Threshold-based context analysis approach for ubiquitous systems

Nesrine Khabou, Ismael Bouassida Rodriguez

Threshold-based context analysis approach for ubiquitous systems

Nesrine Khabou and Ismael Bouassida Rodriguez
ReDCAD, University of Sfax, B.P. 1173, 3038 Sfax, Tunisia
Email: nesrine.khabou@redcad.org
Email: bouassida@redcad.org

***Abstract*—In ubiquitous computing systems, applications must be able to respond to dynamic context changes in order to provide suitable services to users. A promising solution consists of developing context aware applications which automatically change their behavior according to the user needs, the user preferences, the available resources and the surrounding environment. Furthermore, a context aware application is characterized by a closed feed back loop (MAPE) with four phases: Monitoring, Analysis, Planning and Execution. In this paper, we focus on the second phase of the MAPE loop. We elaborate a layered approach composed of a context capture layer, a context management layer and a context adaptation layer. Our approach enables systems to become pervasive in a transparent manner handling context monitoring and context analysis. We focus our work on the second layer (i.e context management). We propose first a context classification which takes into acount the context parameters evolution behavior. Then, we propose a context analysis approach for ubiquitous systems which aims at analyzing context information and detecting significant changes. The proposed approach uses a threshold based technique in order to detect context changes. When relevant context changes are detected, the context aware application will be notified to trigger its suitable process dynamically in order to deal with the changes.**

***Keywords*-Ubiquitous computing, context awareness, adaptation, analysis techniques, tendency, peak.**

I. Introduction

Nowadays, the tremendous development of wireless communication technologies and the widespread of devices such as laptops, sensors, RFID tags, etc, have led to the appearance of ubiquitous computing systems that takes place as the successor of mobile computing systems. The notion of pervasive/ubiquitous computing refers to user-centric provisionning of services and applications that are adaptive to user preferences and monitored conditions, namely the related context information, in order to consistently offer value-added and high level services [1]. This new paradigm brings new challenges to the traditional applications. In fact, ubiquitous computing is distinguished by heterogenous environments characterized by context changes. These changes are related for example to the availability of mobile devices resources, the devices joining or leaving the network, the network topology varying, the changing execution environment (temperature, pressure, noise, etc) and the application execution context (user location, device screen size). Hence, an application deployed on ubiquitous envrionments needs to be able to detect these changes in order to adapt its behavior to these varieties according to the corresponding context information. This ability to sense and to detect the environment changes is called "context awareness". Moreover, context awareness [2] is considered to be the key issue for making devices aware of the situation of their users and their environment.

As a result, the design and the implementation of context aware applications on top of ubiquitous environments is a challenging task. First, these applications need to manage context continuously, including the collection of a multitude of context information using various technologies such as sensors, operating systems, widget [3], etc.

Context information includes all aspects of the computing environments. e.g, the device characteristics such as the available memory, the available bandwidth, etc. [4] Second, a classification of context information into different categories should be performed in order to facilitate the context use. Then, the context aware applications analyze and interpret context information so that being able to detect the conditions under which adaptation actions are required. Finally, the context aware applications trigger adaptation actions when noticing context changes that can affect the application performance or functionalities.

It become obvious that there is a need to enable the development of ubiquitous computing environments offering best adapted services to users according to the changing context. This paper presents an approach that satisfy the afformentioned need.

In this paper, we propose a novel resource context classification taking into account the resource evolution behavior. Then, we introduce an approach that allows a application to detect the context changes, trigger notifications when necessary and adapt its functionalities to the current context. Our approach is based on thresholds in order to track down context changes. Moreover, thresholds are configured by domain experts or application designers and notifications are then raised by the corresponding entities in order to trigger appropriate reconfiguration actions.

The remainder of this paper is structured as follows: In section II, we introduce the context classification research studies in ubiquitous environments followed by some work treating methods used for context changes detection in such environments. Section III presents the case study called

"Entreprise hardware monitoring" that aims at monitoring resources state and raising appropriate notifications when resource violations are detected. In section IV, we introduce the proposed approach that allows for context classification and context changes analysis in ubiquitous systems using thresholds. In section V, we motivate and illustrate the feasibility of our approach through an illustrative scenario involving a remote hardware maintenance application. The last section concludes the paper and gives some directions for future work.

II. Related Work

Since we focus on context classification as well as context analysis, in the following, we give an overview of some studies dealing with context and context classification. Then, we discuss approaches proposed for context analysis (i.e context changes detection)

A. Context and context classification

Since the advent of context awareness, researchers may have different understandings of context. So far, there isn't a strict definition of context. For that reason, most studies try to define context through enumerating context examples.

