is to identify the different type of links existing between the transactions to be scheduled, namely aggregation links and semantic links. To do this, it retrieves the list of user transactions to schedule and examines the operations that make up each one as well as the data accessed, in order to extract all the aggregation links if they exist. Thereafter, it proceeds by the lexical database *WordNet* which offers a set of methods to easily navigate between the semantically linked words of the generated database in order to extract any semantic links.
- *AEDF-TAL-DSL scheduler:* It is the component that characterises our simulator whose function is to decide the order of execution of transactions in the system. These are indeed the user transactions received from the admission controller in addition to the update transactions admitted by the precision controller. With our scheduler, the determination of the priorities of user transactions is based, initially, on the criticality levels of the transactions which are recovered by accessing the user choice list. Secondly, it exploits the different type of links between the user transactions that it recovers from the aggregation link analyser. Being composed each one of a single write operation of a real-time data item, the update transactions are always assigned the highest priorities by our scheduler compared to the user transactions. This comes down to the fact that we need to keep real-time data as fresh as possible so as not to block the execution of user transactions that access it in a read mode.
- *Transaction execution:* In this module, the transactions received in the order predefined by the scheduler begin their executions while accessing their data. To this end, we see this module interact with the other modules of the simulator as shown in Figure 2, in order to ensure the execution of transactions under the required conditions.
- *Concurrency controller:* During their execution, the transactions may confront data access conflicts. It is in this module that these conflicts are resolved by implementing, in our simulator, the 2PL-HP protocol (Bernstein et al., 1987).

- *Freshness manager:* The underlying objective of this module is to check the validity of real-time data accessed in read mode by user transactions. Once the required data is obsolete, the corresponding transaction will be queued in the blocked transaction queue, until its data is refreshed by an update transaction.
- *Replication manager:* The role of this module is to manage transaction access to the different copies of data, while following the appropriate replication control protocol. In our simulator, this involves implementing the real-time replication control protocol (RT-RCP) (Said et al., 2007). It ensures that replicas are updated without affecting compliance with transaction deadlines. It does this by allowing some inconsistencies between copies while preventing access to obsolete data until it is updated. Thus, each site in the system has a set of lists of available copies, each of which is called list of available copies (LAC). Each LAC holds all the sites with fresh copies of the corresponding data. It is with the help of these lists that sites can adequately decide on the choice of sites participating in the execution of their transactions, while choosing sites where the target data is fresh. Regarding the replication policy used for our simulated RDBMS, it is the semi-total replication policy proposed in Ben Salem et al. (2014), and which demonstrated better results compared to the number of sites that we will choose.
- *Transactions validation:* This module intervenes each time a transaction completes its execution. Its operation follows the PROMPT validation protocol (Haritsa et al., 2000), implemented in our simulator. In order to coordinate between the different sites invoked during the execution of the transaction in question, this module uses the message manager.
- *Message manager:* This module serves as a link between the various sites of the system and is indispensable in a distributed environment. In fact, collaboration between DRTDBMS sites is ensured via the exchange of messages. Especially, for the validation of distributed transactions, we need such a component to make a decision (validation or abandonment) regarding the ongoing transaction, from its coordinating site and the other sites participating in its execution. We also need it to update copies of data on the different sites in the system.
- *Input interface* and *output interface:* these interfaces serve the main function of making the use of our simulator more convenient. Indeed, the input interface is an initialisation tool through which the users enter, at their choices, the values of the parameters necessary for the simulations they will perform. On the other hand, the output interface assists in visualising the results of the simulations for the users to benefit from during their possible interpretations.

4.2 Simulation settings

Our goal in proposing the DFCS-AEDF-TAL-DSL architecture is to validate the use of the AEDF-TAL-DSL scheduling protocol in the context of DRTDBMS while guaranteeing better QoS in these systems. Now, it remains for us to evaluate the performance of our new architecture by conducting a series of experiments within the DFCS-AEDF-TAL-DSL-Simulator that we have just described above. To do this, we specify all the necessary parameters, the chosen values of which are given in Table 1.

