HAL open science

QoS Based Desing Process for Pervasive Computing Applications

Christine Louberry, Philippe Roose, Marc Dalmau

To cite this version:

Christine Louberry, Philippe Roose, Marc Dalmau. QoS Based Desing Process for Pervasive Computing Applications. ACM Mobility, Sep 2009, Nice, France. pp.7. hal-00416015v2

HAL Id: hal-00416015

https://hal.science/hal-00416015v2

Submitted on 30 Nov 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

QoS-Based Design Method for Constraint Device Based Applications

Christine Louberry
LIUPPA
IUT de Bayonne
64600 Anglet, France
+33559574326
Christine.louberry@univ-pau.fr

Philippe Roose
LIUPPA
IUT de Bayonne
64600 Anglet, France
+33559574348
Philippe.roose@univ-pau.fr

Marc Dalmau
LIUPPA
IUT de Bayonne
64600 Anglet, France
+33559574321
Marc.dalmau@univ-pau.fr

ABSTRACT
We are currently facing with an important evolution in communication area. People use more and more some mobile and tiny devices to interact with their favourite services. However, these devices have resource constraints which are necessary to be aware in order to offer effective applications in terms of quality and lifetime. In this article we propose a design method allowing to manage constraint and particularities of such devices to react to context evolution with reconfiguration reflecting quality and sustainability of the deployed solution.

Categories and Subject Descriptors
D.2.9 [**Management**]: Software configuration management

D.2.10 [**Design**]: Methodologies

General Terms
Design.

Keywords
Design method, QoS, sensor network, software components, supervision platform.

1. INTRODUCTION
We are currently facing with an important evolution of usages in the communication domain. The recent technological overhangs have allowed the democratization of wireless networks and the mass-marketing of devices allowing users to communicate such as mobile phones, Smart phones, PDA and mini PC. These devices present the advantage to be tiny, mobile and to be equipped of one or several wireless communication medium. They embed a software part allowing receiving, computing and transmitting information. So-equipped, these peripherals contribute to the ubiquity of services in the sense where users can access to their favorite services from anywhere and any support.

Characteristics of mobile devices present both advantages and disadvantages. Advances in microelectronics domain allowed to strongly increase the abilities of such devices. However hardware resources such as CPU, memory, data rate and battery as well as software resources are still very limited. Also the mobile characteristic is a source of constraint. Mobility may imply some connection losses and disrupt the functioning of the services provided. To resume, these characteristics evolve during time and affect the QoS of the service provided [1].

We are particularly interested in QoS management of distributed applications faced with hardware limits, users' needs and current circumstances of use. Although the characteristics offered by such mobile devices improve the QoS, their integration into traditional systems implies to take up the challenge of the durability of the application and the continuity of services faced with connectivity problems due to mobility, weak data rate and limited battery.

In this article, we are interested in the adaptation to the context by QoS management of the applications. Indeed, when context evolves, the QoS evolves too and thus the application must be adapted in order to provide to the user an efficient service. We propose to resolve the problems raised by the changes of context by dynamic reconfiguration of the applications. For that purpose, we defined a supervision platform for distributed context sensitive applications where the context is the main source of information for the QoS evaluation [11]. However, in order to provide to the platform all the necessary information for configurations' evaluation, we propose a design method of the architecture of such applications allowing the integration of QoS information.

The section 2 presents the definition of the quality of service that we use for these works. In the section 3, we present an example of application to illustrate the design method we propose. The section 4 details the steps of this method since the unrefined decomposition of the application up to the detail of every possible configuration. The section 5 lists the events leading to a reconfiguration of the application and the corresponding actions. The section 6 presents elements driving the choice of the configuration to be implemented. The section 7 concludes and presents the perspectives of these works.

2. QoS DEFINITION
There is no consensus about QoS definition. None definition is convenient for all the domains. Nevertheless the standard X.902, defined by International Telecommunication Union [6], describes the quality of service as a set of requirements concerning the collective behaviour of one or several objects.

