AeDEn : An Adaptive Framework for Dynamic Distribution over Mobile Environments

AeDEn : une plateforme adaptative pour la distribution dynamique d'applications en environnements mobiles

F. Le Mouël[*] — F. André[**] — M.T. Segarra[***]

[*] *École des Mines de Nantes*

Département informatique

4, rue Alfred Kastler, BP 20722

44307 Nantes Cedex 3, FRANCE

Frederic.Le-Mouel@emn.fr

[**] *IRISA/Université de Rennes 1*

Campus de Beaulieu

35042 Rennes Cedex, FRANCE

Francoise.Andre@irisa.fr

[***] *ENST Bretagne*

Département informatique

Technopôle de Brest Iroise, BP 832

29285 Brest Cedex, FRANCE

MT.Segarra@enst-bretagne.fr

Annales des télécommunications. Volume 57 - n° 11-12/2002

ABSTRACT. *Mobile computing is a domain in great expansion. Wireless networks (GSM, satellite, etc) and Portable Information Appliances PIAs (laptops, PDAs, cellular phones, etc) are developing very rapidly. More and more mobile users would like to execute their applications with the same quality of service as on their desktop station, whatever their needs in memory and computing power. Using such applications in a mobile environment raises new challenges. Some of these applications are 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. Moreover, due to the variety of distribution policies, we propose a framework providing adaptive distribution policies.*

RÉSUMÉ. *L'informatique nomade est un secteur en pleine évolution aussi bien au niveau des terminaux mobiles (ordinateurs portables, assistants personnels, téléphones portables, etc), qu'au niveau de l'infrastructure et des protocoles des réseaux de communication sans fil (GSM, satellites, etc). De plus en plus d'utilisateurs mobiles veulent pouvoir utiliser leurs applications de la même manière qu'en environnement fixe et cela, quel que soit le volume de données à traiter et quel que soit le temps de calcul demandé. L'utilisation de telles applications à partir de mobiles posent néanmoins certains problèmes. D'un côté, ces applications sont coûteuses en ressources système et/ou réseau, et de l'autre côté, le mobile offre peu de ressources et celles-ci sont fluctuantes. Nous proposons une approche consistant en une distribution adaptative des applications permettant d'utiliser dynamiquement les ressources de l'environnement pour pallier l'insuffisance des ressources du mobile. De plus, face à la diversité des politiques de distribution existantes, nous proposons un cadre général de conception permettant une utilisation de politiques adaptables en fonction de l'environnement.*

KEYWORDS: *Mobile Computing, Distributed Computing, Adaptive Systems, Generic Approach, Adaptive Distribution Policies.*

MOTS-CLÉS : *Informatique nomade, informatique répartie, systèmes adaptatifs, approche générique, politiques adaptables de distribution.*

1. Introduction

Mobile computing is a fast-expanding domain which benefits from great technical advances in wireless networks (IrDA, WLAN, GSM, Satellite) and Portable Information Appliances (PIAs) domains (smart cards, cellular phones, Personal Digital Assistants, handheld computers, notebooks, laptops) [JON 99]. Meanwhile, there is an increasing interest in executing applications on a mobile environment. However, most of them are interactive (text editors, browsers, etc) and costly in system and network resources (pictures viewers, music players, video-conferences, etc). Their execution may lead to a QoS degradation for three main reasons:

– Due to weight and size limitations, PIAs are resource-poor devices compared to fixed stations in terms of computation power, available memory and secondary storage, screen display, etc [SAT 96]. Moreover, these resources may vary considerably when PCMCIA cards are used.

– Wireless connections used by PIAs suffer from a low and extremely variable bandwidth[1] due to interferences with the environment (material obstacles such as a door, a wall or other electronic appliances), handoff blank out in cellular networks or the user roaming-off outside the coverage area of the wireless network. Disconnections may also result from user's voluntary decision or downtime of the PIA for battery lifetime reasons or hostile events such as security problems, theft or destruction.

– PIAs may experience dramatic changes in their environment either at the hardware level, with the arrival or removal of stations (mobile or not) or devices (printers, scanner, etc), or at the software level with the arrival or removal of system functionalities (prefetching, etc) or application services (weather, traffic, etc).

In this paper, we propose the AeDEn system (Adaptive Distribution Environment) as a mean to effectively distribute application code and data over a rapidly varying environment. Changes in the environment are taken into account at two very different levels: the application and the distribution system itself. The first one concerns the mapping of the application components onto a set of stations. AeDEn uses a distribution algorithm in order to dynami-

1. Disconnections are considered as a case of extreme variability.

cally change the placement of application components according to environmental changes. The second level allows to dynamically adapt the distribution algorithm used to decide about the mapping. Both characteristics make AeDEn an extremely flexible system well-suited for the targeted environments.

The paper is organized as follows: next section explains to what extent distribution can improve the execution of applications in mobile environments and justifies the need for adaptive distribution algorithms. Our system, AeDEn, is detailed in Section 3 and some adaptive and dynamic distribution schemes are introduced in Section 3.2. Section 4 compares AeDEn system to existing approaches. Finally, Section 5 concludes and gives future research directions.

2. Applications Distribution over Mobile Environments: Interests and Problems

In this section, we will refer as *environment of a PIA* the set of *hardware and software resources* in the PIA's current cell (GSM, WLAN or ad-hoc network cell). Hardware resources refer to hardware elements of a PIA (CPU, memory, disk, bandwidth, etc) as well as the environment (scanner, fax, printer, other stations resources). Software resources are software elements at the system level (naming service, caching service, etc) or the application level (electronic commerce, electronic press, etc). An application is modeled as a set of communicating components. Each component knows the resources and the other components it needs.

PIA's mobility makes resources appearance/disappearance a common event in mobile environments. Resources disappearance may lead to abrupt termination of applications execution, or to execution in degraded mode. Meanwhile, new resources can appear and be made available to applications. Using distribution techniques, applications execution can be improved by exploiting environment resources. Improvements may concern:

– **User interaction quality**: user interaction quality is measured through the response time of the system or applications. This response time depends on mobile computer physical constraints such as memory size, available CPU, disk size or bandwidth. To improve the user interaction quality, distribution techniques may allow entities, that are directly in-

teracting with the user, to be placed on the portable device, and to distribute the others (non-interactive) over the environment. This results in (i) a decreasing utilization of resources of the portable device, (ii) a better use of environment resources allowing to optimize the execution time, and (iii) a way to cope with disconnections since computation or information retrieval can continue off-line and results can be transmitted upon reconnection.

– **Data quality**: data are made up of texts, images, videos, sounds, etc. These data can be transformed by operations such as compression, truncations, distillation of images, etc, for very different reasons (gain in response time, adaptiveness to screen size, etc). By taking advantage of the environment resources, the quality of the modified data may either be improved while maintaining the same response time or either be preserved while improving the response time.

– **Accessibility**: distribution allows applications to transparently use the environment resources by appropriate placement of application entities. This placement is dynamically readapted to new situations, and especially when environment resources appear or disappear.

Many distribution systems have been proposed to satisfy applications resources needs. The good performances they are ensuring are mainly based on load sharing and load balancing techniques according to a CPU criterion ([LIT 90], [ZHO 91], [FOL 94]). However, they do not take into account new criteria introduced by mobile environments: (i) movement criteria such as highly variable bandwidth, probability of disconnection and reconnection (previous approaches assume a permanent connection), predictive connection time, (ii) consumption criteria such as battery index, (iii) autonomous criteria such as the decision of which resources to share and when. Moreover, they do not allow to dynamically change the distribution algorithm itself. This is particularly important in a mobile environment as the best algorithm greatly depends on the changing location of the PIA.

For example, let us consider a distribution system fixing the distribution policy of entities belonging to an application initiated on a PIA. The distribution is based on information concerning the environment resources maintained by the system. If the PIA is connected through a wireless link to a fixed network, the information about the environment can be maintained by a fixed server. The PIA can directly address this server to know where to distribute the applications entities. But, if the PIA is part of an ad-hoc network, the distribution

system cannot rely on a fixed server. In that case, the better is to let the PIA maintain the information about its resources and interrogate its neighbours in order to decide about the layout of the entities. So, for a given PIA moving through different networks, the distribution system should dynamically adapt to the environment.

Providing a distribution algorithm well-suited to all execution conditions and applications constraints is not a realistic approach. Meanwhile, letting designers building a distribution system from scratch according to environment evolutions is an unacceptable solution. Moreover, taking into account environmental variations is even a more complex task as it requires designers to consider real-time changes on the distribution policy. Therefore, we aim at providing a generic distribution system called AeDEn that (i) is highly reusable in order to minimize the programming effort, (ii) is customizable to different applications needs, (iii) allows the integration of new distribution algorithms in order to consider environment evolutions such as new network protocols or new PIAs, and (iv) dynamically adapts to changes in the environment.