1) Context definitions: Dey et al. [2] defined context as "any information that can be used to characterize the situation of an entity. An entity is a person, a place, or an object that is considered relevant to the interaction between a user and applications themselves".

As depicted in the Figure 1, Schilit et al. [5] believe that the most important aspects of context are defining by answering the questions: Where you are? whom you are with? and what resources are nearby? For that reason, they classify context into three parts: A computing context comprised of available processors, the communication costs, the network capacity and the network connectivity. A user context formed by the user's location, the collection of nearby people, the user's profile, etc. Finally, the physical context which englobes environmental parameters such as temperature, noise, humidity, pressure, etc.

In order to achieve a better understanding of context accross a time span, Chen et al. [6] add time context englobing the time of a day, week, month, season of the year, etc. In addition, emotional state, the orientation, the date and time of the day can define the context [2].

Other studies tailor the contextual information into two categories. On the one hand, as shown in Figure 1, Schmidt [7] partitions context into two classes such as physical environment including the surrounding resources of computation, and human factors which is formed by the mental state, the colocation of others, the collaborative tasks, etc. On the other hand, Mitchell [8] divides the context into two categories. A personal environment formed by the user's interest, the user's location and an environmental context formed by the weather forecast. Furthermore, Prekop et al. [9] consider that context information can be defined by considering two categories. Internal logical context such as the emotional state, the general goals, the business process, and external physical context as lighting, pressure, surrounding resources of computation, etc.

Dix et al. [10] proposed to divide context into four classes. A physical context formed by the nature of the device, the user's environment. An infrastructure context which includes the network connectivity, the communication bandwidth, etc, the system context composed of the application, the user a,d finally, a domain context formed by the application domain, the identification of the user, etc.

Due to the huge amount of context information, the aforementioned context classes are overlapping. Hence, some studies classify context information according to their application domains.

2) Context categorization: Context classification can discover the entire contextual information easily and simplify the context manipulation including context analysis.

Rassaque et al. [11] consider two large categorization viewpoints. A conceptual viewpoint which describes contextual space in terms of actors, actions and relationships between them. In fact, six context classes are identified: The user context, the physical context, the network context, the activity context, the device context and the service context. And a measurement viewpoint that contains continuous context (the value of context changes continuously), enumerative context (the values of context are a set of discrete values and defined in a list or set) and descriptive context, physical and virtual context. Other studies categorize context from the following point of view internal context or external context, material context or social context and real-time context or unreal context [12].

Some studies classify context based on the characteristics of the context. In fact, Soylu et al. [13] investigate some specific context characteristics. Moreover, they consider that context can be either static or dynamic. Static context means that context information remains constant within time. However, dynamic context means that context information keeps changing in different frequencies such as age, location, etc. Moreover, Soylu et al. [13] have categorized context from the application point of view into low level context and high level context. Low level context is usually sensed by sensors or collected by means of application logs. Contrary, high level context information can be derived from low level context information. Context can be also classified from the collection point of view into two classes: direct context that means also sensed context and indirect context (by means of inferring from direct context). Soylu et al. propose also a classification from the temporal point of view which contains static and dynamic context. They propose then their own classification of context. The authors categorize the context into eight categories. A user context formed by internal parameters such as the user's feelings and external

