Context-Aware Adaptation for Mobile Devices

Tayeb Lemlouma and Nabil Layaïda
WAM Project, INRIA, Zirst 655 Avenue de l'Europe – 38330, Montbonnot, Saint Martin, France
{Tayeb.Lemlouma, Nabil.Layaida}@inrialpes.fr

Abstract

This paper discusses the problem of content adaptation for mobile devices. The adaptation considers the context of the client and also the environment where the client request is received. A device independent model is defined and used in order to achieve automatic adaptation of the content based on its semantic and the capabilities of the target device. Our system includes a context description model and a client repository and offers device contexts management and querying functions. The proposed system uses the XQuery language to query the profiles and delivers the results in the form of SOAP services.

1. Introduction

In the recent years, mobile networking has been pushed by the emergence of an increasing number of new types of mobiles and embedded devices. The proliferation of this variety of devices such as mobile phones, Pocket PCs has created the need to provide these terminals with access to web content traditionally achieved on more powerful machines (laptops, desktop machines). Displaying a complex content on the mobile terminals, that has been initially authored for machines with high capabilities, represents a real challenge. In particular, if one needs to make a common infrastructure to serve such heterogeneous clients where small devices co-exist with more powerful machines. Some of the low-end devices have sometimes severe limitations and constraints (displaying capabilities, access mechanisms, user interfaces, etc.). Therefore, providing an adapted content to each type of terminal requires the consideration of several aspects that are involved in the adaptation and the delivery of the final content.

This paper discusses the problem of context-based adaptation for mobile devices. A device independent model is proposed and integrated in a prototype implementation called Negotiation and Adaptation Core (NAC) architecture [1]. NAC is a complete architecture that is designed for the content adaptation for such heterogeneous environments. It includes several structural and media transformations with a complete negotiation protocol for wireless networks [1]. The proposed model is device independent [2] and allows the adaptation of the server content while taking into account the semantic of the content itself and the client capabilities. We integrate in NAC a new semantic adaptation of the content. This type of adaptation addresses the layout of documents (based on the client display capabilities) and the automatic generation of hyperlinks between pages. In order to ensure the client context management and processing, we designed a profile repository. This repository allows the storage of mobile characteristic. The repository is accessed a set of SOAP services and uses the XQuery language to query the device profiles.

2. Context Description

The universal profiling schema (UPS) [3] is defined to serve as a universal model that provides a detailed description framework for different contexts. UPS is built on top of CC/PP and RDF [4]. Unlike CC/PP, UPS does not describe only the context of the client but it describes also all the entities that exist in the client environment and that can play a role in the adaptation chain from the content server to the target client.

UPS includes six schemas: the client profile schema, the client resources profile, the document instance profile, the adaptation method profile and the network profile. The description of the user is included in the client profile schema.

3. Context Extraction

The process of context extraction consists of instantiating the context variables and parameters (called generally *context dimensions*) for a client/server session. The instantiation is first step

toward the delivery of an adapted content that matches the current context instance. Evaluating the context dimensions can be done before or during the client request. Dynamic context dimensions (dimensions that may change from a session to another) are usually re-evaluated after each client request. The context dimensions can be instantiated from:

1. Profile repositories (see Sec. 8);
2. User updated variables (e.g. user preferences);
3. Real time evaluated dimensions (e.g. network speed, current connection protocol, etc.)

The real time evaluation allows at any moment to calculate the current value of a given dimension and update its value. Extracting the context dimensions from profiles requires the use of a query language that analyzes the profile and retrieves the value of a given dimension. In our approach, UPS [3] facilitates the context dimensions extraction by adopting the component principle of CC/PP [4] and extending it to all kinds of environment contexts. Thus the context processor can easily select the appropriate component according to the context that we need to evaluate. Example: The user preferences in UPS are included in the *cc/pp component* having *rdf:about = 'BrowserUA'*. The following query is an example of an XQuery expression that lists all the resources names of a server and which have as format *jpeg* and as a *ResourceSize* a value less than 3000:

```
let $resource =
Document("resources.rdf")/rdf:RDF/rdf:Description/
neg:Resource/rdf:Description[neg:ResourceFormat="jpeg"
and neg:ResourceSize<3000]/neg:ResourceName
```