3. AeDEn : Adaptive Distribution Environment

Traditionally, a distribution algorithm is composed of three policies [ZHO 88]: *the Information Policy* which specifies the nature of informations needed for the election (CPU load, etc) and how they are collected, *the Election Policy* which elects processes that must be suspended, placed or migrated according to the collected informations, and *the Placement Policy* which chooses the station for each elected process.

In our system, these distribution policies are provided by different services which are detailed in the next section. Adaptive and dynamic schemes of these services are discussed in Section 3.2.

3.1. *Architecture*

Our distribution system AeDEn is a middleware layer made up of a set of independent services. Our service-based architecture allows us to extend a service without modifying other services and to use each service independently from the other ones. Figure 1 shows the

logical and physical architecture of our system. Four main services are available: the *Detection and Notification Service (DNS)*, the *Local Environment Management Service (LEMS)*, the *Environment Management Service (EMS)*, and the *Distribution Service (DS)*.

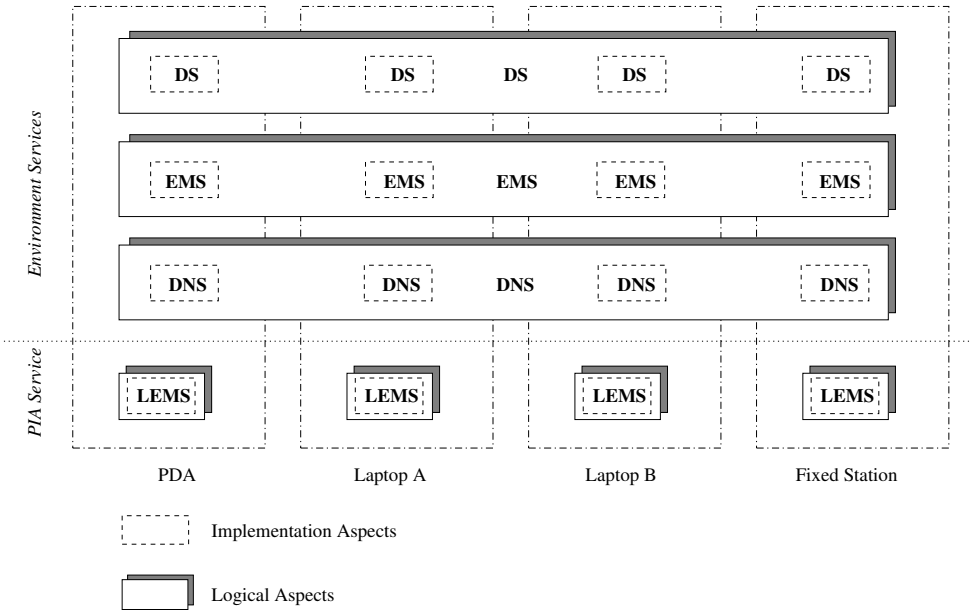

Figure 1. *AeDEn architecture – Architecture du système AeDEn*

A LEMS executes on each station. Each LEMS allows the local station[2] to manage its local resources according to its own constraints. The DNS, the EMS, and the DS are unique and global for the environment, but, for scalability reasons, they are physically implemented in a distributed way on each station. Therefore, the DNS, the EMS, and the DS on a station can be locally accessed and eventually cooperate with their corresponding services on other stations according to the algorithms they implement.

AeDEn services implement the policies of a traditional distribution algorithm. *The Election and Placement Policies* are part of the DS. *The Information Policy* is shared among the LEMS, the EMS, and the DNS. The LEMS implements a *Registration Policy* which is in charge of managing the local resources of the station. The EMS implements an *Environment Management Policy* in charge of managing the resources of the environment, and the

2. We refer to local station of a program the device on which the program executes.

DNS implements a *Variations Detection and Notification Policy* in charge of monitoring and notifying the concerned services of resources changes. The role of each service is described below while the adaptive and dynamic aspects are detailed in the next section.

Detection and Notification Service

The primary role of the DNS is to provide the LEMS and the EMS with local and environment changes detected on the underlying operating system or network layers.

The LEMS and the EMS register their resources interests in the DNS. These interests can be expressed either as simple conditions on one resource state or as complex conditions on several resource 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) or power consumption. Conditions can be expressed as acceptable values for an attribute. High-level monitors represent complex interests expressed by a boolean expression depending on low-level monitor conditions and other high-level monitor conditions. Once a condition is satisfied in a monitor, the LEMS or the EMS is notified of the change.

Local Environment Management Service

The primary role of the LEMS is to maintain a description of local station resources, and to register or unregister them in the environment according to the station sharing wishes. This service is implemented as a database of resources, and each resource is associated with a name, a state value, a set of offers, and the list of granted access components. It also receives notifications from the DNS when a change has been detected in monitored resources. It updates consequently their state, and notifies, if needed, the EMS.

Environment Management Service

The primary role of the EMS is to maintain a description of environment resources. This description results from the registration of station resources of each LEMS. In addition to the LEMS description, the EMS maintains for each resource the owner host, the current attached

host, the possibility to move, and the components which are using it. When an application references a resource, the EMS retrieves it and indicates its location and the components using it. It receives notifications from the LEMS concerning resource modifications, registering or unregistering.

Distribution Service

The DS is in charge of distributing application components. Each component contains a set of resource demands. The DS asks the EMS about the available resources in the environment, and, using these informations and the components demands, it computes the load indices and decides which components to distribute. Load indices calculation can rely on one or multi-criteria approaches. On top of the classical criteria (i.e. CPU, memory, etc), it can also take the new criteria introduced by mobile environments (i.e. movement criteria, consumption criteria, and/or autonomous criteria) into account. Then, the DS places the components accordingly and registers them in the EMS. Upon an environment change, the EMS is notified of the change and notifies the DS. The DS re-engages, if necessary, the distribution process: election, suspension, migration, placement.

3.2. *Adaptive and Dynamic Distribution*

First, let's examine the definitions[3] of two words that characterize our approach:

adaptive: *adj.* Readily capable of adapting or of being adapted.

dynamic: *adj.* Characterized by continuous change, activity, or progress.

The term adaptive can be applied to AeDEn as well as the applications it is managing. Indeed, these applications are adaptive because AeDEn is able to adapt the placement of their constituent components. The AeDEn system itself is also adaptive since it is readily capable of adapting the mechanisms of placement to applications and environment changes. Our system is also dynamic since it takes all these changes into account at run-time.

3. Source: The American Heritage® Dictionary of the English Language, Fourth Edition. Copyright © 2000 by Houghton Mifflin Company.