Usually used in network to mesure the performance of transmissions according to quantitative criteria such as delay, gigue, rate of error, etc, the most common definition found in literature says that it is not possible nowadays to consider QoS only as a network and hardware criterion [14] [12] [2].The

consideration of the user's point of view is necessary but not sufficient for the QoS evaluation when dealing with constraint devices. Indeed the works actually done in the QoS sensor networks domain show that this one is estimated in terms of precision of the collected data and the lifetime of the network. Moreover, the precision of the data is a function of the level of energy of nodes [7]. To obtain relevant and effective results, it is of interest to optimize the energy consumption and to protect at most the lifetime of the nodes.

In previous works [9], we defined the quality of service as "the adequacy between the service wished by the user and the service which is provided to him". This definition allows taking into account the classic definition of network QoS but also the functional aspects like the ergonomics and the customization of the service. However it is not sufficient when we use constrained devices, so we wish to act at the three levels, represented in the figure 1.

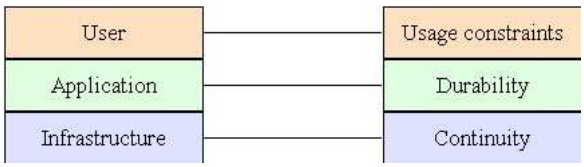


Figure 1. Qos types and levels

At the infrastructure level, we have to guarantee the continuity of the service, whatever are the evolutions of the infrastructure and in spite of hardware or network failure. At the application level, we have to guarantee the durability of the application. Indeed, the use of energy-autonomous peripherals raises the problem of their lifetime. Finally, at the user level, we have to guarantee the respect for the constraints of use of the application.

Continuity of service. Considering application QoS, the main objective is to guarantee the continuity of the service in spite of the hardware, software and network failures. Indeed, the use of wireless networks implies disturbances such as rate getting lower, data unavailability, etc. Furthermore we have to face the problems of heterogeneousness due to the use of several kinds of devices, as well as the problems of hardware fails like the level of the battery of mobile devices.

Durability of application. We wish to guarantee the continuity of the service devices on which are executed the applications have limited resources. One way to guarantee the continuity of service is to maximize the lifetime of the application because a device which does not have energy anymore causes the disconnection of all services it provides and consequently may compromise the continuity of service. Works on wireless sensor radio showed that data transmission on the network consumes between ten and hundred times more energy than computation. Solutions based on service mobility may decrease transmissions and therefore increase lifetime.

Usage constraints. We defined the usage constraints as the functional specifications of the application. There are constraints on the functioning which the system has to respect. For example, in a remote surveillance application, the designer can express constraints as follows:

- when the luminosity is lower than the required minimal threshold, activate the infrared function of cameras.
- when a movement is detected, activate cameras and microphones.

3. AN EXAMPLE OF AN APPLICATION IN A CONSTRAINT ENVIRONMENT: VISIT OF MUSEUM

Throughout this article, we illustrate our method with an example of an application intended for the tourism: the visit of a museum.

The application is used by three visitors, each of them having a mobile device (PDA or smartphone for example). The museum provides a server proposing a video information service (fig. 2). We consider that the best QoS is to broadcast a color video.

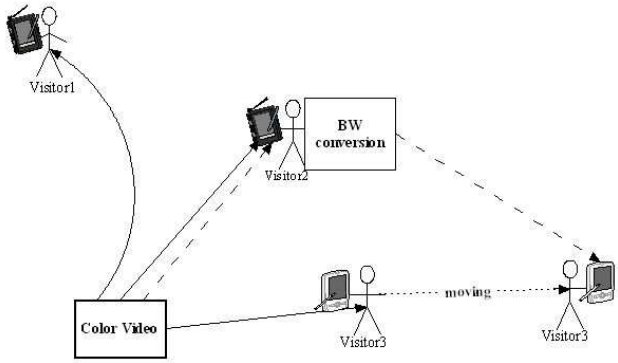


Figure 2. Visit of a museum application

The following paragraph details possible scenarios for the use of this application. The three visitors are getting the information video service. The visitor #3 moves in the museum. The platform is advised of this movement and consequently estimates the quality of the video service provided to the user #3. Three scenarios appear. First, although it moves, the device of the visitor #3 always can reach the video server and the rate still allows transmitting a color video. Secondly, the rate is weak but the device can always reach the server. To ensure the continuity of service, a solution is to reduce the number of data transmitted. The server does not broadcast a color video anymore, but a black and white one. Thirdly, the device cannot reach the server. The platform has to look for a new route to reach the device #3 and assure the continuity of the service. For example, the device of the visitor #2, which also uses the video service, can be a relay for the device #3.