4. Content Adaptation

An adaptation method transforms the content from a state to another in order to meet the constraints of the client context. In some situations, several adaptations are applied on the original content to reach that goal. In our model the adaptation tasks can be represented by a directed acyclic graph called *adaptation graph*. The adaptation graph includes paths from a piece of content *C* to the target client *U*. A path can be of type *A* (Fig. 1) if the original piece of content *C* can be delivered directly to the user. A path of type *B* is a path that contains other nodes M_i and means that we need to apply all the methods M_j in that order from the content node to the user node. The application of a path *p* is equivalent to the execution of the following algorithm:

(1) *content* = C;
(2) **from** i = 1 **to** |p | **do**
(3) *content* = M_i (*content*);
(4) deliver *content*;

(3) means that the content generated by an adaptation method M_i will be used by the method M_{i+1}. M_1 is always applied on the original content that we need to adapt. $M_{|p|}$ delivers its result directly to the end user.

Fig. 1. Adaptation paths

After receiving a client request and in the case where the original content does not match the requirement of the client context, the adaptation process searches for a content version that can be supported by the client. If no such version exists, the process tries to find a method that adapts the original content in order to meet the client constraints. If many versions match the client profile, the adaptation process chooses the variant having the smallest size in order to minimize the delivery time. The adaptation process extracts, from the environments (content, network, etc.), parameters and variables related to the the client context. The adaptation process tries to satisfy the client context parameters that are not satisfied by the original content. Thanks to the UPS adaptation profiles, the adaptation process applies adaptation methods that satisfy the client context parameters. The adapted content is then sent back to the client. If the adaptation process fails to find an adaptation method that satisfies some parameters related to the client capabilities, the adaptation process sends a negative answer to the client. If parameters that are not satisfied are related to the user preferences, the content is sent with the original parameters of the content. This strategy prevents inefficient exchanges for example sending a content that the client is unable to understand. If some context dimensions are not available in the profiles, the adaptation process tries to extract them using the available extraction methods. If the process fails to extract a parameter, a previously applied adaptation strategy is chosen. Again, if the parameter represents a client capability dimension a negative reply is sent, if it is a user preference the original content is sent.

5. Semantic Hierarchy Adaptation

By semantic content adaptation, we mean the content adaptation which depends on user preferences and context and on the semantic of the received content. We say also semantic hierarchy adaptation

when adapting the content according to its semantic and using a hierarchy structure of the original content. In our approach the hierarchy organization of the content concerns two aspects: the semantics of the content and the presentation organization of the delivered content. Different levels of hierarchy are defined in order to facilitate the adaptation of the content regarding the user interest and the client capabilities. For instance, the semantic hierarchy level of a section starts from the section title (higher in the hierarchy) then the section content itself. At the delivery phase, the presentation of the section content will depend on the client displaying and processing capabilities. The developed process of splitting the content into different presentation units [2], called content pagination, considers the client context dimension *page_max_length* which is instantiated from the delivery context and depends on the display size of the device screen.

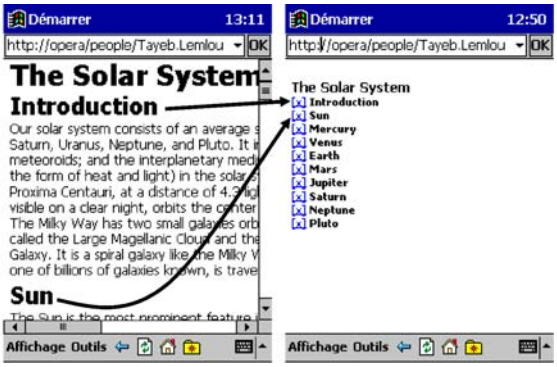
6. Device Independence Model

Most of the systems reported in the literature in the area of content adaptation for small devices consist of tailoring several presentations of the same content to different kind of devices [5][6]. The difficulty of such an approach is that it requires a lot of human efforts in authoring and storing the different content variants. Some studies tried to resolve this problem by providing mechanisms that allow a more dynamic adaptation starting from an exiting content model like HTML [7]. Unfortunately most of proposed models are device dependent and include already presentation-based information which is not suitable for all types of devices. Since the presentation of the content depends on the display capabilities of the client which are known only at delivery time, we adopted a more abstract approach by providing a device independent model which excludes presentation information from content. A device independent model [2] reduces human efforts regarding the content adaptation for different contexts. Indeed, such model makes easy to perform automatic and on demand adaptations on the original content when the authored content respects the universal accessibility rules [2].

