# Improvement of the QoS via an Adaptive and Dynamic Distribution of Applications in a Mobile Environment


Françoise André
*IRISA/University of Rennes 1*
*Campus de Beaulieu*
*35042 Rennes Cedex, FRANCE*
*fandre@irisa.fr*

Anne-Marie Kermarrec
*Microsoft Research Ltd.*
*St George house*
*1, Guildhall Street*
*Cambridge CB2 3NH. UK*
*annemk@microsoft.com*

Frédéric Le Mouël
*IRISA/INRIA*
*Campus de Beaulieu*
*35042 Rennes Cedex, FRANCE*
*flemouel@irisa.fr*



## Abstract

*Mobile computing is a domain in great expansion. Wireless networks and Portable Information Appliances (PIAs) are developing very rapidly. More and more mobile users would like to perform their multimedia applications with the same facility as on their desktop station. Use of such applications in a mobile environment raises new challenges. These applications are interactive and extremely costly in system and network resources, whereas PIAs resources are poor and wireless networks offer a very variable quality of connection. In this paper, we propose an adaptive and dynamic distribution of applications on the local environment to overcome the poorness of available resources on PIAs, and to reduce and regulate variability effects.*


## 1. Introduction

Mobile computing is a fast-expanding domain which benefits from the great technical advances in wireless network domain (IrDA, WLAN, GSM, Satellite) and in Portable Information Appliances domain (PIAs: smart cards, cellular phones, Personal Digital Assistant, handheld computers, notebooks, laptops) [14]. In this context, more and more mobile users would like to execute their applications with the same facility as on their desktop station and would expect the same results. Most of multimedia applications are interactive (text editors, browsers, etc) and costly in system and network resources (pictures viewers, music players, video-conferences, etc). Use of such applications in this mobile context raises new challenges which are not handled by traditional systems [8]:

- Due to width and height limitations, PIA are resource-poor compared to static stations in terms of computation power, available memory, storage capabilities, screen display, etc. Moreover, these resources may vary considerably. For example, PCMCIA port makes possible to add or remove different devices such as a printer, a disk, etc. CPU or disk sleeps, or screen resolution variations can also be the result of a low-power strategy due to the battery life limit.
- Wireless connections used by PIA suffer from a low and extremely variable bandwidth and from frequent unexpected disconnections caused by interferences with the environment (material obstacles such as a door, a wall, etc or other electronic appliances), by handoff blank out in cellular network or by the user roaming-off outside the coverage area of the wireless network. Other disconnections may also be the fact of user's voluntary decision or may be due to a long downtime of the PIA for battery lifetime reason or hostile events such as security problems, theft or destruction.
- PIA may experience dramatic changes in their environment either at the hardware level, with the arrival or the removal of stations (mobile or not) or devices (printers, scanner, etc), or software changes may also happen with the arrival or the removal of system functionalities (prefetching, etc) or application services (weather, traffic, etc).

In this paper, we propose to overcome the effect of such variations with an adaptive and dynamic distribution of applications over mobile environments. This distribution allows to overcome the poorness of available resources on PIAs by taking benefit of available resources of stations or other appliances of the environment. It also allows to reduce and regulate the effect of variability by placement and load balancing techniques. First, our distribution mechanism is adaptive in the sense that it takes application needs into account to define the relevant strategies in terms of resources and services needs (for instance, the search for a workstation where a compression algorithm exists) or pertinent distribution (for instance, the search for a high bandwidth link to transmit video streams). Second, as the environment varies, our distribution scheme is dynamic in two different ways: changes of the environment are dynamically taken into account to adapt both the placement, and the policy of placement itself.

The rest of the paper proceeds as follows. First, we examine, in Section 2, other works both in mobile computing and in distributed computing. Next, in Section 3, we explain to what extent our approach improves the Quality of Service (QoS). In Section 4, we introduce our system AeDEn which provides a framework to implement our adaptive and dynamic distribution scheme: architecture, implementation and some preliminary results. We conclude and give indications of our future works in Section 5.

## 2. Related works

Most of works in mobile computing area deal with the disconnection problem and the bandwidth bottleneck whereas few approaches deal with the poorness of PIAs resources and the use of remote resources to overcome this lack. On the contrary, works in distributed computing area aim at using remote resources in a transparent and efficient way but do not take mobile environment characteristics into account. Since our proposition aims at integrating both approaches (mobility and distribution), we successively present these two areas.

### 2.1. Mobile computing

Many architectures or systems supporting adaptiveness have been introduced in the area of mobile computing.

Most of these works deal with the disconnection problem and the bandwidth bottleneck. We distinguish between two approaches: *application-transparent* and *application-aware approaches* [23]. Application-transparent approaches aim at providing a mobile device with the same set of services as on a desktop machine. They consist in masking mobile aspects at a network or system level by emulating the behavior of protocols of a static system. For example, Mobile-IP [22] is a protocol which allows mobile devices to keep their own address in spite of mobility. The Coda File System [16] offers the user the possibility to work in disconnected mode, masking disconnection periods by caching and hoarding techniques. Nevertheless, masking cannot be complete: since these approaches ignore totally or partially the application behavior, they are not able to handle all variations.

