A COMMUNICATION MIDDLEWARE FOR UBIQUITOUS MULTIMEDIA ADAPTATION SERVICES

Ning Li[1], Hillary Tarus[2], Klaus Moessner[3], James M. Irvine[2]

[1] *Knowledge Media Institute, The Open University, UK, n.li@open.ac.uk*
[2] *Department of Electrical and Electronic Engineering, University of Strathclyde, UK, {hillary.tarus,j.m.irvine}@ee.strath.ac.uk*
[3] *Centre for Communication System Research, University of Surrey, UK, k.moessner@surrey.ac.uk*

Abstract. Ubiquitous services have gained increasing attentions in the area of mobile communication aiming to allow service access anywhere, anytime and anyhow while keeping complexity to a minimum for both users and service providers. Ubiquitous environment features a wide range and an increasing number of access devices and network technologies. Context-aware content/service adaptation is deemed necessary to ensure best user experience. We developed an Adaptation Management Framework (AMF) Web Service which manages the complexity of dynamic and autonomous content adaptation and serves as an invisible enabler for ubiquitous service delivery. It remains challenging to manage the tasks involved in the communication between the AMF Web Service and the user's environment, typically represented by various types of intelligent agents. This work presents a middleware which manages those tasks and serves not only as a protocol gateway, but also as a message translator, a service broker, a complexity shield etc., between AMF Web Services and User Agents.

Keywords: Agent, Web Service, Content Adaptation, Ubiquitous Service, Middleware

1 INTRODUCTION

Ubiquitous services have attracted a lot of attentions in the area of mobile communication among service providers, telecommunication operators and technology

[0] MSC 2000 Symbols: 94A99:Communication, information, 68M14:Distributed systems

manufactures in recent years. As the name implies, it represents communication scenarios where services can be accessed anytime, anywhere and anyhow without explicit involvement from any players in the service delivery process. Though it shares similar visions of ubiquity with, and may be seen as one of the areas defined by "ubiquitous computing" or "pervasive computing" which has been rapidly developed in last decades, it has distinctive features when it is viewed from mobile communication and service provision points of view. Ubiquitous computing, envisioned by Mark [1], is best described as "a shared vision of small, inexpensive, robust networked processing devices, distributed at all scales throughout everyday life and generally turned to distinctly common-place ends". By this definition, pervasive computing is mostly approached with an aim to embed computing intelligence into user's surrounding environment and therefore with focus mainly on embedded devices and technologies such as sensors, processors, actuators and their connectivity, user interfaces etc. However, in ubiquitous services, the aims are geared towards "ensuring best user experience when the user access services from anywhere, at anytime, with any particular preferences and using any accessible devices while keeping minimum complexity to the user". The services mainly refer to network connectivity services as well as content services in this context. Ubiquitous services are being brought closer to reality, from the network connectivity perspective, with today's advancements in network access technologies, such as WiFi, WiMAX, UMTS etc. and their interconnectivity. They are still remotely perceivable from the content perspective without some mechanisms in place that support the adaptation of a multitude of content mix to a wide range of access devices as well as user's different preferences. Content providers are making efforts from their perspectives, for example, to produce multiple versions of each piece of content for every conceivable device, or simply differentiate their contents for *mobile* devices from PCs typically for web contents by starting the content URL with mobile or ending with *.mobi* etc. However, these efforts are neither flexible nor efficient in coping with the dynamicity and heterogeneity of ubiquitous communication environment. In addition, they are not cost-effective for content providers, and therefore a dynamic content adaptation mechanism is needed. Designing and developing such a dynamic adaptation mechanism has been actively investigated by research communities and industries. For example, Digestor [1] and WebAlchemist [2] systems are two of the early adaptation systems designed for web pages that perform automatic HTML re-authoring based on various heuristics located at an HTTP proxy. The first significant deployment of dynamic adaptation mechanism into service delivery network by industry was achieved by Vodafone UK [2] when they launched Mobile Internet in 2007. It was deemed to bring the richness of the PC-internet to mobile phones, and effectively changed the way customers use their phones to browse data. The core technology deployed is a Web Translation Engine (WTE)[3] placed in its core network between

[1] http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html

[2] http://www.vodafone.co.uk

[3] http://www.betavine.net/bvportal/resources/vodafone/mics

content providers and end users to do dynamic adaptation when needed. This is a significant milestone in the direction towards ubiquitous services in reality though it is under the theme of "Mobile Internet" and achieves only a small portion of targets envisioned for ubiquitous services.