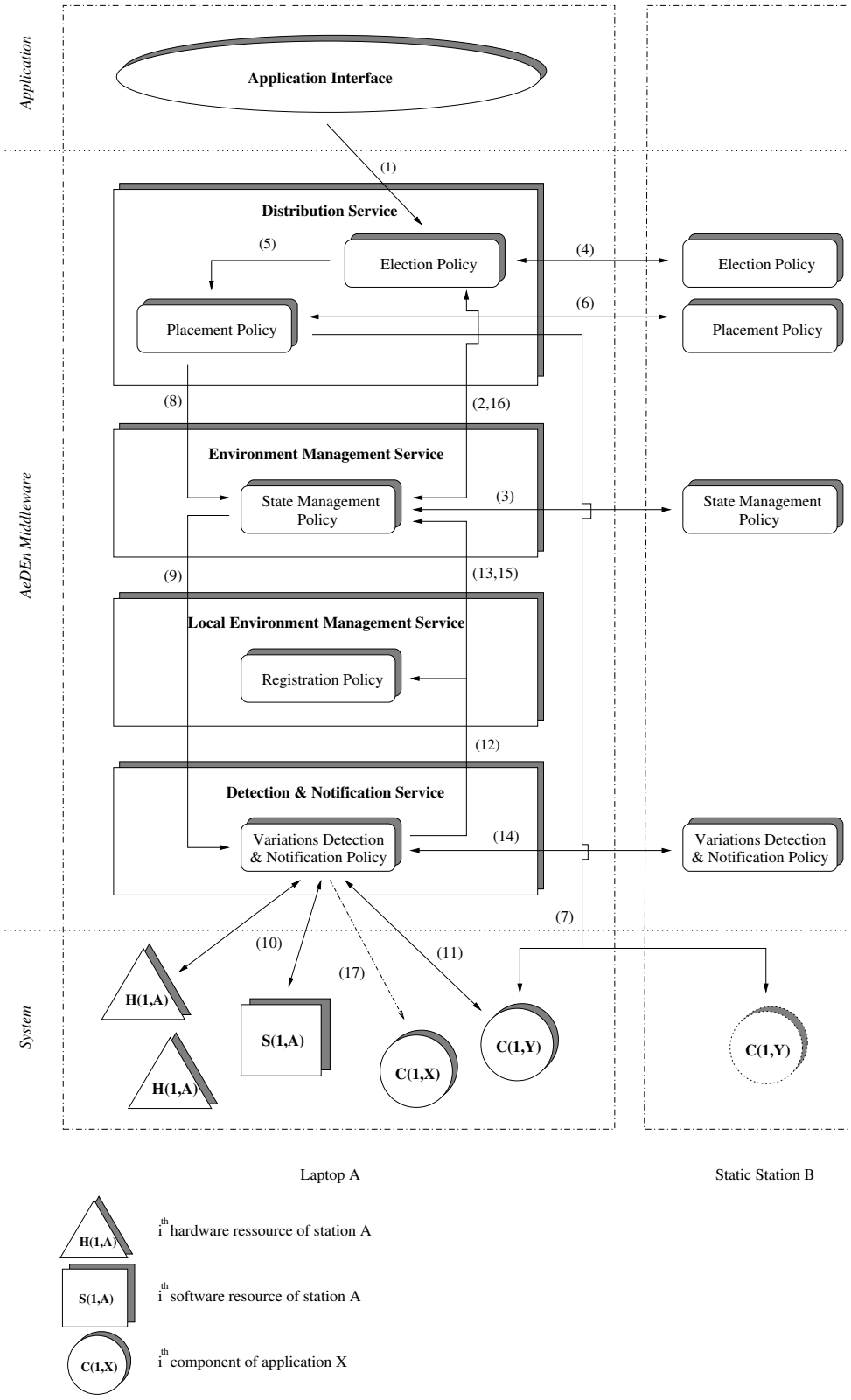

Figure 2. *AeDEn services interactions – Interactions entre les services d'AeDEn*

3.2.1. *Application-adaptive Distribution*

Applications interaction with AeDEn takes place at launching time. Figure 2 illustrates the steps performed by AeDEn at this time. Applications should provide the DS, and more concretely the Election Policy, with the set of components that can be distributed (1). Each component knows the resources (set of demands) and the other components it needs. The Election Policy interacts with the EMS in order to ask for resources states (2). As the EMS is implemented in a distributed way, the State Management Policy may interrogate its homologue on other stations in order to obtain the requested information (3). According to the components needs and to the resources states, the Election Policy calculates the load indices and determines which components are well-adapted for the distribution. This election may be realized in cooperation with the Election Policy of other stations (4). The chosen components are then submitted to the Placement Policy (5) which determines the most suitable stations for them and performs their transfer (7). The placement decision may also be taken in cooperation with other stations (6). Finally, the Placement Policy registers the components in the EMS (8) which activates the monitoring of these components (9).

3.2.2. *Environment-adaptive Distribution*

Changes in the environment are taken into account by AeDEn services at two different levels: the application and the distribution system itself. The former concerns the mapping of the application onto a set of stations. According to environment changes and to policies decisions, AeDEn services dynamically change the placement of applications components. The second level allows to dynamically change each policy of the distribution algorithm achieving the mapping.

3.2.2.1. Dynamically Adapting Placement

When the DNS detects a change in some of the environment resources (step 10 in Figure 2), or in components (11), two cases are possible. If it is a local resource, the DNS notifies the LEMS (12) which can decide, according to its Registration Policy, to propagate the change to the EMS (13). If it is an environment resource (14), it directly notifies the EMS which updates its information database (15) and notifies the DS if some of the distributed

components are involved[4] (16). Then, the Election Policy determines if these changes affect the placement. In this case, a new placement is performed (steps 2-9) which may imply the placement of new components, the suspension, and/or the migration of existing ones.

3.2.2.2. Dynamically Adapting Policies

As we said in Section 2, a particular algorithm can be more or less relevant depending on the application needs and constraints. In order to ensure the suitability of AeDEn in all cases, we introduce *adaptive policies* which change their algorithm as changes occur in the environment. Each adaptive policy has only one algorithm at time t but can dynamically change and have a different algorithm at time $t + 1$ (see Figure 3).

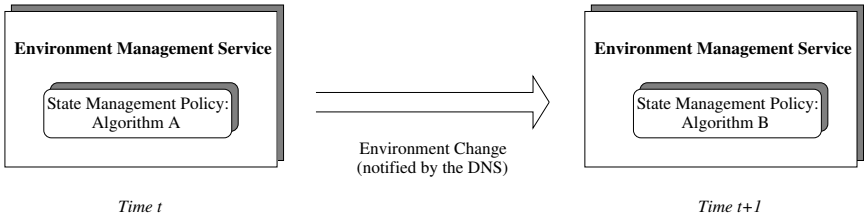

Figure 3. *State Management Adaptive Policy for the EMS – Exemple de politique adaptative de gestion de l'environnement*

The mechanism of switching from an algorithm to another is performed according to a *strategy*. The strategy is defined by an automaton where the states correspond to the algorithms, and the transitions are labelled with the environment changes conditions and the actions to perform when switching to the new state.

According to the considered policy, transition conditions can depend on local resources or environment resources states. For example, Figures 4, 5 & 6 show possible strategies for the Registration Policy of the LEMS, the State Management Policy of the EMS, and the Placement Policy of the DS, respectively. The Registration Policy strategy depends on a local resource state (battery) while the State Management Policy strategy and the Placement Policy strategy depend on several environment resources states (network topology and disconnection/reconnection probability).

4. Application components can also be designed to adapt themselves according to environment changes. In this case, they are directly notified by the DNS (17).

According to the considered policy, algorithms may involve the mobile computer (local) or the environment (globally distributed). For example, the Registration Policy algorithms describe the mobile own decisions about the sharing of its resources while the State Management Policy and Placement Policy algorithms describe environment management choices. Algorithms can also be adapted to the mobility or not. The Placement Policy algorithms chose to place on the mobile or not while the Registration Policy and the State Management Policy algorithms don't use any mobile criteria.

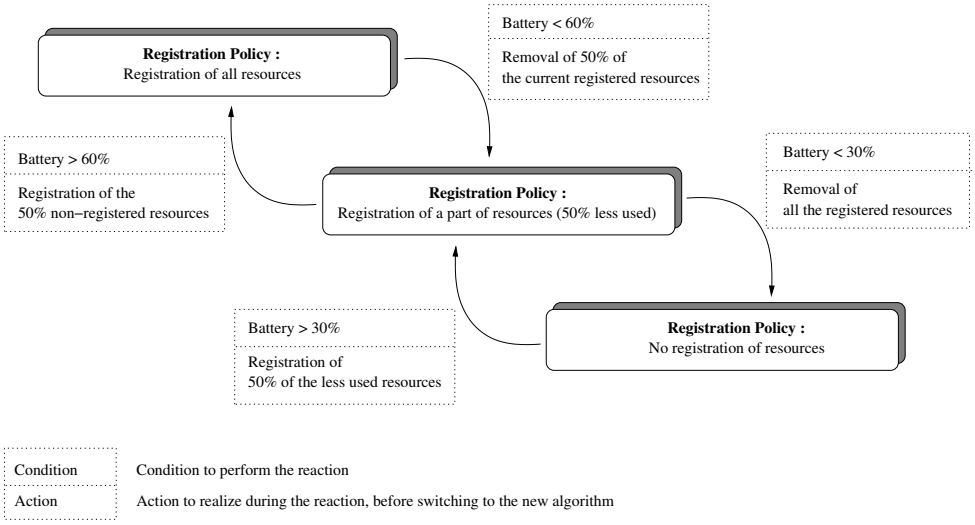

Figure 4. *Registration Policy strategy example – Exemple de stratégie pour la politique d'enregistrement des ressources*

3.3. *AeDEn Implementation*

To implement AeDEn, we used the MolèNE framework that we have previously developed. Next Section details one aspect of this framework, the MolèNE component, that we particularly used to implement our policies.

