Hindawi Publishing Corporation
Mobile Information Systems
Volume 2015, Article ID 652462, 14 pages
http://dx.doi.org/10.1155/2015/652462

Research Article

Flexible Framework for Real-Time Embedded Systems Based on Mobile Cloud Computing Paradigm

Higinio Mora Mora, David Gil, José Francisco Colom López, and María Teresa Signes Pont

Specialized Processors Architecture Laboratory, Department of Computer Science Technology and Computation, University of Alicante, 03690 Alicante, Spain

Correspondence should be addressed to Higinio Mora Mora; hmora@ua.es

Received 9 December 2014; Revised 29 May 2015; Accepted 2 June 2015

Academic Editor: Salil Kanhere

The development of applications as well as the services for mobile systems faces a varied range of devices with very heterogeneous capabilities whose response times are difficult to predict. The research described in this work aims to respond to this issue by developing a computational model that formalizes the problem and that defines adjusting computing methods. The described proposal combines imprecise computing strategies with cloud computing paradigms in order to provide flexible implementation frameworks for embedded or mobile devices. As a result, the imprecise computation scheduling method on the workload of the embedded system is the solution to move computing to the cloud according to the priority and response time of the tasks to be executed and hereby be able to meet productivity and quality of desired services. A technique to estimate network delays and to schedule more accurately tasks is illustrated in this paper. An application example in which this technique is experimented in running contexts with heterogeneous work loading for checking the validity of the proposed model is described.

1. Introduction

Many of the advances that are experimenting contemporary societies are based on the development of systems that act as sensors environment (Smart Cities, Ambient Intelligence, eHome, Smart Drive, etc.). These systems usually consist of embedded devices, provide insight, and provide intelligence to the interactions that occur with the environment. One of its common functions is signal processing from sensors that incorporate themselves. This processing occurs with the execution of other tasks of the application that are incorporated.

There exist a variety of examples of embedded systems whose functioning is adapted to this pattern: surveillance cameras with motion detection, RFID packet tracking systems, driver assistance systems, smart thermostats, and so forth. Usually, in most of the above applications, embedded systems are connected to a communications network to coordinate their behaviour with other systems and provide a better customer service. One of the most common types of embedded systems lies in mobile terminals, whose expansion in society has been spectacular in recent years.

This context has promoted the proliferation of business strategies for embedded systems in general and especially on mobile devices that aim to leverage its high penetration in society to reach a wider audience as well as open new markets. In this situation there are initiatives such as applications of mobile payment, tracking and tracing, resource monitoring, and so forth.

However, in this situation of technology adoption and deployment of new applications, the fundamental challenge of providing sufficient benefits for the execution of processes in terminals without penalizing user satisfaction is interposed. The performance required for the execution of processes can overflow the resources of most devices, delaying response times and heavily penalizing the expansion of such technologies. Moreover, the limitations on processing capabilities may also come from further additional aspects of the device capabilities. Environmental conditions, the power consumption requirements, or configuration issues, may also impact on the services that is able to offer. Due to the above, we observe the eventual existence of difficulties in meeting

with the requirements of productivity and response times that some applications require.

Moreover, one of the most innovative paradigms regarding the adoption of Information and Communications Technology (ICT) by society is Cloud Computing. The advantages of this model of ICT management approach the improving efficiency and reducing costs, while providing resources and services accessible to the whole society. Any progress that occurs in this area has a multiplier effect that will affect many companies and users of these technologies. Therefore, the design of computational models that combine the development of embedded and mobile systems with Cloud Computing paradigms may provide new ways of processing that allow avoid the difficulties related to real time execution of applications in these systems.

The main objective of this research is to study how real-time can be performed in Cloud Computing paradigms. More precisely, how embedded and mobile systems can take advantage of the remote computing resources to meet with real time constraints.

In this way, this work proposes a computational model of processing integrated management for embedded or mobile systems in order to answer the following questions: Is it possible to derive some running processes to the cloud and thus meet response times, productivity, and quality of service desired? Can we predict the behaviour of remote computing resources to develop strategies for dynamic management according to the Cloud Computing paradigm?

We start as a working hypothesis the fact that the conception and development of flexibly processing models based on schemes of Cloud Computing can overcome some drawbacks on this issue. These include the supply of processing capacity in running applications when they are executed on embedded devices with limited performance; and the auxiliary use on demand of cloud computing infrastructure will provide flexibility in order to execute the necessary tasks and mechanisms to support the service quality maintenance, even with process low-capacity devices.

This paper is organized as follows: in Section 2, we review the related work about this issue; in Section 3, some important issues on real time in embedded and mobile systems are highlighted and the contributions of this work about them are exposed; in Section 4, the formal framework of the computational model is introduced; in Section 5, it is exposed how to predict the network delay; next, Section 6 describes an application example in which the model is simulated. The paper is finally concluded in Section 7 and some approaches for future work are also pointed.

2. Related Work

The research areas related to the topics covered in this paper are experiencing an intense research activity, as evidenced by the number of recent works found. The following briefly describes the current state of knowledge on the different aspects that encompass this research. The conclusions of this related work study are also indicated.

The increasing development of embedded systems and mobile computing systems in recent times has allowed its extension into new business areas. Advanced e-commerce applications, positioning, monitoring and surveillance, health, wellness and leisure, among others [1–3], represent opportunities to exploit the high degree of penetration of these devices among the population and its new features. However, to properly continue the development in these areas, it is necessary to make a qualitative leap in design, taking into account the requirements of performance and response time that these applications require.

The quality of service (QoS) is essential to ensure the proper operation of many applications and, for embedded systems, it becomes a critical aspect due to the inherent processing limitations normally shown by devices.

In these applications, embedded systems must provide predictability both in response time and quality of the results. This feature raises them to the status of real-time systems [4]. In such systems, the validity of the results is given not only for their correction but also because they are on time. That is, there are some restrictions that limit the time of its operation. Therefore, the layout and design of these systems should propose architectures that address the aspects of correctness, adaptability, predictability, security, and fault tolerance.

There are many works that provide solutions to these issues. The technological evolution of devices currently provides sufficient performance to implement complex planning strategies on them. These strategies delegate to a real-time operating system embedded in devices, the execution planning, and management of tasks to meet the constraints imposed by the applications [5–7]. In environments involving multiple devices, it is possible to establish planning methods that take into account multiprocessing scenarios in one [8, 9] or more embedded elements with heterogeneous characteristics [10, 11]. Another step further in this strategy is the embedded distributed systems interacting through a communications network. For these cases, proposals also have been made to ensure the quality of service of the results [12, 13].

Although such solutions provide significant levels of satisfaction of restrictions, some applications may be temporarily overwhelmed by the characteristics of its execution and may require extra performance that exceed its capacity. In these cases, previous systems should decline to execute the tasks that are beyond excessive response times to ensure compliance with real-time scheduling. However, such decisions may cause service interruptions, unaffordable in some critical applications. For example, e-health systems that monitor and control biometric variables of several individuals simultaneously, may experience increased computing needs due to the increase of the number of individuals to supervise or, for example, a traffic management system in a Smart City in which each vehicle collects and transmits status information to other vehicles, can be equally saturated in dense scenarios with multiple vehicles.

One end in the configuration of distributed systems is systems composed essentially of sensors/actuators that lack processing power to make decisions on their own. These elements, which basically operate as transceivers, transmit the information in order to be treated remotely by a host with sufficient capacity [14, 15]. However, this approach may underutilize the possibilities of devices themselves, decrease

fast response, and require additional infrastructure to maintain permanent communication for proper processing. In those sensor networks scenarios where sensors have absolutely no chance for the execution of these tasks [16, 17], they incorporate only the minimum functions to protect sent or received data by using simple techniques, and generally, periodic audit and control strategies are released to check if any device has been compromised [18–20].