Application-aware approaches partially solve this latter drawback. Some environment informations are provided for the applications in order to enable them to adapt their behavior and their quality to the changing conditions. For example, the BARWAN project [15] developed a network and transport layer which supports adaptation to the available bandwidth and latency of the wireless network and provides different functionalities for the applications such as an adaptive QoS support, low latency handoff, user tracking, etc. Odyssey [21] is a software platform which monitors resources, detects the changes and notifies interested applications, which are in charge to react and adapt consequently. The multimedia application presented in [13] (an MPEG player) uses software feedback to dynamically discover the appropriate data stream to use (resolution, color depth, palette, frame rate) and detects when it becomes inappropriate.

In comparison with these previous approaches which try to overcome the bandwidth bottleneck, few approaches aim at overcoming the poorness of PIAs resources. The Mobile Agent model is one of them [11]. A mobile agent is an active program able to move through a network under its own control and to interact locally with resources or services. This model allows to delegate actions such as informations gathering and filtering to the mobile agent on the fixed network. Only the resulting data are transmitted to the mobile client. For example, Alycta [7] provides an efficient access to the WWW over GSM by delegating downloading documents and degrading their quality to mobile agents that performs the tasks off-line. The Client/Proxy/Server model is another approach which also uses resources external to the PIAs. In the C/P/S model, the proxy and the server are in the fixed network. The proxy uses resources of the static station where it is located to provide network or application facilities for the mobile user such as efficient protocols (Mowgli [17]), storage of results while disconnections, filters such as compression, degradation, distillation, display media conversion (WAP [27]).

Distribution of proxies can also be used to increase the number of available resources. For example, distribution techniques are used in Daedalus [10] to deploy adaptation-based proxy services (TACC servers) on a cluster of PCs. The applied strategy consists in increasing the number of proxies according to the number of requests in waiting queues. This approach is nevertheless specific: distributed processes are exclusively of a specified type (proxy) defined by the adaptation system designer for a particular task; the distribution strategy is also fixed and cannot take new criteria into account. To distribute an application in a more general way, distribution techniques proposed in the distributed computing area should rather be considered.

### 2.2. Distributed computing

Many distributed systems have been proposed to satisfy an important need of resources. They can be classified according several properties such as *performance*, *transparency*, *load-balancing and stability*, *scalability* and *fault-tolerance*.

CORBA [19], one of the first advance in terms of standardization of distribution which ensures transparency, does not propose load-balancing and fault-tolerance functionalities. NoW system approach [1] provides applications with a transparent distribution over any resources (CPU, memory and disk) of a local network of workstations.

Legion [12] and Globe [26] are meta-systems which principally aim at ensuring scalability. They provide an extensible, efficient and secure software infrastructure supporting millions of computers, users and distributed objects. Owing to an object-oriented approach, they allow to separate object semantic from its interface and its implementation.

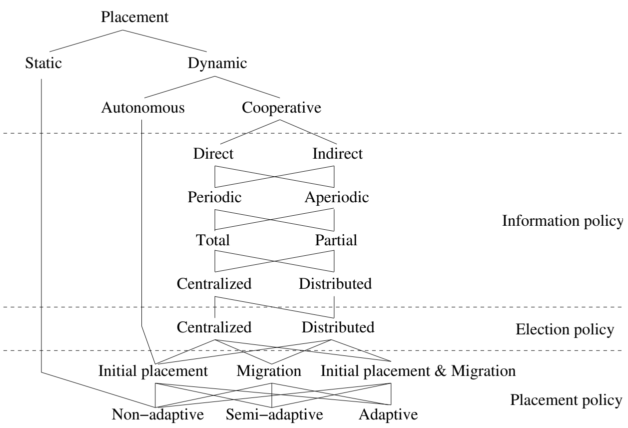


**Figure 1. Placement algorithms**

Many algorithms exist which ensure good performances, load-balancing and stability. These algorithms are composed of three policies [28]: *the information policy* which specifies the nature of informations needed for the election (CPU load, etc) and defines how they are collected, *the election policy* which elects processes which must be suspended, placed or migrated according to the collected informations and *the placement policy* which chooses the machine for each elected processes. The main algorithms [2] are resumed in Figure 1 and have been experimented in different systems such as Radio [3] for non-adaptive and non-preemptive algorithms, Condor [18] for preemptive ones, GatoStar [9] for placement and migration ones, PM$^2$ [20] and Startdust [5] for adaptive ones. In these systems, performance is mainly based on load sharing according to CPU criterion.

Other algorithms aim at ensuring the QoS needed by the application. These algorithms are mainly based on an adaptation to bandwidth variations such as the lost of packets, the delay of transmission or the bit and packet transmission rate [25]. To support the network infrastructure in the best possible way, some of these algorithms are distributed such as the *distributed QoS adapters* in [6]. Moreover, the QoS can be improved by taking other criteria into account such as the end-station load in [4].

