THE NEGOTIATION OF MULTIMEDIA CONTENT SERVICES IN HETEROGENEOUS ENVIRONMENTS

TAYEB LEMLOUMA AND NABIL LAYAïDA
OPERA Project, INRIA Rhône-Alpes
ZIRST-655 Avenue de l'Europe -38330 Montbonnot Saint Martin France.
Phone: +33 4 76 61 52 81 / +33 4 76 61 53 84 Fax: +33 4 76 61 52 07
E-mail: Tayeb.Lemlouma@inrialpes.fr, Nabil.Layaida@inrialpes.fr

Few number of works have discussed the problem of multimedia services negotiation in heterogeneous environments. This paper presents a new negotiation strategy of services in such environments. This strategy is based on the concept of profiling and document selection. We show how CC/PP-RDF model can be used to describe clients preferences and capabilities, and how SMIL 2.0 modularization can serve as a basis to ensure services profile definition. We give the rules to follow in order to achieve the negotiation task. The negotiation protocol that we propose uses a method of profiling evaluation that we call the *TL evaluation*. We show how our protocol can respond efficiently to a best deliverance of adapted multimedia services for the end users, while taking into account device limitations and user preferences.

1 Introduction

Heterogeneous multimedia environments are characterized by the presence of a wide variety of client devices, which are very heterogeneous with respect to their capacity in terms of memory, CPU or display, etc. A device can be a workstation, laptop, WAP phone, PDA or any other terminal. In order to offer services and satisfy the needs of such different terminal types, the multimedia system must be adaptable to obey users and environment constraints: user interests or preferences, terminal capabilities, service requirements, characteristics of the network and location information. A negotiation mechanisms must be designed to enable interactions between servers of content and users. The goal of this operation is to make these parties reach an agreement (a consensus) on the best service to be delivered.

Few numbers of works have explored negotiation and adaptation frameworks for such systems. In the present paper, the adaptable multimedia systems is discussed in a general way, which means that the considered architecture isn't depending specifically on any particular kind of devices or networks. This approach permits to give general solutions deployable at a very large scale, for example, the web.

Providing a global architecture that ensures negotiable multimedia services requires different components: document and user profiling specification, transformation and adaptation methods, client/server negotiation mechanisms and rules followed by these mechanisms represent the main components of the aimed architecture.

As we will see later in the paper, the XML model [23] and related tools represent the basis of our framework. It provides many advantages such as the clean separation between the content and its presentation and a common representation that can be communicated between different platforms and system entities. CC/PP [9] and RDF [14] can guarantee a good description of the environment capacities and preferences. Document profiling can be nicely achieved using an extensible mechanism like the SMIL Basic profile [18]. This profile can be augmented by the incorporation of new modules of the super set defined by the SMIL 2.0 profile [19]. Transformation mechanisms, based on but not limited to XSLT [24], represent an efficient manner to ensure the content transformation in such architectures.

The central aspect of our paper is a negotiation strategy that can be executed in order to guide the adaptation process and documents selection. The negotiation is based on an algorithm that uses a new method of profiles selection, that we call the *TL evaluation*. The later permits to return an ordered set of profiles that can respond to the client demand according to its needs and preferences.

The paper is organized in seven sections. In the second section, we discuss the profiling concept and we show its importance in describing the context of services negotiation. The concept is discussed from the client and the document side. In the third section, we introduce the modularization principle introduced in SMIL. Section 4 introduces the content negotiation task and its requirements. Section 5 discusses the proposed negotiation strategy. In section 6, we propose our negotiation protocol and we introduce a new method of profiles selection. We show how our protocol helps to make server and users agree on the best adapted services. In section 7, we give a conclusion of our work and present some future work.

2 Universal profiling

The first step, which precedes any negotiation process, is ensuring a good description of the environment. Client/server interactions in a negotiation need the complete description of their mutual constraints in order to identify the best service. This description must include all the environment components, i.e. services, terminals, users, networks etc. The description covers two sides: Hardware and Software. In the hardware side, a device augments the constraint set with its physical capacities and capabilities. For example with its screen display resolution, etc. In the software side, the user agent is described in terms of supported functionality; for example a user agent can support a particular type of language or protocol. Note that hardware and software descriptions may be dependent, consequently any adaptation process must take into account both aspects. For example, consider a Screen-Phone accessing to the web through a browser with a screen size of '240x320 and 8 colors', and 'the browser supports only GIF and JPEG image formats but doesn't support scrolling'. In that case, the browser can't display a GIF (or JPEG) image directly without applying a size transformation to fit the hardware constraint.

2.1 User profiling

The user profile contains a description of the resources and capabilities of the user. Following the web standards, a good solution is to use CC/PP [9], which is based on the RDF meta data description model [14]. A CC/PP profile describes client capabilities and preferences in terms of attributes for each component. An example of a profile, will be the one including the following essential components: hardware platform, software platform and user application. With CC/PP, model extensibility is always possible, since RDF permits the modeling of a wide range of data structures. However it is recommended to avoid complex data models for profile attribute values.