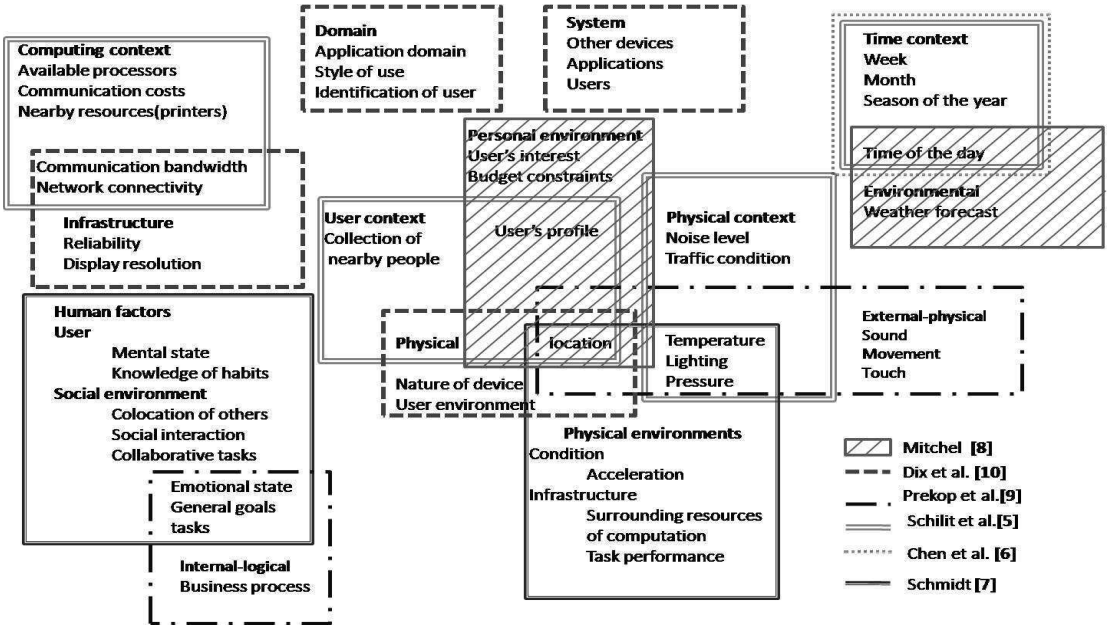


Figure 1. Context classification

paramaters like the user's name, his weight, his height, etc. Then, a device context which comprises its physical properties (CPU, memory, etc)(hard) and the available software in the device (soft), an application context, an information context, an environmental context, a time context, an historical context and finally a relational context. Han et al. [14], introduce a human centric classification of context. Indeed, they categorize context into three classes. A physical context which refers to real world, nearby user, making up of physical things, such as computer, print, fax. An internal context which refers to inside people such as feeling, thought, task, action and finally social context formed by the user's social surrounding (i.e, social relationship of the user).

Kim et al. [15] classify the context information into three types. Profiled, sensed and derived according to the context acquisition methods. Sensed context means that context information are accessed direclty from the surrounding environments via sensors. Derived context refers to context information that are derived from sensed context information through some transformation or interpretations. The profiled context means that context information are acquired using the context providers directly. For example, the users preferences can be obtained when they are explicitly communicated to the requesting application [12].

All the mentioned studies propose different classification of contextual information according to different points of view. However, the proposed classifications don't cover all the context parameters. In fact, they are restricted to their application domains.

B. Context changes detection techniques in ubiquitous environments

In the context changes detection research direction, several techniques and models are proposed in order to track context changes. Ciora et al. [16] propose a self adapting algorithm that can automatically detect the context changes and decide how the application should react once changes are detected. The self adapting algorithm is characterized by the MAPE loop with its four phases. In the analysis phase, the authors use the context entropy concept for detecting the context changes and determining the degree of fulfilling a predefined set of policies. The system context entropy measures the level of the system's self and execution environment disorder by evaluating the degree of fulfilling the policies. If the evaluated system context entropy exceeds a predefined threshold T_E, then all the policies are respected and adaptation is not needed. Otherwise, the system must trigger adaptation action in order to keep the system entropy below T_E. Despite the rich functionalities offered by the algorithm, it doesn't cover all context parameters. In fact, only temperature, humidity and light are considered.

Malatras et al. [1] propose a context aware framework that enables adaptation in pervasive computing environments. The adaptation is based on the monitored context information. The proposed framework is composed of three different layers. A device context management component. This component is responsible for collecting Data from the sensors

that are attached to a device and processing it into high-level context that can be later utilized to enable pervasive environments. Second, a user context management component which handles collectively the various user devices and performs relevant adaptation and coordination tasks. Finally, a space context management containing a user monitor and a space monitor used to monitor the context sources. They consider that adaptation in pervasive environments is guided by the enforcement of policies that are triggered by context information. Using the framework, context analysis is performed using thresholds. Indeed, adaptation is launched whenever context parameters exceed or fall below a certain threshold for a particular device. The proposed context aware framework allows an application to execute adaptation based on monitored context information. But the technique used to trigger adaptation is based on predefined thresholds which may lead to false detections. Moreover, only the network connectivity is considered as context parameter.

Birje et al. [17] propose a multiagent model to monitor the resource availability and control the device state. The mobile agent models aims to provide not only a resource monitoring scheme to keep track of all devices and their resource utilization at any instant but also a device state control such that avoiding an overloaded state. For that reason, they consider three resource states: overload/poor state, an underload/excellent state and a normal state. To decide the resource state, the authors define two thresholds. A minimum threshold T_{min} and a maximum threshold T_{max}. If the current resource utilization is less than T_{min}, then the resource is in an underload condition. If the current resource utilization exceeds T_{max}, then the resource is in an overload condition. The resource is in a normal condition if the current resource utilization is between T_{min} and T_{max}. The proposed multi agent model monitors and controls the resource utilization, the resource availability, the device mobility and the device state. In order to achieve this goal, different thresholds are used to detect changes. Although the proposed approach takes into account various parameters, the use of predefined thresholds can lead to false or missing detections.