Figure 2 The global architecture of the DFCS-AEDF-TAL-DSL-simulator (see online version for colours)

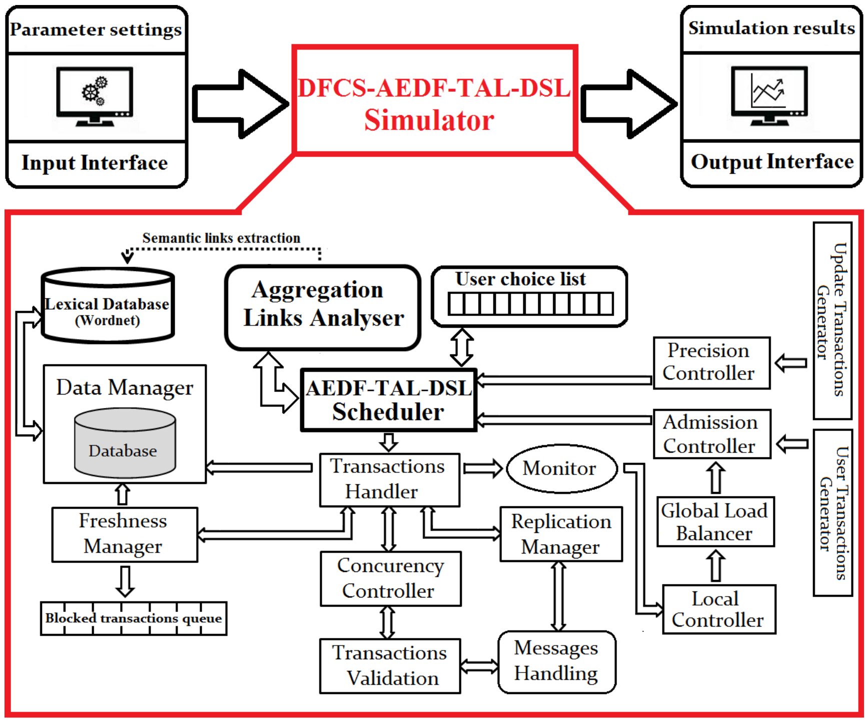


Table 1 further summarises the parameters relating to the generation of real-time and classic data of the database, as well as the generation and execution of *user* and update transactions. The values of the thresholds relating to the semi-total replication policy that we have chosen are presented in Table 2.

In our simulations, we opt for a DRTDBMS composed of eight sites, on each we generate 200 real-time data and 10,000 classic data. Indeed, a classic data is defined by its value as well as by its label which represents its unique identifier in the database. On the other hand, a real-time data is characterised, in addition to its value and its label, by its validity period and its timestamp. The value of each real-time data item is modified following each access of its update transaction. The validity period is the phase during which the data can be used, and the timestamp is the date of its last update. In our experiments, the validity period of each real-time data varies from 500 to 1,000 milliseconds (ms) and the MDE value varies by one step between 2 and 5.

Each user transaction is made up of a series of read operations of real-time and classic data and writing only operations of classic data. The rate of remote data to be accessed by the operations of each user transaction is set at 20%. On the other hand, each update transaction is only a single real-time data write operation executed locally. We set the value of the transaction slack factor to 10.

At each queue, the transactions are scheduled according to the AEDF-TAL-DSL protocol with the 2PL-HP protocol for concurrency control. For the transactions validation, we use the distributed PROMPT protocol.

Table 1 The main settings of the DFCS-AEDF-TAL-DSL-simulator

Parameter	*Value*
Simulation duration (ms)	3,000
Number of system sites	[1, 8]
Number of real-time data	200/site
Number of classical data	10,000/site
Real-time data validity duration (ms)	[500, 1,000]
Maximum data error (MDE) of real-time data	[2, 5]
Number of sub-transactions of a user transaction	[1, 2]
Number of read operations of a sub-transaction	[0, 2]
Number of write operations of a sub-transaction	[1, 2]
Execution time of a read operation (ms)	1
Execution time of a write operation (ms)	2
Constant transaction slack factor	10
Remote data ratio (%)	20
Concurrency control algorithm	2PL-HP
Scheduling algorithm	AEDF-TAL-DSL
Distributed validation algorithm	PROMPT