2.2 CC/PP Profile structure

Handling profiles requires to conform a given syntax that will help profile processors to achieve their tasks properly. Following the CC/PP model, a profile is composed of a number of components, represented by resources of type 'ccpp:component' and related to the client profile by a 'ccpp:component' property. The type of a component may be indicated by a 'rdf:type' property, or equivalent RDF structure. Attributes are represented as 'named properties', linking a 'subject resource' to an associated 'object resource' or literal value. It's important to note that RDF offers the possibility to indicate a resource description in a separate document. In this case, the document is identified using 'Uniform Resource Identifiers' (URIs)[2]. For example, a resource's default property may be specified using a URI reference. In this case, the URI part of the default resource identifier is used to retrieve an RDF document containing the default resource description. Thus, if the default resource is named 'http://Company-resources.fr/WAPDeviceProfile#HardwarePlatform', the URI 'http://Company-resources.fr/WAPDeviceProfile' is used to retrieve an RDF document, and the resource identified -locally- by #HardwarePlatform within that document, is taken as the default resource. In such

case, the resource might be defined within the target document using: about='http://Company-resources.fr/WAPDeviceProfile#HardwarePlatform' or ID='HardwarePlatform'.
Examples of a CC/PP profile in the form of an RDF graph notation can be found in [9].

2.3 Document profiling

The concept of a document profile is complementary to the user profile. A document profile specifies the syntax and semantics of a document or a collection of documents (DTD or document type). Document profiling allows easy handling and processing of the specified group of documents. The profile specification gives a detailed description about the appropriate type, for example in terms of supported image formats, scripting level, etc. The central aspect of this approach is the definition of the elementary functionality required for the rendering of multimedia documents. Once this is step is achieved, super sets -built based on the elementary predefined elements- can thus be described in terms of functionality: 1) What does exactly the user agent supports and, 2) What does exactly the server of content offer.

The first point concerns the client side. Here the user agent (or which is commonly known as a player) can be designed in a way that guarantees the best client use of the served content. The second point concerns the server side, it requires a complete description of all the content versions potentially useful for the client. This description helps to guide the content delivery based on the global picture of the client capacities and the available content services. This principle allows determining, if necessary, the content transformations which allow to create custom documents with the demanded set of functionalities.

3 The modularization principle

The main goal of the modularization is to move, from monolithic languages such as older versions of HTML, to more flexible and scalable languages adaptable to a variety of constraints. The modularization permits to support new devices and applications, by defining subsets of modules and recombining them. For example we can provide services for a WAP phone which may only support a subset of predefined modules. The modularization doesn't concern only the server's side (services). The modules or functionality supported by a device must also be defined in order to guide the server identifying the best content to be delivered.

The 'SMIL 2.0' approach provides good descriptions of abstract functionality collections, either on the client side or on the server one. SMIL 2.0 [20] offers a scalable framework for encoding several aspects of multimedia presentations. It specifies what media items will be presented, when, where and how. It also accounts for whom the media is played and, the most important, how the presentation is adapted for different end users on different contexts. This last point can be ensured, for instance, using the switch element and the test attributes. So we can specify that one or zero selections are to be made among a set of alternatives for inclusion at individual locations in the temporal hierarchy of a SMIL document[a] [20,16]. Alternatives can be used, for example, to give different formats of a media object according to client preferences and capabilities. In addition, some of the document alternatives related to the environment can be evaluated on the server and the result sent to the client. This allows to reduce the the document size and prevents the user agent from exploring some of the switch branches.

In addition to using SMIL to encode rich multimedia content, the modularization of SMIL [19,18] permits to cover and describe the diversity of the environment functionality, which allows easy design of negotiation strategies. The use of the SMIL modules has already gained an important acceptance, such as in the third Generation Partnership Project (3GPP)[8].

3.1 The SMIL profiling model

The importance of SMIL model is that it allows the definition of a collection of individual modules. Each module (or functionality) can be incorporated into other XML languages. This makes SMIL facilities usable in different types of documents and presentations. SMIL [20] defines a module as a collection of semantically related XML elements, attributes and attribute values that represents a unit of functionality. This definition allows, by composition, the creation of new languages, called language profiles, based on the combination of atomic modules. The same principle allows the servers to adapt the content when providing services. This approach requires a complete description of the supported functionality in the multimedia environment. The SMIL 2.0 specification responds to these needs through a naming scheme of its modules (see 'The SMIL 2.0 modules' section of [20]). SMIL 2.0 modules may be independent, or related by dependency relations. For example, 'BasicLayout' and 'BasicLinking' are independent modules but 'BasicAnimation' module depends on the 'BasicLayout' module.

