A Threshold based Context Change Detection in Pervasive Environments: Application to a Smart Campus

Nesrine Khabou, Ismael Bouassida Rodriguez, Ghada Gharbi, Mohamed Jmaiel

Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 00 (2014) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014)

A Threshold Based Context Change Detection in Pervasive Environments: Application to a Smart Campus

Nesrine Khabou[a,*], Ismael Bouassida Rodriguez[b,c], Ghada Gharbi[b,c], Mohamed Jmaiel[a]

[a] *ReDCAD Research laboratory, Sfax University, Sfax, 3078, Tunisia*
[b] *CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France*
[c] *Univ de Toulouse, LAAS, F-31400 Toulouse, France*

Abstract

Context awareness is an essential feature of pervasive applications and runtime detection of context changes is necessary for enabling context awareness. Context aware applications are applications that are able to adapt their behavior (self adaptivity property) according to the available resources, environmental conditions, user needs, etc. The self adaptivity property is enforced by four phases. Monitoring/Collecting context parameters, analyzing the collected context parameters to detect changes, planning appropriate adaptation actions and finally executing the planned actions to respond to the changes. In this paper, we focus and discuss the second phase (Analysis phase). Contrary to the most research activities in which the analysis phase is performed by comparing context parameters against fixed thresholds, we propose an analysis approach that uses a threshold comparison technique. Fixed as well as adaptive thresholds are applied to detect changes and trigger notifications that are used to deal with the changes.

Selection and peer-review under responsibility of Elhadi M. Shakshuki.

Keywords: Ubiquitous computing; context awareness; adaptation; context analysis techniques

1. Introduction

The widespread of mobile devices (laptops, palmtops, smart phones) and sensors and the seamless connectivity between them have transcended the traditional computing era to the pervasive/ubiquitous computing. Pervasive/ ubiquitous computing aims at building smart environments where computing and communication are embedded in almost every surrounding object and can be accessed all the time and everywhere.

Context aware applications have thus emerged as one of the most important fields of pervasive/ubiquitous computing. These applications are those which can adapt their behavior in response to the context change without user intervention allowing to increase the application effectiveness by taking environmental context, user requirements into account[?].

* Corresponding author. Tel.: +216 251-530-13
E-mail address: nesrine.khabou@redcad.org

1877-0509
Selection and peer-review under responsibility of Elhadi M. Shakshuki.

The development of context aware applications is a complex process that requires the fulfilement of four phases. This process starts by collecting/monitoring context. This phase is considered a main phase of any context aware application. Collecting/monitoring context have a variety of sources such as sensors, networks, devices, users and other sources. The collected context parameters are analyzed in the second phase (Analysis) to detect the context changes. Several techniques such as threshold comparison, conditions or rules are used to detect changes. Once context changes are detected, adaptation actions are planned in the third phase (Planning). Finally, the planned actions are executed in the fourth phase (Execution) to react to the changes.

In this paper, we discuss the second phase (Analysis). This phase is divided into three consecutive steps: Context storage, context processing and classification, and finally, context change detection. The context storage is out of the scope of this paper. We first define the context classification step that attributes context categories for each context parameter. Second, we propose the context change detection step that targets detecting context changes and raising notifications when changes occur.

The rest of the paper is organized as follows. In section 2, we give an overview of some research studies dealing with context classification and context change detection in pervasive/ubiquitous environments. Then, in section 3, we introduce a case study named Smart Campus system. Its purpose is to provide suitable services to users according to the context changes. The analysis approach is presented in section 4. We define first the context classification step that takes into account the context parameter evolution. Then, we present the context change detection step that allows an application to analyze context and detect context changes using thresholds. In section 5, we motivate and illustrate the feasibility of our approach through an illustrative scenario. The last section concludes the paper and gives some directions for future work.

2. Related work

Many researchers have proposed several classifications of context. Some studies classify the context into two categories such as Schmidt[?], Prekop and Burnett[?] and Mitchell[?]. Schmidt[?], states two context categories such as physical environment and human factors. Mitchell[?] divides the context into two categories. A personal environment and an environmental context. Other studies classify context to several categories such as the studies of Rodden et al.[?] and Schilit et al.[?]. In order to achieve a better understanding of the concept across a time span, Brown et al.[?] add time context such as the time of the day, the week, the month and the season of the year.

