A Context-Aware Middleware Platform for Autonomous Application Services in Dynamic Wireless Networks

Nicolas Le Sommer, Frédéric Guidec, Hervé Roussain

A Context-Aware Middleware Platform for Autonomous Application Services in Dynamic Wireless Networks

Nicolas Le Sommer, Frédéric Guidec and Hervé Roussain
VALORIA Laboratory
University of South Brittany, France
{Nicolas.Le-Sommer, Frederic.Guidec, Herve.Roussain}@univ-ubs.fr

***Abstract*— Dynamic wireless networks, and especially mobile ad hoc networks, impose new constraints regarding the design and the implementation of service-oriented middleware platforms dedicated to mobile computing. Indeed from now, these platforms must be able to capture the context in which they operate in order to provide the services they host with an abstraction of their running context, and to notify them of the variations occurring in this context. These middleware platforms must also implement communication means that make it possible for services to support the frequent and unpredictable connection disruptions. In this paper, we present the design of a service-oriented middleware platform capable of environmental context introspection and of asynchronous communications.**

I. INTRODUCTION

In a near future, thanks to the recent technological advances realised in the wireless networking and computing hardware domains, the number and the diversity of the computing devices populating the physical environment in which we leave should increase significantly, thus giving to this environment some smartness capacities. Sensors and actuators are some examples of these new types of devices. The prospect of integrating such devices in pre-existent infrastructure-based networks, or in ad hoc networks, to provide people with information about their environment (e.g. forecast weather, road conditions, accident warnings) appears as an attractive one. However, the mobility, the volatility, the limitation of the communication range of wireless interfaces and the massive decentralisation of devices impose new constraints regarding the design and the implementation of the services deployed on such devices, as well as on the middleware platforms hosting these services. Indeed, the context in which these services and platforms operate suffers from frequent and unpredictable changes, requiring therefore from these middleware and services some capabilities that traditional services and platforms do not support, such as context-awareness, autonomous behaviour, adaptivity, proactivity and collaboration capabilities. Furthermore, in view of the dynamicity of this context, the synchronous communication paradigm appears to be not suitable, since it assumes that the involved services are present at the same time during the interactions. In contrast, with asynchronous communications, messages can be stored temporarily on certain hosts while being routed or disseminated in the network, and can be forwarded later when circumstances permit. Such a communication paradigm thus makes it possible for a message to eventually reach a destination that was not reachable at the time the message was sent originally.

In this paper, we present the design of a middleware platform with which we address the issues raised by the discovery and the access of resources in dynamic networks composed of heterogeneous devices. This middleware is able to perform a proactive and reactive introspection of the context in order to discover what resources are available in the environment (e.g. neighbouring devices, remote services), and to dynamically reify some abstractions modelling this context in order to provide the services it hosts with facilities for adapting their behaviour to the variations occurring in their running context. It also implements a collaborative and asynchronous communication paradigm to support the frequent and unpredictable transmission disruptions.

The remainder of the paper is organised as follows. Section II describes an application example, brings to the fore some fundamental issues using this example, and outlines our approach to address them. Section III presents the asynchronous communication service we have implemented in our middleware platform, and Section IV details the context-aware functionalities provided by this platform. Section V presents related work, and finally, Section VI concludes the paper.

II. MOTIVATIONS AND BACKGROUND

In this section, we present an application example involving both mobile and static devices operating within a dynamic ubiquitous environment. On the basis of this example we bring out some issues that must be considered when designing and implementing the application services and the middleware platforms that are likely to be used in such a context.

A. Motivating example

For the sake of illustration, let us consider the scenario presented in Figure 1. This figure illustrates a dynamic ubiquitous environment on a city scale. This environment includes distinct or discontinuous infrastructure-based networks and of mobile ad hoc networks. These networks are composed of both static devices, such as sensors and IEEE 802.11 (a.k.a. Wi-Fi) access points providing wireless access to wired domains

(including the Internet), and mobile devices equipped with Wi-Fi communication interfaces, such as the portable devices used by nomadic people (e.g. personal digital assistants, smartphones) and those embedded in vehicles. The standard IEEE 802.11 makes it possible for mobile devices to be connected to the access points of infrastructure-based networks or to communicate directly in ad hoc mode (i.e. without resorting to any kind of infrastructure-based network). The ad hoc networks considered in this scenario can appear and evolve spontaneously as mobile devices themselves appear, move and disappear dynamically in/from the network.