3.3.1. *MolèNE Component*

In order to cope with variations in resources availability to offer a sustained level of QoS, we have designed and implemented MolèNE [SEG 00], an object-oriented framework that considers applications as sets of components and provides a *dynamic adaptation frame-*

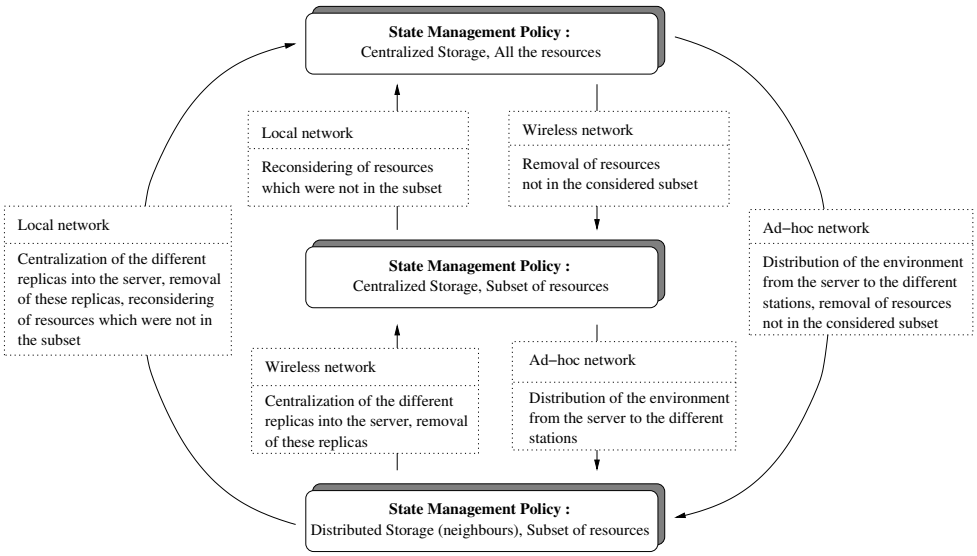

Figure 5. *State Management Policy strategy example – Exemple de stratégie pour la politique de gestion de l'environnement*

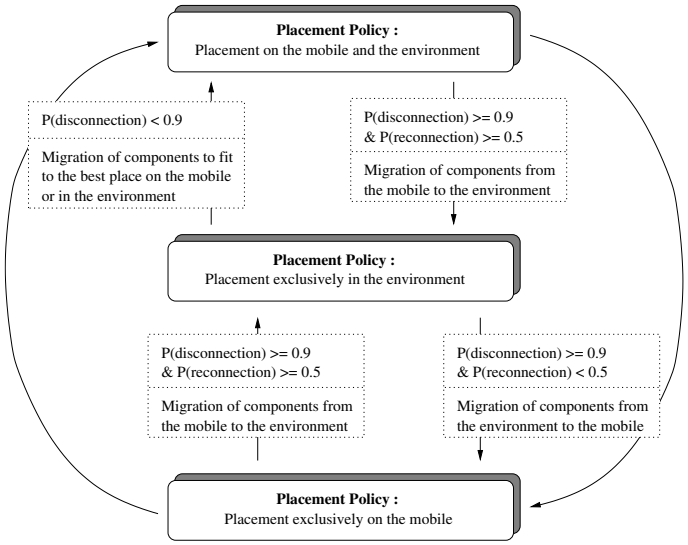

Figure 6. *Placement Policy strategy example – Exemple de stratégie pour la politique de placement*

work that allows each component to change its behavior at run-time depending on resources availability. The dynamic adaptation framework is based on an *Adaptation Strategy*, which decides when to proceed to an implementation change and a *Controller* which executes the change. Figure 7 shows the set of classes and their relations that constitutes a MolèNE component.

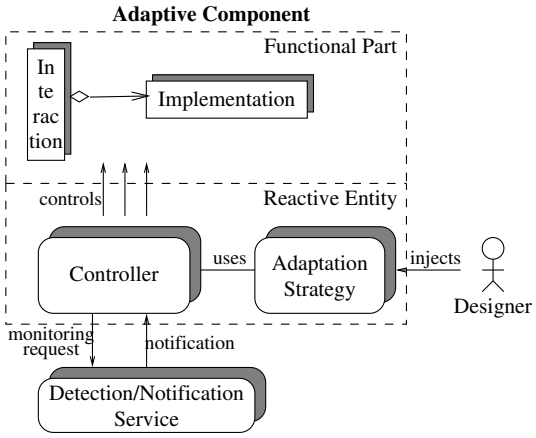

Figure 7. *MolèNE component structure – Structure d'un composant MolèNE*

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 on execution conditions and to the *Adaptation Strategy* designed for the component decides whether or not to switch to another *Implementation*. If a switch is decided, the current *Implementation* is replaced by another code, while keeping the necessary informations for the new *Implementation* and the links with the *Interaction* part. This mechanism allows to dynamically include new *Implementation* classes and new *Adaptation Strategy* classes that were not available at the component creation.

Each of AeDEn's *adaptive policies* have been implemented as a MolèNE component. The different algorithms have been implemented as *Implementation* classes and the different strategies of algorithms switching as *Adaptation Strategy* classes.

3.3.2. *Results*

To test our system, we have implemented an application commonly used in mobile environments: a browser transforming HTML pages to the display possibilities of the considered PIA. We break up this application into three components: a *Download* component (DO in Figure 8) in charge of downloading HTML pages and all associated files, a *Transformation* component (TR in Figure 8) in charge of transforming these pages to fit to the display possibilities of the PIA, and a *Display* component (DI in Figure 8) in charge of displaying the transformed pages. In our test, the DO component downloads an HTML page along with its pictures, and the TR component replaces them by the text of the corresponding `ALT` attribute of the `IMG` tag[5]. Each of these components have particular needs: the DO component requires an Internet access, the TR component requires a minimum of computing power and the DI component requires a screen to display the page.

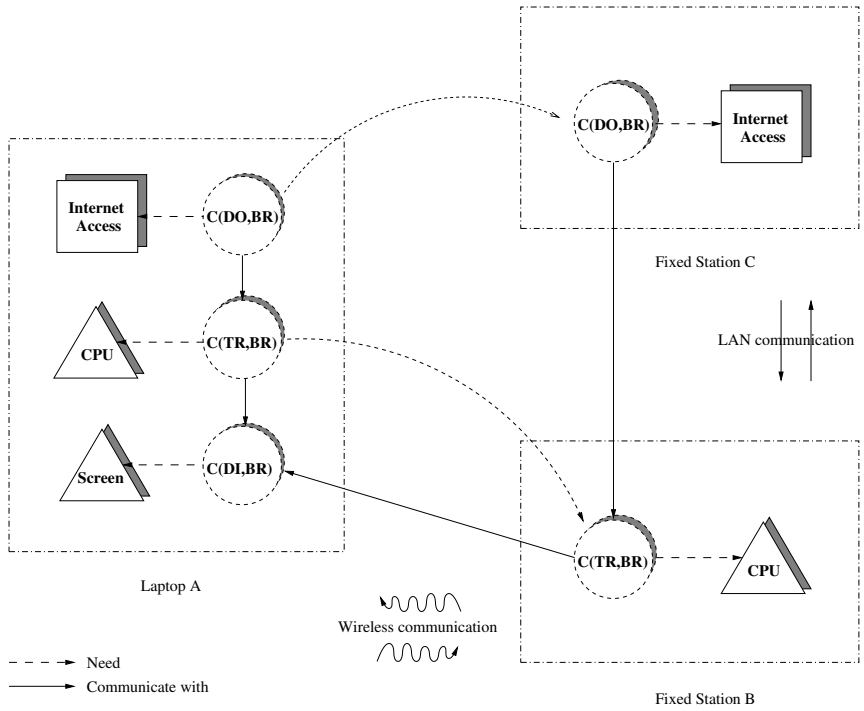

Figure 8. *Browser components distribution – Placement des composants du navigateur*

5. We can easily imagine more complex transformations such as the degradation of images, etc.

Three stations constitute the execution environment considered for this application: a laptop computer (A) offering a screen, some computing power and an Internet access, a fixed station (B) offering computing power, and a fixed station (C) offering an Internet access. In our test, the laptop (A) is an IBM ThinkPad with a processor at 166MHz, connected to the local network through an Ethernet link at 10Mb/s when the laptop is static, and through a WaveLAN link at 1.6Mb/s when the laptop is mobile. These connections allow direct access to Internet. The fixed station (B) is an Ultra Sparc 60 with two processors at 360MHz, and the fixed station (C) provides an Internet access through a proxy server on the same Ethernet LAN than the two fixed stations.

To test the benefits of the distribution and particularly of the adaptive policies, we have simultaneously tested three adaptive policies: the Variations Detection and Notification Policy, the Registration Policy, and the Placement Policy. The Variations Detection and Notification Policy strategy has two algorithms: an immediate algorithm (IVDNP) which immediately notifies when a variation occurs and a periodic algorithm (PVDNP) which periodically notifies changes. The condition making the algorithm to change is the bandwidth variation from the Ethernet link to the WaveLAN one. The Registration Policy strategy has two algorithms: an algorithm of registration of the local resources of the laptop (i.e. CPU and Internet access) in the distribution environment (LRRP), and an algorithm which registers no resource (NRRP). The condition of switching is the remaining battery lifetime. The Placement Policy strategy also has two algorithms: an algorithm of placement which only uses the local resources (LPP), and an algorithm of placement which uses all the environment resources (EPP). We switch to the EPP algorithm when no more local resources are available.

First, when we launch the browser, the Registration Policy is implemented by the LRRP algorithm and the Placement Policy by the LPP one. So the three components of the browser are placed on the laptop (A) (see Figure 8). Then, when the remaining battery lifetime reaches a given threshold, the Registration Policy switches to NRRP algorithm and remove its resources from the distribution environment. Also, the Placement Policy switches to EPP algorithm and chooses to place the DO component to fixed station (C) and the TR component to fixed station (B). At the same time, when the laptop is static and connected

through the Ethernet link, the Variations Detection and Notification Policy is implemented by the IVDNP algorithm, and when the laptop moves and use the WaveLAN link, the Variations Detection and Notification Policy is implemented by the PVDNP algorithm.