Context classification is an important step to discover the possible context easily, simplify the context manipulation including context analysis. In fact, the context analysis's purpose is to analyze context parameters and identify context changes. In the context changes detection research direction, several techniques and models are proposed in order to detect the context changes.

In fact, Cioara et al.[?] propose to use the context entropy concept for detecting the context changes and determining the degree of fulfilling a predefined set of policies. Moreover, context situation entropy defines the level of the systems self and execution envionment disorder which is measured by evaluating the degree of respecting a set of policies. Hence, once the context entropy exceeds a fixed threshold, then the system is in a critical state and it must execute adaptation actions. Although this approach allows a self adaptation following context changes, it does not consider many context parameters to study as it is restricted to context parameters related to the environment such as temperature, humidity and light, etc. In other studies, context changes are detected by comparing a context parameter value saved in a repository with a new context parameter value. In fact, Zheng et al.[?] 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 stored 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. In the work of Baloch and Crespi,[?], they have proposed a framework for context acquisition, management and distribution in a ubiquitous environment. Context is collected from various sources and devices. Then, whenever a context change occurs, the device publishes its context in order to take the appropriate decision. However, the authors do

not specify how context changes can be detected using their framework. Another approach for dynamic context management is proposed by Taconet et al.[?]. Taconet et al. present CA3M, a context aware middleware, which enables applications to adapt their behavior by dynamically taking into account context changes. The authors model the application by entities, which represent a physical or a 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. However, using a given threshold can cause false detections. Other approaches are used to analyze context. In fact, Bouassida Rodriguez et al.[?] 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

3. Case study: Smart Campus

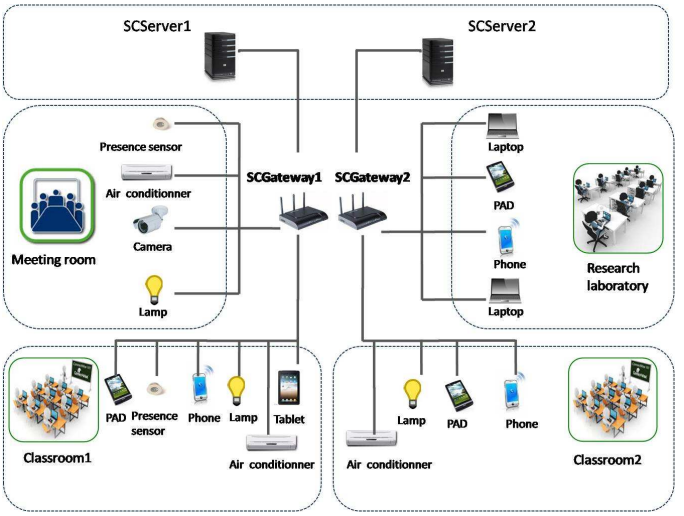


Fig. 1. Case study: Smart Campus

To illustrate the use of our approach, we describe in the following an example of a Smart Campus system illustrated in Fig. 1. Smart Campuses, with the ability to collect and analyze data, are built in order to benefit the institutions, the students, the teachers and the staff by providing services which facilitate interaction between the actors such as the teachers, the students and the staff. For instance, every actor is equipped with a personal device (PDA, smart phone, PC, tablet, etc) and the Smart Campus system provides different services to actors based on their current situations. To maintain the cooperation between students and teachers, the actors devices need to be aware of their environmental, execution context, etc. Consequently, Smart Campuses contain an infrastructure that allows devices and systems to be monitored and adapted automatically/autonomously according to the context. Our work focuses on several context parameters related to the environment and the user such as temperature, position and light which need to be monitored. Furthermore, a multitude of resources related to the devices and to the network such as the available memory, the energy, the CPU load and the available bandwidth should be considered and supervised to assess both the devices and the communication state.

An efficient Smart Campus relies on the cooperation of smart devices. The campus architecture depicted in the Fig. 1 involves different kinds of participants. On the one side, two controlling servers called Smart Campus Servers (SCServers) namely SCServer1 and SCServer2 which are Ethernet connected and equipped with important storage and high computational capabilities. On the other side, fixed (Presence sensors, cameras, lamps, air conditioners)

and mobile (laptop, PDA, phones) devices are placed. The mobile devices are usually resource constrained in terms of memory, bandwidth and energy for example. Hence, a periodic monitoring by the controlling servers is needed in order to check their state which changes according to context. Two gateways called Smart Campus Gateways (SCGateways), SCGateway1 and SCGateway2 implementing software interfaces are used to connect devices to the corresponding controlling servers (SCServers) to exchange relevant information. The campus space is composed of separate rooms (Research laboratory, Meeting room, Classroom1 and Classroom2). Each room is equipped with fixed devices as well as mobile devices that are carried by the different actors. Because of the complexity of the interaction between the different entities, we propose to focus our study on a part of the Smart Campus . It consists in the gateway SCGateway1 connected to both the SCServer1 and the devices located in the Classroom1.