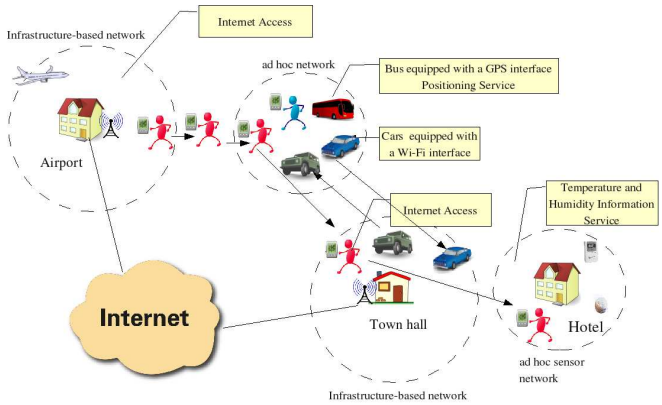

Fig. 1. Illustration of a dynamic ubiquitous environment on a city scale.

In this scenario, we mainly focus on the person who arrives by plane in the city, and who wants to go to its hotel all by herself. We also suppose that this person has a PDA (Personal Digital Assistant) equipped with an IEEE 802.11 communication interface, and that she has installed on her device a cartography and positioning application service that is able to display the map images it downloads from the Internet and to indicate the position of the user on these images thanks to the information it obtains from remote positioning services.

B. Context-awareness and discovery of resources and services

The environment in which the nomadic person operates is brought to evolve dynamically throughout its journey because networks, devices and services are likely to appear and disappear spontaneously. So as to provide acceptable quality of service to its user and, consequently, to improve the satisfactions of the later, the cartography and positioning application service has to be context-aware and able to adapt its behaviour to context changes. For instance, by performing a proactive and/or reactive introspection of the context, this application service should be able to discover itself the wireless network deployed in the airport when the person arrives in, and to identify what neighbouring devices offer a positioning service. Another application, also having communication needs with third devices, should be able to achieve the same context introspection task. Consequently, it is suitable that the introspection of the context be performed by the middleware rather than by the services themselves. Such an approach requires from the middleware the ability to provide the services it hosts with abstractions modelling their running context and to notify these services of the variations occurring in it, and from services the ability to exhibit their non-functional properties regarding their running conditions. By providing such information, services ask their platform for not being notified of all variations systematically, but only of the variations they are interested in. For example, at startup, the cartography and positioning application service could indicate to its hosting middleware platform that it requires a positioning service for running. In return, this middleware platform is expected to notify this application service when it discovers a positioning service in the environment.

C. Selection and provision of services

The devices populating an environment comparable to that presented in Figure 1 can be extremely heterogeneous, and can potentially act as clients and/or providers of services. Consequently, a service provider can hardly found a uniform quality of service for all end clients because these ones can have radically different characteristics. Similarly, it can be difficult for clients to found and to select services that fit their own software and hardware capacities. We thus believe that client services should be able to provide information about the hardware and software capabilities of the devices on which they run, and that providers should be able to exhibit both their functional and non-functional properties. For example, by providing service providers with information about the hardware characteristics of the PDA used by the nomadic person considered in Figure 1, the cartography and positioning application service helps these providers to deliver images whose resolution is compatible with the display capabilities of the device. By specifying the image resolution they can provide, service providers are expected to help the cartography and positioning application service to select suitable services.

Service access continuity is another important issue in dynamic networks. Indeed, in contrast of the wireless standards fostering the ”Always On, Always connected” scenario, and where mobile devices are actually considered as terminals with which users can access services offered by infrastructure-based networks, the dynamic networks we consider involve static and mobile devices equipped with communication interfaces whose transmission range is limited. Consequently, any device can only communicate directly with neighbouring devices. Multi-hop transmissions can sometimes be obtained by implementing a dynamic routing algorithm on each device, but it is observed that a realistic ad hoc network often presents itself as a fragmented network, and that devices that belong to distinct islands cannot communicate together, because no transmission is possible between islands. An additional problem with dynamic ad hoc networks is that in many realistic scenarios such networks present themselves as disconnected networks. As a consequence, direct transmissions between any pair of devices are not always feasible, as such transmissions require that both devices are active simultaneously in the network, and that a connected-path can be established between these devices at transmission time. Asynchronous communications