Multi-criteria approaches exist [29] but they do not take into account the new criteria introduced by mobile computing such as a low and variable bandwidth, a probability of disconnection (previous approaches assume to have a high and continuous connection) or a battery index. Moreover, in a mobile environment, the choice of a good strategy will greatly depend on the location of the PIA, which changes overtime. For example, let's consider the information policy of a laptop. As long as the laptop is connected via a dock-station to a local network with a central server, it is reasonable to have a centralized, total, direct, and periodic information policy. When the laptop moves into the building, keeping the connection via a wireless link, a better policy would be centralized, partial, direct and aperiodic in order to reduce the network traffic. If the laptop moves out-door and is connected to an ad-hoc network, the policy should be switched to a distributed, partial, indirect and aperiodic one.

The approach proposed in this paper addresses the changes of the environment, first, to distribute the load, and second, to adapt the way the distribution is performed.

## 3. Distribution: a way of improving mobile computing QoS

In this paper, we name the *environment* of a PIA the set of *data*, *resources* and *services* in the PIA cell (cell of GSM, WLAN or ad-hoc network): *resources* are hardware elements of the PIA (CPU, memory, disk, bandwidth, printer, etc) and hardware elements of the environment (scanner, fax, other station resources, etc); *services* are software elements at a system level (naming service, caching service, etc) or at an application level (electronic business, electronic press, etc). The *application* is modeled as a set of communicating *components*. Each component knows data, resources and services that it needs.

Our approach aims at improving the QoS of applications running in mobile environments by taking advantage of resources or services of this environment through distribution techniques. The improvement is obtained along two axes. First, when a PIA used in a fixed way, becomes mobile, our distribution aims at degrading the QoS as less as possible. Second, when a PIA is already in movement, the distribution aims at improving the QoS as soon as possible when resources and services appear and disappear. These QoS improvements can be expressed in terms of:

- **User interactivity quality**: the user interactivity quality is measured through the response time of the system or applications. This response time depends on mobile physical constraints as the memory, the available CPU, the disk size especially if several applications are running. To improve user interactivity quality, distribution allows to place components which directly interact with the user on the PIA, and to deport the others (non-interactive) in the local environment. This results in (i) a decrease of the use of resources and services of the PIA, (ii) a better use of resources and services of the environment which allows to optimize the execution time, (iii) a way to overcome disconnections because computation continues off-line and results are transmitted while reconnecting.

- **Data quality**: data are constituted by texts, images, videos, sounds, etc. For different reasons (gain in response time, adaptiveness to screen size, etc), original data needs sometimes to be transformed by operations such as compression, distillation of images, truncations of PIA non-adapted data, etc. These degradations or modifications of data may be prevented or lessened in some cases by the use of multi-machines environment. By using the environment resources and services, transformations results quality may either be improved (over standard execution) while maintaining the same response time, or remain the same while improving the response time.

- **Accessibility**: distribution allows applications to transparently use resources and services of the environment by an appropriate placement of the application components. This placement is dynamically readapted to new situations, for instance when resources or services appear or disappear.

In order to achieve these goals, our distribution mechanism is designed to be application-adaptive, so that it can be customized to different applications having different needs and constraints (see Figure 2). This is achieved without loosing generic aspects, thanks to the use of MolèNE [24], an object-oriented framework which adaptation facilities are described in Section 4.3.1. Besides, to deal with the great variability of mobile environments, the distribution mechanism must also be dynamic and that at two levels:

- Upon a notification of environment changes, a given strategy reacts by changing the current distribution (for example, if some station is going down, the work it carries out should be redistributed).

- The distribution mechanism itself should be able to radically change its strategy (for example, if the PIA moves from an in-door to an out-door network, the algorithm implementing the distribution policy should be changed).

Our distribution algorithm is composed of the classical distribution policies: information policy, election policy and placement policy and the strategies implementing these policies can dynamically change according to application or environment changes.

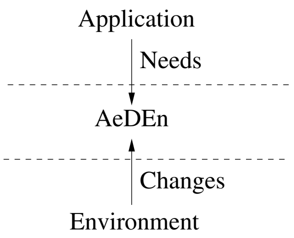


**Figure 2. Dynamism and adaptation**

# 4. AeDEn: an Adaptive Distribution Environment

## 4.1. AeDEn services

Our distribution system AeDEn is a middleware layer made up of services. We divide our system in services for extensibility and reuse reasons. Indeed, it must be possible to extend a service with a new policy without modifying the other services. Moreover, each service can be used independently of the other ones. For example, an application can monitor the bandwidth and react in function of the variations, independently of the other distribution services.

Figure 3 shows how we divide our system in three main services: the *Detection & Notification Service*, the *Environment Management Service* and the *Distribution Service*. These services implement the policies of the distribution algorithm: the election and placement policies are part of the distribution service, the information policy is shared between the Environment Management Service which stores the resources and services states (state management policy) and the Detection & Notification Service which detects environment changes (variation detection policy) and notifies the Environment Management Service accordingly.

**Detection & Notification Service**

The primary role of the Detection & Notification Service is to provide the Environment Management Service with environment changes detected by the underlying operating system or network layers.