Discovering a new route to transmit information is a classic problem, many protocols as AODV [5] can be implemented. However, we use a supervision platform distributed on each host permitting to have a complete and global knowledge of the application. It allows us proposing more interesting solutions not only based on technical criteria of feasibility. Indeed because the color video is already broadcasted to the users #1 and #2, it is possible to use one of these two users as a relay for the user #3 which cannot reach the video service anymore. However if the relay device does not have enough energy available, the brodcasting of a color video is not possible. We can thus propose to install a color/black and white conversion component on the

relay device in order to transmit a black and white video which is more compatible with the energy available. So, to ensure the continuity of service, before sending the video to the device #3, the device #2 performs a conversion into black and white. The energy consumption depends of the quantity of data transmitted. The conversion of the video into black and white allows to reduce the quantity of data to transmit to the device #3 and thus to preserve its energy as well as that of the transmitter. Furthermore distributing a part of the services allows balancing of responsibilities in the network. Obviously such a solution requires a good knowledge of the whole application in terms of services. Our approach by reconfiguration allows proposing reliable solutions from both points of view of the infrastructure and the QoS.

The following paragraphs detail the design of these solutions and the decision of reconfiguration.

4. DESIGN METHOD FOR APPLICATIONS IN CONSTRAINT ENVIRONMENT

This paragraph presents the various steps of the design method we propose. This method aims at bringing the necessary information to the reconfiguration platform so that it manages in best possible way the quality and the lifetime of the application which it supervises.

4.1 Identification of services

As we presented it in the section 2, our definition of the QoS is not only user centered. We are particularly interested in functional aspects and not only in the QoS perceived by the user. The method we used in [9] was centered on the categories of user and was too specific. Therefore, we decide to decompose the application into services. Each service can be implemented by one or several assemblies of components, dynamically changing during time according to the required QoS. These components are interconnected by connectors and data streams (fig. 3).

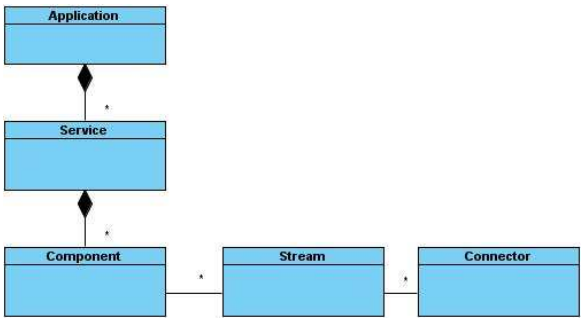


Figure 3. Application composition

Each assembly provide the same service but with various QoS. The role of the platform is to choose, for each provided service, the assembly offering the best quality compatible with the QoS criteria expressed in 2. The deployment of components onto the infrastructure is also done by the platform. The platform can dynamically adapt the application according to the context's evolutions because it can choose both the composition of a service and the deployment of a service. As we said in the section 3, the minimization of the energy consumption is essentially achieved by minimizing transmissions. That is why solutions requiring many components can be more economic than simpler ones. So, in our previous example, the broadcasting of a color video towards the user #3 could perfectly operate using the users 1 and 2 as relay but asking them to transmit only half of the video (for example one image or one block of images out of two). In order to have such an available solution, during design, the video service should have been proposed to be realized by such an assembly and components of partial broadcasting and shares reception should have been available.

The first step of the design method we propose consists in identifying all the services constituting the application. In our example, the visit museum application provides services of video, audio and text.

4.2 Configurations

Once services are identified, we decompose them and determine each configuration.