should help at overcoming these constraints and at improving service access. Indeed, as mentioned in the Section 1, messages are not simply routed and disseminated within the network, they can be stored temporarily on certain hosts while travelling from host to host, and be forwarded later when circumstances permit. This propagation model is inherently a probabilistic one, as eventual message delivery cannot be guaranteed. However, theoretical studies based on simulations (on variants of this model [1], [2]) show that is many realistic scenarios, the probability that messages actually reach their destination can be quite high. For example, before leaving the network island where the owner of the PDA and the bus reside, the car may have collected some messages that must be sent in the Internet (e.g. request sent by the cartography and positioning application service for downloading map images). When, it comes in the area covered by the access points of the infrastructure-based network deployed in the Town hall, this car can sent these messages in the Internet and broadcast the responses in the network, thus allowing other devices to store these responses. Thus, the Jeep should be able to deliver the data requested by the cartography and positioning application service when it joins the ad hoc network formed by the bus and the PDA owned by the nomadic person. With such a approach, the application service can have an Internet access even if it is not in a network providing such a access.

III. Asynchronous Communication service

In the recent years, message delivery issues in disconnected mobile ad hoc networks have been considered several times and following different ways. For instance, a new network architecture relying on the general principle of message switching in a store-and-forward mode has been proposed in [3]. In this approach data are transported as so-called bundles between bundle forwarders, which are capable of storing messages (or bundles) before sending them again in the network. In epidemic routing [4], [1], [5], messages are buffered in mobile hosts, and the exchanges of messages between these hosts are expected to allow message delivery in partially-connected networks. The models proposed in these papers mostly address the problem of message delivery in disconnected networks from a theoretical point of view. Indeed, they propose new algorithms and heuristics for delivering messages in such networks, and they report the results of simulations that are meant to demonstrate how these algorithms should run in realistic conditions. In contrast, the asynchronous communication service we present in this section is more practical, since it consists in an implementation of a middleware-level service for message dissemination in dynamic mobile networks. This service is meant to be used by application services running on our middleware platform.

The general architecture of our communication service is presented in Figure 2. With this message-oriented communication service, any message sent in the network is maintained as long as possible in a local cache by as many devices as possible, so it can remain available for devices that could not receive it at the time it was sent originally. The underlying idea is that the dissemination of multiple copies of the same message may contribute to overcome the volatility of devices, while the mobility of these devices can itself help transport information between islands in a fragmented network or in distinct networks. Besides providing a caching system where messages can be maintained in mobile devices, our service also provides facilities for message advertisement, message discovery, and message transport between neighbouring devices. For example, a device can sporadically or periodically notify its neighbours about all or part of the messages stored in its cache. It can also look for specific messages in its neighbourhood, and either push messages toward or pull messages from its neighbours. Like JMS [6], our communication service provides basic primitives for sending and receiving messages, and for looking for messages from the cache. It also offers facilities for application services for being notified when a new message concerning them is inserted in the cache.

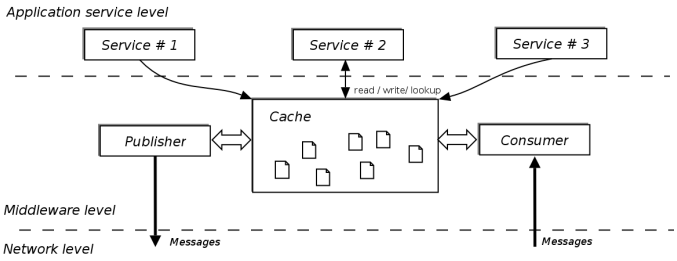

Fig. 2. General architecture of the asynchronous communication service.

A. Message structure

A message descriptor is associated with a each message. The descriptor of a message is meant to provide information about the message (type, client service, keywords, content, etc.). A message may encapsulate its own descriptor, but the descriptor can also be handled separately (which means it can for example be sent and stored separately).

```
<transfer-descriptor
   message-id="fb0097820f0b371"
   origin="172.17.96.37/map_service
   destination="*/gps_service"
   number-of-hops="5"
   date="Jan 30 08:26:32 CET 2004"
   lifetime="12:00:00"
   advertisement-period="00:20:00"
/>
```

Fig. 3. An XML-formatted transfer descriptor of a message exchanged by application-level services.

When an application-level message must be sent in the network, it must be encapsulated in a *transfer message*, whose descriptor specifies transmission parameters for this message. Figure 3 shows a typical XML-formatted transfer descriptor. The value of attribute *message-id* in a descriptor should be unique. This condition makes it possible for a device receiving

a message from the network to verify if a copy of this message is already available in its local cache.