4. The proposed approach

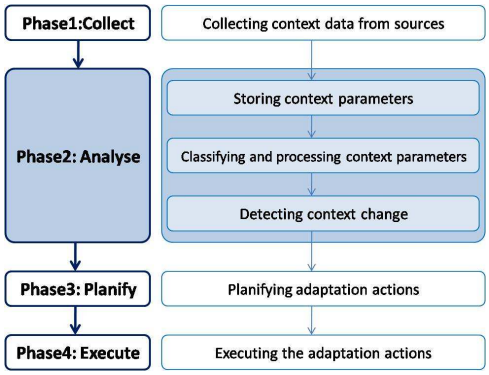


Fig. 2. An overview of the self adaptivity process of a context aware application

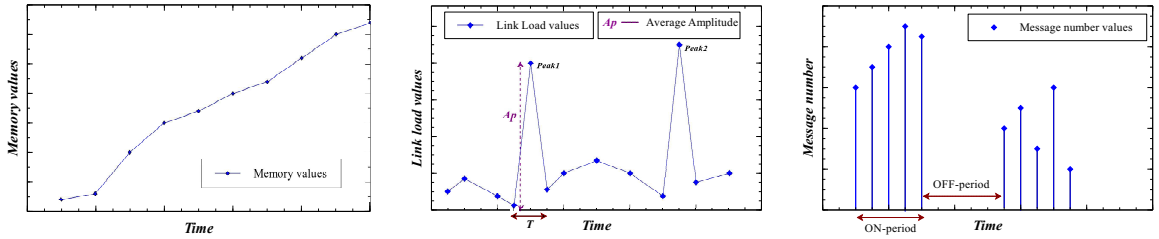
Towards detecting context changes, we propose an analysis process which aims at raising notifications if context changes occur. The analysis process is performed in three steps as highlighted in Fig. 2 (The filled rectangles).

1 Storing context parameters: This step aims at storing context parameters collected from the first phase (Collect). The context parameters are saved in log files. This step is out of the scope of this paper.
2 Classifying and processing context parameters: This step takes as input the retrieved context parameters from log files. It's aim is to attribute categories to each context parameter. The context classification step is detailed in Section 4.1.
3 Detecting context change: This step aims at analyzing the context parameters and detecting changes based on thresholds. This step is detailed in Section 4.2.

4.1. Context parameters classification step

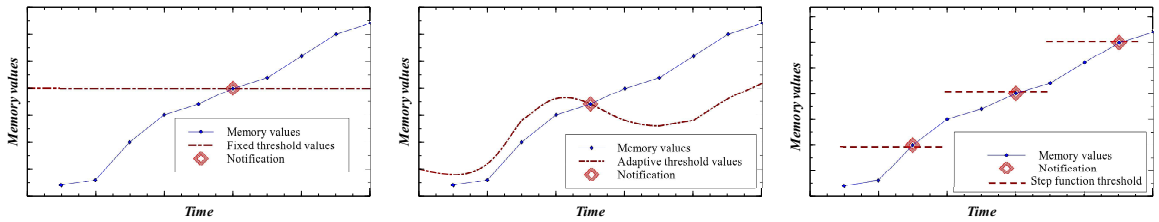
With a wide range of context parameters, context parameters should be classified into categories towards using context effectively. We propose a context classification that takes into account the evolution of the context parameters. Three categories are distinguished. Context parameters which evolve in a trend way, context parameters which are characterized by peaks and context parameters which are characterized bu bursts. The three context categories are depicted in Fig. 3.