Table 2 Basic settings specific to data replication

Parameter	*Value*
Maximum threshold for sites supporting replication	0.5
Minimum threshold for sites supporting replication	0.2
Maximum threshold of data to replicate	0.2
Minimum threshold of data to replicate	0.1

4.3 Performance metrics

The key performance metrics we consider in our work are:

- The success ratio (*SR*) of transactions serves as a QoS parameter. It quantifies the percentage of transactions that successfully complete their executions before their deadlines. The formula for calculating this ratio is as follows:

$$SR = 100 \times \frac{\#Timely}{\#Late + \#Timely} (\%), \tag{1}$$

  where #Late and #Timely represent, respectively, the number of transactions that missed their deadlines and the number of successful transactions.

- The user satisfaction ratio (*USR*) which represents the *SR* of transactions that are qualified as critical for users. It is defined as a QoS parameter measuring the percentage of transactions that meet their deadlines and having the following formula:

$$USR = 100 \times \frac{\#TimelyC}{\#LateC + \#TimelyC} (\%) \tag{2}$$

where #LateC and #TimelyC respectively denote the number of critical transactions that have missed their deadlines and the number of successful critical transactions.

In the following we present and interpret the simulation results we obtained.

4.4 Results and discussions

Experiment results demonstrating the performance of the DFCS-AEDF-TAL-DSL architecture are based on the performance metrics we defined in Section 4.3.

In our experiments, at first we simulate the behaviour of our system by setting the number of sites (denoted Nb_sites) to 1. Subsequently, we increase the size of the system to eight sites to finally be able to implement the contribution of our approach by comparing the results we obtained. For each experiment we set the number of transactions (denoted Nb_transactions) to generate in the system.

4.4.1 Results of experiment 1

The different values of the *SR* obtained for each number of generated transactions are recorded in Table 3. In order to facilitate the interpretation of the result obtained, we give a graphical representation of this table in Figure 3.

Table 3 SR values using the DFCS-AEDF-TAL-DSL architecture

SR (%) / *Nb_transactions*	*Nb_sites = 1*	*Nb_sites = 8*
600	52	65
1,000	33	61
1,400	26	46
1,800	23	41
2,200	18	37

Each curve shown in Figure 3 exposes the variation of the SR according to the number of generated user transactions in the system. According to these curves, we see that the performance of the AEDF-TAL-DSL protocol used in a distributed context (Nb_sites = 8) is considerably better than that obtained in a centralised context (Nb_sites = 1). This improvement is further clarified by Figure 4. We also emphasise that even when the number of transactions in the system increases, our DFCS-AEDF-TAL-DSL architecture has a considerable contribution. The value of this contribution is of the order of 20% for approximately 2,000 user transactions submitted to the system. This reveals the advantage that arises from our architecture in supporting a load balancing mechanism between the different sites of the system thanks to the feedback mechanism that it adopts.

Furthermore, our approach allows to lighten the system by minimising the number of concurrent transactions through the extraction of related ones. Regarding the decline in the rates of both curves, it is attributed to the system's reduced capacity to handle all the present transactions. Even under these conditions, the DRTDBMS of eight sites that we manage with our approach has successfully satisfied approximately 40% of the 2,000 user transactions.