The message descriptor in Figure 3 also specifies that this message has been sent by the application service *map_service* (attribute *origin*) and that it is addressed to all devices running the service *gps_service* (attribute *destination*). The attribute *number-of-hops* plays approximately the same role as the field TTL *(Time-To-Live)* in IP packets. It ensures that a message will not be propagated eternally in the network. The attribute *date* specifies the date when the message was originally sent in the network, and the attribute *lifetime* specifies that this message should be considered as being valid for only twelve hours after this date. The last attribute *advertisement-period* indicates that once the message has been put in the local cache of a device, this device may periodically announce that this message is available, with a periodicity of 20 minutes.

Advertisement and request messages make it possible to implement on each device some procedures ensuring proactive and/or reactive behaviours. A device behaving as a proactive consumer can send requests in order to ask for the direct transmission of specific messages, or in order to discover that some of its neighbours own messages it may be interested in. After receiving an advertisement for a message it is interested in, a reactive consumer can ask for the immediate transmission of this message. Similarly, a device behaving as a proactive producer can send advertisement messages in order to provide its neighbours with descriptors of the messages it can send them on demand.

It must be notice that a device that considers that its cache needs to be cleaned up has no obligation to maintain a message in this cache until it becomes obsolete. Likewise, a device has no obligation to announce that it owns a message, even if the descriptor of the message suggests an announcement period. In any case, attributes specifying a message's lifetime or suggesting a period for announcing the availability of this message are meant to serve as guidelines about how this message should be handled by devices. The actual behaviour of each device with respect to the messages it maintains in its cache can be guided by such suggestions, but it can also conform to default strategies defined by higher-level services, or according to user-defined global strategies applying equally to all messages.

B. Implementation details

The asynchronous communication service presented in this section was implemented in Java as an OSGi service [7]. Messages and message descriptors are thus reified as standard Java objects, which can be sent in the network either as serialised Java objects, or as XML-formatted documents. In the current implementation, message dissemination is performed by broadcasting UDP datagrams. We thus assume, for the time being, that each message is small enough to fit in a single datagram. In the future we plan to improve this communication service so as to address issues pertaining on message segmentation and reassembly. Both implementation details and utilisation examples of this service can be found in [8].

IV. Introspection and modelling of the context

As mentioned previously, our work aims to foster the design and the implementation of application services endowed with context-awareness and autonomous behaviour capabilities. Seen from this point of view, we have designed and implemented a middleware-level service capable of proactive and reactive context introspection. Likewise the previous service, this service is implemented in Java as an OSGi service. This section presents this service.

A. Modelling of resources

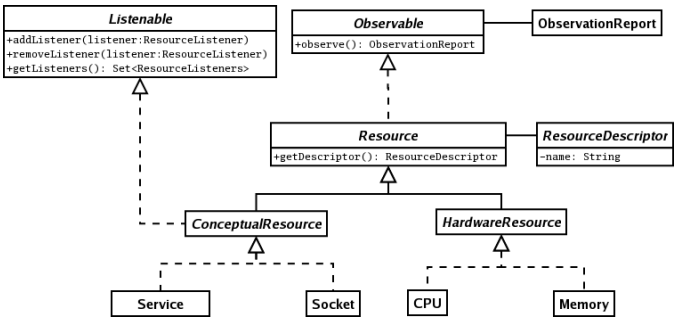

Fig. 4. Object-oriented modelling of resources

Software and hardware resources populating dynamic pervasive environments, such as that considered in Figure 1, can be of various nature. Consequently, the service responsible of performing context introspection must be able to define and to handle dynamically some abstractions modelling these different kinds of resources. These abstractions must also be used by the application-level services hosted by the middleware platform in order to adapt their behaviour to their running context. Three kinds of abstractions can be dynamically reified as first-class objects: resources, resource descriptors and resource observation reports. A resource object implements functionalities to use and control the resource it reifies. Resource descriptor provides "static" information about the resource, whereas the resource observation report gives information about the current state of the resource.

All resource objects implement the interface *Resource* presented in Figure 4. This interface extends the interface *Observable*, which defines a method *observe()* that returns an observation report (i.e. an object implementing the interface *ObservationReport*). A resource object is also designed to return a resource descriptor (i.e. an object implementing the interface *ResourceDescriptor*) when the method *getDescriptor()* is called. A resource descriptor provides a reference to the considered resource, a textual description of this resource and a set of attributes (objects implementing the interface *ResourceAttribute*) that describe the properties of this one (e.g. hardware characteristics for devices, functional and non-functional properties for services). Furthermore, in order to make resource reification easier, our framework provides a set

of interfaces reflecting a resource taxonomy. This taxonomy makes it possible to classify resources following their resource category (e.g. hardware, conceptual) and depending on whether they are *observable*, *listenable,* etc. These interfaces are used to define the operations that should be provided by the resource objects (i.e. by the objects reifying resources). For instance, an object modelling a CPU resource should implement the interfaces *Observable* and *HardwareResource*. This implementation makes it possible to observe the CPU resource by invoking appropriate methods on the CPU object. Similarly, conceptual resources –which mostly make sense at application level (sockets, files, threads, the software packages and the services, etc.)– should implement the interfaces *ConceptualResources*, *Observable* and *Listenable*. Further details about the reification of resources are given in [9].