---

[a] an alternative approach of the SMIL 'switch', called CMIF Channel model, is presented in [3], but is equivalent to a set of 'switch' statements.

A language profile must include all the modules on which depends other included modules (figure 3.1)). New languages must be build on the base of this constraint. According to the specification of [19], each module is named with a unique identifier associated with it, for instance the identifier of the 'Structure' module is given as: 'http://www.w3.org/2000/SMIL20/CR/Structure'.

In the context of heterogeneous environments a common description of the different constraints is important. Generic descriptions enables the construction of common building blocks for user agents and document descriptions. Furthermore, it allows -when respected during document authoring- to obtain more interoperability since documents are initially created to be used under different platforms[b]. The scalability framework defined for the SMIL Basic Language profile [18] can be efficiently used to reach this goal.

The SMIL Basic makes multimedia authoring available for a large number of web users and not especially those running under a 'rich media' context. The SMIL Basic profile consists of a reduced subset of the full SMIL modules[c]. It may be supported by a wide variety of SMIL players even those running, for instance, on mobile devices with limited resources such as : small displays, limited network connectivity, limited input methods, etc. SMIL Basic has been recently adopted by the 3GPP Consortium in the scene descriptions of the PSS [d]clients and servers [8]. Indeed PSS SMIL collection, includes the SMIL 2.0 Basic language profile plus three additional modules.

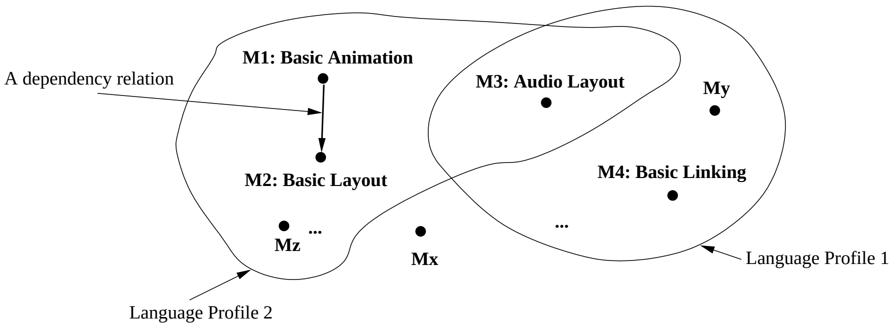

Figure 3.1: SMIL profiling

[b]for more details see the notion of language profile conformance in [19]: 'The SMIL 2.0 modules' section.
[c]conformance between SMIL Basic and SMIL Language 2.0 holds.
[d]Packet-switched Streaming Service.

A typical use of our architecture can be described by: −A client that supports a SMIL player, and −A server that provides content authored following the SMIL Basic model. The last requirement means that the content must be either authored in the SMIL Basic manner or transformed into it.

The goal of the negotiation is to ensure a form of consensus between servers and demanders. The server of services represents multimedia content servers or intermediate proxies acting as adaptable content servers. Demanders represent the set of clients and the proxies. From the client perspective, the global architecture of the multimedia system can be seen as a black box which admits as inputs client requests, and outputs services. It contains mainly two tasks: the negotiation task and the adaptation task. Requests can be exchanged between the client and the server in order to achieve the negotiation step. The final choice of the adapted service is the result of the negotiation step (figure 3.1)).

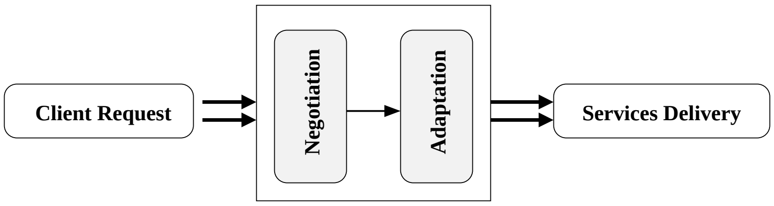

Figure 3.1: A global view of service adaptation and negotiation architecture

Ensuring a complete negotiation architecture means:

1. To take into account the diversity of users capacities and also preferences (for example an XML web page requirements) and to specify and apply rules allowing to match these requirements.

2. To determine target formats of services in term of selected set of modules.

3. To determine the transformations to achieve (how to transform and which ones to apply).

4. To support user agents constraint changes.

An efficient negotiation process, requires knowing many details about the target schema. However, we think that negotiation solutions must use a common set of concepts and tools independently from the target environment under which it will be applied.

3.2 *Negotiation requirements*

One negotiation approach is based on the document selection principle. In this approach, documents must be authored in many versions, for instance with different multi-lingual support or in different levels of abstractions. Here the multi-authoring depends on the diversity of client constraints. Document selection consists to choose the nearest document in terms of constraints satisfaction. This presupposes that document functionality are made explicit and requires to know the existing diversity of clients when the documents are first authored. Indeed, this simple manner to achieve services negotiation[e] can be used in systems where users diversity and the documents volume are very limited. After all, authored once services, without an explicit functionality definition is the most deployed solution in the Internet today. Multi-authoring puts a heavy burden on document authors and is to avoid when possible.