The program of "removing the barriers to ubiquitous services" [4]initiated by the Mobile Virtual Center of Excellence (MVCE)[5] which collates the views and visions from network operators, content providers and device manufactures, has looked into the challenges of ubiquitous services from three major perspectives, that is *User Perspective*, *Network Perspective* and *Content/Service Perspective*. Each perspective has a centric view of the challenges and problems in the respective domain. From the *User Perspective*, the challenges are to manage the dynamically configured devices in a way that they work as a whole to provide users with ubiquitous service experience. From the *Network Perspective*, the focuses are mobility, security and Quality of Service (QoS) management across heterogeneous networks. More importantly when they are integrated, their individual performances are not discounted. From the *Content/Service Perspective*, designing an Adaptation Management Framework (AMF) that manages the complexity of dynamic and autonomous content adaptation will be of the most importance. The vision of ubiquitous services can not be easily realised without close collaborations among these three perspectives, particularly when best performance is sought. For example, from the *Content/Service Perspective*, a complete and efficient adaptation solution is to have a system that has the appropriate logic to analyze the content and all aspects of the delivery context and formulate an adaptation strategy that will deliver the most-suited result and ensure best user experience. The delivery context undoubtedly comes from user's environment as well as network providers that are specifically looked at from the *User Perspective* and *Network Perspective* respectively. Of paramount importance to the complexity addressed from each perspective are the communications between solutions sought from each perspective. In this paper, we will focus on the presentation of a middleware component, called Dispatcher, which provides the required functionality for communications between the *Content/Service Perspective* and the *User Perspective*. In other words, the Dispatcher supports the incorporation of an adaptation management system into a service delivery network for ultimate ubiquitous services. Prior to the introduction of this component, we introduce some fundamental background work from both perspectives in the next section.

2 BACKGROUND

2.1 Personal Distributed Environment (PDE)

From *User Perspective*, the devices and network access technologies available to the user for content/service access have become more volatile in ubiquitous environment

[4] http://www.mobilevce.com/frames.htm?core4frame.htm

[5] http://www.mobilevce.com

where the user moves around. A mechanism is required to maintain a record of those dynamically configured devices and connections. The Personal Distributed Environment (PDE) concept [3], developed by MVCE, implements a user-centric view of communication environment that "encompasses a user perspective of multiple devices (both local and remote) accessing multiple services via multiple networks, all of which can be changing dynamically". The core component of the PDE architecture is called the Device Management Entity (DME) [3]. It is responsible for maintaining a set of registers that contain information about the existence, location and capabilities of the various devices within the PDE. In the program of "removing the barriers to ubiquitous services", the PDE's functionality is further extended from the *User Perspective* with a new component, referred to as Personal Assistant Agent (PAA). This is to target innovations that introduce automation and intelligence in the management of user environment context as well as user preferences in order to provide personalized context-aware services. The PAA is the entity that possesses the intelligence to efficiently detect, extract, and process this context information and make appropriate decisions on behalf of users. It provides meaningful contextual information to the adaptation framework proposed from the *Content/Service Perspective*, and thus to improve the end user experience from the adapted content/services.

2.2 Adaptation Management Framework (AMF)

From the *Content/Service Perspective*, the main task is to develop an Adaptation Management Framework (AMF) that manages the complexity of dynamic and autonomous content adaptation. Two major functional entities are proposed to provide the core functionalities, they are: Adaptation Manager (AM) for context acquisition, context analysis and adaptation decision making [4], and Content Adaptor (CA) for coordinating adaptation operations to fulfil the adaptation task once adaptation decisions are made by the AM [5]. The AMF is typically represented by the AM since it exposes APIs and communicates with the external world while keeping the CA hidden in the background. With Web Service being the de-facto means of exposing services over the Internet, we implement the AMF as a Web Service so it can be pluggable in and out of the legacy content delivery network without casting much redesign or re-installation issues. Following the W3C recommendation of Web Service standards, the AMF uses Web Services Description Language (WSDL) to describe the service, Simple Object Access Protocol (SOAP) as the transport protocol and Universal Description Discovery and Integration (UDDI) as the publishing and discovery mechanism. In the AMF architecture, there is another important component we refer to it as Dispatcher which, though loosely coupled with the AM-CA, is crucial to make the AMF actually of practical use. Its tasks, as the name implies, is to dispatch content to user's PDE and then to end device as well as dispatching information from user's PDE, such as requests and accompanying contexts, to the AM-CA, over heterogeneous networks. The Dispatcher forms the main contribution of this paper and will be the focus of the following sections.