In other studies, context changes are picked up by comparing a context value saved in a repository with a new context value. In fact, Zheng et al. [18] have addressed the issue of context change detection by proposing a context-aware middleware which conforms to the CORBA component model. The proposed middleware is composed of context aware services such as a context collector, a context interpreter, a context repository and a context analyzer. The latter is in charge of filtering and analyzing context information to determine relevant context changes and notifies the application afterwards. Furthermore, context filtering is based on a comparison of the context values saved in the context repository with the new context value in order to detect context changes. The proposed middleware enables to save the scare resources. In fact, the component deployment is performed "just-in-time". However, this middleware does not specify context information to take into account.

Another approach for dynamic context management is proposed by Taconet et al. [19]. The authors present CA3M, a context aware middleware, which enables applications to adapt their behavior by dynamically taking into account context changes.

They model the application by "entities", which represent a physical or logical phenomenon (person, concept, etc.) and "observable", which defines something to observe. For instance, a mobile device state is an example of an observable which may take a finite number of values (e.g low battery, almost low battery or normal battery). They consider that the change of an "observable" state or even the observation goes past a given threshold from the last notified value leads to a different application behavior.

Bouassida et al. [20] proposed a model driven approach for collaborative ubiquitous systems. In order to detect context changes, they specify predefined thresholds. Then, once context values remain below/under the threshold values, a notification is raised. Although this approach enables to detect instantly context changes, it may cause false detections as well as missing alarms by using fixed thresholds.

III. Case study: Entreprise Hardware Monitoring

We present in this section an example of an M2M application named "Entreprise Hardware Monitoring" (EHM) shown in Figure 2 that aims at controlling and monitoring the entreprise resources and acting accordingly when detecting indesirable situations.

In the EHM case study, we focus on some context parameters which we find relevent to our work. We consider the battery level, the available bandwidth, the available memory, the CPU load, etc.

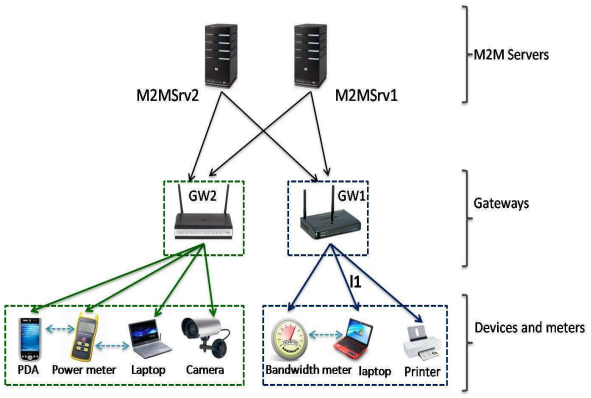


Figure 2. Entreprise hardware monitoring use case

The EHM application depicted in Figure 2 involves three kinds of participants: Controlling M2M servers, M2MSrv1 and M2MSrv2, gateways such as GW1 and GW2 conntected to the M2M servers via communication links, some meters(Power meter and bandwidth meter) used to monitor respectively the power consumption and the available bandwidth of the communication links and heterogeneous devices such as PDA, Camera, Laptop that communicate information to the controlling servers via the gateways.

The application components and their possible values are illustrated in Table 3

Application component	***Type***	***Possible values***
Temperature	Sensor	x Celsuis
Power	Meter	x Watt
Face recongnition	Camera	(Known, Unknown)
Bandwidth	Meter	x Bit/s

Figure 3. *The application components description*

The M2M servers implement analysis algorithms, and process monitored data received from meters in order to analyze them. Once results are obtained, the corresponding M2M servers notify the device and/or act to reconfigure the architecture by switching or disabling the affected device(s).

For example, the M2M servers send requests to the power meter connected to the gateway GW1 in order to monitor the available battery level of some devices. Another bandwidth meter is used to calculate the connection speed between the different devices. In fact, the bandwidth meter is able to perform the bandwidth measurement. On the other hand, at the operating system level, a probe is able to measure the state of hardware resources such as the available RAM. Thus, the corresponding device have to send a report to the appropriate M2M controlling server in order to notify it about its current state as well as its emergency degree.

IV. The proposed approach

The proposed approach aims at detecting context changes and raising notifications when context changes occur in order to adapt the application behavior accordingly.