From this experiment, we analyze the impact of distribution and adaptations in terms of application performances improvements as well as laptop resources consumption.

Application performances

Table 1 shows the execution time of the DO and TR components according to the station where they were executed. This time has been calculated as the average of ten requests on popular web sites[6]. The download execution time is decreased by 50-60% when DO component executes on station (C) because of the overcome of the wireless link bottleneck and the use of an Internet access with a proxy. The transformation execution time is decreased by 40-50% when the TR component executes on station (B) because of the use of a powerful processor.

	DO (A) (ms)	DO (C) (ms)	Speed Up %	TR (A) (ms)	TR (B) (ms)	Speed Up %
1 - Microsoft	3648	1722	52.8	966	564	41.7
2 - AOL	6567	2561	61.0	1841	1064	42.2
3 - Yahoo	2677	1283	52.1	720	414	42.5
4 - Lycos	4930	2863	41.9	1206	694	42.4
...	...	...	...	...	...	...
8 - NBC	5548	2613	52.9	1152	684	40.6
11 - Disney	3813	1853	51.4	927	541	41.7
15 - Real	5232	2378	54.5	1384	810	41.5
17 - Fortune City	6041	2417	60.0	1782	1018	42.9

Table 1. *Browser execution time according to the execution station – Mesures du temps d'exécution des composants du navigateur selon la station d'exécution*

As the Table 2 shows, the amount of data transfered through the wireless link is also reduced since only pages resulting from the transformation are sent to the DI component. Obviously, this result depends on the transformation, and/or the size of pictures embedded in the HTML page. For most tested sites, the amount of transfered data is reduced by 40-60%, but, in picture-rich sites, it can reach up to 70-80%.

6. Extract from the Global Top 50 Web published by Jupiter Media Metrix in April 2001.

	DO (A) TR (A) Html + Img (Kbyte)	DO (C) TR (B) Html (Kbyte)	%
1 - Microsoft	22.9 + 17.3 = 40.2	20.1	50.0
2 - AOL	43.6 + 27.0 = 70.6	37.6	46.7
3 - Yahoo	16.0 + 12.0 = 28.0	15.6	44.3
4 - Lycos	26.7 + 13.5 = 40.2	25.6	36.3
...	...	...	...
8 - NBC	29.1 + 66.4 = 95.5	23.2	75.7
11 - Disney	22.6 + 78.6 = 101.2	18.3	81.9
15 - Real	32.6 + 71.8 = 104.4	29.5	71.7
17 - Fortune City	41.8 + 90.8 = 132.6	35.8	73.0

Table 2. *Data transfered through the wireless link – Mesures des données transférées sur le lien sans fil*

We also measured the impact of migration on application performances. For the experiment, we used Java serialization and we measured the introduced overhead. The migration is splitted into component transfer time and component restoring time. Table 3 shows both times for a DO and a TR component and takes into account the amount of data they are processing: small amount of data $\searrow$ (Yahoo HTML page: 28Kb) and important amount of data $\nearrow$ (Fortune City HTML page: 132Kb). As we can see, the restoring time is constant while the transfer time greatly depends on data which have to be serialized. These measurements allowed us to calculate the number of HTML requests necessary to counterbalance the migration cost ($DO(A) \times Nb_{request} > DO(C) \times Nb_{request} + Transfer + Restoring$). The result is shown in the last column of Table 3: the requests number varies according to considered component, 4 requests for the DO component, and the amount of data, from 7 to 10 requests for the TR component.

	Transfer time (ms)	Restoring time (ms)	Migration balancing (> number of requests)
DO (A $\rightarrow$ C) + Data $\searrow$	5437	165	4
DO (A $\rightarrow$ C) + Data $\nearrow$	12920	178	3.6
TR (A $\rightarrow$ B) + Data $\searrow$	2616	213	9.3
TR (A $\rightarrow$ B) + Data $\nearrow$	4690	236	6.4

Table 3. *Impact of migration on browser performances – Mesures de l'impact de la migration sur les performances du navigateur*

These tables show the interest in distributing the application components according to the best resources available at any time. As explains previously, this implies to have a dynamic knowledge of the environment.

From these results, we can conclude that our AeDEn system improves the user interaction quality (see Section 2) as the application execution time decreases when executed in a more powerful environment than a PIA. Moreover, the amount of data transfered through the wireless link is also reduced since only the results are sent to the PIA. In our example, the data quality remains the same as we use the same transformation.

Laptop resources

Battery

In our experiment, we also measured the battery use. Figure 9 shows battery consumption according to the laptop mode (suspended, stand by, running) and to the resources in use (screen, processor, memory, network). When the browser is executing on the laptop, the curve representing the execution of our application has the same slope, -1.5, than the curves representing battery consumption when several resources are in use. When the remaining battery lifetime reaches 40%, the Registration Policy switches to the NRRP algorithm and decides to remove the shared resources. The Placement Policy also switches to the EPP algorithm and decides to migrate the DO and TR components to the environment. This change is reflected in the second part of the curve. Its slope, -0.84, is just about the same than the one of the curve with only the system and the screen active (-0.94). This results in a battery lifetime increase of 30% (88mn instead of 64mn).

AeDEn improves accessibility (see Section 2) as the distribution allows transparent access to the environment resources. The benefit is a reduced use of PIA resources resulting in an increased battery lifetime.

Bandwidth

Bandwidth was also tested in our experiment. Indeed, we stressed the communication link by simulating a variation every 100 ms and by sending their detection to the EMS that is globally distributed. Overhead introduced by these communications showed the interest of

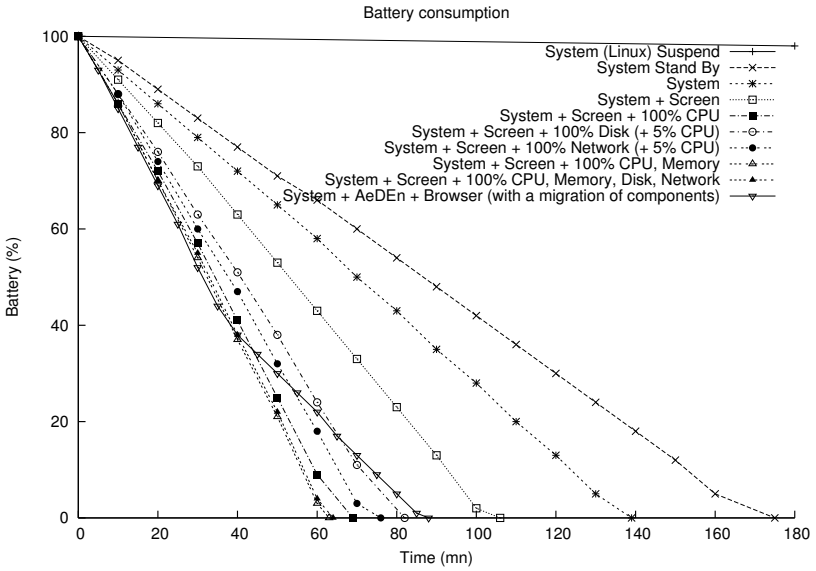

Figure 9. *Battery consumption according to the use – Décharge de la batterie selon son utilisation*

having different algorithms for the Variations Detection and Notification Policy depending on the network the laptop computer is connected to.

We considered a static laptop using an Ethernet link. When the IVDNP algorithm was used, we observed an overhead of 400Kb which corresponds to 4% of total bandwidth. This overhead is low enough to adopt the IVDNP algorithm in this case.

We also consider the laptop being connected to a WaveLAN network. For this environment, we measured the bandwidth used by each algorithm. Results are shown in Figure[7] 10. From $t = 0$ to $t = 90000ms$, the Variations Detection and Notification Policy is implemented by the IVDNP algorithm, then it switches to the PVDNP one parametrized with a period of 5000ms.

7. In the figure, the theoretical maximal bandwidth is exceeded in some points. This can be explained by the fact that the bandwidth saturation produces a delay which prevents the control packets used to calculate the bandwidth to be taken into account at the right period. Some packets sent during the interval t are taken into account during the interval $t + 1$.