The model takes advantage from existing models that use in their specifications device independent mechanisms (e.g. XHTML [8], SMIL [9]). In our model, a document is a set of sections where each section has a title and can be organized in several levels of detail. The section title represents the level zero of the section content. Any section of the document can use a media content (video, audio, image, etc.) by providing a *REF* element that is only a reference to the media and do not specify an absolute path to a unique media. A media reference denotes a set of media elements that represent the same information but with different levels of details and presentation richness. At the delivery step, the adaptation process delivers only the media that satisfies the delivery context. The model gives also the possibility to the author to specify a delivery context (in the form of a set of attributes and values, e.g. *bandwidth* = 512) associated to a media using a *SWITCH* element [9].

7. Content Pagination and Generation of Navigation Links

In order to apply an efficient content pagination, we do not use any data caching mechanisms to avoid the size explosion of cached content. The generated pieces of content store a link to a real time adaptation process located on the proxy side. The adaptation link is included in the generated piece of the content with some parameters (Fig. 2). The pagination process will use these parameters to apply the appropriate adaptation task: selecting the corresponding part of the content (e.g. a new section, the next section of the last viewed section on the device, etc.) and deliver the appropriate fragments of it (Fig. 3). The pagination process is executed when the user selects the link.



(a) **(b)**

Fig. 2. Semantic content extraction

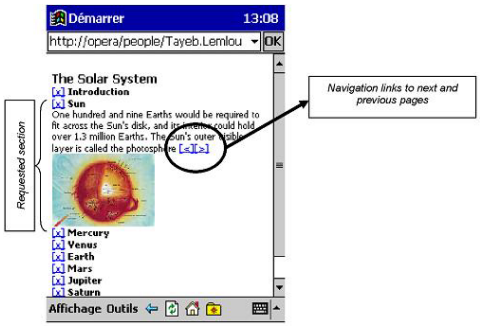


Fig. 3. Content pagination and links generation

8. Profiles Repository and SOAP services

A profiles repository is defined as a server that provides profiles in response of protocol-specific requests. In our system, it is implemented as a SOAP-based repository in the NAC architecture [1]. The new architecture is presented in figure 4. The proxy which is responsible to perform different adaptation methods on the original content is seen as a SOAP client. RPC calls are used to request the profiles from the repository in order to retrieve the device context related to the adaptation processing of the requested content.

In the current implementation, the repository contains 118 mobile device profiles written in UPS [3] and includes many recent device models [10]. The main SOAP services are presented in Table 1. The repository uses the XQuery language to handle contextual queries of the profiles and extract the device context according to the RPC call parameters. The received parameters correspond to query results. In the following, we show an example of RPC method invocation:

```
URL url = new URL (NAC_RPC_ROUTER); ..
call.setMethodName("getContextAtomicValue"); ..
Response resp = call.invoke (url, "" ); ..
Parameter result = resp.getReturnValue();
```

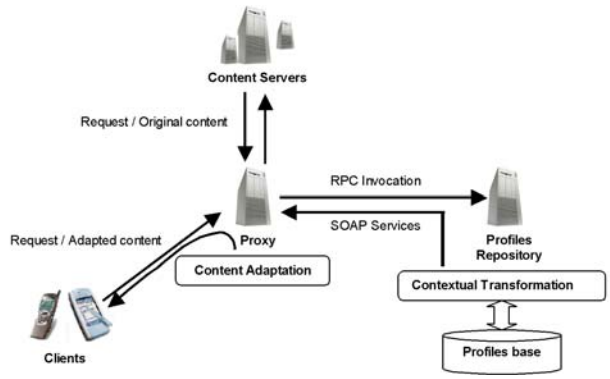


Fig. 4. The adaptation architecture

Table 1. Implemented methods of the repository

Exposed Methods	Parameters	Signification
getProfile	profileID	request a profile in the repository
getContextAtomic-Value	profileID CCPPComponentID contextEntity	request a value in a profile
getSubContext	profileID XPathExpression	request a subcontext in a profile
updateContext-AtomicValue	profileID CCPPComponentID contextEntity	update a value in a profile