3 DISPATCHER FUNCTIONALITY

An adaptation process is described first to help the understanding of the Dispatcher. Typically, an adaptation process begins when the Personal Assistant Agent (PAA) in user's Personal Distributed Environment (PDE) sends a request on user's behalf, via HTTP, FTP for example, with accompanying contexts, such as device capability or user preferences, to the Adaptation Manager (AM) in the Adaptation Management Framework (AMF) via the Dispatcher. The AM decides if the requested content needs adaptation based on the received contexts. If the decision is that an adaptation is required, it sends the content link along with adaptation decisions to the Content Adaptor (CA) to get content fetched and adapted. Otherwise it simply fetches the content. The output of the AM-CA, which is now the suited version of the original content to user's contexts, will be passed through the Dispatcher to the user's PDE for presentation. In this process, the Dispatcher forms the crucial gateway between the user's PDE, i.e. PAA technically, and the AMF for both directions. We identify that the Dispatcher should include at least but not necessarily limited to the following functionalities:

1. Protocol gateway. Firstly and most importantly, the Dispatcher enables components built on different technologies or paradigms to easily communicate. It acts as a protocol gateway for different schemes/protocols that exist between the PDE and the AMF. The PAA was notably implemented as software agent using Agent technology to reflect its nature of flexible, responsive, autonomous etc., while the AMF was implemented as Web Services. The Dispatcher's work is to marry these separate protocols so that the PAA and the AMF can communicate with each other effortlessly.
2. Service advertisement and publishing on the capabilities of the AMF. Being the initial point of contact for the interaction between the PAA and the AMF, the Dispatcher advertises and registers AMF services at the PDE side so that it makes the PAA aware of the service options it can utilise and meanwhile keeps the AMF services looking like as within the PDE.
3. Complexity shield. The Dispatcher hides the distributive nature of the AMF by offering only one and homogeneous access point to the PDE. The Dispatcher also hides from the user the fact that there can be several distributed communication mechanisms being used. If well designed, the Dispatcher should be able to automatically choose among the available communication mechanisms. In our proposal, and based on performance comparisons done in [6], the Dispatcher implements the Java Remote Method Invocation (RMI), OMG Common Object Broker Architecture (CORBA), HTTP/SOAP and Java Messaging Service (JMS).
4. Content redirection. The Dispatcher forwards and routes requests from the PDE to the AMF, and after adaptation, the AMF outputs to appropriate user devices. Since all AMFs are registered with the Dispatcher, the Dispatcher is able to route requests to the most appropriate AMF or group of

AMFs depending on capability, availability and capacity in case of a one-to-many negotiation. Inversely, the response of PAA requests is not necessarily returnable to the device in which the PAA currently resides, but another device within the PDE. An exemplar scenario is where a user watching video clips on a personal computer with a large display makes an adaptation request for the same to be displayed on a mobile device with small screen dimensions because he wishes to continue watching the video while travelling. The Dispatcher comes into play by receiving such requests and redirecting response, i.e. AMF-adapted content, to the intended device.

5. Service broker. Related to the above functionality is the negotiation broker role of the Dispatcher. The Dispatcher receives the user's context profiles along with content request and negotiates with all the registered AMFs to find the one that provides the best service or value for money according to user preferences among all the capable AMFs. During negotiation, the Dispatcher should intrinsically consider whether the AMFs are under-loaded or over-loaded to control load balance. The negotiation, which generally involves many rounds of back and forth from the PAA to the AMF, bears in mind the cost of passing data through the small capability wireless links.

Among all the functionalities mentioned above, a majority of them can be solved straightforwardly with the help of agent technology, in particular agent platform which is a bunch of APIs for developing and interacting with agents. Services provided by agent platforms include communication, discovery, coordination, mitigation, security etc. Among the existing agent platforms, which include FIPA-OS [6], Grasshopper [7], JACK Intelligent Agents [8] , JADE [9], and ZEUS [10]. JADE is chosen for our purpose for being open-source, complying with the FIPA [11] standard and more importantly, it has an active development community plus an add-on called LEAP which provides the small footprint agent runtime platform required for smart devices, typically featured in ubiquitous environment.

One specific feature of the Dispatcher, which requires particular emphasis in this paper is, its negotiation skill which enables PAA to negotiate with the relevant AMFs depending on capability, availability and capacity of the AMFs as well as a user's preferences (e.g. towards cost). This is analogous to electronic commerce trading, which is nothing more than electronic catalogues available online allowing users to choose a product or a service at a fixed-offered price [7]. However, in a ubiquitous environment, the factors being considered in the trading usually do not have a predetermined fixed value but can be negotiated depending on certain context.

[6] http://sourceforge.net/projects/fipa-os/
[7] http://cordis.europa.eu/infowin/acts/analysys/products/thematic/agents/ch4/ch4.htm
[8] http://www.agent-software.com
[9] http://jade.tilab.com/
[10] http://sourceforge.net/projects/zeusagent/
[11] http://www.fipa.org