The negotiation process requires mainly three elements:

1. A complete description of the server content (document profiling).

2. A complete description of the meta data of the client: the hardware platform, the system software and applications used by the user agent (user profiling).

3. The negotiation protocol, or the set of rules which must be followed in content adaptation to satisfy user constraints. To satisfy the user requests, negotiation rules apply particular mechanisms to adapt or/and to select the appropriate document version.

4 The negotiation strategy

The role of the negotiation is to make user agents and servers agree on the final service following a client request. We think that developing an independent layer, responsible for achieving negotiation rules, is a better solution than incorporating it into a particular application. This allows a transparent content delivery to the user agent and common negotiation layer for different players on the same client (eg. an HTML browser). In this section we describe the negotiation strategy. Solutions and descriptions may change from a particular environment to another, this is why we try to give a global strategy that ties generalities and details at the same time.

[e] in fact, we haven't here a negotiation with the proper meaning of the term.

4.1 *Basic steps preceding invocation of the negotiation process*

The negotiation strategy requires achieving some steps before its invocation. The concepts that we have discussed serve as tools to launch properly the negotiation process. We define the global architecture as a tuple S=(UA_P,T,D_P) where:

- UA_P is the user agent set of profiles (we suppose that we have one user agent, if there are more, other sets must be created)

- T is the set of the adaptation methods (transformations) used by the server: T outputs as a result the D_P component.

- D_P is the set of document profiles existing on the server: i.e. the description of content documents in term of supported modules or functionality. According to the negotiation strategy, this set can be fixed or variable over time.

A transformation method 't' that exists in the server side can be defined by its output document profiles when it is applied. T can be implemented using XSLT processors plus style sheets collection. The formula 't(x)=y' means: after applying the 't' transformation on the content x, we obtain as a result a document that have 'y' as its document profile.

Our negotiation strategy defines the rules to apply when some steps are achieved. The main steps are:

- Preparing style sheets that can be used, this depends on the particular characteristics of the considered multimedia system, i.e. handled documents, network constraint, user preferences, etc.

- Ensuring an XSLT transformation processor.

- Preparing profiles of the content server potentially requested by the different clients.

- Preparing user agents sets of profiles.

As introduced earlier, profiles are described using CC/PP and RDF and the SMIL modularization model.

4.2 Kinds of negotiation strategies

According to the invocation of the adaptation process, we can distinguish two kinds of negotiation[f]:

1/ **On-demand negotiation**: In this kind, the adaptation process is invoked upon the reception of the user agent request. Results of the adaptation are directly transmitted to the user agent. In this case, document profiles set existing on the server side doesn't change, formally we can say that D_P is stable.

2/ **Cached negotiation**: Unlike the first kind, cached negotiation uses already produced adapted documents versions. This means that on the reception of a user agent request, the server doesn't invoke an adaptation on the fly. It looks first for a stored version (including the original version) of the demanded document[g]. If the suitable version doesn't exist, the server invokes the adaptation process, using the best transformation. The adapted document is then cached for future uses.

A document version can be identified by the tuple <Document Identifier, Document Profile Identifier>, where the Document Identifier denotes[h] the original version of the document. Document Profile Identifier denotes the set of document modules in term of atomic supported functionality. Note that the received value of Document Profile Identifier can be the same for different clients, which means simply that these clients are intended to use the document in the same context. Moreover, the Document Profile Identifier value can be the same for different documents.

Cached negotiation is preferable if the use of a same document is frequent. However if the client context changes frequently on-demand negotiation will be more suitable. Mixed strategies can be adopted according to: user requests trace and frequency, the client context changes, etc. Other techniques can be used, such as: cache techniques, timing out for saved profiles, counters, etc.

4.3 The proposed approach

The key idea for ensuring an efficient negotiation strategy lies in making the best effort to:

[f] this classification is made on the basis of adaptation initialisation and server behavior, and not on the basis of context changes.
[g] profiles search depend on the adopted approach in the profiles acquisition and distribution, this constitutes an other aspect to be studied.
[h] using a value, a pointer, etc.

1. Find the appropriate content (a document or an existing document version if it exists); or
2. The transformation method permitting to create such content.

The served content must satisfy the user agent preferences and take into account the complete set of constrains presented in the global environment. Making a best effort means identifying the process allowing to produce an optimal content, even if the selected content can not be provided after applying another negotiation strategy.