The updateContext-AtomicValue allows at any moment to update the profiles if any change occurs in the user context.

Some adaptations using different contexts: The following figure shows the adaptation of the same SMIL document for two different contexts. Here the considered context dimension is the user *language*. The SMIL presentation is adapted for an English and a Russian user. (Fig. 5)

Fig. 5. SMIL adaptation

The following figures show the adaptation of HTML documents. Figure 6 shows the content as it is received by a desktop PC. Figure 7 shows the same content as it is received by a mobile phone that supports only WML documents.

Fig. 6. The content sent for a PC

Fig. 7. The content sent for a mobile phone

9. Experimentations

We carried out a set of experimentations in a platform with two network infrastructure types: a 802.11 wireless LAN and a wired network. Three kinds of devices are used to access the content: a Pocket PC (iPAQ 3600) running under Windows CE and connected through the wireless network, a laptop computer using both the wired and wireless connection and a personal computer that uses only the wired network. The Pocket PC is used to simulate different types of mobile devices. Most of the adaptation methods are implemented in JAVA for content transformation and using XSLT [11] and XQuery [12] for structural transformations. The SOAP repository uses the Apache implementation of SOAP [13] and the Tamino implementation [14] of the XQuery language.

The repository is a Compaq PC with an Intel Pentium 4 processor of 2.00 GHz and 266 MByte of main memory. Figure 2.a shows the received HTML content as it is displayed on the Pocket PC using a classical architecture. Figure 2.b shows the received content, authored using our model served dynamically to the device. Figure 3 shows the received content after a user interaction: only the requested part of the content is received and split according to the device capabilities.

In order to observe more precisely the behavior of our system, we have measured the performance of the repository regarding a) the time to process queries (XQuery) applied on profiles, b) the delivery time of the repository after the RPC calls and c) the delivery time regarding the different components of the same profile of a device (a Sony Ericsson T68R502 phone [10]). No particular grouping is considered regarding the mobile devices axis of figures 8 and 9.

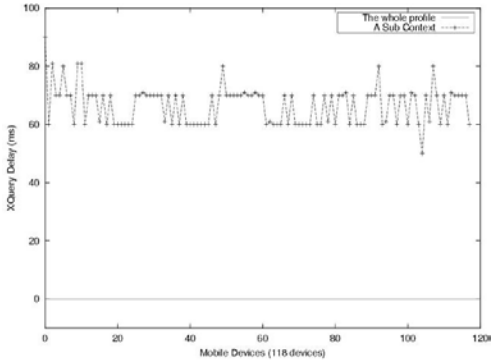


Fig. 8. Performance of XQuery application

Figure 8 gives the measurements of the delay time required to apply XQuery requests after the reception of two kinds of SOAP RPC calls: *GetProfile* and *GetSubContext*. The sub context considered was the *OnlySupportedResources* [3] of each UPS profile. The measured delivery time corresponds to the time taken by the repository to deliver a context to the NAC proxy, it does not include the time taken for content transformation. The null query delay of the whole profiles is presented in order to show that when a SOAP client requests the whole profile, no XQuery method processing is necessary. Therefore, the time of profile query is null. However, to request a sub context, the repository instantiates and evaluates an XQuery to select some parts of the profile context.

Figure 9 gives the measurements concerning the delivery time of the whole profiles and the *OnlySupportedResources* [3] sub contexts. The measurements concern 10 RPC calls of types: *GetProfile* to request the whole profile and *GetSubContext* to request the *OnlySupportedResources* part of the profile.