3.1 Negotiation Functionality

Generally, as social and economic representatives of humans, agents equally need to be endowed with strong negotiation skills for use when dealing with other agents or humans. First of all, we need to clarify the meaning of the terms used in a negotiation process: *negotiation*, *negotiation protocols* and *negotiation strategies*. Agent negotiation is defined as a form of interaction in which a group of agents with the desire to cooperate but with potentially conflicting interests seek to reach a mutually acceptable division of the scarce resource or resources [8]. From this definition, it is clear that the underlying features of negotiations include mutual agreement, division of a limited resource, and haggling among others or etc. However the same should be considered with the limiting constraints of time, trust etc. This is to say that the haggling for example cannot go on forever but has to stop at some given amount of time, probably set by the PAA/user. Consequently, the *negotiation protocol* is defined as a set of rules that govern the interactions and covers permissible types of participants for example negotiators and relevant third parties, negotiation states of accepting bids or closing a negotiation, state transitions and valid actions by the negotiators [9]. FIPA provides a contract-net protocol [12] to facilitate the negotiation process. In essence, contract-net specifies the interaction between distributed agents for fully automated competitive negotiation through the use of contracts. Under a contract-net system, a user could specify, for example, the good he wanted as well as a maximum price he was willing to pay. In our case, Dispatcher uses *cost* and *time* constrains specified by the user as one type of user's contexts when requesting services as the contract term to negotiate with candidate AMFs. Different from *negotiation protocols* which, as just described, define the way participants in a market carry about interacting with each other i.e. what to say, when to say, and possibly how to say, a *negotiation strategy* defines the way in which the participants in the negotiations behave within the set rules. This behaviour includes but is not limited to when and what to concede, when to hold firm, when to avoid negotiation at all etc. [10]. Several negotiation strategies have been discussed primarily in conflict resolution literature and lately in the agent negotiation schemes [11]. Of particular interest are the five negotiating strategies postulated by Blake and Mouton [12], which have been implemented in the Dispatcher. These strategies are: *Compromising*, *Accommodating*, *Collaborating*, *Competing*, *Avoiding*. Different negotiation strategies yield different utilities to the negotiating partners as we observed in our experiments.

4 DISPATCHER IMPLEMENTATION

Fig. 1 depicts the architecture of Dispatcher implementation. The PAA submits service request on user's behalf in the form of Agent Communication Language (ACL) as defined by FIPA. This request may be transmitted over wireless, e.g. WIFI,

[12] http://www.fipa.org/specs/fipa00029

GPRS, GSM, Bluetooth etc. or in rare cases over wired links, due to the mobile and dynamic nature of ubiquitous environment. Therefore, most often this connection is intermittent and with low capacity. The Dispatcher receives the request from the PAA first. Since the Dispatcher is agent and talks also in the ACL language, there are no interface problems. Upon receipt the request message from the PAA, the Dispatcher creates a PAA Helper Agent to communicate and negotiate with the AMs on behalf of the PAA. The main advantage of using such architecture is that we are able to reduce the communication overhead between the PAAs and the AMs over low capacity wireless links because PAA does not have to move and only its Helper Agent does.

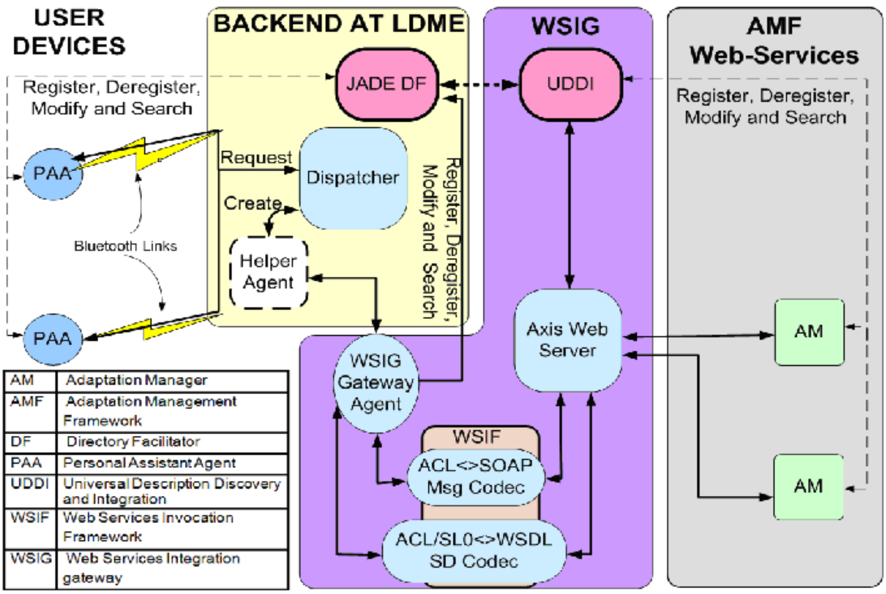

Fig. 1. Dispatcher implementation architecture

The PAA Helper Agent in the Dispatcher framework then negotiates on behalf of the PAA with the AMFs for the best suitable adaptation services according to user's preferences. We consider monetary cost and service processing time in this development. More factors can be similarly added for consideration when necessary. The negotiation process is generally an iterative procedure and will involve a relatively larger amount of conversation messages. However, contrary to the existence of numerous PAAs which are able to roam from one sub-network to another of the same user, the Dispatcher is an integral part of the AMF and has a one-to-one mapping relation with the AM-CA bundle. Though they are not necessarily in the same physical location since it is a matter of a deployment and will be subject to business model, they can be most conveniently connected with each other via a wired link to avoid problems associated with the amount of conversation data. Even if the