Furthermore, in order to adapt an application to the changing context, the application should perform the following steps. It should collect context information. Context information includes computing context parameters such as available bandwidth, available memory, bettery level, etc. After being monitored using sensors, widgets, operating system, etc. contextual data are analyzed and finally adaptation actions are triggered when context changes are detected. Our approach depicted in Figure 4 is structured around this issue. We distiguish three layers of our proposed approach. These layers include: “Context Providing” responsible for providing context information. A second component, a “Context Management” is in charge of managing context information. The third layer illustrate the “Context Adapter” responsible for adapting the application based on monitored context information. We focus our work on the second layer (i.e. Context Management). In fact, the management of context information in ubiquitous environments can be decomposed into four stages: Context collection, context interpretation, context storage and context analysis.

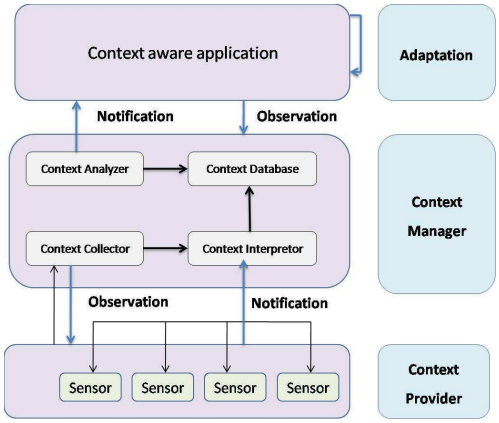


Figure 4. The proposed approach

Context acquisition from different sources (physical or logical sensors) is the first stage of context management. This task is performed by the “Context Collector”. Acquired context information may include information about the current weather condition (temperature, humidity, etc.), current user location, etc.

Context information captured by sensors is usually low-level and valueless. High level and meaningfull context information are generated using the context interpretor that can deduce useful high-level context information by combining a number of atomic, low-level context information. Once achieved, processed context information are stored in a “Context Database” which is used finally by the “Context Analyzer”.

The “Context Analyzer” is responsible for analyzing stored context information and detect context changes.

In this paper, we focus on context information analysis performed by the “Context Analyzer”. The “Context Analyzer” retrieves context information from the context database, analyzes it and tracks down the context changes using thresholds. Afterwards, the “Context Analyzer” notifies the application in order to execute the appropriate adaptation actions.

In our work, we consider computing context called resource context. Resource context constitutes the constraints imposed by the surrounding environment. For instance, the battery level, the available memory, the CPU load are examples of resource context.

A. Resource context classification

The diversity and the heterogeneity of devices and the network connectivity that ubiquitous environments provide, open the door to different resource context classification methods where context resources are combined in different ways to exploit.

A first classification divides resource context into two categories: Resource context related to the devices and resource context related to the network communication.

- ***Resource context related to the device***
  This category corresponds to the resources which characterize the device such as the available memory, the CPU frequency, the CPU load and the battery level.
- ***Resource context related to the network communication***
  This category deals with the resources that characterize the network communications as the network bandwidth, the network connectivity, the communication link load, the loss rate and the latency.

Despite the simplicity and the ease of this classification, it remains inappropriate to use. Furthermore, this context categorisation doesn't cover all the context parameter types. Moreover, this classification doesn't take into account the resource behavior which is necessary especially in ubiquitous environment that are characterized by an important evolution of the resource behavior and impredictable context changes. Second, we recall that our purpose is to define an approach that aims to detect context changes based on thresholds.

As a result, our idea consists on classifying resource context according to the resource context evolution behavior.

This classification divides the resource context into two categories: Resources whose behavior is characterized by a tendency and resources whose behavior is characterized by peaks. In the following, we present each family separately.

1) Resources with behavior characterized by a tendency: This family concerns resources whose model roughly coincides to a trendline as denoted in the Figure 5. In this class, we are interested to the resource tendency.

This class includes resources whose behavior evolves (increase, decrease) in a constant way within time.

The battery level mentioned in the case study detailed previously is an example of this category. For example, the battery level of a laptop device when not plugged, decreases within time as illustrated in Figure 5. The available memory, the latency belong also to this class resource context classification.

2) Resources with behavior characterized by peaks: In this category described by the Figure 6 the resource evolution behavior is characterized by abrupt changes called also peaks such as CPU load, link load, etc. The available bandwidth mentioned in the case study detailed before belongs to this family. For instance, as described in Section III, each M2M server recieves information via the gateways to analyze them. Hence, the CPU load of the M2M Servers can rise reaching high values causing an overload. When overloaded, the M2M server can delay some requests in its queue causing thereby an augmentation of the transmission latency.