1 The term trend refers to a stable tendency of growth or decline exhibited in the data as illustrated in Fig. 3(a). For example, each mobile device such as PDA, mobile phone, etc. mentioned in section 3 periodically monitors its resource state (Battery level, memory consumption) that belong to the trend category.
2 Peaks are defined as high values with sharp rise followed quickly by sharp fall implying a narrow period width[?] . We define a peak P by its amplitude "A_p" and its period T as depicted in Fig. 3(b). In fact, the peak is a very



(a) Trend category (b) Peak category (c) Burst category

Fig. 3. The proposed context categories



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

Fig. 4. An example of threshold calculation for a context parameters whose evolution is characterized by a trend

narrow period of high values- That is, its amplitude "A_p" exceeds for "n" times the average amplitude"A_{av}" of the time series formed by the context parameter values. For example, the link load context parameter illustrated in Fig. 3(b) belongs to the peak category.So, a peak is defined by the following formulas:

$$\begin{cases} A_p = n \times A_{av} \\ T \leq m \times TimeUnit \end{cases}$$

Where A_p defines the peak's amplitude, A_{av} defines the average amplitude of the parameter curve, n and m are constants fixed by the application designers.

3 A burst consists on a relatively wide contiguous region of values. A burst is defined as a large number of occurring events Klan et al.[?] . As depicted in the Fig. 3(c), we can model the wide region by an ON-period and the other by an OFF-behavior Yang[?] . The ON-period models a single flow such as the transfer of a single web page, and the OFF-period models the users thinking time. Noting that the ON-period and the OFF-period are strictly alternating. The message number is modeled as a burst as depicted in Fig. 3(c).

4.2. Context change detection step

Since the context parameters classification step is based on the context parameters evolution, we detail in the following the threshold calculation for each context category defined previously. For each context parameter belonging to a category, we propose to assign a number of thresholds. Each threshold is characterized by a level and for each level corresponds a notification type. When context parameters curve cuts the threshold, a notification is raised (See Fig. 4).

4.2.1. Threshold calculation for trend based context parameters evolution

For this category, the context parameters evolution is described by a trend. In order to avoid false detections as well as missing alarms, we need to define thresholds which are uncorrelated with the context parameter evolution. The notification raised when the context parameter value crosses the threshold is illustrated in Fig. 4. Different kinds of

thresholds can be applied for this category, such as fixed thresholds, adaptive thresholds and step function thresholds. For instance, fixed thresholds may be defined by the application designers according to the context parameter characteristics. Then, a notification is raised once the context parameter value crosses the fixed threshold as depicted in the Fig. 4(a). For the adaptive threshold denoted in the Fig. 4(b), mathematical models need to be applied in order to update threshold values at runtime such as the Exponential Weighted Moving Average technique [?] used by Lahyani et al.[?] . However, for this kind of context parameters characterized by a trend, adaptive threshold must be uncorrelated with the parameter evolution in order to avoid false detections and missing alarms. Finally, for the step function threshold described in the Fig. 4(c), thresholds are defined per period and notifications are raised when the context parameter values crosse the thresholds. In our case study: Smart Campus, the memory consumption can be modeled by a trend. Each mobile device needs to monitor and analyze the memory consumption to evaluate its state. So each mobile device compares periodically its memory consumption state with the threshold. A notification is raised when there is an intersection between the threshold and the memory values as illustrated in Fig. 4.

4.2.2. Threshold calculation for peak based context parameters evolution

Identifying and detecting peaks in a given time series is important in order to react accordingly and to avoid undesirable effects. In this context category, the idea consists of specifying adaptive thresholds. In fact, using fixed thresholds in this category could cause false alarms. As mentioned in the Smart Campus case study (section 3),

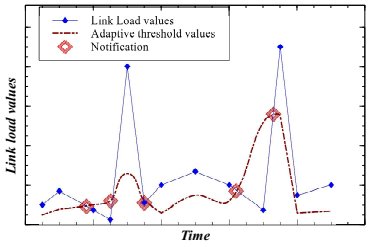


Fig. 5. Threshold calculation for a context parameter whose evolution is characterized by a peak

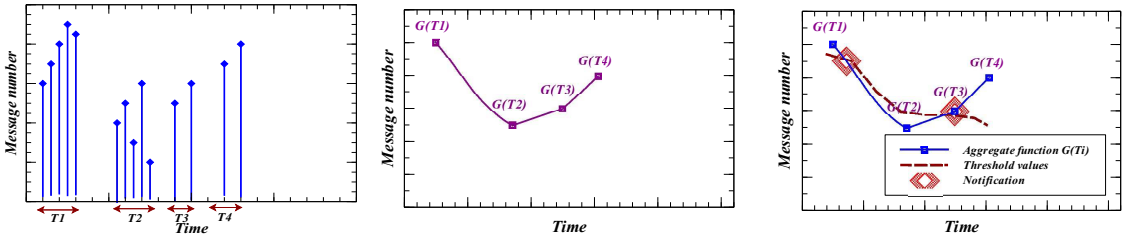
the servers receive periodically contextual information from the devices through the gateways. In case of a sudden large amount of data, the servers could be overloaded and need to perform an important processing. This overload is identified when the CPU load oversteps the adaptive threshold for this context category as illustrated in Fig.5.