The Environment Management Service, according to the needs specified by the application, registers resources and

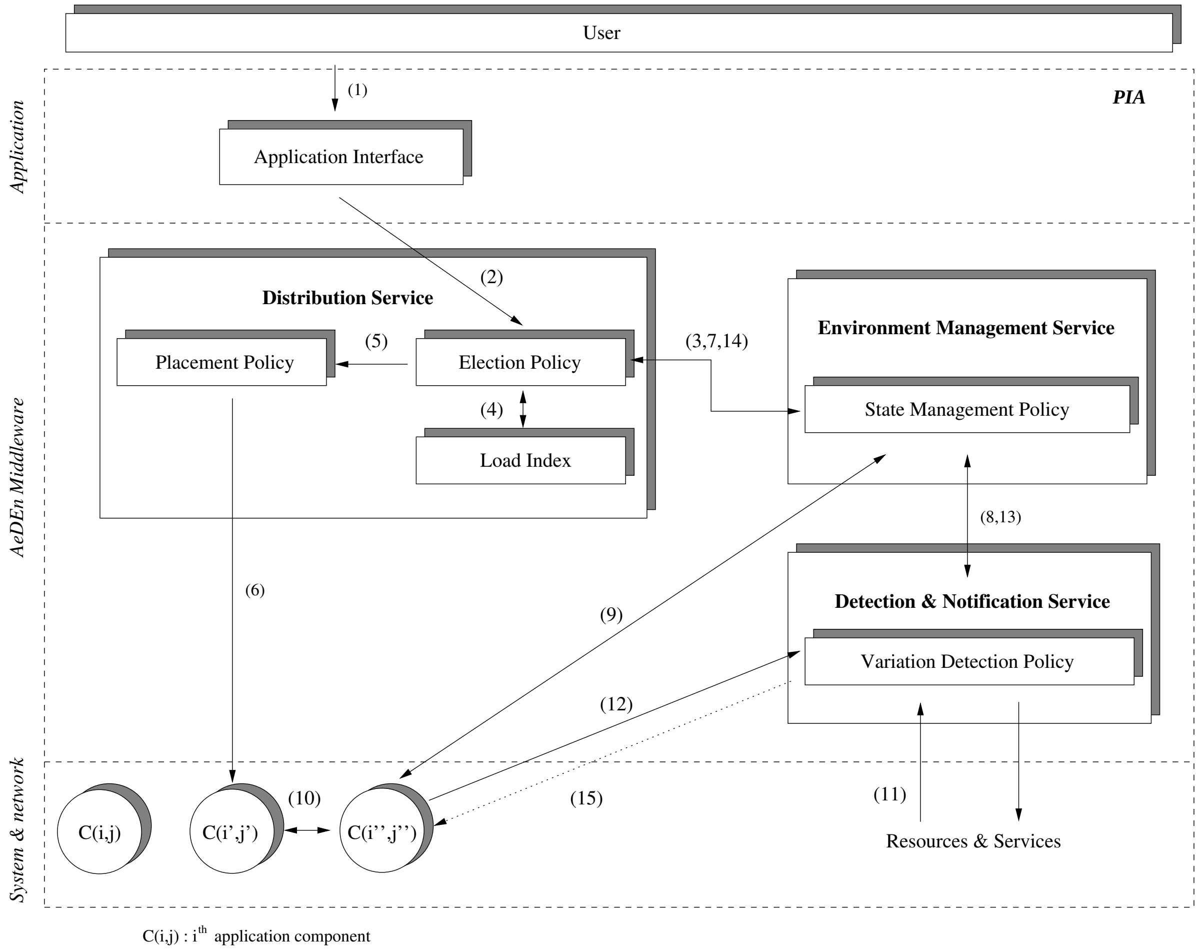


**Figure 3. AeDEn architecture**

services interests in the Detection & Notification Service. These interests can be expressed either as simple conditions on one resource (or one service) state or as complex conditions on several resources and/or services states. Simple interests are implemented by low-level monitors which are composed of a set of low-level attributes on resources they are in charge of, and conditions on these attributes. Low-level attributes are for instance CPU workload, memory size, disk capacity, network performance (bandwidth, latency), power consumption. Conditions can be expressed by acceptable values for an attribute. High-level monitors represent complex interests expressed by a boolean expression depending on low-level monitors conditions and other high-level monitors conditions. Once a condition is not satisfied in a monitor, the Environment Management Service is notified of the changes.

### Environment Management Service

The Environment Management Service maintains the description of resources and services of the current environment. This service is implemented as a database of elements (resources or services), and each element is associated with a name, a value, a host or a PIA with which it is attached and a set of applications components which use this element. When an application references a component, the Environment Management Service retrieves the component and indicates its location and the resources or services it is using. It also receives informations from the Detection/Notification Service when a change has been detected in monitored resources or services and updates consequently the current environment.