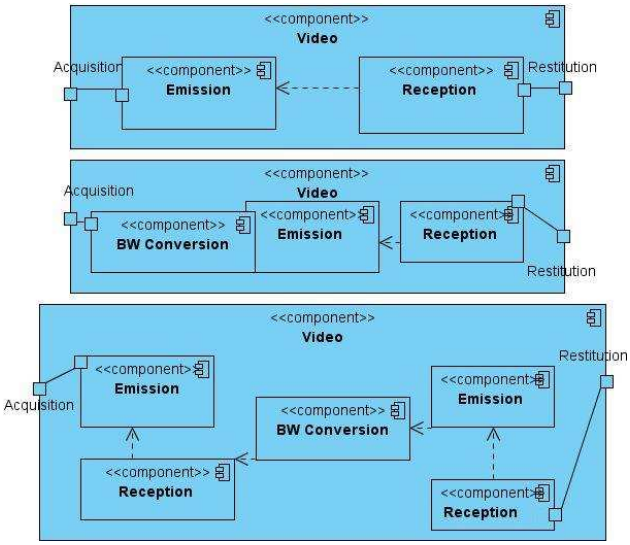


Figure 4. Possible assemblies realizing the video service

In the case of the video service we list three possible assemblies. The first one is made of two components: a component of color video emission and a component of color video reception. The second is also made of two black and white video components: one for emission and one for reception. Finally the third assembly is made of these last four components and a black and white conversion one (fig. 4). Naturally, the compositions listed here are not exhaustive. The number of configurations proposed by the designer will depend on the type of applications and on the context in which they will be used. In our example, we can satisfy QoS with only few assemblies because we can suppose that there will be numerous visitors in the museum and that it will always be possible to use them as relays to reach services. It would not be the same in a situation where the distances would be more important and bigger the risk of having an isolated user would be bigger.

Moreover, we propose to classify these configurations in order to drive the platform during the choice of the most appropriate configuration. The designer has to classify the configurations by QoS level. The assembly in first position is the assembly providing the result of upper quality and so on. This classification determines the order in which the configurations will be evaluated by the platform when a reconfiguration event is raised. Such a classification allows improving the QoS. Indeed, when a reconfiguration event is raised, the platform first evaluates the configuration of upper level to try to improve the QoS, and if it does not suit, it then evaluates the configurations less classified. Furthermore this classification allows reducing strongly the number of solutions to evaluate indeed some works showed that the research of the optimal assembly is a NP-complete problem [15] [8] [3].

In the example of the video service, the assembly situated in first position is the one providing a color video. Then comes the assembly providing a black and white video. The one using direct broadcast and the one with relays are equivalent. However to use a relay allows to balance the responsibility of the network and when a device is very far from the server, using a relay will consume fewer resources than maintaining a rather powerful signal to receive the data.

5. RECONFIGURATION EVENTS

Our applications are supervised by a platform which role is evaluate the provided QoS and reconfigure the application. Choices of reconfigurations are based on three levels: 1) continuity, 2) durability and 3) constraints services (section 2).

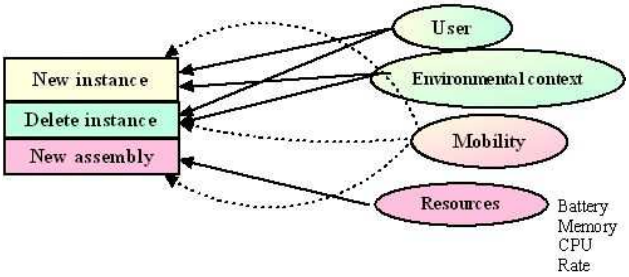


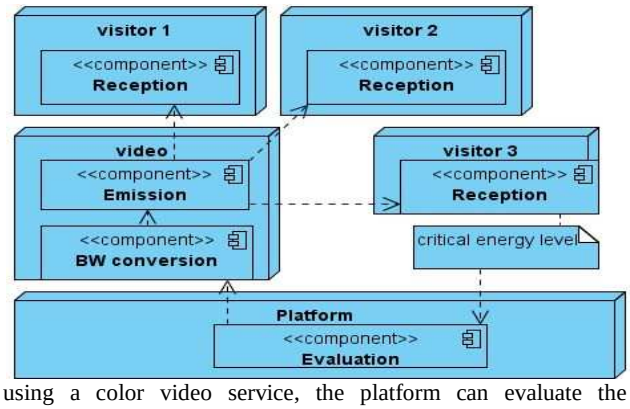
Figure 5. Dependences links between action and events triggering reconfigurations

Here are the three operations that we propose in order to provide adaptation policy:

- Create a new instance of a service,
- Suppress a new instance of a service,
- Implement a new configuration of a service. By deploying a new assembly providing the same result but with different QoS in order to ensure the current service.