A computer model to address cases in which the computing needs go beyond the capabilities of the device is Mobile Cloud Computing (MCC) [21–23]. In this paradigm, the workload is divided between distributed devices and a central element located in the cloud. Thus, devices can move processing needs to the cloud (computation offloading) where they will run as services on Cloud Computing servers [24, 25]. The most common uses of this paradigm are primarily targeted to extend the battery life of mobile elements [26–28], without considering the versatility that the remote computer can provide to facilitate the provision of adequate QoS. Proposals are arranged under two different approaches [29, 30]: on one hand, systems that try to adapt existing applications by identifying portions of offloadable code [31–33], and on the other hand, new applications having into account this idea in conception and preparing the process code accordingly [34, 35]. In all those proposals, the influence of environmental conditions in process planning is also twofold: first, works that consider a static scenario in which it is possible to plan the optimal execution strategy [36, 37] and, secondly, dynamic environments where communication conditions can be varied [38, 39]. In these methods, although they offer valid solutions for some contexts and applications, the maintenance of quality of service in the results for realistic application scenarios remains as an open problem.

In Mobile Cloud Computing strategies, the communications management and its role in maintaining response times in systems where it is applied are especially important. By extension, maintaining quality of service in the field of communications is one of the areas of major research intensity. In this field, there have been contributions related to the intelligent adaptive analysis of service times [40] and architectures oriented to meet the QoS requirements have been proposed [41, 42]. These works not only take into consideration parameters of energy efficiency but they also suggest strategies for compliance with real-time specifications [43] by routing and classification of net traffic.

The joint use of distributed computing infrastructures to ensure QoS is an option that is also being widely discussed recently [44, 45]. The services that combine the resources of distributed infrastructures of different types (clusters, grids, cloud, etc.) are a mechanism that reinforces QoS commitments for these systems as they can use other computer elements from their immediate environment. Other approaches provide greater communication capacity (bandwidth) to connected devices when required in order to reduce response times in the cloud access [46]. Therefore, these strategies facilitate specification of RT restrictions on remote computing elements.

Concern for maintaining response times of cloud computing models is present in many works, where QoS solutions for cloud computing systems have been analyzed and proposed [47–49]. Although its focus is specifically targeted towards multimedia applications (online games, theater videostreaming) its findings can be transferred to other sectors (business, telemedicine, automotive, etc.) for the provision of remote services [50]. However, the results of these works are especially dependent mostly on execution context and communication conditions.

Regarding strategies providing flexibility in computing processes, the application of imprecise computing techniques [51, 52] to the tasks execution of application involved can offer satisfactory solutions. With this technique, processes are broken down into two types of tasks, mandatory and optional, for parameterizing restrictions and establishing checkpoints to explicitly manage response times. However, the sacrifice of processing time for a task is at the expense of making a bounded mistake and therefore providing an inaccurate response. Most systems using this model assume that tasks to plan are monotonous and that the error is a function of the amount of work discarded. These algorithms seek a balance between output quality and runtime, based on minimizing objective functions such as average error, total error, maximum error, number of optional tasks eliminated, and average response time. The original imprecise computing model assumes that the input values are precise for each task and the mandatory and optional time can be known a priori [53]. Other contributions deal with imprecise computing systems in cooperative tasks in which the results of operations are dependent on each other. When the result of a producer task is partially wrong, the consumer task must somehow compensate for this error. This results in increasing processing time for subsequent tasks and changes at preset times for each individual task [54]. The conception of this technique is essentially oriented to process planning in real-time systems and compliance with timing constraints [55, 56], but also in maintaining the quality of the results [57–59].

3. Real Time in Embedded Systems Issue

From the study carried out in the previous section, we will highlight some of the major problems obtained in the development of real-time embedded systems as well as major contributions of this research in the resolution of themselves.

3.1. Problem Statements. The problems emphasized are as follows:

(a) The embedded and mobile systems with real-time functioning need to properly respond to their design considerations in most cases. Improvements in computer technology as multicore and multiprocessor systems contribute to this effort when they have conveniently handled with appropriate planning methods. However, these new capabilities do not provide mechanisms to eventually increase the processing load beyond a specified level and this limits its application to the established functioning situations.

New applications of cyber-physical systems operating in the real world lack the flexibility to deal with situations

of processing demand when interactions excess with the environment are required. Having more powerful systems for these cases which can be unfeasible for many environments due to higher power requirements would require.

(b) The use of remote resources of cloud computing from mobile devices according to the scheme *Mobile Cloud Computing* can be a strategy to ease devices processing loading based on the needs of each moment. This approach is not sufficiently developed for all cases and their wider use is oriented towards saving consumption in the running applications rather as a strategy with flexible planning of the workload. The lack of adjustment mechanisms in the processing needs produces rigid planning strategies and it can lead to poor judgment on what parts of the application should run on locally and which ones remotely.

(c) In regard to the use of Cloud Computing systems by themselves, there are just a few real-time applications which rely on its performance due to the difficulties to fully predict their response times. In addition, mobile systems which experiment a wide variety of contexts and different situations in bandwidth and cover are the most affected, since the delays caused by the network can be variable depending on a lot of factors. In these cases, it is difficult that the planning strategies take into account the cloud resources and host remote processing in order to meet the requirements for applications with certain satisfying features.

3.2. Contributions and Significance of This Work. Power consumption and processing delay cost are very important aspects to be taken into account in embedded/mobile systems operation (particularly when they are powered by batteries). These two aspects are related because the settings on energy cost have effect on completion time of the tasks. However, we think that the processing delay is a more general aspect than energy cost for the embedded system operation because power mode is a global characteristic of the embedded system and it is no feasible to establish different consumption limits to each individual task; and there are some variables of MCC paradigm that could not be considered within the local model because they occurs outside of the embedded system (e.g., data transmission over the net and task execution on cloud server). So, this research focuses on the response time as a key aspect to provide flexibility to the application processing.

Therefore, the contributions of this paper to solve the problems described in previous subsection are as follows:

(a) The configuration of systems with sufficient computing elements to adequately address the most common situations is often the most popular solution to find a balance between installed capacity and consumption needs. In this paper, we aim to leverage the same configuration and support it with remote computing elements hosted into the cloud in order to increase the computing capabilities. Although this idea is not new, the novel approach is geared especially for applications with real-time requirements. The contribution to achieve this objective focuses on developing a computational model which ensures the formal framework to provide expressive capacity needed and to address specific problems with real-time requirements using the Mobile Cloud Computing paradigm.

Several previous works exist proposing MCC operation schemes, but they lack the necessary flexibility to take into account both priority and response time instead focus on other goals such as power efficiency, faster response of the device or routing, and classification of traffic in the network level.

In the following section the details of the specification will be presented:

(b) Given the lack of flexibility in the running applications, in this paper the use of imprecise computation techniques is proposed to decide the tasks to be performed on the embedded device. The application of imprecise computation to Cloud Computing paradigm is a novel approach to address the compliance with time constraints. The imprecise computing provides mechanisms for processes scheduling with maintenance response times criteria. Combining these methods with processing schemes Mobile Cloud Computing can provide strategies to accomplish with adequate QoS when the features functioning so require. Besides the above, we propose to use strategies implementation based on stored logic to provide higher prediction to the running operations. This method is based partly on our previous results and work in the design area of arithmetic operators and specialized processors [60, 61].

(c) As discussed in the related work, the techniques of maintaining QoS for open networks are producing some advances that may allow their application to Cloud computing strategies for specific functioning scenarios. However, it cannot be extended to systems with real-time constraints. The contribution to this issue that it is carried out in this paper is to implement a hybrid method of monitoring and predicting the performance of communication which will go periodically determining what are the delays introduced by network during the access towards remote processing resources. The novelty of this method lies in the combination of online delay measure with offline historical data depending of aspects such as working environment and running application. The integration of this procedure in the above computation model will allow taking into account the costs associated at any time and take better criteria in making planning decisions.

4. Flexible Computational Model

In this section, the formal framework of this issue is described in order to state the problem formulation and to define the flexible computing proposals.

4.1. General Framework. The computational model proposed in this section specifies the aspects involved in scheduling the tasks of an application in a multiplatform execution environment with heterogeneous characteristics. For such scenarios, the set of available computing platforms would be defined by

$$\wedge = \{P_1, P_1, \ldots, P_m\}. \quad (1)$$

The set of tasks will be defined by the application workload. The variety and type of tasks for each workload depend

Table 1: Function definitions.