Figure 3 Comparison of SR values with DFCS-AEDF-TAL-DSL

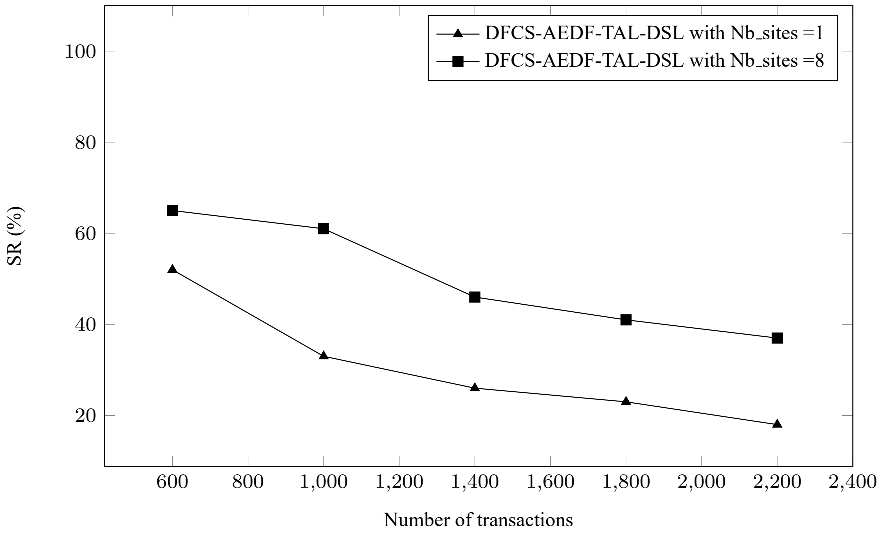
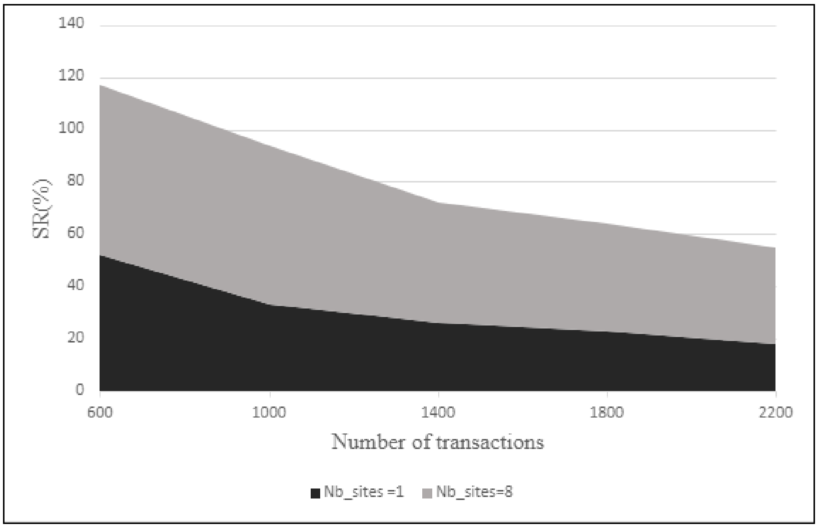


Figure 4 Stacked area chart for visualising the SR variation



4.4.2 Results of experiment 2

In this series of experiments, we save the different USR values acquired for each number of user transactions generated in the system. Table 4 summarises the values thus obtained and which are represented graphically in Figure 5 in order to simplify their interpretation.

Table 4 USR values using the DFCS-AEDF-TAL-DSL architecture

USR (%) / *Nb_transactions*	*Nb_sites = 1*	*Nb_sites = 8*
250	98	74
350	88	78
450	75	80
650	67	82
750	66	85

The curves in Figure 5 show the variation of the USR as a function of the number of user transactions generated in the system. Referring to these curves, we see that by using our AEDF-TAL-DSL scheduler in a DRTDBMS of eight sites, the number of satisfied users is greater than that obtained by using it as part of a centralised system. This result is validated in particular when the number of *user* transactions qualified as critical begins to be greater. Indeed, when the number of critical transactions is not as large (between 250 and 350 transactions) and the requested data is on the same site as their transactions, the result of executing them locally was better (see Figure 5). The more the number of transactions increases, the more the contribution of our scheduler in the DFCS-AEDF-TAL-DSL architecture is revealed. The increasing shape of the curve is due to the fact that each time the number of critical transactions increases, more links appear between them, thus giving them more chances to be validated. The gap is also clear when we refer to Figure 6. Really, user satisfaction using our approach is mainly due to taking into account user choice regarding the assignment of priorities to transactions.

5 Conclusions and future work

Our objective in this paper was to propose the DFCS-AEDF-TAL-DSL architecture for QoS management in DRTDBMS. Our approach is to improve the performance of our AEDF-TAL-DSL scheduling protocol by making it functional in a distributed context. In this way, we improve the scalability of systems using our protocol while guaranteeing a certain QoS. The improvement in QoS brought about by our new architecture has been approved on the basis of the experimental results we have obtained. These results showed a significant increase in the number of transactions that succeeded in validating their executions within the required time constraints. The results also showed a considerable improvement in the user satisfaction with our system.