Dispatcher is physically deployed in user's PDE, typically in a Local DME (LDME) as seen in Fig. 1, it will not be as mobile and dynamic as PAAs. Therefore it can be put onto any device with a wired connection. The AMFs, as in the current case, are implemented as Web Services instead of agents and thus ACL messages are not understood to them. The Dispatcher *protocol gateway* function now comes into play. The Dispatcher converts ACL message to SOAP message that the AM uses and vice versa for the responses from the AM. In fact, the advantage of blending agents and web services has been highlighted in Zehreddine et. al. [13]. However, only until recent years, a protocol convector platform named Web Services Integration Gateway (WSIG)[13] had been developed and become available as a JADE add-on. We utilise this to help the realisation of the Dispatch functionality, but enhance the capability of WSIG such that service acquisition and invocation can be more focussed on the AM Web Services rather than agents which is the case in current WSIG implementation. Our implementation of the gateway interface involves the use of WSIG with major modifications to automate the Web service invocation. We developed a dynamic invocation module that parses the ACL or WSDL retrieved by WSIG and automatically creates a meaningful SOAP or ACL response respectively depending on the direction of communication. The invocation module utilises some APIs from Apache Web Service Invocation Framework (WSIF)[14] with appropriate modifications for this usage.

In a nutshell, after the Dispatcher framework is implemented on a JADE platform, it looks like what is shown in Fig. 2. Highlighted are the three agents forming the Dispatcher framework, the most important of which, i.e. WSIG, will be described in detail in following sections. The other agents: the Remote Management Agent (RMA), the Agent Management Service (AMS) and the Directory Facilitator (DF) are part of JADE and hence control the overall functioning of the agent platform.

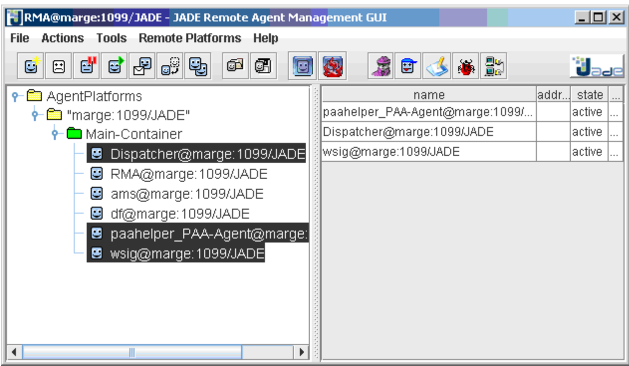

Fig. 2. Dispatcher framework

[13] http://jade.tilab.com/doc/tutorials/WSIG_Guide.pdf

[14] http://ws.apache.org/wsif/

4.1 The WSIG gateway

The WSIG gateway provides the crucial service of protocol conversion from ACL to SOAP and vice versa so that agents and Web Services can communicate. WSIG is a JADE add-on developed mainly to enable agents to communicate with Web Services in much the same way as they do with other agent services. The reverse is also true. However, as presently constituted, the WSIG platform does not quite satisfy the requirements of the Dispatcher because of its bias towards the Web Services' consumption of agents. In our case, the bias is more the other way around towards PAA agents' consumption of AMF Web Services, because the PAA, in quest of satisfying user needs, will at some point need the adaptation service provided by the AMF. So we proposed to enhance the capability of WSIG by providing Web Service invocation mechanism. As seen in Fig. 1, Web Service Invocation Framework (WSIF), which contains dynamic invocation modules, is added to the WSIG. Once the Dispatcher finds services advertised by the AMFs, the routing functionality of the Dispatcher is realised using a simple routing algorithm based on the names of the addressee to route messages to Web Services. Once the routing module decides and finds which AMF to use, the chosen AMF will be invoked and an adaptation request based on the ACL commands received from the PAA can be created and is then send to the WSIG module.