At the time of this writing, it was difficult to compare our solution with other negotiation strategies as we did not find any in the literature. [4] shows how CC/PP and RDF can be used to guarantee powerful descriptions necessary for the content negotiation, however no negotiation rules are detailed. This notion consists of applying the translation of a profile into another. When this is achievable, we can translate either of the client or server profiles to the other, or both to a common format. However [5] gives just general descriptions and no specific methods to achieve such translations are discussed.

We try to give in the following part, a new method of negotiation that can be used in a heterogeneous multimedia system. We propose a set of rules to be applied after achieving the steps introduced in section 5.1. Our strategy is based on the following idea:

"The negotiation strategy must provide at the end step, a service which doesn't support more than the atomic functionality set supported by the client, as it is described in its profile".

To understand this simple -but very important- principle, we consider the following example. Suppose that the client supports {X,Y,Z} set of atomic functionality. We suppose that in the server side, existing documents have {Y}, {X,Y} and {X,Y,Z,T} document profiles. It's clear that the use of the {X,Y,Z,T} profile can cause errors, because the 'T' functionality is not supported by the client. Moreover, the use of {Y} profile isn't preferable because another profile, which covers larger supported functionality, exists. The best negotiation strategy will output as a result the document having the profile: {X,Y}. The problem is how to make profiles selections automatic, i.e. achieved by a server process, thought that in some cases we haven't all the available possible profiles. Indeed some profiles may exist only after applying transformation methods supported by the server (the T set).

Before we start describing the global negotiation strategy, we give briefly a description of the fields used in the request formats. We don't provide here the details of the requests structure (which can be done nicely by description

tools like XML/RDF based mechanisms), we just list the important fields related to the negotiation task.
We have four kind of requests: client request, client reply, server request and server reply
We propose that requests must include, at least, the following fields:
Client request:
<Client Identifier, Server Identifier, Document Identifier, User Agent Supported Functionality, User Agent Preferred Functionality>
Client reply:
<Client Identifier, Server Identifier, Document Identifier, Selected Profile>
Server Request:
<Client Identifier, Server Identifier, Document Identifier, Profiles Set>
Server reply:
<Client Identifier, Server Identifier, Document>

The User Agent Preferred Functionality describe the user preferences to apply on the requested document. The content of the field may be different than the User Agent Supported Functionality field. For instance, a client browser may support AVI and GIF animations (User Agent Supported Functionality), but prefers to receive a GIF animation format (User Agent Preferred Functionality). The main difference between the two fields is that we can permit to deliver a service supporting a functionality included in the User Agent Supported set but not preferred by the user. The opposite is, of course, impossible to achieve. For example: Consider a client that can run under two different software platforms P1 et P2 using the same browser B. The browser supports M1, M2 and M3 under P1 but only M2 and M3 under P2. User Agent Preferred Functionality = M1, M3. If the client runs under P1, a service supporting M2 can be delivered, even thought that M2 is not preferred. If the platform used is P2, a service supporting M1 must not be delivered, even thought the client prefers it. Another solution can consist in sending only the User Agent Supported Functionality set to the server. This solution imposes more restrictions to the negotiation strategy on the client side, moreover the User Agent Preferred Functionality set can be used to define a priority order between a subset of the client preferred and supported functionality.

5 The negotiation protocol

For the purposes of the proposed solution, the following abbreviations are used:

CI Client Identifier.

SI	Server Identifier.
DI	Document Identifier.
P_S	Profiles Set, used to declare and to find the supported document versions.
CC_P	Client Constraints Profiles, an additional set of constraints, that allow the client to select when many choices are proposed by the content server, this set of profiles may be empty and updated any time.
SC_P	Server Constraints Profiles, the same role of CC_P but in the server side. It concerns services delivery.
SP	Selected Profile, the selected profile by the client.
D	Document, denotes the final adapted document provided by the server.
US	User agent Supported functionality, or modules.
UP	User agent Preferred functionality.
USP	User agent Supported and Preferred functionality.
T_S	Set of profiles that can exist after applying transformation methods of the T set.

a) The TL evaluation

The negotiation strategy that we propose in this document uses a new notion that we call Tailored Levels priorities evaluation, or in short TL evaluation. The necessity of ensuring such mechanism lies in the fact that the negotiation strategies must execute several steps of profile selections. The criteria of these selections are not trivial and, sometimes, not predefined efficiently, which can not be described with a simple formula or a short selective code instructions.

Our approach consists to tailor a level of priority that guides selections in a predefined collection of sets (set of profiles, in our context). The priority levels are tailored and processed with the same method, which allows the reuse of the same mechanism in different use cases. In the proposed solution, we distinguish two use cases of the TL evaluation principle. The module, which is responsible to achieve selections, is called: a TL evaluator.

A TL evaluator admits as input: a set of elements I, and a collection of sets CS. It outputs a set of elements O, where the elements are ordered in a predefined priority. An order must be predefined between the CS elements (which are sets). To well understand how the TL mechanism works, we discuss the different use cases existing in our negotiation strategy.