### Distribution Service

The Distribution Service is in charge of distributing application components. It receives application components needs and asks the Environment Management Service about the current resources and services in the environment. Using these informations, it calculates the load indices and decides which components are elected for the distribution. The load indices can be calculated by taken into account the environment using a one or multi-criteria approach, or by also considering components needs using a QoS manager. Then, the Distribution Service places the components accordingly and registers them in the Environment Management Service. It also records the placement that has been realized. Upon an environment change, the Environment Management Service is notified of the change and notifies the Distribution Service. The Distribution Service re-engages, if necessary, the distribution process (suspension, migration, placement) according to the defined election and placement policies.

## 4.2. Services interactions

AeDEn services interact with the application, the operating system and the network. Interactions with the application constitute the adaptive aspects of our system, and interactions with the environment constitute the dynamic ones.

**4.2.1. Application-adaptive distribution.** Relationships with the application take place in two circumstances (see Figure 3):

- *Launch of the application*: the user starts the application (1). The different components of the application are submitted to the Distribution Service and particularly the election policy (2). The election policy interacts with the Environment Management Service to have resources and services states (3), then it calculates load indices (4). The components are submitted to the placement policy (5) which transfers them in the adequate places (6), registers them in the environment (7) and activates a monitoring of these components (8).

- *Communication between components*: when a component wants to communicate with an other one, it asks the Environment Management Service (9) and receives the location of this component (10).

**4.2.2. Dynamic placement.** When the Detection & Notification Service detects a change in resources or services of the environment (11) or in components (for example removal) (12), it notifies the Environment Management Service which updates its information database (13). This one notifies the Distribution Service if distributed components are involved[1] (14). Then the election policy determines if these changes affect the placement. In this case, a new placement is realized (3,4,5,6). The step (6) performs the placement of new components, the suspension, the migration of existing components, as needed.

[1] Application components can also have been designed to adapt themselves according to environment changes, and so they are directly notified by the Detection & Notification Service (15).

## 4.3. Dynamic policies

As we said in Section 2.2, a strategy can be more or less relevant depending on the location of the PIA. To deal with these changes, we introduce dynamic policies which allow the system to switch dynamically from one strategy to another one as changes occur in the environment. Each policy implements only one strategy at time $t$ but can dynamically change and have a different strategy at time $t + 1$.

This opportunity to dynamically change the implementation of a policy requires well-adapted and efficient programming tools. We use the MolèNE object-oriented framework which provides this facility.

**4.3.1. MolèNE framework.** Thanks to the reflection and introspection techniques of object-oriented languages, we have built an object-oriented framework MolèNE [24] that provides a way to dynamically change the implementation of a component (the smallest software unit in MolèNE). In this section, we present the mechanism which allows to switch from one implementation to another one. It is based on an *Adaptation Algorithm*, which decides when to proceed to an implementation change and a *Controller* which executes the change. Figure 4 shows the set of classes and their relations that constitutes a MolèNE component.

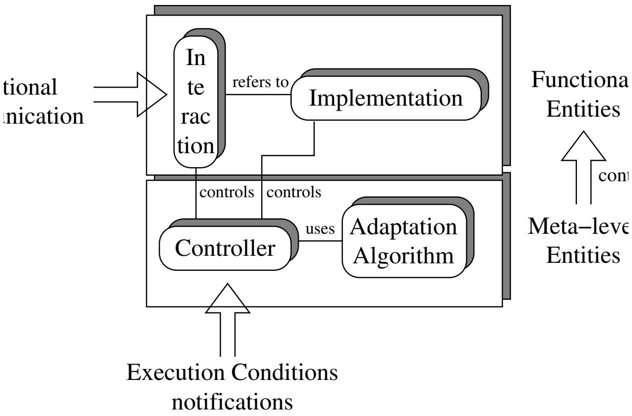


**Figure 4. Infrastructure of a MoleNE component**

The *Interaction* receives functional *Events* (for example, the set of the applications components which have to be placed in the case of the election policy) and transmits them to the current *Implementation* which is the code performing the functionality of the component. The *Controller*, according to the non-functional events that it receives (i.e. changes in execution conditions) and to the *Adaptation Algorithm* can decide whether or not to switch to another strategy. If a switch is decided, the current *Implementation* is replaced