Name	Definition
Start	Start: $\Gamma \rightarrow \wedge$
Delay	Delay: $\Gamma \times \wedge \rightarrow \mathbb{R}^+$
Data	Data: $\Gamma \rightarrow \mathbb{R}^{2+}$
Net	Net: $\wedge^2 \times \mathbb{R}^{2+} \rightarrow \mathbb{R}$

on the scope of each system. Let Γ be the workload of an application environment:

$$\Gamma = \{t_1, t_2, \ldots, t_n\}, \quad (2)$$

where each t_i is a task of the application environment. In a real-time system, each of the tasks will be associated with a deadline from which the result is invalid (hard) or loses value (soft). Therefore, for each task (t_i), constraint(t_i) informs the maximum duration of execution as defined by the following function:

$$\text{constraint}: \Gamma \longrightarrow \mathbb{R}^+. \quad (3)$$

In this kind of system, to undertake the scheduling, the delay cost associated with the execution of the tasks of workload on each platform will be known. The delays are defined by the following functions: *start*, *delay*, *data*, and *net*. Its definition is detailed in Table 1.

The *start* function indicates the native platform of a task, that is, the platform on which the task is created for execution. Generally this platform will be the interface system with the user's environment or the embedded system itself.

The *delay* function obtains the execution time, in time units, when running the task on a given platform, without taking account of its current workload, that is, if all processing resources of the platform were dedicated to running that task. In this way, $\text{delay}_p(t_i)$ gets the delay of executing the task i on platform p. In an embedded system, the results of this function may vary due to different operation scenarios or configuration of the system. For example, at low power consumption settings, the processing resources can be reduced (e.g., lowering the clock frequency or disabling cores) and produce higher delays. Other maximum performance scenarios could improve standard delays.

In addition to the computational requirements, in distributed processing scenarios it is necessary to know the amount of data required to run the task. Thus, the *data* function obtains this size of data required for processing a task and the size of the results produced. It is independent of the platform on which it runs.

Finally, the *net* function obtains temporary costs associated with data communication between platforms through the communication network. That is, the function $\text{net}_{Pa,Pb}(\text{data}(t_i))$ returns both delays caused by the transmissions of data required to run t_i from the platform P_a to P_b and caused to return the results from P_b to P_a. It is assumed that there is no delay to move information on the same platform; namely, $\text{net}_{p,p}(\text{data}(t_i)) = 0$.

As defined by the above functions, the execution cost of a task (t_i) on a platform (p) will be defined by the following expression:

$$\text{TimeCost}_p(t_i) = \text{net}_{\text{start}(ti),p}(\text{data}(t_i)) + \text{delay}_p(t_i). \quad (4)$$

Nevertheless, considering each platform is running a set of tasks, the response time of the task must consider the delay cost on the platform of pending tasks when this new task (t_i) arrives. In this way, let ${W_p}^t(t_i)$ be the aggregate delay cost of the list of tasks assigned to the p platform at time t, with a deadline less than DeadLine(t_i). This cost is dependent on the platform and on its internal configuration of processing elements. For example, a platform can be composed of many processing elements capable of parallel computing.

According to this aggregate cost, the expression (4) on the delay cost of executing a task (t_i) on a platform (p) at time t is set as follows:

$$\text{TimeCost}_p{}^t(t_i) = \text{net}_{\text{start}(ti),p}(\text{data}(t_i)) + \text{delay}_p(t_i) + {W_p}^t(t_i). \quad (5)$$

The execution of the tasks in this model does not consider accessing to shared resources other than the processor, so that no unnecessary delays occur by blocking resources. This assumption is consistent with many applications composed of many autonomous and noncollaborative tasks.

From the above processing cost expression (5), a scheduling method based on *Shortest Job First* (SJF), or *Shortest Remaining Processing Time First* (SRPT) in preemptive case, can be implemented to minimize the average delay time of application workload. It is proven that these scheduling algorithms (SJF and SRPT) are the optimal online methods to minimize the average delay time for a single processing platform [62, 63]. In addition, recent studies about this issue demonstrate that they can be also very competitive even in multiprocessing platforms [63].

However, although simple, these methods do not take into account compliance with the time constraints present in the tasks of real-time systems.

4.2. Real Time Embedded-Cloud Scheduling. The real-time constraints in the execution of tasks imply a temporal restriction or deadline for each task related to the time at which the results must be ready. After that time, the results have a lower or null value.

The following function obtains the remaining time of each task in which the results must be ready:

$$\text{DeadLine}: \Gamma \longrightarrow \mathbb{R}^+. \quad (6)$$

As time passes, the value of DeadLine approaches zero for each task.

It is not the purpose of this work to propose a method of planning as complex as described in the previous scenarios. This problem has been studied in other researches [64, 65] and is, in its most general version, of category NP problem. This scheduling can only be resolved by heuristic or search algorithms which consume a considerable part of processing resources [66].

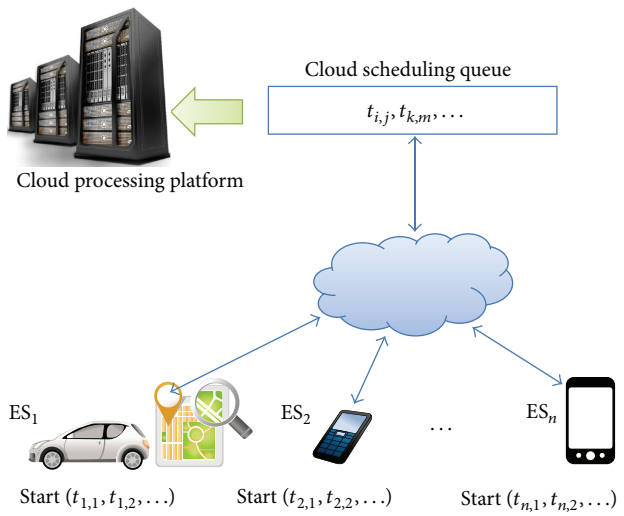


FIGURE 1: Infrastructure as a service.

Instead, in this section we will focus on a subset of this general problem in which there are only two heterogeneous platforms. This scenario is quite common in many contexts, for example, in a workstation which has central processor unit (CPU) and a graphics accelerator (GPU) installed on it [67]. Under this principle, we consider therefore that applications will be launched at an application platform (which corresponds to the embedded system) and there is another additional computing platform (cloud infrastructure) on which move part of the processing work. Processing platforms may have different performance and characteristics. Therefore, in this study, expression (1) shall be defined by the following set of available computer platforms:

$$\wedge = \{P_{ES}, P_{Cloud}\}, \tag{7}$$

where P_{ES} will correspond to the computing platform of embedded system and P_{Cloud} to the processing platform available in the Cloud. Furthermore, we assume that tasks will be created in embedded system as part of the running application. Therefore, start$(t_i) = P_{ES}$.

The Cloud platform may have a nonexclusive use of the application and it can serve many devices corresponding to the same or several different applications, so this approach can be extended to scenarios in which various embedded or mobile systems share the cloud infrastructure to complement its performance. This case corresponds to infrastructure as a service model, where the Cloud platform can execute many tasks when required. Figure 1 illustrates this scenario.

The scheduling method proposed for this case aims maximize compliance with the time constraints reducing the time spent on scheduler management. We do not intend in this work provide the optimal solution in the execution of the tasks of the workload, but to provide a feasible solution for the management of real-time embedded systems valid for many nowadays applications. According to this goal, heuristic used is to schedule tasks on platforms that meet their timing constraints. In each platform, tasks are placed in a dispatcher queue ordered by shorter DeadLine (EFD—*Earliest-Deadline First*). This method has been proven effective even in multiplatform systems under certain conditions [68, 69].

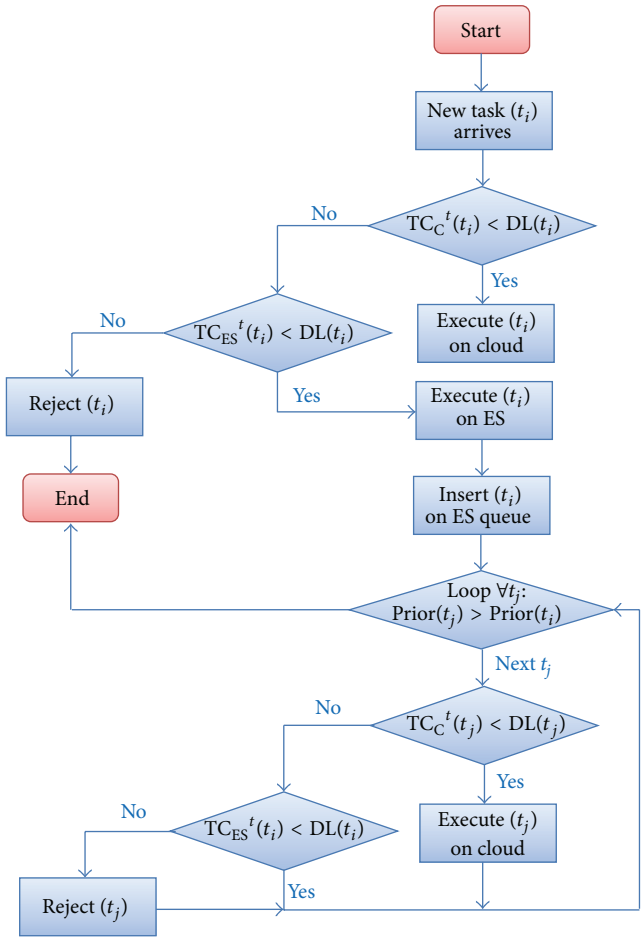


FIGURE 2: Imprecise Computation Scheduling.