In many circumstances, resource descriptors and observation reports must be sent in the network (e.g. at the resource advertisement process time). So as to facilitate both their dissemination within the network and their treatment, these objects are transformed into XML documents. An XML resource descriptor includes the identifier of the resource it describes, the object class of the descriptor, and a description of the resource. A resource descriptor can be reified dynamically from a XML document. For example, such descriptors can be used to provide an XML-formatted description of the hardware characteristics of a personal digital assistant. Similarly, to describe services running on this PDA, it is possible to use the simple XML description of application services provided by our middleware platform.

This context introspection service was designed so as to be highly extensible. Hence new classes can be developed in the future in order to model new resources, new observation reports and new resource descriptors.

B. Tracking and selection of resources

Resources can appear and disappear frequently in the environment. It is thus suitable to have a mean to identify and to keep track of available resources. Our middleware provides such a mean *via* the resource objects and a resource register. Indeed, all resource objects are designed to register a description of themselves (i.e. their own resource descriptor) with a resource register at creation time (see Figure 5). By consulting this register periodically, it is thus possible for an adaptive service deployed on a mobile device to obtain information about its local execution context. For instance in Figure 5, the service *Service#1* can call the method *getResources()* of the resource register in order to obtain a list of the local resources. Such a service can also act as listener of the resource register, and thus can be notified of the variations of the local context (i.e. of the appearance and disappearance of resources). As shown in Figure 5, the service *Service#2* has registered itself as listener with the resource register in order to be notified when a new resource appears in the environment (e.g. the resource *Resource#2* in Figure 5).

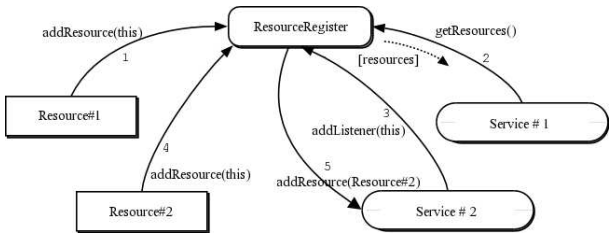

Fig. 5. Tracking of resources

In certain circumstances, it could be suitable to focus only on a specific resource –or on a specific kind of resources. For example, a service such as the service *Service#2* in Figure 5 should be notified only when a given resource become available in the environment. Such a need requires to be able to select resources according to a predefined criterion. Our framework provides such a functionality with the notion of ”resource pattern”. A resource pattern defines a function *isMatchedby()*, which takes a resource descriptor object as a parameter, and returns a boolean whose value depends on whether this object satisfies the considered selection criterion or not. In a simple case resource selection can rely on the actual type of the resource which is submitted to the test. But one can also implement more sophisticated selection mechanisms by taking resource type and resource attributes into account.

C. Introspection of the local system

In the approach we propose, the introspection of the network context is fundamental, but it is not sufficient. Indeed, services must also have information about the resources offered by the device on which they run. The introspection of the local system can be easily performed by obtaining a description of the local resources from the resource register and by monitoring the local resources. These resources can be monitored following either a synchronous or an asynchronous mode. The synchronous monitoring is obtained by implementing a callback mechanism in resource objects. Any resource objects can admit one or several listeners. Whenever a variation of the resource occurs –or when a method is called on a conceptual resource–, the resource object informs all its registered listeners (see Figure 5). All resources cannot be monitored using the synchronous approach, though. For example, access to the CPU is not achieved by calling methods explicitly. Instead it is controlled directly by the scheduler of the underlying operating system. In order to deal with system resources such as the CPU (system memory, network interfaces, etc.), which can hardly be monitored synchronously, we propose to do with asynchronous monitoring. Monitoring a resource asynchronously consists in consulting the state of this resource explicitly every now and then, in such a way that the time of the observation does not necessarily coincide with the time of an attempt to use the resource. This observation can be performed by calling the method *observe()* on the object that models the resource that

must be observed. This method gives an observation report (i.e. an object implementing the interface *ObservationReport* described in Figure 4) in return.