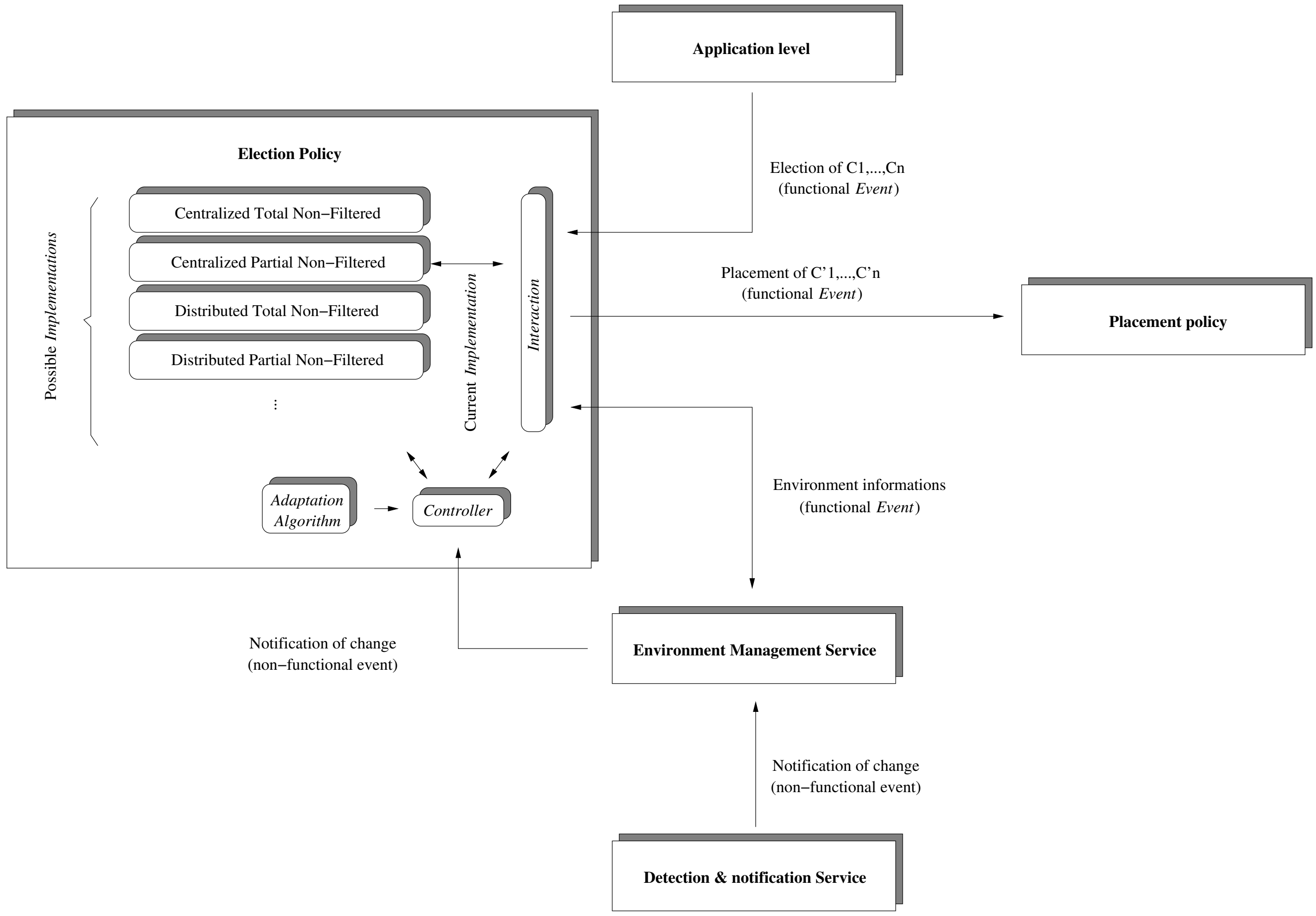


**Figure 5. Strategies for the Election Policy**

by another code, while keeping the necessary informations for the new strategy and the links with the *Interaction* part.

**4.3.2. Example of a Dynamic Policy: the Election Policy.** We used the MolèNE framework to implement the first AeDEn prototype. Each AeDEn service is implemented as a set of MolèNE components. MolèNE components are perfectly adapted to implement the dynamic policies. Services also use functional *Events* to interact among themselves, with the application or with the environment.

We illustrate the use of dynamic policies with the election policy (see Figure 5). The election policy receives functional events via its interface from the application level and the Environment Management Service. It also sends functional events to the placement policy. These events correspond to those described in Section 4.2.1 and 4.2.2. The election policy may also receive non-functional events via its controller. These events are notifications of changes in the environment which indicate that the current election strategy may be no longer adequate. The current implementation is then changed by the controller according to a defined adaptation algorithm. For example, when a user moves from an in-door wireless network to an ad-hoc network, the controller switches the current election strategy from a centralized and total one to a distributed and partial one. Implementations of strategies can also be added or removed dynamically.

### 4.4. Implementation and preliminary results

Existing distribution strategies have already been studied in literature in terms of performance, migration cost, etc. Our approach introduces a new factor: the possibility of switching from a strategy to another one which copes with the need of adaptation of mobile computing. We examine below the cost of this facility in terms of bandwidth use and response time. We have implemented a preliminary AeDEn prototype in Java that we have evaluated. All measurements have been realized with an IBM ThinkPad with a WaveLan link as the mobile environment, and a Sun Ultra-1 and an Ultra-60 with a 100Mb/s link representing the fixed network.

We have tested the bandwidth variations produced by the switch from a centralized information strategy to a distributed information strategy. Informations on the state of the environment, that we consider, are the CPU, the battery and the bandwidth and the latency of the network. To measure the bandwidth, we have used a continuous stream of data and we have measured, every $100ms$, the stream rate. This monitoring introduces an overhead of $25Kbytes/s$ but allows to have fine-grain measurements of the variations produced by our switching mechanism. This measurement has been conducted 10 times with similar results. Of course, the monitoring overhead depends on the variation detection policy used.