Use case 1: General form: O = **TL_evaluator**(I,CS);
Example: P_S= **TL_evaluator**(US,D_P);

Here, the input set of elements (I), is represented by the US set, (i.e. the user agent set which contains supported atomic modules). The target collection of sets (CS) -in where we must find the best set, or collection of sets, according to a predefined criteria- is represented by D_P, i.e. the set of document profiles existing in the server side. The output set O, here P_S, will contain the wanted document profiles ordered by priority. The predefined priority of selecting is represented as follows:

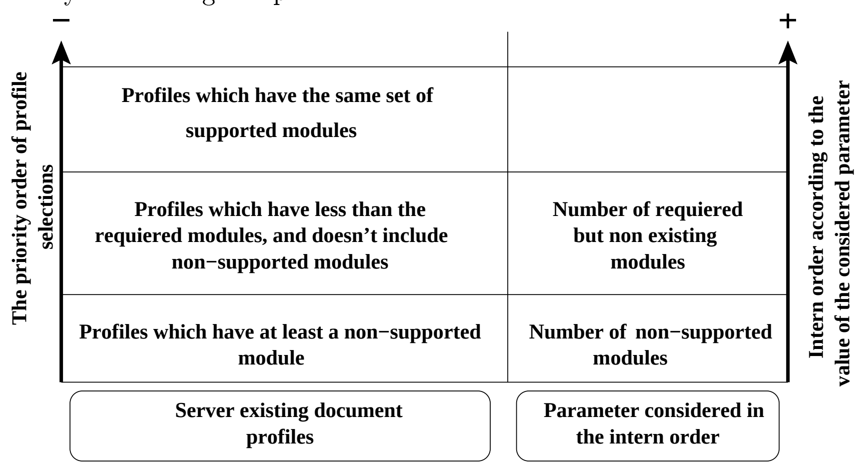

Figure 5: The predefined priority of selecting

Profiles are ordered according to the global priority order into three collections. Inside of each collection, profiles are ordered according to another considered parameter (figure 5)). In our solution, TL evaluator must not, in this use case, include profiles belonging to the low level of priority. This means that profiles, which have at least a non-supported module according to US, are not included in P_S. Note that the evaluation may return an empty set.

Consider the following example:

Let US = {Y,T,U} and
D_P = {{X,W},{T},{X,Y,Z,T,U},{Y,U},{X,U,T},{T,U,Y},{Y,T,X},{T,Y}}.
According to the predefined priority order, profiles are ordered as follow:
First collection (low priority): {X,W}: intern order parameter = 2, {X,Y,Z,T,U}: intern order parameter = 2, {X,U,T}: intern order parameter = 1, {Y,T,X}: intern order parameter = 1.
Second collection (medium priority): {T}: intern order parameter = 2, {Y,U}: intern order parameter = 1, {T,Y}: intern order parameter =1.
Third collection (high priority): {T,U,Y}.

After applying TL evaluator on the server document profiles, P_S will contain the following elements in this increasing order:
P_S = {{T,U,Y},{Y,U},{T,Y},{T}}, which respond correctly to the negotiation principle.

This use case is the same (the same principle and the same predefined global and intern order of priority), as: P_S= **TL_evaluator**(US,T_set). Here T_set represents all the profiles that can be obtained after applying T methods.

Use case 2: General form: O = **TL_evaluator**(CS,I);
Example: P_S = **TL_evaluator**(P_S,USP);

In this use case, the TL evaluator admits as input a collection of sets CS, in the example P_S, and a set of elements, in the example USP (the user agent supported and preferred modules profile). Here, the principle is very simple, it consists in ordering CS according to the elements existing in I. In the example, profiles found in P_S, are ordered according to user preferred and supported modules. In this use case, the priority order can be given in two manners:

1. If the preferences, UP, represent constraints intended to be only added to the constraints set included in US. The priority parameter equal to the number of preferred modules, and the consideration order is increasing.

2. If the preferences describe all, and only all the preferred modules, the priority is considered using two parameters: a) The number of preferred modules with an increasing order. b) The number of the non-preferred modules, but with the decreasing order. The final way to mix these two parameters is left to the users. The simplest formula will be the first parameter minus the second.

Consider the previous example discussed in the first use case.
If USP = {Y,T} for example, the profiles of P_S will be ordered -after applying
P_S = **TL_evaluator**(P_S,USP)- as follows:

- Manner 1:

  {T,U,Y} : priority parameter = 2
  {T,Y} : priority parameter = 2
  {Y,U} : priority parameter = 1
  {T} : priority parameter =1
  and thus P_S={{T,U,Y},{T,Y},{Y,U},{T}}.