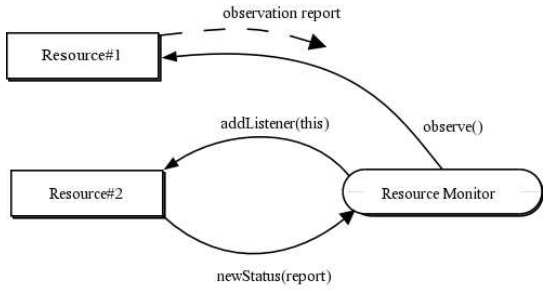

Fig. 6. Monitoring of resources.

D. Introspection of the network neighbourhood

The introspection of the network neighbourhood aims to discover what resources are available in the physical environment. In contrast of the introspection of the local system, which can be simply achieved using the resource register and the objects modelling the local resources, the introspection of the network neighbourhood requires a more sophisticated mechanism to discover automatically the resources available in the environment. Indeed, as brought out in the Section II, the network context in which mobile and static devices operate suffers from frequent and unpredictable changes.

The context introspection service implemented in our middleware platform provides facilities to address this issues. First, it provides facilities to discover the close wireless networks, to configure the wireless communication interface of mobile devices automatically, and to identify what hosts can be reached in these networks. Obviously, owners of mobile devices can configure this functionality in order to select themselves the network they want to join. In its current implementation, this service relies on the system commands of the Linux operating system (e.g. wireless tools). The discovery of neighbouring hosts and of remote services is achieved using the asynchronous communication service presented in the previous section.

The discovery of hosts can thus be made either proactively by sending host discovery requests in the network, or reactively by analysing the unsolicited advertisements sent by other devices in the network (each device is responsible to manage its perception of its environment). When the service responsible of the context introspection receives an advertisement from a remote host, it reifies this advertisement as a resource object (i.e. as a host descriptor object) and registers it in the resource register dedicated to remote resources.

The discovery of the remote services is performed following the same mode. In the proactive approach the middleware sends in the network an XML document containing the service request. This request is formally describe using a resource pattern. The service providers, which have a service whose descriptor fits the pattern specified in the request, send in response an XML document containing the description of the service they can offer. The service discovery can also be performed following a reactive mode. In this mode, service providers send unsolicited advertisements (i.e. service descriptors).

V. Related-Work

A. Asynchronous Communication in mobile networks

The asynchronous communication service presented in Section III can obviously be compared with that proposed by Vahdat and Becker in [1], where they introduce the concept of *Epidemic Routing*. In their model, messages are buffered in mobile hosts and are exchanged randomly among hosts. This model makes the delivery of messages possible in partially-connected ad hoc networks. Although Vahdat and Becker are mostly concerned with unicast transmissions in their paper, they suggest that Epidemic Routing is also appropriate for supporting multicast traffic. A variant of Epidemic Routing has been proposed in [2]. This variant introduces probabilistic heuristics in order to increase the chances that a message is routed toward its destination.

The approach we investigate with our asynchronous communication service can also be compared with that of IRTF[1]'s group working on delay-tolerant networks (DTNRG: *Delay-Tolerant Network Research Group*[2]). This group focuses on the deployment of networks in high dynamic environments in which end-to-end connectivity cannot be guaranteed. It has defined a new network architecture relying on the general principle of message switching in a store-and-forward mode. This approach consists in transporting pieces of information – called *bundles*– and implementing forwarders that are capable of storing messages (i.e. the bundles) before they can be sent againin the network [3].

B. Middleware Support for Adaptive Services and Pervasive Computing

The middleware Centaurus [10] and Mobishare [11] provides owners of mobile devices with facilities to discover the services that are offered spontaneously within an infrastructure-based network, as well as to access to these services remotely. Both Centaurus and Mobishare implement a centralised approach of service management: service as are deployed on predefined servers, service discovery relies on the utilisation of a unique service repository, etc. Such middleware are designed to be used in close smart environments (e.g. home, office) including low mobile devices. Since service clients do not suffer from frequent and unpredictable service disconnection, these middleware do not implement mechanisms to ensure service access continuity for nomadic people. Moreover, the centralised approach implemented by these middleware do not allow to take advantages of wireless communication. Indeed, in contrast of a peer-to-peer approach,

[1]IRTF: *Internet Research Task Force* (http://www.irtf.org).
[2]http://www.dtnrg.org

a centralised approach does not favour the collaboration of mobile devices to achieve service provision.