4.3. Imprecise Computation Scheduling. The main idea of this method to adjust the amount of tasks that are executed depending on the time available and the computational power of the system lies in considering the relative priority of each task to decide the execution order of workload. The priority of each task is given by its importance in the application or in the security of the system. This priority may be related with *DeadLine* of the task or have a different value in line with how critical is the task in the overall application.

According to the imprecise computation technique, lower priority tasks may be discarded when the time constraints require a partial execution of the application. In such cases, only the critical tasks will be executed in the time available, obtaining a partial functionality. For this purpose, the system will have a function called *Priority* will determine the priority of each task:

$$\text{Priority}: \Gamma \longrightarrow \mathbb{R}^+. \tag{8}$$

The scheduler method is described in Figure 2. As in the previous subsection, first, when the task is created in the embedded system, it is determined what platform can run the

task within the available time: (a) if both can, decision about which platform to choose can be established according to system configuration: you can derive all possible processing to the cloud if compliance with the restrictions or select local processing by default; (b) if no platform can deliver on time, the task can be rejected with a warning of out of time.

Once decided the platform in which the task will be executed, the management of tasks in scheduling queue is driven by *Priority* value of the tasks. When a new task arrives to the platform, it will be inserted in the position corresponding to their priority. In this case, the system must ensure that all tasks with lower priority than it can meet their time constraint. Otherwise, the task can be scheduled on the cloud platform if possible, and if not, reject the task execution. During the time in the platform scheduler queue, the conditions of remote execution can change and therefore some tasks can be driven to the cloud.

This method is not geared to meet with the deadlines of all tasks but to execute them according to their importance. Therefore, it is only applicable in applications for which it is assumable this type of operation. In this case, a partial execution of the application will be made when the system is not able to perform all tasks on time and an overload exists. In this case, when the most important tasks are processed in first place, the imprecise result is feasible for the user. This behavior is consistent with the assumptions made in the introduction where the possibility of addressing workloads that exceed the theoretical computing capabilities of the devices was proclaimed.

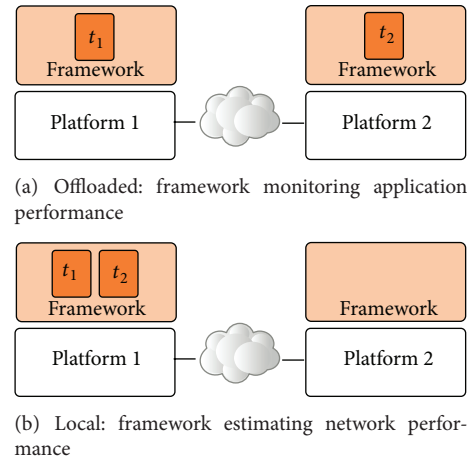
An application example in which implements this method is described in Section 6 of this paper.

5. Net Performance Prediction Methodology

The proposed framework requires measurement and prediction of the network performance between platforms. First of all, network performance concept must be properly specified according to application requirements. For example, in some real-time applications a maximum delay must be guaranteed; however, in other applications, a stable minimum bandwidth is enough for producing valuable results.

Several tools have been developed and are available to flexibly measure different network performance parameters [70–73]. However, periodically probing the performance between each pair of platforms is a resource intensive task and poorly scalable [74]. For this problem, a number of solutions have been proposed, most of them in the context of heterogeneous computing [75]. Again, the nature of the application to be implemented will determine the best strategy to take. In addition, most of the networks linking mobile platforms use shared medium among a number of users. This condition often makes network performance very difficult to predict.

In this section, a method for measuring and predicting network performance is experimented. For this purpose, a standard wireless network is used as a test environment and two processes are run. These processes implement task t_1 and task t_2 shown in Figure 3. Just for the purpose of this test, the first process captures input from a camera device; then, it performs frame selection and sends the relevant frames to the second process; the second process performs some common operations typically involved in signal processing tasks, such as Fast Fourier Transform (FFT). The number of relevant frames exchanged between processes will depend on the data captured (variable workload).



(a) Offloaded: framework monitoring application performance

(b) Local: framework estimating network performance

Figure 3: Test environment with two processes and two platforms linked by a network.

In this scenario, two factors must be taken into account when predicting network performance. First, the transfer rate received by the processes when running on different platforms (Figure 3(a)); this will depend on the application workload, which in turns will change over time (in our experiment, random frame selection at different rates has been implemented). Second, the available network bandwidth when the processes are run on the same platform (Figure 3(b)); this will also change over time depending on the network usage by other users or applications. In other words, if the two processes are run on the same network node (platform), the network link performance between that node and other candidate platforms must be estimated for possible offload; otherwise, the network suitability can be deduced from the transfer rate shown by the communicating processes.

In order to evaluate the network performance, a number of multiplatform tools are available. One simple solution is the Iperf (*http://iperf.fr*) tool. It allows to set target nodes (servers, in Iperf terminology), by running an Iperf process in server mode on each platform. With a period of 20 seconds, an Iperf process is launched in client mode, in order to measure the bandwidth by sending random data during one second over a TCP connection. The best period and test duration are highly dependent on the application and network characteristics, and therefore it will require further setting. In addition, other network performance parameters could be considered if required by the application; for example, the Iperf tool can also measure average delay (although this could be also achieved by a standard ping) and packet loss.

For testing purposes a framework script has been developed. It is in charge of running the network performance tests. In addition, it acts as a proxy between processes and the network devices, so it can monitor the effective transfer rate

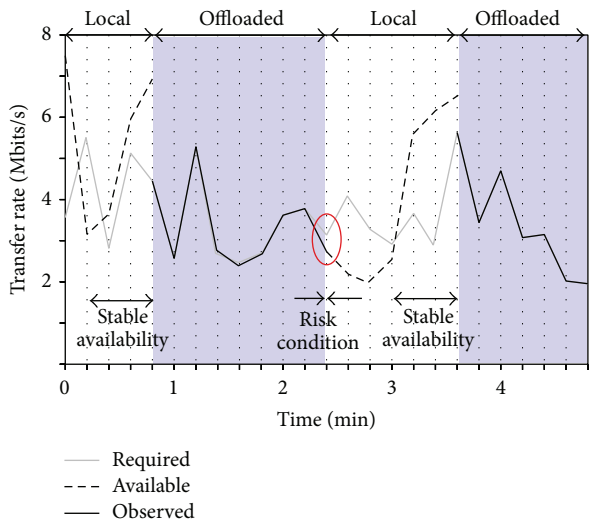


Figure 4: Network performance: required, observed, and maximum available.

between offloaded processes over time. Moreover, it logs the measures taken for further prediction.

In summary, the proposed method consists of three parts: (1) periodic measure of relevant performance parameters provided by the network, (2) communicating tasks monitoring, for current shown performance, and (3) analysis of past performance data for reasonable prediction.

For part (1), the aforementioned tools for network performance measuring has been used. By using those tools, each platform constructs a history database that can be used later in order to help to find stability periods from the network performance point of view.

In part (2), the provided framework is in charge of checking that processes are running under affordable network conditions. Otherwise, tasks must be rescheduled for execution on the same platform (if it is possible, according to current system workload, priorities, etc.), avoiding network communication.

The result of the analysis performed in part (3) can be used in conjunction with current provided performance values, in order to support the decision about the platform on which the task should be run. The past performance data can be dynamically configured depending on aspects such as working environment and running application. To adequately compose this function it is necessary define the former operation details of the embedded system.