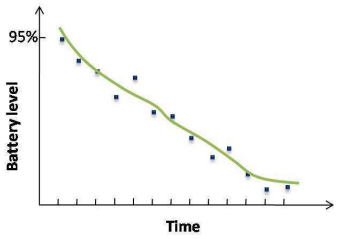


Figure 5. Battery level evolution within time

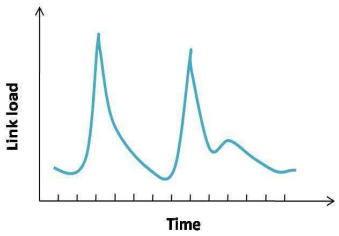


Figure 6. Link load evolution within time

B. Threshold calculation

We have proposed a resource context classification which takes into account the resource evolution behavior.

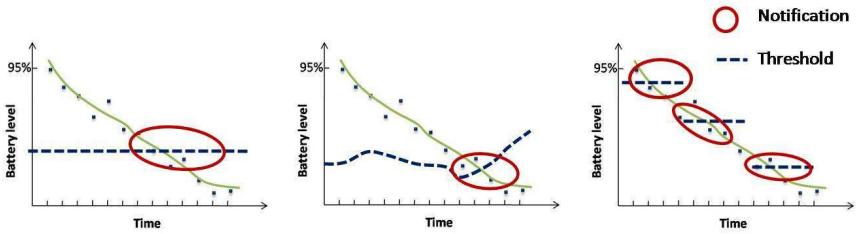
Since we base our approach on thresholds for detecting context changes, we present the threshold calculation for each resource category.

We propose to attribute for each resource parameter belonging to a category, n thresholds, then for each threshold, we assign a notification or a signal. This notification, defined by an expert, corresponds to an emergency degree or a need for adaptation that depends on the need of the expert or the application itself.

The thresholds can be either predefined or adaptive ones.

Afterwards, we give some elements about threshold calculation for each category detailed previously.

1) Threshold calculation for the resources whose evolution behavior is characterized by a tendency: For this category, we remind that the resource evolution behavior is described by a tendency. In order to avoid false detections as well as missing alarms, we need to define thresholds which are uncorrelated with the resource tendency evolution behavior. Thus, a notification is raised whenever the tendency curve crosses the threshold one.



(a) Fixed Threshold (b) Adaptive Threshold (c) Step Function Threshold

Figure 7. Threshold calculation for the resource whose evolution behavior is characterized by a tendency

So, different kinds of thresholds can be applied for this category, such as fixed thresholds, adaptive thresholds and step function thresholds as illustrated in Figure 7.

For instance, fixed thresholds may be defined by the application designer according to the resource characteristic. Then, a notification is raised once the resource crosses the threshold as depicted in the Figure 7(a).

For the adaptive threshold denoted in the Figure 7(b), mathematical methods can be applied in order to update threshold values at runtime such as the EWMA, Exponential Weighted Moving Average technique, [21] used by Lahyani et al. [22]. This techniques aims to update the threshold according to an observation window. However, for this kind of resource context characterized by a tendency, adaptive threshold must be uncorrelated with the resource evolution behavior in order to avoid false detections and missing alarms. Finally, for the step function threshold described in the Figure 7(c), thresholds are defined per period and notifications are raised when the resource behavior crosses the thresholds.

Let's consider the example of the battery level depicted in the section III. In this example, whenever the battery level values remain under the threshold, a notification is raised. In the Figure 7(c), we use the thresholds modeled as a step function. Hence, we raise different level depending on the battery state. For example, in Figure 7(a), Figure 7(b), when the battery level values decrease until reaching a critical value under the threshold, an alarm is forwarded to the M2M server in order to switch the corresponding device.

2) Threshold calculation for the resources whose evolution behavior is characterized by peaks: In the second category of resource context whose evolution behavior is characterized by abrupt changes, specifying adaptive thresholds that are correlated with the resource evolution behavior tendency is an appropriate method for setting thresholds. Indeed, in this family, peaks characterize sudden changes form a normal behavior to an abnormal one. For that reason, adaptive thresholds correlated with the resource evolution behavior remain under the resource shape. Furthermore, a violation of the threshold reveals a context change.

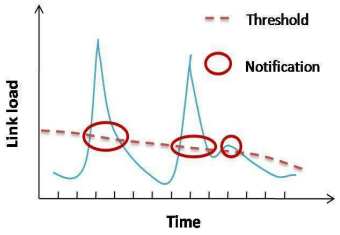


Figure 8. Threshold calculation for the resources whose evolution behavior is characterized by a peak

V. ILLUSTRATIVE SCENARIO

We illustrate the feasibility and the efficiency of our approach through a remote hardware maintenance application. The scenario highlight the ability of the application to adapt its behavior according to the context changes and taking into consideration the pervasive nature of ubiquitous environments such as resources constraints, devices heterogeneity, devices mobility, etc.