First, the Ultra-60 station acts as information server and the others act as clients and, then, they all switch to the distributed mode. Figure 6 shows the bandwidth variations of the WaveLan link: at $t = 10s$, stations switch to the distributed mode and receive informations on the state of the environment, at $t = 20s$, stations switch again to the client/server mode and send the environment state to the server. We observe that the bandwidth use at the switching time represents less than $20\%$ of the total bandwidth. Moreover, this bandwidth variation is not so different from the other variations due to the wireless connection.

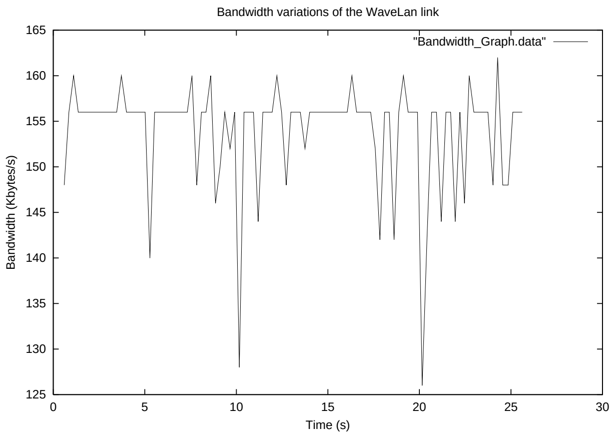


**Figure 6. Bandwidth Variations/Information Strategies Changes**

We have also measured the time to switch from an implementation to another by using local clocks of stations. The actual switch mechanism takes $5ms$ ($25ms$ when called the first time because of the class load). In the case of the information strategy, we should also take into account the transmission of the environment state which takes about 275 to $295ms$.

The cost of our switching mechanism is thus reasonable. It does not use too much bandwidth (the bandwidth use can be compared to the one needed to transfer a reasonable web page including images and text). Moreover, we can expect that the change of strategy will only be decided when drastic environment changes occur such as when the PIA goes from one network to another.

We have implemented a preliminary prototype and we tuned the parameters by hand in the first place. Automatic switching of policies is not yet available. Further experiments need to be conducted in order to set up the reasonable and adequate thresholds of monitored resources to be used to switch from one policy to another.

## 5. Conclusion and Future Work

In this paper, we have presented our system AeDEn which provides an adaptive and dynamic distribution of applications in mobile environments. This distribution allows to overcome the poorness of PIAs resources by using resources and services of the environment. It also allows to reduce variability effects by executing components of the application on entities which are less sensitive to variations. Proposed distribution mechanisms are dynamic in the sense that the placement and the distribution policies change according to environment changes, and they are also adaptive in the sense that they take into account applications needs.

We are currently working on the influence of the different criteria on the different policies to elaborate adaptation algorithms. We are also working about the horizontal consistency (between strategies on different PIAs) and the vertical consistency (between information, election and placement policies). Indeed, changing a strategy on a station affects the others and one should avoid inconsistent states. Finally, we plan to test our system with a real electronic press application. This type of application is suitable for such an implementation since it is interactive, manipulates an important volume of data and requires quite a lot of computation power.

## References


[1] T. E. Anderson, D. E. Culler, D. A. Patterson, and the NOW team. A case for NOW (networks of workstations). *IEEE Micro*, 15(1):54–64, Jan. 1995.

[2] G. Bernard and B. Folliot. Caractéristiques générales du placement dynamique : Synthèse et problématique. In *Tutoriel invité dans les actes de l'école d'été MASI-IMAG-INT-PRISM "Placement dynamique et répartition de charge : application aux systèmes parallèles et répartis"*, Presqu'île de Giens, France, July 1996.

[3] G. Bernard, D. Stève, and M. Simatic. Placement et migration de processus dans les systèmes répartis faiblement couplés. *Technique et Science Informatiques*, 10(5), May 1991.

[4] J. Bom, P. Marques, M. Correia, and P. Pinto. QoS control: an application integrated framework. In *Proceedings of the*

*IEEE Internationale Conference on ATM*, Colmar, France, June 1998.

[5] G. Cabillic and I. Puaut. Dealing with heterogeneity in stardust: An environment for parallel programming on networks of heterogeneous workstations. In *Proceedings of 2nd International Euro-Par Conference (Euro-Par'96)*, pages 114–119, Lyon, France, Aug. 1996.

[6] A. Campbell and G. Coulson. A QoS adaptive transport system: Design, implementation and experience. In *Procceedings of the 4th ACM International Conference on Multimedia'96*, pages 117–127, Boston, USA, Nov. 1996.

[7] X. Delord, S. Perret, and A. Duda. Efficient mobile access to the WWW over GSM. In *Proceedings of the 8th ACM SIGOPS European Workshop Support for Composing Distributed Applications*, pages 1–6, Sintra, Portugal, Sept. 1998.

[8] D. Duchamp. Issues in wireless mobile computing. In *Proceedings of the 3rd IEEE Workshop on Workstation Operating Systems*, pages 2–10, Key Biscayne, Floride, USA, Apr. 1992.