Figure 4 shows different aspects of the transfer rate evolution in the experiment. The grey line shows the required transfer rate between the running processes, in order to fit real-time requirements. This changes over time, because it depends on application workload. In the conducted experiments, the processes exchange selected frames, and therefore the required transfer rate will vary depending on captured data.

When the processes are run on the same platform (sections of the chart labelled as local in Figure 4), the framework is measuring the network conditions. In the experiment, this is the maximum transfer rate that could be achieved if the process were offloaded. It is also changing over time, because of the effect of different factors as mobility or other applications sharing data through the network. The result of these measures is drawn with a dashed line. As shown in the chart, a sequence of three consecutive periods of increasing available bandwidth is considered a stability condition, so it is decided to switch to offloaded mode. In other words, the processes are run on different platforms when the network conditions history is good enough to reasonably guarantee required transfer rate. In a real scenario, this decision would depend on a number of factors, as the application nature and the scheduling policy.

Figure 5: Vehicle-to-vehicle environment.

When the processes are run on different platforms (sections of the chart labelled as offloaded in Figure 4), the framework is checking if the transfer rate is enough to fit process requirements. The result of these measures is drawn with a solid line. When a risk condition is detected (point marked with a red circle in the chart), the system is switched back to local mode. Again, the particular definition of risk condition would depend on the application nature in a real scenario.

6. Application Example

In this section, an application example is exposed in order to test the operation of the system in a today's context with a realistic workload. The proposed scenario is a *SmartDrive* application for autonomous decision-making aimed at providing increased security and convenience to the user. This application is part of the *SmartCity* concept and the construction of vehicular networks in which several vehicles intercommunicate with each other and with city infrastructure to exchange information [76].

In this application, the vehicle is equipped with a variety of sensors that capture the state of the environment to inform the driver and adapt driving to traffic and context circumstances (see Figure 6). These features entail the execution of signal processing tasks from the data collected by these sensors. Typically these tasks have real-time constraints, since the processing results have to be on time to make decisions on the fly. Figure 5 illustrates a real scenario in which several vehicles interact.

To provide processing capabilities that this sensing requires several configurations exists: (a) each sensor subsystem can have its own embedded system; (b) the system

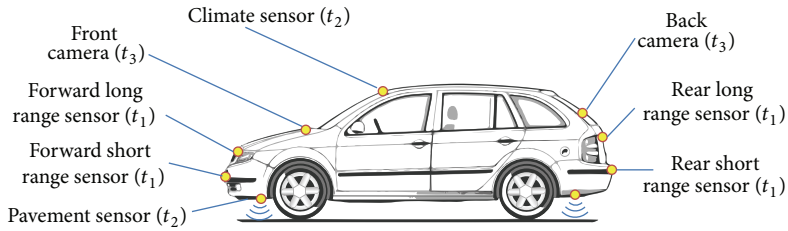


Figure 6: Simplified Automotive Sensors.

has a central processor built-in-vehicle running all tasks; (c) the processing device may be provided by the user and integrate with the vehicle sensors to collect data. This last option provides greater flexibility to the user and allows the installation of applications (apps) on mobile devices carried by the user (phones, tablets, etc.). This scenario is increasingly common in many applications that can be downloaded to mobile user devices to interact with sensors and actuators of our home, work or transport vehicles. In this case, the capabilities that can be offered by the application depend on the performance of the device used.

There are many signal processing elements in the *SmartDrive* application with real-time constraints that can be addressed with the proposed model; however, to illustrate a glaring example of how it works, in the example analyzed, it is considering only 3 kinds of tasks which analyze different types of signals collected by sensors.

> *Task 1.* It is responsible for analyzing the signal from the radar sensors to obtain the following information from other vehicles: distance, relative speed, and risk of collision.
>
> *Task 2.* Analyzes the signal from a set of sensors that measure the physical environment of the vehicle to obtain the following information: type of pavement, asphalt status, static friction, moisture, rain, and so forth.
>
> *Task 3.* Analyzes the traffic signs and posters from digital images captured by the car's cameras to identify the type of each signal and interpret the texts they contain.

For example, Figure 6 shows a schematic of the vehicle with various sensors and what tasks supply the signal collection [77].

Results obtained by Task 1 enable vehicles to know where the vehicles in its vicinity are and what they are doing and to perform actions such as the following: forward collision warning, automatic braking if there is a risk of collision, intersection movement assist, not passing warning, and so forth. Results from task 2 will allow calibrate the operation of dampers, distance to the ground, brake pedal feel, traction control, ABS, and so forth. Results obtained by task 3 inform the driver of road signals, speed limits warning, road departure warning, and so forth.

Tasks are started in the vehicle as a result of events produced by the sensors when capturing information (Start(t_i) = car). Thus, when sensors located another vehicle the task 1 is started, when a traffic signal or a sign is identified starts a task 3. Task 2 runs periodically to check the physical condition of the environment.

According to the proposed computational model, each task type (t_i) requires a processing time known through $\text{Delay}_{\text{SE}}(t_i)$ function, needs a data size measured by $\text{Data}(t_i)$, and its results must be ready before $\text{DeadLine}(t_i)$ to be useful for SmartDrive application.

With this simple example described, the system performance can present problems in certain situations when the processor cannot provide sufficient computational power and/or when the frequency of arrival of the tasks overflows its processing capacity. These situations could occur if the user does not have a sufficiently powerful device to perform all tasks or if the vehicle is in intensive scenarios with many circulating elements and traffic signs (e.g., city centers).

In realistic implementation of this idea, requirements on computational capabilities of such devices must be set by the manufacturer according to criteria that guarantee the safety of the driver. for example, devices able to run real-time type-1 tasks (to avoid collisions) in dense traffic contexts. Specifically, it means

$$\forall i \text{ over time}, \text{Delay}_{\text{SE}}\left(\text{task } 1_i\right) < \text{DeadLine}\left(\text{task } 1_i\right). \quad (9)$$

From this minimum restriction, all other features of "SmartDrive" system can be provided by the manufacturer, as value-added services, depending on the power of the mobile device and/or under the possibilities for communication with the cloud.

In situations in which the user's system cannot meet the time constraints in the execution of all tasks, the integration with Mobile Cloud Computing paradigm can provide the necessary performance offloading some of the processing work to the cloud. Furthermore, this solution offers to the application signal processing capacity for an extensive fleet of vehicles and sharing computing infrastructure in the cloud for it. With this configuration a multitude of embedded systems (vehicles) can share the same cloud platform to collaborate with the necessary processing work as shown in the figure scheme (Figure 7).

The flexible framework proposed in this paper offers an approach to the problem of tasks scheduling and decisions about upload execution to the cloud to achieve the best quality of service. To do this, the application must be composed by a set of task and a priority associated with each type of task must be provided in order to be used as a criterion

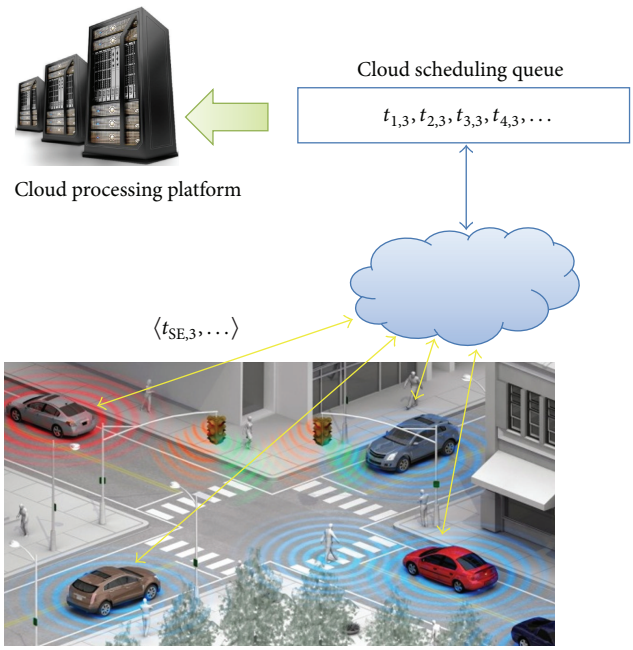


Figure 7: Mobile Cloud Computing framework for SmartDrive.