- Manner 2: (priority parameter = number of module preferred - number of module non-preferred)

  {T,Y} : priority parameter = 2-0 =2
  {T,U,Y} : priority parameter = 2-1 =1
  {Y,U} : priority parameter = 1-1 =0
  {T} : priority parameter = 1-0=1
  and thus P_S = {{T,Y},{T,U,Y},{T},{Y,U}}.

  This use case is the same (the same principle and the same predefined global and intern order of priority), as : P_S = **TL_evaluator**(P_S,SC_P) and **TL_evaluator**(P_S,CC_P). SC_P and CC_P represent sets of additional constraints made by the server or the client to guide more the selecting process. This additional selecting is optional and can be omitted. The priority parameter is calculated according the satisfaction of the presented constraint by each module.

b) Negotiation protocol

1- Client side: Here is the client code part: ...

```
User Agent Supported Functionalities
  = Determine_Actual_User_Agent_Supported_Functionality();
  //Returns an identifier value, an CC/PP-RDF structure
  //pointer, or other.
User Agent Preferred Functionality
  = Determine_Actual_User_Agent_PreferreD_Functionality();
Server Identifier
  = Determine_Document_Server(Document Identifier);
  //Determine the wanted document server or proxy.
Client_Request
  = (<Client Identifier, Server Identifier, Document Identifier,
  User Agent, Supported Functionality, User Agent
  Preferred Functionality>);
send (Client_Request) to Server Identifier;
...
repeat forever {
 when receive Server_Request form Server Identifier do
  {
   (CI,SI,DI,P_S) = Server_Request;
   SP = TL_evaluator(P_S,CC_P);  {use case 2}
   Send (CI,SI,DI,SP) to SI;
  } //end when
```

```
 when receive Server_Reply form Server Identifier do
  {
   (CI,SI,D) = Server_Request;
   if D ≠ ∅ then use D;
  }//end when
 }//end repeat
 ...
```

2- Server side: The server code part is given as follows:

```
...
repeat forever {
when receive Client_Request form Client Identifier do
{
 (CI,SI,DI,US,UP)= Client_Request;
 USP = UP-(UP-US);
          // Determine the supported AND preferred modules
 P_S = TL_evaluator(US,D_P);  {use case 1}
 P_S= P_S ∪ TL_evaluator(US,T_set);  {use case 1}
 // Search for possible profiles, after applying existing
 // transformations methods that can satisfy the user agent
 // profile (US).
 if (P_S ≠ ∅) then
          P_S = TL_evaluator(P_S,USP);  {use case 2}
          // Apply further TL selections on the base of clients preferences
          P_S = TL_evaluator(P_S,SC_P);  {use case 2}
          // Apply further Server Constraints (SC_P). For example the
          // server can detect that the network bandwidth become very
          // low, and consequently we mustn't provide video services.
          Server_Request = <CI,SI,DI,P_S>;
          send (Server_Request) to CI
 else
          Server_Reply = <CI,SI,∅>;
          // Existing profiles doesn't reply to the demanded profile.
          send(Server_Reply)to CI;
 fi;
 }//end when
when receive Client_Reply form Client Identifier do
{
 (CI,SI,DI,SP)=Client_Request;
 if SP exists in D_P then
          send (CI,SI,Document(SP)) to CI
```

```
        // send the document having as profile SP
  else
        t = find_transformation(SP);
        D = t(DI);   //apply corresponding transformation;
        D_P = D_P ∪ SP;   //save document profile;
        save(D);//save document version;
        send(CI,SI,D) to CI;   //send the adapted document
  fi;
}//end when
}//end repeat
...
```

The algorithm that we propose here is suited to servers and clients personalizing. It doesn't impose any restrictions on the executing contexts. For example, the client can define freely its TL constraints evaluation, its personal preferences and additional constraints that can be updated any time without changing the basic descriptions. Network connectivity parameters such as bandwidth that may affect the services delivery can be included in the server constraint profile SC_P.

6 Conclusion

Nowadays, accessing to the Web content through a variety of devices is increasing rapidly. Powerful methods of transformation, adaptation and profiles negotiation strategies to deliver a customized service at the end user level, must be designed in an efficient manner.

In this paper, we have given the main steps and tools to use in the design of a negotiation-based multimedia service for heterogeneous environments. We have seen that, thanks to the related transformation technologies, XML content models are a key issue for authoring and storing adaptable multimedia documents on the servers. CC/PP and RDF mechanisms play a primary role in the description of client preferences and capabilities, which are used in the negotiation phase. SMIL modular approach provides necessary flexibility for both final presentations in different contexts, and the expression of the supported functionality either on the client or on the server. We have proposed a method of negotiation, based on profiling and document selecting. We have described a related protocol and shown how that strategy can respond efficiently to customized content delivery services to the end users.

Several open issues need further exploration. Among the future work that we intend to achieve is to build a detailed specification of how to use an end to end solution based on this approach: this includes an authoring model and a generic transformation framework.