[9] B. Folliot and P. Sens. GATOSTAR: A fault tolerant load sharing facility for parallel applications. In K. Echtle, D. Hammer, and D. Powell, editors, *Proceedings of the 1st European Dependable Computing Conference*, volume 852 of *Lecture Notes in Computer Science*. Springer-Verlag, Oct. 1994.

[10] A. Fox, S. D. Gribble, Y. Chawathe, and E. A. Brewer. Adapting to network and client variation using infrastructural proxies: Lessons and perspectives. *IEEE Personal Communications (invited submission)*, 5(4):10–19, Aug. 1998.

[11] R. Gray, D. Kotz, S. Nog, D. Rus, and G. Cybenko. Mobile agents for mobile computing. Technical Report PCS-TR96-285, Department of Computer Science, Dartmouth College, May 1996.

[12] A. S. Grimshaw and W. A. Wulf. The legion vision of a worldwide virtual computer. *Communication of the ACM*, 40(1):39–45, Jan. 1997.

[13] J. Inouye, S. Cen, C. Pu, and J. Walpole. System support for mobile multimedia applications. In *Proceedings of the 7th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'97)*, pages 143–154, St. Louis, Missouri, USA, May 1997.

[14] A. Jones. Mobile computing to go. *IEEE Concurrency*, 7(2):20–23, Apr. 1999.

[15] R. H. Katz, E. A. Brewer, E. Amir, H. Balakrishnan, A. Fox, S. Gribble, T. D. Hodes, D. Jiang, G. T. Nguyen, V. N. Padmanabhan, and M. Stemm. The bay area research wireless access network (BARWAN). In *Proceedings of the 41st IEEE Computer Society International Conference: Technologies for the Information Superhighway (COMPCON'96)*, pages 15–20, Santa Clara, California, USA, Feb. 1996.

[16] J. Kistler and M. Satyanarayanan. Disconnected operation in the coda file system. *ACM Transactions On Computer Systems*, 10(1):3–25, Feb. 1992.

[17] M. Liljeberg, H. Helin, M. Kojo, and K. Raatikainen. Enhanced services for world-wide web in mobile WAN environment. In *Proceedings of the IEEE Global Internet 1996 Conference (revised version)*, London, England, Nov. 1996.

[18] M. J. Litzkow and M. Livny. Experience with the condor distributed batch system. In *Proceedings of the IEEE Workshop on Experimental Distributed Systems*, pages 97–101, Huntsville, Alabama, USA, Oct. 1990.

[19] T. J. Mowbray and W. A. Ruh. *Inside CORBA*. Addison Wesley, 1997.

[20] R. Namyst and J.-F. Méhaut. $PM^2$: Parallel multithreaded machine. A computing environment for distributed architectures. In E. H. D'Hollander, G. R. Joubert, F. J. Peters, and D. Trystram, editors, *Parallel Computing: State-of-the-Art and Perspectives, Proceedings of the Conference ParCo'95, 19-22 September 1995, Ghent, Belgium*, volume 11 of *Advances in Parallel Computing*, pages 279–285, Amsterdam, Feb. 1996. Elsevier, North-Holland.

[21] B. D. Noble, M. Satyanarayanan, J. Tilton, J. Flinn, and K. Walker. Agile application-aware adaptation for mobility. In *Proceedings of the 16th ACM Symposium on Operating Systems Principles (SOSP'16)*, Saint-Malo, France, Oct. 1997.

[22] C. E. Perkins. Mobile networking through mobile IP. *IEEE Internet Computing*, 2(1):58–69, Jan. 1998.

[23] M. Satyanarayanan. Fundamental challenges in mobile computing. In *Proceedings of the 5th Annual ACM Symposium on Principles of Distributed Computing (PODC'96)*, Philadelphia, Pennsylvania, USA, May 1996.

[24] M. Segarra and F. André. A framework for dynamic adaptation in wireless environments. In *Proceedings of the Technology of Object-Oriented Languages and Systems (TOOLS'2000)*, Saint Malo, France, June 2000.

[25] T. Talley and K. Jeffay. Two-dimensional scaling techniques for adaptive, rate-based transmission control of live audio and video streams. In *Proccedings of the 2nd ACM International Conference on Multimedia'94*, pages 247–254, San Francisco, USA, Oct. 1994.

[26] M. van Steen, P. Homburg, and A. S. Tanenbaum. Globe: A wide-area distributed system. *IEEE Concurrency*, pages 70–78, Jan. 1999.

[27] Wireless Internet Today. *Wireless Application Protocol - White Paper*, Oct. 1999.

[28] S. Zhou. A trace-driven simulation study of dynamic load balancing. *IEEE Transactions on Software Engineering*, 14(9):1327–1341, Sept. 1988.

[29] S. Zhou, J. Wang, X. Zheng, and P. Delisle. Utopia: A load sharing system for large, heterogeneous distributed computer systems. Technical Report 257, Computer Systems Research Institute, Toronto University, Dec. 1991.