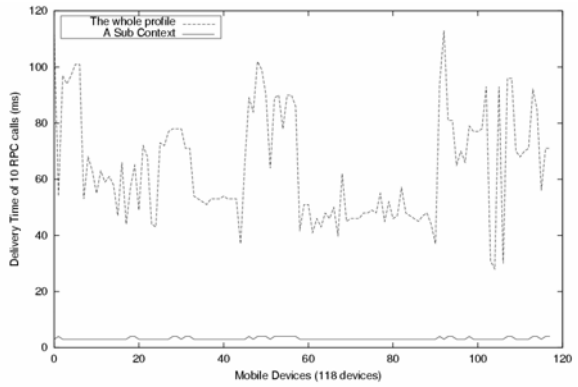


Fig. 9. Performance of the RPC calls from the proxy to the repository

The figure shows that the repository takes more time to deliver the whole profiles when compared to applying XQuery and delivering only the part of the device context that has a direct relation with the content adaptation.

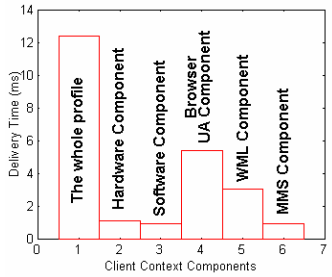


Fig. 10. The delivery time of the different profile components

Figure 10 compares the delivery time of the whole profile of a device with the delivery time of its different context components. These experiments show that for an efficient content adaptation, the UPS structure based on the CC/PP component principle should be used. This confirms our initial supposition that selecting only the component that may influence the delivery of a requested content is more efficient even if we have to evaluate some queries.

10. Conclusions

In this paper, we presented a context-based adaptation infrastructure for mobile devices. As the adaptation is applied on the content, we have proposed a device independent content model that facilitates the adaptation processes and opens the doors for a genuine universal content delivery in different contexts. The model considers the adaptation at the presentation level based on an important aspect related to the content semantics as a hierarchy of elements. In addition, we have taken advantage of the UPS description and Web services in the context management and profiles querying. As we have shown, going toward a device independent content model facilitates the generation of adapted presentations for small devices. Furthermore, querying and exchanging only context fragments with a direct relation with content adaptation can really increase the performance of the global system.

11. References

[1] T. Lemlouma, N. Layaïda, Adapted Content Delivery for Different Contexts, SAINT 2003 Conference, Orlando, Florida, USA, January 27-31, 2003. IEEE Computer Society Publication

[2] Work in progress with the Device Independence Working Group of the W3C

[3] T. Lemlouma and N. Layaïda, Universal Profiling for Content Negotiation and Adaptation in Heterogeneous Environments. W3C Workshop on Delivery Context, W3C INRIA Sophia-Antipolis, France, 4-5 March 2002.

[4] Graham Klyne et al, Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies, http://www.w3.org/TR/CCPP-struct-vocab/, W3C Working Draft, 15 March 2001

[5] Pocket Info. http://www.pocketinfo.nl/

[6] InfoRover Application, http://www.pendragon-software.com/

[7] K. K. Hoi, D. L. Lee and J. Xu, Document Visualization on Small Displays, Mobile Data Management, pp 262-278, Melbourne, Australia, 2003

[8] XHTML 1.0, W3C Recommendation, http://www.w3.org/TR/xhtml1/

[9] SMIL 2.0, W3C Recommendation, http://www.w3.org/TR/smil20/

[10] T. Lemlouma, UPS Client Repository , http://wam.inrialpes.fr/people/lemlouma/MULTIMEDIA/UPS-Client-Repository/ups-client-repository.html

[11] XSL Transformations (XSLT) Version 1.0, W3C Recommendation, http://www.w3.org/TR/xslt

[12] XQuery 1.0: An XML Query Language. W3C Working Draft, http://www.W3.org/TR/xquery, 02 May 2003.

[13]Apache SOAP v2.3.1, http://ws.apache.org/soap/

[14] QuiP Version 2.2.1 , a W3C XQuery Prototype, http://developer.softwareag.com/tamino/-quip/