4.2 Negotiation of PAA, Dispatcher and AM

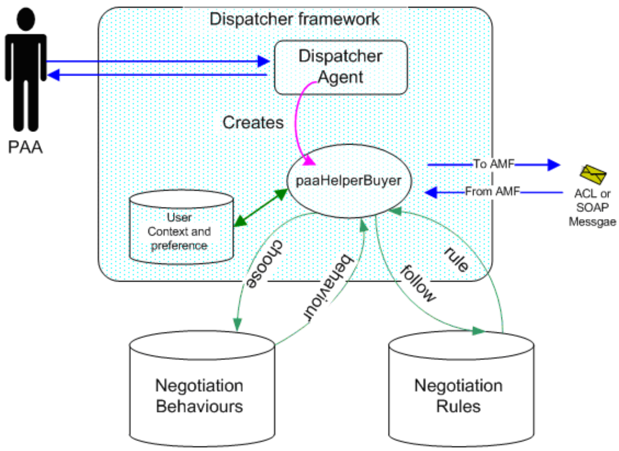

Fig. 3. Negotiation architecure

Fig. 3 shows the general negotiation architecture we implemented in Dispatcher. As shown, the PAA, acting on behalf of the user as a buyer agent, issues an adaptation request to the AM through the Dispatcher Framework. The Dispatcher Framework plays a number of crucial functions as a negotiation broker. Here is how it works. When the *Dispatcher Agent* receives a request from the PAA, it creates a *paaHelperAgent* to act for the particular PAA's request. In the creation of this helper agent, the *Dispatcher Agent* passes all context and user preference information received from the PAA to the helper agent. The Dispatcher Agent will search for *AMFs* advertised in the UDDI. After identifying the available *AMFs* that are capable and willing to perform the requested adaptation process, the *paaHelperAgent* will then begin to negotiate by sending an *"Ask"* or *"Call for proposal"* message to all the AMFs. To improve efficiency, in choosing the appropriate *AMFs* to negotiate with, the *paaHelperAgent* could consider peer-review of any *AMFs* based on previous negotiation and deal-execution behaviour. Lower ranked AMFs may be avoided in the negotiation process. Similarly the *AMFs* can also rank the paaHelperAgent that it can negotiate with. The *paaHelperAgent* accesses the behaviour repository to modify its negotiation behaviour. The implemented available behaviours follow the negotiation strategies described in the Section 3.1. The chosen behaviour could be statically used for a whole negotiation session or could be dynamically changed or varied as the negotiation progresses. In addition, the *paaHelperAgent* accesses a negotiation policy rules and ontology repository that is shared with the PAA, the *Dispatcher Agent* and the *AMFs*. This repository is implemented in a MySQL relational database. *paaHelperAgent* begins the negotiation by making an offer to an *AMF* for a given adaptation service with an initial price and cost constraints. The *AMF* responses with an agreement, a counter-offer or an outright rejection to the offer. This negotiation process proceeds back and forth until the specified prices or time are reached or more specifically when the utility expectations of both the buyer and seller agents are maximised. Fig. 4 shows the sequence diagram of the overall negotiation process.

5 DISPATCHER VALIDATION: A DEMO FROM THE USER PERSPECTIVE

The Dispatcher, as one of the comprising components of the Adaptation Management Framework (AMF), the solution sought from *content/service perspective* in the efforts to *"remove the barriers to ubiquitous services"*, will, together with other AMF components, serve as a service enabler behind any visible scenes to support ubiquitous services. It is a bridge connecting user's personal environment, represented by PAA, with the AMF. Therefore, to validate its functionality and the implementation we have presented in this paper, a use case scenario, a experimental setup and technically a user interface of PAA are required.

5.1 Use case scenarios

The following scenarios are defined to help demonstrating the AMF and hence the Dispatcher. The scenarios are extracted from a storyboard describing football fan