Many recent middleware platforms designed to be used in mobile dynamic environments (e.g. ad hoc networks) implement peer-to-peer technologies in order to make it possible for mobile devices to collaborate spontaneously. These technologies provide means to describe, to advertise, to request, and to retrieve services. For instance, in domotic environments, such technologies permit people to control their domestic devices (e.g. light, tv). UPnP (Universal Plug and Play) [12], SLP (Service Location Protocol) [13] are some example of these technologies. The middleware Konark [14] relies on an implementation of the UPnP technology in order to describe, to announce and to discover services in mobiles ad hoc networks. The services considered in Konark are Web Services that can be invoked using the protocol SOAP. Like Centaurus and Mobishare, Konark is designed to be used in wireless environments including low mobile devices.

The middleware 7DS [15] and XMIDDLE [16] are not designed to provide services, but are designed to find, to relay, to disseminate and to share information in peer-to-peer mobile ad hoc networks. 7DS implements sophisticated prefetching and collaborative mechanisms that could be used to relay, and to provide access to services asynchronously in high dynamic mobile networks, such as delay-tolerant ad hoc network. The middleware XMIDDLE is dedicated as for it, to the sharing of XML documents in mobile networks composed of heterogeneous devices. It implements replication and reconciliation mechanisms in order to allow nomadic people to have copies of documents on their own devices, and thus to make it possible for them to access to these documents autonomously. The reconciliation mechanisms aims at ensuring the update of the copies when devices are connected.

None of the above-presented works are designed to support adaptive services, and thus do not the services deployed on the mobile devices with mechanisms that allow them to perform an introspection of both their network context and their local context. Yet, as shown in this paper, it could be relevant to deploy adaptive services on mobile devices in order to offer service access continuity to owners of mobile devices. In the remainder of this section we present some works designed to support adaptive services.

The project Molène [17] aims at offering an object-oriented framework to implement adaptive applications. This framework defines a notification system of context variations based on a set of resource monitors organised in two levels, and a customisable system of reaction. The monitors of the first level are specialised monitors (e.g. battery monitor, network interface monitor). The monitor of the second level are designed to analyse information they received from the monitors of the first level, and to notify applications of the variations of their context according the directives they received at startup.

The aim of the project AMPROS [18] is to design and to develop a middleware platform in dynamic wireless networks with the generalisations that can be used for such services as emergency aid, crisis managements etc. Seen from this point of view, AMPROS proposed an approach, that is similar to ours, since it mostly relies on the deployment of disconnected-components on mobile devices in order to ensure service continuity (the disconnected-components provide a service identical to those offer by the remote components). AMPROS also implements reconciliation mechanisms in order to ensure that the state of the local disconnected-components remains the same that that of the remote components. However, the deployment of component – and of services– require to have security mechanisms in order to monitor the components at runtime. Indeed, a component could be introduced in the network by a malevolent device in order to put the system in jeopardy by destroying crucial data files for instance.

Like Molène and AMPROS, the middleware Carisma [19] implements some context introspection mechanisms in order to make the development of adaptive application easier. Seen from this point of view, this middleware provides to the application it hosts an abstraction of their running context and a set of reflexive mechanisms in order to act on this context using this abstraction.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have presented key services of a service-oriented middleware platform we have designed and implemented in order to foster the development of application services with context-awareness, autonomous behaviour, adaptivity, proactivity and spontaneous collaboration properties. The first service presented in this paper permits the asynchronous dissemination of messages in dynamic wireless networks, such as those composed of highly mobile and volatile communicating devices. It proposes an asynchronous, peer-to-peer, message-oriented propagation model, where each message received by a device can be maintained in a local cache in this device, so it can later be sent again in the network, either spontaneously, or after a request for this message has been received from another device. This approach is expected to help do with the volatility of devices, since it permits that messages reach devices that are only active sporadically in the network. It is also expected to permit information dissemination in a fragmented network, taking advantage of the mobility of devices which can serve as carriers between disconnected parts of the network or between distinct networks. The second service presented in this paper is designed to perform a continuous context introspection, to reify dynamically objects modelling the resources populating the context, and to notify the application services hosted by the middleware platform of the variations occurring in their running context.

Since the development of the middleware platform is not complete, no extensive performance evaluation has been performed so far. However, preliminary experiments we have made prove to be promising. Before starting an evaluation process of our middleware, we plan to improve the discovery and delivery of services. Indeed, the asynchronism inherent to delay-tolerant networks raises specific difficulties. For instance, the scope in time and space of the discovery queries and service announcement must be controlled. To achieve this goal we plan to introduce the concept of context attribute as an effective, flexible means to exploit relevant context

information during the service discovery and delivery process. Context attributes can express various context information including user's preferences, quality of service, network conditions, service availability, etc. Such attributes aims at helping middleware –or users– to choose the best services that fulfil their needs.