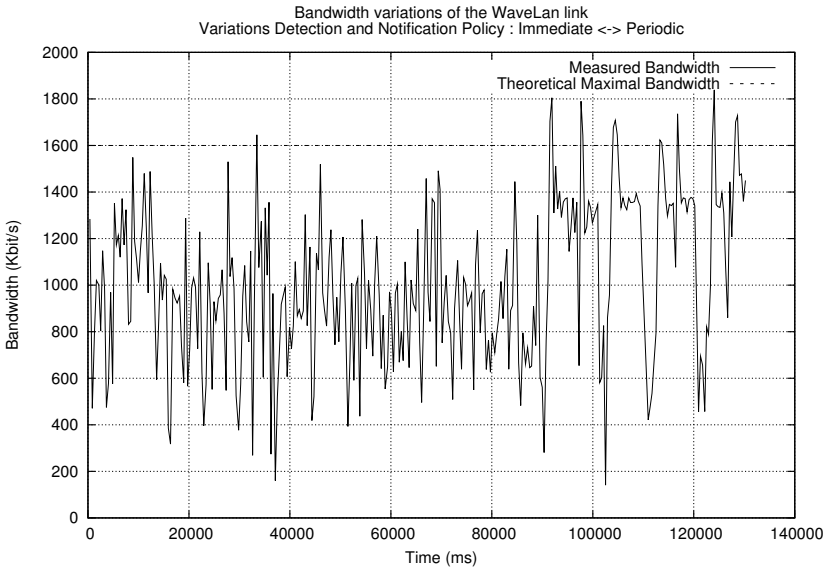

Figure 10. *Bandwidth use according to the Variation and Detection algorithm used – Bande passante mesurée selon l'algorithme de détection utilisé*

In order to determine the overhead of each algorithm, we used the curve in Figure 10 and we calculated the probability fonction of the available bandwidth at time t (Figures 11 and 12). We then considered two intervals so that the probability of the available bandwidth being in them is the same when using IVDNP and PVDNP algorithms. These intervals are [800,1000] and [1200,1400] :

$$P_{IVDNP}(800 \leq bw < 1000) = \sum_{X=800}^{X=1000} P_{IVDNP}(X) = 0.317 \quad [1]$$

$$P_{IVDNP}(1200 \leq bw < 1400) = \sum_{X=1200}^{X=1400} P_{IVDNP}(X) = 0.085 \quad [2]$$

$$P_{PVDNP}(800 \leq bw < 1000) = \sum_{X=800}^{X=1000} P_{PVDNP}(X) = 0.104 \quad [3]$$

$$P_{PVDNP}(1200 \leq bw < 1400) = \sum_{X=1200}^{X=1400} P_{PVDNP}(X) = 0.279 \quad [4]$$

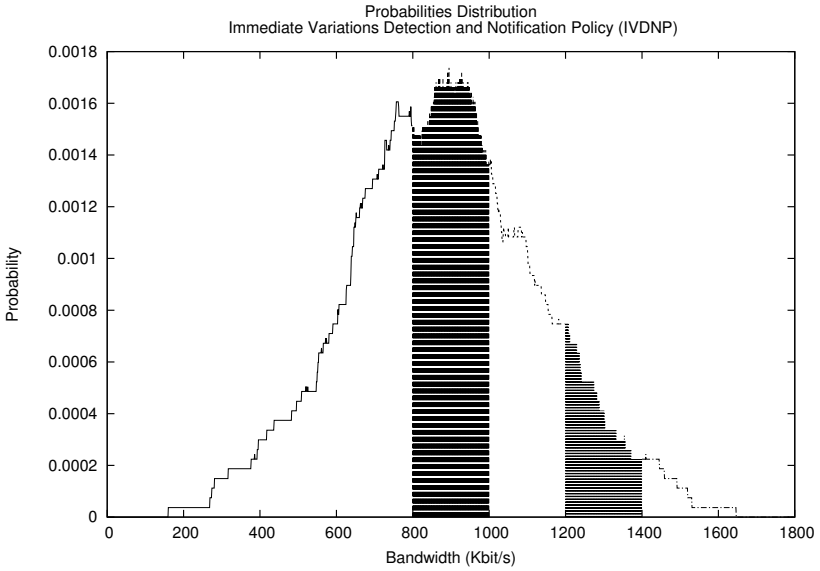

Figure 11. *Probabilities distribution of the bandwidth when the immediate algorithm is used – Courbe de distribution des probabilités dans le cas de l'algorithme immédiat*

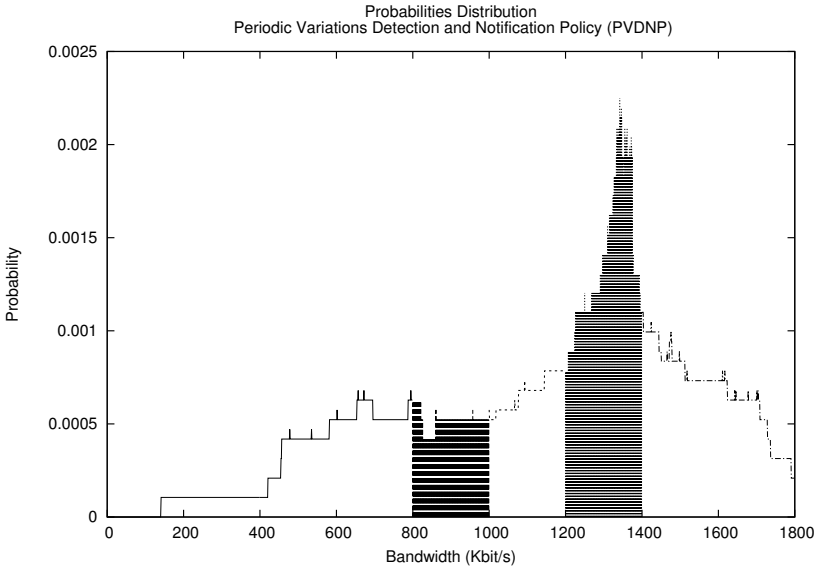

Figure 12. *Probabilities distribution of the bandwidth when the periodic algorithm is used – Courbe de distribution des probabilités dans le cas de l'algorithme périodique*

For the same bandwidth probability (30%), the IVDNP algorithm introduces an overhead of 600 - 800Kb which corresponds to 37.5 - 50% of the maximal bandwidth of the WaveLAN link while the PVDNP algorithm introduces an overhead of 200 - 400Kb which corresponds to 12.5 - 25% of the maximal bandwidth.

We were also interested on studying the way both algorithms react to bandwidth variations. We, thus, calculated the mean and the standard deviation of the bandwidth probability fonction for them:

$$\mu_{IVDNP}(800 \leq bw < 1000) = \frac{\sum_{X=800}^{X=1000} X \cdot P_{IVDNP}(X)}{\sum_{X=800}^{X=1000} P_{IVDNP}(X)} = 899 \quad [5]$$

$$\mu_{PVDNP}(1200 \leq bw < 1400) = \frac{\sum_{X=1200}^{X=1400} X \cdot P_{PVDNP}(X)}{\sum_{X=1200}^{X=1400} P_{PVDNP}(X)} = 1311 \quad [6]$$

$$\sigma_{IVDNP}(800 \leq bw < 1000) = \sqrt{\frac{\sum_{X=800}^{X=1000} P_{IVDNP}(X) \cdot (X - \mu_{IVDNP})^2}{\sum_{X=800}^{X=1000} P_{IVDNP}(X)}} = 59 \quad [7]$$

$$\sigma_{PVDNP}(1200 \leq bw < 1400) = \sqrt{\frac{\sum_{X=1200}^{X=1400} P_{PVDNP}(X) \cdot (X - \mu_{PVDNP})^2}{\sum_{X=1200}^{X=1400} P_{PVDNP}(X)}} = 51 \quad [8]$$

The PVDNP algorithm probability distribution is more compact around the mean than the IVDNP one ($\sigma_{PVDNP} < \sigma_{IVDNP}$). For the PVDNP algorithm, the probability that the available bandwidth is in the intervals [1000,1200] or [1400,1600] decreases quickly ($\sim$13-16%), while, for the IVDNP algorithm, the probability for intervals [600,800] and [1000,1200] remains important ($\sim$21-24%). From these results, we can conclude that the PVDNP algorithm is less sensitive to variations making it better suited to a WaveLAN environment.

In order to built a Variations Detection and Notification Policy that does not introduce a too costly bandwidth overhead and reacts fairly well to existing variations of the link, the algorithm implementing it should dynamically change. When using an Ethernet link, the IVDNP algorithm should be used. When a WaveLAN wireless connection is available, the PVDNP algorithm is a better suited approach. In another environments such as ad-hoc networks, the PVDNP algorithm is not really suitable [VAH 00], but well-adapted algorithms can be implemented [YU 00] and dynamically used in our adaptive policies.

4. Related Work

4.1. *Adaptation and Mobile Computing*

Many architectures or systems supporting adaptiveness have been introduced in the area of mobile computing. These works particularly deal with the adaptation to the bandwidth variability and the PIA's resources poorness.

Most of these works try to overcome the disconnection problem and the bandwidth bottleneck. We distinguish between two approaches: *application-transparent* and *application-aware approaches* [SAT 96]. 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 [PER 98] is a protocol which allows mobile devices to keep their own address in spite of mobility. The Coda File System [KIS 92] offers the user the possibility to work while disconnected, masking disconnection periods by caching and hoarding techniques. Nevertheless, completely masking mobile aspects is not possible. Indeed, since these approaches ignore totally or partially the application behavior, they are not able to handle all variations.