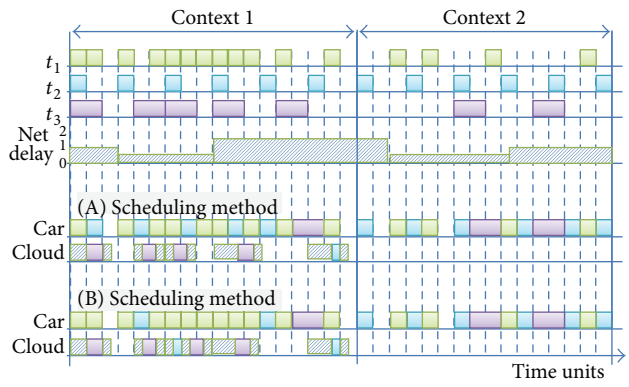


Figure 8: Scheduling example of a simulation workload under MCC paradigm.

Table 2: Simulating workload.

Task type	Context 1	Context 2
Task 1	12	4
Task 2	6	6
Task 3	5	2

for planning. In our example, the order of priority of tasks is as follows (high to low priority): 1, 2, and 3. That is to say, identify closest vehicles to avoid collisions has priority over environmental analysis and traffic signals interpretation.

Figure 8 depicts a scheduling example simulating a workload for this application with two operating contexts and different task arrivals. Table 2 shows the workload of this example.

The MCC paradigm is implemented between automotive embedded system provided by the mobile device of the user and the cloud. The figure shows two scheduling methods: A-method driven only by deadline and B-method based on previous priority level described.

Table 3: Scheduling results of tasks.

	Context 1		Context 2	
	A-method	B-method	A-method	B-method
Avg. turnaround time (tu)				
Task 1	1.3	1	1.25	1.25
Task 2	1.9	1.75	1.3	1.3
Task 3	2.5	2.7	2.5	2.5
Avg. wait time (tu)				
Task 1	0.3	0	0.25	0.25
Task 2	0.91	0.75	0.33	0.33
Task 3	0.9	1.1	0.5	0.5
Task lost				
Task 1	3	0	0	0
Task 2	0	2	0	0
Task 3	0	0	0	0

The task operation conditions are the following: task 1: deadline = startTime + 1; task 2: deadline = startTime + 1; task 3: deadline = startTime + 2; $\text{delay}_{\text{Cloud}} = 1/2\text{delay}_{\text{Car}}$ for all tasks. When the embedded device cannot meet time constraints, the tasks are sent to the cloud infrastructure if they can be processed on time according its *TimeCost* function.

The simulation results show that none of the methods can schedule all tasks of the application on time when cost exceeds the capabilities of embedded device. Nevertheless, the imprecise computation (B-method) allows less important tasks are the remaining unexecuted in that case. The example shows that with A-method three tasks of type 1 have been lost, while the B-method only lost two tasks of type 2 and none of type 1. Obviously, the results depend on the simulations made and the cadence of work that comes; however, the same behavior has been observed in all cases.

Table 3 shows the scheduling results of task by type.

Statistical data in Table 3 clearly shows that B-scheduling method does not produce loss of completion time of most priority tasks and improves turnaround and wait time for them.

With this priority criterion, the system ensures that in case of not having enough processing power, the lower priority tasks will run last. Analysis of the internet coverage and bandwidth of the device can move their execution to the cloud to be processed in parallel with the work that runs on the device and offer their additional service. In addition, some kind of tasks (e.g., task 3 of image analysis) may have faster execution in the cloud where it can take advantage of powerful computing resources and where will not be subject to restrictions of power consumption or silicon size.

The simulation results about throughput and utilization of computing resources are shown in Table 4. Although the utilization of the embedded system in the car is the same

Table 4: Throughput and utilization of computing resources.

	Context 1		Context 2	
	A-method	B-method	A-method	B-method
$\text{Throughput}_{\text{Car}}$	9 task 1	12 task 1	4 task 1	4 task 1
	5 task 2	2 task 2	6 task 2	6 task 2
	1 task 3	1 task 3	2 task 2	2 task 2
$\text{Throughput}_{\text{Cloud}}$	0 task 1	0 task 1	not used*	
	1 task 2	2 task 2		
	4 task 3	4 task 3		
$\%\text{Utilization}_{\text{Car}}$	88.8%	88.8%	87.5%	87.5%
$\%\text{Utilization}_{\text{Cloud}}$	25%	27.7%	0%	0%

*This vehicle does not use the Cloud Computing resources in this simulation context. However, Cloud server can be used at this time by other vehicles in the system.

in both scheduling algorithms, with the same environmental conditions the B-method uses more resources of the cloud.

7. Conclusions and Future Work

The future is coming. The number and variety of applications for embedded systems and mobile devices are growing. It is getting more necessary to provide methods and higher performance to execute applications with real-time constraints on these devices. Many of these applications are characterized by need of signal processing-intensive as a result of processes of sensing the environment in which they run to provide services to the user.

Mobile Cloud Computing is the key paradigm that provides the necessary processing power. It is referred to as the infrastructure where both the data storage and the data processing happen outside of the embedded system or mobile device. Therefore, cloud-based mobile apps can scale beyond the capabilities of any embedded or smartphone system.

One of the biggest challenges of this paradigm is the integration between the two infrastructures. That is to say, scheduling processes for execution considering the many aspects involved, especially when tasks are real-time constraint and cloud resources are accessible through the public communication infrastructure.

In this paper, we have presented a solution to this challenge that uses computational techniques to determine the most appropriate scheduling design between the local device and the cloud. To do this, a computational model based on imprecise computation method is proposed that provides flexibility for running applications over embedded systems. A method to know the delay induced by the network has been designed. This method is used to predict the bandwidth and the delay costs associated with communication and remote processing in the cloud. Thus, this type of apps has the power of a server-based computing infrastructure accessible through an embedded or mobile device, in which is taken into account, for the proper scheduling process, the extra delays associated with the remote access. With this model, specification and processing applications as a series of tasks with timing constraints are allowed. It is possible to prioritize their execution based on priority parameters, so that in the event of being unable to meet the computational requirements to processing all tasks, the most important ones have been satisfied and user satisfaction has been maximized.

A simple application example has been developed to show a real scenario that meets the points raised in this research. It is also made a simulation which shows the simplicity of the proposed model and its ease to design scheduling tasks. As a result, it is found that model gives preference to compliance with the time constraints of critical tasks in first place.

Based on the current outcomes, our future work will be unfolded along two directions: one is extend the proposed model to consider more complex application scenarios: for example, composed by a collection of tasks with timing constraints that maintain precedence relationships and share the use of other system resources. The other direction is going further into the net delay prediction issue, because this is the key aspect in scheduling decisions that take into account the remote resources in an MCC context.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work is funded by grant from the Conselleria de Educación, Cultura y Deporte of the Valencian Community Government, Spain.

References

[1] S. Tennina, M. Di Renzo, E. Kartsakli et al., "A protocol architecture for energy efficient and pervasive eHealth systems," in *Proceedings of the IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI '14)*, pp. 452–455, Valencia, Spain, June 2014.

[2] M. Penhaker, M. Stankus, J. Kijonka, and P. Grygarek, "Design and application of mobile embedded systems for home care applications," in *Proceedings of the 2nd International Conference on Computer Engineering and Applications (ICCEA '10)*, pp. 412–416, IEEE, Bali Island, Indonesia, March 2010.

[3] A. B. Waluyo, D. Taniar, and B. Srinivasan, "The convergence of big data and mobile computing," in *Proceedings of the 16th International Conference on Network-Based Information Systems (NBiS '13)*, pp. 79–84, September 2013.

[4] J. A. Stankovic, "Real-time and embedded systems," *ACM Computing Surveys*, vol. 28, no. 1, pp. 204–208, 1996.

[5] C. Tianzhou, H. Wei, X. Bin, and Y. Like, "A real-time scheduling algorithm for embedded systems with various resource requirements," in *2006 International Workshop on Networking, Architecture, and Storages, NAS'06*, pp. 43–46, chn, August 2006.

[6] D. Matschulat, C. A. M. Marcon, and F. Hessel, "A QoS scheduler for real-time embedded systems," in *Proceedings of the 9th International Symposium on Quality Electronic Design*, pp. 564–567, March 2008.