This work opens up interesting new areas of research. As a first step, we plan to extend the DFCS-AEDF-TAL-DSL architecture to manipulate real-time ontological data (Ben Salem et al., 2019; Achour et al., 2021b; Abid et al., 2021). In this way, we can help distributed transactions accessing ontological data to meet their time constraints. As a second step, we aim to apply the architecture we proposed in this paper to plan and execute critical tasks in embedded systems for autonomous vehicles (Hu et al., 2023). This helps ensure real-time responses for operations such as navigation, perception and decision-making. Finally, it would be interesting to take advantage of our semantics-based scheduling approach by integrating it into the field of connected health (Bayoudhi et al., 2021). Indeed, real-time scheduling protocols play a crucial role in the evolution of connected health. By scheduling the real-time operations of medical devices, monitoring sensors and tracking applications, they guarantee rapid responses to changing medical conditions. This perspective also extends to the integration of the internet of things (IoT) and cloud computing technologies (Elshahed et al., 2022; Sharma et al., 2023). This can help creating a robust ecosystem that enhances healthcare efficiency, real-time data collection and medical decision making.

Figure 5 Comparison of USR values with DFCS-AEDF-TAL-DSL

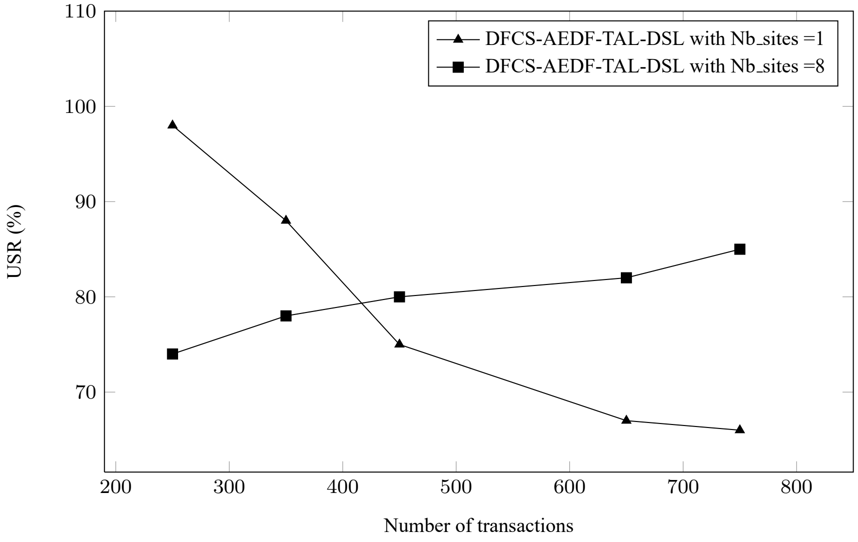
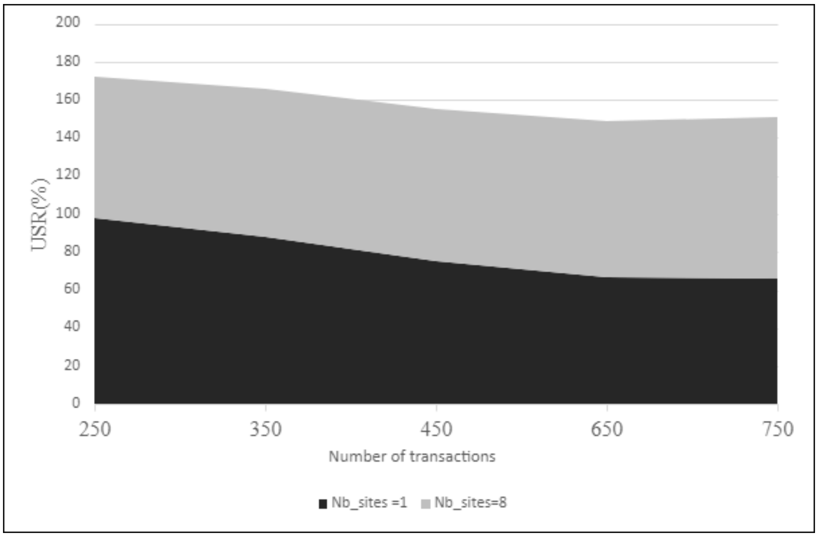


Figure 6 Stacked area chart for visualising the variation of the USR



References

Abbott, R.K. and Garcia-Molina, H. (1992) 'Scheduling real-time transactions: a performance evaluation', *ACM Trans. Database Syst.*, Vol. 17, No. 3, pp.513–560.