References

1. M. Balabanovic, An Adaptive Web Page Recommendation Service, Proceeding of first International Conference en Autonomous Agents, 1997.
2. T. Berners-Lee, R. Fielding, L. Masinter, *RFC 2396: Uniform Resource Identifiers (URI): Generic Syntax*, IETF Request for Comments: ftp://ftp.isi.edu/in-notes/rfc2396.txt
3. D. C. A. Bulterman, *User-Centered Abstractions for Adaptive Hypermedia Presentations, CWI (Centrum voor Wiskunde en Informatica)*, NL-1090 GB Amesterdam, The Netherlands.
4. W3C, *Composite Capabilities/Preference Profiles (CC/PP): A user side framework for content negotiation*, W3C Note, http://www.w3.org.
5. W3C,*Composite Capabilities/Preference Profiles: Requirement and Architecture* , W3C Working Draft, http://www.w3.org.
6. M. Chen, D.D. Kandlur, and P.S. Yu, *Support for Fully Interactive Playout in a Disk-array-based Video Server*, In Proc. ACM Multimedia'94, San Fransisco, USA, October 1994.
7. H. Chen, A. Krishnamuthy, T.D.C. Little, and D. Venkatesch, *A Scalable Video-on-Demand Service for the Provision of VCR-like Functions*, In Proc. ICMCS'95, Washington DC, USA, May 1995.
8. 3GPP, *Technical Specification Group Services and System Aspects, Transparent end-to-end PSS, protocols and codecs (Release 4)*, 3GPP TS 26.234 v1.5.1, March 2001.
9. Graham Klyne, Franklin Reynolds, Chris Woodrow, and Hidetaka Ohto, *Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies*, http://www.w3.org/TR/CCPP-struct-vocab/, W3C Working Draft, 15 March 2001.
10. S. Hollfelder, A. Kraiss, and T.C. Rakow, *A Client-Controlled Adaptation Framework for Multimedia Database Systems*, In R. Steinmetz and L.C. Wolf, editors, Proc. IDMS'97, pp 397-409, Darmstadt, Germany, September 10-12, Springer 1997.
11. N. Layaïda, *Madeus: Système d'édition et de présentation de documents structurés multimédia*, PhD thesis, Joseph Fourier University - Grenoble I, June 1997, France.
12. Tayeb Lemlouma, *Routing in Mobile Ad Hoc networks*, Master Thesis, Computer Science Institute, USTHB University, Algiers, Algeria, October 2000.
13. F. Moser, A. Kraib, and W. Klas, *L/MRP: A Buffer Management Strat-*

egy for Interactive Continuous Data Flows in a Multimedia DBMS, In Proc. VLDB 1995, USA, 1995, Morgan Kaufmann.
14. Ora Lassila, Ralph Swick, *Resource Description Framework (RDF) Model and Syntax Specification*, W3C Recommendation: http://www.w3.org/TR/1999/REC-rdf-syntax.
15. S. V. Raghavan, Satish K. Tripathi, *Networked Multimedia Systems: Concepts, architecture and design*, Prentice-Hall, 07458 New Jersey,1998.
16. L. Rutledge, L. Hardman and J. V. Ossenbruggen, *The use of SMIL: Multimedia Research Currently Applied on a Global Scale, CWI (Centrum voor Wiskunde en Informatica)*, NL-1090 GB Amesterdam, The Netherlands.
17. G. Salton and M. J. McGill, *An Introduction to Modern Information Retrieval*, McGraw-Hill 1983.
18. Kenichi Kubota, Aaron Cohen, *SMIL Basic Language Profile, 22 June 2000*,
http://www.w3.org/TR/2000/WD-smil-boston-20000622/smil-basic-profile.html.
19. N. Layaïda, J. V. Ossenburggen, *SMIL 2.0 Language Profile, 01 March 2001*,
http://www.w3.org/TR/smil20/smil20-profile.html.
20. W3C, *Synchronized Multimedia Integration Language (SMIL 2.0) Specification*, W3C Candidate Recommendation, 12 March 2001, http://www.w3.org/AudioVideo/ Group/2001/CR-smil20-20010312/.
21. Susanne Boll, Wolfgang Klas and Jochen Wandel, *A Cross-Media Adaptation Strategy for Multimedia Presentations*, DBIS, Computer Science Department, University of Ulm, Germany, ACM 1999.
22. W3C HTML working group, *XHTML: The Extensible HyperText Markup Language, A Reformulation of HTML 4 in XML 1.0*, http://www.w3.org/TR/xhtml/
23. W3C, *Extensible Markup Language (XML) 1.0, W3C Recommendation*, 10 February 1998, http://www.w3.org/TR/1998/REC-xml-19980210.
24. W3C Working Group, *http://www.w3.org/Style/XSL*.