The scenario involves a computer campany owning different and heterogenous devices and sensors. These devices may be laptops, PCs, PDAs, cellphones, etc. Each device embeds an analysis entity that implements our approach which enables detecting context changes such as devices mobility, device overload, battery level changes, memory consumption, available bandwidth, etc. The main target of this campany is to ensure healthy state of its devices as well as the communication links by monitoring resource utilization at any instant of time. For that reason, the manager decides to control daily the devices state and the communication links. The employee selected to perform this task brings his mobile device (PDA, smart phone) which contains a notebook application, equipped with a logical sensor enabling it to capture some context parameters and connected to a power meter allowing for monitoring the battery level of its device as well as the other devices in the campany. In order to monitor the devices state (ON,OFF) and to check the network connectivity, the employee broadcasts periodic messages from its personal device through the

network. Once a timestamp is elapsed without receiving a signal or a response from the devices, then, the employee is notified of a bad connection causing lossy links or faulty devices or both. Afterwards, a bandwidth meter connected to the employee's personal device as depicted in the Figure 2 starts monitoring the communication links by analyzing the available bandwidth. So, at each moment, the message number transmitted through a link l1 for example is recorded. Then, by executing the analysis approach, the algorithm detects that the message number exceeds the corresponding threshold. For that reason, the employee decides that the link l1 is overloaded so that the bandwidth is in a critical state which engenders lossy links. Since the device state is affected by various parameters like the battery power, the device mobility, etc. the employee decides to monitor the battery level in order to assess his device state. Different thresholds are defined for this resource context. Since the battery level belongs to the first category detailed in section IV, three thresholds are defined. Each threshold corresponds to an emergency degree: "Normal Battery", "Almost Low Battery" and "Low Battery". When the power meter attached to his mobile device detects that the battery level reaches the low battery state, then the mobile device switches to a poor mode omitting pictures and reducing contrast. The above scenario justifies the context awareness of the application in order to cope with different resource constraints and to adapt its behavior accordingly.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have proposed an approach for the context changes detection in ubiquitous environments. Our approach is composed of two major components. A "Context Provider" and a "Context Manager". The "Context Provider" is out of the scope of this paper. In fact, we focus on the "Context Manager". The latter is divided into four components such as a "Context Collector", a "Context Interpreter", a "Context Database" and a "Context Analyzer". The "Context Analyzer" is the key component of our approach. Indeed, it is responsible for analyzing context information, detecting the context changes and forwarding notifications to the application when necessary. The "Context Analyzer" relies on thresholds in order to detect context changes. In our work, we have considered computing context to deal with as we have focused on resource context. Furthermore, we have presented a two classification of resource context according to the resource evolution behavior. A first category deals with resources whose behavior is modeled by a trendline. A second category concerns resources whose behavior is characterized by abrupt changes.

The proposed approach deals with the resource context. Although it allows detecting context changes, some issues are still challenging. Indeed, a detailed analysis of real applications should be performed in order to evaluate more the approach. Furthermore, the apporach should focus also on other context parameters (light, temperature, user's location, etc.) rather than the resource context parameters.

As future work, we plan to implement our analysis approach and to integrate it into the framework FACUS [23]. Then we intend to use ontologies in order to model context information.

ACKNOWLEDGMENT

This research is supported by the ITEA2's A2NETS (Autonomic Services in M2M Networks) project[1].

REFERENCES

[1] A. Malatras and B. Hirsbrunner, "A context-aware framework to enable adaptation in pervasive computing environments," in *Network-Based Information Systems, 2009. NBIS '09. International Conference on*, Aug 2009, pp. 182 –187.

[2] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, ser. HUC '99. London, UK: Springer-Verlag, Sept 1999, pp. 304–307.

[3] A. K. Dey, G. D. Abowd, and D. Salber, "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications," *Hum.-Comput. Interact.*, vol. 16, no. 2, pp. 97–166, Dec. 2001.

[4] A. K. Dey, "Understanding and using context," *Personal Ubiquitous Comput.*, vol. 5, no. 1, pp. 4–7, Jan. 2001.

[5] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *In Proceedings of the Workshop on Mobile Computing Systems and Applications*, 1994, pp. 85–90.

[6] P. J. Brown, J. D. Bovey, and X. Chen, "Context-aware applications: from the laboratory to the marketplace," *Personal Communications, IEEE [see also IEEE Wireless Communications]*, vol. 4, no. 5, pp. 58–64, 1997.