Abid, W.B., Mhiri, M.B.A., Bouazizi, E. and Gargouri, F. (2021) 'Ontological data replication in a distributed real-time database system', *New Trends in Intelligent Software Methodologies, Tools and Techniques – Proceedings of the 20th International Conference on New Trends in Intelligent Software Methodologies, Tools and Techniques, SoMeT 202*, Cancun, Mexico, 21–23 September, Vol. 337 of *Frontiers in Artificial Intelligence and Applications*, pp.567–580, IOS Press.

Achour, F., Bouazizi, E. and Jaziri, W. (2016) 'Scheduling approach for enhancing quality of service in real-time DBMS', *Databases and Information Systems – 12th International Baltic Conference, DB&IS 2016, Proceedings*, 4–6 July, Riga, Latvia, Vol. 615 of *Communications in Computer and Information Science*, pp.126–135, Springer.

Achour, F., Bouazizi, E. and Jaziri, W. (2021a) 'Improving the quality of service of real-time database systems through a semantics-based scheduling strategy', *Int. J. Intelligent Information and Database Systems*, Vol. 14, No. 1, pp.96–114.

Achour, F., Jaziri, W. and Bouazizi, E. (2021b) 'Semantics-based scheduling approach of ontology-based real-time DBMS', *New Trends in Intelligent Software Methodologies, Tools and Techniques – Proceedings of the 20th International Conference on New Trends in Intelligent Software Methodologies, Tools and Techniques, SoMeT 2021*, 21–23 September, Cancun, Mexico, Vol. 337 of *Frontiers in Artificial Intelligence and Applications*, pp.553–566, IOS Press.

Bayoudhi, L., Sassi, N. and Jaziri, W. (2017) 'Overview and reflexion on OWL 2 DL ontology consistency rules', *Proceedings of the Second International Conference on Internet of Things, Data and Cloud Computing*, Association for Computing Machinery, New York, NY, USA, Vol. 133.

Bayoudhi, L., Sassi, N. and Jaziri, W. (2018) 'How to repair inconsistency in OWL 2 DL ontology versions?', *Data & Knowledge Engineering*, Vol. 116, pp.138–158.

Bayoudhi, L., Sassi, N. and Jaziri, W. (2021) 'An overview of biomedical ontologies for pandemics and infectious diseases representation', *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 25th International Conference KES2021, Procedia Computer Science*, Vol. 192, pp.4249–4258.

Ben Salem, M. (2017) *Gestion de la qualité de service dans les SGBD temps réel distribués*, PhD thesis, MIRACL, Université de Sfax.

Ben Salem, M., Bouazizi, E., Bouaziz, R. and Duvallet, C. (2014) 'Quality of service management in distributed feedback control scheduling architecture using different replication policies', *International Conference on Foundations of Computer Science & Technology (CST-2014)*, Zurich, Switzerland, pp.75–87.

Ben Salem, M., Bouazizi, E., Duvallet, C. and Rafik, B. (2019) '(m,k)-firm constraints and derived data management for the QoS enhancement in distributed real-time DBMS', *Int. Arab J. Inf. Technol.*, Vol. 16, No. 3, pp.424–434.

Bernstein, P.A., Hadzilacos, V. and Goodman, N. (1987) *Concurrency Control and Recovery in Database Systems*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Buttazzo, G.C. (2004) *Hard Real-time Computing Systems: Predictable Scheduling Algorithms And Applications (Real-Time Systems Series)*, Springer-Verlag TELOS, Santa Clara, CA, USA.