[7] F. Pianegiani, "QoS-based dynamic allocation in embedded systems: a methodology and a framework," in *Proceedings*

of the IEEE International Instrumentation and Measurement Technology Conference (I2MTC '11), pp. 1035–1039, May 2011.

[8] F. J. Cazorla, A. Ramirez, M. Valero, P. M. W. Knijnenburg, R. Sakellariou, and E. Fernández, "QoS for high-performance SMT processors in embedded systems," *IEEE Micro*, vol. 24, no. 4, pp. 24–31, 2004.

[9] C. Boneti, F. J. Cazorla, R. Gioiosa, and M. Valero, "Soft real-time scheduling on SMT processors with explicit resource allocation," in *Architecture of Computing Systems—ARCS 2008: Proceedings of the 21st International Conference, Dresden, Germany, February 25–28, 2008*, vol. 4934 of *Lecture Notes in Computer Science*, pp. 173–187, Springer, Berlin, Germany, 2008.

[10] M. L. Dertouzos and A. K. Mok, "Multiprocessor on-line scheduling of hard-real-time tasks," *IEEE Transactions on Software Engineering*, vol. 15, no. 12, pp. 1497–1506, 1989.

[11] B. Andersson and G. Raravi, "Real-time scheduling with resource sharing on heterogeneous multiprocessors," *Real-Time Systems*, vol. 50, no. 2, pp. 270–314, 2014.

[12] N. Thai, "Real-time scheduling in distributed systems," in *Proceedings of the International Conference on Parallel Computing in Electrical Engineering*, pp. 165–170, IEEE, 2002.

[13] Z. Zheng, X. Wu, Y. Zhang, M. R. Lyu, and J. Wang, "QoS ranking prediction for cloud services," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1213–1222, 2013.

[14] F. Bai, L. Huo, L. Bai, and J. Wang, "Distributed system for transfusion supervision based on embedded system," in *Proceedings of the International Conference on Computer Science and Software Engineering (CSSE '08)*, pp. 206–210, December 2008.

[15] W. K. Edwards and R. E. Grinter, "At home with ubiquitous computing: seven challenges," in *Ubicomp 2001: Ubiquitous Computing: Proceedings of the International Conference Atlanta Georgia, USA, September 30–October 2, 2001*, vol. 2201, pp. 256–272, Springer, Berlin, Germany, 2001.

[16] X. Chen, K. Makki, K. Yen, and N. Pissinou, "Sensor network security: a survey," *IEEE Communications Surveys and Tutorials*, vol. 11, no. 2, pp. 52–73, 2009.

[17] Y.-X. Li, L. Qin, and Q. Liang, "Research on wireless sensor network security," in *Proceedings of the International Conference on Computational Intelligence and Security (CIS '10)*, pp. 493–496, IEEE, Nanning, China, December 2010.

[18] X. Du and Y. Xiao, "A survey on sensor network security," in *Wireless Sensor Networks and Applications*, Signals and Communication Technology, pp. 403–421, Springer, Boston, Mass, USA, 2008.

[19] L. Gómez and C. Ulmer, "Secure sensor networks for critical infrastructure protection," in *Proceedings of the 4th International Conference on Sensor Technologies and Applications (SENSORCOMM '10)*, pp. 144–150, Venice, Italy, July 2010.

[20] X. Zhao, "The security problem in wireless sensor networks," in *Proceedings of the 2nd IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS' 12)*, vol. 3, pp. 1079–1082, November 2012.

[21] L. Guan, X. Ke, M. Song, and J. Song, "A survey of research on mobile cloud computing," in *Proceedings of the 10th IEEE/ACIS International Conference on Computer and Information Science (ICIS '11)*, pp. 387–392, IEEE, Sanya, China, May 2011.

[22] M. Satyanarayanan, "Mobile computing: the next decade," in *ACM Workshop on Mobile Cloud Computing*, 2010.

[23] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: a survey," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84–106, 2013.

[24] A. Aijaz, H. Aghvami, and M. Amani, "A survey on mobile data offloading: technical and business perspectives," *IEEE Wireless Communications*, vol. 20, no. 2, pp. 104–112, 2013.

[25] X. Zhuo, W. Gao, G. Cao, and S. Hua, "An incentive framework for cellular traffic offloading," *IEEE Transactions on Mobile Computing*, vol. 13, no. 3, pp. 541–555, 2014.

[26] A. Saarinen, M. Siekkinen, Y. Xiao, J. K. Nurminen, M. Kemppainen, and P. Hui, "SmartDiet: offloading popular apps to save energy," *ACM Sigcomm Computer Communication Review*, vol. 42, no. 4, pp. 297–298, 2012.

[27] A. Khairy, H. H. Ammar, and R. Bahgat, "Smartphone energizer: extending smartphone's battery life with smart offloading," in *Proceedings of the 9th International Wireless Communications and Mobile Computing Conference (IWCMC '13)*, pp. 329–336, July 2013.

[28] L. Corral, A. B. Georgiev, A. Sillitti, G. Succi, and T. Vachkov, "Analysis of offloading as an approach for energy-aware applications on android OS: a case study on image processing," in *Mobile Web Information Systems*, vol. 8640 of *Lecture Notes in Computer Science*, pp. 29–40, Springer, Cham, Switzerland, 2014.

[29] F. Liu, P. Shu, H. Jin et al., "Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications," *IEEE Wireless Communications*, vol. 20, no. 3, pp. 14–22, 2013.

[30] D. Kovachev, Y. Cao, and R. Klamma, "Mobile cloud computing: a comparison of application models," Computing Research Repository (CoRR), http://arxiv.org/abs/1107.4940.

[31] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: elastic execution between mobile device and cloud," in *Proceedings of the 6th ACM EuroSys Conference on Computer Systems (EuroSys '11)*, pp. 301–314, April 2011.

[32] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proceedings of the IEEE INFOCOM*, pp. 945–953, March 2012.

[33] H. Flores and S. N. Srirama, "Adaptive code offloading for mobile cloud applications: exploiting fuzzy sets and evidence-based learning," in *Proceedings of the 4th ACM Workshop on Mobile Cloud Computing and Services (MCS '13)*, pp. 9–16, June 2013.

[34] V. March, Y. Gu, E. Leonardi, G. Goh, M. Kirchberg, and B. S. Lee, "μCloud: towards a new paradigm of rich mobile applications," *Procedia Computer Science*, vol. 5, pp. 618–624, 2011, Proceedings of the International Conference on Ambient Systems, Networks and Technologies.

[35] X. Zhang, A. Kunjithapatham, S. Jeong, and S. Gibbs, "Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing," *Mobile Networks and Applications*, vol. 16, no. 3, pp. 270–284, 2011.

[36] C. Wang and Z. Li, "A computation offloading scheme on handheld devices," *Journal of Parallel and Distributed Computing*, vol. 64, no. 6, pp. 740–746, 2004.

[37] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: can offloading computation save energy?" *Computer*, vol. 43, no. 4, Article ID 5445167, pp. 51–56, 2010.

[38] S. Ou, K. Yang, A. Liotta, and L. Hu, "Performance analysis of offloading systems in mobile wireless environments," in

Proceedings of the IEEE International Conference on Communications (ICC' 07), pp. 1821–1826, June 2007.

[39] P. Angin and B. Bhargava, "An Agent-based optimization framework for mobile-cloud computing," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 4, no. 2, 2013.

[40] E. Gelenbe, R. Lent, and A. Nunez, "Self-aware networks and QoS," *Proceedings of the IEEE*, vol. 92, no. 9, pp. 1478–1489, 2004.

[41] B. Jennings, S. van der Meer, S. Bala-Subramaniam, D. Botvich, M. O. Foghlú, and W. Donnelly, "Towards autonomic management of communications networks," *IEEE Communications Magazine*, vol. 45, no. 10, pp. 112–121, 2007.

[42] T. Frantti and M. Majanen, "An expert system for real-time traffic management in wireless local area networks," *Expert Systems with Applications*, vol. 41, no. 10, pp. 4996–5008, 2014.

[43] Z. Hamid and F. B. Hussain, "QoS in wireless multimedia sensor networks: a layered and cross-layered approach," *Wireless Personal Communications*, vol. 75, no. 1, pp. 729–757, 2014.

[44] S. Delamare, G. Fedak, D. Kondo, and O. Lodygensky, "SpeQuloS: a QoS service for hybrid and elastic computing infrastructures," *Cluster Computing*, vol. 17, no. 1, pp. 79–100, 2014.