Application-aware approaches partially solve this 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 [KAT 96] 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 applica-

tions such as an adaptive QoS support, low latency handoff, user tracking, etc. The MOST project [FRI 96] developed a protocol called QEX (Quality-of-service driven remote EXecution) which provide feedback to applications when changes to the communications QoS occurs. Ensemble [REN 97] is a network protocol architecture based on protocol stack. The system is build from simple *micro-protocol* modules which can be stacked in a variety of ways to meet the communication demands of its applications. Changes on bandwidth availability are notified to applications which can demand the reconfiguration of the *micro-protocols*. Odyssey [NOB 97] extends a file system interface to allow applications to specify a data access strategy for a level of resources availability. When this level changes, the application is notified and must decide about a new policy.

Few approaches aim at overcoming the poorness of PIAs resources. The solution commonly adopted is to use external resources with an appropriate distributed programming model. One of them is the Client/Proxy/Server model with Remote Procedure Calls or Remote Evaluations. In this model, the proxy and the server are in the fixed network. The proxy uses resources of the fixed station where it is executed to provide network or application facilities to the mobile user such as efficient protocols (Mowgli [LIL 96]), storage of results while disconnections, caching (WebExpress [HOU 96]), filters such as compression, degradation, distillation, display media conversion (WAP [Wir99]). Rover [JOS 97], for example, provides applications with relocatable dynamic objects (RDOs) and queued RPC. The former can be replicated or migrated from the mobile computer to a fixed station. The latter allows the non-blocking communication with remote RDOs in the disconnected mode of a mobile computer.

The Mobile Agent model is also used to design applications in mobile environments [GRA 96]. 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 computing, informations gathering and filtering to the mobile agent on the fixed network [POR 98, BAN 99]. Only the resulting data are transmitted to the mobile client. For example, Alycta [DEL 98] 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.

Distribution of proxies can also be used to increase the number of available resources. For example, distribution techniques are used in Daedalus [FOX 98] 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: processes to distribute only concern the proxies which are 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.

4.2. *Adaptation and Distributed Computing*

Many distributed systems have been proposed to satisfy the important need of resources. Early approaches aimed at transparently sharing the resources (CPU, memory and disk) of a local network of workstations (NoW system [AND 95]). Many algorithms [CAS 88, BER 96], ensuring good performances, load-balancing and stability, have been experimented in different systems. By using preemption and migration techniques, some of these systems take the execution conditions into account to adapt their placement (Sprite [DOU 91], Condor [LIT 90], GatoStar [FOL 94]). By using replication techniques, other systems take the execution conditions into account to adapt the number of distributed processes (for fault-tolerance reasons) (PM^2 [NAM 96], Stardust [CAB 96]). In these systems, performance is mainly based on load sharing according to the CPU criterion.

To provide the best adaptation with the QoS needed by applications, new criteria have been added in multi-criteria approaches [ZHO 91, FOL 92]. 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 [TAL 94]. To support the network infrastructure in the best possible way, some of these algorithms are distributed such as the *distributed QoS adapters* in [CAM 96]. Moreover, the QoS can be improved by combining these criteria with the end-station load in [BOM 98] or local objects information [SIL 01].

The object-oriented programming methodology is increasingly recognized for structuring large, extensible, and flexible software. Recent approaches use these paradigms to ensure the scalability of distributed applications. SOS [SHA 89] is an object-oriented operating system based on fragmented objects. Each object consists of a set of *fragments*, a client interface, a group interface, and a *connective* object [MAK 94]. The system does not include automatic migration mechanisms, but customized policies for communication minimization can be specified in the connective object by defining communication protocols between fragments.

Legion [GRI 97] and Globe [STE 99] are meta-systems which mainly aim at supporting wide-area distributed computing. Rather than implementing system-wide policies, Legion provides a framework through which programmers can supply customized object-placement policies. In Globe, objects are similar to the fragmented objects in SOS and location transparency is provided by the system. Each distributed objects is composed of several subobjects (semantics, communication, replication, control) which can implement different algorithms, but deciding on the placement or migration of an object is currently left to applications [JAN 01].

5. Conclusion

In this paper, we have presented AeDEn, a system which provides adaptive and dynamic mechanisms to distribute applications in an extremely varying environment. This distribution allows to overcome the poorness of available resources on portable devices by using resources of the environment. It also allows to reduce variability effects by executing applications components on stations less sensitive to variations.

In such highly varying environments, applications should be distributed in a dynamic way to take environment variations such as resources arrival or removal into account. Moreover, distribution services themselves should be adaptive. As results demonstrate, the distribution policy and the mechanisms to implement it should be well-adapted to the current available resources, for example not to introduce a too costly overhead or to use too much bandwidth. To achieve this property, our AeDEn system allows to dynamically change the

parameters of the current policy or to switch to another one. These different aspects have been demonstrated in the prototype developed by using the MolèNE components.

Future work will improve AeDEn prototype along different axes. Among these ones, an interesting aspect is the coordination of adaptive policies, and more generally adaptive components. Indeed, changing the implementation of a component can affect the components which are dependent on. These ones can consequently also change and affect other components. This may lead to inconsistency and instability problems that should be solved by contracts negotiation. We are also currently working on a more precise model of an adaptive component allowing to better specify its different elements such as the adaptation strategy, the interaction events, the resources monitors and the detection conditions.

6. References

[AND 95] ANDERSON (T. E.), CULLER (D. E.), PATTERSON (D. A.), THE NOW TEAM, A Case for NOW (Networks of Workstations), *IEEE Micro*, (1995), **15**, n° 1, pp. 54–64.

[BAN 99] BANDYOPADHYAY (S.), PAUL (K.), Evaluating the performance of mobile agent-based message communication among mobile hosts in large ad hoc wireless network, *Proceedings of the 2nd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, (August 1999), pp. 69–73.

[BER 96] BERNARD (G.), FOLLIOT (B.), Caractéristiques Générales du Placement Dynamique : Synthèse et Problématique, *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).

[BOM 98] BOM (J.), MARQUES (P.), CORREIA (M.), PINTO (P.), QoS Control: an Application Integrated Framework, *Proceedings of the IEEE Internationale Conference on ATM*, Colmar, France, (June 1998).

[CAB 96] CABILLIC (G.), PUAUT (I.), Dealing with Heterogeneity in Stardust: An Environment for Parallel Programming on Networks of Heterogeneous Workstations, *Proceedings of 2nd International Euro-Par Conference (Euro-Par'96)*, Lyon, France, (August 1996), pp. 114–119.

[CAM 96] CAMPBELL (A.), COULSON (G.), A QoS Adaptive Transport System: Design, Implementation and Experience, *Proccedings of the 4th ACM International Conference on Multime-*

dia'96, Boston, USA, (November 1996), pp. 117–127.

[CAS 88] CASAVANT (T.), KUHL (J.), A Taxonomy of Scheduling in General Purpose Distributed Computing Systems, *IEEE Transactions of Software Engineering*, (1988), **14**, n° 2.

[DEL 98] DELORD (X.), PERRET (S.), DUDA (A.), Efficient Mobile Access to the WWW over GSM, *Proceedings of the 8th ACM SIGOPS European Workshop Support for Composing Distributed Applications*, Sintra, Portugal, (September 1998), pp. 1–6.

[DOU 91] DOUGLIS (F.), OUSTERHOUT (J.), Transparent Process Migration: Design Alternatives and the Sprite Implementation, *Software-Practice and Experience*, (1991), **21**, n° 8, pp. 757–785.

[FOL 92] FOLLIOT (B.), Méthodes et Outils de Partage de Charge pour la Conception et la Mise en Œuvre d'Applications dans les Systèmes Réparties Hétérogènes, *PhD thesis*, MASI laboratory, Paris 6 University, France, (1992).

[FOL 94] FOLLIOT (B.), SENS (P.), GATOSTAR: A Fault Tolerant Load Sharing Facility for Parallel Applications, ECHTLE (K.), HAMMER (D.), POWELL (D.), Eds., *Proceedings of the 1st European Dependable Computing Conference*, **852** of *Lecture Notes in Computer Science*, Springer-Verlag, (October 1994).

[FOX 98] FOX (A.), GRIBBLE (S. D.), CHAWATHE (Y.), BREWER (E. A.), Adapting to Network and Client Variation Using Infrastructural Proxies: Lessons and Perspectives, *IEEE Personal Communications (invited submission)*, (1998), **5**, n° 4, pp. 10–19.

[FRI 96] FRIDAY (A.), BLAIR (G.), CHEVERST (K.), DAVIES (N.), Extensions to ANSAware for Advanced Mobile Applications, *Proceedings of the 1st International Conference on Distributed Platforms*, Dresden, Germany, (February 1996).