4.2.3. Threshold calculation for burst based context parameter evolution

In this category, our idea consists on transforming a bursty model into a peaked or a trend model. So, we propose to apply an aggregate function G in each ON-period. The aggregate function application is illustrated in Fig. 6(b). As depicted in the Fig. 6(b), the obtained model roughly coincides with a trend function. After applying thresholds, a notification is raised when the context parameter values cross the threshold calculated in an ON-period. For the Smart Campus case study (section 3), the gateway SCGateway1 analyzes the received data from the devices. The maximum queue size of the gateway is set by the application designer. So, if the traffic received by this gateway in a period T_i exceeds a maximum, then a burst is identified in T_i. Towards detecting bursty periods, the function G is applied. In our case study, the function G calculates the slope of the scatter diagram obtained in each ON-period. And, the intensity of each slope formed in each ON-period is computed (Result shown in Fig. 6(b)). Consequently, if the slope intensity exceeds the threshold, then a burst is detected and appropriate adaptation actions are launched as shown in Fig. 6(c).

5. Illustrative scenario

In this section, we consider the case of a Smart Campus as shown in section 3. We focus in the interaction of the actors of the Classroom1 and SCServer1 through SCGateway1. This Classroom1 is used by the researchers, the teachers, the students. The context parameters considered in this scenario are the following:



(a) Burst model (b) Aggregation function application (c) Threshold calculation

Fig. 6. An example of threshold application for burst based context parameters evolution

- The actors' position: It belongs to the peak category. Adaptive thresholds are applied to this context parameter.
- The CPU load: It belongs also to the peak category. Adaptive thresholds are applied to this context parameter.
- The memory consumption: It belongs to the trend category. Fixed, adaptive or step function thresholds are applied to this context parameter.

The scenario starts as follows: At the beginning of each course session, the presence sensor captures and localizes the mobile actors. Their positions are then forwarded to SCServer1. SCServer1 runs the analysis algorithm based on thresholds on the context parameter (mobile actors position) and takes the appropriate desicion. Since the SCGateway1 communicates information received from the devices to the corresponding SCServer, in case of a large amount of data received by the gateway SCGateway1 at T_i , the load of SCGateway1 increases rapidly reaching a high value (i.e a peak) and exceeding the adaptive threshold specified for this context parameter. The analysis algorithm integrated in SCGateway1 raises a notification and the gateway decides to split its load with its neighboring gateway namely SCGateway2.

Each student participating to a course uses a tablet device to which the course will be dispatched through bluetooth. The tablet device holds the analysis algorithm in order to detect context changes. During the course, the students tablet display appropriate slides and follow their courses. Furthermore, the students can write annotations on their tablets and publish their comments to share knowledge between all the group members to enrich the course and enhance the cooperation. A student holding a tablet is participating to the course by exchanging information and slides. For each amount of data received, the tablet device retrieves the memory consumption from the operating system using probes. Then, it compares each value with the threshold which can be either fixed, adaptive or step function since memory consumption belongs to the trend context parameter category as illustrated in Fig. 7. The tablet detects that at an instant T_i, the memory consumption exceeds the threshold (Notification2). So, the device closes unecessary applications (Specified by the user). If this action is not enough (Notification3), the device displays a message proposing to the user to close some other running applications . This scenario highlights the interaction of the campus application with

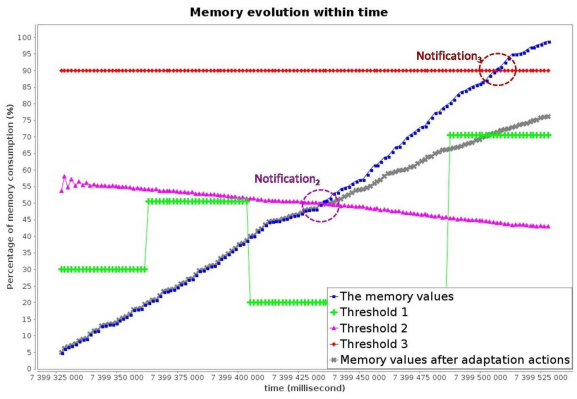


Fig. 7. An example of threshold application on the memory context parameter

context in such a way that it can detect context changes using the analysis approach in order to adapt its behavior accordingly.