[45] S. Kianpisheh and N. M. Charkari, "A grid workflow Quality-of-Service estimation based on resource availability prediction," *Journal of Supercomputing*, vol. 67, no. 2, pp. 496–527, 2014.

[46] S. Misra, S. Das, M. Khatua, and M. S. Obaidat, "QoS-guaranteed bandwidth shifting and redistribution in mobile cloud environment," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 181–193, 2014.

[47] J. M. Pedersen, M. T. Riaz, J. Celestino Jr., B. Dubalski, D. Ledzinski, and A. Patel, "Assessing measurements of QoS for global cloud computing services," in *Proceedings of the International Conference on Dependable, Autonomic and Secure Computing*, pp. 682–689, December 2011.

[48] P. Nadanam and R. Rajmohan, "QoS evaluation for web services in cloud computing," in *Proceedings of the 3rd International Conference on Computing Communication & Networking Technologies (ICCCNT '12)*, pp. 1–8, Coimbatore, India, July 2012.

[49] B. Zeng, J. Wei, and H. Liu, "Research of optimal task scheduling for distributed real-time embedded systems," in *Proceedings of the International Conference on Embedded Software and Systems (ICESS' 08)*, pp. 77–84, July 2008.

[50] S.-Y. Lee, D. Tang, T. Chen, and W. C.-C. Chu, "A QoS assurance middleware model for enterprise cloud computing," in *Proceedings of the IEEE 36th Annual Computer Software and Applications Conference Workshops (COMPSACW '12)*, pp. 322–327, IEEE, İzmir, Turkey, July 2012.

[51] K. J. Lin, S. Natarajan, and J. W. S. Liu, "Imprecise results: utilizing partial computation in real-time systems," in *Proceedings of the 8th IEEE Real-Time Systems Symposium*, San Francisco, Calif, USA, December 1987.

[52] J. W. S. Liu, W.-K. Shih, K.-J. Lin, R. Bettati, and J.-Y. Chung, "Imprecise computations," *Proceedings of the IEEE*, vol. 82, no. 1, pp. 83–94, 1994.

[53] W.-K. Shih and J. W. S. Liu, "Algorithms for scheduling imprecise computations with timing constraints to minimize maximum error," *IEEE Transactions on Computers*, vol. 44, no. 3, pp. 466–471, 1995.

[54] H. Aydin, R. Melhem, D. Mossé, and P. Mejía-Alvarez, "Optimal reward-based scheduling for periodic real-time tasks," *IEEE Transactions on Computers*, vol. 50, no. 2, pp. 111–130, 2001.

[55] H. Mora-Mora, J. Mora-Pascual, J. M. García-Chamizo, and A. Jimeno-Morenilla, "Real-time arithmetic unit," *Real-Time Systems*, vol. 34, no. 1, pp. 53–79, 2006.

[56] J. M. G. Chamizo, J. M. Pascual, H. M. Mora, and M. T. S. Pont, "Calculation methodology for flexible arithmetic processing," in *Proceedings of the International Conference on Very Large Scale Integration of System-on-Chip*, pp. 350–355, Darmstadt, Germany, December 2003.

[57] W.-C. Feng and J. W.-S. Liu, "Algorithms for scheduling real-time tasks with input error and end-to-end deadlines," *IEEE Transactions on Software Engineering*, vol. 23, no. 2, pp. 93–106, 1997.

[58] H. Yu, B. Veeravalli, and Y. Ha, "Dynamic scheduling of imprecise-computation tasks in maximizing QoS under energy constraints for embedded systems," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC '08)*, pp. 452–455, March 2008.

[59] A. Quagli, D. Fontanelli, L. Greco, L. Palopoli, and A. Bicchi, "Design of embedded controllers based on anytime computing," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 4, pp. 492–502, 2010.

[60] H. Mora-Mora, J. Mora-Pascual, M. T. Signes-Pont, and J. L. Sánchez Romero, "Mathematical model of stored logic based computation," *Mathematical and Computer Modelling*, vol. 52, no. 7-8, pp. 1243–1250, 2010.

[61] H. Mora-Mora, J. Mora-Pascual, J. L. Sánchez-Romero, and J. M. García-Chamizo, "Partial product reduction by using look-up tables for *MXN* multiplier," *Integration, the VLSI Journal*, vol. 41, no. 4, pp. 557–571, 2008.

[62] A. S. Tanenbaum, *Modern Operating Systems*, Pearson Education, Upper Saddle River, NJ, USA, 3rd edition, 2008.

[63] W.-T. Chan, T.-W. Lam, K.-S. Liu, and P. W. Wong, "New resource augmentation analysis of the total stretch of SRPT and SJF in multiprocessor scheduling," *Theoretical Computer Science*, vol. 359, no. 1–3, pp. 430–439, 2006.

[64] B. Andersson, G. Raravi, and K. Bletsas, "Assigning real-time tasks on heterogeneous multiprocessors with two unrelated types of processors," in *Proceedings of the 31st IEEE Real-Time Systems Symposium (RTSS '10)*, pp. 239–248, December 2010.

[65] S. Baruah, "Feasibility analysis of preemptive real-time systems upon heterogeneous multiprocessor platforms," in *Proceedings of the 25th IEEE International Real-Time Systems Symposium*, pp. 37–46, December 2004.

[66] Y.-S. Chen, H. C. Liao, and T.-H. Tsai, "Online real-time task scheduling in heterogeneous multicore system-on-a-chip," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 1, pp. 118–130, 2013.

[67] G. A. Elliott, B. C. Ward, and J. H. Anderson, "GPUSync: a framework for real-time GPU management," in *Proceedings of the IEEE 34th Real-Time Systems Symposium (RTSS '13)*, pp. 33–44, December 2013.

[68] J. A. Stankovic, M. Spuri, K. Ramamritham, and G. C. Buttazzo, *Deadline Scheduling for Real-Time Systems: EDF and Related Algorithms*, vol. 460 of *The Springer International Series in Engineering and Computer Science*, Kluwer Academic Publishers, 1998.

[69] X. Li and X. He, "The improved EDF scheduling algorithm for embedded real-time system in the uncertain environment," in *Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE '10)*, pp. V4563–V4566, IEEE, Chengdu, China, August 2010.

[70] B. Lee, H. Son, S. Yoon, and Y. Lee, "End-to-end flow monitoring with IPFIX," in *Managing Next Generation Networks and Services*, vol. 4773 of *Lecture Notes in Computer Science*, pp. 225–234, Springer, Berlin, Germany, 2007.

[71] A. Shriram, M. Murray, Y. Hyun et al., "Comparison of public end-to-end bandwidth estimation tools on high-speed links," in *Proceedings of the 6th International Conference on Passive and Active Network Measurement (PAM '05)*, vol. 3431, pp. 306–320, ACM, April 2005.

[72] S. Srivastava, S. Anmulwar, A. M. Sapkal, T. Batra, A. K. Gupta, and V. Kumar, "Comparative study of various traffic generator tools," in *Proceedings of the Recent Advances in Engineering and Computational Sciences (RAECS '14)*, pp. 1–6, March 2014.

[73] E. Yildirim, I. H. Suslu, and T. Kosar, "Which network measurement tool is right for you? A multidimensional comparison study," in *Proceedings of the 9th IEEE/ACM International Conference on Grid Computing (GRID '08)*, pp. 266–275, October 2008.

[74] Y. Liao, W. Du, P. Geurts, and G. Leduc, "Decentralized prediction of end-to-end network performance classes," in *Proceedings of the 7th ACM International Conference on Emerging Networking EXperiments and Technologies (CoNEXT '11)*, ACM, December 2011.

[75] D. Katramatos and S. J. Chapin, "A scalable method for predicting network performance in heterogeneous clusters," in *Proceedings of the 8th International Symposium on Parallel Architectures, Algorithms and Networks*, pp. 288–295, December 2005.

[76] G. Karagiannis, O. Altintas, E. Ekici et al., "Vehicular networking: a survey and tutorial on requirements, architectures, challenges, standards and solutions," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 4, pp. 584–616, 2011.

[77] W. J. Fleming, "New automotive sensors—a review," *IEEE Sensors Journal*, vol. 8, no. 11, pp. 1900–1921, 2008.