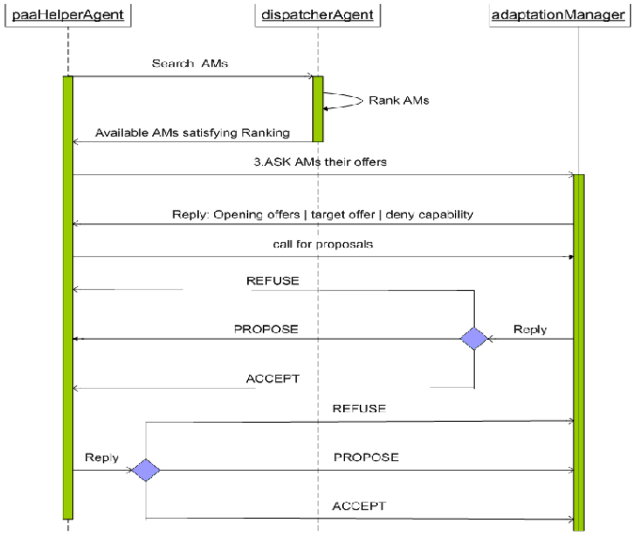

Fig. 4. Negotiation sequence diagram

Tom who is going to watch a match in the stadium after finishing a meeting in the office first. The scenarios start when the football match starts. The scenarios are titled with the reprehensive context when the scenarios take place and the corresponding adaptation services they trigger (in bracket). **Scenario 1: Small screen context (Video to Video adaptation)** Tom's meeting in the office overruns and the game has already begun in the stadium. Not the one to miss his favourite team's play, he sets his mobile phone to receive video content (by installing PAA) from the game's web portal. Since the content is meant for large screens but small ones like his, AMF is requested by his PAA to adapt video content to a suitable size and format for his phone. **Scenario 2: Voice only context (Video to Audio adaptation)** Tom dashes to his car to take the 20 minutes drive to the football stadium. Since he needs to concentrate on driving, he instructs his PAA to request for a video to audio adaptation of the game so that he can still listen to the game. **Scenario 3: No video/voice context (Video to Image adaptation)** Meanwhile, Tom has set his PAA to continually capture important scenes of the football game for later review and archiving. Because of the small memory capacity of the phone combined with the low battery life, Tom's PAA decides to capture only images of the important scenes of the football match. His PAA, after recognizing a goal or something else interesting, sets to automatically capture that and then asks the AMF to carry out a video to image adaptation. We have implemented the AMF to work with these scenarios, as can be seen in Fig. 6. However, we can only illustrate the whole lifecycle of one scenario, which will be described in Section 5.3.

5.2 Demonstration set-up

Four computers and one mobile device have been used for the demonstration as depicted by Fig. 5. The roles of the computers are as follows: AMF-1: this computer hosts, with Apache Tomcat Server, one copy of the AMF web service, that is one AM and one CA together. AMF-2: this computer runs another copy of the AMF web service. DA: this computer runs the Dispatcher, responsible for translating messages, negotiating with AMFs, routing adaptation requests and results between the PAA and the AMFs. UDDI server that registers the two AMFs is also run on this machine in this setup. ADMEs Content Server: this computer runs servers that host the content that can be requested by the user and ADaptation MEchanisms (ADMEs) that are concrete implementations of Adaptation Operations used by the CA to get content adapted. Take the video to image adaptation scenario for example, Adaptation Operation is video-image conversion and the ADMEs are converters like mpeg-jpg, 3gp-png with different parameter settings. For the mobile device, we use a Sony Ericssion phone with a Bluetooth link representing user's devices within his PDE and therefore have PAA installed.

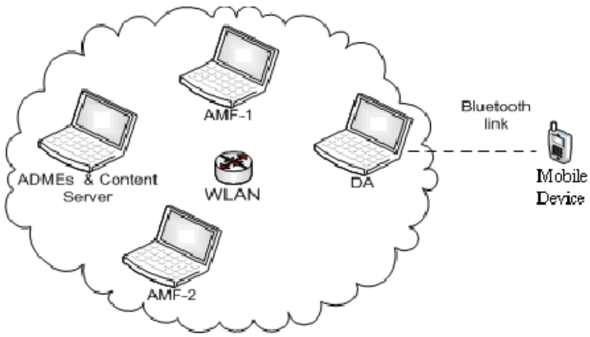

Fig. 5. Demonstration setup

5.3 Illustrations

From the *user perspective*, a majority, if not all, of the user's tasks are delegated to the PAA for autonomous process, whose implementations are not all presentable here. In this work, we only demonstrate a simplified version of PAA Graphical User Interface (GUI), as shown in Fig. 6 (a) and (b), which means it is not yet all autonomous and the user has to get involved in providing necessary information. Note that when in a full-fledged PAA implementation, such information should be autonomously given or detected. The first item the user chooses is the content to be fetched. This content location is given as a Uniform Resource Location (URI). Next the user makes a choice of factors that he considers as priority in the adaptation process and delivery of content. In the given example, only adaptation time and cost are taken into account. Other aspects that can be considered in this category include quality of service etc. Finally, the user's device context and other contextual information are provided in the last combo box of the GUI, which corresponds to the three scenarios defined above. After selecting the "small screen context" scenario, the PAA compiles an adaptation request and sends it to the Dispatcher when the users selects the "inquire" command located on the bottom left corner of the PAA user interface. After the Dispatcher runs the negotiation process based on time and cost, the adaptation results are displayed in Fig. 6 (c) and Fig.6 (d). Fig. 6 (c) shows the progress of the negotiation between the PAA helper agent and the AMs while Fig. 6 (d) shows the results after content being delivered. In Fig. 6 (c), we can see that there were two AMs involved in the negotiation, i.e. AM1 and AM2. At the end of negotiation, in this example, AM1 agrees to adapt the given content for a price of 12pence and in 15ms while AM2 indicates that it cannot adapt the content. After the user agrees to the adaptation cost and price constraints that are offered, the AM adapts the content to the preferred format or specification and returns a link of the adapted content to the Dispatcher. The Dispatcher then routes the content to the right device within user's PDE. In this case it is the device which originates the request.