The figure 5 illustrates the dependencies between events likely to trigger a reconfiguration of the application.

The first and the second operations are due to a change in user's needs, environment context or moving of a device. When a new visitor comes into the museum, the platform deploys the assembly which provides the video service to this user. Reciprocally, when a visitor does not want any more this service or when he is out of the area where this service is interesting, the platform suppresses it from the host. An environment contextual change can imply a functional change. For example, if a speech is given into a room of the museum, it is necessary not to disturb it, so to forbid the access to the audio service of the application for users entering into this room and to replace it with a text service. The third operation is related to a resources' change on the host onto the service is running. A too low level of energy, an overload of the processor, a fall in the network data rate can provide a negative evolution of the QoS and therefore, imply to provide a new assembly of components in order to ensure the continuity of the current service and to preserve the lifetime of the application. In the previous museum example, when the battery of the peripheral of the visitor #3 indicates a 50% of remaining charge, the platform will evaluate the possibility to deploy an assembly consuming less energy than the current one. For example, if the visitor #3 was



using a color video service, the platform can evaluate the assembly providing a black and white one (fig. 6).

Figure 6. Reconfiguration caused by resource changes

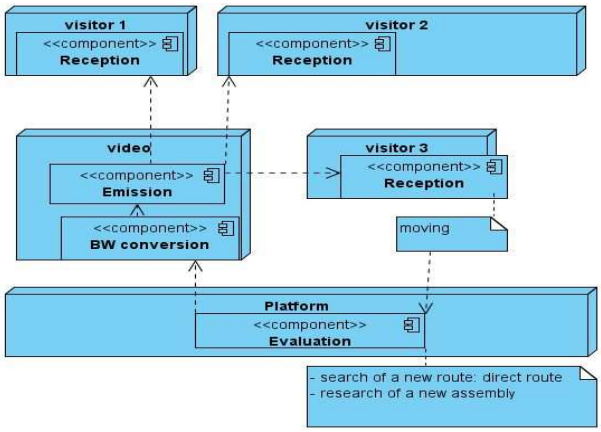


Figure 7. Reconfiguration caused by host mobility

The mobility has the particularity to provoke the three adaptation policies. Let's take the example of the visitor #3 (fig. 7). If he moves too far from the visitor #2 and can no more reach him, the

part of the assembly deployed on visitor #2's host has to be deleted. If visitor #3 goes back and visitor #2 becomes reachable again, a new instance of the video service has to be deployed on visitor #2's host. Consequently, if visitor #3 is near from video server and if it has enough energy, it is not necessary to use the visitor #2 as a relay and the platform can directly deploy the assembly and send a black & white video.

6. CHOICE OF A SOLUTION

The objective of our method is to propose to users a good QoS and to ensure the continuity of the service, whatever happens. Moreover, we particularly focus on the case of embedded peripherals for which the lifetime is essential. Our method guarantees the durability of the service at a functional and time point of view. The choice of the solution to deploy will consequently be guided by two criteria: the quality of the solution and its durability.

Durability is mainly a resources problem. We must preserve hosts' resources by implementing solutions adapted to the energy consumption problem, the power of computation and the network data rate.

The designer must establish an ID Card for each component and each host of the application. This ID Card is made of 2 parts:

- A static part listing all characteristics that do not evolve. For example, for one component, the designer has to indicate the amount of needed memory and for one host, its type which it can be: fix, CDC[1] or CLDC[2]. Fixed hosts have insignificant resources constraints, whereas CDC or CLDC hosts are the most constrained.
- A dynamic part representing the current state of the component or the host. Such characteristics evolve during execution time like state (active/inactive), memory space, data rate, battery level, CPU charge or available memory.

These ID Cards allow avoiding compatibilities errors (deployment of a CDC component on a CLDC capable host) or a too big component on a host without enough remaining memory. ID Card will guide the platform's heuristic to get the best solution. Optimizing a whole application is too complex problem, when a reconfiguration event is raised, the platform evaluate only services currently running on the host where the event comes from. The heuristic is based on two criteria: the intrinsic QoS and the durability: $QoS = f(In, Du)$

In previous works, we choose to evaluate the first configuration with minor change in order to ensure application stability [10]. Nevertheless, both the environment and the time have a negligible impact on QoS evolution comparatively to the one they have when mobile peripherals are used.