6. Conclusion

The challenges for context aware applications design and implementation are to handle monitoring/collection, analysis, planning and acting. In this paper, we have discussed the second phase (Analysis). The analysis phase is divided into three steps. We have defined first a context classification step based on context parameters evolution. Three categories have been identified. A trend category, a peak category and a burst one. Then, we have presented a context change detection step which aims at analyzing context and identifying context changes. In order to identify changes, we propose a threshold comparison technique. Different thresholds are used to detect changes such as fixed, adaptive and step function. As future work, we plan to stretch the context classification step. For further development, we intend to implement our analysis approach and to use it by generic applications.

Acknowledgments

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

References

. Luo, H., Ci, S., Wu, D., Tang, H.. Adaptive wireless multimedia communications with context-awareness using ontology-based models. In: *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*. 2010, p. 1–5.
. Schmidt, A.. *Ubiquitous Computing - Computing in Context*. Ph.D. thesis; Computing Department, Lancaster University,England, U.K.; 2002.
. Prekop, P., Burnett, M.. Activities, context and ubiquitous computing. *Comput Commun* 2003;**26**(11):1168–1176.
. Mitchell, K.. *Supporting the Development of Mobile Context-Aware Computing*. Ph.D. thesis; Lancaster University; 2002.
. Rodden, T., Chervest, K., Davies, N., Dix, A.. Exploiting context in hci design for mobile systems. In: *in Workshop on Human Computer Interaction with Mobile Devices*. 1998, .
. Schilit, B., Adams, N., Want, R.. Context-aware computing applications. In: *In Proceedings of the Workshop on Mobile Computing Systems and Applications*. 1994, p. 85–90.
. Brown, P.J., Bovey, J.D., Chen, X.. Context-aware applications: from the laboratory to the marketplace. *Personal Communications, IEEE [see also IEEE Wireless Communications]* 1997;**4**(5):58–64.
. Cioara, T., Anghel, I., Salomie, I., Dinsoreanu, M., Copil, G., Moldovan, D.. A self-adapting algorithm for context aware systems. In: *Roedunet International Conference (RoEduNet), 2010 9th*. 2010, p. 374 –379.
. Zheng, D., Wang, J., Han, W., Jia, Y., Zou, P.. 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*; GCC '06. Washington, DC, USA: IEEE Computer Society. ISBN 0-7695-2694-2; 2006, p. 454–457.
. Baloch, R.A., Crespi, N.. Profile context management in ubiquitous computing. *Int J Ad Hoc Ubiquitous Comput* 2012;**11**(4):237–245.
. Taconet, C., Kazi-Aoul, Z., Zaier, M., Conan, D.. 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*; OTM '09. Berlin, Heidelberg: Springer-Verlag. ISBN 978-3-642-05147-0; 2009, p. 513–530.
. Bouassida Rodriguez, I., Sancho, G., Villemur, T., Tazi, S., Drira, K.. 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*; AUPC 09. New York, NY, USA: ACM. ISBN 978-1-60558-647-2; 2009, p. 15–20.
. Palshikar, G.K.. Simple Algorithms for Peak Detection in Time-Series. In: *Proc. 1st Int. Conf. Advanced Data Analysis, Business Analytics and Intelligence*. 2009, .
. Klan, D., Karnstedt, M., Politz, C., Sattler, K.. Towards burst detection for non-stationary stream data. In: *KDML*. 2008, p. 57–60. URL: `http://www.dbis.prakinf.tu-ilmenau.de/publications/files/DBIS:KlaKarPoe08.pdf`.
. Yang, X.. Designing traffic profiles for bursty internet traffic. In: *In Proceedings of IEEE Global Internet*. 2002, .
. Lucas, J.M., Saccucci, M.S., Baxley Jr., R.V., Woodall, W.H., Maragh, H.D., Faltin, F.W., et al. Exponentially weighted moving average control schemes: properties and enhancements. *Technometrics* 1990;**32**(1):1–29.
. Lahyani, I., Khabou, N., Jmaiel, M.. 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; 2012, p. 120–124.

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