REFERENCES

[1] A. Vahdat and D. Becker, "Epidemic Routing for Partially-Connected Ad Hoc Networks," Tech. Rep. CS-2000-06, UCSD, July 2000.

[2] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic Routing in Intermittently Connected Networks," in *Proceedings of the Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2003)*, June 2003.

[3] K. Fall, "A Delay-Tolerant Architecture for Challenged Internets," in *Proceedings of the ACM Special Interest Group on Data Communication 2003 (SIGCOMM 2003)*, (Karlsruhe, Germany), ACM, Aug. 2003.

[4] P. Poupyrev, M. Kosuga, and P. Davis, "Analysis of Wireless Message Broadcast in Large Ad Hoc Networks of PDAs," in *Proceedings of the Fourth IEEE conference on Mobile and Wireless Communications Networks*, pp. 299–303, 2002.

[5] A. Qayyum, L. Viennot, and A. Laouiti, "Multipoint Relaying; an Efficient Technique for Flooding in Mobile Wireless Networks," Reseach Report 3898, INRIA, Mar. 2000.

[6] S. Microsystems, "Java Message Service Specification - Version 1.1," Apr. 2002.

[7] O. Alliance, "Osgi service platform, release 3," Mar. 2003. http://www.osgi.org/.

[8] F. Guidec and H. Roussain, "Asynchronous Document Dissemination in Dynamic Ad Hoc Networks," in *Second International Symposium on Parallel and Distributed Processing and Applications (ISPA'04)* (J. C. et al., ed.), vol. 3358 of *LNCS*, (Hong-Kong, China), pp. 44–48, Springer Verlag, Dec. 2004.

[9] N. Le Sommer, "Towards Dynamic Resource Contractualisation for Software Components," in *2nd International Working Conference on Component Deployment (CD 2004)*, vol. 3083 of *LNCS*, (Edinburg, Scotland, UK), Springer Verlag, May 2004.

[10] L. Kagal, V. Korolev, H. Chen, A. Joshi, and T. Finin, "Centaurus: A Framework for Intelligent Services in a Mobile Environment," in *International Workshop on Smart Appliances and Wearable Computing (IWSAWC), at the 21st International Conference on Distributed Computing Systems (ICDCS)*, Apr. 2001.

[11] E. Valavanis, C. Ververidis, M. Vazirgianis, and G. Polyzos, "MobiShare: Sharing Context-Dependent Data and Services from Mobile Sources," in *IEEE/WIC International Conference on Web Intelligence (WI'03)*, (Halifax, Canada), Oct. 2003.

[12] UPnP Forum, "UPnP Device Architecture," tech. rep., UPnP Forum, June 2000.

[13] E. Guttman, C. Perkins, J. Veiades, and M. Day, "Service Location Protocol, Version 2," RFC 2608, Internet Engineering Task Force (IETF), June 1999.

[14] S. Helal, N. Desai, V. Verma, and C. Lee, "Konark : Service Discovery and Delivery Protocol for Ad-hoc Networks," in *Third IEEE Conference on Wireless Communication Networks (WCNC)*, (New Orleans, USA), Mar. 2003.

[15] M. Papadopouli and H. Schulzrinne, "Seven Degrees of Separation in Mobile Ad Hoc Networks," in *IEEE GLOBECOM*, Nov. 2000.

[16] C. Mascolo, L. Capra, S. Zachariadis, and W. Emmerich, "XMIDDLE: A Data-Sharing Middleware for Mobile Computing," *Personal and Wireless Communications Journal 21(1)*, 2002.

[17] F. Andre and M. Segarra, "A Framework for Dynamic Adaptation in Wireless Environments," in *Proceedings of TOOLS Europe 2000*, (Mont St. Michel, France), June 2000.

[18] D. Conan, C. Taconet, D. Ayed, L. Chateigner, N. Kouici, and G. Bernard, "A Pro-Active Middleware Platform for Mobile Environment," in *International Conference on Software Engineering (IASTED 2004)*, (Innsbruck, Austria), Feb. 2004.

[19] L. Capra, W. Emmerich, and C. Mascolo, "CARISMA: Context-Aware Reflexive mIddleware System for Mobile Applications," in *IEEE Transactions on Software Engineering*, Nov. 2003.