6 CONCLUSIONS AND DISCUSSIONS

In this paper, we described our work in developing an agent-based middleware which plays an important role in paving the road to ubiquitous services. This middleware, named Dispatcher, forms part of the solution sought by the programme "removing the barrier to ubiquitous services" from content/service perspective. That is to develop an Adaptation Management Framework (AMF) which manages the complexity of dynamic and autonomous content adaptation. The Dispatcher, with no visible interfaces in itself, provides supports for the interaction between user environment, represented by PAA, and the AMF. The Dispatcher is developed on JADE agent platform. The Dispatcher implements negotiation protocols and interfaces to Web Services taking cognizance of the fact that the AMF were not implemented as agent but as Web Service. The Dispatcher enhances the JADE add-on implementation of interaction interfaces between agent and Web Service, to enable autonomic invocation of AMF Web Services. The Dispatcher is validated by using a PAA user interface to showcase how it works along with other components of the AMF to support dynamic content adaptation and thus serve as invisible enablers for ubiquitous services.

The agent-based approach to design such a middleware, though initially can be applied to any communication between agent platform and Web Service platform, have been enhanced with a number of specific features to meet the particular requirements of communications between user's personal environment with AMF Web Services in this work. In particular, its incorporation of negotiation protocol pro-

Fig. 6. PAA demo user interfaces

vides invaluable enhancements to the capability of agents as representatives of the user. In this work, only user's time and cost preferences of using the AMF services are the contextual information used by the Dispatcher for negotiations. Other information collected by the PAA, such as device capability, network condition, QoS etc, can also be used by the AMF to guide the adaptation process.

7 REFERENCES

REFERENCES

[1] Bickmore, T.W. and Schilit,B.N.: Digestor: Device-Independent Access to the World Wide Web. Proc. 6th World Wide Web Conf (WWW 6). ,1997, pp. 655–663.

[2] Hwang, Y., Jung, C., Kim, J. and Chung S.: WebAlchemist: A Web Transcoding System for Mobile Web Access in Handheld Devices. Proc. ITCom, 2001, pp. 37–46.

[3] Dunlop, J., Atkinson, R., Irvine, J. and Pearce, D.: A Personal Distributed Environment for Future Mobile Systems. Proc. of IST Mobile Summit, Aviero Portugal, 2003.

[4] Li, N. and Moessner,K.: The MVCE Management Framework for Context-aware Content and Service Adaptation. Proc. of the 7th International Workshop on Applications and Services in Wireless Networks, May, 2007.

[5] Shahidi, M., Attou, A. and Aghvami, H.: Content Adaptation: Requirements and Architecture. Proc. of the 10th International Conference on Information Integration and Web-based Applications Services (MoMM), Nov. 2008.

[6] Tarus, H., Bush, J., Irvine, J. and Dunlop,J.: Multi-Agent Mediated Electronic-Marketplace for Adaptation Services. Proc. of 5th Annual IEEE Consumer Communications Networking Conference. Harrah Las Vagas, USA. Jan. 2008.

[7] Lomuscio, A., Wooldridge, M. and Jennings, N. R.: A Classification Scheme for Negotiation in Electronic Commerce. In: Dignum, F. and Sierra, C. (Eds.): Agent-Mediated Electronic Commerce: A European AgentLink Perspective, Springer Verlag, 2001, pp.19-33.

[8] Walton, D. N. and Krabbe, E. C. W.: Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning. SUNY Press, Albany, NY, USA, 1995.

[9] Beer, M., d'Inverno, M., Luck, M., Jennings, N. R., Preist, C. and Schroeder, M.: Negotiations in Multi-Agent Systems. Knowledge Engineering Review, Vol. 14, No. 3, 1999, pp. 285-289.

[10] Ludwig, S. A., Kersten, G. E. and Huang, X.: Towards a Behavioural Agent-based Assistant for e-Negotiations. Proc. of the Montreal Conference of E-Technologies (MCETECH), 2006.

[11] Dubeck, M. B., Krishnamurthy, A. and Luong M.A.: Negotiation Strategies and Market Efficiency among Autonomous Negotiating Agents. Available online http://zoo.cs.yale.edu/classes/cs490/00-01b/dubeck.matthew.mbd22/.

[12] Blake, R. R. and Mouton, J. S.: the Managerial Grid. Houston: Gulf Publications. 1964.

[13] Zahreddine, W. and Mahmoud, Q.H.: Blending Web Services and Agents for Mobile Users. Proc. of ISADS Workshop on Cooperative Computing, Internetworking, and Assurance, Chengdu, China, April 2005.

8 BIOGRAPHIES

Ning Li is a research fellow in the Knowledge Media Institute at the Open University. She was previously a research fellow in the Center for Communication System Research at the University of Surrey. Contact her at n.li@open.ac.uk.

Hillary Tarus is an engineer at Communications Commission of Kenya. He was a PhD candidate in the Mobile Communications Group in the Department of Electronic Electrical Engineering at the University of Strathclyde. Contact him at Tarus@cck.go.ke.

Klaus Moessner is a professor in the Mobile Group of Centre for Communication System Research at the University of Surrey. Contact him at k.moessner@surrey.ac.uk

James M. Irvine is a lecture in the Mobile Communications Group in the Department of Electronic Electrical Engineering at the University of Strathclyde. Contact him at j.m.irvine@strath.ac.uk