[7] A. Schmidt, "Ubiquitous computing - computing in context," Ph.D. dissertation, Computing Department, Lancaster University, England, U.K., 2002.

[8] K. Mitchell, "Supporting the development of mobile context-aware computing," Ph.D. dissertation, Lancaster University, 2002.

[9] P. Prekop and M. Burnett, "Activities, context and ubiquitous computing," *Comput. Commun.*, vol. 26, no. 11, pp. 1168–1176, Jul. 2003.

[10] A. Dix, T. Rodden, N. Davies, J. Trevor, A. Friday, and K. Palfreyman, "Exploiting space and location as a design framework for interactive mobile systems," *ACM Trans. Comput.-Hum. Interact.*, vol. 7, no. 3, pp. 285–321, Sep. 2000.

[1] https://a2nets.erve.vtt.fi/

[11] P. N. MA Razzaque, S Dobson, "Categorization and modelling of quality in context information," in *Proceedings of the IJCAI 2005 Workshop on AI and Autonomic Communications*, 2005.

[12] W. Liu, X. Li, and D. Huang, "A survey on context awareness," in *Computer Science and Service System (CSSS), 2011 International Conference on*, Jun 2011, pp. 144 –147.

[13] A. Soylu, P. De Causmaecker, and P. Desmet, "Context and adaptivity in context-aware pervasive computing environments," in *Ubiquitous, Autonomic and Trusted Computing, 2009. UIC-ATC '09. Symposia and Workshops on*, Jul 2009, pp. 94 –101.

[14] L. Han, S. Jyri, J. Ma, and K. Yu, "Research on context-aware mobile computing," in *Advanced Information Networking and Applications - Workshops, 2008. AINAW 2008. 22nd International Conference on*, Mar 2008, pp. 24 –30.

[15] E. Kim and J. Choi, "A context management system for supporting context-aware applications," in *Embedded and Ubiquitous Computing, 2008. EUC '08. IEEE/IFIP International Conference on*, vol. 2, Dec 2008, pp. 577 –582.

[16] T. Cioara, I. Anghel, I. Salomie, M. Dinsoreanu, G. Copil, and D. Moldovan, "A self-adapting algorithm for context aware systems," in *Roedunet International Conference (RoEduNet), 2010 9th*, Jun 2010, pp. 374 –379.

[17] M. Birje and S. Manvi, "Multiagent model for device state control in the wireless grid," in *Electronics Computer Technology (ICECT), 3rd International Conference on*, vol. 3, Apr 2011, pp. 456 –460.

[18] D. Zheng, J. Wang, W. Han, Y. Jia, and P. Zou, "Towards a context-aware middleware for deploying component-based applications in pervasive computing," in *Proceedings of the Fifth International Conference on Grid and Cooperative Computing*, ser. GCC '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 454–457.

[19] C. Taconet, Z. Kazi-Aoul, M. Zaier, and D. Conan, "Ca3m: A runtime model and a middleware for dynamic context management," in *Proceedings of the Confederated International Conferences, CoopIS, DOA, IS, and ODBASE 2009 on On the Move to Meaningful Internet Systems: Part I*, ser. OTM '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 513–530.

[20] I. Bouassida Rodriguez, G. Sancho, T. Villemur, S. Tazi, and K. Drira, "A model-driven adaptive approach for collaborative ubiquitous systems," in *Proceedings of the 3rd workshop on Agent-oriented software engineering challenges for ubiquitous and pervasive computing*, ser. AUPC 09. New York, NY, USA: ACM, 2009, pp. 15–20.

[21] J. M. Lucas, M. S. Saccucci, R. V. Baxley, Jr., W. H. Woodall, H. D. Maragh, F. W. Faltin, G. J. Hahn, W. T. Tucker, J. S. Hunter, J. F. MacGregor, and T. J. Harris, "Exponentially weighted moving average control schemes: properties and enhancements," *Technometrics*, vol. 32, pp. 1–29, Jan 1990.

[22] I. Lahyani, N. Khabou, and M. Jmaiel, "Qos monitoring and analysis approach for publish/subscribe systems deployed on manet," in *Parallel, Distributed, and Network-Based Processing, Euromicro Conference on*. Los Alamitos, CA, USA: IEEE Computer Society, Feb 2012, pp. 120–124.

[23] G.Sancho, "Adaptation d'architectures logicielles collaboratives dans les environnements ubiquitaires. contribution è l'interopérabilité par la sémantique," Ph.D. dissertation, Université de Toulouse, 2010.