With the particular context of embedded peripheral, it is not possible to loose time and energy to grope about in order to find an optimal solution. We must implement a solution good enough to rapidly adapt the application to the evolution of the environment [13].

[1] Connected Device Configuration

[2] Connected Limited Device Configuration

Our heuristic evaluates solutions according to the classification made at the second step of the method, and then evaluates the durability using components' and hosts' ID Cards. The algorithm stops when it finds a solution candidate to be deployed. The last step consists in ensuring that the selected solution can be implemented using the topology map of hosts.

In our example, visitors #1, #2 and #3 want to use the video service. ID Cards of hosts indicate that there is neither energy nor data rate problem. The solution #1 broadcasting colour video stream is deployed onto the three hosts. If visitor #3 moves and becomes out of reach of the video service, the platform is advised about this move and look for a route to reach it. This route needs to go through visitor #2. Each peripheral has consumed a bit of energy. According to the QoS service function, the more durable solution is one that preserves energy of the host #2. The solution #2 allows reducing the amount of data to transmit by including a black and white conversion software component. So, the solution #2 is deployed with this conversion software component implemented on visitor #2's host. Next, this visitor does not want anymore this video service. However, its position does not change and visitor #3 still wants to use this video service. As well as visitor #2's resources allows it, the solution #2 remain implemented. But, as soon as an event concerning visitor #2's resources is raised, the heuristic looks for a new solution to deploy, starting by looking for a new relay allowing keeping the solution #2. If it is not possible, it will check for solution #3.

Reconfigurations events are those presented at figure 5. They are transmitted to the platform by the software components and the data flows. Software components and data flows are encapsulated into containers (called Osagaia for software components and Korrontea for data flow containers). Both containers are first class containers. The role of containers is to allow hosts compatibility (one implementation for each type of host – ie: fixed, CDC or CLDC). They are the main information source for the platform. When a container detects an evolution of its context, it raises an event to inform the platform. This one runs its heuristic in order to reconfigure the application if needed, ie - suppress/add/move software components and reconnect them.

7. CONCLUSION AND FUTURE WORK

Pervasive computing is becoming a reality. Nowadays, people can contact each other at anytime, anywhere with any device. This brings new challenges to traditional applications. As said in [16], applications should be context aware because of limited resources of the devices and the variability of the execution context. [16] propose to solve the constrained devices problem using a fault-tolerance driven approach. In [4] authors propose an ILP model to take into account different context parameters essentially based on resources constraints. Although theses approaches allow to provide service adaptation in a seamlessly way to the end user, they do not take into account the QoS dimension as we define it in section 2. Our approach monitors and adapts the whole application since the devices to end users.

So we propose a method allowing to identify all needed information in order to guaranty the continuity of applications' services and to maximize their lifetime. The durability is a fundamental notion with embedded hosts because a high quality application is useless if it just run for a few time.

The classification step that we propose allows limiting the number of solutions to be checked when an event occurs and so it avoids the well knowing NP-complete problem to look for an optimal assembly in order to reach the best QoS. This classification allows guiding the choice of the solution to deploy. It is associated to a durability criterion evaluated with the ID Cards of hosts.

However, it still exist an obvious limit to our method. The algorithm to select a solution is a heuristic that just evaluates the QoS of the host where the reconfiguration event occurred, and not the whole application. When reconfiguring, we offer the possibility to get the best QoS for one service. The reconfiguration of one service does not imply the reconfiguration of another one on another host. But, the modification and de implementation of the application (its deployment) has consequences on the execution context because it modifies the charge of hosts and the network traffic. So, a reconfiguration of one service may induce the raise of events that will trigger new reconfigurations.

Future works focus on the design and test of this configuration choice heuristic. We must specifically work on:

- Does the platform have to manage priorities on events in order to manage quicker reaction for some of them?
- When an event is managed, do we have to manage those waiting or ignore them? Doing a reconfiguration modify the context and consequently some events produced before this configuration may be obsolete.
- The evaluation function $f(In, Du)$ manages intrinsic QoS and durability. The importance between these to criteria depends on the application. For example, a video surveillance application needs to give priority to durability whereas the one presented for museums visits gives priority to intrinsic QoS in order to produce good quality information corresponding to users' demands.