[GRA 96] GRAY (R.), KOTZ (D.), NOG (S.), RUS (D.), CYBENKO (G.), Mobile agents for mobile computing, *Technical Report* n° **PCS-TR96-285**, (May 1996), Department of Computer Science, Dartmouth College.

[GRI 97] GRIMSHAW (A. S.), WULF (W. A.), The Legion Vision of a Worldwide Virtual Computer, *Communication of the ACM*, (1997), **40**, n° 1, pp. 39–45.

[HOU 96] HOUSEL (B.), LINDQUIST (D.), Webexpress: A system for optimizing web browsing in a wireless environment, *Proceedings of the 2nd Annual International Conference on Mobile Computing and Networking (MobiCom'96)*, Rye, New York, USA, (November 1996), pp. 108–116.

[JAN 01] JANSEN (M.), KLAVER (E.), VERKAIK (P.), VAN STEEN (M.), TANENBAUM (A.), Encapsulating Distribution in Remote Objects, *Information and Software Technology*, (2001), **43**,

n° 6, pp. 353–363.

[JON 99] JONES (A.), Mobile Computing to Go, *IEEE Concurrency*, (1999), **7**, n° 2, pp. 20–23.

[JOS 97] JOSEPH (A. D.), TAUBER (J. A.), KAASHOEK (M. F.), Mobile Computing with the Rover Toolkit, *IEEE Transactions on Computers: Special issue on Mobile Computing*, (1997), **46**, n° 3, pp. 337–352.

[KAT 96] KATZ (R. H.), BREWER (E. A.), AMIR (E.), BALAKRISHNAN (H.), FOX (A.), GRIBBLE (S.), HODES (T. D.), JIANG (D.), NGUYEN (G. T.), PADMANABHAN (V. N.), STEMM (M.), The Bay Area Research Wireless Access Network (BARWAN), *Proceedings of the 41st IEEE Computer Society International Conference: Technologies for the Information Superhighway (COMPCON'96)*, Santa Clara, California, USA, (February 1996), pp. 15–20.

[KIS 92] KISTLER (J.), SATYANARAYANAN (M.), Disconnected Operation in the Coda File System, *ACM Transactions On Computer Systems*, (1992), **10**, n° 1, pp. 3–25.

[LIL 96] LILJEBERG (M.), HELIN (H.), KOJO (M.), RAATIKAINEN (K.), Enhanced Services for World-Wide Web in Mobile WAN Environment, *Proceedings of the IEEE Global Internet 1996 Conference (revised version)*, London, England, (November 1996).

[LIT 90] LITZKOW (M. J.), LIVNY (M.), Experience With The Condor Distributed Batch System, *Proceedings of the IEEE Workshop on Experimental Distributed Systems*, Huntsville, Alabama, USA, (October 1990), pp. 97–101.

[MAK 94] MAKPANGOU (M.), GOURHANT (Y.), JEAN-PIERRE (L. N.), MARC (S.), Fragmented Objects for Distributed Abstractions, CASAVANT (T. L.), SINGHAL (M.), Eds., *Readings in Distributed Computing Systems*, pp. 170–186, IEEE Computer Society Press, (July 1994).

[NAM 96] NAMYST (R.), MÉHAUT (J.-F.), PM^2: Parallel Multithreaded Machine. A Computing Environment for Distributed Architectures, D'HOLLANDER (E. H.), JOUBERT (G. R.), PETERS (F. J.), TRYSTRAM (D.), Eds., *Parallel Computing: State-of-the-Art and Perspectives, Proceedings of the Conference ParCo'95, 19-22 September 1995, Ghent, Belgium*, **11** of *Advances in Parallel Computing*, Amsterdam, (February 1996), Elsevier, North-Holland, pp. 279–285.

[NOB 97] NOBLE (B. D.), SATYANARAYANAN (M.), TILTON (J.), FLINN (J.), WALKER (K.), Agile Application-Aware Adaptation for Mobility, *Proceedings of the 16th ACM Symposium on Operating Systems Principles (SOSP'16)*, Saint-Malo, France, (October 1997).

[PER 98] PERKINS (C. E.), Mobile Networking Through Mobile IP, *IEEE Internet Computing*, (1998), **2**, n° 1, pp. 58–69.

[POR 98] PORTA (T. F. L.), RAMJEE (R.), WOO (T. Y. C.), SABNANI (K. K.), Experiences with Network-Based User Agents for Mobile Applications, *Mobile Networks and Applications (MONET)*, (1998), **3**, n° 2, pp. 123–141.

[REN 97] RENESSE (R. V.), BIRMAN (K.), HAYDEN (M.), VAYSBURD (A.), KARR (D.), Building Adaptive Systems Using Ensemble, *Technical Report* n° **TR97-1638**, (July 1997), Department of Computer Science, Cornell University.

[SAT 96] SATYANARAYANAN (M.), Fundamental Challenges in Mobile Computing, *Proceedings of the 5th Annual ACM Symposium on Principles of Distributed Computing (PODC'96)*, Philadelphia, Pennsylvania, USA, (May 1996).

[SEG 00] SEGARRA (M.), ANDRÉ (F.), A Framework for Dynamic Adaptation in Wireless Environments, *Proceedings of the Technology of Object-Oriented Languages and Systems (TOOLS'2000)*, Saint Malo, France, (June 2000).

[SHA 89] SHAPIRO (M.), GOURHANT (Y.), HABERT (S.), MOSSERI (L.), RUFFIN (M.), VALOT (C.), SOS: An Object-Oriented Operating System — Assessment and Perspectives, *Computing Systems*, (1989), **2**, n° 4, pp. 287–338.

[SIL 01] SILAGHI (B. D.), KELEHER (P. J.), Object Distribution with Local Information, *Proceedings of the 21st IEEE International Conference on Distributed Computing Systems (ICDCS'2001)*, Mesa, Arizona, USA, (April 2001), pp. 381–388.

[STE 99] VAN STEEN (M.), HOMBURG (P.), TANENBAUM (A. S.), Globe: A Wide-Area Distributed System, *IEEE Concurrency*, (1999), **7**, n° 1, pp. 70–78.

[TAL 94] TALLEY (T.), JEFFAY (K.), Two-Dimensional Scaling Techniques for Adaptive, Rate-Based Transmission Control of Live Audio and Video Streams, *Proccedings of the 2nd ACM International Conference on Multimedia'94*, San Francisco, USA, (October 1994), pp. 247–254.

[VAH 00] VAHDAT (A.), LEBECK (A.), ELLIS (C. S.), Every Joule is Precious: The Case for Revisiting Operating System Design for Energy Efficiency, *Proceedings of the 9th ACM SIGOPS European Workshop*, (September 2000).

[Wir99] Wireless Internet Today, Wireless Application Protocol - White Paper, (October 1999).

[YU 00] YU (H.), VAHDAT (A.), Design and Evaluation of a Continuous Consistency Model for Replicated Services, *Proceedings of the 4th Symposium on Operating Systems Design and Implementation (OSDI'2000)*, (October 2000).

[ZHO 88] ZHOU (S.), A Trace-driven Simulation Study of Dynamic Load Balancing, *IEEE Transactions on Software Engineering*, (1988), **14**, n° 9, pp. 1327–1341.

[ZHO 91] ZHOU (S.), WANG (J.), ZHENG (X.), DELISLE (P.), Utopia: A Load Sharing System for Large, Heterogeneous Distributed Computer Systems, *Technical Report* n° **257**, (December 1991), Computer Systems Research Institute, Toronto University.

Frédéric Le Mouël *holds a "Diplôme d'Ingénieur en Informatique et Communication", French equivalence of a Master's Degree in Languages and Operating Systems. He is a PhD candidate in computer science at the University of Rennes 1 / IRISA Research Laboratory. He is currently working at the École des Mines de Nantes as Temporary Research and Teaching Assistant. His research interests includes adaptive systems, mobile computing and distributed computing.*

Françoise André *is a professor at the computer science department (IFSIC) of the University of Rennes 1. Until 1996, she was the leader of the Pampa Group at the IRISA Research Laboratory, and was working about programming and execution environments for parallel and distributed computers. Since 1996, she performed her researches in the Solidor Group and her interests includes adaptive systems and mobile computing. One result of these researches is the development of a framework and a toolkit MolèNE allowing to build adaptive applications which have to take mobility aspects into account.*

Maria-Teresa Segarra *is graduated in Computer Sciences from the Universidad Politécnica de Valencia, Spain. She received her Ph.D. in 2000 from the University of Rennes 1, France. After one year as a Temporary Research and Teaching Assistant at IRISA Research Laboratory and IFSIC (University of Rennes 1), she was appointed at ENST Bretagne (Brest, France) in 2001 as an assistant professor. She contributes to the research activities of the ENST Bretagne Computer Sciences department in the areas of distributed systems, mobile computing, adaptive systems and object-oriented approaches.*

List of Figures

List of Tables