Elshahed, E., Abdelmoneem, R., Shaaban, E., Elzahed, H. and Altabbakh, S. (2022) 'Prioritized scheduling technique for healthcare tasks in cloud computing', *The Journal of Supercomputing*, Vol. 79, No. 5, pp.1–22.

Haritsa, J.R., Ramamritham, K. and Gupta, R. (2000) 'The prompt real-time commit protocol', *IEEE Trans. Parallel Distrib. Syst.*, Vol. 11, No. 2, pp.160–181.

Hu, B., Shi, Y., Cao, Z. and Zhou, M. (2023) 'A hybrid scheduling framework for mixed real-time tasks in an automotive system with vehicular network', *IEEE Transactions on Cloud Computing*, Vol. 11, No. 3, pp.2231–2244.

Jaziri, W., Bayoudhi, L. and Sassi, N. (2019) 'A preventive approach for consistent OWL 2 DL ontology versions', *International Journal on Semantic Web and Information Systems (IJSWIS)*, Vol. 15, No. 1, pp.76–101.

Jiang, X., Guan, N., Long, X., Tang, Y. and He, Q. (2020) 'Real-time scheduling of parallel tasks with tight deadlines', *Journal of Systems Architecture*, Vol. 108, p.101742.

Lin, Y., Deng, Q., Han, M., Feng, Z., Wang, S. and Peng, Q. (2024) 'Lag-based schedulability analysis for preemptive global EDF scheduling with dynamic cache allocation', *Journal of Systems Architecture*, Vol. 147, p.103045.

Mahmood, B. and Ahmad, N. (2023) 'An optimal semi-partitioned algorithm for scheduling real-time applications on uniform multicore processors', *Sustainable Computing: Informatics and Systems*, Vol. 38, p.100854.

Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D. and Miller, K. (1990) 'WordNet: an on-line lexical database', *International Journal of Lexicography*, Vol. 3, No. 4, pp.235–244.

Said, A.H., Amanton, L., Sadeg, B. and Ayeb, B. (2007) 'Contrôle de la réplication dans les sgbd temps réel distribués', *Proceedings of the 5th MAnifestation des JEunes Chercheurs en Sciences et Technologies de L'Information et de la Communication (MajecSTIC)*, Caen, France, pp.41–49.

Salamun, K., Pavić, I., Dzapo, H. and Durasević, M. (2023) 'Evolving scheduling heuristics with genetic programming for optimization of quality of service in weakly hard real-time systems', *Applied Soft Computing*, Vol. 137, p.110141.

Salem, M.B., Bouazizi, E., Duvallet, C. and Bouaziz, R. (2019) '(m, k)-firm constraints and derived data management for the QoS enhancement in distributed real-time DBMS', *Int. Arab J. Inf. Technol.*, Vol. 16, No. 3, pp.424–434.

Sharma, P., Rezazadeh, J., Bello, A., Dawoud, A. and Albabawat, A.A. (2023) 'Real time remote cardiac health monitoring using iot wearable sensors – a review', in Daimi, K. and Al Sadoon, A. (Eds.): *Proceedings of the Second International Conference on Innovations in Computing Research (ICR'23)*, Springer Nature Switzerland, Cham, pp.280–291.

Wei, Y., Son, S.H., Stankovic, J.A. and Kang, K.D. (2003) 'Qos management in replicated real time databases', *Proceedings of the 24th IEEE International Real-Time Systems Symposium, RTSS '03*, IEEE Computer Society, Washington, DC, USA, pp.86–97.

Wu, P., Han, C., Yan, T., Chen, L. and Li, Y. (2024) 'A BSF-EDZL scheduling algorithm for heterogeneous multiprocessors', *Electronics Letters*, Vol. 60, No. 2, p.e13093.

Zhang, L., Yang, C., Yan, Y. and Hu, Y. (2022) 'Distributed real-time scheduling in cloud manufacturing by deep reinforcement learning', *IEEE Transactions on Industrial Informatics*, Vol. 18, No. 12, pp.8999–9007.

Zou, A., Li, J., Gill, C.D. and Zhang, X. (2023) 'RTGPU: real-time gpu scheduling of hard deadline parallel tasks with fine-grain utilization', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 34, No. 5, pp.1450–1465.