Even if we do not develop this aspect in this paper, we also manage the integration of wireless sensors into such applications. We particularly work with Sun Spots through cooperation with Sun Microsystems. However such peripherals are very constrained, they can host software components and do some local processing. Moreover, they are a good information source about the environment (temperature, light, movements, etc.) and can strongly enrich applications. Such hosts can be considered by the platform as hosts participating to the infrastructure of the application and so, can be included in software component deployments.

8. REFERENCES

[1] David, P. C. et T. Ledoux. 2005. Wildcat: a generic framework for context-aware applications. In S. Terzis et D. Donsez (Eds.), MPAC, Volume 115 of ACM International Conference Proceeding Series. 1–7. ACM.

[2] Franken, L. J. N. et B. R. Haverkort. 1997. Quality of service management using generic modelling and monitoring techniques. Distributed Systems Engineering. 4, 1 (1997), 28–37.

[3] Garey M. R., Johnson D. S. 1979. Computers and Interactibility: a guide to the theory of NP-completeness, W. H. Freeman and Compagny (San Francisco, 1979).

[4] Hens R., Boone B., De Turk F., Dhoelt B. 2007. Runtime Deployment Adaptation for Resouce Constrained Devices. IEEE International Conference on Pervasive Services. (2007) 335-340.

[5] Ian D. Chakeres and Elizabeth M. Belding-Royer. 2004. AODV Routing Protocol Implementation Design. In Proceedings of the International Workshop on Wireless Ad Hoc Networking (Tokyo, Japan, March 2004). WWAN'04.

[6] ITU. International Telecommunications Union. Technical report. (htpp://www.itu.int/net/home/index-fr.aspx).

[7] Karl, H., A. Willig, and A. Wolisz (Eds.). 2004. Wireless Sensor Networks. In Proceedings of the First European Workshop on Wireless Sensor Networks (Berlin, Germany, January 19-21, 2004). EWSN 2004. Volume 2920 of Lecture Notes in Computer Science. Springer.

[8] Kuipers F., Van Mieghem P. 2002. MAMCRA : a constrained-based multicast routing algorithm. Computer Communications. 15, 802-811.

[9] Laplace, S. 2006 Conception d'Architectures Logicielles pour intégrer la qualité de service dans les applications multimédias réparties. Doctoral Thesis. Université de Pau et des Pays de l'Adour.

[10] Laplace, S., Dalmau M., Roose P.. 2007. Modèle de qualité de service pour les applications multimédias reconfigurables. Numéro Spécial Revue ISI Conception : patrons et spécifications formelles. 12,4 (2007), 115–136.

[11] Louberry, C., Dalmau M., Roose P. 2008. Architecture logicielle pour des applications hétérogènes, distribuées et reconfigurables. In Proceedings of the 8th Conférence Internationale sur les Nouvelles TEchnologies de la REpartition (Lyon, France, 2008). NOTERE 2008.

[12] Moomena, F. J. 2007 Modélisation des architectures logicielles dynamiques : application à la gestion de la qualité de service des applications à base de services Web. Doctoral Thesis. Institut National Polytechnique de Toulouse.

[13] Tournier J. C., Babau J. P., Olive V. 2005. Qinna, a component-based QoS architecture. In Proceedings of the 8th International Symposium on Component Based Software Engineering. (St. Louis MO, 14-15 May 2005). CBSE'05. Lecture notes in computer science. Springer.

[14] Vogel, A., B. Kerhervé, G. von Bochmann, et J. Gecsei. 1995. Distributed multimedia and qos : A survey. IEEE MultiMedia. 2, 2, 10–19.

[15] Wang Z., Crowcroft J. 1996. Quality-of-service routing for supporting multimedia applications. Journal of Selected Areas in Communications. 14, 7 (September, 1996), 1228-1234.

[16] Zheng D., Wang J., Jia Y., Han W., Zou P. 2007. Deployment of Context-Aware Component-Based Applications Based on Middleware. UIC. Volume 4611 of Lecture Notes in Computer Science. Springer.